

Data Communications and Networks

Second Edition

About the Authors



Achyut Godbole is currently the Managing Director of Softexcel Consultancy Services, Mumbai. His professional career spans 32 years, and during this time, he has served in world-renowned software companies in India, UK and USA. He has contributed to the multifold growth of companies such as Patni, Syntel, L&T Infotech, Apar, Disha, etc. He did his BTech from IIT Bombay in Chemical Engineering, and henceforth, worked for the welfare of Adivasi tribes in Maharashtra, for a year.

Mr Godbole has authored best-selling textbooks from Tata McGraw-Hill such as *Operating Systems, Data Communications and Networking, and Web Technologies*, including international editions and Chinese translations. In addition, he has authored several very-highly rated books on various subjects (such as computers, management, economics, etc.) in Marathi and has written several popular columns in Marathi newspapers/magazines on science, literature, medicine, and technology. He has conducted numerous programmes on television pertaining to technology, science, and economics.

He has won several awards, including an award from the Prime Minister of India, 'Udyog Ratna', 'Distinguished Alumnus' from IIT, 'Kumar Gandharva' award from Pandit Bhimsen Joshi, 'Navaratna' from Sahyadri TV channel, the 'Indradhanu Puraskar', and 'Parnerkar Puraskar' for his contributions to the field of economics. Besides, he was ranked 16th in merit in Maharashtra State Board Examinations. A brilliant student, he has won several prizes for distinguished performance in Mathematics as well. He also runs a school for autistic children.

He has a website (www.achyutgodbole.com) and can be reached at achyut.godbole@gmail.com.

Atul Kahate is working with Oracle Financial Services Software Limited as Senior Consultant. He has been associated with this organisation for over a decade. He has 16 years of experience in Information Technology in India and abroad in various capacities. Previously, he has worked with Syntel, L&T Infotech, American Express and Deutsche Bank. He has a Bachelor of Science degree in Statistics and a Master of Business Administration in Computer Systems.



Mr Kahate has authored 28 highly acclaimed books on Technology, Cricket, and History published by Tata McGraw Hill, and other reputed publications. Some of these titles include *Web Technologies—TCP/IP to Internet Application Architectures, Cryptography and Network Security, Fundamentals of Computers, Information Technology and Numerical Methods, Introduction to Database Management Systems, Object Oriented Analysis and Design, and Schaum's Series Outlines—Programming in C++, XML and Related Technologies*. Two of these have been published as international editions by Tata McGraw-Hill and have been translated into Chinese. Several of his books are being used as course textbooks or sources of reference in a number of universities/colleges/IT companies all over the world. He has authored two books on cricket, and has written over 3000 articles on IT and cricket in leading Marathi newspapers/magazines/journals in India and abroad.

He has deep interest in history, science, economics, music, cricket, and teaching, besides technology. He has conducted several training programs in a number of educational institutions and IT organizations on a wide range of technologies. He has done a series of programmes for many TV channels for explaining complex technologies to a common viewer in a simplified manner.

Mr Kahate has won several awards, both in India and abroad, including the Computer Society of India (CSI) award for IT education and literacy, the noted 'Yuvonmesh Puraskar' from Indradhanu–Maharashtra Times, and the 'IT Excellence Award' from Indira Group of Institutes.

He has a website (www.atulkahate.com) and can be reached at akahate@gmail.com.

Data Communications and Networks

Second Edition

ACHYUT S GODBOLE

*Managing Director
SoftExcel Services Limited, Mumbai*

ATUL KAHATE

*Senior Consultant
Oracle Financial Services Software Limited, Pune*



Tata McGraw Hill Education Private Limited

NEW DELHI

McGraw-Hill Offices

New Delhi New York St Louis San Francisco Auckland Bogotá Caracas
Kuala Lumpur Lisbon London Madrid Mexico City Milan Montreal
San Juan Santiago Singapore Sydney Tokyo Toronto



Tata McGraw-Hill

Published by the Tata McGraw Hill Education Private Limited,
7 West Patel Nagar, New Delhi 110 008.

Data Communications and Networks, 2e

Copyright © 2011, 2002, by Tata McGraw Hill Education Private Limited.

No part of this publication may be reproduced or distributed in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise or stored in a database or retrieval system without the prior written permission of the publishers. The program listing (if any) may be entered, stored and executed in a computer system, but they may not be reproduced for publication.

This edition can be exported from India only by the publishers,
Tata McGraw Hill Education Private Limited.

ISBN (13): 978-0-07-107770-5

ISBN (10): 0-07-107770-7

Vice President and Managing Director—McGraw-Hill Education, Asia Pacific Region: *Ajay Shukla*

Head—Higher Education Publishing and Marketing: *Vibha Mahajan*

Publishing Manager—SEM & Tech Ed.: *Shalini Jha*

Development Editor: *Surbhi Suman*

Sr Copy Editor: *Nimisha Kapoor*

Sr Production Manager: *Satinder S Baveja*

Proof Reader: *Yukti Sharma*

Marketing Manager—SEM & Tech Ed.: *Biju Ganesan*

Sr Product Specialist—SEM & Tech Ed.: *John Mathews*

General Manager—Production: *Rajender P Ghansela*

Asst General Manager—Production: *B L Dogra*

Information contained in this work has been obtained by Tata McGraw-Hill, from sources believed to be reliable. However, neither Tata McGraw-Hill nor its authors guarantee the accuracy or completeness of any information published herein, and neither Tata McGraw-Hill nor its authors shall be responsible for any errors, omissions, or damages arising out of use of this information. This work is published with the understanding that Tata McGraw-Hill and its authors are supplying information but are not attempting to render engineering or other professional services. If such services are required, the assistance of an appropriate professional should be sought.

Typeset at Bukprint India, B-180A, Guru Nanak Pura, Laxmi Nagar, Delhi 110 092 and printed at
Avon Printers, Plot No. 16, Main Loni Road, Jawahar Nagar Industrial Area, Shahdara, Delhi 110 094

Cover Printer: SDR Printers

RQXQCRCCRBBCB

The McGraw-Hill Companies

Dedicated to

Late Shriram Pujari Sir

for teaching us to love everything that is beautiful in life

Contents



<i>Preface</i>	xv
1. Introduction to Data Communications and Networking	1
1.0 Introduction	1
1.1 Fundamental Concepts	1
1.2 Data Communications	3
1.3 Protocols	4
1.4 Standards	4
1.5 Standards Organizations	5
1.6 Signal Propagation	6
1.7 Analog and Digital Signals	8
1.8 Bandwidth of a Signal and a Medium	9
1.9 Fourier Analysis and the Concept of Bandwidth of a Signal	13
1.10 Data Transmission Rate and Bandwidth	18
<i>Summary</i>	20
<i>Key Terms and Concepts</i>	21
<i>Questions</i>	21
2. Analog and Digital Transmission Methods	24
2.0 Introduction	24
2.1 Analog Signal, Analog Transmission	24
2.2 Digital Signal, Digital Transmission	24
2.3 Digital Signal, Analog Transmission	26
2.4 Baud Rate and Bits Per Second	30
2.5 Analog Signal, Digital (Storage and) Transmission	32
2.6 Nyquist Theorem	34
<i>Summary</i>	36
<i>Key Terms and Concepts</i>	37
<i>Questions</i>	37
3. Modes of Data Transmission and Multiplexing	40
3.0 Introduction	40

3.1	Parallel and Serial Communication	40
3.2	Asynchronous, Synchronous and Isochronous Communication	42
3.3	Simplex, Half-duplex and Full-duplex Communication	46
3.4	Multiplexing and Demultiplexing	50
3.5	Types of Multiplexing	50
3.6	FDM versus TDM	57
	<i>Summary</i>	57
	<i>Key Terms and Concepts</i>	59
	<i>Questions</i>	59
4.	Transmission Errors: Detection and Correction	62
4.0	Introduction	62
4.1	Error Classification	62
4.2	Types of Errors	63
4.3	Error Detection	63
	<i>Summary</i>	75
	<i>Key Terms and Concepts</i>	76
	<i>Questions</i>	76
5.	Data Compression and Encryption	78
5.0	Introduction	78
5.1	Simple Coding Scheme	78
5.2	Based on the Context of Symbols	79
5.3	Based on the Relative Frequencies of Symbols	80
5.4	Information Security	85
5.5	Cryptography	87
5.6	Symmetric and Asymmetric Key Encryption	89
5.7	Digital Certificates	93
5.8	Digital Signatures	94
5.9	Secure Socket Layer (SSL)/Transport Layer Security (TLS)	95
5.10	Firewalls	97
5.11	Email Security	99
	<i>Summary</i>	102
	<i>Key Terms and Concepts</i>	103
	<i>Questions</i>	103
6.	Transmission Media	106
6.0	Introduction	106
6.1	Guided Media	106
6.2	Unguided Media	111
6.3	Shannon Capacity	120
	<i>Summary</i>	121
	<i>Key Terms and Concepts</i>	121
	<i>Questions</i>	122

7. Network Topologies, Switching and Routing Algorithms	124
7.0 Introduction	124
7.1 Mesh Topology	124
7.2 Star Topology	125
7.3 Tree Topology	125
7.4 Ring Topology	127
7.5 Bus Topology	128
7.6 Hybrid Topology	128
7.7 Basics of Switching	130
7.8 Router and Routing	136
7.9 Routing Algorithms	139
<i>Summary</i>	121
<i>Key Terms and Concepts</i>	149
<i>Questions</i>	122
8. Networking Protocols and OSI Model	153
8.0 Introduction	153
8.1 Protocols in Computer Communications	155
8.2 The OSI Model	159
8.3 OSI Layer Functions	163
8.4 Queuing Theory and M/M/1 Queues	172
<i>Summary</i>	172
<i>Key Terms and Concepts</i>	173
<i>Questions</i>	174
9. Local Area Networks (LAN), Metropolitan Area Networks (MAN) and Wide Area Networks (WAN)	176
9.0 Introduction	176
9.1 Local Area Networks (LAN)	177
9.2 Ethernet	177
9.3 Virtual LAN (VLAN)	184
9.4 Fast and Gigabit Ethernet	186
9.5 Token Ring	188
9.6 Fiber Distributed Data Interface (FDDI)	192
9.7 Comparison of Ethernet, Token Ring and FDDI	195
9.8 Metropolitan Area Network (MAN)	195
9.9 Distributed Queue Dual Bus (DQDB)	195
9.10 Switched Multimegabit Data Services (SMDS)	198
9.11 Wide Area Network (WAN)	199
9.12 WAN Architecture	200
9.13 WAN Transmission Mechanism	201
9.14 WAN Addressing	202
9.15 Packet Forwarding	203
9.16 Next-hop Tables and Routing	205
9.17 Pure and Slotted ALOHA	206

<i>Summary</i>	207
<i>Key Terms and Concepts</i>	208
<i>Questions</i>	209

10. Medium Access Sublayer and ISDN **212**

10.0	Introduction	212
10.1	Static and Dynamic Channel Allocation	212
10.2	Medium Access Control (MAC) Sublayer	214
10.3	MAC in LAN and WAN	215
10.4	Classification and Study of MAC Sublayer Protocols/Collisions	216
10.5	ISDN and Its Background	217
10.6	ISDN Architecture	219
10.7	ISDN Interfaces	221
10.8	Functional Grouping	223
10.9	Reference Points	225
10.10	ISDN Protocol Architecture	225
10.11	Narrowband-ISDN (N-ISDN) and Broadband ISDN (B-ISDN)	231
	<i>Summary</i>	231
	<i>Key Terms and Concepts</i>	232
	<i>Questions</i>	232

11. X.25 Protocol **235**

11.0	Introduction	235
11.1	Understanding How X.25 Works	235
11.2	Characteristics of X.25	238
11.3	Packet Format	240
11.4	X.25 Operation	243
11.5	CCITT X.21	245
	<i>Summary</i>	246
	<i>Key Terms and Concepts</i>	246
	<i>Questions</i>	246

12. Frame Relay and Congestion Control **248**

12.0	Introduction	248
12.1	The Need for Frame Relay	248
12.2	How Frame Relay Works	253
12.3	Frame Relay Frame Format	256
12.4	Congestion Control	257
12.5	Congestion Control Algorithms	258
12.6	Traffic Control	260
12.7	Frame Relay Assembler/Disassembler (FRAD)	261
12.8	Other Features	261
	<i>Summary</i>	261
	<i>Key Terms and Concepts</i>	262
	<i>Questions</i>	262

13. Asynchronous Transfer Mode (ATM)	265
13.0 Introduction	265
13.1 Overview of ATM	266
13.2 Packet Size	268
13.3 Virtual Circuits in ATM	270
13.4 ATM Cells	272
13.5 Switching	273
13.6 ATM Layers	276
13.7 Miscellaneous Topics	281
<i>Summary</i>	283
<i>Key Terms and Concepts</i>	284
<i>Questions</i>	284
14. Wireless Communication	287
14.0 Overview of Wireless Networks	287
14.1 IEEE Standards for LAN, MAN, and WAN – 802.1, 802.2, 802.3, 802.4, 802.5, 802.11	288
14.2 Infrared Communication	290
14.3 Bluetooth	291
14.4 802.11 Wireless LAN	298
<i>Summary</i>	305
<i>Key Terms and Concepts</i>	305
<i>Questions</i>	306
15. Internetworking Concepts, Devices, Internet Basics, History and Architecture	308
15.0 Introduction	308
15.1 Why Internetworking?	309
15.2 Problems in Internetworking	309
15.3 Dealing with Incompatibility Issues	310
15.4 A Virtual Network	313
15.5 Internetworking Devices	314
15.6 Repeaters	315
15.7 Bridges	316
15.8 Routers	321
15.9 Gateways	324
15.10 A Brief History of the Internet	325
15.11 Growth of the Internet	327
15.12 Internet Topology	329
15.13 Internal Architecture of an ISP	331
<i>Summary</i>	335
<i>Key Terms and Concepts</i>	336
<i>Questions</i>	337
16. Ways of Accessing the Internet	340
16.0 Introduction	340

- 16.1 Dial-up Access for an Individual User 340
- 16.2 Leased Lines 343
- 16.3 Digital Subscriber Line (DSL) 343
- 16.4 Cable Modems 348
- 16.5 DTE-DCE Interface 350
- 16.6 EIA RS-232 and EIA RS-449 Interface 351
- 16.7 SONET/SDH – Synchronous Transport Signals 352
- 16.8 SONET Layers—Applications 353
 - Summary* 354
 - Key Terms and Concepts* 355
 - Questions* 355

17. TCP/IP—Part 1: Introduction to TCP/IP, IP, ARP, RARP and ICMP 357

- 17.0 Introduction 357
- 17.1 TCP/IP Basics 358
- 17.2 Why IP Addresses? 360
- 17.3 Logical Addresses 363
- 17.4 TCP/IP – An Example 364
- 17.5 The Concept of IP Address and IP Datagram/Package 371
- 17.6 Address Resolution Protocol (ARP) 380
- 17.7 Reverse Address Resolution Protocol (RARP) 383
- 17.8 Internet Control Message Protocol (ICMP) 384
- 17.9 Datagram Fragmentation and Reassembly 388
- 17.10 Comparison of OSI and TCP/IP Protocol Suites 393
 - Summary* 393
 - Key Terms and Concepts* 394
 - Questions* 395

18. TCP/IP—Part 2: TCP and UDP 399

- 18.0 Introduction 399
- 18.1 TCP Basics 399
- 18.2 Features of TCP 400
- 18.3 Relationship between TCP and IP 403
- 18.4 Ports and Sockets 404
- 18.5 Connections – Passive Open and Active Open 408
- 18.6 TCP Connections 409
- 18.7 What makes TCP reliable? 410
- 18.8 TCP Packet Format 411
- 18.9 Persistent TCP Connections 412
- 18.10 User Datagram Protocol (UDP) 414
- 18.11 UDP Packet 414
- 18.12 Differences between UDP and TCP 415
 - Summary* 417
 - Key Terms and Concepts* 418
 - Questions* 418

19. TCP/IP—Part 3: DNS, Email, FTP and TFTP	421
19.0 Introduction	421
19.1 Domain Name System (DNS)	421
19.2 Electronic Mail (Email)	429
19.3 File Transfer Protocol (FTP)	445
19.4 Trivial File Transfer Protocol (TFTP)	452
<i>Summary</i>	453
<i>Key Terms and Concepts</i>	454
<i>Questions</i>	454
20. TCP/IP—Part 4: WWW, HTTP and TELNET	457
20.0 Introduction	457
20.1 The Basics of WWW and Browsing	458
20.2 Locating Information on the Internet	464
20.3 Hyper Text Markup Language (HTML)	465
20.4 Web Browser Architecture	470
20.5 Web Pages and Multimedia	472
20.6 Remote Login (TELNET)	474
20.7 Static, Dynamic, and Active Web Pages	477
<i>Summary</i>	480
<i>Key Terms and Concepts</i>	482
<i>Questions</i>	482
21. Multimedia Communications	485
21.0 Introduction	485
21.1 Basics of Multimedia	485
21.2 Multimedia Applications	486
21.3 Multimedia Protocols	492
21.4 Session Initiation Protocol (SIP)	496
<i>Summary</i>	497
<i>Key Terms and Concepts</i>	497
<i>Questions</i>	497
Appendix A: Internet Protocol Version 6 (IPv6)	499
Appendix B: Hardware for Error Detection	504
Appendix C: Network Management and Monitoring	507
Appendix D: Answers to True or False	513
Appendix E: Answers to Multiple-Choice Questions	517
Glossary	521
References	535
Index	536

Preface



We are very happy to present a second edition of the book Data Communications and Networks. This book is designed to be a primary level comprehensive text for students of this subject. It can also be used as a good source of reference by IT professionals who want to gain deeper insights into areas pertaining to computer networks and data communications.

Planned meticulously, this book tries to explain data communication and networking concepts in a manner suitable for novice readers. Identifying the need for a basic text amid several complex books available today, a conscious effort has been made to keep the discussions in this text very simple. This edition adheres to lucid explanations with a good mix of theoretical concepts and practical examples.

New to this edition

In this second edition, we have replaced some obsolete topics with new and more relevant themes after careful perusal of various syllabi. Content of chapters has been enhanced holistically to include latest information from this field. Key features of this edition are listed below.

- Chapter on **Wireless Communication** including latest topics such as IEEE Standards, Bluetooth, Wireless LANs, and Cellular Telephones
- Inclusion of topics—Medium Access Sublayer, EIA RS-232C, EIA RS-449, Comparison of OSI and TCP/IP models, Firewalls, Web Security, Subnetting, DTE-DTC Interface, Queuing Theory, Checksum, Hamming Code
- Expanded coverage of important topics—Congestion Control, Ethernet, Multiplexing, X.25 Protocol, ALOHA, Asynchronous Transfer Mode, ISDN
- Updated coverage of various protocols—OSI, ATM, Frame Relay, TCP/IP, ISDN, X.25
- Enhanced chapter-end exercises comprising 600 objective questions (true/false and multiple-choice type) and 275 review questions

Book Overview

The book is organized in a progressive manner that first introduces the reader to the basic concepts in data communication and networks, then discusses key areas such as transmission media, network types and topologies, error handling, security, and data compression towards the middle and finally elucidates some advanced concepts. A detailed discussion of the Internet and TCP/IP protocol suite has been interspersed. The chapter-wise organization of the book is as follows.

Chapter 1 introduces the fundamental concepts in data communications and networks. It also covers the idea of standards and explains key concepts in signal handling, signal types, signal propagation, etc. **Chapter 2** explains the idea of analog and digital signals and their transmission in greater detail. It covers the ideas of sampling, quantizing, and the limits on maximum data rates. **Chapter 3** covers topics related to various modes of data transmission, such as serial and parallel; synchronous and asynchronous; simplex, half-duplex and full-duplex, etc. It also explains the key concepts in multiplexing of signals. **Chapter 4** deals with the idea of transmission errors. It explains the various approaches used and the algorithms employed for this purpose. **Chapter 5** discusses data compression and encryption. It covers data compression techniques that help reduce transmission requirements. The chapter further elaborates all fundamental principles in transmission security such as encryption and decryption, message digests, digital signatures, firewalls, etc.

Chapter 6 explains the use of various media for data transmission. These include copper wires, coaxial cables, and optical fibers. **Chapter 7** focuses on various network topologies, viz., ring, bus, star, etc., and covers very important ideas in switching and routing of packets. It discusses circuit and packet switching, the idea of a router and its working. **Chapter 8** elucidates the theory behind the OSI protocol stack. Each layer is described in detail with examples. **Chapter 9** describes the working of three types of networks, viz., LAN, MAN, and WAN. It explains the various protocols used by all these three types of networks. The chapter also deals with the issue of why these different types of networks are needed, and how they are technically different from each other. **Chapter 10** covers the digital telephony protocol of ISDN along with the basics of the Medium Access Control (MAC) Sublayer.

Chapter 11 aims at explaining WAN communication using the X.25 protocol. Although increasingly new technologies are replacing X.25, it is important to know how X.25 functions so as to appreciate why we are moving away from it. **Chapter 12** deals with the improved version of X.25, i.e., Frame Relay. Apart from handling all the key topics in Frame Relay, this second edition also covers congestion control algorithms such as leaky bucket and token bucket. **Chapter 13** deals with the synchronous protocol of Asynchronous Transfer Mode (ATM). **Chapter 14** is a new chapter that includes all key wireless/mobile technologies such as infrared, Bluetooth, and Wi-Fi. **Chapter 15** introduces the ideas of internetworking. This forms the basis for understanding how the Internet functions.

Chapter 16 details the various ways in which home/corporate subscribers can access the Internet. This includes dial-up connection, cable Internet, ADSL broadband, etc. **Chapter 17** deals with the Internet Protocol (IP) and various areas related to IP. It explains how an IP datagram looks like and how it moves across the Internet. **Chapter 18** explains the transport layer protocols in the TCP/IP protocol suite. Hence, it covers the TCP and UDP protocols. The chapter elaborates the differences between the two protocols and explains the rationale behind the use of these protocols in a given situation. **Chapter 19** covers the various application layer protocols such as FTP, DNS, Email, etc. **Chapter 20** deals with the remaining application layer protocols, such as WWW, HTTP, and TELNET. **Chapter 21** explains the working of multimedia communication.

Appendices **A** and **B** discuss Internet Protocol Version (IPv6) and Hardware for Error Detection, respectively. **Appendix C** has been updated to present the latest in Network Management and Monitoring. New **Appendices D** and **E** comprising answers have been added as this edition.

Every chapter starts with an introduction for the reader to have an overview of the chapter content. At the end of every chapter, a summary is presented to recap the key points along with a

list of important terms for easy reference. Exhaustive chapter-end exercises comprising true/false, multiple-choice and detailed questions have been included.

Online Resources

The Web supplement of the book contains the following additional material for instructors and students.

For Instructors

- PowerPoint Presentations
- Answers to selected Review Questions

For Students

- Self-Test Quizzes
- Extra Reading – Chapter on Information Coding and Appendix on Fundamental Mathematics for Problem Solving from the first edition

All of these online resources can be accessed at <http://www.mhhe.com/dcn2>.

Acknowledgements

We are grateful to the following reviewers for their valuable comments on the manuscript:

Narendra Kohli

Harcourt Butler Technological Institute, Kanpur, Uttar Pradesh

G M Rather

National Institute of Technology Srinagar, Jammu & Kashmir

Narottam Chand Kaushal

National Institute of Technology Hamirpur, Himachal Pradesh

Chiranjeev Kumar

Indian School of Mines, Dhanbad, Jharkhand

Mahua Banerjee

Xavier Institute of Social Service, Ranchi, Jharkhand

Deepanwita Daptary

National Institute of Technology, Durgapur, West Bengal

Ashish Tiwari

Visvesvaraya National Institute of Technology, Nagpur, Maharashtra

Bhakti Raul

KJ Somaiya College of Engineering, Mumbai

xviii *Preface*

Tomy Thomas

Mohandas College of Engineering and Technology, Trivandrum, Kerala

P Krishnakumari

Sri Ramakrishna College of Arts and Science for Women, Coimbatore, Tamil Nadu

M Rajasekar

Shree Venkateshwara Hi-tech Engineering College, Erode district, Coimbatore, Tamil Nadu

P Seetharamaiah

College of Engineering, Andhra University, Vishakhapatnam, Andhra Pradesh

R Sridevi

College of Engineering, Jawaharlal Nehru Technological University, Hyderabad, Andhra Pradesh

We would like to thank all our family members, friends, and colleagues for their tremendous help in terms of constant encouragement, support, and patience. Special thanks to Shobha Godbole and Anita Kahate for their understanding and patience. Nihar Godbole and Jui and Harsh Kahate deserve a note of thanks, too!

We are grateful to the staff at Tata McGraw-Hill, most notably Vibha Mahajan, Shalini Jha, Surbhi Suman, Nimisha Kapoor and Yukti Sharma for their wonderful help in completing the book on time.

We would be very happy to receive feedback on this book. You can email us at *achyut.godbole@gmail.com* and *akahate@gmail.com*.

Achyut S Godbole
Atul Kahate

Publisher's Note

Remember to write to us. We look forward to receiving your feedback, comments and ideas to enhance the quality of this book. You can reach us at *tmh.csefeedback@gmail.com*. Please mention the title and author's name as the subject. In case you spot piracy of this book, please do let us know.

Introduction to Data Communications and Networking

1

1.0 INTRODUCTION

This chapter introduces the basic concepts of data communications. Before we understand how computer networks and inter-networks work, it is essential to know how data can be transmitted from a source to a destination in the first place. This forms the basis for all data communications. The principle of signal propagation is used for this purpose.

We shall first have an overview of data communications, some important concepts and the standards related to data communications along with the organizations who govern these standards.

We shall start our technical discussion with an understanding of how we can transmit data in the form of signals from one point to another. Using this as the base, we shall then consider how a signal can be analyzed for understanding data communications. We shall introduce several important terms as we go along, that are very commonly used in the discussions of data communications.

1.1 FUNDAMENTAL CONCEPTS

1.1.1 Basic Idea

Communication can be defined as exchange of information between two humans. Data communications can be defined the exchange of information between two computers. In its simplest form, the data communication process can be shown in Fig. 1.1. The figure shows one computer (sender) sending a message to another computer (receiver) over a wire (called **transmission medium**).

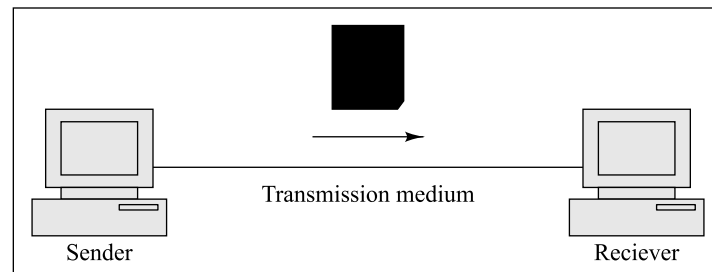


Fig. 1.1 *Data communication*

Of course, this is an oversimplified view. Real-life data communication process involves many hardware devices and software techniques. This makes data communication an extremely complex process. Real-life data communication process (still far more simplified) is shown in Fig. 1.2.

1.1.2 Real-life Data Communications

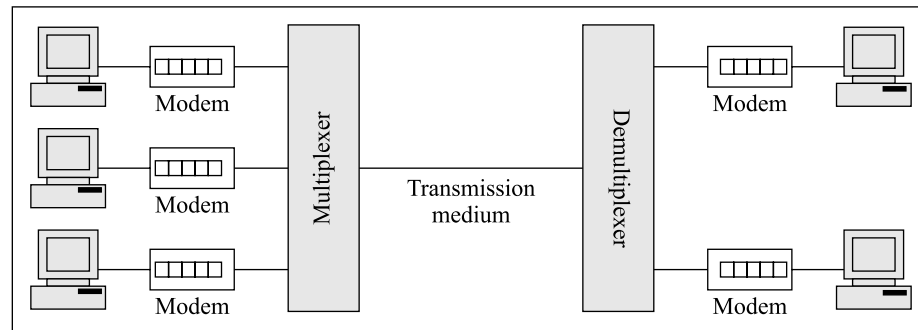


Fig. 1.2 Real-life data communication systems

Figure 1.2 shows a real-life data communication system. It contains various components (shown) as well as other aspects (hidden) as discussed below:

- **Modem** – A modem is connected to every computer that is involved in data communications (either for sending or receiving data). Why is a modem required? To understand this we must understand the concept of **signals** and **signal propagation**, as discussed in this chapter. This is followed by a discussion of analog and digital transmission techniques in Chapter 2, which are closely related to the way modems work.
- **Multiplexer and Demultiplexer** – What is a multiplexer? What is a demultiplexer? The figure shows the multiplexer at one end and the demultiplexer at the other end. These components and their importance are the subjects of discussion in Chapter 3.
- The subject of data communications is closely related to the understanding of errors that can occur during transmission. How to detect and correct these errors is the topic of discussion in Chapter 4.
- Data can be compressed, so that its size is reduced, and it can be sent faster. Modern data communications demand a lot of security mechanisms to be in place. These are the two topics of discussion in Chapter 5.
- **Transmission medium** – Transmission medium, or wire, is the means of transferring data from the sender to the receiver. Modern data communications can also be wireless. Wired and wireless transmission media are discussed in Chapter 6.
- There are various ways in which computers can be connected to each other to form a **computer network**. These different ways are discussed in Chapter 7. This chapter also deals with the concepts of **routing**, a process that decides how to forward data items from one computer to another on a network.
- Computers cannot communicate with each other arbitrarily, like humans. For instance, when we dial someone's phone number, we first greet the person who picks up the phone, inform who is calling, and then proceed. Computers also need to follow certain **protocols** during data communications. This concept is discussed in Chapter 8.

- Chapter 9 discusses the three major types of computer networks depending on the available physical area, number of computers to be connected, and the purpose of networking. These types are **Local Area Networks (LAN)**, **Metropolitan Area Networks (MAN)** and **Wide Area Networks (WAN)**.
- Chapters 10 to 13 discuss some of the most popular data communications protocols, namely, **Integrated Services Digital Networks (ISDN)**, **X.25**, **Frame Relay** and **Asynchronous Transfer Mode (ATM)**.
- Wireless technologies such as **IEEE 802.11 (WiFi)**, **Bluetooth**, **WiMAX**, etc., are becoming immensely popular as the world is increasingly moving towards wireless access to information. Chapter 14 discusses these technologies.
- Chapter 15 introduces the concept of inter-networking, and discusses the basics of the worldwide network of computers and computer networks, i.e., the **Internet**.
- The Internet can be accessed in a number of ways. We discuss all these possibilities in Chapter 16.
- These days, the **Transmission Control Protocol/Internet Protocol (TCP/IP)** is perhaps the most widely discussed data communications protocol suite. We discuss the various protocols in the TCP/IP protocol suite in Chapters 17 through 20.
- Chapter 21 describes multimedia communications.

1.2 DATA COMMUNICATIONS

Data communications has become an extremely important aspect of the modern world. It has been a subject of great interest for many decades. However, the possibilities of exchanging business-critical documents and information, conducting mission-critical transactions across geographical boundaries, sharing personal information with friends, etc., have made it very important to understand how to exchange data and how it works internally.

In the simplest form, when we talk of data communications, it involves exchange of data between two computers. Computers work with the binary language of zeros and ones. Therefore, one computer generates a stream of zeroes and ones, and sends it to another computer, to which it is connected in some fashion. The connection can be either a simple wire, or it can be through wireless media as well. Moreover, these two computers need not, necessarily, be close to each other—they can be in different rooms, streets, cities, states, countries or even continents. The magic of data communications technology enables this exchange of zeroes and ones from one computer to another.

For enabling data communications, a combination of hardware and software is essential. In any data communications system, the following three characteristics are desired:

1. **Correct delivery** – When a sender transmits data for an intended recipient, the data must reach only the intended recipient and not someone else.
2. **Accurate delivery** – Data sent by the sender must be received by the receiver in the same form as the one in which it was sent. There must not be any sort of alterations in it while in transit.
3. **Timely delivery** – Data must travel from the sender to the receiver in a finite amount of time. The term *finite* is quite vague, and would depend on the reasons why data communication is taking place.

Two key aspects of data communication systems need a good amount of understanding. These are, namely, **transmission medium** and data communication **protocols**.

Transmission medium is the physical path over which data travels from the sender to the receiver. The transmission medium can be a twisted pair of copper wires, coaxial cable, optical fiber or wireless media such as radio waves. We shall discuss all of these in detail later.

1.3 PROTOCOLS

A protocol is a set of rules and conventions that govern data communications. The sender and the receiver, the two key parties in data communication, must agree on a common set of rules, i.e., protocols before they can communicate with each other. Just as a person speaking only French cannot communicate with another person who understands only English, two devices that are connected to each other need not necessarily be able to communicate with each other unless they agree on a set of data communication protocols. There are many protocols for data communication, some of which are more popular than others. We shall have a quick overview of protocols now and study these in detail later.

Two devices wishing to communicate with each other cannot just begin data transmission arbitrarily. That is, one device cannot simply start sending bit streams to the other. The two devices must agree on a set of rules before this transmission can begin. Otherwise, how would the receiver know what the sender has sent? Conversely, how would the sender know if the receiver has correctly received the data that it had sent?

A protocol defines the following (in addition to a few other things):

1. **Syntax** (What is to be communicated?) – Syntax defines the structure or format of data. This means that the order in which it is to be sent is decided. For instance, a protocol could define that the first 16 bits of any data transmission must always contain the receiver's address.
2. **Semantics** (How it is to be communicated?) – Semantics define the interpretation of the data that is being sent. For example, the semantics could define that if the last two bits of the receiver's address field contain a 00, it means that the sender and the receiver are on the same network.
3. **Timing** (When it should be communicated?) – This refers to an agreement between the sender and the receiver about data transmission rates and duration. For instance, a protocol could demand that the sender must send 1000 bytes and then wait for an acknowledgement from the receiver before sending any more data.

1.4 STANDARDS

Standards are necessary in every walk of life. For instance, when you want to replace a light bulb in your home because it has been damaged, you expect the new bulb to fit in the holder straightaway and work like the old bulb. What is the use if the bulb does not fit in the holder or if it fits in the holder but does not illuminate because it requires a different voltage level? Consequently, everything that we use in our daily life has some common features, some standards that every manufacturer must abide by. In the absence of standards, every manufacturer can theoretically manufacture a set of goods or services that are incompatible with other manufacturers.

To avoid such anomalies, a set of standards is established, which governs the rules that manufacturers must obey. In exactly the same fashion, standards for data communications have been set. These standards ensure elimination of incompatibility issues, which is highly desirable in all data communications.

Data communication standards can be classified into the following two categories: **de facto** (which means *by convention*) and **de jure** (which means *by regulation*).

De facto standards can be further divided into *proprietary* and *non-proprietary* standards. Proprietary standards are invented and owned by an organization. These standards gain popularity post the owner's successful usage. This is because once the products of the organization using these standards are popular, the standards automatically gain popularity. Another name for *proprietary* standards is *closed*, because they close off communication with devices/systems of other vendors. *Non-proprietary* or *open* standards are those that are developed by an organization/committee/group, which become popular and vendors start supporting them. They are *open* because anybody adhering to those automatically gain access to all others following those standards.

De jure standards are the standards that have been legislated by an official body. These are usually led by governments or government-appointed agencies.

1.5 STANDARDS ORGANIZATIONS

Various standards organizations for data communications exist. Broadly, they can be classified into the following three categories: **Standards creation committees**, **Forums** and **Regulatory agencies**.

1.5.1 Standards Creation Committees

There are a number of organizations serving as standards creation committees. Let us name a few key players in this area.

- **The International Standards Organization (ISO)** is a well-known multinational standards body. Most members of ISO are representatives of their respective governments. Created in 1947, the ISO is a non-profitable standards creation organization. Members from over 80 developed nations actively represent the ISO. The **Open Systems Interconnection (OSI)** model as a networking protocol is a big contribution of ISO to the data communications world.
- **The International Telecommunications Union-Telecommunications Standards Sector (ITU-T)** was earlier known as the Consultative Committee for International Telegraphy and Telephony (CCITT). The name was changed on 1 March 1993 to ITU-T. This committee was formed by the United Nations in response to the demands from some nations, who were developing their own national standards for data communications in the early 1970s, leading to issues of incompatibility with each other. The V-series standards for use in modems (e.g., V.32), the X-series standards for public digital networks (e.g., X.25), email (e.g., X.400), directory services (e.g., X.500) and the Integrated Digital Services Network (ISDN) are some of the major contributions of ITU-T to data communications.
- The **American National Standards Institute (ANSI)** is a private non-profit organization that does not have any direct ties with the US federal government. However, generally, all ANSI projects are undertaken for the social benefit of US citizens. Professional groups, industry representatives, government, regulatory bodies, and consumer groups represent ANSI. ANSI is an affiliate of the ITU-T.
- The **Institute of Electrical and Electronics Engineers (IEEE)** is the biggest professional engineering body in the world. As the name suggests, IEEE focus areas are developments in the areas of electric and electronic engineering, and radio sciences. IEEE also oversees the development and adoption of international computer and communications standards.

- The **Electronic Industries Association (EIA)** is a non-profit organization that is aligned with ANSI. Its focus is public awareness and lobbying for standards. Its main contributions to data communications technology are the development of interfaces for physical connections and electronic signal specifications for data communications.

1.5.2 Forums

Standards committees are notorious for the slow pace of developments and decision making. Consequently, user groups, university students, industry representatives and experts come together and set-up forums to address the various issues and concerns of data communications technology, and come up with standards from time to time. These forums generally concentrate on a particular technology, and this specialization helps them achieve a great amount of throughput with contributions from a variety of forum members.

The Internet Society (ISOC), Internet Engineering Task Force (IETF), Frame Relay Forum, ATM Forum, ATM Consortium are some of the most well-known forums. The Frame Relay Forum and ATM Forum have interests in specialized technologies such as frame relay and ATM, while the ISOC and IETF are concerned with the development of Internet-related standards and protocols.

1.5.3 Regulatory Agencies

Government-appointed agencies such as the **Federal Communications Commission (FCC)** of the US are always involved in regulating the standards. These agencies help protect interests of the general public in areas such as radio, television and wired communications. For instance, every portion of communications technology must be approved by FCC before it can be sold in the market. FCC periodically reviews the rates charged by service providers, technical specifications of communication hardware and divides, and allocates radio frequencies, etc.

1.6 SIGNAL PROPAGATION

At the basis of data communications is the concept of **signal propagation**. Let us understand what this means with an analogy of heat transfer.

Suppose you have a steel rod with two ends *A* and *B*, with *X* as an intermediate point as shown in Fig. 1.3.

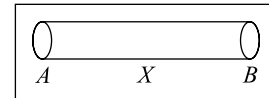


Fig. 1.3 A steel rod

Now suppose that we apply heat to the end *A* (either by immersing it in hot water or a furnace) as shown in Fig. 1.4. Initially, the heat would be felt only at the end *A*. However, after some time, we would also feel the heat at the point *X*. Why does this happen? This is where the principle of energy transfer (transmission) and consequently signal propagation comes into the picture. Because of the heat (rise in temperature), atoms at end *A* start oscillating, and while they are moving, they collide with other adjacent atoms in the way, which also gain momentum because of this. Therefore, the adjacent atoms also start oscillating. This continues, and the atoms at point *X* also start oscillating. Kinetic energy contained in this movement is converted into heat energy, and therefore, the temperature at point *X* increases. Clearly, if this process continues, after some more time, atoms at point *B* would also start oscillating. Again, kinetic energy would get converted into heat energy, and we would feel the heat event at point *B*.

Now, if we put one end of the steel rod at point A in an ice box, the reverse process will take place and after a finite time called *propagation delay*, we will start feeling cold at point X first and then subsequently at point B . If we continuously put the end A in the furnace and the ice box one after the other very rapidly, we will get the graph of temperature versus time at point A as shown in Fig. 1.4.

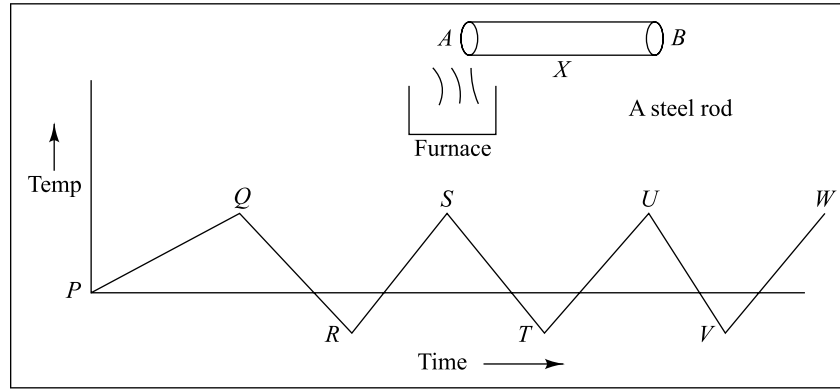


Fig. 1.4 Graph of temperature versus time at one end A

Note that this figure shows the graph of temperature versus time at point A . Interestingly enough, due to the principles of signal propagation discussed above, the same graph is reproduced at point X after some propagation delay, and eventually at point B after some more propagation delay (assuming that there is no distortion). This is shown in Figs 1.5(a), (b) and (c). This means that the signal at point A is reproduced at points X and B after a while, i.e., the signal travels or propagates over a medium.

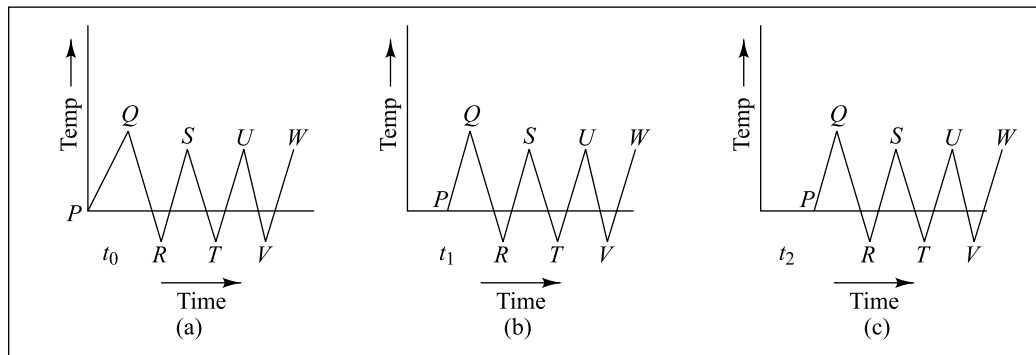


Fig. 1.5 (a) Graph of temperature versus time at point A at time $t = t_0$
 (b) Graph of temperature versus time at point X after the propagation delay t_1
 (c) Graph of temperature versus time at point B after the propagation delay t_2

The electrical signal behaves exactly in the same way. Instead of applying heat, we apply voltage to the electrical wire A - X - B . Instead of atoms oscillating and transferring kinetic energy, the electrons transfer electrical charge to adjacent electrons and the graph of voltage versus time, i.e., electrical signal, you get at point A in the wire is reproduced at point X after some finite propagation delay

and the same is reproduced at point *B* after a little more propagation delay. However, the shape of the signal at point *B* more or less resembles that at point *A*.

If this were not true, telephone systems would be impossible. When we talk on the telephone, our speech generates sound waves, which generate electrical signals of similar shape as the sound waves. These signals traverse across the telephone wires through various switches/exchanges and reach the telephone set of the receiver. At the receiver's end, the electrical signal generates sound waves of similar shape. Due to the principle of signal propagation discussed above, the shape of the signal traverses as it was generated, and therefore, we hear the same speech.

The above process signifies that if we apply some kind of signal (such as heat, voltage, current, etc.) at one end of a conducting material such as a rod or a wire, eventually, the signal is propagated to the other end of the material.

This forms the basis of all data communications.

1.7 ANALOG AND DIGITAL SIGNALS

Any signal can be classified into one of the following two types: **analog** and **digital**. An analog signal is a continuously varying signal, similar to a sinusoidal waveform. For instance, if we measure the room temperature continuously and plot its graph with time on the *x*-axis and temperature on the *y*-axis, we would get a continuous waveform as shown in Fig. 1.6, which is an analog signal. Analog is always continuous.

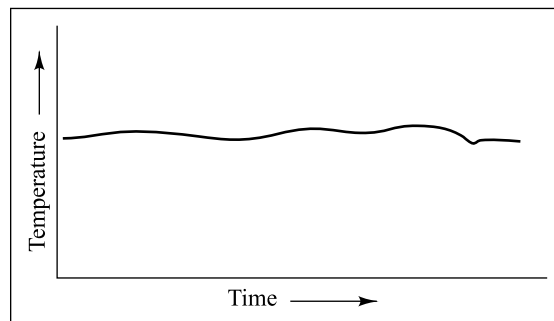


Fig. 1.6 Analog signal

In contrast, digital signal takes the form of pulses, where we have *something* or *nothing*, (i.e., 1 or 0 in digital terms). The absence or presence of something (i.e., 0 or 1) can be used to plot a digital signal. A typical digital signal is shown in Fig. 1.7.

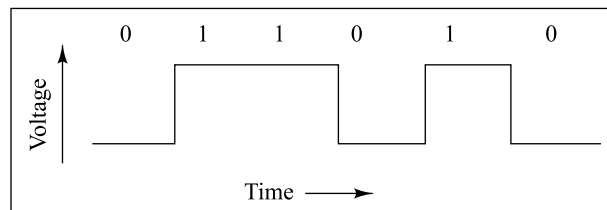


Fig. 1.7 Digital signal

Thus, we can say that an analog signal is smooth and continuous in nature. In contrast, a digital signal represents a sudden jump from one value to another. Since the flat maximum and minimum values shown for the digital signal are fixed, we can assume that these maximum and minimum values are also fixed.

Again, regardless of whether the signal is analog or digital, if we apply this signal at one end of the wire, the signal form will travel at the other end (with some distortions caused due to noise, which we shall study later). For instance, let us imagine that we have a wire $A-X-B$ (like the one shown in Fig. 1.3). Let us imagine that we apply 0 voltage to point A for some time (representing 0), then 5 volts for a longer duration (representing 1), then again 0 voltage for some time (representing 0), etc., This will mean that we are applying a digital signal as shown in Fig. 1.7 at point A . Due to the principle of signal propagation, the same signal (i.e., 011010 in this case) will traverse to point X and then to point B with some distortion and after a finite propagation delay.

1.8 BANDWIDTH OF A SIGNAL AND A MEDIUM

1.8.1 Introduction

The term **bandwidth** is very commonly used in data communications. The basic idea behind bandwidth can be understood quite easily with a simple example of pipes carrying water to our homes. How much water can a pipe carry, at full capacity, at any given time? This maximum capacity of the pipe at a given instance is its bandwidth.

To understand this term in the context of data communications, we would need to take a look at the three major components of a transmission signal. These three components help us understand the concept of bandwidth. The concept of bandwidth is essential in understanding how much data can be transmitted through a wire, similar to the rate at which water can flow through a pipe. Bandwidth is a fundamental property of any wire that is used for data transmission. Bandwidth is also a property of any signal, as we will see.

A question is what can we say about any signal, apart from expressing it in a graphical form on paper or a screen? Can we analyze it further? Can we compare any two signals?

French mathematician, Fourier, gave the answer to these questions in the affirmative. He proved that any signal could be shown to be equivalent to the addition of one or many sinusoidal signals, and that this was true about analog as well as digital signals. This was truly a revolutionary concept in communications technology.

In order to understand this concept better, let us first study how a sinusoidal signal looks like. Figure 1.8(b) shows such a signal.

The signal depicts voltage on the y -axis and time on the x -axis. We also know that if we apply this signal at one end of the wire, after a finite propagation delay, the (more or less) same signal pattern would be received at the other end of the wire, due to the principle of signal propagation. Figure 1.8(a) shows a circle and a point S on the x -axis. We can imagine that a particle at point S starts rotating clockwise at 10 revolutions/second along the circumference of the circle. As it rotates, if we plot its height on the y -axis against time t , we will get a sinusoidal graph as shown in Fig. 1.8(b). For instance, at point S , its height or y -coordinate equals 0, which corresponds to the point J or $(0, 0)$ or the origin in Fig. 1.8(b).

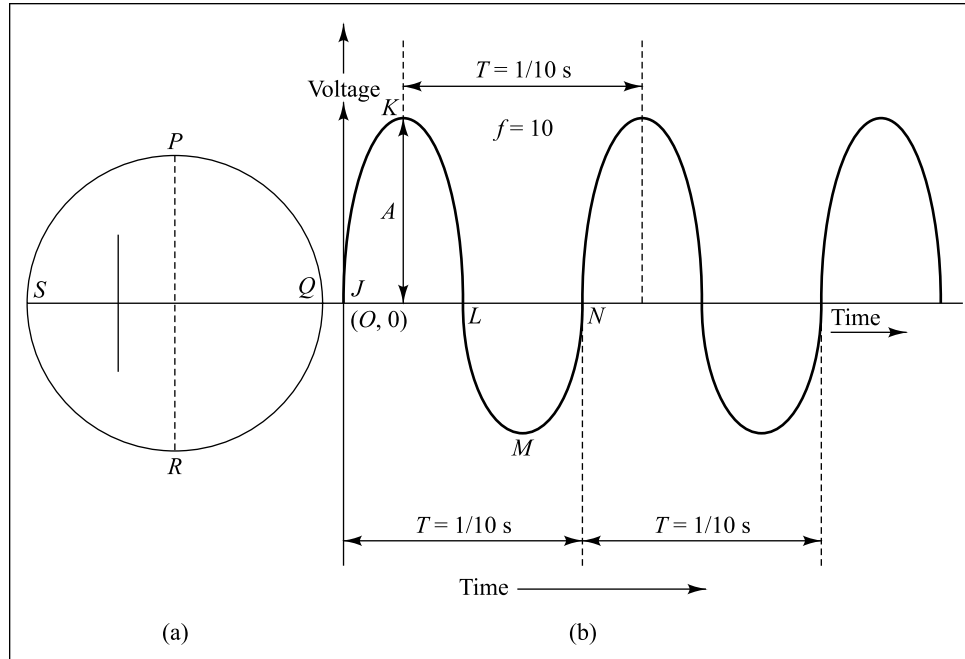


Fig. 1.8 A sinusoidal signal

When the particle traverses to the point P of Fig. 1.8(a), it reaches the highest value, which is depicted by point K in Fig. 1.8(b). When the particle traverses from P to Q in Fig. 1.8(a), its y -coordinate becomes zero again (point Q is on the x -axis) which is depicted by point L in Fig. 1.8(b). This completes half the cycle. It is now easy to see that as the particle traverses further, the point R in Fig. 1.8(a) corresponds to point M in Fig. 1.8(b), and point S in Fig. 1.8(a) corresponds to point N in Fig. 1.8(b). This completes the next half cycle. Both half cycles together complete one revolution. As the particle continues to rotate, the sinusoidal waveform is repeated if we continue to plot the height versus the time. This was only to illustrate the concept of a sinusoidal waveform. Let us now imagine that we have a machine that will change the voltages applied to one end of the wire (A) to generate a sinusoidal electrical signal in the wire at point (A). This signal pattern now will be repeated across the wire to the other end of the wire.

1.8.2 Amplitude, Period, Frequency and Phase

At any point, the strength of the signal or **amplitude** is given by the y -axis. Thus, the amplitude of a signal is equal to its vertical distance from the horizontal x -axis. The figure shows that the signal has maximum value, called *amplitude* (A) shown as A at point K . Amplitude is measured in volts (representing voltage), amperes (representing current) or watts (representing power), depending on the type of the signal.

We see that the signal repeats itself, i.e., completes one *cycle* after time $= T$. That is, the signal reaches the peak and comes back to its original starting position in time T . This time taken for the completion of one cycle is called **period**, which is shown as $1/10^{\text{th}}$ of a second in the figure. This is because the particle in our example of Fig. 1.8(a) was rotating at 10 revolutions per second.

This means that the signal value will be same at times t and $t + 1/10^{\text{th}}$ of a second, for any value of t . Period is expressed in seconds or smaller units of a second such as milliseconds or ms (i.e., 10^{-3} seconds), microseconds or μs (i.e., 10^{-6} seconds), nanoseconds or ns (i.e., 10^{-9} seconds), or picoseconds or ps (i.e., 10^{-12} seconds).

Another term, which is used very often, is **frequency**. The frequency (f) is nothing but the number of cycles or periods a signal completes in one second. We can notice that this would be same as the number of revolutions that our particle would make in one second. Figure 1.9 shows that $f = 10$.

It can be easily shown that $f = 1/T$. For instance, in our example, $T = 1/10^{\text{th}}$ of a second, i.e., the signal completes one cycle in $1/10^{\text{th}}$ of a second. Therefore, in one second, it will complete $1/(1/10) = 10$ cycles. Therefore, the frequency of this signal is 10, i.e., $f = 10$ cycles per second and $f = 1/T$.

Frequency is measured in **hertz (Hz)**, named after a German mathematician. 1 Hz is 1 cycle/second. Therefore, in the example above, the frequency is 10 Hz. We use 1 **kilohertz** or 1 **kHz** to mean 1000 Hz, and 1 **megahertz** or 1 **MHz** to mean 1000 kHz or 1000000 Hz.

Let us discuss a few simple examples to understand how period and frequency relate with each other.

Example 1.1

A sine wave has a frequency of 8 Hz. What is its period?

Solution

Since period (T) is the inverse of frequency (f), we have:

$T = 1/f = 1/8 = 0.125$ seconds is the period of the sine wave.

Example 1.2

A sine wave has a frequency of 5 Hz. What is its period?

Solution

Since period (T) is the inverse of frequency (f), we have:

$T = 1/f = 1/5 = 0.2$ seconds is the period of the sine wave.

Example 1.3

A sine wave completes one cycle in 3 seconds. What is its frequency?

Solution

Since period (f) is the inverse of period (T), we have:

$f = 1/3 = 1/3 = 0.33$ Hz is the frequency of the sine wave.

Example 1.4

A sine wave completes one cycle in 15 μs . What is its frequency?

Solution

Since period (f) is the inverse of period (T), we have:

$f = 1/3 = 1/(15 \times 10^{-6}) = 66,666.66$ Hz or approximately 66×10^3 Hz = 66 KHz is the frequency of the sine wave.

Figure 1.9 shows periodic signals with frequencies of 1 Hz, 2 Hz and 3 Hz, respectively. We notice that as the frequency reduces, the shape of the curve becomes flatter because the signal changes slowly. Also as the frequency increases, the shape of the curve becomes more vertical as the signal changes more rapidly. Thus, in general, the more often the value of a signal changes,

higher is the frequency of the signal. In contrast, when a signal changes from one value to another slowly, its frequency is low.

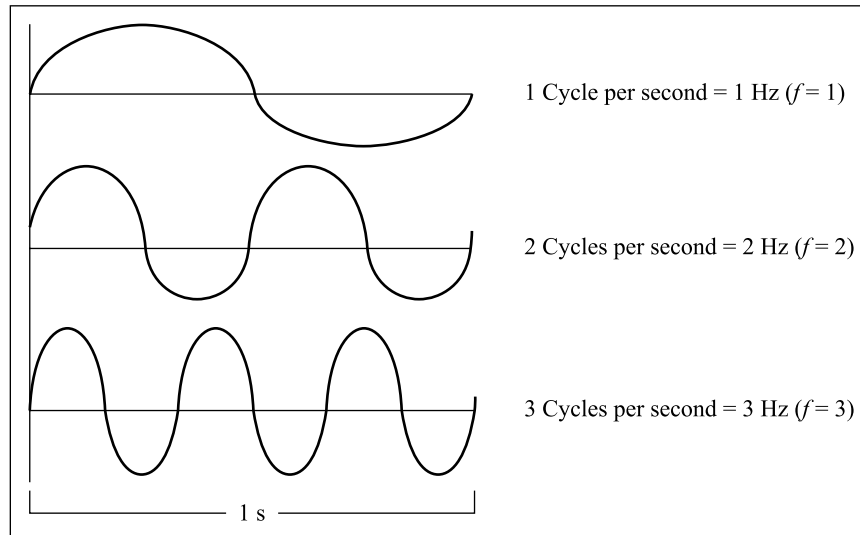


Fig. 1.9 Frequency

When the value of a signal does not change at all, it never completes even one cycle. Thus, there is no question of having a period for signal measurement at all. Thus, its frequency is 0, and we get a horizontal straight line (DC Signal). As we know, this is simply because the value of the signal does not change at all.

In contrast, when a signal changes its value instantaneously, its period is 0, because it does not take even a small amount of period for the signal to change its value. Thus, the frequency in this case is equal to $1/0$, i.e., is infinite (∞), and we get a vertical line.

These ideas are shown in Fig. 1.10.

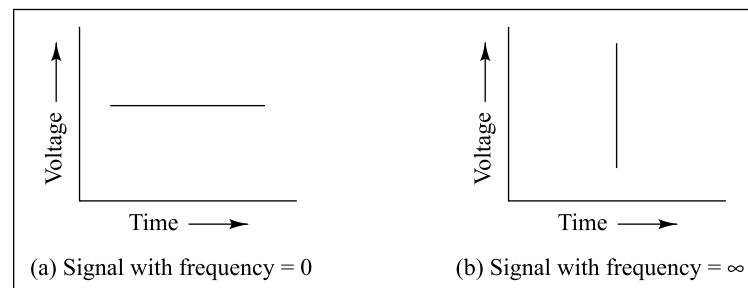


Fig. 1.10 Signals with frequency 0 and ∞

We will notice from Fig. 1.10 that any digital signal is made up of horizontal and vertical lines. Therefore, it must contain signals of both 0 and ∞ frequencies. We shall talk more about it later.

Another concept, which is used to describe a sinusoidal signal, is its **phase (P)**. Figure 1.11 shows four sinusoidal waves with phase differences of 90° from the previous one in sections (a), (b), (c) and (d).

Phase of a signal is related to the position of a waveform relative to time zero. A good way of understanding the concept of the phase of a signal is to imagine it as something that can be shifted to the left-hand side or the right-hand side along the x -axis (i.e., relative to time). The shift represents the phase of the signal. Phase is measured in **degrees** or **radians**. Thus, when we say that a signal has a phase shift of 360 degrees, we actually mean that a complete period of the signal has been shifted. Similarly, a phase shift of 180 and 90 degrees, respectively, represent a half-and-quarter shift of the signal's period.

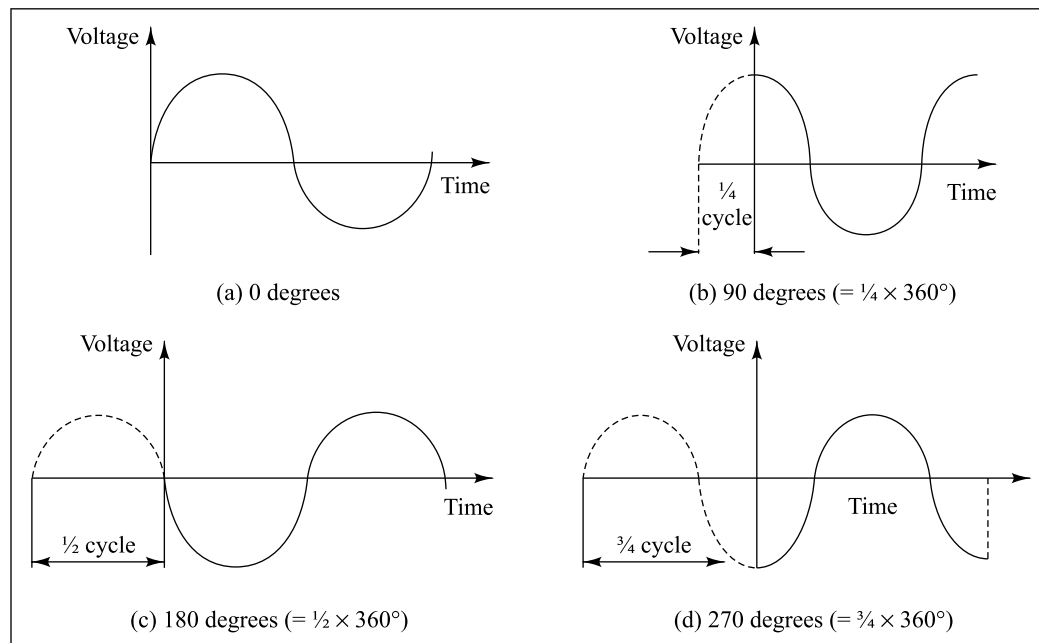


Fig. 1.11 Phase in a sinusoidal signal

We notice that amplitude (A), frequency (f) and phase (P) completely describe a sinusoidal signal. We have been plotting a sine wave against time. Such a diagram shows the changes in a signal's amplitude with respect to time. However, such a diagram is generally not used to measure the frequency and phase of a signal. These parameters of a signal are measured using concepts that we shall study now.

1.9 FOURIER ANALYSIS AND THE CONCEPT OF BANDWIDTH OF A SIGNAL

Now, we return to Fourier's theory. How can we decompose a random signal such as that generated by a telephone conversation into various sinusoidal signals of varying amplitudes, frequencies and phases? We do not want to give a mathematical proof of this here. We will just take an example to illustrate this concept.

Figures 1.12(a), (b) and (c) show three sinusoidal signals with different amplitudes and frequencies (e.g., 10 Hz, 30 Hz and 50 Hz). In Fig. 1.12(a), the signal completes one cycle in $1/10^{\text{th}}$ of a second, i.e., its frequency = 10 Hz (10 cycles/s). Similarly, it is easy to see that the signals in Fig. 1.12(b) and 1.12(c) have frequencies 30 and 50, respectively.

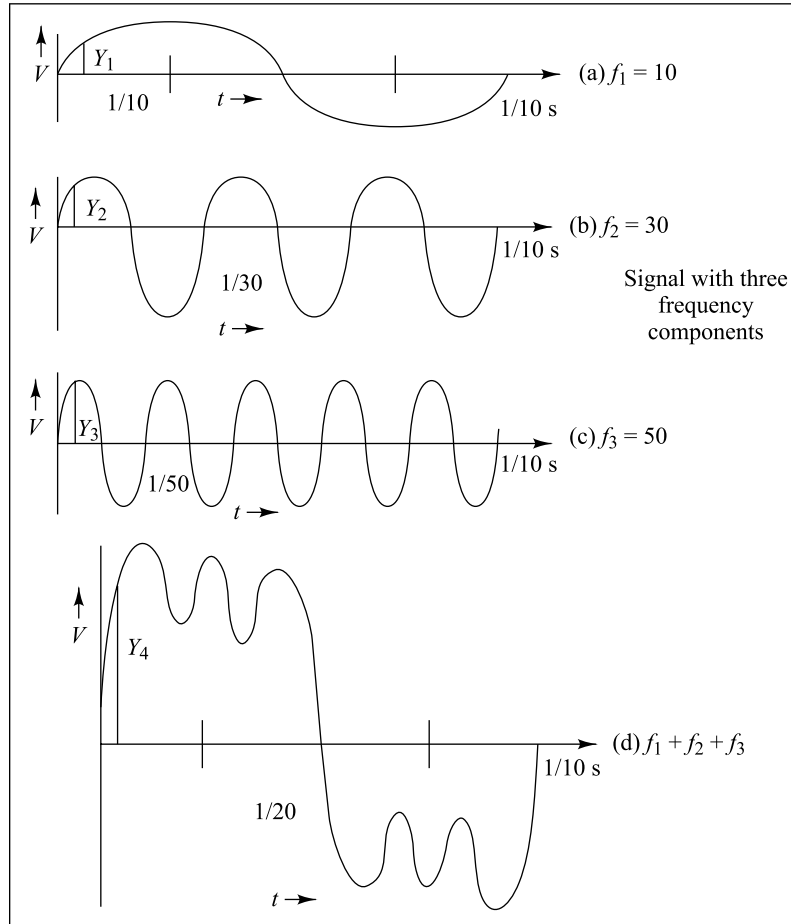


Fig. 1.12 Fourier analysis illustrated

For every value of time = t , on the x -axis, if we add the signal strength (amplitude) of these three signals and plot the result as the signal strength of the resultant signal on the y -axis, we get the resultant analog signal as shown in Fig. 1.12(d). For instance, if we add Y_1 , Y_2 and Y_3 , we will get Y_4 . If we do that for all values, we will get the resultant signal shown in Fig. 1.12(d). Conversely, we can show that the analog signal shown in Fig. 1.12(d) is derived by superimposing the three sinusoidal signals in Figs 1.12(a), (b) and (c). Similarly, it can be shown that any signal can be decomposed into different sinusoidal signals. This process is called **Fourier analysis**.

We can now similarly imagine that if we carry out a Fourier analysis on any complex signal, we will get a set of sinusoidal signals of different frequencies. Now, if we find out the difference between the highest and the lowest frequencies from within the sinusoidal signals making up that complex

signal, we get a very useful concept. We call this difference **bandwidth** of the complex signal. For instance, the bandwidth of the signal in Fig. 1.12(d) is $50 \text{ Hz} - 10 \text{ Hz} = 40 \text{ Hz}$. The simplest way to describe bandwidth is the fastest continually oscillating signal that can be sent across a wire.

Let us understand this with a few examples.

Example 1.5

A periodic signal has been decomposed using Fourier analysis to yield four sine waves of frequencies 100, 400, 600 and 800 Hz. What is the bandwidth of the resulting periodic signal?

Solution

We know that the bandwidth of a signal is the difference between the highest and the lowest frequencies. Therefore, the bandwidth of the periodic signal = $800 - 100 = 700 \text{ Hz}$

Example 1.6

A signal has a bandwidth of 20 Hz and its highest frequency is 60 Hz. What is its lowest frequency?

Solution

Since the bandwidth of a signal is the difference between its highest and lowest frequencies, in this case, we have the following solution:

$$\text{Lowest Frequency} = \text{Highest Frequency} - \text{Bandwidth} = 60 - 20 = 40 \text{ Hz}$$

Similarly, we can find out the bandwidth of any signal of any shape and size. It is clear that if the signal changes very rapidly (as in a video signal), the higher frequency will be very high, i.e., when we decompose this signal into various sinusoidal signals, we will have some signals with very high frequencies. Therefore, the bandwidth of such signals will be very high. If the signal changes very slowly, its bandwidth will be very small. At an extreme, a digital signal has both horizontal and vertical lines, and therefore, signals with 0 and ∞ frequencies. Therefore, the bandwidth of a digital signal is $\infty - 0 = \infty$. We will return to this theme shortly.

We must point out that similar to the bandwidth of a signal, there is a bandwidth of a medium. What does this mean? Suppose that we have a thin water pipe and a very forceful water flow. The thinness of the water pipe (i.e., the *medium*) in such a case represents the fact that its bandwidth is quite small as compared to the bandwidth of water (i.e., the *signal*) that needs to be carried through it. This should clarify that both the signal and the medium that will carry it have their own bandwidths.

As we can imagine, even in case of a medium, its bandwidth is the difference between the maximum and the minimum frequencies that it can carry. For many media (except radio), the minimum frequency is 0. Therefore, generally, the bandwidth of a medium is also the same as the maximum frequency that it can carry. Why should there be a limit on this highest frequency? The reason lies in the nature of the medium. Depending on the metal or the material used for the medium (i.e., free electrons, etc.), changes in the signals at one end are transmitted to the other end faithfully only up to a particular rate at which the signal changes (i.e., frequency). If this rate increases more than a certain limit, before the change in the signal is transmitted from points *A* to *B*, the signal at point *A* would change, and therefore, faithful transmission would not be possible. The signal would get distorted beyond repair. In simple terms, if the sender attempts to transmit signals that are faster than the bandwidth, the underlying hardware would not be able to keep up

with the pace, because it would not have sufficient time to complete one change before the sender attempts to send another. This defines the maximum rate at which hardware can change a signal, i.e., the bandwidth of a medium. As we shall see, the bandwidth of an optical fiber is far more than that of a coaxial cable, which, in turn, is far higher than that of a twisted wire pair.

An interesting point is that a signal can be carried by a medium only if the bandwidth of the signal is less than that of the medium. It is for this reason that a telephone conversation, whose bandwidth is small, can be sent across a twisted wire pair, but TV/video signals, whose bandwidth is high, require a coaxial cable or an optical fiber.

We earlier mentioned that Fourier analysis could be carried out on any signal. Let us study how this can be done using Fig. 1.13 on a digital signal.

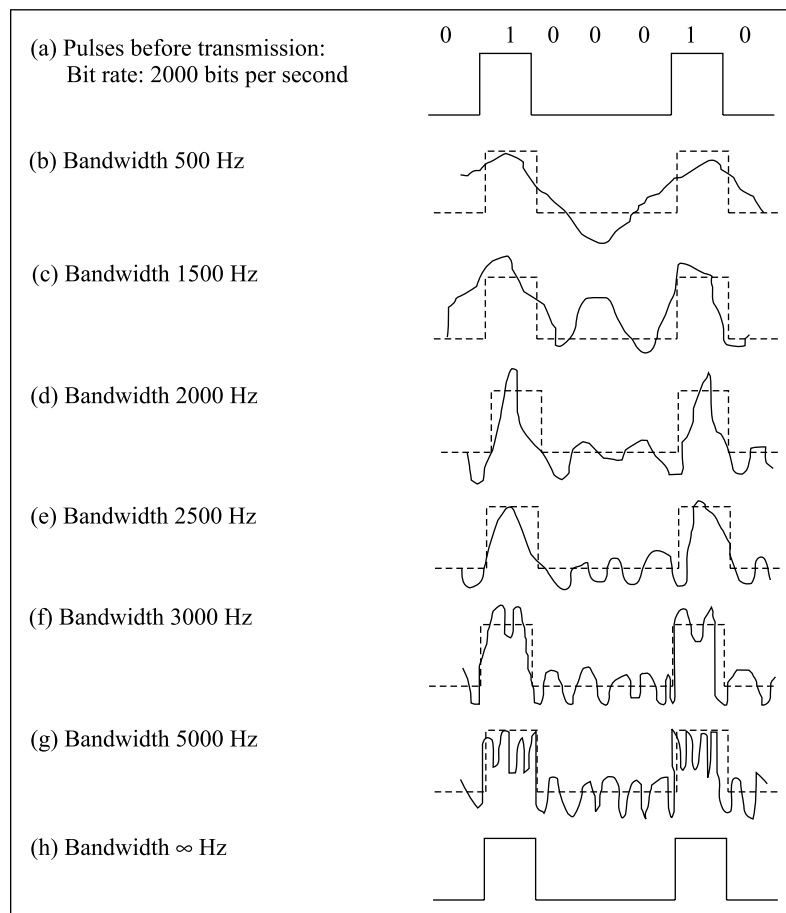


Fig. 1.13 *A digital signal with infinite bandwidth*

Figure 1.13(a) shows the original digital signal. Figure 1.13(b) shows the resultant signal if we start with a sinusoidal signal of bandwidth 500 Hz, i.e., we superimpose the signals with frequencies between 0 and 500 and get this resultant signal. We notice some similarity between the two signals at the peak value of the signal (highest amplitude) but there is still a large difference between them. As

we move down the Figs 1.13(c), (d) and so on, we add higher and higher frequencies, i.e., increase the bandwidth of the resulting signal, the resulting signal starts looking more and more like the original digital signal. It is now easy to imagine that if we add all frequencies from 0 to infinity (∞), we will get a signal with infinite bandwidth, which will be the same as the digital signal, as shown in Fig. 1.13(h). This can also be proved mathematically. Therefore, the digital signal is said to have infinite bandwidth. Let us look at this concept a little more closely.

We will notice from Fig. 1.13(a) that the digital signal increases its value from 0 to maximum *instantaneously*, i.e., literally in no time, whereas a sinusoidal signal takes a finite time to reach the maximum value. It is easy to imagine that the higher the frequency of the analog signal, the quicker it takes to reach this maximum value, but still it takes some finite time. That is why a vertical line parallel to the y -axis represents a signal with $^\circ$ frequency, and a horizontal line parallel to the x -axis represents a signal with 0 frequency. As a digital signal is composed of signals which are parallel to the x - and y -axis (i.e., horizontal and vertical lines) it should be now easy to imagine why a digital signal will have a bandwidth of 0 to ∞ , i.e., ∞ bandwidth. The question is that if no medium has an infinite bandwidth, how can it carry a digital signal at all?

To understand this, let us take a digital wave or signal consisting of bits 010101... as shown in Fig. 1.14. The figure shows that the signal repeats itself in 1 second, i.e., it has a frequency of 1 Hz. Mathematically it can be shown that this signal is made-up of a simple sinusoidal signal having the following frequencies (1 Hz + 3 Hz + 5 Hz + 7 Hz + 9 Hz + ... to infinity) and different amplitudes. Each component has a particular frequency and particular amplitude.

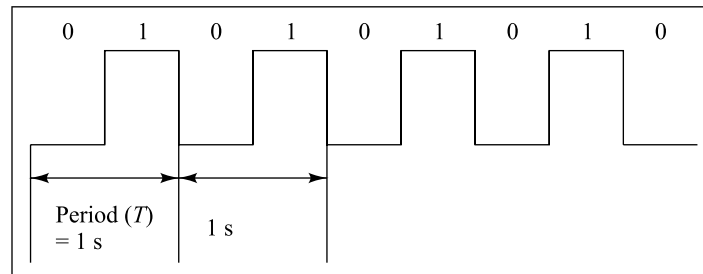


Fig. 1.14 A digital signal of 1 Hz

As the frequency of the constituent signal increases, the amplitude of that constituent signal decreases. For instance, the amplitude associated with the signal with frequency = 7 Hz is more than the one for the signal with frequency = 9 Hz and so on. Therefore, generally, higher frequency signals are not as important as the lower frequency ones, as their amplitudes are far smaller. That is how we get the shape of the resultant signal similar to the digital signal, even if we remove some of the very high frequencies. That is why even if the bandwidth of a medium is not infinite, and therefore, constituents of higher frequencies are likely to be chopped off, it can still more or less resemble the digital signal. Consequently, all we will need is some equipment which can *recognize* that the signal has *some* amplitude or *none at all*. (i.e., it is 1 or 0). Then, it still can be recognized as a digital signal consisting of 0 and 1, where 0 and 1 are mere conventions for the absence or presence of amplitude.

Obviously, due to the chopping-off of the higher frequencies, or due to the distortions caused due to noise, if the equipment measures the resultant signals, the voltages will not be exactly

corresponding to the binary values 0 and 1. The equipment should still be wise enough to carry out these approximations. For instance, if 5 V is designated for a bit value 1, and if the receiving equipment receives it as 4.90 V, it still can recognize it as a bit value 1. However, if it goes down to 1.5 V, should it be read as 0 or a 1? We will learn more about this later, when we talk about repeaters and how they help overcome some of these problems.

Unfortunately, this *intelligence* cannot be built in the analog equipment. As the signal values are continuous, any value that the signal takes is legitimate. If, therefore, the signal value changes from 5 V to 4.90 V, the analog equipment will treat it as legitimate. Recall that in case of a digital signal, this would have been considered as 5 V, and not 4.90 V, because there was no concept of 4.90 V in case of a digital signal.

Therefore, errors or distortions in analog systems are irrecoverable. On the other hand, in digital systems, they are recoverable. This is why sound of a compact disc (CD) is more reliable than a cassette, as they use digital and analog systems, respectively. This is one of the most important reasons why the world is going digital in all respects.

1.10 DATA TRANSMISSION RATE AND BANDWIDTH

What is the connection between the bandwidth of a signal and the data rate that it represents? Let us explore this now.

For simplistic understanding, we can consider a sinusoidal wave again as shown in Fig. 1.15. The wave has frequency = 10 Hz, i.e., it completes one cycle in $1/10^{\text{th}}$ of a second, or it completes 10 cycles in one second.

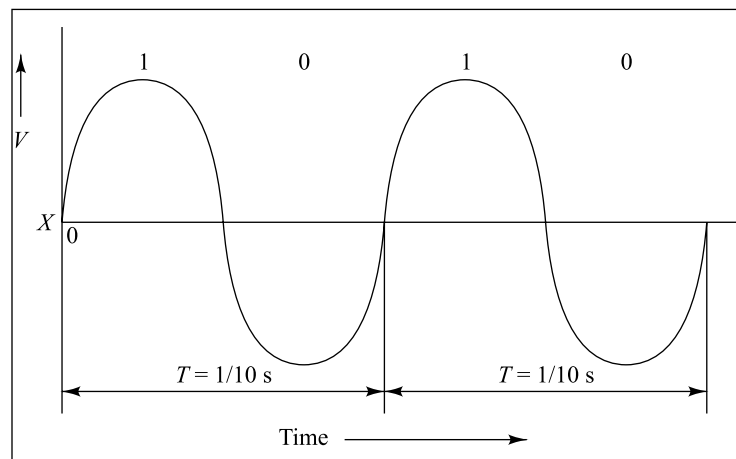


Fig. 1.15 A sinusoidal wave with frequency = 10 Hz

We can imagine that the peak represents a binary 1 and valley represents a binary 0. Clearly, with this understanding, this signal also can represent digital data. The figure shows that the wave completes 1 cycle in $1/10^{\text{th}}$ of a second. Therefore, in 1 second, it will complete 10 cycles. As each cycle represents 2 bits, the effective data rate is $10 \times 2 = 20$ bits per second or 20 bps. If we increase the frequency of the sinusoidal signal to 20 Hz, we will get a data rate of 40 bps, and so

on. Therefore, as the frequency becomes very high, the data rate that it represents also becomes very high (in our case, twice the frequency).

However, representing digital data by a pure sinusoidal analog signal is dangerous. The equipment, which has to measure the voltage to recognize 1 and 0, will have to be super accurate and super sensitive. It has to measure voltages exactly at the peaks and valleys. A small shift will result into a very large error and it will continue for all the subsequent readings also.

In a pure digital signal, the voltage levels corresponding to values 1 and 0 remain steady for a while for the reading to be taken more or less accurately. How can we get such a signal? We know that to get a digital signal would mathematically mean to get signals with frequencies between 0 to (infinite) bandwidth. This is clearly not practicable. There is not yet a medium that is available, which has an infinite bandwidth, which can carry any frequency. There has to be some alternative. Thus, if we get a signal, which is close to a digital signal, it will also do.

Let us look at Fig. 1.13 again. We see that the resultant signal has three components of 10 Hz, 30 Hz, and 50 Hz frequencies. Therefore, we concluded that the bandwidth of that signal is $50 \text{ Hz} - 10 \text{ Hz} = 40 \text{ Hz}$. If we look at the resultant signal, we see that it resembles a digital signal, which is transmitting a bit every $1/20$ th of a second, i.e., with data rate of 20 bps. We know that this signal is not a perfect digital signal, but we also know that we can construct equipment, which will measure the voltages at different points, and will be able to *recognize* 1 and 0. If we had not added the signal with frequency = 50 Hz, the bandwidth of a resultant signal would have been $30 \text{ Hz} - 10 \text{ Hz} = 20 \text{ Hz}$. However, the resultant signal would be further away from a digital signal not *recognizable* by the equipment, thus increasing the possibilities of errors. Therefore, it is advisable to add the 50 Hz frequency. If we had included the frequency of 70 Hz, it was also ok, but not necessary. The equipment is good enough to recognize it as a digital signal even if we go up to 50 Hz. Therefore, we will assume that in our example, we use frequencies of 10 Hz, 30 Hz and 50 Hz to generate a *near-digital* signal. This means that we have used a signal of 40 Hz bandwidth to encode a digital signal with the data rate of 20 bps.

Let us imagine now, that we double the frequencies of all the signals, i.e., we have signals with frequencies 20 Hz, 60 Hz and 100 Hz. It is not difficult to imagine that the resulting near-digital signal will complete one cycle (i.e., 2 bits) in $1/20$ th of second instead of a $1/10$ th of a second as in the earlier case. That means that the data rate has increased (doubled) to $20 \times 2 = 40 \text{ bits/s} = 40 \text{ bps}$, instead of the 20 bps earlier. However, the bandwidth of the signal also has increased (doubled), which is now $100 \text{ Hz} - 20 \text{ Hz} = 80 \text{ Hz}$, instead of the 40 Hz earlier. It will be clear from this that as the bandwidth of a signal doubles, the data rate that it encodes also doubles. It would be clear from the preceding discussion that the rate at which the information can be transmitted is directly proportional to the bandwidth. This was discovered as early as 1928 by a researcher named Hartley.

We know that every medium has a limited bandwidth. Imagine a hypothetical medium M_1 with a bandwidth of only 40 Hz and a data rate of 20 bps as discussed earlier. Imagine another medium M_2 with a bandwidth of 80 Hz, with a data rate of 40 bps as the preceding discussion shows. We can divide the physical, actual bandwidth of this second medium into a number of *channels* with 40 Hz bandwidth each. In our example, one channel could consist of frequencies between 20 Hz and 60 Hz. The other one could be between 60 Hz and 100 Hz. In this case too, the maximum data rate that can be achieved in the medium as a whole still remains at 40 bps with each channel having a data rate of 20 bps as shown in Fig. 1.16.

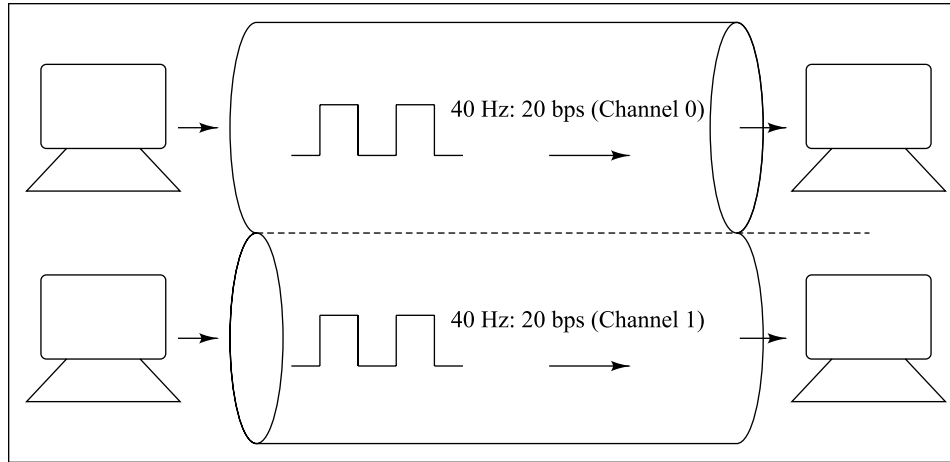


Fig. 1.16 *A medium and channels*

If we had dissected the bandwidth of the medium in four channels of 20 Hz each, the data rate of each channel would have been 10 bps so that the total frequency bandwidth and the total data rate do not change. We know that the data rate of a medium is proportional to the bandwidth of a medium. In many systems, the physical bandwidth of a medium is logically divided into a number of channels and each channel carries data between two separate computers. However, not all computers may be sending the data at the same time, thereby wasting the channel's capacity. Techniques that permit intelligent dynamic allocation of channels to a computer will exist. When a computer makes a request for sending some data, the remaining available capacity in a medium is consulted and a channel is carved out of that capacity and then allocated. Obviously, when a computer stops transmitting even temporarily, the channel capacity is returned to the available, allocable pool. In this way, you can connect 10 computers, each transmitting at 40 bps to a medium whose maximum data rate is only 200 bps.

SUMMARY

At the basis of all computer-to-computer data communications is the concept of signal propagation. The simple idea in signal propagation is that when some kind of signal is applied in any form (such as heat, voltage or current) to one end of a conducting material, after some time the signal reaches the other end of the material. Thus, a wire can also be used for carrying data from between two points which it connects.

Fourier proved that any signal could be shown to be equivalent to the addition of one or many sinusoidal signals, and that this was true about analog as well as digital signals. Any sinusoidal signal has the following three properties: amplitude (the strength of the signal), period (the time taken by a signal to complete one cycle) and frequency (the number of signal changes in one second). The bandwidth of a signal is the difference between its highest and lowest frequencies. Frequency can be calculated as $F = 1 / T$, where F is the frequency and T is the time period. An analog signal is a continuous waveform. A digital signal is a discrete waveform. The bandwidth of a digital signal is infinite.

KEY TERMS AND CONCEPTS

American National Standards Institute (ANSI)	Institute of Electrical and Electronics Engineers (IEEE)
Amplitude	International Standards Organization (ISO)
Analog	International Telecommunications Union-Telecommunications Standards Sector (ITU-T)
Bandwidth	Kilohertz (kHz)
Bits per second (bps)	Megahertz (MHz)
Bit rate	Open Systems Interconnection (OSI)
Consultative Committee for International Telegraphy and Telephony (CCITT)	Period
Data transmission rate	Phase
<i>de facto</i>	Protocol
<i>de jure</i>	Radians
Degrees	Regulatory agencies
Digital	Semantic
Electronic Industries Association (EIA)	Signal propagation
Federal Communications Commission (FCC)	Standards Creation Committees
Forums	Syntax
Fourier analysis	Timing
Frequency	Transmission Medium
Hertz (Hz)	

QUESTIONS**True/False**

- Standards are irrelevant to data communications.
- Protocols govern a set of rules for data communications.
- Forums are generally set-up by user groups.
- A signal can be shown to be equivalent to the addition of one or many sinusoidal signals.
- Amplitude is the same as the frequency of a signal.
- Frequency can be termed as the number of revolutions that a particle would make in one second.
- As the frequency reduces the shape of the curve, a wave becomes more vertical.
- The difference between the highest and the lowest frequencies from within the sinusoidal signals making up that complex signal can be called bandwidth of the complex signal.
- Bandwidth is the slowest continuously oscillating signal that can be sent across a wire.
- Data transmission rate decreases with increase in the frequency.

Multiple-Choice Questions

- The three important goals of data communications are correct delivery, accurate delivery and _____.

22 *Data Communications and Networks*

- (a) late delivery (b) on the spot delivery
(c) timely delivery (d) delivery on the next day
2. A set of rules that govern data communications between the sender and the receiver is called _____.
- (a) rule (b) protocol
(c) law (d) option
3. The de jure standards apply because of _____.
- (a) conventions (b) agreement
(c) regulation (d) choice
4. _____ standards are highly desirable because anyone can use them.
- (a) Free (b) Open
(c) Closed (d) Ready
5. The fundamental basis of data communications is _____.
- (a) light (b) electricity
(c) voltage (d) signal propagation
6. The time required for passing energy from one point to another is called _____.
(a) propagation delay (b) movement delay
(c) change delay (d) passing delay
7. _____ signals are continuous in nature.
- (a) Analog (b) Digital
(c) Mixed (d) None of the above
8. _____ signals are on-off in nature.
- (a) Analog (b) Digital
(c) Mixed (d) None of the above
9. The concept of _____ can be understood with the help of the example of a water pipe.
- (a) analog signals (b) digital signals
(c) amplitude (d) bandwidth
10. _____ showed that any waveform can be shown as made-up of a number of smaller signals.
- (a) Fourier (b) Newton
(c) Einstein (d) Hartley
11. The strength of a signal at any point of time is called as its _____.
(a) amplitude (b) frequency
(c) period (d) phase
12. The time required for the completion of one signal cycle is its _____.
(a) amplitude (b) frequency
(c) period (d) phase
13. The number of cycles completed by a signal in one second is its _____.
(a) amplitude (b) frequency
(c) period (d) phase
14. The unit used to measure frequency is _____.
(a) hertz (b) volts
(c) ampere (d) seconds

15. When a signal changes its value instantly, its frequency is _____.
(a) zero (b) one
(c) two (d) infinite
16. The position of a signal relative to time zero is _____.
(a) amplitude (b) frequency
(c) period (d) phase
17. When the bandwidth of a signal is 100 Hz, the underlying hardware can send signals that are _____ 100 cycles per second.
(a) more than (b) less than
(c) more than or equal to (d) less than or equal to
18. If a signal's lowest frequency component is 100 Hz and its highest frequency component is 5000 Hz, then its bandwidth is _____ Hz.
(a) 4900 (b) 5100
(c) 5000 (d) infinite
19. A digital signal with frequency 2 has a data rate of _____ bps.
(a) 1 (b) 0
(c) 4 (d) 2
20. Data rate is closely related with the _____ of a signal.
(a) phase (b) modulation
(c) frequency (d) size

Detailed Questions

1. What are protocols and standards?
2. Discuss three important bodies that help standardize data communications.
3. What is the meaning of signal propagation?
4. Explain the term bandwidth. Why is it useful?
5. Discuss the terms amplitude, period, frequency and phase.
6. Explain Fourier's theory using analog signals.
7. How can Fourier analysis be applied to digital signals?
8. What is data transmission rate and how is it calculated?

2 Analog and Digital Transmission Methods

2.0 INTRODUCTION

We have studied that the two major types of signals are analog and digital. However, the methods in which these two types of signals can be transmitted are also of the same two types, that is, analog and digital. This means that an analog signal can be transmitted as it is, that is, as an analog signal. Alternatively, an analog signal can be transmitted as a digital signal, by encoding the signal by certain methods. Similarly, a digital signal can be transmitted as it is, that is, as a digital signal, or it can be encoded as an analog signal before transmission. Thus, we have the following four possible combinations now:

- Analog Signal, Analog Transmission
- Digital Signal, Digital Transmission
- Digital Signal, Analog Transmission
- Analog Signal, Digital Transmission

We shall discuss these four possibilities in this chapter.

2.1 ANALOG SIGNAL, ANALOG TRANSMISSION

The term *analog* is very common and has been used for decades in the field of telephony. The human voice generates an analog (i.e., continuously varying) signal, which is transmitted as an analog signal over the medium. On the way, the signal suffers **attenuation**. **Amplifiers** are used to overcome this problem, but then amplifiers amplify noise along with the original signal. The problem with this type of combination is that if the signal gets distorted, it cannot be reconstructed at all. It is a permanent loss. At the destination, it is very difficult to imagine, from the received distorted signal, what the signal *should have been*. This is the reason why this type is not used where high level of accuracy is desired. However, in human communications, a slight distortion does not normally matter to the gist or the content of the conversation. Therefore, this has played a major role in telephony in so many past decades. Figure 2.1 shows this.

2.2 DIGITAL SIGNAL, DIGITAL TRANSMISSION

We know that the information coming out of a computer is in the form of digital signals. We also know that a digital signal has an infinite bandwidth, whereas any medium has only a limited

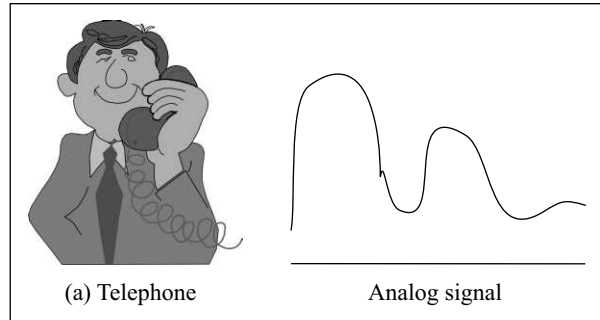


Fig. 2.1 Analog signal, analog transmission

bandwidth. Therefore, as the signal is generated, and enters any medium at that point itself, only limited frequencies are permissible on the medium depending upon its *bandwidth* (this has nothing to do with the noise). If we add all the frequencies admitted on the medium, the resultant signal would not be the same as a digital signal. Therefore, the signal is distorted from the original digital signal to begin with. As it traverses over the medium, noise adds further distortion. Beyond a certain distance, the signal becomes unrecognizable from the original one. Therefore hardware equipment called **regenerative repeater** or **repeater** is used to regenerate the digital signal as shown in Fig. 2.2. We show three points on the path, viz., *A*, *B* and *C*. At point *A*, the signal is in its original digital form. It gets distorted at point *B*. However, you can still recognize the signal as 0100101. The repeater *recognizes* the bits and outputs the signal in its original form at point *C*.

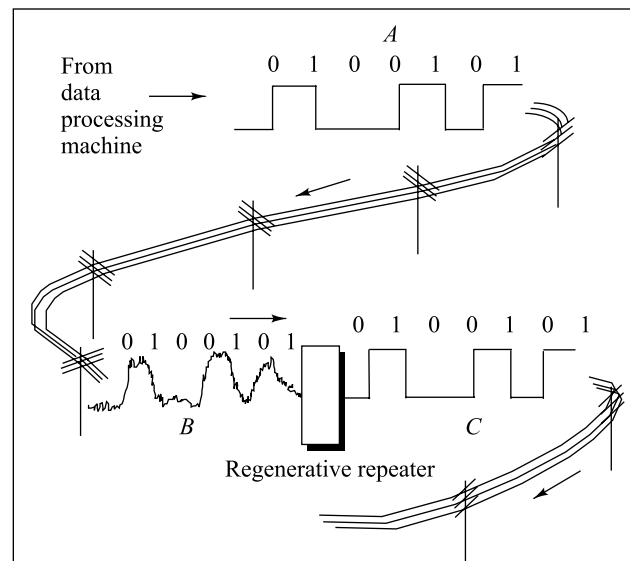


Fig. 2.2 Regenerative repeaters

The input to the regenerative repeater is a signal, which *looks like* a digital signal. Therefore, the repeater measures the signal values at regular intervals to *recognize* the 0s and 1s in the signal and regenerates them. Hence, there is no loss of information.

However, only one repeater will not do; many such repeaters as shown in Fig. 2.3 will be required on the line. The distance between the repeaters is very crucial. We may like to increase that distance as much as possible to reduce the cost but then there is also a disadvantage of this modification. If this distance is very large, the original signal may get so distorted that it may be difficult to differentiate between 0 and 1. Reconstruction of the original signal, therefore, would become very difficult or at best erroneous.

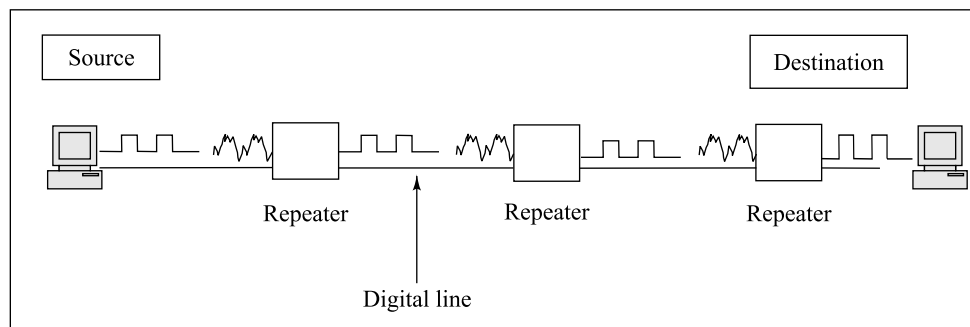


Fig. 2.3 A T_1 line contains many repeaters

Any line with repeaters placed at the appropriate distance is called a **digital line**. Such a line can reproduce the original digital signal at the other end as a computer or any other equipment sent it. AT&T put such repeaters on the wire pairs used for telephonic conversations, separated by a distance of only 6000 feet. This *digital line* is called a T_1 line, and it can carry a data rate of 1,54,400 bits per second (1.54 Mbps). This is shown in Fig. 2.3.

2.3 DIGITAL SIGNAL, ANALOG TRANSMISSION

2.3.1 Modem

In practice, we do not have all digital lines with repeaters on all the lines. Otherwise, we would have sent digital data directly between two computers. When computers came into existence, the telephone network already existed. However, telephones use analog signals and analog circuits. Designers, then, had two choices in front of them. One was to create a new digital network with repeaters, etc., or use the existing telephone network. The former choice was expensive. The problem with the latter was how to send digital signals over an analog network? Some codification technique was necessary to convert a digital signal into an analog one, which could be carried over the telephone network (i.e., with bandwidth of 4000 Hz per channel), and at the other end, convert it back into a digital signal.

We use a **modem** for this purpose. A modem is derived from two components, viz., a **modulator** and a **demodulator**. A modulator basically uses some convention or a coding scheme and converts a digital signal into an analog signal that can be transmitted through that channel (i.e., the derived analog signal must have a bandwidth less than that of the channel of a telephone conversation, i.e., 4000 Hz) and a demodulator converts the analog signal back into the digital signal. Normally, between two nodes A and B, communication takes place in both the directions. Therefore, both modulator and demodulator components are required at both the nodes, and subsequently modems are used at each node as shown in Fig. 2.4.

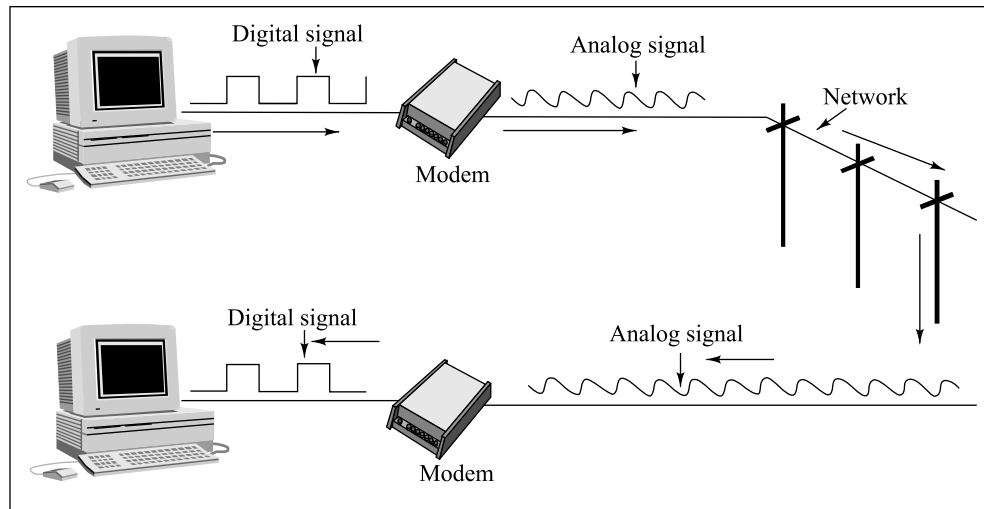


Fig. 2.4 Use of modem for sending digital data over analog lines

As Fig. 2.4 shows, digital signals originating from the computer go through the modem where they are converted (i.e., codified or modulated) into analog signals whose bandwidth is < 4000 Hz. This is because the channel for telephonic conversations requires a bandwidth of 4000 Hz. This analog signal traverses over the telephone line. At the other end, this signal is fed to another modem where it is converted back (i.e., demodulated) into the original digital signal, which the computer at the other end understands.

These days, there is a modem interface provided in a desktop computer itself, as its integral part, so that when the PC is connected to the telephone circuit, the output from the PC is directly converted from digital to analog signal, which traverses along the telephone circuit as if it were a voice signal, and then at the other end it is converted back into the digital signal.

2.3.2 Modulation Techniques

To understand the functioning of a modem at a conceptual level is very easy. Modulation basically uses a coding scheme or a convention. This coding can be achieved using basically the three properties of a signal, viz., Amplitude, Frequency and Phase. Depending upon the technique used, it is called **Amplitude Shift Keying (ASK)**, **Frequency Shift Keying (FSK)** or **Phase Shift Keying (PSK)**. In all these techniques, we choose a carrier signal with given amplitude, frequency and phase, and then choose a form of encoding. Another variation of the basic techniques is **Quadrature Amplitude Modulation (QAM)**, which is a combination technique.

Amplitude Shift Keying (ASK)

In amplitude shift keying or **amplitude modulation**, we do not alter the frequency or phase of the carrier signal. However, we specify different amplitudes, i.e., shift the amplitude values, to represent a binary 1 and a binary 0. Figure 2.5 illustrates this. It is clearly a coding scheme.

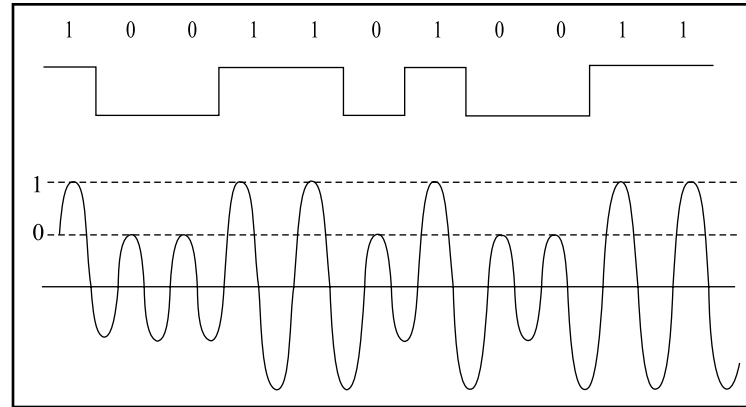


Fig. 2.5 Amplitude Shift Keying (ASK)

The figure shows a bit string 10011010011 modulated as an analog signal. Note that only the amplitudes of the signal change as per the values 0 and 1, but the phase and the frequency for the signals for both 0 and 1 are same and the frequency is between 0 and 4000 Hz. Therefore, the signal can be sent over telephone lines.

At the receiving end, the modem working on this principle basically measures amplitudes at regular intervals to *decode* them as 0s and 1s and then generates a digital signal. The binary bits then can be stored at the destination node. This technique is normally used for transmitting data over optical fibers (which we shall study later, when we discuss transmission media). However, on other lines, it is a very inefficient technique due to noise. Since amplitude is the only aspect of ASK encoding technique, noise is a major problem. Noise usually affects the amplitude of a signal. Therefore, ASK is highly susceptible to noise.

Frequency Shift Keying (FSK)

In Frequency Shift Keying (FSK) or **frequency modulation** technique, the amplitude and the phase of the carrier signals are kept unaltered. A certain frequency f_1 to denote 1 and f_2 to denote 0 is assigned. The frequency of the carrier signal is varied to represent binary 1 (using f_1) and binary 0 (f_2). Both f_1 and f_2 must be in the bandwidth of the channel, i.e., between 0 and 4000 Hz, which can be easily carried by the telephone wires.

This is depicted in Fig. 2.6. The signal component with slower cycle is f_1 , and the signal component that shows rapid cycle portions is f_2 . The two respectively represent 1 and 0 of the input digital signal. It is obvious that the modem at the destination *decodes* these signals into 0s and 1s by measuring the frequencies of the received signals at regular predefined time intervals.

The figure shows the same bit string 10011010011 sent using frequency shift keying. Note that the phase and amplitude for both 0 and 1 are same. Only frequency varies. But both f_1 and f_2 are between 0 and 4000 Hz.

This scheme is less error prone than amplitude shift keying. Noise is not an issue with FSK. Unlike ASK, the voltage changes (i.e., amplitude changes) are ignored by FSK. This is simply because FSK does not look for amplitude changes, but it looks for varying frequency patterns. Therefore, even if there are unwanted amplitude changes (i.e., noise) they are not of any consequence to FSK.

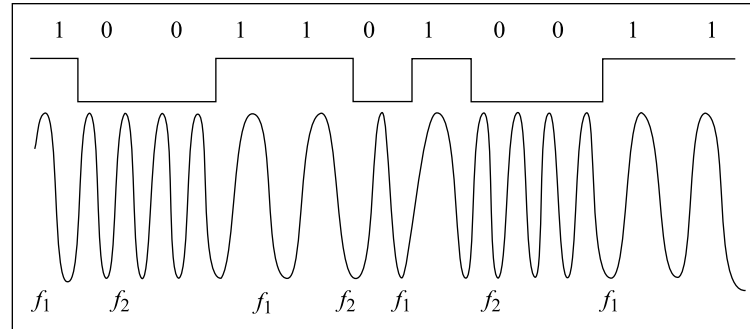


Fig. 2.6 Frequency Shift Keying (FSK)

Phase Shift Keying (PSK)

In Phase Shift Keying (PSK), we keep the amplitude and the frequency of the carrier signal unchanged; only change the phase to denote 0s and 1s. For instance, we can start with a phase of 0 degrees to represent binary 0 and then change the phase to 180 degrees to represent binary 1. In PSK, we change the timing of the carrier wave abruptly to encode data. After a phase shift happens, the carrier wave still continues to oscillate, but it immediately jumps to a new point in its cycle. The phase of the signal during each bit duration is constant and its value depends on whether it is 0 or 1. This is depicted in Fig. 2.7.

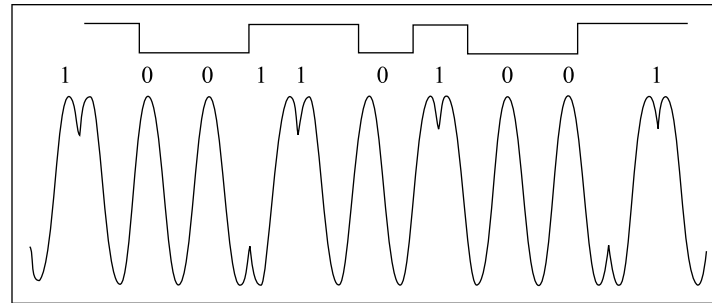


Fig. 2.7 Phase Shift Keying (PSK)

The figure shows that the 0 bit indicates no phase change at all. This is true even if the preceding bit is either 0 or 1. However, a bit 1 represents a phase shift of 180°. Again this is true irrespective of whether the previous bit is 0 or 1.

This technique is more noise resistant. It is also more efficient than frequency shift keying.

Quadrature Amplitude Modulation (QAM)

The main limitation of PSK is the ability of the hardware equipment to distinguish small differences in terms of phase changes. This puts a limitation on its data rate. In ASK, FSK or PSK, we alter only one characteristic (respectively amplitude, frequency, and phase) of the carrier wave. What if we alter two of these together? Since the bandwidth of the transmission medium is a major limitation, we cannot combine FSK with anything else. Therefore, the only possibility we have is to combine ASK and PSK. Thus, we can have x variations in phase and y variations in amplitude. QAM does just that.

QAM makes higher data rates possible. A detailed discussion of QAM is beyond the scope of this text.

Quadrature Phase Key Shifting (QPSK)

Quadrature Phase Key Shifting (QPSK) is a phase key modulation technique that works with four phases, instead of just one phase. By working with four phases, QPSK is able to encode two bits per symbol. Hence, QPSK can effectively double the data rate of simple phase key shifting technique. Of course, for this to be possible, QPSK systems are more complicated in nature.

The four phases in QPSK are at 45, 135, 225, and 315 degrees. The encoding of bits with reference to the phase angles is as shown below.

Phase	Bit encoding (Binary values)
45	00
135	01
225	11
315	10

The main limitation of PSK is the ability of the hardware equipment to distinguish small differences in terms of phase changes. This puts a limitation on its data rate. In ASK, FSK or PSK, we alter only one.

2.4 BAUD RATE AND BITS PER SECOND

Many people confuse between **baud rate** and **bit rate** or **bits per second** (bps). They use these terms synonymously. However, there is a difference between them. The baud rate is the number of times the signal level changes in a channel per second. This signal level could be amplitude, frequency or phase. As we know, physically, the hardware puts limitations on how many maximum times this signal change can happen. For instance, the bandwidth of a transmission medium defines the maximum and minimum frequency that it allows for the carrier wave. It is simply not possible to breach these limits, as the laws of Physics ascertain these limitations. Therefore, we have to think about other mechanisms to transmit more data, keeping the hardware limitations in mind. How can we achieve higher data rates, then?

Normally, we associate one bit (0 or 1) with each change in the signal level (i.e., change in amplitude/frequency/phase). By associating more than one bit for each signal level, one can achieve a higher data rate. That is, the *bit rate* will be higher than the *baud rate* in such a case. All this has to be built into the modem.

Let us first consider the traditional case wherein bit rate = baud rate. For instance, let us consider a frequency shift keying scheme with a base frequency = 1700 Hz. We can now have two frequencies, say 1200 Hz to represent 0 and 2200 Hz to represent 1 (both between 0 and 4000). This is shown in Fig. 2.8. In this scheme, the baud rate and the bit rate are the same, because one signal level (in this case frequency) represents 1 bit. If the frequency changes the bit value also changes.

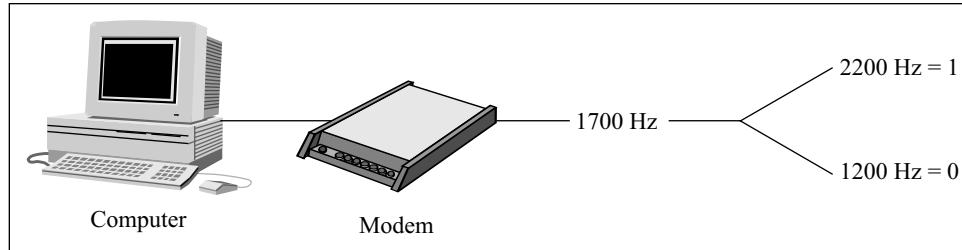


Fig. 2.8 Single bit transmission using FSK

In this case, the bit coming from the computer is examined by the modem, and depending upon whether it is 0 or 1, the carrier signal is modulated at 1200 Hz or 2200 Hz frequency, keeping the amplitude and the phase same as we have seen. Note that as both of these are between 0 and 4000 Hz, a telephone channel can carry them. At the other end, the other modem demodulates it, i.e., reads the frequency of the incoming signal and generates a bit 0 or 1 accordingly, which the computer can store.

Now let us consider Fig. 2.9.

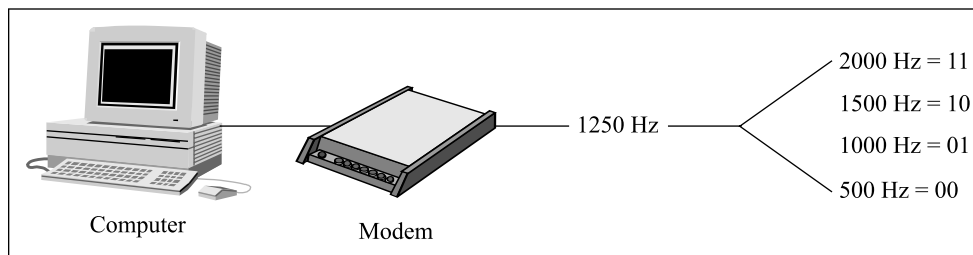


Fig. 2.9 Double bit transmission using FSK

What will happen if we reserve two bits to represent a frequency level? In this case, we can code four frequency levels (all between 0 and 4000 Hz) to represent 00, 01, 10 and 11. In this case, the modem examines the bits coming from the computer, two at a time. It checks the value. It has to be one of 00, 01, 10 and 11. Depending on the value, it generates the signal with the frequency level as shown in the figure. In this case, each signal level (i.e., frequency) denotes two bits. A new signal level would mean a pair of two new bits. Therefore, the data rate in bits/second is twice the rate at which the signal changes (i.e., the baud rate). This is built into the modem. This enables us to increase the data rate on the same line.

Can we not code three bits for each frequency level? Yes, we can. We get eight different values for three bits, viz., 000, 001, 010, 011, 100, 101, 110 and 111. We need to allocate eight frequency levels, one for each of these but all falling in the range of 0 to 4000 Hz. Now, the modem examines three bits at a time and generates the corresponding frequency. At the receiving end, the frequency is measured and the three bits are generated again, and presented to the receiving computer. In this case, the bit or data rate (bps) is three times the baud rate. This further increases the data rate on the same line for the same baud rate (i.e., the rate at which the signal changes).

Can we go on increasing the bit or data rate endlessly? Certainly not! The reason is simple. You will notice that as the data rate increases, the distance between the adjacent frequencies goes on decreasing. For instance, for bit rate = baud rate, it was $2200 \text{ Hz} - 1200 \text{ Hz} = 1000 \text{ Hz}$. For bit rate = $2 \times$ baud rate, it was $2000 \text{ Hz} - 1500 \text{ Hz} = 500 \text{ Hz}$, and so on. As this goes on decreasing, the system becomes more and more error prone, because a slight error in the measurement can result into a wrong conclusion. When an error is detected at the destination, a retransmission has to be requested. This brings down the effective data rate. Thus, a trade-off is required. The bit rate is so chosen that the *effective* data rate is maximum.

2.5 ANALOG SIGNAL, DIGITAL (STORAGE AND) TRANSMISSION

2.5.1 Pulse Code Modulation (PCM) using Sampling and Quantizing

This type of transmission is becoming very popular due to many reasons that we will discuss later. The idea is somehow to represent an analog signal into digital bits and then transmit it as a digital signal. There are several techniques make it possible to achieve this and we have discussed a general overview of the basic idea. **Pulse Code Modulation (PCM)**, among all, is the most popular. We will discuss this with the help of Fig. 2.10.

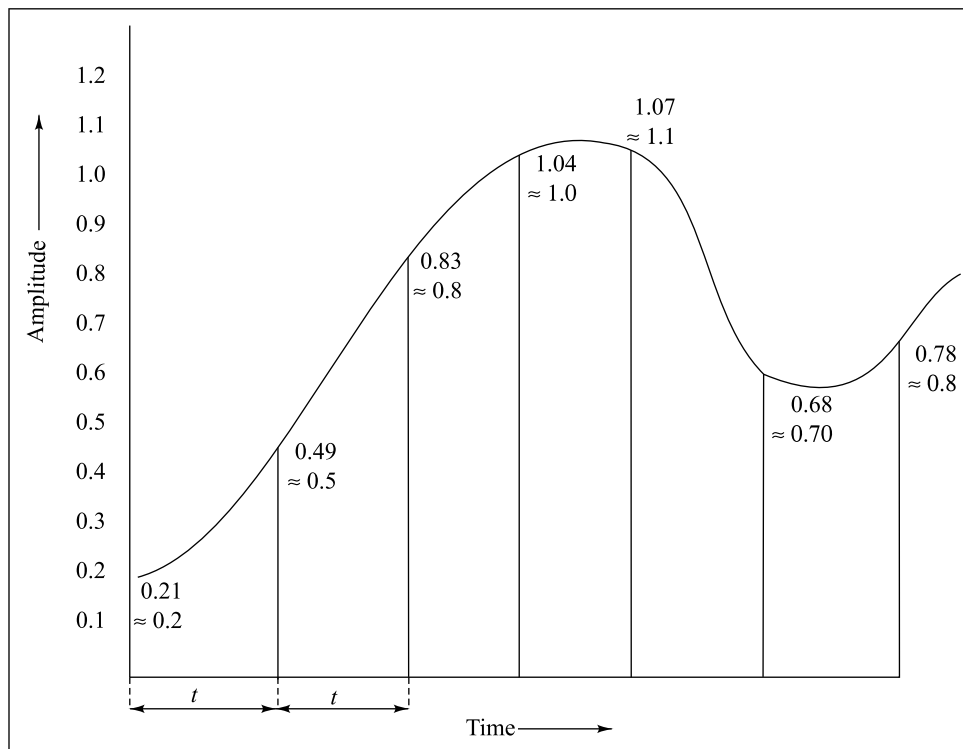


Fig. 2.10 Pulse Code Modulation (PCM)

The basic steps in PCM are as given below:

At source:

1. **Sample** the analog signal at regular interval say t as shown in the figure.
2. Convert the analog signal into some discrete values.
3. Convert these values into binary numbers by assigning a fixed number of bits for each value.
4. Convert the binary numbers as a digital signal by concatenating all these binary numbers.

At destination:

1. Convert the digital signal into binary numbers.
2. Separate out the discrete values of signals by using the number of bits for each discrete value.
3. Reconstruct the original analog signal.

We require an equipment called **codec** (Coder/Decoder) at both the source and the destination to perform these functions. We can also call it **A/D (Analog to Digital) converter and D/A (Digital to Analog) converter**. Let us now study the functionality of such equipment.

Figure 2.10 shows an analog signal. Let us assume that we have an equipment (a codec) to measure its amplitude at some given time interval (shown as t in the figure). This is as good as slicing the analog signal. It is called **sampling**. The discrete values of the amplitude shown are 0.21, 0.49, 0.83, 1.04, 1.07, 0.68 and 0.78. We need to represent these as binary numbers. If we represent these values as they are (i.e., fractions or floating numbers), we will require a large number of bits—thereby increasing the load on the communication system even if we can get accuracy in return. Therefore, we approximate these values to the nearest numbers such as 0.2, 0.5, 0.8, 1.0, 1.1, 0.7 and 0.8 as shown in the figure. This is obviously at the expense of some accuracy. These numbers are still fractions. We still have to use the floating-point notation to represent these, requiring a huge number of bits for each value. Therefore, we use a trick here. We multiply each value by 10 to get the numbers 2, 5, 8, 10, 11, 7 and 8. This whole process is called **quantization**. We now can convert them into binary numbers as 0010, 0101, 1000, 1010, 1011, 0111 and 1000 again. Notice that we have chosen four bits to represent each value, because the maximum value to be represented is 11, which will require four bits. We can now send these bits as a long bit stream as given below:

0010010110001010101101111000

This bit stream travels as it is if the line is digital or can travel as analog signals using modem if we send it through the normal telephone line. However, we will assume that at the other end, the bits are converted back into the original form again.

At the destination, the equipment at that end (another codec) can now split the received bit stream into chunks of four bits each, find out its decimal value (i.e., 2, 5, 8, 10, 11, 7 and 8) to get the values 0.2, 0.5, 0.8, 1.0, 1.1, 0.7 and 0.8 by dividing these by 10, and generate an analog signal with those specific values at the time interval t . Therefore, at destination, the analog signal generated would have values 0.2, 0.5, 0.8, 1.0, 1.1, 0.7 and 0.8 at times $0t$, $1t$, $2t$, $3t$, $4t$, and $5t$. Notice that this signal is slightly different than the original signal due to the approximation, which we carried out at the source (e.g., 0.21 was approximated to 0.2). This difference is known as **quantization noise** or **quantization error**. However, in the whole process, we have saved a lot in the number of bits that we needed to send. Thus, there is always a trade-off between accuracy and cost or speed.

The aim of any good PCM strategy would be to reduce the quantization noise to a negligible level without increasing the load on the network significantly. The current PCM standard assumes eight bits/sample.

There are two variations of the basic PCM method: **Adaptive Differential Pulse Code Modulation (ADPCM)** and **Delta Modulation**. Let us discuss these in brief.

1. **Adaptive Differential Pulse Code Modulation (ADPCM)** – In this method, rather than outputting the absolute value of the digitized amplitude. After the first reading is sent in totality, only the difference between the current amplitude value and the previous amplitude value is outputted. Because sudden jumps of ± 16 bits are unlikely; five bits, rather than seven suffice in case of ADPCM.
2. **Delta Modulation** – In this approach, only one bit is reserved for the difference between two successive readings. But the signal is sliced very rapidly to have a number of samples. The number of samples is so large that two successive readings hardly change. If the next reading is more than the previous one, a bit 1 is output; if it is less, a bit 0 is output. Then, after the first reading in totality, one only needs to mention a series of zeroes and ones to represent the shape of the signal as shown in the figure. The problem with this scheme is when the signal changes too fast as shown in Fig. 2.11 or if it is just a horizontal line.

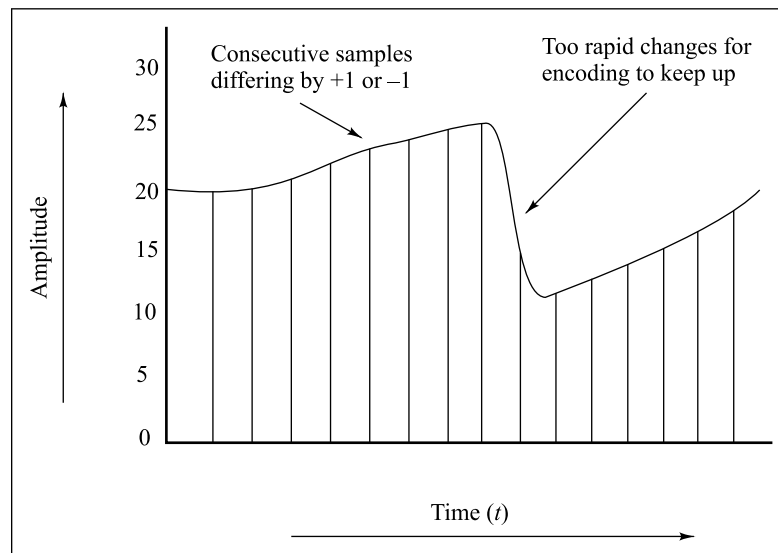


Fig 2.11 *Delta modulation*

2.6 NYQUIST THEOREM

An interesting question is how do we choose the time interval for sampling, or slicing the analog signal? If we sample too fast (i.e., time t in Fig. 2.10 is very low), the equipment at both ends have to be capable of handling that high a speed of sampling and reconstructing the analog signal. Also, in this case, the number of bits that will be needed to be sent for the same signal will be very high, putting a very high load on the whole communications system (unless at the other extreme, we

use delta modulation where the sample size is only of one bit, but the sampling rate is very high). However, at higher speeds of sampling, the signal is more likely to be reproduced faithfully than if the speeds are low. This is obvious. At higher sampling speeds, it is less likely to miss the ups and downs than at lower speeds.

Figure 2.12 shows two signals. Part (a) shows a signal with very low frequency, which changes very slowly and smoothly. Part (b) shows a signal with high frequency, which changes rapidly. The figure shows the sampling of both the signals at time interval t . At low frequency, the sampling at that speed is good enough. However, at higher frequency (Part (b)), the sampling rate is low. It misses many ups and downs. For instance, refer to points x, y, z in the figure. If we know the reading at points x and y , there are a number of ways in which these points could be connected. Only one of them is shown in Fig. 2.12. Therefore, reconstruction of the signal would be quite erroneous. Consequently, for a signal with higher frequencies, a higher sampling rate is required.

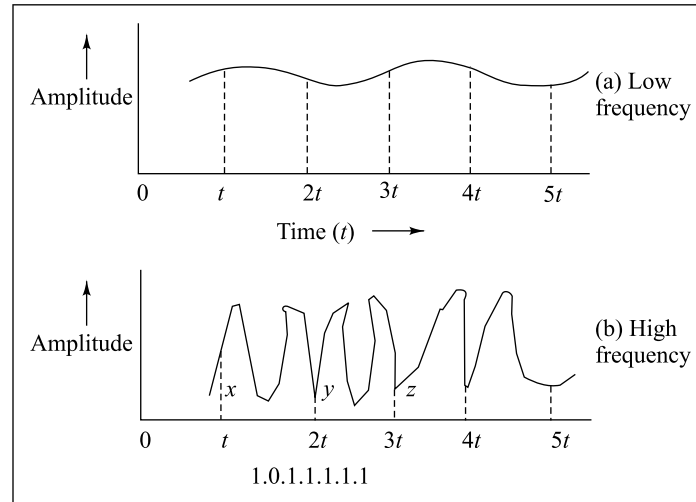


Fig. 2.12 Sampling speeds for different frequencies

In fact, the sampling speed is quite related to the highest frequency in a signal. If we go back to the Fourier analysis of a signal, we know that any analog signal can be shown to be consisting of a number of sinusoidal signals, leading to the concept of bandwidth of a signal. Nyquist showed that the sampling speed should be $2 * f_{\max}$ where f_{\max} represents the highest frequency in that signal resulting out of Fourier analysis. This is called **Nyquist theorem**. Thus, if we want to sample telephone voice with a maximum frequency of 4000 Hz, we must have a sampling rate of 8000 samples per second. This is exactly what is used in PCM standard.

This theorem proves that to reproduce an analog signal into its equivalent digital form using a modulation technique such as PCM, the sampling rate must be at least twice the highest frequency of the original signal.

Thus, for example, if we want to send an analog signal whose highest frequency is 100 Hz as a digital signal, we must sample it at least 200 times per second for no loss of information.

The video signals have a far higher bandwidth with very high frequencies as compared to voice signals. It is for this reason that the sampling for digitizing video signals has to be done at a far higher rate than for voice signals. This leads to a demand for a far higher data rate and volume for video signals than voice signals. This is obvious from the data contents and rates of VCD and DVD player (for Video Disk) as compared to that of a CD (Compact Disk) for music.

Human voice has various frequency components in the range of 0 to 20000 Hz. However, out of this range, frequency range of only 300 to 3300 Hz is sufficient to recognize a voice in a telephone conversation. The telephone company normally provides for 0 to 4000 Hz as shown in Fig. 2.13.

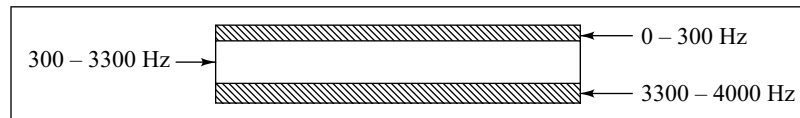


Fig. 2.13 Guard bands

As the figure shows, the frequencies (shown shaded) 0 to 300 Hz and 3300 to 4000 Hz act as **guard bands**, so that *multiplexing* of many signals in a single wire is possible. Therefore, the maximum frequency in this case is 4000 Hz.

In other words, to be able to understand and decipher human voice, all its signal components between 0 and 4000 Hz must be transmitted. Using Nyquist theorem, we can conclude that to transmit human voice over digital telephone lines, we must have a sampling rate of $4000 \times 2 = 8000$ samples per second. Moreover, if each sample consists of 8 bits, we can have the following equation for the bandwidth required for telephone lines to carry digitized human voice:

$$\begin{aligned}
 &\text{Highest frequency of human voice} \times 2 \times \text{Number of bits in each such sample} \\
 &= 4000 \times 2 \times 8 \\
 &= 64,000 \text{ bits per second} \\
 &= 64 \text{ Kbps}
 \end{aligned}$$

Thus, to carry human voice frequencies as sent in the telephone channel, we need telephone lines having a bandwidth of 64 Kbps.

SUMMARY

Any signal can be either analog or digital. Further, it can be transmitted as an analog or digital signal. Therefore, we have four combinations, viz., (a) Analog signal, analog transmission (b) Digital signal, digital transmission, (c) Digital signal, analog transmission and (d) Analog signal, digital transmission. That is, a signal can be transmitted as it is, or can be transformed into the other form (from analog to digital or vice versa) before it is transmitted.

Normal telephone is an example of analog signal being transmitted as analog. Human voice is an analog sound, which can be sent as an analog electrical signal over telephone wires. When computer data is sent over telephone wires as digital pulses, it is a case for digital transmission of digital data. Here, repeaters are used for reproducing the signal at periodic distances, so that it does not become distorted beyond repair until it reaches the destination. Creating digital lines all over the place was a very expensive and time-consuming process. Therefore, engineers wanted to use the existing analog telephone wires to send the digital computer data.

For transmitting digital data as analog signals, modems are used. A modem transforms a digital signal into its analog equivalent at the sender's end in a process called modulation. At the receiver's end, it does the opposite, i.e., it transforms the analog signal back into its original digital form in a process called demodulation. Modulation and demodulation of a signal can be done in one of the three basic ways, viz., Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK) or Phase Shift Keying (PSK). The three techniques respectively modify the amplitude, frequency and phase of the carrier signal to suit it to the data being carried to achieve modulation. Various combinations of these can also be used.

For transmitting analog data as digital pulses, a technique called Pulse Code Modulation (PCM) is used. Here, the analog signal is *sampled* and *measured* at predetermined intervals. For this, the techniques of sampling and quantizing are used. A device called codec is used for this purpose. ADPCM and Delta Modulation are variations of PCM.

Nyquist theorem proves that for transmitting any analog signal into its equivalent digital form, it must be sampled at a rate, which is at least twice its highest frequency component. Consequently, telephone companies use 64 Kbps lines for transmitting digitized voice, which has the highest frequency of 4000 Hz, with each sample consisting of eight bits at 8000 samples/second.

KEY TERMS AND CONCEPTS

Adaptive Differential Pulse Code Modulation (ADPCM)	Frequency Modulation (FM)
Analog-to-Digital Converter (A-D)	Frequency Shift Keying (FSK)
Amplifier	Guard bands
Amplitude Modulation (AM)	Modem
Amplitude Shift Keying (ASK)	Modulator
Attenuation	Nyquist theorem
Baud rate	Phase Shift Keying (PSK)
CODEC	Pulse Code Modulation (PCM)
Delta Modulation (DM)	Quadrature Amplitude Modulation (QAM)
Demodulator	Quadrature Phase Key Shifting (QPSK)
Digital line	Quantizing errors
Digital-to-Analog Converter (D-A)	Repeater
	T1 line

QUESTIONS

True/False

1. An analog signal can be transmitted as a digital signal by encoding the signal by certain methods.
2. A digital signal cannot be transmitted as it is, instead it must be encoded as an analog signal before transmission.
3. The term modem is derived from a single component, i.e., modulator.
4. A modem cannot be provided as an interface in a desktop computer itself.

5. Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK) and Phase Shift Keying (PSK) are all modulating techniques.
6. Bits Per Second is the number of times the signal level changes in a channel per second.
7. A codec is required at both the ends (the source and the destination) to perform *analog to digital* and *digital to analog* conversion.
8. Nyquist theorem suggests that to transmit an analog signal into its digital equivalent, it must be sampled 1000 times a second.

Multiple-Choice Questions

1. An amplifier amplifies _____.
 (a) only the original signal (b) the original signal and noise
 (c) only the noise (d) the original signal but not noise
2. Amplifiers are used in _____ transmission of _____ signals.
 (a) analog, digital (b) digital, analog
 (c) digital, digital (d) analog, analog
3. Repeaters are used in _____ transmission of _____ signals.
 (a) analog, digital (b) digital, analog
 (c) digital, digital (d) analog, analog
4. A _____ detects zeroes and ones and regenerates them.
 (a) amplifier (b) repeater
 (c) multiplexer (d) modem
5. A line containing repeaters is called a _____.
 (a) analog line (b) digital line
 (c) combined line (d) None of the above
6. A _____ is used to send digital data over analog telephone lines.
 (a) multiplexer (b) repeater
 (c) amplifier (d) modem
7. Modulation of a digital signal can be done by varying either _____ or _____ or _____ of a signal.
 (a) frequency, amplitude, phase (b) phase, amplitude, frequency
 (c) frequency, phase, amplitude (d) phase, period, amplitude
8. _____ is highly affected by noise.
 (a) FSK (b) ASK
 (c) PSK (d) Modulation
9. The _____ technique combines the features of ASK and FSK.
 (a) QAM (b) PCM
 (c) ASK (d) FSK
10. Bit rate can be _____ the baud rate.
 (a) smaller than (b) greater than or equal to
 (c) smaller or greater than (d) None of the above
11. The reason why we cannot go on increasing the bit rate infinitely is that the distance between the adjacent frequencies goes on _____.
 (a) increasing (b) being constant
 (c) becoming arbitrary (d) decreasing

12. The mechanism for transmitting analog signals in the digital form is _____.
(a) ASK (b) FSK
(c) PSK (d) PCM
13. In _____, the difference between the current amplitude and the previous amplitude is shown.
(a) PCM (b) ADPCM
(c) delta modulation (d) None of the above
14. Nyquist theorem says that the sampling rate required to transmit a signal having bandwidth 4000 Hz is at least _____ samples per second.
(a) 4000 (b) 8000
(c) 2000 (d) 16000
15. For sending telephone conversations across using PCM, we need a bandwidth of _____ Kbps with 8-bit samples.
(a) 54 (b) 64
(c) 56 (d) 48

Detailed Questions

1. Explain analog signal, analog transmission.
2. Explain the term repeater with respect to digital signal, digital transmission.
3. What is a modem? How is it useful for digital signal, analog transmission?
4. What is Amplitude Shift Keying (ASK)?
5. What is Frequency Shift Keying (FSK)?
6. Explain Phase Shift Keying (PSK).
7. Discuss the difference between the terms *baud rate* and *bits per second*.
8. Explain the use of Pulse Code Modulation (PCM) in analog signal, digital transmission.
9. How is Delta Modulation different from PCM?
10. What does the Nyquist theorem say about the transmission rate of digitized signals, which are originally in the analog form?

3 Modes of Data Transmission and Multiplexing

3.0 INTRODUCTION

Digital data can be transmitted in a number of ways from the source to the destination. These modes of data transmission can be outlined as follows:

- Parallel and Serial Communication
- Asynchronous, Synchronous and Isochronous Communication
- Simplex, Half-duplex and Full-duplex Communication

Let us discuss these techniques in this chapter. We follow this discussion with the concept of multiplexing. Multiplexing allows more than one signal to be sent on a single transmission path using techniques that help a more effective use of the transmission medium than is possible otherwise.

3.1 PARALLEL AND SERIAL COMMUNICATION

Let us imagine that we want to send digital data stored as bytes of 8 bits each, and in words of may be 16 or 32 bits, depending upon the word length.

3.1.1 Parallel Communication

In **parallel communication**, we transfer a word or a byte at a time. Therefore, we need to have those many wires parallel to each other, each carrying a single bit. This is shown in Fig. 3.1, where eight wires are transmitting a byte at a time. In this example, the byte contains 10011001.

This is a very fast method of transmitting data from one place to another. However, there are problems in this scheme. Firstly, it is a very expensive method because it requires several wires as well as various sending and receiving equipment. In addition, it demands extraordinary accuracy, which cannot be guaranteed over long distances. Digital pulses may not traverse at the same speed,

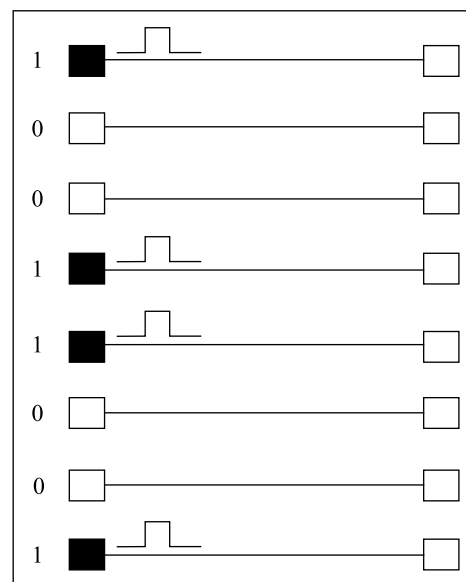


Fig. 3.1 Parallel transmission

because the underlying media through which the data passes from one computer system to another may not be absolutely or exactly identical. This gives rise to the problem of **skew** as shown in Fig. 3.2.

When the skew happens, the bits 10011001 are sent from the source to the destination, but they traverse at different speeds. At the destination, the measurements of signal values to determine whether it was a bit 0 or 1 have to be done at the same time for all the bits. Therefore, the problem of skew can result in an inaccurate interpretation of bits.

To avoid this problem, parallel transmission is used only for a very short distance and there too, all parallel wires have to be absolutely identical. This method is used for data transmission within the computer system such as from the CPU registers to the memory or vice versa through the data bus. The data bus essentially implements the parallel transmission of data. This is done because speed, rather than cost, is of paramount importance in this case. However, due to the problems outlined above, one cannot use this method over long distances.

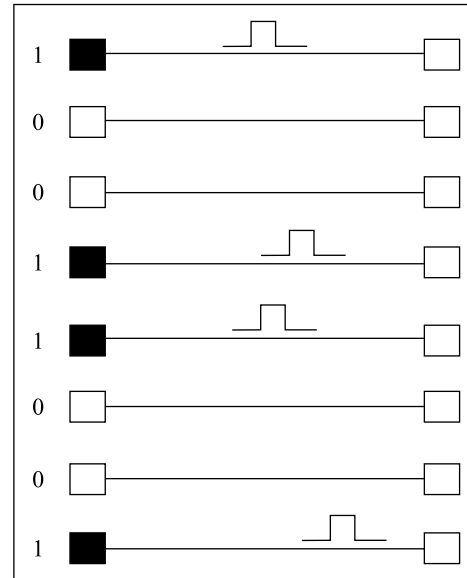


Fig. 3.2 The problem of skew

3.1.2 Serial Communication

Over long distances, **serial communication** is used. While sending the data serially, characters or bytes have to be separated out and sent bit by bit. Thus, there is some hardware equipment involved in converting the data from parallel to serial. At the destination, the measurement of signal values is done in the middle of the bit durations as shown in Fig. 3.3. This is because, if the values are taken at the edge or a point where the bit value changes (shown as point A in the figure), the reading will be indeterminate.

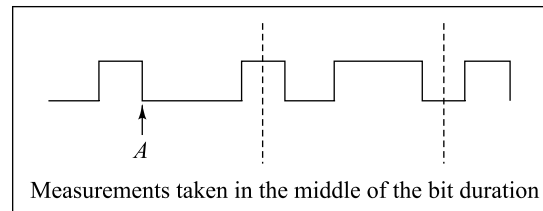


Fig. 3.3 Serial transmission

At the destination, all the bits are collected, measured and put together as bytes in the memory of the destination. This requires conversion from serial to parallel.

In serial data transmission, we have to identify where the character starts. We have to also identify the middle position of each bit interval so that measurement can be taken. Normally, the transmitter and the receiver have two different clocks. The point is to synchronize the clock of the receiver exactly with that of the transmitter, so that correct readings result and the bit values are understood correctly. This is essentially a problem of synchronization. We can illustrate this through an example.

Let us say that we have to send 8 bits b_0 to b_7 from point X to point Y as shown in Fig. 3.4. We have arranged the bits in a circular fashion to illustrate the point better. Let us say that arms A_1 and A_2 at point X and Y rotate in clockwise directions. When both A_1 and A_2 point to the bit b_0 at X and Y respectively, the connection is made and the bit b_0 is transferred from X to Y . Now due to rotation, after some time, A_1 will point towards b_1 at point X . At that time A_2 also has to point to b_1 at point Y , otherwise the bit will not be transferred correctly. The same is true for the transfer of b_2 through b_7 as A_1 and A_2 rotate clockwise. Therefore, the two arms shown as A_1 and A_2 have to be perfectly synchronized and they have to rotate at the same speed in order to succeed in sending and receiving all the bits accurately.

In order to achieve this, initially, we have to adjust and synchronize the clocks in the electronics of the source and the destination. There are three approaches in which this can be achieved. These are asynchronous, synchronous and isochronous transmission. We will look at them very shortly.

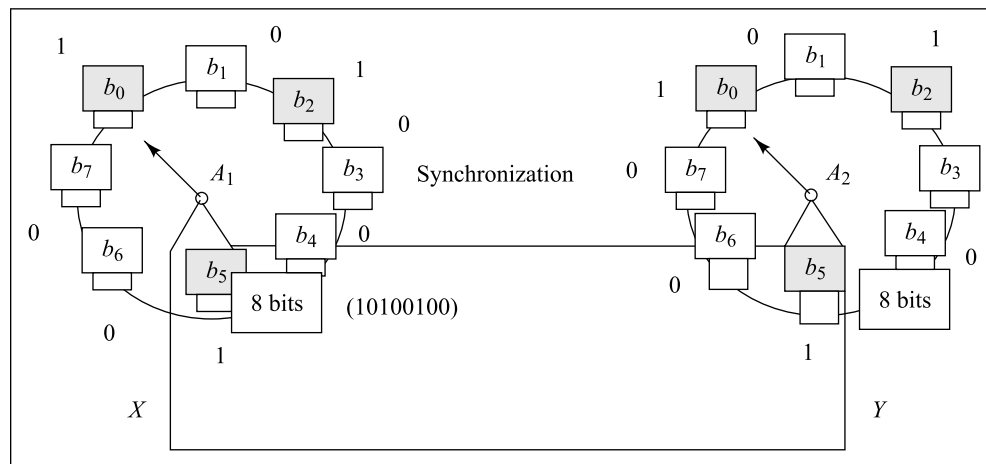


Fig. 3.4 Serial transmission and synchronization

A line adapter or interface generally achieves this task of synchronization. It achieves a few other things also. We will call it only 'serial interface' in this text.

Electronics devices called **Universal Asynchronous Receiver Transmitter (UART)** and **Universal Synchronous Asynchronous Receiver Transmitter (USART)** are examples of such an interface. USART can be used for both the asynchronous and synchronous transmission. On the other hand, UART is used only when the transmission is asynchronous.

3.2 ASYNCHRONOUS, SYNCHRONOUS AND ISOCHRONOUS COMMUNICATION

3.2.1 Asynchronous Communication

We have learnt why serial communication is preferred to parallel communication over a long distance. However, we have only one problem in serial communication, i.e., how do we identify individual bits? Essentially, the problem is that of synchronizing the sender (source) and the receiver (destination) as outlined above.

In **asynchronous communication**, the time when a character will be sent cannot be predicted. Imagine, for instance, a data entry operator. S/he may key in a few characters, then may consult her/his boss, then key in a few more characters before s/he goes off for lunch. In such a scenario, where the data is transmitted as characters and at an unpredictable pace, we have to synchronize the source and destination for each character. This is called asynchronous communication.

In this case, each character is preceded with a start bit and succeeded with 1, 1.5 or 2 stop bits. Figure 3.5 depicts this. Normally NRZ-L signaling is used for asynchronous transmission. According to this convention, a negative voltage denotes a binary 1 bit and a positive voltage denotes a binary 0 bit. When the line is idle, i.e., when no character is being sent over the line, a constant negative voltage signifying a binary '1' is generated. When the character is to be sent, first, a bit, which is bit 0 is sent. This is called **start bit**. A positive voltage is generated according to the convention to denote this. All the 5 to 8 bits of the character according to the coding scheme used (Baudot, BCD, ASCII, EBCDIC) then follow; each of which could have a 0 or 1 value. A **parity bit**, if used, then follows. In the end, 1, 1.5 or 2 **stop bits** are added. The stop bit again corresponds to the *idle* state, which is bit 1. In this context 1.5 bits only means that the signal with negative voltage denoting 1 is generated for 1.5 times the normal bit interval. The sender and receiver have to exactly agree on the number of bits that represent a character (5, 6, 7 or 8); the parity bit scheme, if used; the bit interval (or bit rate); and also what represents 0 and 1 (NRZ-L).

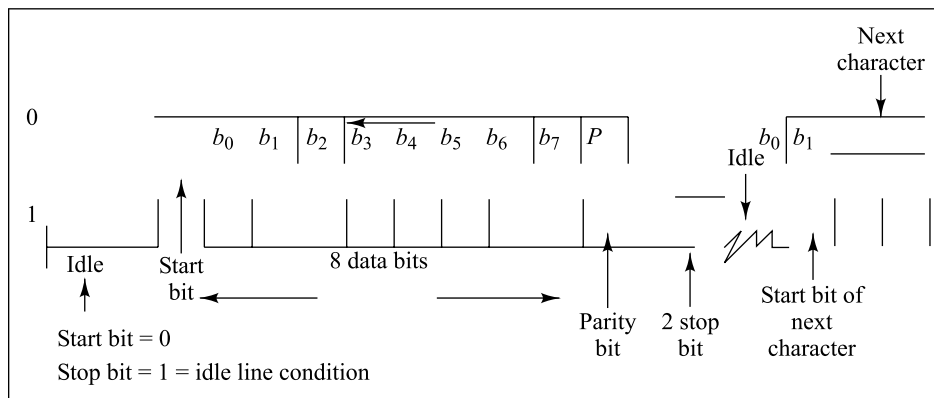


Fig. 3.5 Asynchronous communication

Essentially, when the start bit is recognized by the destination, it *knows* that a character is being sent. It then adjusts and starts its clock so that measurement of signal values can now start in the middle of every bit that follows for the expected number of bits. After all the bits in the character including the parity bit are measured and stored, it expects a stop bit, which is encountered if everything is OK. This is to recognize the bits being sent. This continues until the stop bit is received. At this time, the line becomes idle, and the destination clock also goes to sleep. On receiving the start bit, it *wakes up* for the duration of that character, and after the full character is received, it again goes to sleep until the next character arrives. This makes sense where (a) the rate at which the characters arrive is unpredictable and (b) the speed at which the characters are keyed in is far less than the speed with which they are transmitted or processed.

There has to be a proper understanding between the source and destination about the bit rate in *bps*, from which, bit interval can be computed to determine when the signal measurements have to

be taken. Using this, the destination waits for the start bit and then starts its clock to measure the signal values to recognize the bits including the parity bits.

3.2.2 Synchronous Communication

In synchronous transmission, the whole block of data bits is transferred at once, instead of one character at a time. The block of bits may or may not consist of different characters. It could as well be a digitized image. It could be anything. In this case, we need a bit oriented protocol between the sender and the receiver. If the block of bits consists of different characters, the receiver needs to know the beginning of the first bit of the first character. It then can start its clock and sampling to determine the bit values. Knowing the number of bits per character, it can now recognize the character and store them at the destination. This is shown in Fig. 3.6. In this case, we need to have a byte oriented protocol between the sender and the receiver.

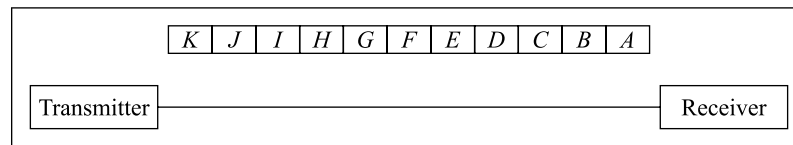


Fig. 3.6 Synchronous communication

The point is how does the receiver know the first bit of the first character, or when to start the clock for sampling? In order to perform this synchronization, each data block is preceded with a unique synchronizing bit pattern. We use the **SYN** (abbreviation for *synchronize*) transmission control character for this. It has a bit pattern of 00101101. This bit pattern is generally not used in any normal communication. Therefore, this SYN character can be reserved for indicating the start of the block, i.e., for synchronization.

However, there is a problem in this scheme. The transmitter will not send a SYN character as a part of data characters. However, by mistake, the bit pattern of two characters could be such that if sent one after the other, they can constitute a SYN character, thereby fooling the receiver. Figure 3.7 illustrates this case. In this case, if ASCII character *b* and *a* are sent one after the other, we get a bit combination of 4 bits from the ASCII character *b* and 3 bits for ASCII character *a* to constitute the SYN character.

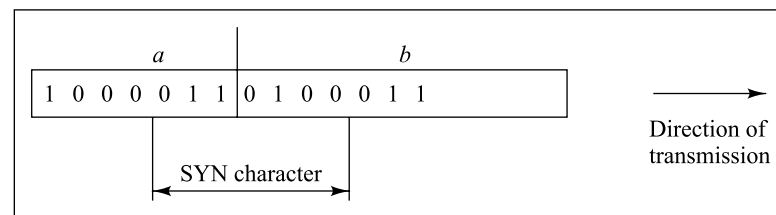


Fig. 3.7 A misleading SYN character

At the receiving end, if the receiver is all the time looking for SYN characters, it can get fooled. It is for this reason that normally two SYN bytes are sent consecutively. The bit combination of two SYN bytes, i.e., 00101100010110 cannot be obtained by concatenating any characters. Therefore, the receiver is asked to *look for two SYN characters*. If not found, s/he is asked to throw them off. In

some systems even three or four SYN characters are sent to make sure that proper synchronization takes place. The receiver gets ready after receiving the required SYN character(s) and then starts its clock for measuring the bit values for the actual data. This is shown in Fig. 3.8.

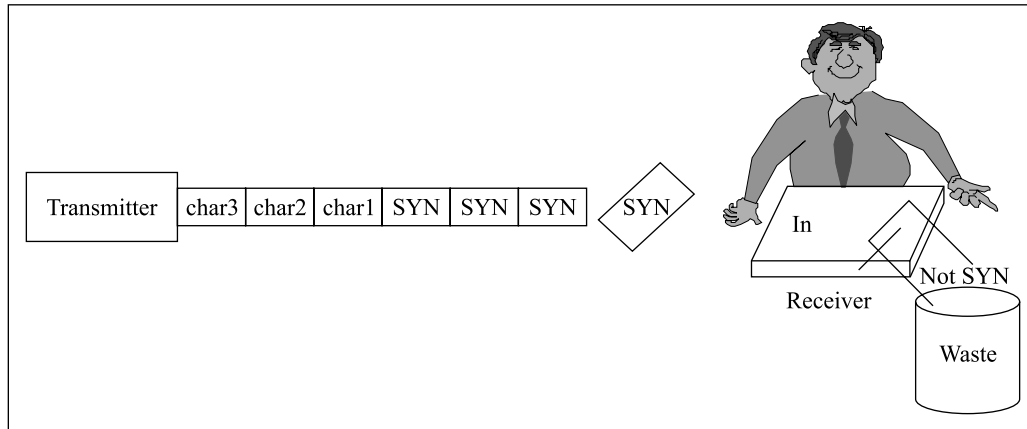


Fig. 3.8 Discarding non-SYN characters until SYN is received for synchronization

When the clock starts measuring the bit values, the counter within a receiver is incremented for every bit received and measured and pushed into the *character assembler* within the receiver. This is as shown in Fig. 3.9.

After a character is assembled, the character is moved to a separate buffer and the bit counter is set to 0 to prepare for the next character. In the case of synchronous transmission, the equipment used at both the ends is called *Universal Synchronous/Asynchronous Receiver Transmitter (USART)*. This is responsible for converting the data from parallel to serial at the transmitter's end and from serial to parallel at the receiver's end. It is also responsible for generating the SYN characters at the beginning of the data blocks, before transmission and recognizing them at the receiving end as shown in Fig. 3.9.

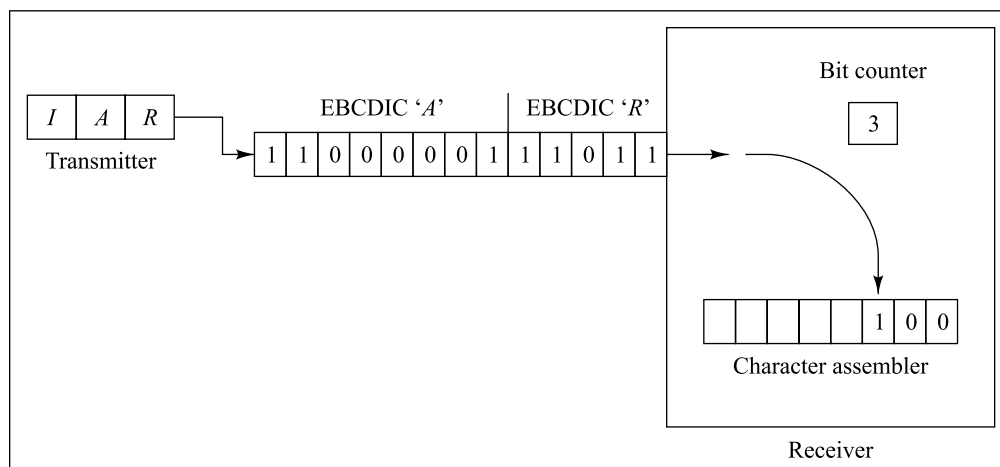


Fig. 3.9 Characters received

Synchronous communication is used when a large amount of data is to be sent from one place to the other. Synchronous transmission has obviously a much higher efficiency because it is a continuous transmission, without any start/stop bits. For example, if we have to transmit a file of 100,000 characters at a rate of 9.6 Kbps, the overheads for asynchronous and synchronous communication will be as follows:

1. Asynchronous Communication

Let us assume that for each character of 8 bits, 1 start and 1 stop bit are sent. Thus, at 2 bits/character as overhead, we will have to send an overhead of (extra bits) of

- $2 \text{ bits/character} * 100,000 \text{ characters} = 200,000 \text{ bits}$
- At 9600 bits/seconds, it will take $200,000 / 9600 = 20.83$ seconds more for the overhead bits

2. Synchronous Communication

We assume that the file of 100,000 characters is broken into blocks of 1200 characters = 9600 bits. We will assume that 48 overhead bits are sent along with each block. (For SYN (synchronize) STX (start of transmission), ETX (end of transmission) and other characters). The overhead is now computed as follows:

- The number of blocks $= 100,000 / 1200 = 250 / 3$
- Numbers of overhead bits $= 250 / 3 * 48 = 4000 \text{ bits}$
- The time taken to send overhead bits at 9600 bps;
 $= 4000 / 9600 \text{ seconds} = 0.4167 \text{ seconds}$

Thus, the time taken for asynchronous communication is much higher, and in this case, 50 times!

3.2.3 Isochronous Communication

This method combines various approaches of asynchronous and synchronous communications. In this method, as in asynchronous method, each character has both start and stop bits. However in isochronous method, the idle period between the two characters cannot be random. For instance, all *idle* periods of no transmission consist of exact multiple of 1 character time interval. Therefore, if the time to transmit a character (including its parity, start, stop bits) is t , the time interval between characters cannot be random as in the asynchronous method. It is also not 0 as in the synchronous method. It has to be $t, 2t, 3t \dots nt$ where n is a positive integer in isochronous communication. Figure 3.10 depicts the differences between the three approaches.

The main reason for using isochronous method over asynchronous method is speed. In practice, asynchronous transmission is limited to a data rate of 2,400 bits per second as per the timing precision of the transmitting and receiving modems. By contrast, isochronous transmission can achieve data rates of up to 19,200 bits per second.

3.3 SIMPLEX, HALF-DUPLEX AND FULL-DUPLEX COMMUNICATION

This classification of data transmission is based on which of the communicating devices can send data, and at what point of time. There are basically three ways in which this can be done, i.e., **Simplex**, **Half-duplex** and **Full-duplex**. These are very simple to understand, and are explained below.

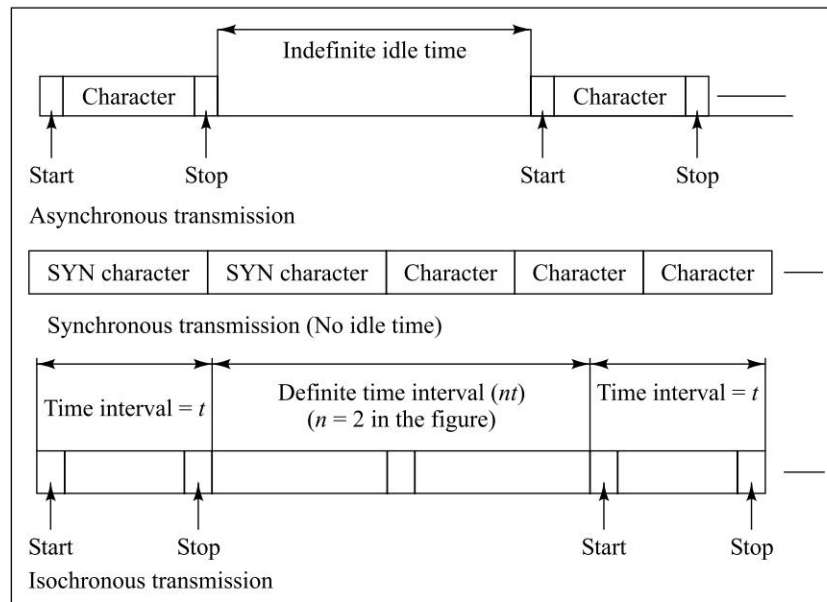


Fig. 3.10 Asynchronous, synchronous and isochronous transmissions

3.3.1 Simplex Communication

In **simplex mode**, the communication is unidirectional only. This is similar to a one-way street, where vehicles are allowed to drive only in a single direction, as shown in Fig. 3.11.

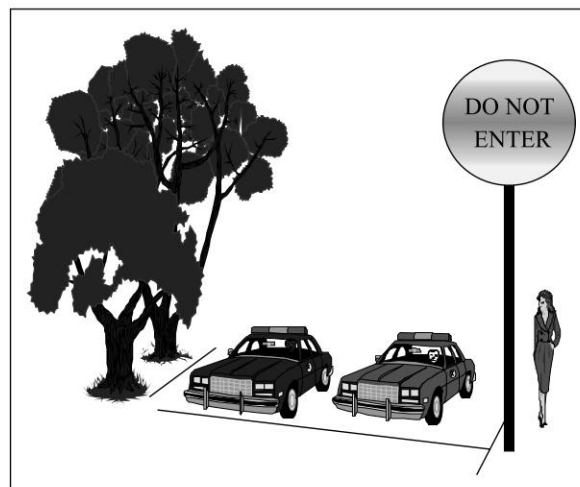


Fig. 3.11 A case of simplex communication

Here, one of the communicating devices can only send data, whereas the other can only receive it like in a radio or a TV nowadays. Keyboard to computer monitor data transmission is a simple

example of this. Another example is shown in Fig. 3.12, where one host can only send data, whereas the other can only receive it.

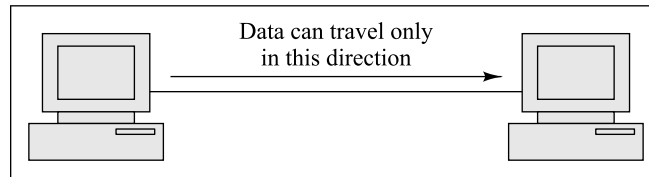


Fig. 3.12 Simplex communication

3.3.2 Half-duplex Communication

Unlike what happens in the simplex mode, in case of **half-duplex mode**, both devices can transmit; however, not at the same time. When one device is sending data, the other must only receive it, and vice versa. This is conceptually similar to a street that has a single lane for vehicle traffic. When vehicles from one side are coming, the vehicles from the other side must wait. This is shown in Fig. 3.13.

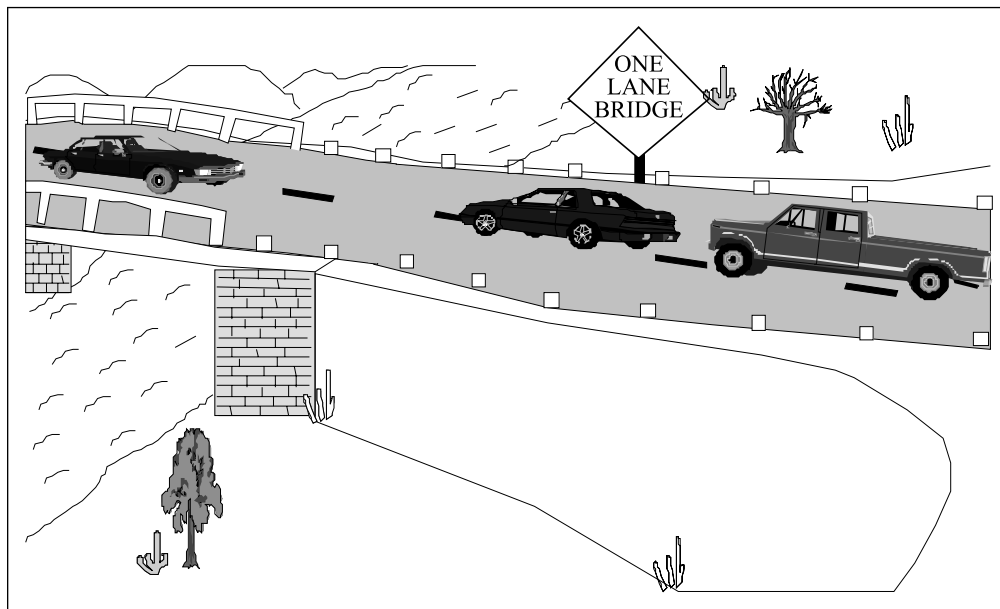


Fig. 3.13 A case of half-duplex communication

Thus, both sides take turns to send data in the case of half-duplex transmission, as shown in Fig. 3.14. This requires a definite *turnaround* time during which the device changes from the receiving mode to the transmitting mode. Due to this delay, half-duplex communication is slower. However, it is more convenient than simplex, as both the devices can send and receive data.

Half-duplex is normally implemented by using a two-wire circuit (1 for data, 1 for ground). In this case, the full bandwidth of the wire is used while sending the data in either direction. Examples of half-duplex communication are conversations over a walkie-talkie.

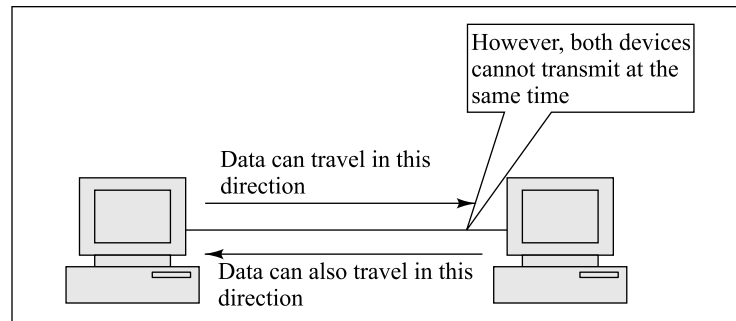


Fig. 3.14 *Half-duplex communication*

3.3.3 Full-duplex Communication

In **full-duplex** (or simply duplex) **communication mode**, both the devices can transmit data at the same time. It means that both devices are capable of sending as well as receiving data at the same time. This is like a two-way street with traffic flowing in both directions at the same time. This is shown in Fig. 3.15. It is also similar to a telephone conversation, where both parties can talk to each other simultaneously.

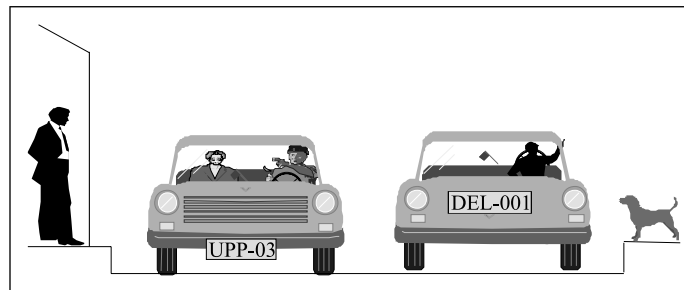


Fig. 3.15 *A case of full-duplex communication*

This can be done using a two-wire circuit or a four-wire circuit. In a two-wire circuit, one wire is used for data and one for ground as in half-duplex. However, the bandwidth of the wire for data is divided in two channels for carrying data in either direction. Thus, each channel can use only half the bandwidth normally. This is shown in Fig. 3.16.

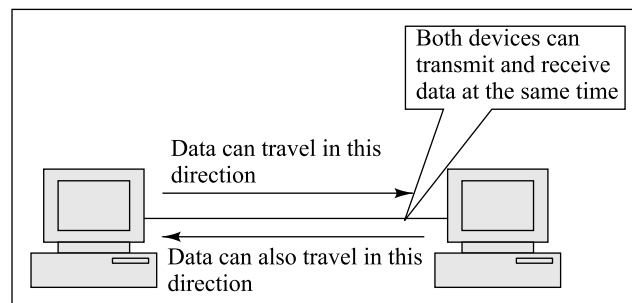


Fig. 3.16 *Full-duplex communication*

In a four-wire circuit, there are two wires for data and two for ground. Thus, we can have one data wire for transmission in each direction; increasing the bandwidth and therefore the data rate.

3.4 MULTIPLEXING AND DEMULTIPLEXING

Multiplexing divides the physical line or a medium into logical segments called *channels*. In multiplexing, different channels carry data simultaneously over the same physical medium. Hardware equipment called **multiplexer** (or **mux** in short) combines (or multiplexes) the inputs from different sources, and loads them on different channels of a medium. The combined data traverses over the medium simultaneously. At the destination, a demultiplexer (also called mux) separates (or demultiplexes) the signals meant for different destinations. The demultiplexer sends these separated signals appropriately to the different destinations. This is depicted in Fig. 3.17. This is obviously cheaper than having three separate lines.

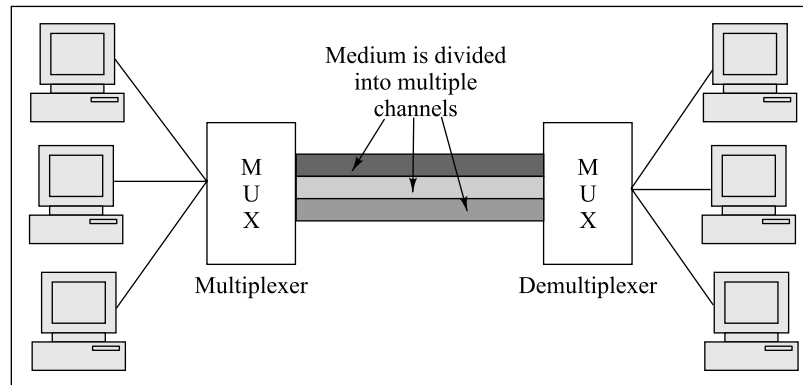


Fig. 3.17 Multiplexing and Demultiplexing

Thus, the mux is responsible for both multiplexing and **demultiplexing**. Therefore, it has to be present at both the ends of a medium. Multiplexing allows us to add new channels in the same existing telephone line without having to install a new line. However, as the total capacity of the medium is limited, the capacity of each channel reduces when we increase the number of channels, as we have studied before.

3.5 TYPES OF MULTIPLEXING

There are basically two ways in which multiplexing and demultiplexing can be achieved. They are **Frequency Division Multiplexing (FDM)** and **Time Division Multiplexing (TDM)**.

3.5.1 Frequency Division Multiplexing (FDM)

Introduction

We are familiar with this technique used in public telephones. **Frequency Division Multiplexing (FDM)** is also used in cable TV systems, where a single cable carries multiple video signals from different channels or stations up to your TV set. With the remote control, we essentially activate the electronic circuits in the television to select a specific frequency band or a channel. This is how

we see a specific program on TV. The signals from other programs also actually traverse to your TV set on the same cable, but they *lie idle* at your TV set for you to choose from.

In FDM, the medium is divided into a number of channels, each with a frequency bandwidth, and therefore, a data rate. Though the composite signal ultimately carried by the medium is analog, the input signals can be analog or digital. If the input signals are analog, multiplexers at both the ends are sufficient as shown in Fig. 3.18.

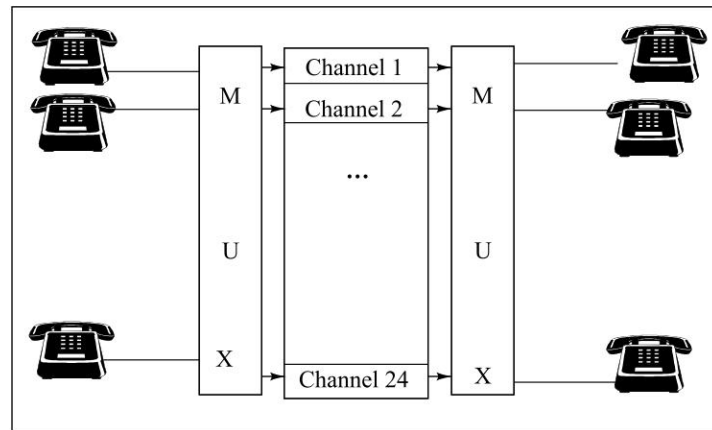


Fig. 3.18 Frequency Division Multiplexing (FDM) with analog input

If the input signals are digital, such as from computers, they need to be passed through modems and multiplexers as shown in Fig. 3.19. Modems convert the digital signals into analog signals and then at the other end, convert them back into digital signals.

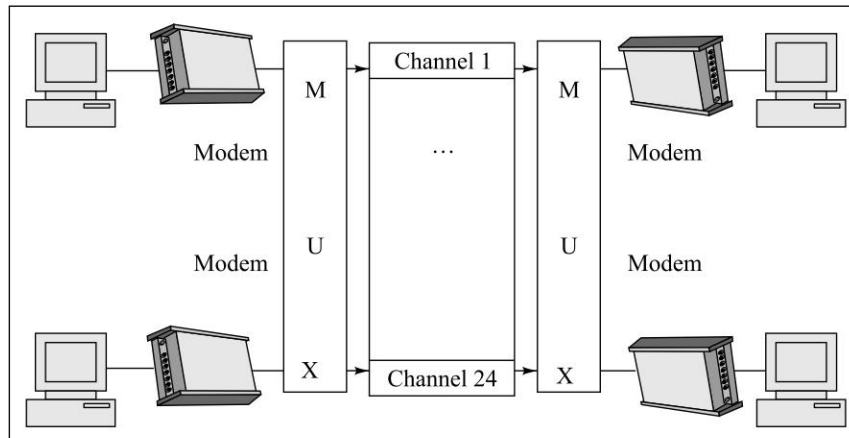


Fig. 3.19 Frequency Division Multiplexing (FDM) with digital input

FDM and Analog Telephone System

FDM is very important in the context of the analog telephone system. In the analog telephone system, voice signals are carried over twisted copper wire pairs as analog signals from your home

to the nearest exchange called **last mile**. There is no conversion from analog signals to their digital equivalents at this stage of transmission. As we have noted previously, frequencies of human voice that are carried by the telephone system are in the range of 0 to 4000 Hz (4 kHz). Therefore, to carry a single voice conversation over telephone, a bandwidth of 4 kHz is necessary and sufficient. However, the capacity of the medium such as twisted wire pair or coaxial cable is far higher. That is where FDM comes handy in enabling multiple conversations take place over a single wire or cable. As we know, this is useful only in sending signals between exchanges or from a local exchange to a higher level exchange. At an exchange, signals from multiple sources could be combined or multiplexed and then sent to a higher or different exchange. At the other end, they are demultiplexed and sent to the appropriate destinations. We must know that in a local loop, i.e., between the user and the local exchange, there is only one user and therefore, multiplexing is not possible.

In order to utilize the underlying infrastructure such as transmission equipment and switches to the maximum possible extent, telephone companies make use of FDM. By multiplexing signals from lines that have lower bandwidth onto those having higher bandwidth, maximum throughput can be achieved. There are standards for this, such as how many smaller lines (also called channels) should be combined into a bigger line, what should be the data rate, etc. These standards in Europe and Japan are different from those in the US. However, the basic principles remain the same. As an example, we shall briefly discuss the structure used by AT&T in the US, as shown in Fig. 3.20.

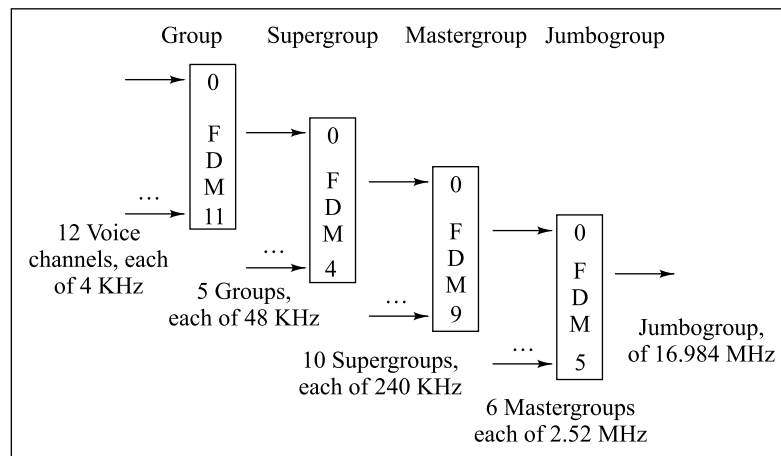


Fig. 3.20 *Analog telephone hierarchy*

1. At the first level, 12 voice channels, each having a bandwidth of 4 KHz, are multiplexed to form a **Group**. This group has a bandwidth of 48 kHz. A question at this stage would be as follows: Between any two channels, should there not be some unused empty portion (called a **guard band**) provided, so that the signals of the two channels do not mingle with each other, causing mixing of signals, and therefore, loss of information? The answer is *no*. As we have discussed previously, the unused frequency bands of 0 to 300 Hz and 3300 to 4000 Hz themselves act as guard bands, and the 300 to 3300 Hz portion is used for voice traffic. The FDM technique employed by AT&T has special equipment, which takes care of this issue. Therefore, we can say that a 48 kHz Group can carry 12 voice signals, each consisting of 4 kHz.

2. At the next level, 5 such groups are multiplexed to form a **Supergroup**. Thus, the bandwidth of a Supergroup is $5 \times 48 \text{ kHz} = 240 \text{ kHz}$. In other words, a supergroup can carry 60 voice signals, each of a bandwidth of 4 kHz.
3. At the third level, 10 supergroups are multiplexed onto a channel of still higher bandwidth, called **Mastergroup**. Thus, the bandwidth of a mastergroup channel is $10 \times 240 \text{ kHz} = 2.52 \text{ MHz}$. Thus, a mastergroup can carry 600 voice signals, each of a bandwidth of 4 kHz.
4. At the last level, 6 mastergroup channels are combined to form a **Jumbogroup**. The bandwidth of a jumbogroup is $6 \times 2.52 \text{ MHz} = 16.984 \text{ MHz}$. This means that a jumbogroup can carry 3600 voice conversations.

3.5.2 Time Division Multiplexing (TDM)

Time Division Multiplexing (TDM) is a technique used for digital transmission only. In this case, we do not divide the frequency bandwidth of a medium into a number of channels. We divide the transmission time into a number of time slices. We then allocate each time slice to a different source node, which wants to send some data. A node could mean a computer, a terminal or any device. The time slice during which a source node is sending some data, the entire bandwidth belongs to that source node. We know that the data rate in a medium is proportional to its bandwidth. Hence, the source node, which is sending some data during that time slice, can send the data at a high rate. However, because the total time span is divided into a number of time slices, the overall data rate from each source node again reduces. Therefore, the concept of a medium being divided into a number of channels is still a logically valid one.

It is thinkable but not practicable to allocate a large chunk of time to a source node until that source node completes its transmission, and then allocate another large chunk of time to the next source node and so on.

However, in this case, the time slice allocated to each source would be different because the time is allocated to a source node until it completes transmission; and each node may want different amount of data to be transmitted. This philosophy may require large amount of buffers at each source node to hold the data when some other source node is transmitting. In addition, this method can be very unfair to a few source nodes as it may involve long waits or indefinite postponement, if one of the source nodes *grabs* the medium for a very long time.

This is the reason why the time span is divided into equal time slices. There are two ways of allocating the time slices to various source nodes. They are given below:

1. **Synchronous TDM**, also known as **TDM**
2. **Statistical TDM**

Synchronous TDM or TDM

In the technique called **synchronous TDM**, also called only **TDM**, the time slice is allocated to a source node regardless of whether it wants to send some data or not. This is a simpler mechanism to identify at a destination node as to which data originated from which source node. This is on the basis of a fixed time slot. Therefore, the position of data within the data frame specifies its origin. However, it can be a very wasteful scheme, because the time slot is allotted to a source node even if it has nothing to send. Figure 3.21 illustrates this scheme.

A small buffer memory is associated with every source node. At any time, not all nodes may want to send some data. Regardless of this, the timing device in the multiplexer allocates some time for each node to transmit data from its buffer, and then repeats this cycle, e.g., A-B-C-D-A-B-

C-D, etc., as shown in the figure. By the time its turn comes next, if a node wants to transmit any data, it will have moved a small chunk of data to its buffer. If there was no data to be transmitted, the buffer will be empty, but then it will still be sent, as depicted in Fig. 3.21. (Refer to node B). Notice that there is no addressing scheme here to identify the source of data. The position of data in the composite message determines the source address.

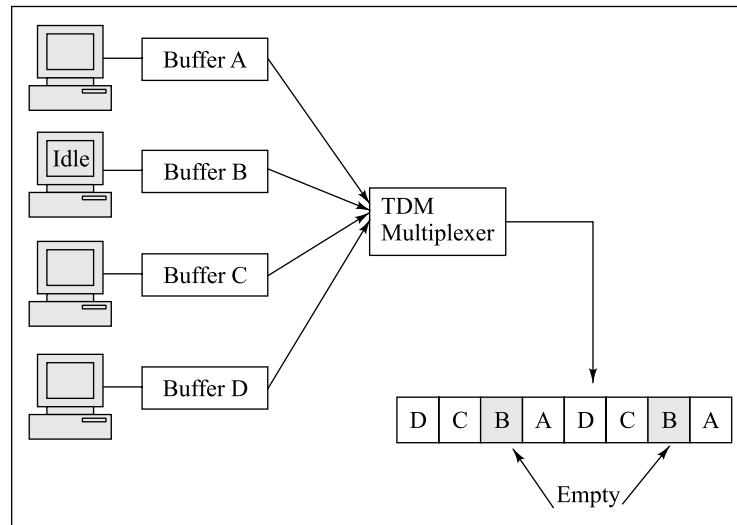


Fig. 3.21 Synchronous Time Division Multiplexing

Statistical Time Division Multiplexing (STDM)

Statistical Time Division Multiplexing (STDM) is a variation of the TDM technique. This technique is more *intelligent*. It monitors which machine or terminal is sending the data more frequently and in more quantity, and allocates the time slices more often to those nodes. Relatively inactive computers/terminals get the time slice less often, while a completely idle computer/terminal may not get any time slice at all. Essentially statistics is maintained about the activity of various source nodes and hence the name. This is shown in Fig. 3.22. In the figure, node B does not send any data at all, while node C is more active, and hence is polled more frequently.

This allows more number of computers/terminals to be connected to the line. For instance, on a 9.6 Kbps line, you could connect only 8 terminals sending data at the rate of 1200 Kbps or 1.2 Kbps using ordinary TDM. With STDM, you could connect more number of terminals, because, at a given time, not all terminals are normally active, and wanting to send data. This technique will work, so long as the number of terminals wanting to send data *at any moment* does not exceed 8.

It should be obvious that the position of data in the total data frames sent does not determine its address. In TDM, this was the case. In STDM, a time slot could be allotted to any of the nodes depending upon its past activity, and present requests. Therefore, in STDM, along with the chunk of data, some control information is sent in the data frame, which contains the addresses of both, the sender and the receiver as well as the number of bytes sent, etc. At the destination, another **stat mux** (a statistical multiplexer is called *stat mux*) captures the frame, separates out the data chunks and the receiver's address and sends data to the appropriate destination. Here, the sender's address tells

the receiver (destination) from whom the data has been received. This is unlike in TDM where the sender was defined by the time slot and therefore, it was not necessary to send the sender's address explicitly. Thus, in STDM, having to send addresses along with the actual data is an overhead, but then it obviated the need of sending nulls if a node did not want to send anything at all (as was the case with TDM). In most cases, this increases the effective throughput of the system.

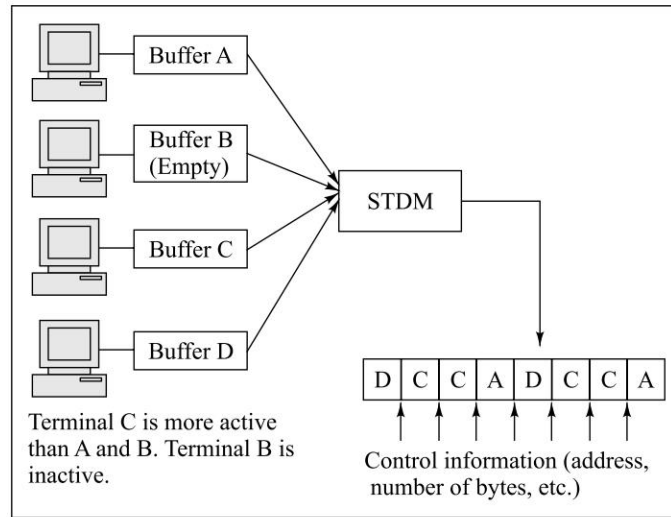


Fig. 3.22 Statistical Time Division Multiplexing (STDM)

TDM and the Digital Telephone System

Just as FDM helps in the creation of an analog telephone system hierarchy, TDM helps in the creation of a digital telephone system hierarchy. This is called **Digital Signal (DS) service**. DS is a hierarchy of digital signals, as shown in Fig. 3.23.

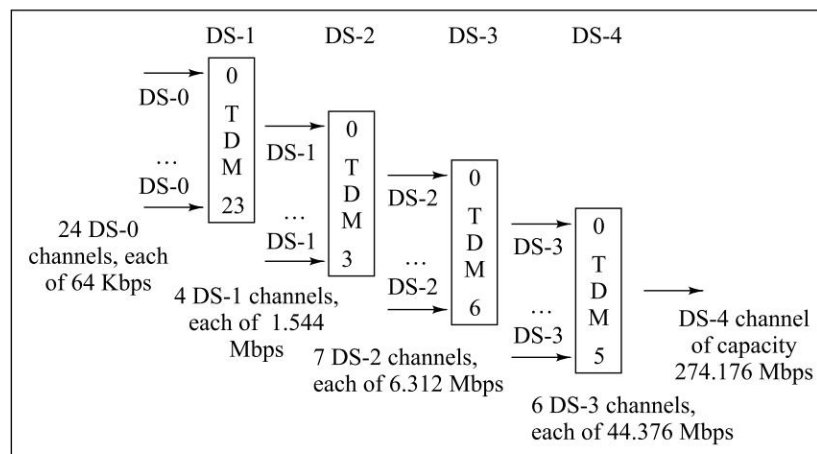


Fig. 3.23 Digital telephone hierarchy

Let us discuss the various channels.

1. **DS-0** is a single digital channel of capacity 64 Kbps, which can carry one voice conversation as we have seen before.
2. At the next level, 24 DS-0 channels are combined to form a **DS-1** service, which is a 1.544 Mbps service, including an overhead of 8 Kbps. This can carry 24 voice conversations simultaneously.
3. At the third level, 4 DS-1 channels are combined to form a **DS-2** service. The capacity of DS-2 service is 6.312 Mbps, which also includes an overhead of 168 Kbps. This can carry 96 voice conversations simultaneously.
4. Next, 7 DS-2 channels are combined to form a **DS-3** service. The capacity of DS-3 is 44.376 Mbps, which also includes an overhead of 1.368 Mbps. This can carry 672 voice conversations simultaneously.
5. At the last level, 6 DS-3 channels are combined to form a **DS-4** service, which has a capacity of 274.176 Mbps, including an overhead of 16.128 Mbps. This can carry 4032 voice conversations simultaneously.

DS-0 through DS-4 are the names of the services. Telephone companies in the US, which provide these services, use the nomenclature of **T-lines**. Thus, DS-0 through DS-4 are called **T-1** through **T-4** lines in the terminology of telephone companies. Figure 3.24 shows the data rates provided by the T-lines.

Name of the Service	Name of the Line	Rate	Voice Channels Supported
DS-0	T-0	64 Kbps	1
DS-1	T-1	1.544 Mbps	24
DS-2	T-2	6.312 Mbps	96
DS-3	T-3	44.736 Mbps	672
DS-4	T-4	274.176 Mbps	4032

Fig. 3.24  *DS and T lines*

3.5.3 Wavelength Division Multiplexing (WDM)

If the transmission medium is not copper wire but it is optical fiber, a variation of the basic FDM scheme is used. In this scheme called **Wavelength Division Multiplexing (WDM)**, transmissions from multiple optical fibers are combined and sent together. In order to ensure that these transmissions from different optical fibers do not overlap with each other and that they do not get mixed, their wavelengths are considered as the point of difference. In other words, just as in the case of FDM, the transmissions from various sources are combined and sent together on the basis of their different frequencies; here the transmissions are combined on the basis of the difference between the wavelengths. So, the process in brief is that transmissions from various optical fibers are combined on the sender's side. These transmissions have different wavelengths. The combined transmission is sent to the other side via a single and more powerful optical fiber. At the receiver's end, the individual transmissions are retrieved by filtering out the individual source signals on the basis of their wavelengths.

WDM is very similar to FDM, except that very high frequencies are used in WDM. Also, because it is based on optical principles and not electrical principles, it has very high accuracy as compared to FDM. Modern research allows as many as 200 different channels to share a single high capacity

optical fiber line. In such a case, where a significant number of input sources are combined into a single channel, we call it as **Dense WDM (DWDM)**.

The concept of WDM is illustrated in Fig. 3.25.

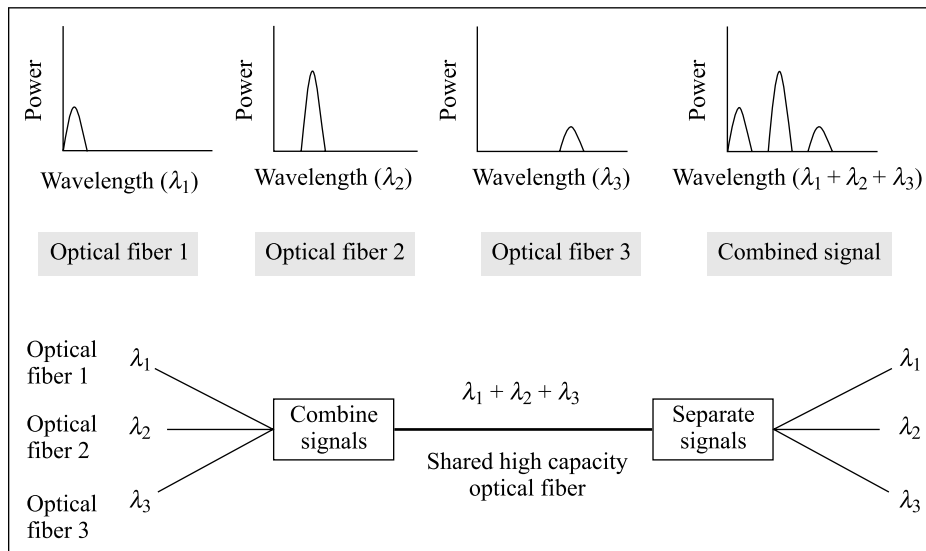


Fig. 3.25 WDM operation

3.6 FDM VERSUS TDM

We have studied both the methods of multiplexing, viz., FDM used for analog signals and TDM for digital signals. Let us study their merits and demerits.

1. With technological advancements, the cost of TDM is dropping rapidly and substantially. This is not true for FDM. Quite a few times, one may require modems along with the TDM system if TDM is to be used over telephone wires using modems.
2. TDM systems are more flexible than FDM systems. This is because we can intermix terminals with different speeds and synchronization methods, channel configurations and so on. This is not so easily achievable with FDM.
3. FDM is a simpler system to implement than TDM. However, the cost advantage of FDM is seen only when a few low speed channels are desired.
4. It is obvious that TDM is becoming more popular whenever a user is faced with a choice. This is quite consistent with the popularity of digital transmission for all types of information such as voice, image, picture and data.

SUMMARY

Data transmission modes can be categorized in a number of ways. The most important ones are Analog and Digital, Parallel and Serial communication, Asynchronous, Synchronous and Isochronous communication, and Simplex, Half-duplex and Full-duplex communication.

In parallel communication, we transfer a word or a byte at a time. Therefore, we need to have those many wires parallel to each other, each carrying a single bit. Over long distances, serial communication is used. While sending the data serially, characters or bytes have to be separated out and sent bit by bit. Thus, there is some hardware equipment involved in converting the data from parallel to serial. At the destination, the measurement of signal values is done in the middle of the bit durations.

In asynchronous communication, the time at which a character will be sent cannot be predicted as in data entry operation. Thus, each character is preceded with a start bit and succeeded with 1, 1.5 or 2 stop bits. In synchronous transmission, the whole block of data bits is transferred at once, instead of a character at a time. Synchronous communication is used when a large amount of data such as a file is to be sent from one place to the other. Synchronous transmission has obviously a much higher efficiency because it is a continuous transmission without any start/stop bits. Isochronous method combines the approaches of asynchronous and synchronous communications. In this method, as in the asynchronous method, each character has both the start and stop bits. However, in isochronous method, the idle time between characters is not random.

In simplex mode, communication is unidirectional. This is similar to a one-way street, where vehicles are allowed to drive only in a single direction. In case of half-duplex mode, both devices can transmit; however, not at the same time. When one device is sending data, the other must only receive it, and vice versa. This is conceptually similar to a street that has a single lane for vehicle traffic. In full-duplex (or simply duplex) communication mode, both the devices can transmit data at the same time. It means that both devices are capable of sending as well as receiving data at the same time. This is like a two-way street with traffic flowing in both directions at the same time.

The whole idea of multiplexing is to ensure better throughput overall and to do justice to multiple data transmissions. Multiplexing divides the physical line or a medium into logical segments called *channels*. In multiplexing, different channels carry data simultaneously over the same physical medium. Thus, if multiple computers are sharing the same transmission medium (wire), they can all transmit and receive data at the same time. A multiplexer and a demultiplexer together allow these multiple transmissions to be made (at the source) and received (at the destination). Multiplexing can be done in two major ways: Frequency Division Multiplexing (FDM) and Time Division Multiplexing (TDM). A variation of FDM is Wavelength Division Multiplexing (WDM).

In Frequency Division Multiplexing (FDM), used for analog signals, the available frequency bandwidth of the transmission medium is logically divided among all the signals to be transmitted. Cable TV technology uses the same principle to bring multiple television transmission to the subscribers' homes.

In contrast, Time Division Multiplexing (TDM) is a digital multiplexing technique wherein each transmission device (usually a computer) is allocated a fixed time slot for transmission. This round-robin strategy allows all the devices to transmit in turn. TDM can be further classified into synchronous TDM and statistical TDM. In synchronous TDM or only TDM, the time slice is allocated to a source node regardless of whether it wants to send some data or not. It is a simpler but wasteful approach. Statistical TDM is an *intelligent* technique. It monitors which machine or terminal is sending the data more frequently and in more quantity, and allocates the time slices more often to those nodes. Relatively inactive computers/terminals get the time slice less often, while a completely idle computer/terminal may not get any time slice at all.

KEY TERMS AND CONCEPTS

Asynchronous transmission	Simplex transmission
Demultiplexer	Skew
Digital Signal (DS) service	Start bit
DS-0	Statistical TDM
DS-1	Stop bit
DS-2	Supergroup
DS-3	SYN bit
DS-4	Synchronous transmission
Frequency Division Multiplexing (FDM)	Synchronous TDM
Full-duplex transmission	T lines
Group	T-0
Guard band	T-1
Half-duplex transmission	T-2
Isochronous transmission	T-3
Jumbogroup	T-4
Mastergroup	Time Division Multiplexing (TDM)
Multiplexing	Universal Asynchronous Receiver Transmitter (UART)
Multiplexer	Universal Synchronous Asynchronous Receiver Transmitter (USART)
Parallel transmission	Wavelength Division Multiplexing (WDM)
Parity bit	
Serial transmission	

QUESTIONS

True/False

1. In parallel communication, we transfer a word or a byte at a time.
2. Serial transmission involves the transfer of one sentence at a time.
3. In asynchronous communication, the time at which a character will be sent can be predicted.
4. Synchronous transmission has much higher efficiency as compared to asynchronous transmission.
5. Simplex means bidirectional data transfer.
6. Full-duplex transmission is the same as half-duplex transmission.
7. Multiplexing divides the physical line or a medium into logical segments called *channels*.
8. A multiplexer separates the signals meant for different destinations and sends them appropriately.
9. The mux is responsible for both multiplexing and demultiplexing.
10. If input signals are digital, they need to be passed through both modems and multiplexers.
11. We can use data compression techniques in FDM.

Multiple-Choice Questions

1. In _____, we transmit all the 8 bits at a time.
(a) serial transmission (b) parallel transmission
(c) synchronous communication (d) mixed transmission
2. Parallel transmission is generally used for _____ distances.
(a) short (b) long
(c) very long (d) intercity
3. _____ involves bit measurements in the middle.
(a) serial transmission (b) parallel transmission
(c) synchronous communication (d) mixed transmission
4. In _____, the time when a character will be sent cannot be predicted.
(a) parallel transmission (b) synchronous transmission
(c) asynchronous transmission (d) isochronous transmission
5. In _____, the whole block of data bits is transferred at once, instead of a character at a time.
(a) serial transmission (b) synchronous transmission
(c) asynchronous transmission (d) isochronous transmission
6. _____ is used when large amount of data is to be sent from one place to the other.
(a) Parallel transmission (b) Synchronous transmission
(c) Asynchronous transmission (d) Isochronous transmission
7. In _____, the idle period between the two characters cannot be random.
(a) parallel transmission (b) synchronous transmission
(c) asynchronous transmission (d) isochronous transmission
8. In simplex communication, _____.
(a) only one party can transmit data
(b) both parties can transmit data, but not at the same time
(c) both parties can transmit data at the same time
(d) no party can transmit
9. In half-duplex communication, _____.
(a) only one party can transmit data
(b) both parties can transmit data, but not at the same time
(c) both parties can transmit data at the same time
(d) no party can transmit
10. In full-duplex communication, _____.
(a) only one party can transmit data
(b) both parties can transmit data, but not at the same time
(c) both parties can transmit data at the same time
(d) no party can transmit
11. Multiplexing _____.
(a) divides one line into multiple channels
(b) combines many channels into one line
(c) is same as modulating
(d) is same as demodulating

12. In _____, the medium is divided into a number of channels, each with a frequency bandwidth.
 - (a) TDM
 - (b) STDM
 - (c) FDM
 - (d) None of the above
13. _____ is an analog multiplexing technique.
 - (a) FDM
 - (b) TDM
 - (c) STDM
 - (d) None of the above
14. _____ is a digital multiplexing technique.
 - (a) FDM
 - (b) TDM
 - (c) Modem
 - (d) None of the above
15. _____ is an intelligent multiplexing technique.
 - (a) Statistical TDM
 - (b) FDM
 - (c) TDM
 - (d) Synchronous TDM
16. The data rate of a T-3 line is _____.
 - (a) 64 Kbps
 - (b) 1.544 Mbps
 - (c) 44.376 Mbps
 - (d) 274.176 Mbps

Detailed Questions

1. Contrast parallel and serial transmission methods.
2. In which situations can parallel transmission be a better choice as compared to serial transmission?
3. What is the difference between synchronous and asynchronous transmissions?
4. What analogies can be used to explain simplex, half-duplex and full-duplex transmissions?
5. Why is full-duplex transmission more challenging than simplex and half-duplex transmissions?
6. What is multiplexing?
7. How does Frequency Division Multiplexing work?
8. Explain Time Division Multiplexing.
9. Explain the technique called synchronous TDM. Contrast it with statistical TDM.
10. Explain the differences between FDM and TDM. What is WDM?

4 Transmission Errors: Detection and Correction

4.0 INTRODUCTION

We know that it is virtually impossible to send any signal, analog or digital, over a distance without any distortion even in the most perfect conditions. This is basically due to various impairments that can occur during the process of transmission as a result of imperfect medium and/or environment. These errors can be classified in three basic categories as given below:

- Delay distortion
- Attenuation
- Noise

We shall discuss these concepts in this chapter. We shall also study the various errors that can take place during data transmission, and how we can trap and correct them.

4.1 ERROR CLASSIFICATION

Let us first discuss the three basic categories of errors.

Delay distortion is caused because signals at different frequencies travel at different speeds along the medium. We know that any complex signal can be decomposed into different sinusoidal signals of different frequencies resulting in a frequency bandwidth for every signal. The property of signal propagation is such that the speed of travel of frequency at the center of this bandwidth is the highest, and this speed is low at both ends of the frequency bandwidth. Therefore, at the receiving end, signals with different frequencies in a given bandwidth will arrive at different times. Hence, at the receiving end, if the receiving frequencies are measured at a specific time, they will not *measure up* to the original signal resulting in its misinterpretation.

Attenuation is another form of distortion. In this case, as a signal travels through any medium, its strength decreases, as shown in Fig. 4.1, just like our voice becomes weak over a distance and loses its contents beyond a certain distance.

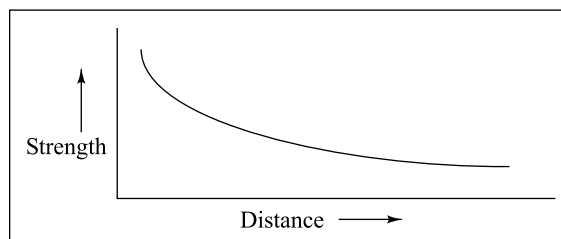


Fig. 4.1 Attenuation

Attenuation is very small at short distances. The original signal therefore can be recognized as such without too much of distortion. Attenuation, however, increases with distance. This is because some of the signal energy is absorbed by the medium. Attenuation is higher at higher frequencies (another reason why there is a limit on the maximum frequency that can be transmitted through a medium). Techniques are available to equalize attenuation for a band of frequencies over a medium. For telephone lines, this is achieved by using loading coils that change the electrical properties of the wire to smooth out the effects due to attenuation. However, it cannot be done away with. One can use amplifiers to boost the signal, but the amplifier boosts not just the signal, but also the accompanying noise.

Noise is yet another component, which poses a problem in receiving the signal accurately at the receiver. We know that signals travel as electromagnetic signals through any medium. Some electromagnetic energy can get inserted somewhere during transmission, which is normally called **noise**. Apart from distortion and attenuation, noise is one of the major limiting factors in the performance of any communication system.

In conclusion, the signal can change over a distance. It can, therefore, lead to a misinterpretation of the signal (0 to 1 and vice versa). This can be very dangerous in many situations.

4.2 TYPES OF ERRORS

If the signal is carrying binary data, there can be two types of errors, viz., **single-bit errors** and **burst errors**. This is shown in Fig. 4.2. In a single-bit error, a bit value of 0 changes to 1 or vice versa. In a burst error, multiple bits of a binary value are changed.

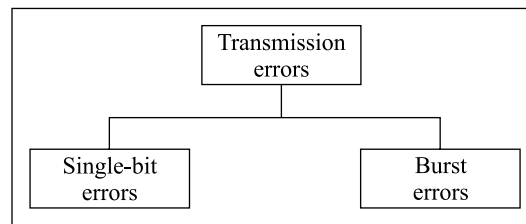


Fig. 4.2 Types of transmission errors

As we mentioned, in single-bit errors, a single bit of the data unit changes. Thus, effectively, either a 0 bit changes to 1, or a 1 bit changes to 0. Single-bit errors are more likely in the case of parallel transmission because it is likely that one of the eight wires carrying the bits has become noisy, resulting in corruption of a single bit for each byte. This can be a case of parallel transmission between the CPU and the memory inside a computer. In case of serial transmission, the duration of noise is usually longer than that of a single bit; hence the chances of corrupting only a single bit are less.

In contrast, a burst changes at least two bits during data transmission because of errors. Note that burst errors can change *any* two or more bits in a transmission. These bits need not necessarily be adjacent bits. Burst errors are more likely in serial transmission, because the duration of noise is longer, which causes multiple bits to be corrupted.

4.3 ERROR DETECTION

There are a number of techniques used for transmission error detection and correction. We shall examine the most common techniques used for this purpose, as described below.

4.3.1 Checksum

A **checksum**, also called **hash sum** is fixed-length data that is the result of performing certain operations on the data to be sent from the sender to the receiver. The sender runs the appropriate **checksum algorithm** to compute the checksum of the data, appends it as a field in the packet that contains the data to be sent, as well as various headers. When the receiver receives the data, the receiver runs the same checksum algorithm to compute a fresh checksum. The receiver compares this freshly computed checksum with the checksum that was computed by the sender. If the two checksums match, the receiver of the data is assured that the data has not changed unintentionally during transit.

Various checksum algorithms are popular. Most common examples of these are **parity check**, **modular sum**, **position-dependent checksum**, etc. We describe parity check in the following section. Later, we will also discuss the Cyclic Redundancy Check (CRC), which is an example of position-dependent checksum. As such, we shall cover the modular sum technique too, in brief here.

In modular sum, the data that the sender is sending is arranged into smaller blocks, called words. For example, if the data stream that needs to be sent is 1100100011111100110, then we may consider that this is made up of 5 words, each containing 4 bits. That is, the 5 words are 1100, 1000, 1111, 1110, and 0110. These are simply added together as follows:

	1100
+	1000
+	1111
+	1110
+	0110
=	111001

As we can see, the result is 111001. Then a 2's complement of the result is found, which is equal to 000111. This value is considered as the checksum and is sent along with the data. The receiver computes a fresh checksum and compares it with this received checksum. If the two checksum values tally, the receiver considers that the data was received without any errors or alterations.

Sometimes, extra carry bits resulting while adding the numbers together are discarded. This means that the sum of the numbers would be reduced from 111001 to 1001, taking just the last two digits. We would then find out the 2's complement of that result, which would be 0111. This would be our checksum in that case.

4.3.2 Vertical Redundancy Check (VRC) or Parity Check

Vertical Redundancy Check (VRC), also known as **parity check**, is quite simple to understand. It is the least expensive technique. In this method, the sender appends a single additional bit, called the **parity bit**, to the message before transmitting it. There are two schemes in this, viz., odd parity and even parity. In the odd parity scheme, given some bits, an additional parity bit is added in such a way that the number of 1s in the bits inclusive of the parity bit is odd. In even parity scheme, the parity bit is added such that the number of 1s inclusive of the parity bit is even.

For example, consider a message string 1100011 that needs to be transmitted. Let us assume the even parity scheme. The following will now happen:

1. The sender examines this message string, and notes that the number of bits containing a value 1 in this message string is 4. Therefore, it adds an extra 0 (so that the number of 1s in the bit string continues to be even) to the end of this message. This extra bit is called a parity bit. This is done by the hardware itself. That is why, it is very fast.
2. The sender sends the original bits 1100011 and the additional parity bit 0 together to the receiver.
3. The receiver separates the parity bit from the original bits, and it also examines the original bits. It sees the original bits as 1100011, and notes that the number of 1s in the message is four, i.e., even.
4. The receiver now computes the parity bit again and compares this computed parity bit with the 0 parity bit received from the sender, it notes that they are equal, and accepts the bit string as correct. This is also done by the hardware itself.

This process is shown in Fig. 4.3.

In contrast, if the original message was 1010100, the number of 1s in the message would have been three (odd), and therefore, the parity bit would have contained a 1.

Clearly, parity checking can detect single-bit errors. However, if multiple bits of a message are changed due to an error (burst), parity checking would not work. Better schemes are required to trap burst errors.

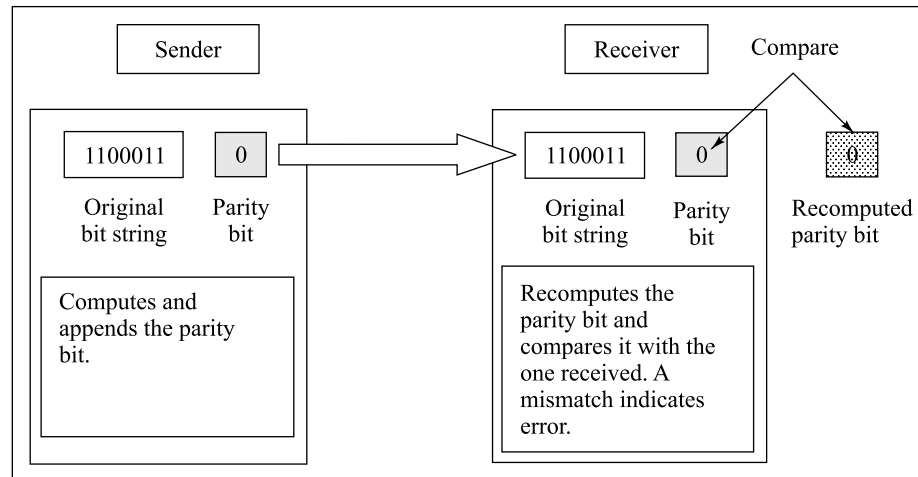


Fig. 4.3 Even parity VRC check

As mentioned earlier, there is one problem with this scheme. This scheme can only catch a single-bit error. If two bits reverse, this scheme will fail. For instance, if the first two bits in the bit stream shown in Fig. 4.3 change, we will get a stream 0000011, yielding a parity bit of 0 again, fooling us.

4.3.3 Longitudinal Redundancy Check (LRC)

A block of bits is organized in the form of a list (as rows) in the **Longitudinal Redundancy Check (LRC)**. Here, for instance, if we want to send 32 bits, we arrange them into a list of four rows. Then

the parity bit for each column is calculated and a new row of eight bits is created. These become the parity bits for the whole block. An example of LRC is shown in Fig. 4.4.

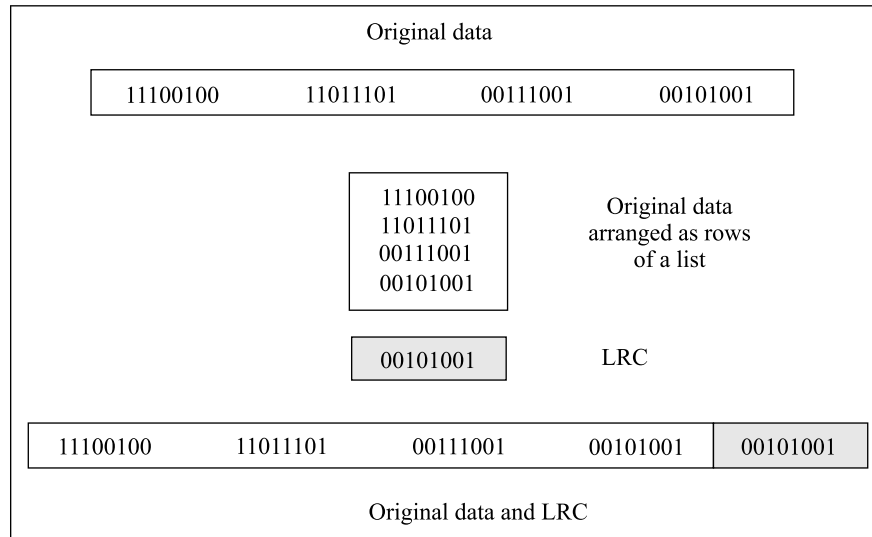


Fig. 4.4 Longitudinal Redundancy Check (LRC)

4.3.4 Cyclic Redundancy Check (CRC)

In **Cyclic Redundancy Check (CRC)**, a sequence of otherwise redundant overhead bits called **CRC** or **CRC remainder** is added to the end of the data to be transmitted. The CRC is so calculated that it can be perfectly divided by a second predecided number. At the receiver, the arriving data is divided by the same predecided number. If this division produces a zero remainder, the transmission is considered as error-free. In such a case, the incoming data is accepted by the receiver. If there is a remainder, it means that the transmission is in error and therefore, the arriving data must be rejected. At a broad level, the process is shown in Fig. 4.5.

The main features of CRC are as follows:

1. The question is why is CRC a better error detection method? Like in the case of VRC, if two adjacent bits change, will the CRC be the same, fooling us? The answer is no. The algorithm to compute the CRC is so chosen that given the length of the data block in bits, there are only a few and finite number of permutations and combinations for which the CRC is the same. The possibilities are also normally such that you have to inverse a number of specific and distant bits (e.g., 1, 43, 91, etc.) to get the same CRC (which can fool us). As we know, normally, errors occur in a burst, causing many consecutive bits and rarely exactly the bits required to get the same CRC. This is what makes CRC a very sturdy method.
2. CRC is normally implemented in the hardware (of the modem), rather than in the software. This makes this operation very fast, though a little more expensive. Depending upon the method of CRC used, the corresponding type of modem has to be used. For computing the CRC, two simple hardware components are used, viz., an XOR gate and a shift register. Using a combination of these two, we can calculate the CRC for any data to be sent. The method of actual calculation of CRC is beyond the scope of this text.

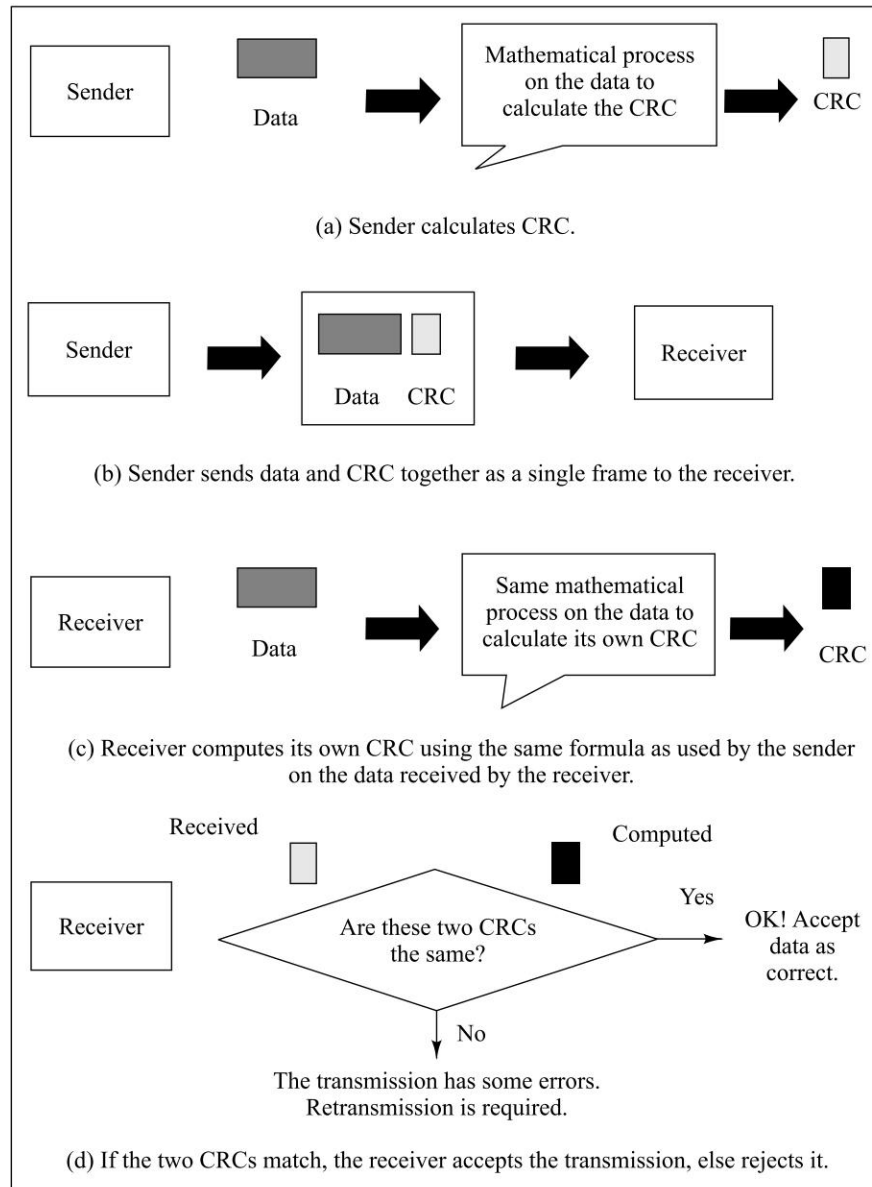


Fig. 4.5 The logical process of CRC computation

3. Data to be transmitted is divided in a number of blocks consisting of several bits each. Thereafter, a block is treated as a mammoth string of 1s and 0s in a binary number. It is then divided by a prime number, and the remainder is again treated as CRC.

4.3.5 Hamming Code

When we have any two binary strings or numbers, we can find out in how many bit positions do they differ. For example, if our two binary strings are 01101011 and 10101010, we can see that they

differ in 3 bits. How do we find out by how many bits do two such binary strings differ? Perhaps the simplest technique is to run an XOR operation on the two strings and then count the number of 1s resulting from this operation. That count value is the number of bits by which the two strings differ. In our example, we can do this as given below:

	01101011
XOR	10101010
Result	11000001

As we can see, the XOR operation gives us three 1s. Hence, we can say that the number of bits by which the two binary strings differ is 3. This count is called **Hamming distance**. Why is it important? This leads to a conclusion that if two strings differ by Hamming distance of k , then we will require k single-bit errors for converting one of the two strings into the other.

In the **Hamming code** technique, we add extra parity bits to identify a single possible error. To understand how this works, let us go through the following sample steps in one implementation of Hamming code:

1. Consider all bit positions that are powers of two as parity bits. Hence, bit positions 1, 2, 4, 8, 16, 32 ... would be parity bits.
2. All the other bit positions would contain data bits. Hence, bit positions 3, 5, 7, 9, 10, 11, 12, 13, 14, 15, 17 ... would contain data bits.
3. Every parity bit position that we have reserved in Step 1 is used to calculate and store the parity value for some bits. Interestingly, the parity bits are computed only for certain data bits, and not for all of them. How do we decide which data bits should be considered while computing the parity bits? It depends on the position of the parity bits. In general, depending on the parity bit position, it checks and skips those many bits for parity computation. We know that parity bits occupy positions 1, 2, 4, 8, 16, etc. Hence, effectively:
 - (a) Parity bit at position 1 would check 1 bit, skip 1 bit, check 1 bit, skip 1 bit, etc. Hence, it would check bits 1, 3, 5, 7, 9, 11, 13, 15.... In other words, it would skip bits 2, 4, 6, 8, 10, 12, 14, 16....
 - (b) Parity bit at position 2 would check 2 bits, skip 2 bits, check 2 bits, skip 2 bits, etc. Hence, it would check bits 2, 3, 6, 7, 10, 11, 14, 15.... In other words, it would skip bits 4, 5, 8, 9, 12, 13, 16, 17....
 - (c) Parity bit at position 4 would check 4 bits, skip 4 bits, check 4 bits, skip 4 bits, etc. Hence, it would check bits 4, 5, 6, 7, 12, 13, 14, 15, 20, 21, 22, 23.... In other words, it would skip bits 8, 9, 10, 11, 16, 17, 18, 19, 24, 25, 26, 27....and so on.
4. The parity bit value is set to 1 if the total number of ones in the bit positions that it is checking is odd. Otherwise, it is set to 0.

For example, consider a bit string 11000101. When we want to add parity bits to this, how would it look like? Obviously, we would like to reserve bit positions 1, 2, 4, and 8 for parity. Hence, we can draw the following picture of how this would look like now.

Bit position	1	2	3	4	5	6	7	8	9	10	11	12
Contents	Parity	Parity	Data	Parity	Data	Data	Data	Parity	Data	Data	Data	Data

Let us first fill in the data bits at the appropriate positions, excluding the parity bit positions. The result is as shown below.

Bit position	1	2	3	4	5	6	7	8	9	10	11	12
Contents	Parity	Parity	1	Parity	1	0	0	Parity	0	1	0	1

Now let us look at the parity bits.

- Parity bit at position 1 would look at data bits numbered 1, 3, 5, 7, 9, and 11. What do these positions contain?
 - Position 1 is itself. So it is discounted.
 - Position 3 contains 1.
 - Position 5 contains 1.
 - Position 7 contains 0.
 - Position 9 contains 0.
 - Position 11 contains 0.

Hence, total number of 1s is 2, which is even. This means it is an even parity. Hence, in bit position 1 reserved for parity, we would write 0.

- Parity bit at position 2 would look at data bits numbered 2, 3, 6, 7, 10, and 11. What do these positions contain?
 - Position 2 is itself. So it is discounted.
 - Position 3 contains 1.
 - Position 6 contains 0.
 - Position 7 contains 0.
 - Position 10 contains 1.
 - Position 11 contains 0.

Hence, total number of 1s is 2, which is even. This means it is an even parity. Hence, in bit position 2 reserved for parity, we would write 0.

- Parity bit at position 4 would look at data bits numbered 4, 5, 6, 7, and 12. What do these positions contain?
 - Position 4 is itself. So it is discounted.
 - Position 5 contains 1.
 - Position 6 contains 0.
 - Position 7 contains 0.
 - Position 12 contains 1.

Hence, total number of 1s is 2, which is even. This means it is an even parity. Hence, in bit position 4 reserved for parity, we would write 0.

- Parity bit at position 8 would look at data bits numbered 8, 9, 10, 11, and 12. What do these positions contain?
 - Position 8 is itself. So it is discounted.
 - Position 9 contains 0.
 - Position 10 contains 1.
 - Position 12 contains 0.
 - Position 12 contains 1.

Hence, total number of 1s is 2, which is even. This means it is an even parity. Hence, in bit position 8 reserved for parity, we would write 0.

Hence, after adding Hamming code parity bits, our resulting string would be as shown below.

Bit position	1	2	3	4	5	6	7	8	9	10	11	12
Contents	0	0	1	0	1	0	0	0	0	1	0	1

The receiver would have to take only the original bit string, i.e. 11000101. It would then recompute the Hamming code parity bits to verify the contents based on the result; accept or reject the original bit string as correctly received or received in error.

4.3.6 Recovery from Errors

Once detected, there are multiple ways of recovery from errors.

If we follow a scheme of necessarily sending some form of acknowledgement, the receiver will send a **Positive Acknowledgement (ACK)** back to the sender if every thing was ok, i.e., after checking that the CRC was matching and correct. If any error were to be found, it will send a **Negative Acknowledgement (NAK)** to the sender. The sender would wait until it receives an ACK or an NAK and would then decide whether to retransmit the same chunk or block or frame of data or not; and only then send the next block.

Example of such a scheme is **Stop-and-wait** method, as discussed next.

Stop-and-wait

This is a very simple method wherein the sender sends one frame of data and necessarily waits for an acknowledgement (ACK) from the receiver before sending the next frame. Only after the sender receives an acknowledgement for a frame, does it send the next frame. Thus, the transmission always takes the form: Data-ACK-Data-ACK ... etc., where the *data frames* are sent by the sender, and the *ACK frames* are sent by the receiver, back to the sender. This is shown in Fig. 4.6.

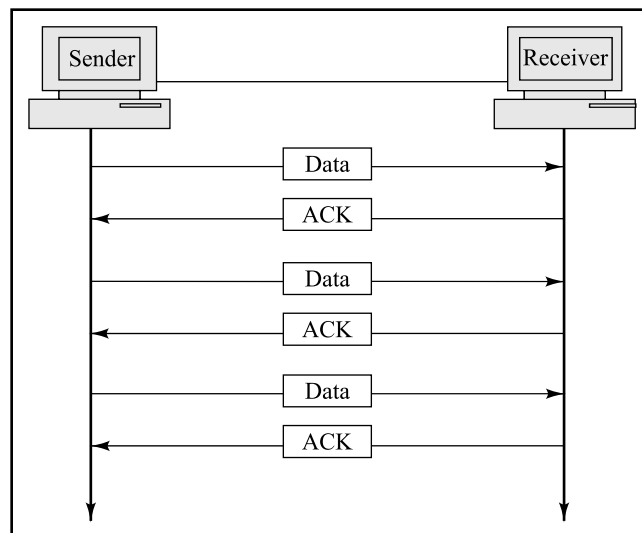


Fig. 4.6 *Stop-and-wait*

The stop-and-wait approach is pretty simple to implement. Every frame must be individually acknowledged before the next frame can be transmitted. However, therein also lies its drawback. Since the sender must receive each acknowledgement before it can transmit the next frame, it makes the transmission very slow.

The sender sets a timer for every frame that it sends. If it does not receive an acknowledgement from the receiver before the timer expires, it sends the same frame again, with a new timer. However, it could also happen that the sender resends the frame while the acknowledgement for the same is already half way through to the sender from the receiver. In this case, the receiver receives two or more instances of the same frame. The receiver is equipped with the understanding that the duplication has to be rejected, in such a case.

There are schemes which improve the efficiency of transmission by sending multiple frames, say, 1 to 8. If everything is ok, the receiver can send an ACK in one shot for all the frames. One of the very interesting and popular schemes is the **Go-back-n**, which is used when more than one frame is sent at a time. Its variant, **sliding window protocol**, is also used when more than one frame is to be sent at a time. We will now study these one by one.

Go-back-n

In **Go-back-n**, the sender starts retransmission with the last unacknowledged frame, even if the subsequent frames have arrived correctly at the receiver.

For example, suppose the sender sends five frames to the receiver (i.e., the window size is 5). Frames 0, 1 and 2 arrive correctly at the receiver, but frame 3 is in error. Therefore, the receiver sends a NAK for frame 3. By the time the NAK reaches the sender, the sender may have already sent the frames 4 and 5. However, after receiving the NAK, the sender now retransmits frame 3 and *all frames transmitted after frame 3*, i.e., frames 4 and 5 to the receiver. Assuming that this retransmission is successful, the receiver would now acknowledge this correct transmission of frames 3, 4 and 5 after discarding the duplicates for frames 4 and 5. This concept is shown in Fig. 4.7.

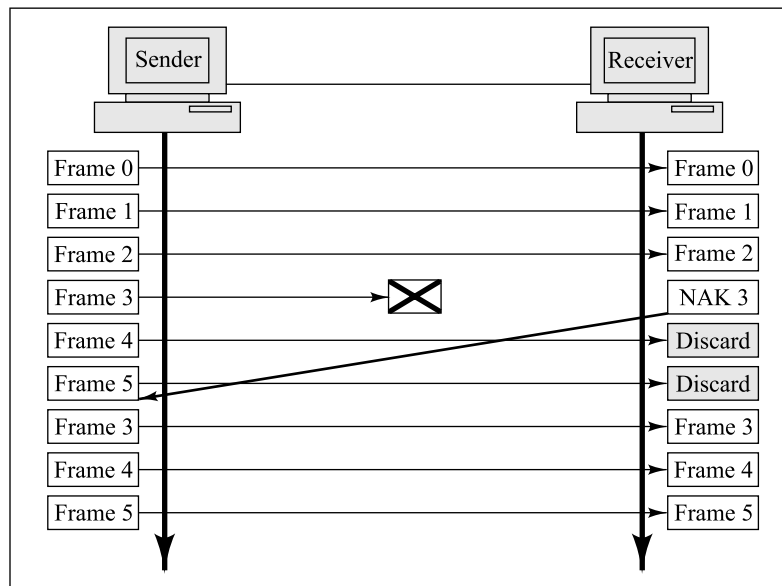


Fig. 4.7 Go-back-n

Let us consider another situation where the sender has transmitted all its frames and is actually waiting for an acknowledgement that has been lost on the way. The sender waits for some time and then retransmits the unacknowledged frames. The receiver detects this duplication, sends another acknowledgement and discards the redundant data. This is shown in Fig. 4.8.

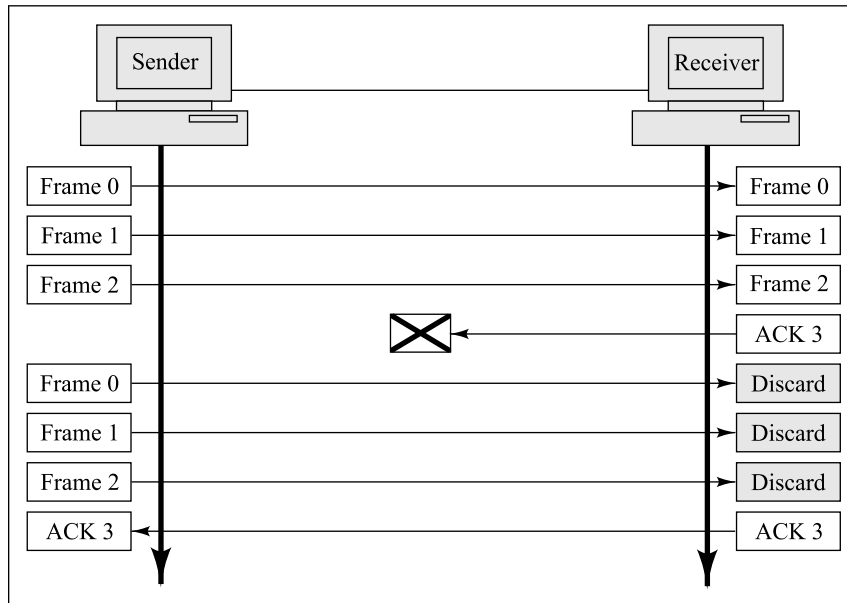


Fig. 4.8 Go-back-n, lost acknowledgement

Another possibility is when a damaged frame is received by the receiver. Here, the receiver receives frames 0 and 1 correctly, but does not immediately acknowledge them. It receives frame 2, which is in error. So, it returns NAK 2. This informs the sender that frames 0 and 1 have been received correctly, but that frame 2 must be resent. However, in this case, the receiver keeps accepting new frames (e.g., frames 3 and 4), and does not reject them. After it receives the resent (correct) frame 2, the receiver will send ACK 5, implying that it is now ready to accept frame 5. Obviously, for this scheme to work, the receiver needs a lot of memory and processing logic to keep a track of all these things. This situation is shown in Fig. 4.9.

Sliding Window

The **sliding window** technique is a variation of the Go-back-n technique. Normally, the data to be sent is divided into frames, for reasons, which we shall study later. The acknowledgement process in the *stop-and-wait* and a few other methods takes place for each frame. Thus, the sender must send a frame, wait for its acknowledgement, and only after it receives that acknowledgement, send the next frame. A trick to improve efficiency would be to send multiple frames at a time, check the CRC of all the frames one by one, send the acknowledgement for all, and request for the next set of frames. The **sliding window** technique is based on this philosophy.

In this technique, the underlying transmission mechanism defines an imaginary window consisting of maximum n frames to be sent at a time. The transmission mechanism allows the data to be transmitted at a time only up to the size of the window. The window defines how much data the

sender can send before it must wait to receive an acknowledgement from the receiver. The term *sliding window* is used because the data window *slides over* the data buffer to be sent.

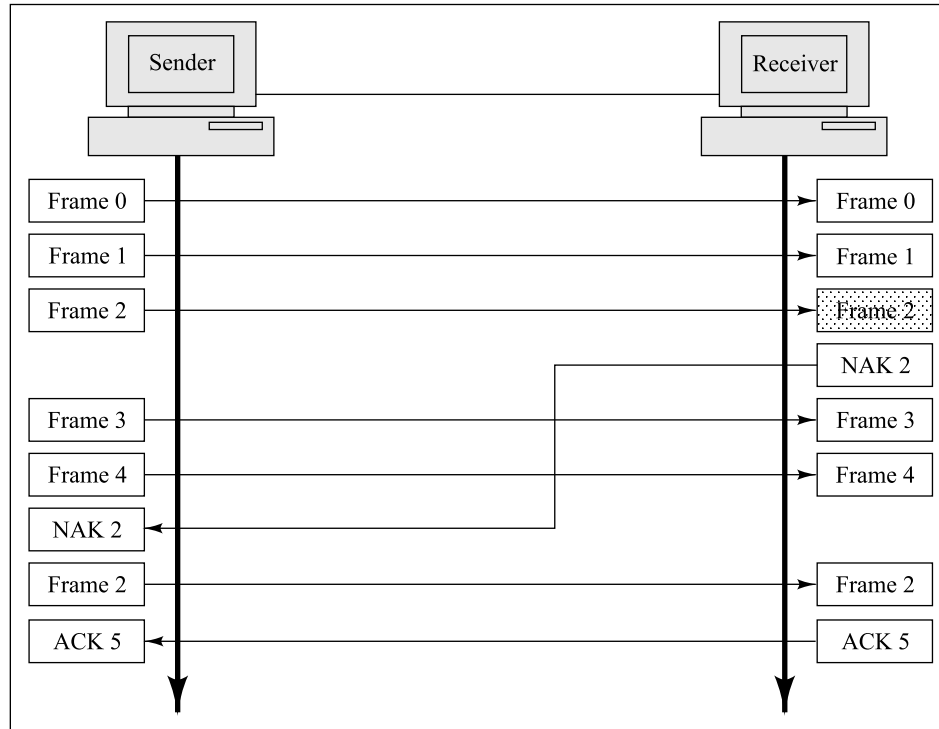


Fig. 4.9 Selective reject, damaged frame

Figure 4.10 shows a sliding window of size 8 frames. That is, the sender can send eight frames before it must wait to receive an acknowledgement from the receiver. Note that after the window of eight frames, the figure shows the next frames as renumbered from 1. This indicates that the first window has ended there, and that the second window has started.

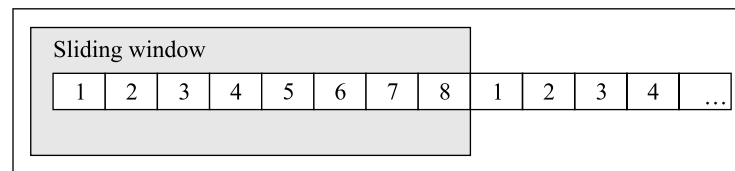


Fig. 4.10 Sliding window

It should be noted that both the sender and the receiver maintain their own sliding windows. The sender sends the number of frames that it is allowed to by its sliding window, and then waits for an acknowledgement from the receiver. When the receiver sends an acknowledgement back to the sender, it includes the number of the next frame that it expects to receive. Thus, if the sender has sent frames numbered 1 to 3 to the receiver, assuming that the receiver has received them correctly, the receiver sends back an acknowledgment that includes the number 4. Thus, the sender knows that the receiver has correctly received frames 1 to 3, and proceeds to send frame number 4.

Now suppose the sender sends these eight frames, as defined by the sliding window, and receives an acknowledgement for the first two frames. What happens next? In such a case, the sender slides the window two frames to the right, and sends the 9th and 10th frames (i.e., frame numbers 1 and 2 of the next lot) as shown in Fig. 4.11 to the receiver in lieu of the two frames received properly by the receiver. Thus, the receiver again has 8 frames. It can now check the CRC of all, one by one, and if it finds an error in frame 7 and finds frame 3, 4, 5 and 6 ok, it will send the acknowledgement that includes the number 7. The sender now sends 8 frames from frame 7 onwards again. This is shown in Fig. 4.11.

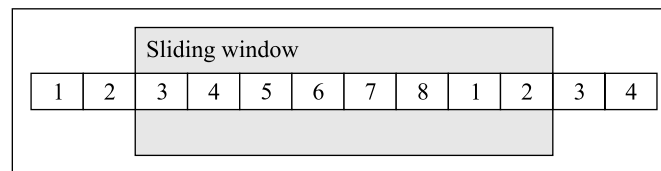


Fig. 4.11 *Sliding window mechanism*

The underlying transmission mechanism uses two buffers and one window to control the flow of data. The sender has a buffer for storing data coming from the sending application program. The application program creates the data to be sent and writes it to this buffer. The sender imposes a window (as discussed earlier) on this buffer, and sends frames till all the frames have been sent. The receiver also has a buffer. The receiver receives the data, checks it for any errors (CRC, duplicate/missing frames, etc.) and stores the correct ones in this buffer. The application program at the receiver's end then picks up the data from this receiver buffer.

To understand these concepts more clearly, let us depict the sender and receiver sliding windows now. Figure 4.12 shows the sender's window and the aspects associated with it.

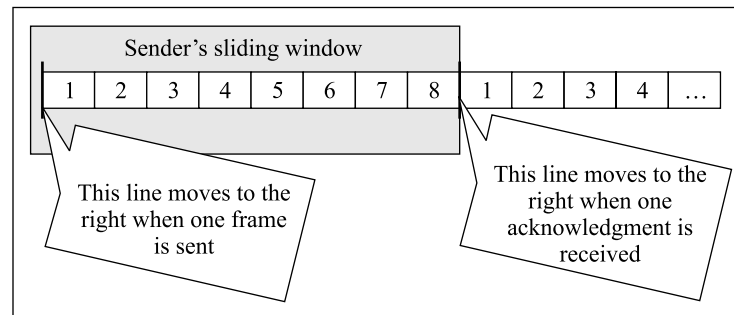


Fig. 4.12 *Sender's sliding window*

Thus, assuming that the sender's window size is 8, if frames 1 to 4 are sent and no acknowledgement has been received so far, the sender's window shall contain four frames numbered 5 to 8. After an acknowledgement bearing number 5 arrives, the sender knows that the receiver has correctly received frames numbered 1 to 4. Thus, the sender's window is now expanded to include the next four frames, making the window consist of frames 5, 6, 7, 8, 1, 2, 3, 4.

Thus, the sender's window shrinks from the left when it sends data frames, and expands to the right when it receives acknowledgements from the receiver.

The receiver's window is shown in Fig. 4.8. It expands to the right when acknowledgements are sent. To take a simple example, suppose that the receiver's window size is 8 (numbered 1 to 8). Suppose so far the receiver has acknowledged frames 1, 2 and 3. Further, let us assume that the receiver has now received frames 4, 5 and 6 correctly. Accordingly, the receiver shall expand its window by three places to the right ($6 - 3 = 3$).

Thus, the receiver's window shrinks from the left when frames are received, and expands to the right when it sends acknowledgements to the receiver. This is shown in Fig. 4.13.

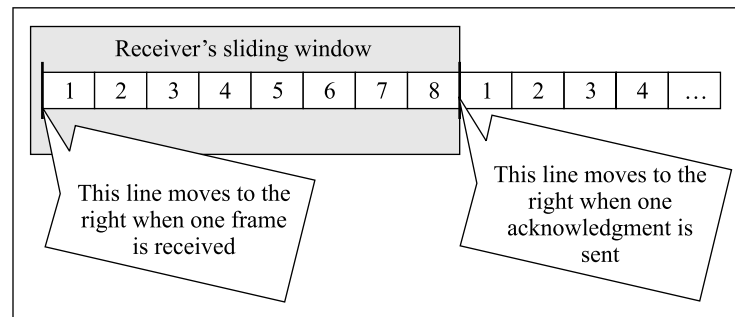


Fig. 4.13 Receiver's sliding window

SUMMARY

When data is transmitted from a sender to a receiver, there is a scope for transmission errors. That is, the data may not reach the receiver in the same form as it was sent, implying some bits could change. Transmission errors can be classified into three main categories, viz., delay distortion, attenuation and noise. Delay distortion is caused because signals at different frequencies travel at different speeds along the medium. As a signal travels along a transmission medium, its strength decreases over the distance. This is called attenuation of the signal. At times, some unwanted electromagnetic energy can get inserted somewhere during transmission, which is normally called *noise*.

To deal with transmission errors, the two steps followed are error detection and error correction. A number of techniques are available for each of these. The most powerful error detection mechanism is the Cyclic Redundancy Check (CRC). It is based on some mathematical principles, which ensure that the receiver can detect transmission errors. For this, the sender calculates and appends additional data bits called CRC (which is derived by applying some mathematical algorithm to the data bits) to the data to be sent, and then sends the data and the CRC together to the receiver. The receiver also calculates its own CRC after receiving the data from the sender using the same algorithm as was used by the sender. If the CRC computed and the CRC received match, the receiver accepts the data, else it rejects it. Hamming code is another example of 1 bit-error detection.

Stop-and-wait, time out and sliding window are some techniques that help in error control. In stop-and-wait the sender sends a frame, and waits for its acknowledgement before it can send the next frame. This simple method is also very slow. A better approach is to send a group of frames and have them acknowledged together. This approach is called the Go-back-n method. In Go-back-n, the sender retransmits all the frames starting with the last unacknowledged frame. The receiver discards the duplicate frames that it receives because of this retransmission. The sliding window method is one form of the Go-back-n method.

KEY TERMS AND CONCEPTS

Attenuation	Negative Acknowledgement (NAK)
Checksum	Noise
Cyclic Redundancy Check (CRC)	Odd parity
Delay distortion	Parity bit
Even parity	Parity checking
Go-back-n	Positive Acknowledgement (ACK)
Hamming code	Sliding window protocol
Hamming distance	Stop-and-wait
Longitudinal Redundancy Check (LRC)	Vertical Redundancy Check (VRC)

QUESTIONS**True/False**

1. Delay distortion is caused because the signals at different frequencies travel at different speeds along the medium.
2. Attenuation decreases with distance, i.e., for short distance, attenuation is large and vice versa.
3. CRC calculation is based on a specific portion of data.
4. The receiver sends a negative acknowledgement if there is any error in the received data.
5. In the sliding window method, only when the first byte is acknowledged by the receiver that the sender would send the next byte.

Multiple-Choice Questions

1. _____ is caused because the signals at different frequencies travel at different speeds along the medium.
 - (a) Noise
 - (b) Delay distortion
 - (c) Attenuation
 - (d) Retransmission
2. As a signal travels through any medium, its strength decreases due to _____.
 - (a) noise
 - (b) delay distortion
 - (c) attenuation
 - (d) retransmission
3. Some electromagnetic energy can get inserted somewhere during transmission, which is normally called _____.
 - (a) noise
 - (b) delay distortion
 - (c) attenuation
 - (d) retransmission
4. Overhead bits are added to data in case of _____.
 - (a) CRC
 - (b) parity checking
 - (c) noise
 - (d) None of the above
5. The receiver sends a _____ back to the sender if every thing was ok.
 - (a) NAK
 - (b) PAK
 - (c) ACK
 - (d) NCK

6. In _____, the sender sends one frame and waits for an acknowledgement from the receiver before sending the next frame.
- (a) stop-and-wait
 - (b) Go-back-n
 - (c) sliding window
 - (d) CRC
7. The _____ defines how much data the sender can send before it must wait to receive an acknowledgement from the receiver.
- (a) buffer
 - (b) memory
 - (c) parity
 - (d) sliding window

Detailed Questions

1. Describe the three categories of distortion, viz., delay distortion, attenuation and noise.
2. Discuss the concept of parity checks.
3. Explain how Cyclic Redundancy Check (CRC) works.
4. Explain stop-and-wait method.
5. How does the sliding window technique work?
6. Describe the Go-back-n technique.
7. What is checksum? How does it work?
8. Write a note on Hamming code. How would you compute the Hamming code for a bit string 11110111?

5

Data Compression and Encryption

5.0 INTRODUCTION

The basics of data compression are very simple. All forms of information represented in the digital form (0 or 1), such as numbers, text, sounds, images, and videos have some elements, which are redundant. For instance, if a text contains the character A 20 times, it is not wise to communicate all 20 characters as A. In fax transmission, various dots on the paper are converted into electrical signals depending upon their greyness, and then sent. At the other end, the dots with the same greyness are generated using these signals, one by one, for each dot, to regenerate the page that was faxed. It is pointless to send signals for thousands of dots if, say, the upper half of a page is all white.

We can have some equipment, which will *study* or *scan* the information to be sent and substitute some kind of control information to represent the repetitive data and transmit it to achieve compression. At the other end, we can have equipment, which will *interpret* these control characters and expand the compressed data to the original form. Obviously, the addition of control information is an overhead, but so long as the amount of data including the overhead to be transmitted is less than the original one, compression will be beneficial.

There are basically the following three approaches to data compression:

- Simple coding scheme
- Based on the context of symbols
- Based on the relative frequencies or occurrences of symbols

We shall now discuss each of these techniques.

5.1 SIMPLE CODING SCHEME

The **simple coding scheme**, as the name suggests, is really simple. Imagine, for instance, that we have to transmit names of cities such as Mumbai or New York, etc. Each character (such as N, e, w, etc.) sent on the communication link will occupy at least 8 bits, plus a few others, for error detection and synchronization. Hence, if a name contains 10 characters, it will contain at least 80 bits.

Instead of this, if we codify these cities, the data to be sent can be reduced substantially. For instance, 01 could mean Mumbai, 02 could mean New York and say 44 could mean London. Now we will have to send only two characters, or 16 bits for the two characters representing the two digits, in this case. In fact, if these codes are numeric and with values between 01 and 99, we can

send only the binary values of these codes. In this case, we would be required to send only 7 bits, as we can have values from 0 to 127 with 7 bits. In this case, we could get substantial savings.

Clearly, this method is the simplest, but for it to work both the sender and the receiver must agree on a common coding scheme and adhere to it. In practice, this may not always be possible; especially when multiple parties are communicating with each other, i.e., the number of senders and receivers is more. Because, here, there could be many widely differing coding schemes, that can lead to a lot of confusion. In addition, it is virtually impossible to codify everything that one wants to communicate.

5.2 BASED ON THE CONTEXT OF SYMBOLS

We will consider two methods in this category as shown below:

- Null suppression
- Run Length Encoding (RLE)

5.2.1 Null Suppression

Null suppression is a very old technique. It is still used in the BISYNC protocol used in IBM 3780. It is based on the observation that while we transmit data items such as names and addresses or in general any descriptive text, there are a number of blanks or spaces that need to be sent. In such a case, sending the actual spaces can be very wasteful if there are too many of them to be sent in succession. For each one, we will have to send at least 8 bits.

Therefore, a coding scheme is used, which specifies how many spaces have to be sent. Figure 5.1 depicts this. In the first line, we see that there are seven spaces (indicated seven times by the character b for ease of representation) between C and D.

String to be sent	: AB C bbbbbbb DEF
String actually sent	: ABC C _s 7bDEF

Fig. 5.1  Null suppression

In the second line, C_s is a special character, which indicates that what follows is a number indicating how many times the space character repeats itself. This number is 7 because there are 7 spaces between ABC and DEF. At the receiving end, the equipment looks for this special character C_s and if found, it stores the number that follows it. It then inserts those many spaces at that place and reconstructs the original string.

Though this method is fairly primitive, it can result in substantial savings (as high as 40–50%) as compared to a transmission, which does not employ any data compression.

5.2.2 Run Length Encoding (RLE)

Run Length Encoding (RLE) is a generalization of the scheme of null suppression. In this scheme, we look for compressing not only the repeating spaces, but also any repeating characters. The basic principle is the same. Figure 5.2 shows the compression format.

As an example of this, if we have to send some text as XYZ.....KL with 9 dots between XYZ and KL, we could send the text as follows.

XYZ C_s9.KL

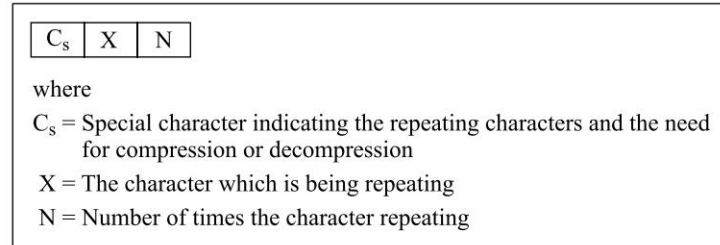


Fig. 5.2 The compression format

At the receiving end, after encountering the special character C_s , that node will repeat the character *dot* nine times to reproduce the original input string. An example of this scheme is shown in Fig. 5.3.

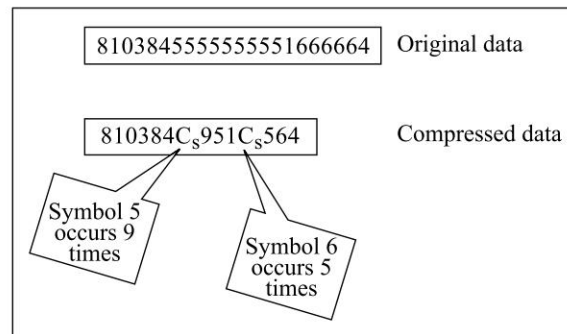


Fig. 5.3 Run Length Coding (RLE)

The efficiency of compression is measured as a ratio as shown in Fig. 5.4.

$\text{Compression ratio} = \frac{\text{Length of uncompressed data}}{\text{Length of compressed data (Including special codes etc.)}}$

Fig. 5.4 Efficiency of compression

It is obvious that if any character repeats only two times, it is not wise to use the compression technique, because the compression format will use three bytes to represent the repeating characters as shown in Fig. 5.3, instead of the normal two. Hence, in this case, using the run length encoding is more expensive. If the character repeats itself three times, it is a break-even situation with compression ratio of one. However, there are overheads of compression/decompression (e.g., interpretation of control character, etc.) that make this method unattractive. Therefore, this method is used only if any character repeats itself at least four times or more.

5.3 BASED ON THE RELATIVE FREQUENCIES OF SYMBOLS


Also called **statistical compression**, this method depends on how many times symbols occur. That is, short codes are used for compressing symbols that occur more frequently, and long codes for

symbols that occur infrequently. This is different from the normal way we use the coding system with fixed number of bits for a code (ASCII = 7, EBCDIC = 8, etc.). By using this approach, the length of data is reduced dramatically. A few compression techniques using this approach are described below.

5.3.1 Huffman Code

The **Huffman code** uses variable length codes made up of zeroes and ones to encode characters and other symbols. It uses the same principle as used in the Morse code. The Huffman code, like the Morse code, assigns small symbols to encode frequently occurring characters, and long symbols to encode the infrequently occurring characters. For instance, the Huffman code assigns just one bit each to the characters E and T, just two bits to the characters A, I, M and N, and so on. However, this is not as simple as it first appears. For instance, suppose we decide to use a coding scheme as shown in Fig. 5.5. Here, we have assigned just one bit each to the most frequently occurring characters (i.e., 0 for E and 1 for T). Similarly, we have assigned two bits for characters that occur quite frequently, but not as frequently as E and T (i.e., 00 for A, 01 for I, etc). In a similar way, we will have groups of three bits for other characters, four bits for a few more characters, and so on.


E = 0 T = 1			
A = 00	I = 01	M = 10	N = 11
C = 000	D = 001	G = 010	K = 011 ...

Fig. 5.5  Bit assignments based on the frequency of characters

The trouble with this scheme can be demonstrated when we have a string of characters in the form of a message. For instance, suppose we receive a message in the codified form of 000100. Now, how do we interpret this? There can be many possible interpretations. For instance, we could interpret this using the table shown in Fig. 5.6 as either EEETEE, or as EAME, or as AIA, and so on. Clearly, having a variable number of bits to represent each character is not going to work in this fashion.

To deal with this problem, we use the Huffman code. In this method, we assign weight to each character based on its frequency in terms of percentage, i.e., how many times out of any 100 characters, a particular character occurs. For instance, studies show that E occurs 15 times out of 100 (i.e., 15%), T occurs 12 times out of 100 (i.e., 12%), and so on, as shown in Fig. 5.6.

E = 15	T = 12	A = 10	I = 08	M = 07	N = 06	C = 05
D = 05	G = 04	K = 04	O = 03	R = 03	S = 02	U = 02

Fig. 5.6  Weights to be assigned to different characters

Now we use these weights to build a tree of characters. For this, the following steps are performed:

1. Organize the entire set of characters into a single row, ordered by the values of their weights (either from the lowest to the highest or vice versa). These can be considered as leaves of a tree-like structure, the tree itself being called the Huffman tree. This is shown in Fig. 5.7.

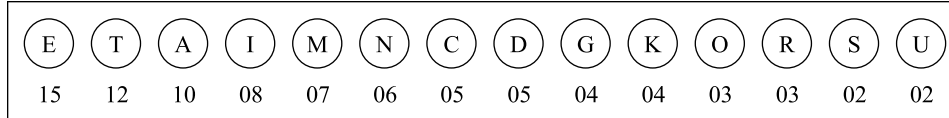


Fig. 5.7 Huffman tree: Step 1

2. Now we find the sum of two adjacent nodes in such a fashion that it is the smallest among all such sums. For instance, if we sum U and S, we have $02 + 02 = 04$. At this time, we check that the lowest three values at the nodes are 3, 3 at nodes O and R respectively (arrived at by this summation). If we add $4 + 3$, we get 7. However, if we add $3 + 3$, we get 6. This is the lowest at this juncture. Therefore, we add $3 + 3$ to get 6. Similarly, now we have 4, 4 at nodes G and K as the lowest values. $4 + 6 = 10$ is greater than $4 + 4 = 8$. Therefore, we add $4 + 4$. Now, if we take all such sums (some of which are shown in Fig. 5.8), we will arrive at a new level of numbers, which are actually the sums.

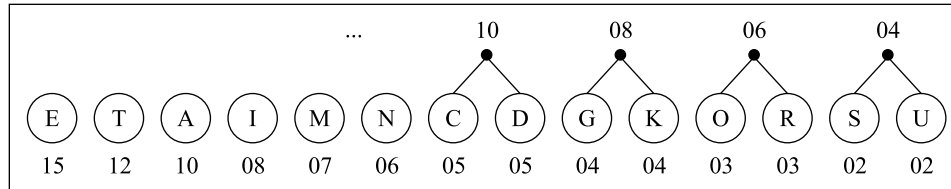


Fig. 5.8 Huffman tree: Step 2

3. We continue this process at all the levels, each time looking for the lowest possible sum of the adjacent nodes at any level. That is, again find the smallest sum of nodes at the first and the second levels. As we can see, we now have $04 + 06 = 10$ as the smallest total. This is shown in Fig. 5.9.

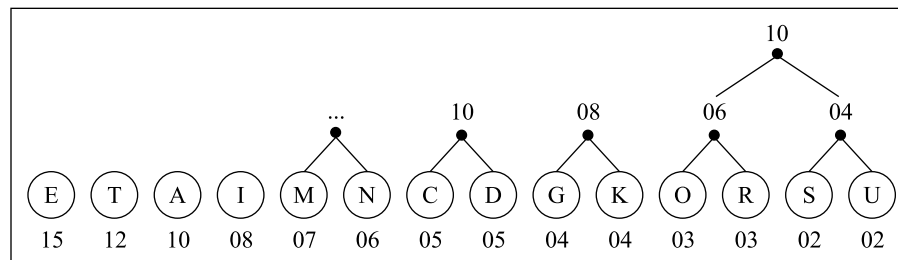


Fig. 5.9 Huffman tree: Step 3

4. We repeat this process. Now we can see that the different possible sums are $10 + 08 = 18$, $10 + 10 = 20$, $10 + 13 = 23$, etc. But remember that even the lowest level nodes are still candidates for participating in this exercise of summation. That is, we can still test for $N + M$, i.e., $06 + 07 = 13$, $I + A$, i.e., $08 + 10 = 18$, and so on. From all these, we can derive the lowest sum as $N + M = 06 + 07 = 13$. This is shown in Fig. 5.10.

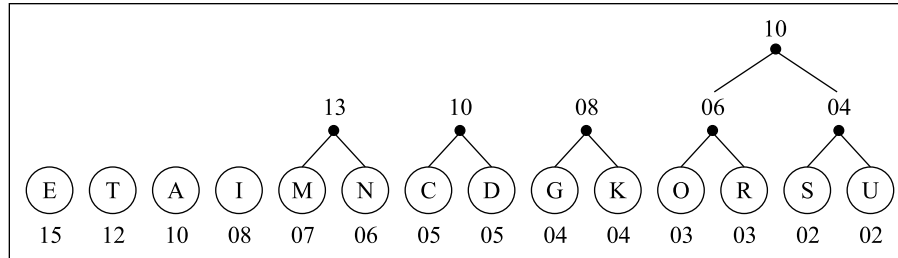


Fig. 5.10 Huffman tree: Step 4

5. We continue this process until all of the nodes, at every level, are combined into a single tree. The final tree looks as shown in Fig. 5.11. We shall leave as an exercise, the steps required to reach this final tree for the reader.

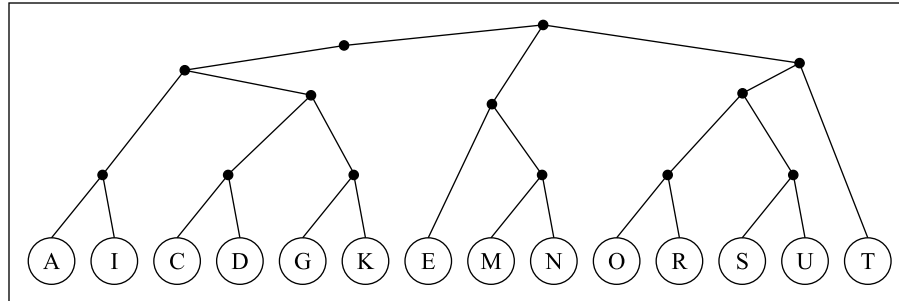


Fig. 5.11 Final Huffman tree

6. The next step is to assign codes to each character. Firstly, we assign a bit value (0 or 1) to each branch. Starting from the root (i.e., topmost node), we assign 0 to the left branch and 1 to the right branch. We repeat this pattern for each node. The resulting tree is shown in Fig. 5.12.

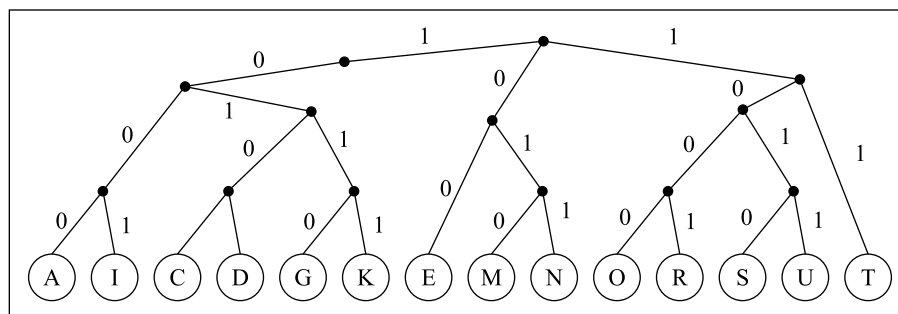


Fig. 5.12 Huffman tree with code assignment

Now, to find the code corresponding to a character, start at the root and follow the branches that lead to the character. The code is the bit values of the entire path taken to reach the character in this fashion. For instance, A = 000, I = 001, C = 0100, and so on. The most important aspect of this scheme is that since each character is at a different and unique path than any other character in the

tree, no code can be the prefix for any other code in the tree. That is, when we say that A = 000, we are confident that there cannot simply be any code such as 0 or 00, i.e., the prefix of 000.

Figure 5.13 shows the code assignment table thus obtained.

Character	Code	Character	Code
E	100	D	0101
T	111	G	0110
A	000	K	0111
I	001	O	11000
M	1010	R	11001
N	1011	S	11010
C	0100	U	11011

Fig. 5.13 Codes for characters using Huffman tree

Let us list the steps required to decode a message that has been codified using the Huffman coding scheme in our example with only a limited number of characters.

1. Read the first three bits of the message and see if they match any of the codes in our table shown in Fig. 5.14. If a match is found, replace the three bits with the appropriate character and discard the three bits. Repeat the process with the next three bits.
2. If no match is found in Step 1 above, read the next single bit and add it to the existing three bits, making it a 4-bit string. Look for the corresponding four-bit pattern in the table. If a match is found, replace the four bits with the appropriate character and discard the four bits. Repeat from Step 1.
3. If no match is found in Step 1 and Step 2 above, read the next single bit and add it to the existing four bits, making it a 5-bit string. Look for the corresponding five-bit pattern in the table. If a match is found, replace the five bits with the appropriate character and discard the five bits. Repeat from Step 1.
4. If still no match is found, it means that the message has some unrecognizable bits, and therefore, some errors. Therefore, take appropriate action such as discarding the message.

For example, suppose that we receive a bit string 01001101111. Let us attempt to decode it by applying the above steps.

- (a) The first three bits 010 do not have a matching pattern in Fig. 5.14. Therefore, let us also consider the fourth bit to have a pattern of 0100. Now, we have a match with character C. Therefore, let us discard the first four bits and start reading with the fifth.
- (b) Again, let us read the three bits starting with the fifth bit. Thus, we have the bit pattern of 110. This does not have a matching pattern in Fig. 5.14. Therefore, let us append the next bit to this pattern to make this 1101. We still do not have a matching pattern. Therefore, let us read one more bit to have a pattern of 11011. Now, there is a match with U. Therefore, our message so far is CU and the bit pattern to be still read is 111.
- (c) The remaining bit pattern is 111. A look at Fig. 5.14 tells us that it matches with T. Therefore, our message is CUT.

There also exists a modified Huffman code, which is popular in compressing fax messages. A detailed discussion of this, however, is beyond the scope of this text.

5.3.2 Lempel-Ziv Code

The **Lempel-Ziv code** looks for repeated strings or words, and stores them in variables. It then replaces the actual occurrence of the repeated word or string with a pointer to the corresponding variable. Since a pointer requires only a few bits of memory as compared to the original string, this method results in the data being compressed.

For instance, consider the following string:
What is your name? My name is Achyut.

Using the Lempel-Ziv coding, we would create two variables, say A and B and replace the words 'is' and 'name' by pointers to A and B, respectively. This is shown in Fig. 5.14.

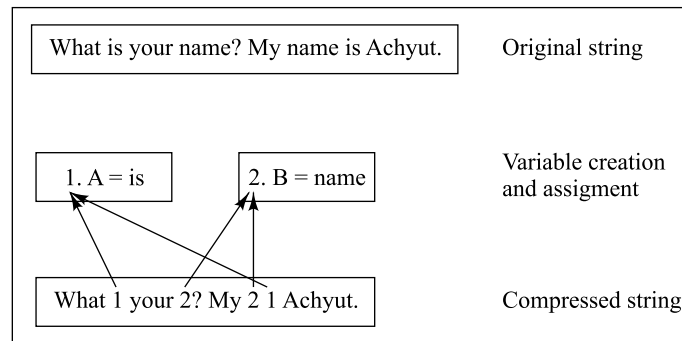


Fig. 5.14 Lempel-Ziv coding

Increasingly new techniques are evolving for compression, which reduce the required disk capacity and also the transmission time. As compression/decompression is carried out mostly in hardware, it is fairly fast and efficient.

Compression can be of the following two types:

1. In **lossless compression**, no information is lost. All the information (that is, all the numbers representing an image in digital form) is compressed and none is discarded.
2. In **lossy compression**, redundant information is discarded. This might be undesirable in some cases, as the information being discarded during compression could be vital.

Using any of this technique, different graphics and audio file formats have been devised by various vendors. These formats define how many bits are required for storing the graphics or audio data.

5.4 INFORMATION SECURITY

Let us assume that a person A wants to send a message to person B over the Internet. This can lead to a number of possible security-related issues. We shall discuss all these security issues and principles in the next few sections.

5.4.1 Confidentiality

The principle of **confidentiality** specifies that only the sender and the intended recipient(s) should be able to access the contents of a message. Confidentiality gets compromised if an unauthorized person is able to access a message. An example where in the confidentiality of a message has been

compromised is shown in Fig. 5.15. Here, user of computer A sends a message to user of computer B. (Hereafter, we shall use the term A to mean *user A*, and B to mean *user B*, etc., although we shall just show the computers of user A, B, etc.). If another user C, gets access to this message, which is not desired, it therefore defeats the purpose of confidentiality. An example of this could be a confidential email message sent by A to B, which is accessed by C without the permission or knowledge of A and B.

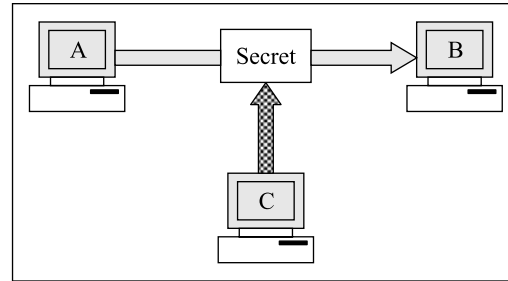


Fig. 5.15 Loss of confidentiality

5.4.2 Authentication

Authentication mechanisms help establish proof of identities. The authentication process ensures that the origin of an electronic message or document is correctly identified. For instance, suppose that user C sends an electronic document over the Internet to user B. However, the trouble is that user C had posed as user A when s/he sent this document to user B. How would user B know that the message has come from user C, who is posing as user A? A real life example of this could be the case of a user C, posing as user A, sending a funds transfer request (from A's account to C's account) to bank B. The bank might transfer the funds from A's account to C's account, for after all, it would think that user A has requested for the funds transfer. This concept is shown in Fig. 5.16.

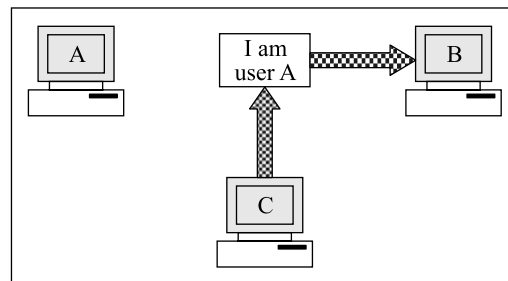


Fig. 5.16 Absence of authentication

5.4.3 Integrity

When the contents of a message are changed after the sender sends it, but before it reaches the intended recipient, we say that the *integrity* of the message is lost. For example, suppose you write a check for \$100 to pay for the goods bought from the US. However, when you see your next account statement, you are startled to see that the check resulted in a payment of \$1000. This is a case of loss of message integrity. Conceptually, this is shown in Fig. 5.17. Here, user C tampers with a message originally sent by user A, which is actually destined for user B. User C somehow manages to access it, change its contents, and send the changed message to user B. User B has

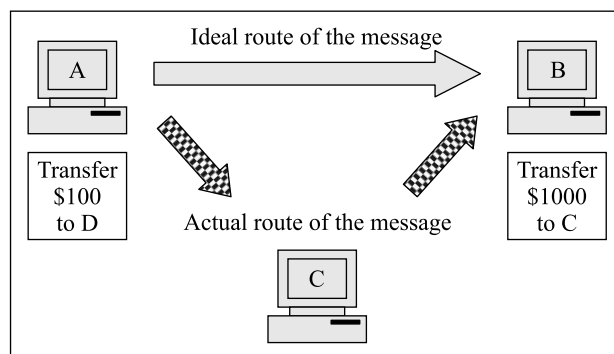


Fig. 5.17 Loss of integrity

no way of knowing that the contents of the message were changed after user A had sent it. User A also does not know about this change.

5.4.4 Non-repudiation

There are situations where a user sends a message, and later on refuses to admit that s/he had sent that message. For instance, user A could send a funds transfer request to bank B over the Internet. After the bank performs the funds transfer as per A's instructions, A could claim that s/he never sent the funds transfer instruction to the bank. Thus, A repudiates or denies her/his funds transfer instruction. The principle of *non-repudiation* defeats such possibilities of denying something, having done it. This is shown in Fig. 5.18.

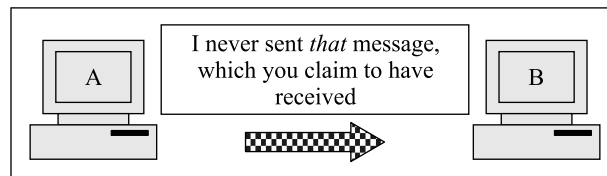


Fig. 5.18 Establishing non-repudiation

Let us now understand how these objectives can be achieved.

5.5 CRYPTOGRAPHY

In simple terms, **cryptography** is a technique of encoding (i.e., encrypting) and decoding (i.e., decrypting) messages, so that they are not understood by anybody except the sender and the intended recipient. We employ cryptography in our daily life when we do not want a third party to understand what we are saying. For instance, you can have a convention wherein *Ifmmp Kpio* actually means saying hello to your boyfriend John (that is, *Hello John!*). Here each alphabet of the original message (i.e., H, e, l, etc.) is changed to its next immediate alphabet (i.e., I, f, m, etc.). Thus, *Hello* becomes *Ifmmp*, and *John* becomes *Kpio*.

In technical terms, the process of encoding messages is called **encryption**. As we know, the original text is called **plain text**. When it is encrypted, it is called **cipher text**. The recipient understands the meaning and decodes the message to extract the correct meaning out of it. This process is called **decryption**. Figure 5.19 depicts this.

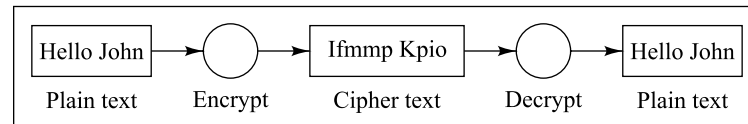


Fig. 5.19 Encryption and decryption process

Note that the sender applies the encryption algorithm and the recipient applies the decryption algorithm. The sender and the receiver must agree on this algorithm for any meaningful communication. The algorithm basically takes one text as the input and produces another as the output. Therefore, the algorithm contains the intelligence for transforming messages. This intelligence is called the **key**. Only the persons having information about the message transformation, which is an access to the key, know how to encrypt and decrypt messages.

Encryption and decryption can be done in two ways, which we shall discuss now.

Substitution Cipher

In the **substitution cipher** technique, the characters of a plain text message are replaced by other characters, numbers or symbols. **Caesar's cipher** is a special case of substitution technique wherein each alphabet in a message is replaced by an alphabet three places down the line. For instance, using Caesar's cipher, the clear text ATUL will become cipher text DWXO (because A will be replaced by D, T will be replaced by W, U will be replaced by X and L will be replaced by O). Other substitution ciphers are variations of this basic scheme. We shall therefore, discuss Caesar's cipher only.

Let us assume that we want to transform the plain text *I LOVE YOU* into the corresponding cipher text using Caesar's cipher. Figure 5.20 shows the process. As we can see, the corresponding cipher text is *L ORYH BRX*. This happens because we replace each plain text character with the corresponding character three places down the line (i.e., I with L, L with O, and so on).

Plain text	I		L	O	V	E		Y	O	U
Cipher text	L		O	R	Y	H		B	R	X

Fig. 5.20 Example of breaking Caesar's cipher

Transposition Cipher

As we discussed, substitution techniques focus on substituting a plain text alphabet with a cipher text alphabet. **Transposition cipher** techniques differ from substitution techniques in that they do not simply replace one alphabet with another; they perform some permutation over the plain text alphabets. The **Rail fence technique** is an example of transposition cipher. It uses a simple algorithm as shown in Fig. 5.21.

1. Write down the plain text message as a sequence of diagonals.
2. Read the plain text written in *Step 1* as a sequence of rows.

Fig. 5.21 Rail fence technique

Let us illustrate the rail fence technique with a simple example. Suppose that we have a plain text message, say, *Come home tomorrow*. How would we transform it into a cipher text message using the rail fence technique? This is shown in Fig. 5.22.

Original plain text message: *Come home tomorrow*

1. After we arrange the plain text message as a sequence of diagonals, it would look as follows (write the first character on the first line, i.e., *C*, then next character on the second line, i.e., *o*, then the second character on the first line, i.e., *m*, then the second character on the second line, i.e., *e*, and so). This creates a zigzag sequence, as shown after the text.

C *m* *h* *m* *t* *m* *r* *o*
o *e* *o* *e* *o* *o* *r* *w*

2. Now read the text row by row, and write it sequentially. Thus, we have *Cmhmtmroococoorm* as the cipher text.

Fig. 5.22 Example of rail fence technique

As we can see, the plain text message *Come home tomorrow* transforms into *Cmhmtmrooeoeoorw* with the help of the rail fence technique.

5.6 SYMMETRIC AND ASYMMETRIC KEY ENCRYPTION

Based on the number of keys used for encryption and decryption, cryptography can be classified into two categories, viz., symmetric key encryption and asymmetric key encryption. We will now discuss these two.

5.6.1 Symmetric Key Encryption

In this scheme, only one key is used and the same key is used for both encryption and decryption of messages. Obviously, both the parties must agree upon the key before any transmission begins, and nobody else should know about it. The example in Fig. 5.23 shows how symmetric cryptography works. Basically at the sender's end, the key changes the original message to an encoded form. At the receiver's end, the same key is used to decrypt the encoded message, thus deriving the original message out of it. IBM's **Data Encryption Standard (DES)** uses this approach. It uses 56-bit keys for encryption.

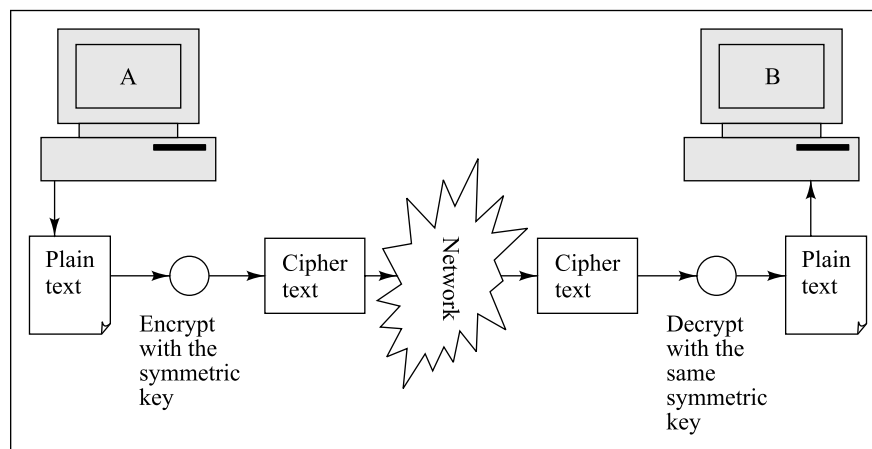


Fig. 5.23 Symmetric key encryption

In practical situations, secret key encryption has a number of problems. One problem is that of key agreements and distribution. In the first place, how do two parties agree on a key? One way is for somebody from the sender (say A) to physically visit the receiver (say B) and hand over the key. Another way is to courier a paper on which the key is written. Both are not exactly very convenient. A third way is to send the key over the network to B and ask for the confirmation. But then, if an intruder gets the message, s/he can interpret all the subsequent ones!

The second problem is more serious. Since the same key is used for encryption and decryption, one key per communicating parties is required. Suppose A wants to securely communicate with B and also with C. Clearly, there must be one key for all communications between A and B; and there must be another, *distinct* key for all communications between A and C. The same key as used by A and B cannot be used for communications between A and C. Otherwise, there is a chance that C

can interpret messages going between A and B, or B can do the same for messages going between A and C. Since the Internet has thousands of merchants selling products to hundreds of thousands of buyers, using this scheme would be impractical because every buyer-seller combination would need a separate key.

5.6.2 Asymmetric Key Encryption

The **asymmetric key encryption** technique is also called **public key cryptography**. In this type of cryptography, two *different* keys (called a **key pair**) are used. One key is used for encryption and only the other key must be used for decryption. No other key can decrypt the message – not even the original (i.e. the first) key used for encryption. The beauty of this scheme is that every communicating party needs just a key pair for communicating with any number of other communicating parties. Once someone obtains a key pair, s/he can communicate with anyone else on the Internet in a secure manner, as we shall see.

There is a simple mathematical basis for this scheme. If you have an extremely large number that has only two factors that are prime numbers, you can generate a pair of keys. For example, consider a number 10. The number 10 has only two factors, 5 and 2. If you apply 5 as an encryption factor, only 2 can be used as the decryption factor. Nothing else, even 5 itself, can do the decryption. Of course, 10 is a very small number. Therefore, with a small effort, this scheme can be broken into. However, if the number is very large, even years of computation cannot break the scheme.

One of the two keys is called **public key** and the other is **private key**. Let us assume that you want to communicate over a computer network such as the Internet in a secure manner. You would need to obtain a public key and a private key. You can generate these keys using standard algorithms. The private key remains with you as a secret. You must not disclose your private key to anybody. However, the public key is for the general public. It is disclosed to all parties that you want to communicate with. In this scheme, in fact, each party or node publishes her/his public key. Using this, a directory can be constructed where the various parties or nodes (i.e. their ids) and their corresponding public keys are maintained. One can consult this and get the public key for any party that one wishes to communicate with, by a simple table search. Suppose A wants to send a message to B without having to worry about its security. Then, A and B should each have a private key and a public key.

- A's private key should be known only to A. However, A's public key should be known to B.
- Only B should know B's private key. However, A should know B's public key.

How this works is simple:

1. When A wants to send a message to B, A encrypts the message using B's public key. This is possible because A knows B's public key.
2. A sends this message (encrypted using B's public key) to B.
3. B decrypts A's message using B's private key. Note that only B knows about her/his private key. Also note that the message can be decrypted only by B's private key and nothing else. Thus, no one else can make any sense out of the message even if one can manage to intercept the message. This is because the intruder (hopefully) does not know about B's private key. It is only B's private key that can decrypt the message.
4. When B wants to send a message to A, an exactly reverse process takes place. B encrypts the message using A's public key. Therefore, only A can decrypt the message back to its original form, using her/his private key.

This is shown in Fig. 5.24.

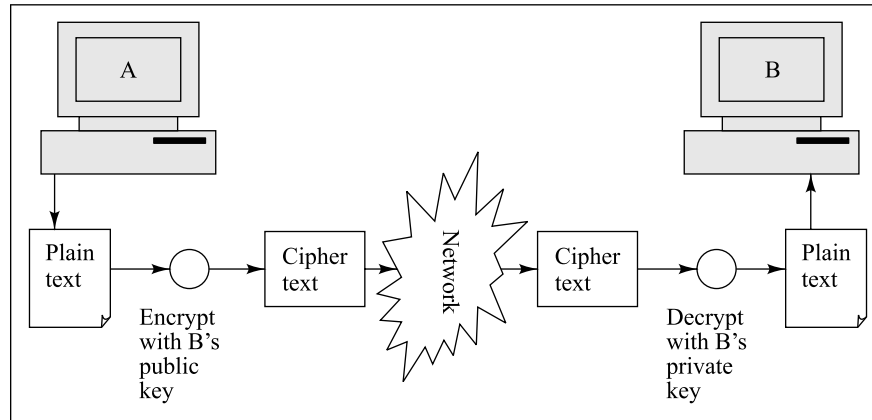


Fig. 5.24 Asymmetric key encryption

This can be shown in another way. For instance, suppose a bank needs to accept many requests for transactions from its customers. Then, the bank can have a private key-public key pair. The bank can publish its public key to all its customers. The customers can use this public key of the bank for encrypting messages before they send them to the bank. The bank can decrypt all these encrypted messages with its private key, which remains with itself. This is shown in Fig. 5.25.

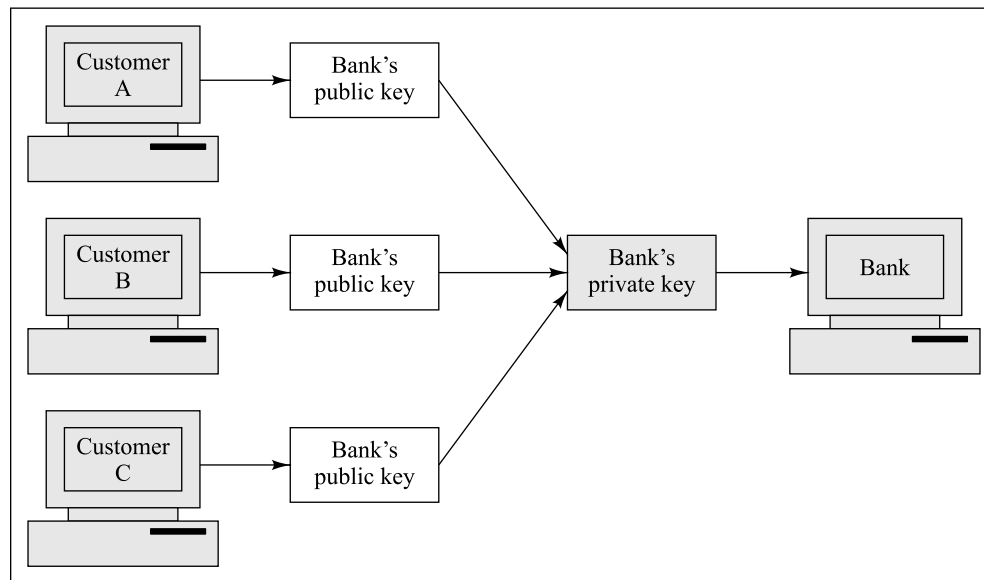


Fig. 5.25 The use of a public key-private key pair by a bank

5.6.3 The RSA Algorithm

Let us examine a practical example of public key encryption. In 1977, Ron Rivest, Adi Shamir and Len Adleman at MIT developed the first major public key cryptography system. This method

is called the Rivest-Shamir-Adleman (**RSA**) scheme. Even today, it is the most widely accepted public key solution. It solves the problem of key agreements and distribution. You do not need to send thousands of keys across the network just to arrive at the agreement. All you need is to publish your public key. All these public keys can then be stored in a database that anyone can consult. However, the private key only remains with you. It requires a very basic amount of information sharing among users.

The RSA algorithm is based on the fact that it is easy to find and multiply large prime numbers together, but it is extremely difficult to factor their product. The following portion of the discussion about RSA is a bit mathematical in nature, and can be safely skipped in case you are not interested in knowing the internals of RSA. However, if you are keen to know the mathematical details, you can continue reading. Let us, therefore, now understand how RSA works. Figure 5.26 shows an example of the RSA algorithm being employed for exchanging messages in an encrypted fashion.

This works as follows, assuming that the sender A wants to send a single character F to the receiver B. We have chosen such a simple case for ease of understanding. Using the RSA algorithm, the character F would be encoded as follows:

1. Use the alphabet-numbering scheme (i.e., 1 for A, 2 for B, 3 for C, and so on). As per this rule, for F, we would have 6. Therefore, first, F would be encoded to 6.
2. Choose any two prime numbers, say, 7 and 17.
3. Subtract 1 from each prime number and multiply the two results. Therefore, we have $(7 - 1) \times (17 - 1) = 96$.
4. Choose another prime number, say, 5. This number, which is the public key, is called K_e in the RSA terminology. Therefore, $K_e = 5$.
5. Multiply the two original prime numbers of Step 2. Therefore, we have $17 \times 7 = 119$.

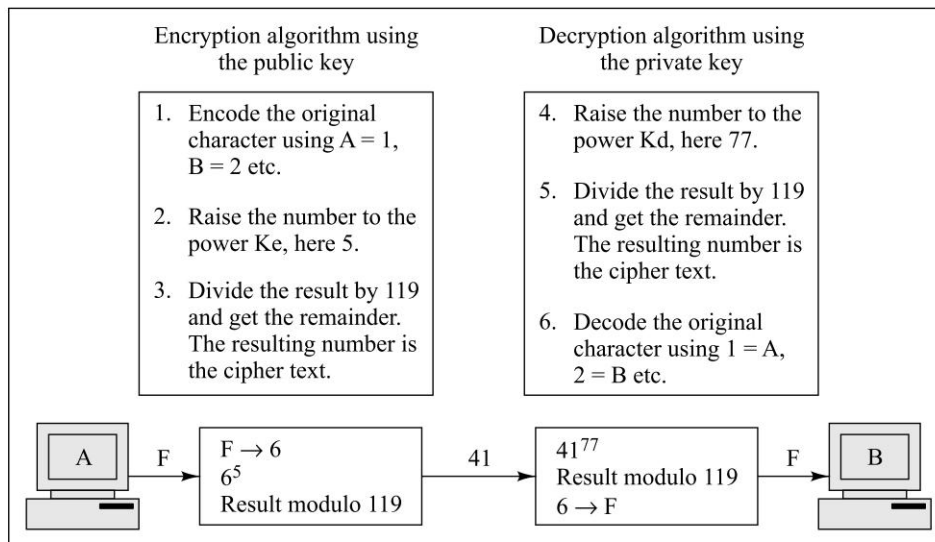


Fig. 5.26 Example of the RSA algorithm

6. Calculate the following:
(Original encoded number of Step 1)^{K_e} modulo (Number of Step 5) i.e., 6⁵ modulo 119, which is 41
7. The number thus obtained (41) is the encrypted information to be sent across the network.

At the receiver's end, the number 41 is decrypted to get back the original letter F as follows.

1. Subtract 1 from each prime number and multiply the two results, i.e., $(7 - 1) \times (17 - 1) = 96$.
2. Multiply the two original numbers, i.e., $17 \times 7 = 119$.
3. Find a number K_d such that when we divide (K_d × K_e) by 96, the remainder is 1. After a few calculations, we can come up with 77 as K_d.
4. Calculate 41^{K_d} modulo 119. That is, 41⁷⁷ modulo 119. This gives 6.
5. Decode 6 back to F from our alphabet numbering scheme.

It might appear that anyone knowing about the public key K_e (5) and the number 119 could find the secret key K_d (77) by trial and error. However, if the private key is a large number, and another large number is chosen instead of 119, it is extremely difficult to crack the secret key. This is what is done in practice.

Now imagine that A is a buyer, who wants to buy something from a merchant B. Using the above techniques, they can form a secure communication mechanism that can be used only by them. If the buyer A wants to buy something from any other merchant C, there is absolutely no problem. All A would need to know is C's public key. Similarly, all C would need to know is A's public key. On the same lines, if merchant B wants to sell something to another customer D, they simply need to know each other's public keys.

5.7 DIGITAL CERTIFICATES

This problem of key exchange or key agreement is quite severe. After a lot of thought, this problem was resolved with a revolutionary idea of using **digital certificates**. We shall study this in great detail.

Conceptually, we can compare digital certificates to other documents such as passports or driving licenses. A passport or a driving license helps in establishing our identity. For instance, my passport proves beyond doubt a variety of aspects, the most important ones being the following:

- My full name and nationality
- My date and place of birth
- My photograph and signature

A digital certificate establishes the relation between a user and her/his public key. Therefore, a digital certificate must contain the user's name and the user's public key. This will prove that a particular public key belongs to a particular user. Apart from this, what does a digital certificate contain? A simplified view of a sample digital certificate is shown in Fig. 5.27.

A **Certification Authority (CA)** is a trusted agency that can issue digital certificates. Usually, a CA is a reputed organization, such as a post office, financial institution, software company, etc. VeriSign is the world's most famous CA. In India, we have SafeScript, IDRBT, NIC, TCS, etc., as CAs.



Fig. 5.27 Example of a digital certificate

5.8 DIGITAL SIGNATURES

If the sender of a message encrypts the message with her/his private key, it forms the basis of **digital signature**, as shown in Fig. 5.28.

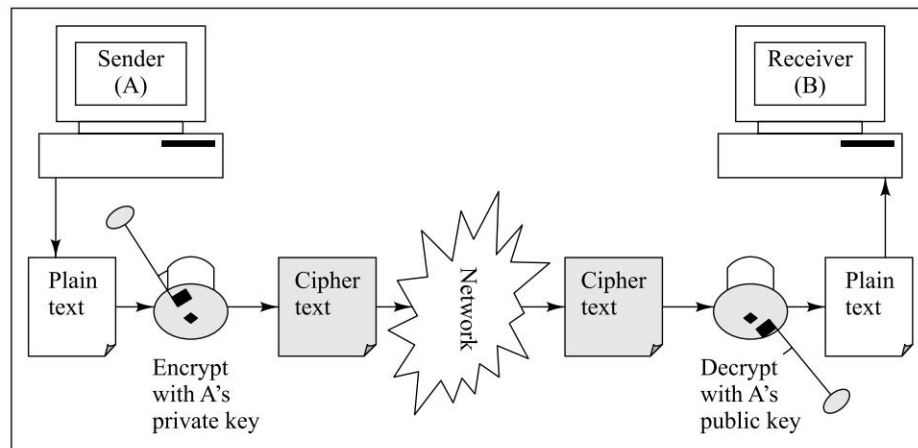


Fig. 5.28 Basis for digital signatures

Digital signatures have assumed great significance in the modern world of web commerce. Most countries have already made provisions for recognizing a digital signature as a valid authorization mechanism, just like paper-based signatures. Digital signatures have legal status now. For example, suppose you send a message to your bank over the Internet, to transfer some amount from your account to your friend's account, and digitally sign the message, this transaction has the same status as the one wherein you fill in and sign the bank's paper-based money transfer slip.

Technically, a digital signature is a two-step process. First, the sender computes the **message digest** or **hash** of a message, which is a fixed-length value. For example, if we use the **MD5** algorithm for hashing, the message digest length is 128 bits. On the other hand, if we use the **SHA-1** algorithm, the message digest length is 160 bits. Next, the sender encrypts the message digest with her/his private key using the RSA algorithm. The outcome of the process is the digital signature.

The receiver uses the sender's public key to reverse the digital signature into the message digest (say, MD-1). Then the receiver computes a message digest hash from the original message (say, MD-2). The receiver now compares the two message digests. If MD-1 = MD-2, then the receiver is assured that the message indeed came from the sender (authentication), that it was not altered (integrity), and that the sender cannot deny having sent the message (non-repudiation).

5.9 SECURE SOCKET LAYER (SSL)/TRANSPORT LAYER SECURITY (TLS)

The **Secure Socket Layer (SSL)** protocol is an Internet protocol for secure exchange of information between a web browser and a web server. It provides the following two basic security services: authentication and confidentiality. Logically, it provides a secure *pipe* between the web browser and the web server. It is now known as **Transport Layer Security (TLS)**, which is a modified version of SSL.

SSL can be conceptually considered as an additional layer in the TCP/IP protocol suite. The SSL layer is located between the application layer and the transport layer, as shown in Fig. 5.29.

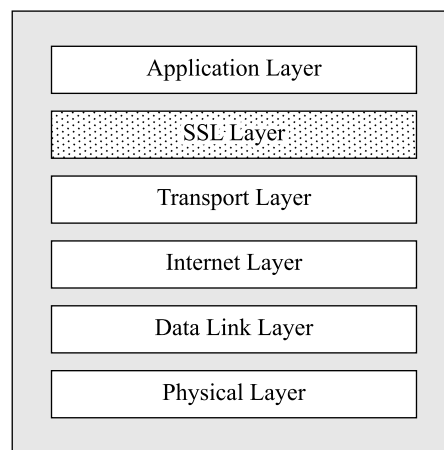


Fig. 5.29 Position of SSL in TCP/IP

As such, the communication between the various TCP/IP protocol layers is now as shown in Fig. 5.30.

As we can see, the application layer of the sending computer (X) prepares the data to be sent to the receiving computer (Y), as usual. However, unlike what happens in the normal case, the application layer data is not passed directly to the transport layer now. Instead, the application layer

data is passed to the SSL layer. Here, the SSL layer performs encryption on the data received from the application layer (which is indicated by a different color), and also adds its own encryption information header, called SSL Header (SH) to the encrypted data. We shall later study what exactly happens in this process.

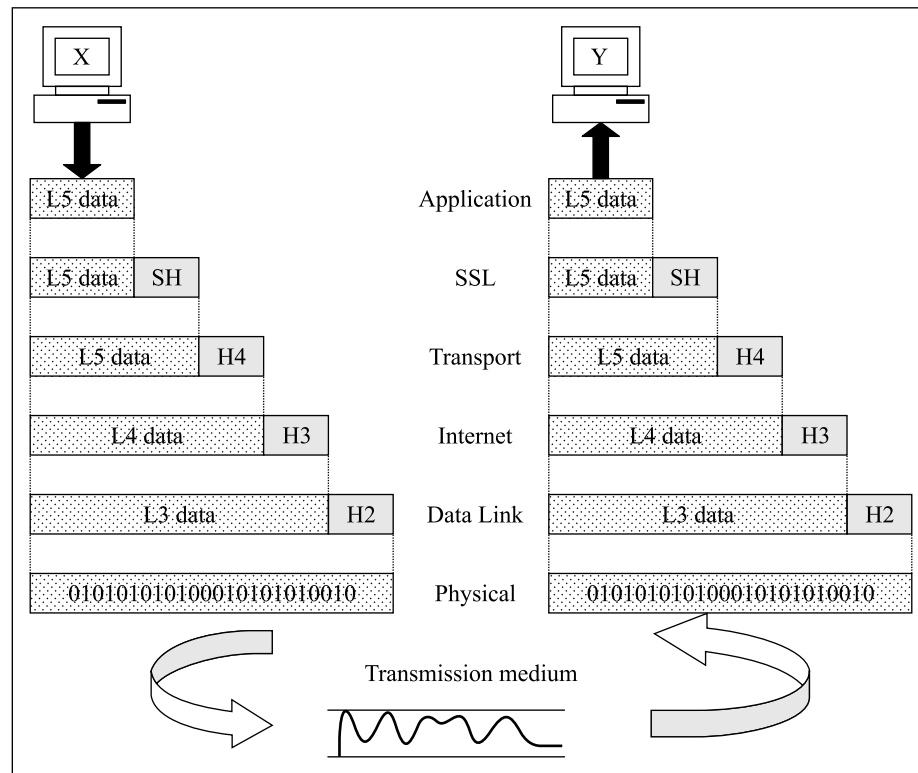


Fig. 5.30 SSL is located between application and transport layers

Thereafter, the SSL layer data (L5) becomes the input for the transport layer. It adds its own header (H4), and passes it on to the Internet layer, and so on. This process happens exactly the way it happens in the case of a normal TCP/IP data transfer. Finally, when the data reaches the physical layer, it is sent in the form of voltage pulses across the transmission medium.

At the receiver's end, the process is quite similar to how it happens in the case of a normal TCP/IP connection, until it reaches the new SSL layer. The SSL layer at the receiver's end removes the SSL Header (SH), decrypts the encrypted data, and gives the plain text data back to the application layer of the receiving computer.

SSL has three subprotocols, namely the **Handshake Protocol**, the **Record Protocol** and the **Alert Protocol**. These three subprotocols constitute the overall working of SSL. In the handshake protocol, the client and the server agree on the various parameters to be used for encryption and compression. This step also results into a one-time symmetric key that the client and server can use for actual encryption and decryption. The record protocol uses the parameters set-up in the handshake protocol to actually perform encryption. Alert protocol takes over only in the case of any possible errors or suspicious activities.

5.10 FIREWALLS

Conceptually, a **firewall** can be compared with a sentry standing outside an important person's house (such as the nation's president). This sentry usually keeps an eye on and physically checks every person that enters into or comes out of the house. If the sentry senses that a person wishing to enter the president's house is carrying a knife, the sentry would not allow the person to enter. Similarly, even if the person does not possess any banned objects, but somehow looks suspicious, the sentry can still prevent that person's entry. A firewall acts like a sentry. If implemented, it guards a corporate network by standing between the network and the outside world. All traffic between the network and the Internet in either direction must pass through the firewall. The firewall decides if the traffic can be allowed to flow, or whether it must be stopped from proceeding further. This is shown in Fig. 5.31.

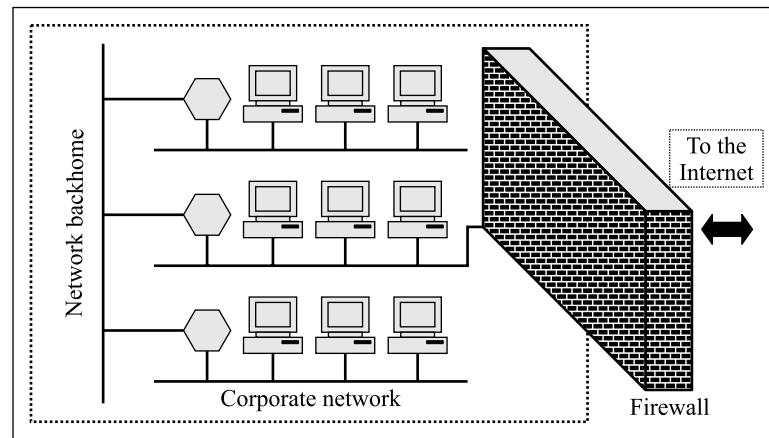


Fig. 5.31 Firewall

Of course, technically, a firewall is a specialized version of a router. Apart from the basic routing functions and rules, a router can be configured to perform the firewall functionality, with the help of additional software resources.

Based on the criteria that they use for filtering traffic, firewalls are generally classified into two types, as shown in Fig. 5.32.

Let us discuss these two types of firewalls one by one.

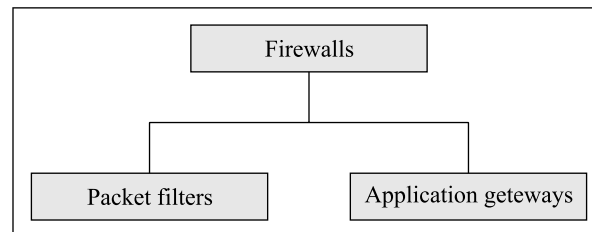


Fig. 5.32 Types of firewalls

5.10.1 Packet Filter

As the name suggests, a **packet filter** applies a set of rules to each packet, and based on the outcome, decides to either forward or discard the packet. It is also called **screening router** or **screening filter**. Such a firewall implementation involves a router, which is configured to filter packets going in either direction (from the local network to the outside world, and vice versa). The filtering rules are based on a number of fields in the IP and TCP/UDP headers such as source and destination IP addresses, IP protocol field (which identifies if the protocol in the upper transport layer is TCP or UDP), TCP/UDP port numbers (which identify the application which is using this packet such as email, file transfer or World Wide Web).

The idea of a packet filter is shown in Fig. 5.33.

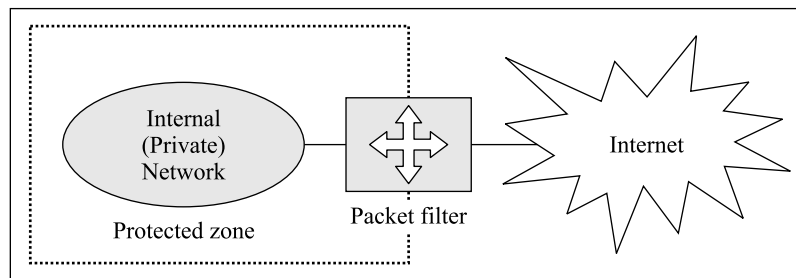


Fig. 5.33 Packet filter

Conceptually, a packet filter can be considered as a router that performs three main actions, as shown in Fig. 5.34.

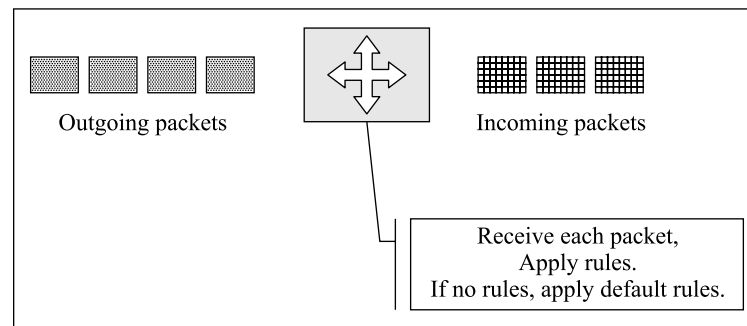


Fig. 5.34 Packet filter operation

A packet filter performs the following functions.

1. Receive each packet as it arrives.
2. Pass the packet through a set of rules, based on the contents of the IP and transport header fields of the packet. If there is a match with one of the set rules, decide whether to accept or discard the packet based on that rule. For example, a rule could specify: Disallow all incoming traffic from an IP address 157.29.19.10 (this IP address is taken just as an example), or disallow all traffic that uses UDP as the higher (transport) layer protocol.
3. If there is no match with any rule, take the default action. The default can be *discard all packets*, or *accept all packets*. The former policy is more conservative, whereas the latter

is more open. Usually, the implementation of a firewall begins with the default *discard all packets* option, and then rules are applied one-by-one to enforce packet filtering.

5.10.2 Application Gateway

An **application gateway** does not work at the level of individual packet contents. Instead, it decides to allow or ban certain protocols. For example, an application gateway can be configured to allow HTTP and SMTP but block FTP and TELNET, as conceptually shown in Fig. 5.35.

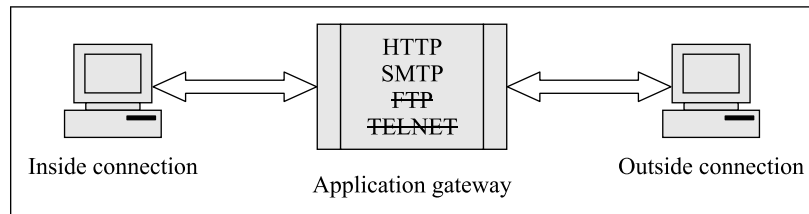


Fig. 5.35 Application gateway

5.11 EMAIL SECURITY

There are three main email security protocols: **Privacy Enhanced Mail (PEM)**, **Pretty Good Privacy (PGP)** and **Secure MIME (S/MIME)**. Of these, we will discuss the first two.

5.11.1 Privacy Enhanced Mail (PEM)

The **Privacy Enhanced Mail (PEM)** standard was initially developed by the Internet Research Task Force (IRTF) and Privacy Security Research Group (PSRG). They then handed over the PEM standard to the Internet Engineering Task Force (IETF) PEM Working Group. PEM is described in four specification documents, which are RFC numbers 1421 to 1424. PEM supports the three main cryptographic functions of encryption, non-repudiation, and message integrity, as shown in Fig. 5.36.

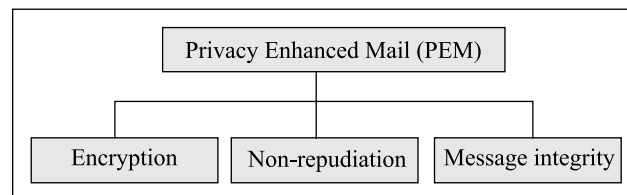


Fig. 5.36 Security features offered by PEM

Four broad steps in PEM are illustrated in Fig. 5.37. As shown, PEM starts with a **canonical conversion**, which is followed by digital signature, then by encryption and finally, **Base-64 encoding**.

PEM allows for three security options when sending an email message. These options are as follows:

- Signature only (Steps 1 and 2)
- Signature and Base-64 encoding (Steps 1, 2 and 4)
- Signature, encryption and Base-64 encoding (Steps 1 to 4)

Let us now discuss the four steps shown earlier in the figure. Note that the receiver has to perform these four steps in the reverse direction to retrieve the original plain text email message.

Step 1: Canonical conversion There is a distinct possibility that the sender and the receiver of an email message use computers that have different architectures and operating systems. This is because the Internet works on any computer that has a TCP/IP stack, regardless of its architecture or operating system. Therefore, it is quite possible that the same thing is represented differently in these different computers. For example, in the MS-DOS operating system, a *new line* (i.e., the result of the keyboard's *Enter* key) is represented by two characters, which is not the case in an operating system such as UNIX, wherein it is a single character. This can create problems when creating message digests, and therefore, digital signatures. For instance, the message digest for an email message composed on a computer that runs MS-DOS can differ from that on a computer that runs UNIX, as the input for the message digest creation is not the same in the two cases.

Consequently, PEM transforms each email message into an abstract, canonical representation. This means that regardless of the architecture and the operating system of the sending and the receiving computers, the email message always travels in a uniform, independent format.

Step 2: Digital signature This is a typical process of digital signature. It starts by creating a message digest of the email message using an algorithm such as MD5 or SHA-1. The message digest thus created is then encrypted with the sender's private key to form the sender's digital signature.

Step 3: Encryption In this step, the original email and the digital signature are encrypted together with a symmetric key. For this, the DES or DES-3 algorithm is used.

Step 4: Base-64 encoding This is the last step in PEM. The Base-64 encoding (also called **Radix-64 encoding** or **ASCII armour**) process transforms arbitrary binary input into printable character output. This is required because otherwise the email message may have characters beyond the basic ASCII character set, which cannot be handled by the SMTP protocol.

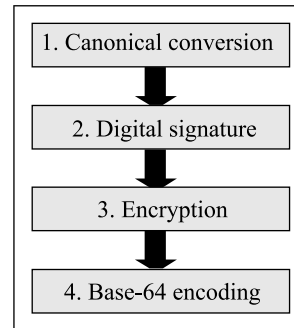


Fig. 5.37 PEM operations

5.11.2 Pretty Good Privacy (PGP)

Phil Zimmerman is the father of the **Pretty Good Privacy (PGP)** protocol. He is credited with the creation of PGP. The most significant aspects of PGP are that it supports the basic requirements of cryptography, is quite simple to use, and is completely free, including its source code and documentation. PGP has become extremely popular and is far more widely used, as compared to PEM. The email cryptographic support offered by PGP is shown in Fig. 5.38.

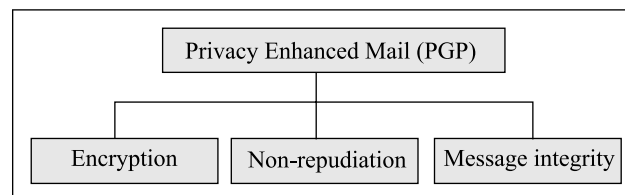


Fig. 5.38 Security features offered by PGP

In PGP, the sender of the message needs to include the identifiers of the algorithm used in the message, along with the value of the keys. Five broad steps in PEM are illustrated in Fig. 5.39. As shown, PGP starts with a digital signature, which is followed by compression, encryption, digital enveloping and finally, Base-64 encoding.

PGP allows for four security options when sending an email message. These options are as follows:

- Signature only (Steps 1 and 2)
- Signature and Base-64 encoding (Steps 1, 2 and 5)
- Signature, encryption, enveloping and Base-64 encoding (Steps 1 to 5)

Let us discuss the five steps in PGP now. Note that the receiver has to perform these four steps in the reverse direction to retrieve the original plain text email message.

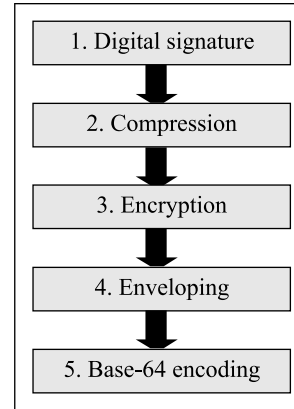


Fig. 5.39 PGP operations

Step 1: Digital signature This is a typical process of digital signature. In PGP, it consists of the creation of a message digest of the email message using the SHA-1 algorithm. The resulting message digest is then encrypted with the sender's private key. The result is the sender's digital signature.

Step 2: Compression This is an additional step in PGP. Here, the input message as well as the digital signature are compressed together to reduce the size of the final message that will be transmitted. For this, the **Lempel-Ziv algorithm** is used.

Step 3: Encryption In this step, the compressed output of Step 2 (i.e., the compressed form of the original email and the digital signature together) are encrypted with a symmetric key. For this, generally the IDEA algorithm is used.

Step 4: Digital enveloping In this case, the symmetric key used for encryption in Step 3 is now encrypted with the receiver's public key. The output of Step 3 and Step 4 together form a **digital envelope**.

Step 5: Base-64 encoding The output of Step 4 is Base-64 encoded now, as described earlier.

When a sender wants to send an email message to a single recipient, there is not too much of a problem. Complexities are introduced when a message has to be sent to multiple recipients. If Alice needs to correspond with 10 people, Alice needs the public keys of all these 10 people. Hence, Alice is said to need a **key ring** of 10 public keys. Additionally, PGP specifies a ring of public-private keys. This is because Alice may want to change her public-private key pair, or may want to use a different key pair for different groups of users (e.g., one key pair when corresponding with someone in her family, another when corresponding with friends, a third in business correspondence, etc.). In other words, every PGP user needs to have two sets of key rings: (a) A ring of her own public-private key pairs, and (b) A ring of the public keys of other users.

The concept of key rings is shown in Fig. 5.40. Note that in one of the key rings, Alice maintains a set of key pairs, while in the other, she just maintains the public keys (and not key pairs) of other users. Obviously, she cannot have the private keys of the other users. Similarly, other users in a PGP system will have their own two key rings.

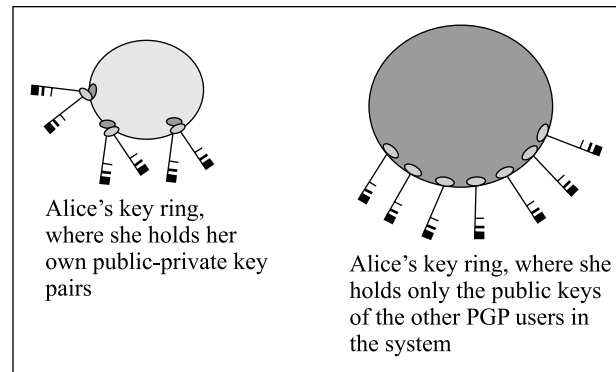


Fig. 5.40 Key rings maintained by a user in PGP

SUMMARY

Compression helps in making data storage and transmission efficient. Using compression techniques, we can reduce the amount of data to be transmitted from the source to the destination, without actually losing any data at all. We can substitute some kind of control information to represent repetitive data and transmit it to achieve compression. It also reduces storage requirements. There are three ways to achieve data compression: (a) Simple coding scheme, (b) Based on the context of the symbols and (c) Based on the relative frequencies or occurrences of the symbols. The simple coding scheme assigns codes to long portions of text and sends the codes instead of the text to achieve compression. However, for this to work, both the sender and the receiver have to agree upon a coding scheme, which is not always possible in practical situations.

In the compression technique that is based on the context of the symbols, there are two main variations: (i) Null suppression and (ii) Run length encoding. In null suppression, blank spaces in the data are not sent out, and instead their count is sent to achieve compression. In run length encoding, not only repeating spaces, but also other repeating characters are not sent, and instead, their count is sent. In both the cases, the receiver looks at the count and expands the data to the original format.

Also called statistical compression, the compression based on the relative frequencies of the symbols depends on how many times the symbol occurs. That is, short codes are used for compressing symbols that occur more frequently, and long codes for symbols that occur infrequently. This is the most sophisticated compression technique available. Techniques/coding schemes such as Morse code, Huffman code, modified Huffman code, and Lempel-Ziv code use this principle.

Cryptography is a technique of encoding and decoding messages, so that they are not understood by anybody except the sender and the intended recipient. The sender encodes the message (a process called encryption) and the receiver decodes the encrypted message to get back the original message (a process called decryption).

Encryption can be classified into secret key encryption and public key encryption. In secret key encryption, the same key is used for encryption and decryption. Thus, for every communicating pair of parties (called participants), there must be a unique secret key. In public key encryption,

each participant has a pair of keys (one private, the other public). This is also called asymmetric encryption because the same key cannot do both encryption and decryption. If encryption is done using a public key, decryption must be done using a private key alone, and vice versa. The private key remains private with the participant; the public key is freely distributed to the general public. For encryption, anybody can use the participant's public key. Only the participant can decrypt the message, since it alone knows its own private key.

Digital signature has become a very critical technology for modern secure data communications. It involves a very intelligent combination of public key encryption techniques to achieve secure communication. Firewalls are used to protect an organization's network from outside attacks. Firewalls can be configured at packet level or at the protocol level. Email security can be handled primarily via PEM and PGP. The SSL protocol ensures message encryption in browser-based interactions.

KEY TERMS AND CONCEPTS

Alert protocol	Lossless compression
Application gateway	Lossy compression
Asymmetric key encryption	Message digest
Authentication	Message integrity
Compression	Non-repudiation
Confidentiality	Packet filter
Cryptography	Plain text
Data Encryption Standard (DES)	Privacy Enhanced Mail (PEM)
Decompression	Pretty Good Privacy (PGP)
Decryption	Private key
Digital certificates	Public key
Digital signature	Public key encryption
Encryption	Record protocol
Firewall	RSA algorithm
Handshake protocol	Secure Socket Layer (SSL)
Hash	Simple encoding scheme
Huffman code	Statistical compression
Key	Symmetric key encryption
Key pair	Transport Layer Security (TLS)
Lempel-Ziv code	

QUESTIONS

True/False

1. In simple coding scheme a small amount of control characters to denote that what is being sent is not any numerical data, but a code, also needs to be sent.
2. Null suppression is a generalization of the scheme run length encoding.
3. By using statistical compression the length of the data is reduced dramatically.
4. Asymmetric key encryption uses 1 key.

5. RSA is a symmetric key encryption technique.
6. Firewalls cannot work at packet level.
7. Cryptography is used only for encoding the messages.
8. The sender and recipient of the message decide on an encoding and decoding scheme and use it for communication
9. The encrypted message is called cipher text.
10. The intelligence of the algorithm for transforming messages is called the key.
11. In asymmetric key encryption, both the parties must agree upon the key before any transmission begins, and nobody else should know about it.
12. IBM's Data Encryption Standard (DES) uses asymmetric key encryption.
13. In asymmetric key cryptography, one key from the pair of keys is used for encryption and only the other key must be used for decryption.
14. A message digest is also called hash.

Multiple-Choice Questions

1. The approach wherein the sender and the recipient agree on a common set of codes is called _____.
 (a) simple coding scheme
 (b) replacement technique
 (c) technique based on frequency of symbols
 (d) technique based on context of symbols
2. In _____, we replace nulls or spaces with their occurrence number.
 (a) RLE
 (b) null suppression
 (c) simple encoding scheme
 (d) None of the above
3. _____ is an extension of _____.
 (a) RLE, null suppression
 (b) Null suppression, RLE
 (c) Simple encoding scheme, RLE
 (d) Simple encoding scheme, null suppression
4. Morse code uses _____ and _____.
 (a) 1, 0
 (b) X, Y
 (c) dots, dashes
 (d) on, off
5. The Huffman code uses _____ length encoding characters.
 (a) fixed
 (b) variable
 (c) constant
 (d) None of the above
6. The _____ replaces repeating word or string patterns with pointers.
 (a) Huffman code
 (b) Morse code
 (c) Lempel-Ziv code
 (d) Caesar Cipher
7. Data can be lost in _____.
 (a) lossless compression
 (b) Morse code
 (c) encryption
 (d) lossy compression
8. In SSL protocol, the first subprotocol is called _____.
 (a) handshake protocol
 (b) record protocol
 (c) alert protocol
 (d) None of these

9. _____ is compromised when an unauthorized person gets access to a message.
 - (a) Integrity
 - (b) Confidentiality
 - (c) Non-repudiation
 - (d) Authenticity
10. _____ is useful to determine a user's identity.
 - (a) Integrity
 - (b) Confidentiality
 - (c) Non-repudiation
 - (d) Authenticity
11. _____ of a message is compromised when it is changed during transmission.
 - (a) Integrity
 - (b) Confidentiality
 - (c) Non-repudiation
 - (d) Authenticity
12. In _____ encryption, the same key is used for encryption and decryption.
 - (a) symmetric
 - (b) asymmetric
 - (c) public key
 - (d) None of the above
13. In asymmetric key encryption, the message is encrypted with the receiver's _____.
 - (a) private key
 - (b) key pair
 - (c) symmetric key
 - (d) public key
14. A _____ is the basis for digital signatures.
 - (a) unsigned message digest
 - (b) signed message digest
 - (c) asymmetric key
 - (a) key pair
15. _____ is the basis for non-repudiation.
 - (a) Confidentiality
 - (b) Message digest
 - (c) Digital signature
 - (d) Digital Certificate
16. When a digital signature is created, _____ is most important.
 - (a) sender's private key
 - (b) sender's public key
 - (c) receiver's private key
 - (d) receiver's public key
17. When a digital signature is verified, _____ is most important.
 - (a) sender's private key
 - (b) sender's public key
 - (c) receiver's private key
 - (d) receiver's public key
18. When a message digest is created, _____ is most important.
 - (a) sender's private key
 - (b) sender's public key
 - (c) receiver's private key
 - (d) None of the above

Detailed Questions

1. Explain why data compression is necessary and list down its three approaches.
2. What is a simple coding scheme?
3. Explain the term null suppression technique.
4. Explain the differences between Huffman code and Lempel-Ziv code.
5. Describe the main kinds of risks involved in data communication over a network.
6. Explain how the symmetric key and asymmetric key encryptions work.
7. Discuss the term *digital signature*.
8. Explain how firewalls work.
9. Explain the various email protocols.
10. Discuss the idea of the SSL protocol.

6

Transmission Media

6.0 INTRODUCTION

Transmission media are the physical infrastructure components, which carry data from one computer to another. They are at the basis of data communications. Examples of simple forms of transmission media are telephone wires that connect telephones to the central offices (i.e., telephone exchanges), and coaxial cables that carry cable television transmission to homes. Transmission media need not always be in the form of a physical wire—they can be invisible as well. We shall study the different types of transmission media in this chapter.

Broadly, all transmission media can be divided into the following two main categories: **Guided** and **Unguided**, as shown in Figure 6.1.

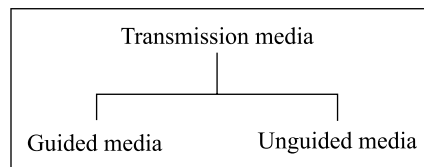


Fig. 6.1 Categories of transmission media

6.1 GUIDED MEDIA

Guided Media can be further subdivided into three main types as shown in Fig. 6.2.

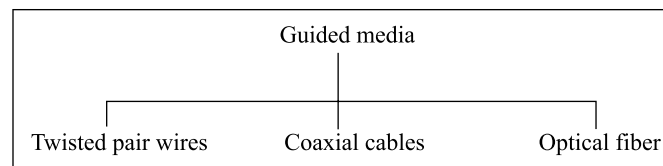


Fig. 6.2 Types of guided media

Guided media are typically based on some physical cable. **Twisted pair wires** and **coaxial cables** carry signals in the form of electrical current, whereas **optical fibers** carry signals in the form of light. We will now study these one by one.

6.1.1 Twisted Pair Wire

There are two classes of twisted pair wires as shown in Fig. 6.3.

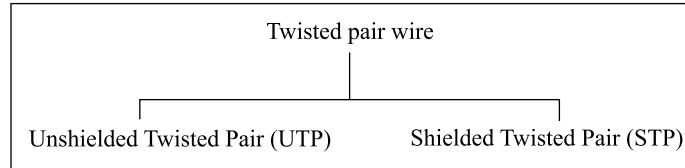


Fig 6.3 Categories of twisted pair wire

Unshielded Twisted Pair (UTP)

This is the most commonly used medium today, due to its usage in the telephone system. This cable can carry both voice as well as data. It consists of two conductors (usually copper). Earlier the wires used to be kept parallel; however, this resulted in far greater levels of noise. Hence, the wires are normally twisted as shown Fig. 6.4. This results in the reduction of noise to a great extent, although it is not eliminated completely. The copper conductors are covered by PVC or some other insulator.

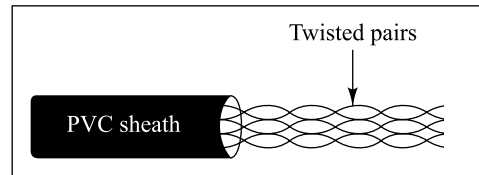


Fig. 6.4 Unshielded Twisted Pair (UTP)

UTP is flexible, cheap and easy to install. Electronic Industries Association (EIA) has developed standards for UTP cables. They are given in Fig. 6.5. Each one is manufactured differently for a specific purpose.

Category	Usage
1	This is the basic cable used in a telephone system. This is suitable for voice communication, but unsuitable for data communication, except for very low speed.
2	Suitable for voice and data communication up to the speed of 4 Mbps.
3	Can carry voice and data up to 10 Mbps. It requires minimum three twists per foot. Today, these are more regularly used in telephone networks.
4	These are similar to Category 3, but can handle data up to 16 Mbps.
5	Can handle data speed of 100 Mbps.

Fig. 6.5 Categories of UTP

Today, UTPs of higher categories are also used in computer networks due to higher speeds and reliability.

Shielded Twisted Pair (STP)

In this case, apart from the insulator, the twisted pair wire itself is carried by a metal shield, and finally by a plastic cover as shown in Fig. 6.6. The metal shield prevents penetration of electromagnetic noise. It also helps eliminate **crosstalk**, an effect wherein one wire picks up some of the signals travelling on the other wire. This effect can be felt during telephone conversations sometimes, when we hear other conversations in the background during our call. The shield prevents such unwanted sounds.

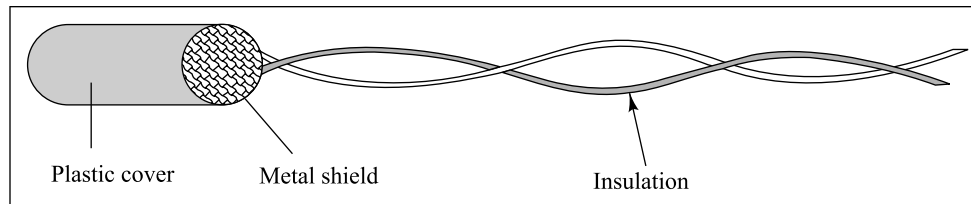


Fig. 6.6 *Shielded Twisted Pair (STP)*

As we saw, the shield reduces noise and crosstalk (noticed in telephone calls sometimes) substantially. However, STP is more expensive than UTP.

6.1.2 Coaxial Cable

Coaxial cable (also called **coax**) has an inner central conductor surrounded by an insulating sheath, which in turn is enclosed in an outer conductor (shield). This acts not only as a second conductor for completing the circuit but also acts as a shield against noise. This outer conductor is covered by a plastic cover. This whole arrangement is shown in Fig. 6.7.

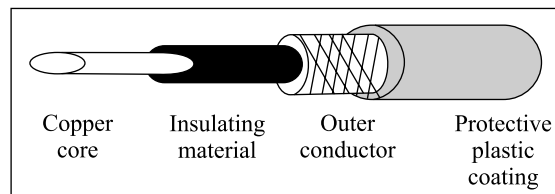


Fig. 6.7 *Coaxial cable*

As compared to UTP or STP, coaxial cable is more expensive, less flexible and more difficult to install in a building where a number of twists and turns are required. However, it is much more reliable and can carry far higher data rates. Coaxial cables are divided into various categories depending upon the thickness and size of the shields, insulators and the outer coating, besides other considerations. They are commonly used by cable companies to carry cable transmissions.

Various coaxial cable standards are RG-8, RG-9, RG-11, RG-58 and RG-59.

6.1.3 Optical Fiber

In case of **optical fibers**, light is used as a means of signal propagation, instead of electrical signals. Optical fibers are made up of glass fibers that are enclosed in a plastic jacket. This allows fibers to bend and not break. A transmitter at the sender's end of the optical fiber sends a light emitting diode (LED) or laser to send pulses of light across the fiber. A receiver at the other end makes use of a light-sensitive transistor to detect the absence or presence of light to indicate a 0 or a 1.

In the fiber, the cladding covers the core (fiber) depending upon the size. Commonly, the fiber is covered by a buffer layer, which protects it from moisture. Finally, an outer jacket surrounds the entire cable. This is shown in Fig. 6.8.

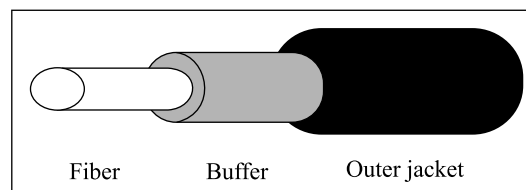


Fig. 6.8 *Optical fiber*

The core and cladding must be extremely pure, and should not vary in size or shape.

The outer jacket, however, can be made of Teflon, plastic or metal. Teflon is more expensive than plastic, but both do not provide structural strength to the fiber. Metal can provide that, but it is very heavy and expensive. The choice depends upon many of these factors.

For data transmission to occur, the sending device must be capable of inducing data bits 0 to 1 into the light source and the receiving device must have a photosensitive cell called photodiode, which would translate this light back again into data bits.

There are two types of light sources: Light Emitting Diode (LED) and Injection Laser Diode (ILD). LED is cheaper but it provides unfocused light that hits the boundaries of the core at different angles and diffuses over distance. This is why LED is used only for a short distance. Laser, however, can provide a more focused beam, which retains its character over a long distance.

Light travels at a speed of 300,000 km/second or 186,000 miles/second in a straight direction as long as it moves through a uniform substance. However, when the medium through which the light passes changes, its speed and therefore, the direction, also changes suddenly. A spoon half-immersed in a glass of water appears bent due to the same reason. This phenomenon is called **refraction**.

If we change the direction of the incoming light, the direction of the refracted light also changes. Finally, we reach a stage, where the refracted light becomes horizontal as shown in Figs 6.9(a) and (b). After this stage, if the angle still changes, the light is not refracted; instead, it is **reflected**, as shown in Fig. 6.9(e).

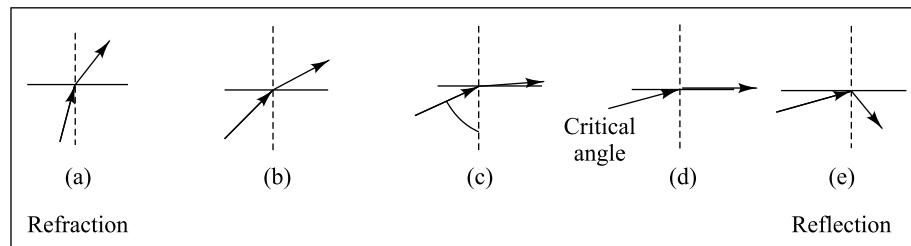


Fig. 6.9 Reflection and refraction

Optical fibers use reflection to guide the light through the optical fiber. A glass or plastic core is surrounded by a cover of less dense glass or plastic. The difference in the densities of the two is adjusted such that reflection, instead of refraction, occurs.

There are different modes of propagation, depending upon the physical characteristics of the fiber and the light source. These are shown in Fig. 6.10.

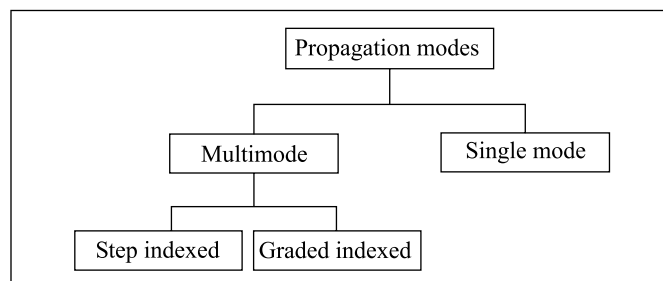


Fig. 6.10 Propagation modes

Multimode

In this case, LED is mostly used as a light source. Therefore, multiple beams pass through the core in different paths. The structure of the core decides the way they travel.

- **Step Index**

Figure 6.11 illustrates this.

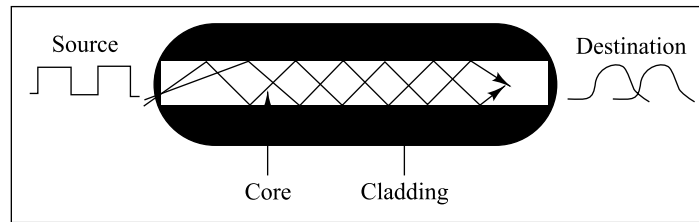


Fig. 6.11 *Multimode step index fiber*

In this case, the core has one density and the cladding has another. Therefore, at the interface, there is a sudden change. That is why it is called **step index**. Multiple beams take different paths on reflection as shown in the figure. The beam, which strikes the core at a smaller angle, has to be reflected many more times than the beam which strikes the core at a larger angle to reach the other end. This means that at the destination, all beams do not reach simultaneously. This creates diffusion and confusion in terms of interpretation. Due to this, this scheme is used only for a shorter distance.

- **Graded Index**

In this case, the core itself is made of a material of varying densities. The density is highest at the core and gradually decreases towards the edge. Therefore, a beam goes through gradual refraction giving rise to a curve as shown in Fig. 6.12.

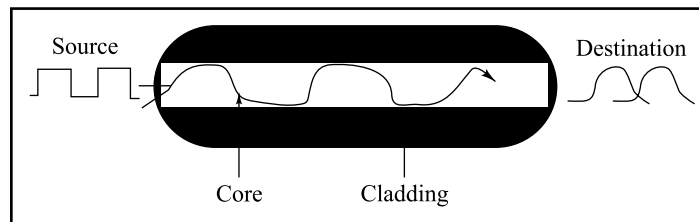


Fig. 6.12 *Multimode graded index fiber*

In this case also, different beams result in different curves or waveforms, but the geometry suggests that they increase at regular intervals. Therefore, if we place the receiver at one of these intersection points, the accuracy of the signal reconstruction enhances substantially.

Single Mode

This uses a highly focused light beam and travels more or less horizontally. The fiber core diameter in this case is much smaller than multimode and also has far lower density. This decrease results in a critical angle close to 90 degrees to make the projection of different beams very close to horizontal. As the propagation of different beams is more or less similar, the delays are negligible and the reconstruction of the signal is that much easier.

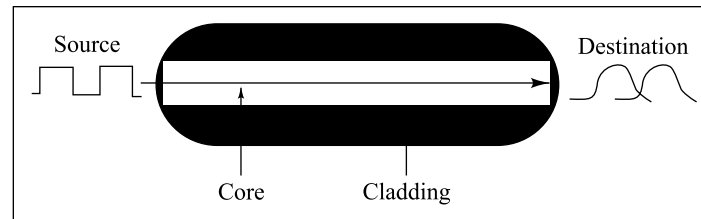


Fig. 6.13 *Single mode fiber*

Optical fibers provide the following advantages over twisted pair wires and coaxial cables:

1. **Resistance to noise** Optical fibers use light rays for communication, rather than electricity. Therefore, noise is not an issue here. The only possible source of interference is the external light, which is also blocked by the outer jacket of the optical fiber, thus providing resistance to interference.
2. **Huge bandwidth** The bandwidth offered by optical fibers is huge as compared to twisted pair wires and coaxial cables.
3. **Higher signal carrying capacity** Unlike twisted pair wires and coaxial cables, the signals carried by optical fibers travel longer distances (several kilometers) without requiring regeneration.

Along with all the advantages mentioned above, optical fibers also have some disadvantages, as summarized below:

1. **Fragility** Optical fibers are more delicate than twisted pair wires and coaxial cables, so the chances of them breaking are higher.
2. **Cost** Optical fibers are quite expensive because these have to be extremely pure to be able to carry signals without damage over longer distances. The laser light source itself is also very costly, as compared to electrical signal generators.
3. **Maintenance overhead** Optical fibers need more maintenance, hence incur more overheads, as compared to twisted pair wires and coaxial cables.

6.2 UNGUIDED MEDIA

Unguided media also called **wireless communication** transport electromagnetic waves without using a physical conductor. The signals propagate through air (or sometimes water). The communication band for unguided media is as shown in Fig. 6.14. We shall concentrate on radio communications.

Given below is the description of various wireless bands.

- **Very Low Frequency (VLF)** waves propagate as surface waves, usually through air, but sometimes also through water. VLF waves do not suffer attenuation, but are affected by atmospheric noise. VLF waves are usually used for long-range radio navigation and submarine communication.
- **Low Frequency (LF)** waves also propagate as surface waves. Attenuation is higher during the daytime. These waves are used for long-range radio navigation or navigation locators.
- **Middle Frequency (MF)** waves rely on line-of-sight antennas to increase and control absorption problems. These waves are used for AM radio, radio direction finding and emergency frequencies.

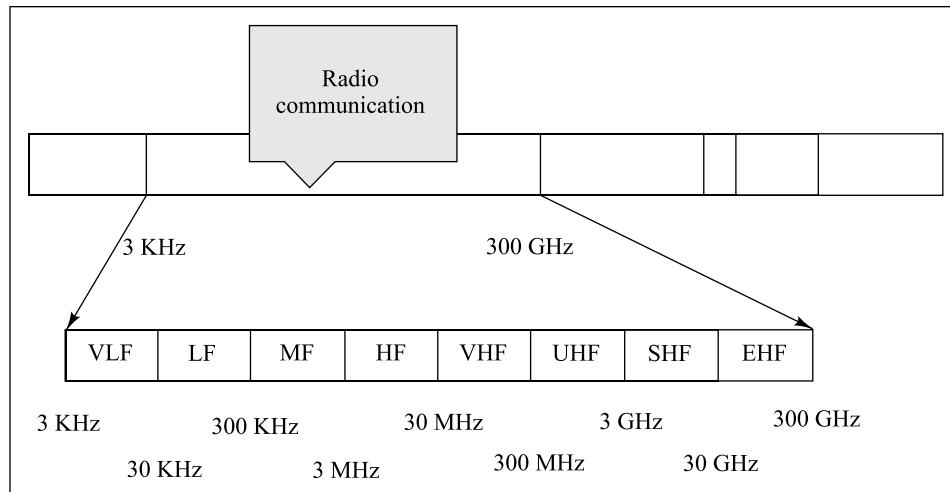


Fig. 6.14 Radio communications band

- **High Frequency (HF)** waves are used for amateur radio, citizen's band radio, international broadcasting, military communication, telephone, telegraph and facsimile communication.
- **Very High Frequency (VHF)** waves use line-of-sight propagation, and are used for VHF television, FM radio, aircraft AM radio and aircraft navigation.
- **Ultra High Frequency (UHF)** waves use line-of-sight propagation. They are used for television, mobile phone, cellular radio, paging and microwave links.
- **Super High Frequency (SHF)** waves are transmitted as either line-of-sight or into the space. They are used for terrestrial and satellite microwave and radar communication.
- **Extremely High Frequency (EHF)** waves are transmitted into space and are used for scientific applications such as radar, satellite and experimental communication.

6.2.1 Microwave Communication

As shown in Fig. 6.15, **microwaves** use the *line-of-sight* method of propagation, as the signals do not travel along the surface of the earth. Therefore, the two antennas must be in a *straight line*, able to *look* at each other without any obstacle in between. The taller the antenna, the more is the distance that these waves travel. Usually, the antennas are positioned on mountains to avoid obstacles. Microwave signals travel only in one direction at a time. This means that for two-way communication such as in telephony, two frequencies need to be allocated. At both ends, a transceiver is used which is a combination of a transmitter and a receiver, operating at the two respective frequencies. Therefore, only one antenna can serve both the functions and cover both the frequencies. Repeaters are used along with the antennas to enhance the signal. The data rates offered are 1 Mbps to 10 Gbps.

Before fiber optics, for decades, microwave communication formed an important part of telephony. The company Sprint was based on optical fiber communication, whereas the company MCI was completely based on microwave communication initially. The name MCI, in fact, stood for Microwave Communications Inc. Lately, MCI is also switching to optical fiber communication. However, microwave communication will continue to be useful as it allows one to communicate from *anywhere*. Microwave is also relatively inexpensive.

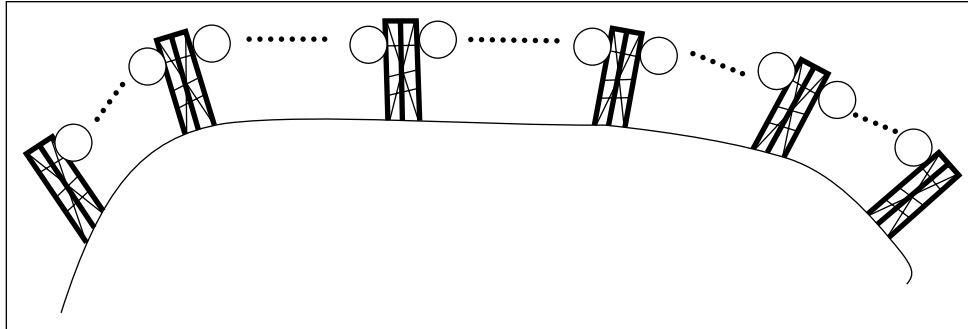


Fig. 6.15 Terrestrial microwave communication

6.2.2 Satellite Communication

Satellite communication is similar to terrestrial microwave communication, except that the satellite acts as one of the stations. Figure 6.16 illustrates this. The satellite performs the functions of an antenna and the repeater together. For instance, as Fig. 6.16 illustrates, ground station A can send information to ground station B via the satellite.

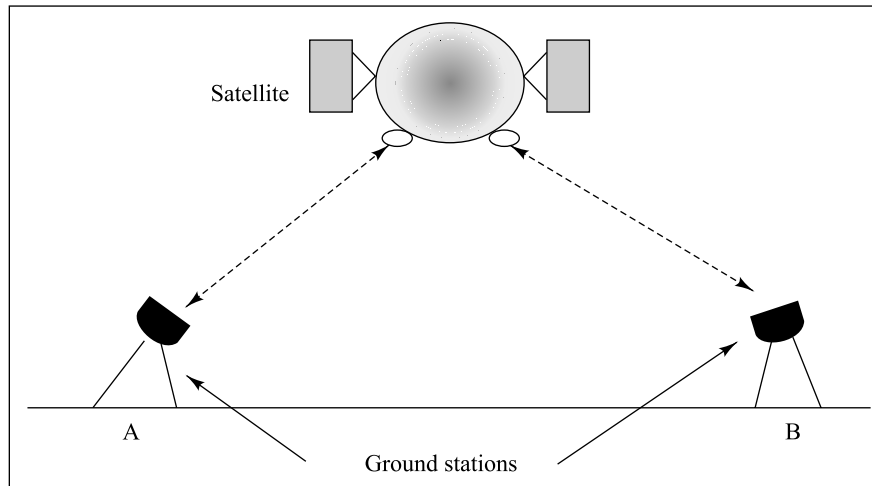


Fig. 6.16 Satellite microwave communication

This, however, poses a problem. If the earth along with its ground stations is revolving and the satellite is stationary, the sending and receiving earth stations and the satellite will be out of sync as time passes by. Therefore, normally **geosynchronous satellites** are used, which move at the same Revolutions Per Minute (RPM) as the earth in the same direction exactly like the earth. Therefore, both the earth and the satellite complete one revolution exactly in the same time, i.e., the relative position of the ground station with respect to the satellite never changes. Therefore, movement of the earth does not matter to communicating nodes based on the earth. Using satellites, we can communicate between any two parts of the world. However, minimum three satellites are needed to cover the earth's surface entirely as shown in Fig. 6.17.

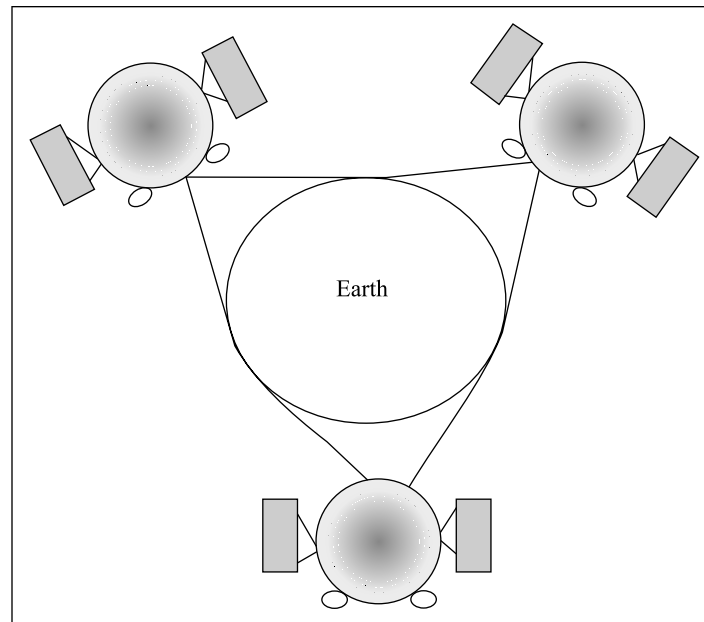


Fig. 6.17 Three satellites to cover the earth

Normally, SHF, which covers a frequency range of 3 GHz to 30 GHz, is used for satellite communications. Two frequency bands are used for signals from the earth to the satellite (called **uplink**), and from the satellite to the earth (called **downlink**).

There are three methods of communication using satellites. These three methods use principles that are similar in concept to normal wired communication. Like in the wired world, satellite communication is also based on modulation techniques. The three primary modulation techniques are as follows: (a) **Frequency Division Multiple Access (FDMA)**, (b) **Time Division Multiple Access (TDMA)** and (c) **Code Division Multiple Access (CDMA)**.

These technologies can be explained at a broad level by breaking down the title of each one.

1. The first word notifies what the access method is and the second word, *division*, notifies that it splits transmissions based on that access method. This can be explained as follows:
 - *FDMA* puts each transmission on a separate *frequency*.
 - *TDMA* assigns each transmission a certain portion of *time* on a designated frequency.
 - *CDMA* gives a unique *code* to each transmission and spreads it over the available set of frequencies.
2. The last words of each category are *multiple access*. This simply means that more than one user (multiple) can use (access) each cell.

We shall now discuss these technologies in brief.

Frequency Division Multiple Access (FDMA)

FDMA splits the total bandwidth into multiple channels. Each ground station on the earth is allocated a particular frequency group (or a range of frequencies). Within each group, the ground station can

allocate different frequencies to individual channels, which are used by different stations connected to that ground station. Before the transmission begins, the transmitting ground station looks for an empty channel within the frequency range that is allocated to it and once it finds an empty channel, it allocates it to the particular transmitting station.

This is the most popular method of communication using satellites. The transmission station on earth combines multiple signals to be sent into a single carrier signal of a unique frequency by multiplexing them, similar to how a TV transmission works. The satellite receives this single multiplexed signal. The satellite, in turn, broadcasts the received signal to the receiving earth station. The receiving station is also supposed to agree on this carrier frequency, so that it can demultiplex the received signals. The basic idea of FDMA is shown in Fig. 6.18.

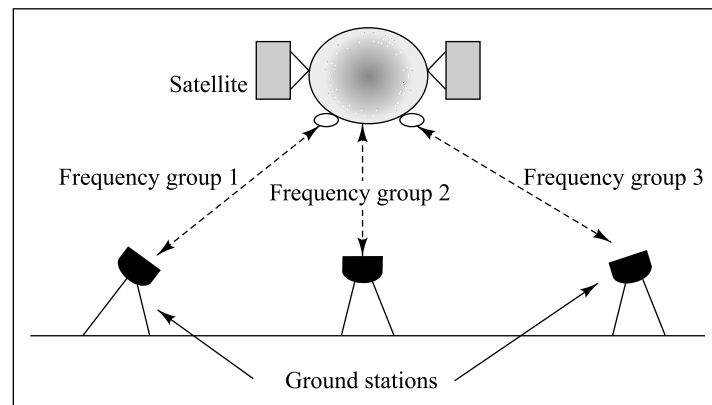


Fig. 6.18 Frequency Division Multiple Access (FDMA)

Time Division Multiplexing Access (TDMA)

Unlike FDMA, TDMA allows access to the full bandwidth of the frequency spectrum. In TDMA, each transmitter is allocated a predefined time slot. Each transmitter receives the time slot in turn. Consequently, each transmitter is allowed to transmit data for the duration of the time slot. This is shown in Fig. 6.19.

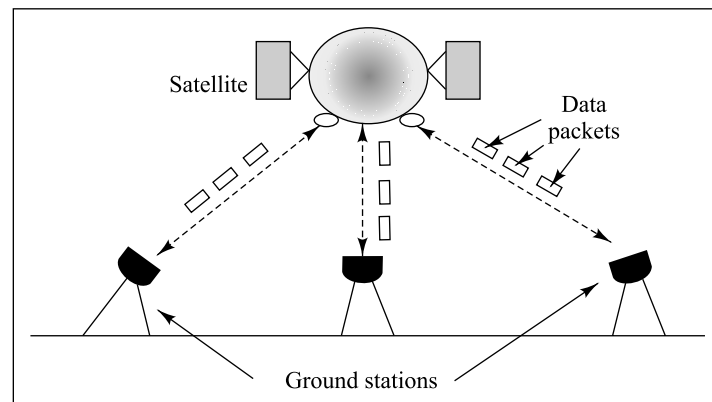


Fig. 6.19 Time Division Multiple Access (TDMA)

After FDMA, TDMA is the second most popular mechanism for communication using satellites. In case of TDMA, there is no modulation of frequencies (unlike FDMA). Instead, the transmitting earth station transmits data in the form of packets of data. These data packets arrive at the satellite one by one (hence the name TDMA). By the same logic that was discussed during TDM, TDMA is also a digital form of data transmission—this is because TDMA operates in time domain, rather than frequency domain. Bit rates of 10 Mbps to 100 Mbps are common for TDMA transmissions. This can be translated into roughly 1800 simultaneous voice calls using 64 Kbps PCM.

Code Division Multiple Access (CDMA)

As we have seen, in FDMA, a fixed frequency channel is allocated to a transmission pair (i.e., the transmitter and the receiver). In the case of TDMA, transmission takes place using predefined time slots. Nothing of this sort happens in the case of CDMA. CDMA allows any transmitter to transmit in any frequency, and at any time. Thus, it is different from both FDMA and TDMA.

An obvious question, therefore is would this not lead to confusion and mixing of different signals? It would. However, to counter this, the transmitter and the receiver agree upon a unique coding scheme (similar to encryption) before the start of the transmission. The analogy often used to explain CDMA is an international party, where there are dozens of people. All are talking at once, and all are talking in different languages that you do not understand. Suddenly, from across the room, you hear a voice speaking in your own language or one familiar to you, and your brain tunes out all the background gibberish and locks onto to that one person. Your brain understands the *code* being used by the other person, and vice versa.

Similarly, CDMA uses coding that is unique to a particular transmission and allows the receiver to disregard other transmissions on the same frequency. In this case, the coding scheme is a unique frequency with which the original signal is modulated to form the codified signal that is to be transmitted. The transmitter then codifies all its transmissions in this manner, before they are sent. Since only the receiver knows how to decode this (as agreed upon before the start of the transmission), there is no scope for confusion. This is shown in Fig. 6.20.

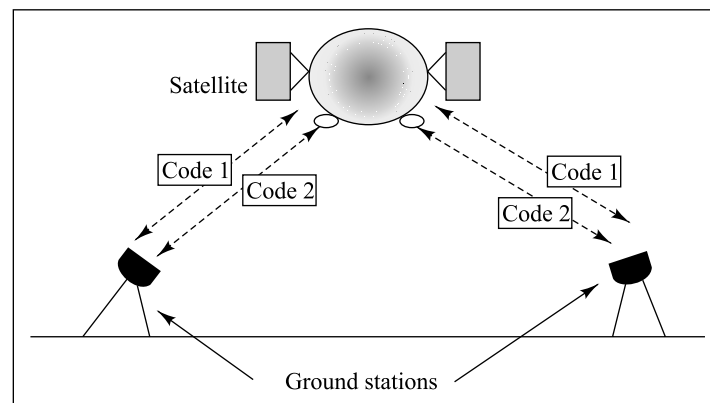


Fig. 6.20 *Code Division Multiple Access (CDMA)*

CDMA is the newest and the least used access method using satellites. It is popular in military installations, but not so much in the case of commercial applications.

6.2.3 Cellular (Mobile) Telephones

St. Louis in the US saw the emergence of the first **mobile telephone** as early as 1946. This system was difficult to operate and had a single channel for both sending and receiving communication. You had to push a button to enable the transmitter and disable the receiver. This half-duplex system, known as **push-to-talk system**, was installed in the big cities in the 1950s. Even today, taxis, CB radio, etc., use the same technology.

The second development took place in the 1960s. This was called **Improved Mobile Telephone System (IMTS)**. It used a strong transmitter, and used two frequencies. One frequency was used for sending and the other for receiving. This full-duplex system had 23 channels and was a great improvement over the *push-to-talk system*. However, in IMTS too, users had to wait for a long time to get a dial tone.

The third step was the development of **Advanced Mobile Phone System (AMPS)**. In England, the same is called **TACS** and in Japan, it is called **MCS-L1**. In this scheme, the area covered is conceptually divided in small regions known as **cells**, thus the name **cellular phones**. Though, actually the cells are circular, they are shown as hexagonal for conceptual clarity in the figure. Each cell has an antenna and a cell office to control that cell. A **Mobile Telephone Switching Office (MTSO)** controls various such cell offices and coordinates the communication between them and the **Telephone Central Office (TCO)** or a telephone exchange. The TCO is a part of the wired land telephone system. The computer at MTSO is responsible for not only connections but also information and billing for calls. A typical cell radius size is 0 to 12 miles. However, it can be changed, depending upon the population of the area.

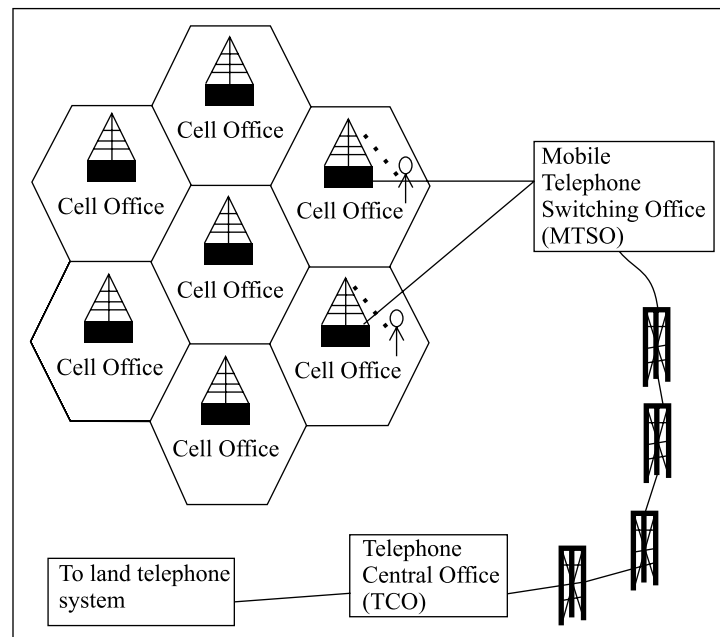


Fig. 6.21 Cellular phone system

Classically, analog transmission is used for cellular telephony. Frequency modulation is used for communication between the mobile phone and the cell office. Normally, two frequency bands are allocated for this purpose. One of them is for communication that is initiated by the mobile phone and the other for that initiated by the land phone. However, each channel requires a full-duplex dialog. For preventing interference, adjacent channels are rarely allocated. Some channels are also required for control purposes. This reduces the number of channels available for each cell. This number is 40 in the US, today. However, the same frequency band can be used for multiple non-adjacent cells as shown in Fig. 6.22. In the US, these bands are controlled by the FCC. The two bands are typically 824–849 MHz and 869–894 MHz.

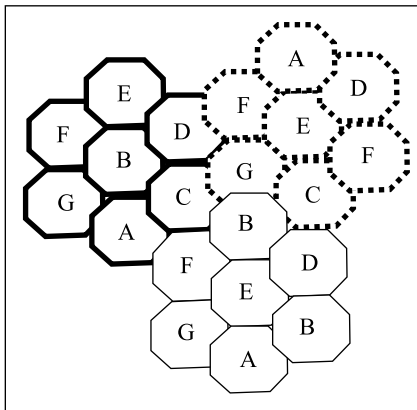


Fig. 6.22 Frequency reuse

When a call is made from the mobile phone by entering a 7, 8 or 10 digit phone number, the mobile phone itself scans the band and seeks a channel for setting-up the call. After this, it sends this number to the closest cell office, which, in turn, sends it to the MTSO. The MTSO, in turn, sends it to the CTO. If the called party is available, the CTO informs the MTSO. At this juncture, the MTSO allocates an empty voice channel to the cell to establish the connection. The mobile phone adjusts its tuning to the new channel and the dialog begins.

When a land phone tries to connect a call to a mobile phone, the telephone central office sends the number to the MTSO. The MTSO performs a look-up to see where the mobile phone is currently placed by sending appropriate query signals to all the cells. This process is called paging. The cell where the mobile phone is currently located reverts to the MTSO. The MTSO then transmits the incoming call signal to that mobile phone, and when the mobile phone is answered, the MTSO assigns a voice channel to the call, thus enabling the conversation.

During the conversation, if the mobile phone crosses the cell, the signal can become weak. The MTSO constantly checks the signal level, and if it finds it low, it immediately seeks a new cell that can look after the communication better. The MTSO then changes the cell-carrying channel so smoothly that the user hardly notices. This process of handling the signal off from the old channel to the new one is called **handoff**. This process is described below.

Step 1: Cell A senses that the user of cell 50 is moving to cell B, and that its signal is becoming weaker. So, it alerts the MTSO of the same. This is shown in Fig. 6.23.

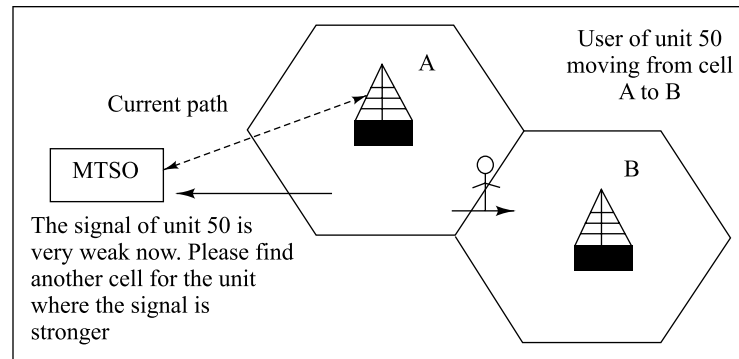


Fig. 6.23 Handoff Part 1—A unit becomes weak in cell A

Step 2: The MTSO wants to check if any of the adjacent units (B, C, etc.) can take up the responsibility of unit 50. Cell C responds by saying that it cannot do so, as unit 50 is weak in cell C, too. However, cell B says that it is ready to take up the responsibility of unit 50 as it is receiving a strong signal from unit 50. This is shown in Fig. 6.24.

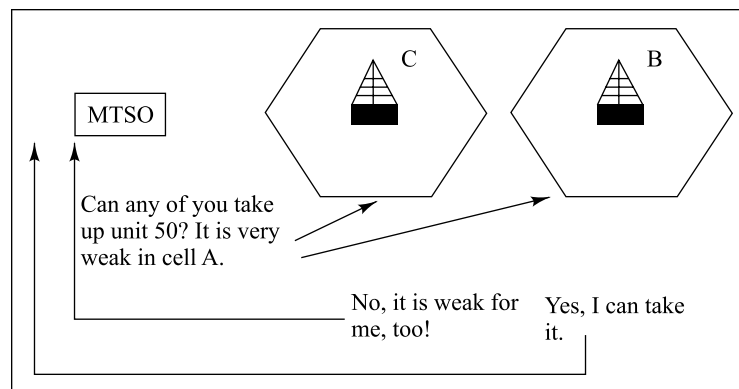


Fig. 6.24 Handoff Part 2—MTSO enquires to see if anybody can take up unit 50

Step 3: The MTSO redirects unit 50 from cell A to cell B. This is shown in Fig. 6.25.

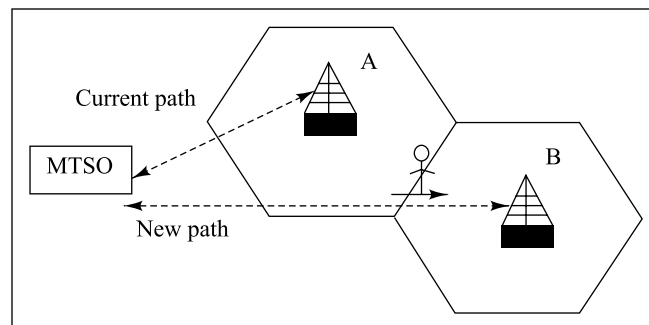



Fig. 6.25 Handoff Part 3—MTSO hands over unit 50 to cell B

Three developments will take place in the near future. One is **digital cellular telephone**. The second is the integration of cellular phones with satellite communication. This will enable people to have a unique but same telephone number throughout the world, unlike today, when we have different numbers for land phones and mobile phones. The third is the integration of mobile telephony with the PC. The scheme is called Mobile Personnel Communication, which will allow people to use a small mobile PC to send and receive multimedia information in all forms, viz., data, voice, image and video.

Figure 6.26 summarizes the various transmission media in terms of their characteristics and other features.

Medium	Speed	Cost	Security	Attenuation
UTP	1 – 100 Mbps	Low	Low	High
STP	1 – 150 Mbps	Medium	Low	High
COAX	1 Mbps – 1 Gbps	Medium	Low	High
Fiber	10 Mbps – 2 Gbps	High	High	Low
Radio	1 – 10 Mbps	Medium	Low	High-Low
Terrestrial Microwave	1 Mbps – 10 Gbps	High	Medium	Variable
Satellite Microwave	1 Mbps – 10 Gbps	High	Medium	Variable
Cellular	1.6 – 1.2 Kbps	High	Low	Low

Fig. 6.26  *Transmission media characteristics*

6.3 SHANNON CAPACITY

Regardless of whether it is a guided or an unguided medium, every transmission medium has a finite data carrying capacity. We are normally very much interested in knowing how much data can a particular transmission medium carry. Claude Shannon solved this mystery in 1944 by introducing a formula that determines the maximum data rate of a channel. This is called **Shannon capacity**.

Shannon capacity essentially depends on one basic factor of a transmission medium, i.e., its signal-to-noise ratio. For understanding how to calculate signal-to-noise ratio we'd have to refer to concepts in Physics, therefore, we shall avoid it. Instead, we shall have a look at the formula to determine the maximum data rate of a transmission medium (called Shannon capacity). The formula reads as follows:

$$C = B \log_2 (1 + S/N)$$

C is the Shannon capacity in bps, B is the bandwidth of the transmission medium, and S/N is the signal-to-noise ratio of the transmission medium.

For instance, let us calculate the theoretical upper limit of transmission rate through a normal telephone channel. Normally, a telephone channel has a bandwidth of 3000 Hz (300 to 3300 Hz). The S/N ratio is usually around 3162. Thus, its capacity C will be as given below:

$$\begin{aligned}
 C &= B \log_2 (1 + S/N) \\
 &= 3000 \log_2 (1 + 3162) \\
 &= 3000 \log_2 (3163) \\
 &= 3000 \times 11.62 \\
 &= 34,860 \text{ bps or } 34.860 \text{ Kbps}
 \end{aligned}$$

This signifies that no matter how hard you try, you cannot transmit more than 34.860 Kbps data through a telephone line. To achieve higher transmission rates, you must try and improve the S/N ratio.

Thus, Shannon capacity formula informs engineers that no amount of intelligent encoding can break the laws of Physics that place a basic limit on the number of bits that can be transmitted per second over a communications system.

SUMMARY

Transmission media can be classified into the following two main categories: Guided media and Unguided media. The difference between the two is the fact that guided media use physical cable for data transmission, whereas in the case of unguided media, air is the transmission medium.

Twisted copper wires, coaxial cables and optical fibers are the three main types of guided media. They differ in their physical characteristics as well as transmission capacities. Optical fibers are the latest and the best of them all. However, it is most expensive option as well. The traditional transmission medium is twisted pair wires. Unguided media can be radio, microwave, satellite or cellular phones. Satellite communications take place using one of the following three techniques: (a) Frequency Division Multiple Access (FDMA), (b) Time Division Multiple Access (TDMA) and (c) Code Division Multiple Access (CDMA). FDMA puts each transmission on a separate frequency, TDMA assigns each transmission a certain portion of time on a designated frequency, and CDMA gives a unique code to each transmission and spreads it over the available set of frequencies.

Security is a major concern in case of unguided media. Of these, microwave and satellite offer the highest data rates. Cellular phones are increasingly becoming popular all over the world, and these offer the most modern way of communication using unguided media.

KEY TERMS AND CONCEPTS

Advanced Mobile Phone System (AMPS)	Mobile telephone
Cell	Mobile Telephone Switching Office (MTSO)
Cellular phone	Optical fiber
Coaxial cable	Push to talk system
Code Division Multiple Access (CDMA)	Satellite communications
Crosstalk	Shannon capacity
Digital cellular system	Shielded Twisted Pair (STP)
Downlink	TACS
Frequency Division Multiple Access (FDMA)	Telephone Central Office (TCO)
Geosynchronous satellites	Time Division Multiple Access (TDMA)
Guided transmission media	Transmission media
Handoff	Twisted pair wire
Improved Mobile Telephone System (IMTS)	Unguided transmission media
MCS-L1	Unshielded Twisted Pair (UTP)
Microwave	Uplink

QUESTIONS

True/False

1. Unguided media can be subdivided into twisted pair wire, coaxial cable, and fiber optic cable.
2. Coaxial cable has an inner central conductor surrounded by an insulating sheath, which in turn, is enclosed in the other conductor.
3. When light passes from one medium to another medium of different density, its speed and direction remain unchanged.
4. Light emitting diode and injection laser diode are same types of light sources.
5. In microwave communication, repeaters are used along with antennas to enhance the signal.
6. Only one satellite is sufficient to cover the earth's surface entirely.
7. FDMA is a digital form of communication.
8. CDMA is the most widely used technique.

Multiple-Choice Questions

1. _____ is the most commonly used transmission medium.
 - (a) Optical fiber
 - (b) Coaxial cable
 - (c) UTP
 - (d) STP
2. The two wires inside a UTP are twisted around each other to reduce _____.
 - (a) noise
 - (b) crosstalk
 - (c) error
 - (d) voltage
3. STP helps eliminate _____.
 - (a) noise
 - (b) crosstalk
 - (c) error
 - (d) voltage
4. _____ is generally used for cable television transmissions.
 - (a) Optical fiber
 - (b) Coaxial cable
 - (c) UTP
 - (d) STP
5. Optical fibers use _____ for data transmission.
 - (a) voltage
 - (b) current
 - (c) light
 - (d) sound
6. Optical fibers use _____ to guide the light through the optical fiber.
 - (a) refraction
 - (b) critical angle
 - (c) reflection
 - (d) waves
7. Out of all the guided media, _____ has the highest data transmission rates.
 - (a) optical fiber
 - (b) coaxial cable
 - (c) UTP
 - (d) STP
8. Geosynchronous satellites move at _____ RPM as that of the earth.
 - (a) same
 - (b) half
 - (c) double
 - (d) thrice
9. _____ transmits each signal on a different frequency.
 - (a) TDMA
 - (b) CDMA
 - (c) TDM
 - (d) FDMA

10. _____ allows access to full bandwidth of the frequency spectrum.
(a) TDMA (b) CDMA
(c) FDM (d) FDMA
11. _____ allows any transmitter to transmit in any frequency, and at any time.
(a) TDMA (b) CDMA
(c) FDM (d) FDMA
12. A _____ controls various cell offices in a cellular system.
(a) MTSO (b) TCO
(c) AMPS (d) satellite
13. The process of handling the signal off from the old channel to the new one is called _____.
(a) transfer (b) change
(c) modification (d) handoff
14. As per Shannon's capacity theorem, every transmission medium has a _____ data rate.
(a) minimum (b) maximum
(c) zero (d) infinite

Detailed Questions

1. Discuss unshielded twisted pair wire.
2. What is shielded twisted pair wire?
3. Describe the term coaxial cable.
4. Describe the structure of optical fibers and the light source for the fiber?
5. What are the advantages and disadvantages of optical fibers?
6. Explain how microwave communication works.
7. Discuss how satellite communication differs from microwave communication.
8. Describe and distinguish between FDMA, TDMA and CDMA.
9. Explain introduction and growth of cellular/mobile telephones.
10. What is handoff? How does it take place?
11. Discuss Shannon's capacity. What implications does it have?

7

Network Topologies, Switching and Routing Algorithms

7.0 INTRODUCTION

Network topology defines how various computers or nodes are connected to one another. There are six basic topologies from which most others can be derived. These are shown in Fig. 7.1.

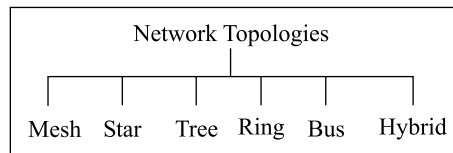


Fig. 7.1 Network topologies

Switching refers to the technique of connecting computers to a central node, called a switch, which can then be used to connect to other nodes. Without switching, we would need to connect every computer in the world to every other computer for anyone to communicate with anyone else. This is clearly not practical, and here switching comes to the rescue.

7.1 MESH TOPOLOGY

In **mesh topology** (also called **complete topology**), each node is connected to every other node by direct links. Obviously, for m nodes, there would be $m(m-1)/2$ physical links. This also means that every node must have $(m-1)$ I/O ports. Figure 7.2 illustrates this.

Mesh topology does not have traffic congestion problems, due to dedicated links. As the links are not shared, a special **Media Access Control (MAC) protocol**—a part of the data link layer of the OSI model (usually needed in LANs due to shared media), is not needed to decide who should communicate to whom and for how long. This topology also has the advantage in terms of data security. It also is robust. If one link is down, the rest of the network can still continue. Fault identification also is easy in this case. The main demerit of this scheme is cable length and consequent cost and complexity. For instance, for 1000 nodes, one will require $1000(1000 - 1) / 2 = 1000 \times 999 / 2 = 49,950$ cables or links. This is clearly impractical for medium to large networks.

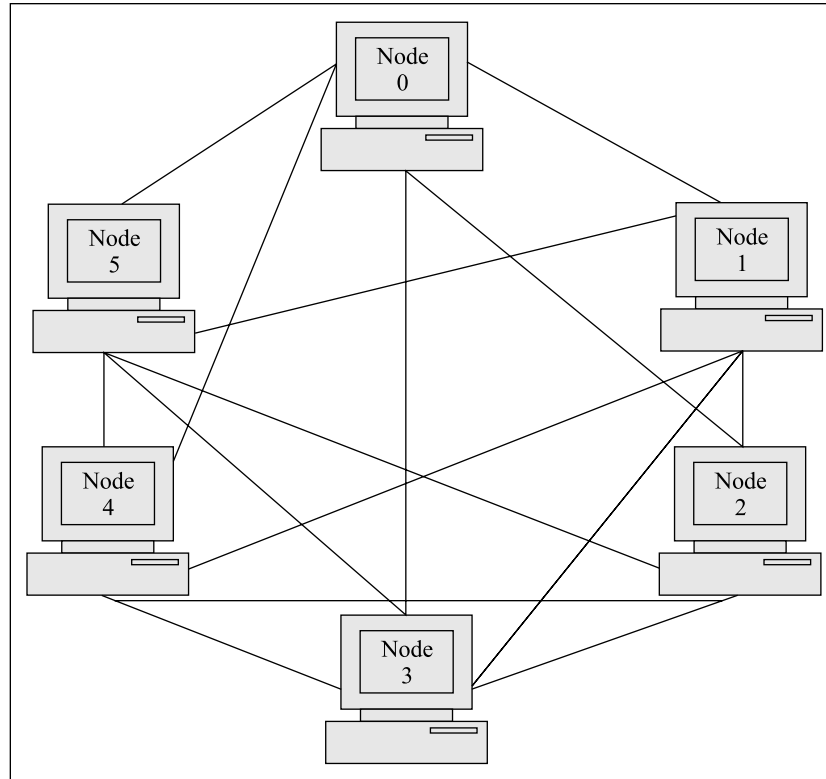


Fig. 7.2 Mesh topology

7.2 STAR TOPOLOGY

In telephone systems, if each telephone were to be physically connected to all others (as in mesh topology); life would be miserable due to the sheer amount of wiring. For this reason we have a telephone exchange to which all phones are connected, and through which connections are established using switching. **Star topology** uses the same principle. There is a central node, often called a **hub**. If a node wants to send some data to another node, it sends it to this hub. The hub, in turn, sends it to the appropriate node. Figure 7.3 depicts this.

Star topology is cheaper than mesh topology. It is also relatively easier to install, maintain and reconfigure. It is also robust. If one link goes bad, all others still continue functioning, except that node and that link. However, if the hub goes down, the entire network becomes defunct. This is a major demerit of this scheme.

7.3 TREE TOPOLOGY

Tree topology can be derived from the star topology. Tree has a hierarchy of various hubs, like branches in a tree; hence the name. Figure 7.4 depicts this. In this case, every node is connected to some hub. However, only a few nodes are connected directly to the central hub.

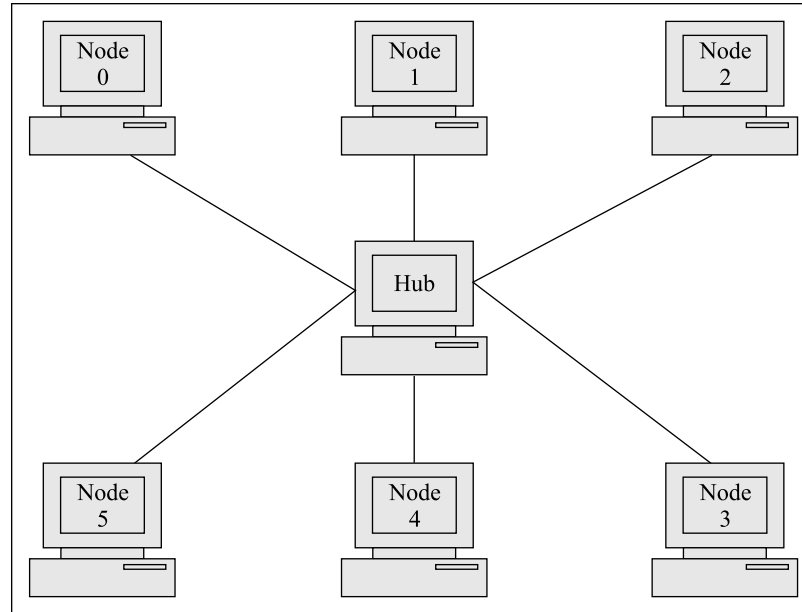


Fig. 7.3 *Star topology*

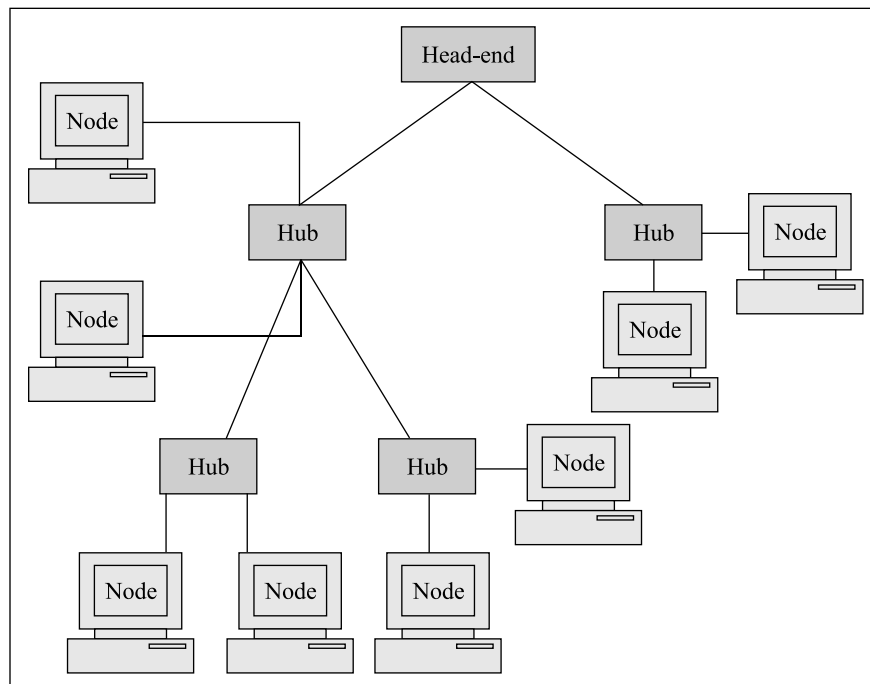


Fig. 7.4 *Tree topology*

The central hub contains a repeater, which looks at the incoming bits and regenerates them afresh as full-blown signals for 0 or 1 as per the case. This allows the digital signals to traverse over longer distances. Therefore, the central hub is also called **active hub**. The tree topology also contains many secondary hubs, which may be active hubs or **passive hubs**. The merits and demerits of tree topology are almost similar to those of star topology.

7.4 RING TOPOLOGY

In **ring topology**, each node is directly connected to only its two adjacent neighbors. If a node wants to send something to a distant node on a ring, it has to go through many intermediate nodes, which act like repeaters, reproducing the incoming bit stream with full signals on the outgoing line. Figure 7.5 depicts this topology.

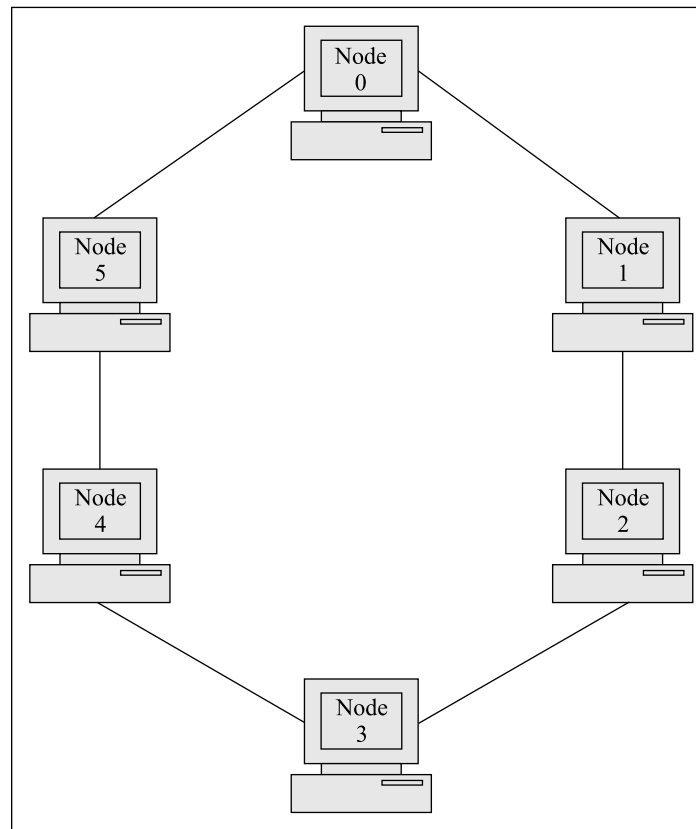


Fig. 7.5 Ring topology

A ring is easy to reconfigure and install. In a ring, normally a signal circulates all the time. A node not receiving any signal for a long time indicates a fault. Therefore, fault isolation is relatively easy in a ring. However, if a node in a simple ring fails, the whole ring cannot function. Therefore, some ring topologies use dual rings. Another demerit is that traffic flows only in one direction. This topology is not used if the number of nodes to be connected is very high.

7.5 BUS TOPOLOGY

Bus topology uses multipoint philosophy. In this case, a long cable called **bus** acts as a backbone for all the nodes as shown in Fig. 7.6.

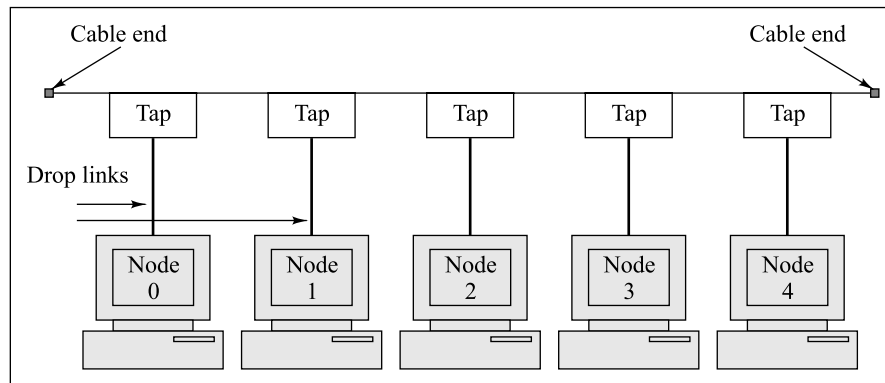


Fig. 7.6 Bus topology

A node wanting to send some data to some other node pushes the data on the bus, which carries it to the other node, where it is received in the same way as passengers get in a bus and get out of it at their destination. Hence this topology has the name *bus*.

A tap is a connector that connects the node with the metallic core of the bus via a drop line. As the signal traverses across the bus, some of the energy is converted into the heat energy, thus weakening the signals. This puts a limit on the number of taps and the distance between them. Therefore, this topology cannot be used for a very large number of computers.

A bus is easy to install and uses less cables than mesh, star or tree topologies. However, in this case, fault isolation becomes very difficult. It is relatively difficult to add new nodes to a bus, thus making it more inflexible. This is because, addition of a node changes the number of taps and the average distance between them, which are generally optimized for a bus length. Another demerit is that, even if a portion of the bus breaks down, the whole bus cannot function.

7.6 HYBRID TOPOLOGY

Hybrid topology is the one, which uses two or more of the topologies mentioned above together. Figure 7.7 depicts this. In this case, bus, star and ring topologies are used to create this hybrid topology. There are multiple ways in which this could be created. In practice, many networks are quite complex but they could be reduced to some form of hybrid topology.

It should be remembered that *topology* refers to the logical arrangement of nodes and not the physical appearance. This is illustrated by Fig. 7.8, which is actually a bus topology, though it appears to be a star topology.

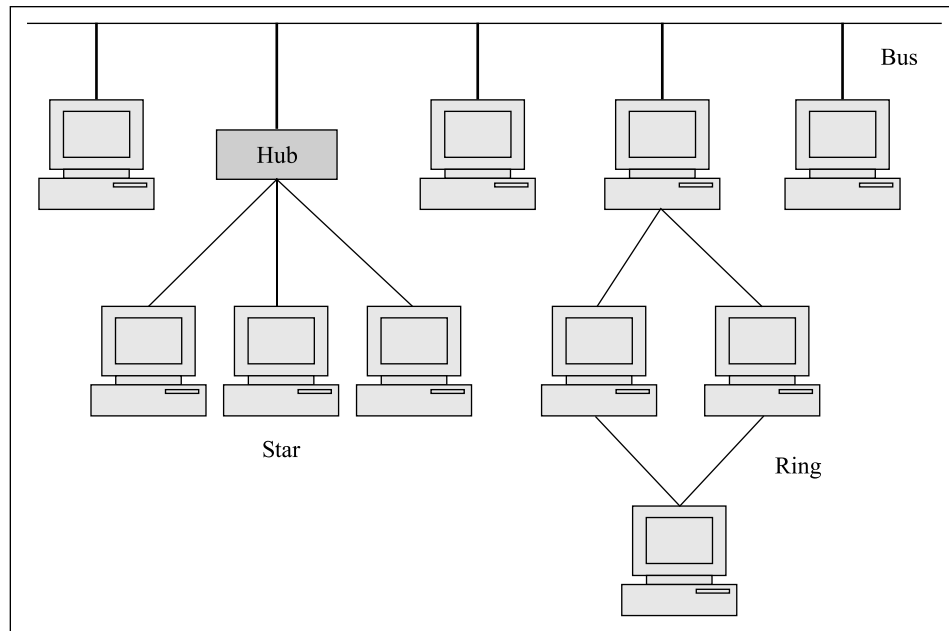


Fig. 7.7 Hybrid topology

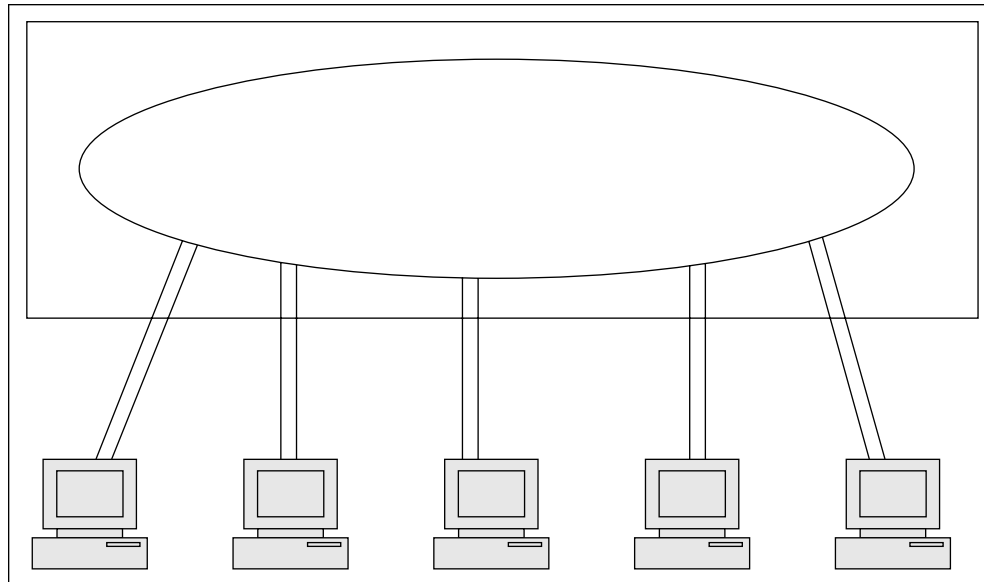


Fig. 7.8 Bus topology that appears to be a physical star topology

7.7 BASICS OF SWITCHING

Many times, we are faced with the problem of interconnecting multiple devices over a long distance, so that they can communicate with each other. At such times, bus and ring topologies cannot be used due to long distances and a large number of nodes. Mesh topology also cannot be used due to the number of lines $(m(m-1)/2)$. Star topology also is vulnerable if the central node becomes defunct.

In such situations, the **switching** mechanism is used. A switched network is made up of a number of interlinked nodes, called **switches**. A switch is a hardware (as well as software) device that allows a connection to be established between two or more devices, which are linked to it (but the devices are not connected to each other). Figure 7.9 illustrates the example of a **switched network**. Computers A through L are connected to various switches, numbered 1 through 7. Each of these switches is connected to multiple links, and thus allows a connection to be established between two computers.

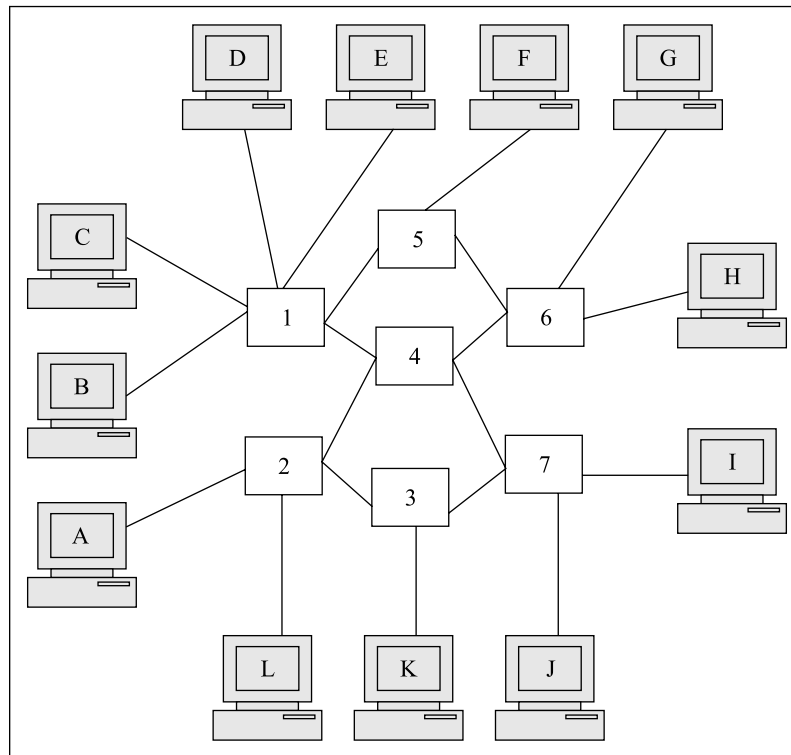


Fig. 7.9 Switched network

As you can see, this concept is very similar to a telephone exchange, which connects the telephones of different users to each other indirectly, rather than having to connect every telephone in the world to every other (which would have meant wires everywhere!). Once a switched network is established, switching can be achieved in multiple ways.

As Fig. 7.10 shows, the most common switching methods are **circuit switching**, **packet switching** and **message switching**. We shall discuss these methods in this chapter. Additionally, new switching methods such as Cell Relay and Frame Relay are also becoming popular. We will study these modern switching methods in another chapter.

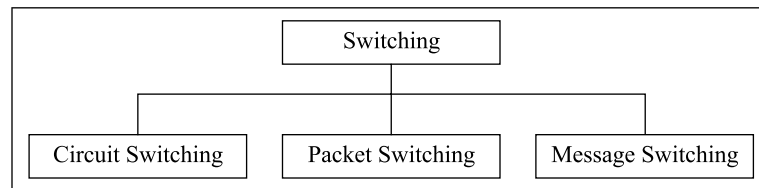


Fig. 7.10 Switching methods

7.7.1 Circuit Switching

In the case of **circuit switching**, a direct physical connection path is established between two computers. This is similar to a telephone call. For instance, when you call up somebody over the phone, a dedicated connection is established between your phone and the phone of the other person; of course, through various exchanges or switches. This dedicated connection remains intact until the call is complete. Even if you or the other person do not speak for a while, the connection is not broken. It remains intact for the entire duration of the call. The same concept is used in circuit switching. When a computer wants to communicate with another computer, a dedicated connection is established between them over the switches. The computers can then communicate using that connection. When the communication is over, either computer can send a request for terminating this connection (similar to how one of the persons can hang up the phone after a discussion is over), and only at that stage would the connection be released.

As long as the two computers communicate over this dedicated connection, no other computer can use this portion of the connection. To understand this, imagine that the bandwidth of a transmission line is sliced into multiple channels for multiple senders and receivers to communicate with each other. One such slice of the total transmission line is exclusively reserved for the two computers using that connection (e.g., using FDM that uses a particular frequency band per connection), even when none of them is sending any data. Using this approach, even the number of switches can be reduced by moving the levers of the switches automatically when a connection is to be established. For example, suppose computer A wants to send some data to computer D as shown in Fig. 7.11. Note how this can be achieved. The dark line shows the dedicated channel between computers A and D. Also note that fewer switches are required.

Thus, a circuit switch is a device with m inputs and n outputs, where m and n need not be equal. It creates a temporary dedicated connection between an input device and an output device, as shown in Fig. 7.12. Of course, the figure is a conceptual depiction. The actual mechanism, in which it achieves this, may be different.

Circuit switching was mainly devised for telephone communications. In telephone calls, this type of temporary dedicated connection is very useful for the duration of the call. However, for computer-to-computer communications, circuit switching is not so efficient. This is because, unlike conversations between humans (which are continuous once started), a computer might send some data to another, and then may not send any more data for quite some time. That is, communication

between computers is less predictable and occurs in bursts. This means that if a dedicated line is established between the two computers, chances are that this line may not be utilized most of the time. Another problem with circuit switching is that once a connection is established, that connection is used throughout the session of the conversation. Even if some other routes get free in the meanwhile, which may be faster than the connection already established, one cannot change over. Also, circuit switching treats all communication requests as equal. There is no concept of message priority, which is required in computer communications to ensure that urgent and high priority messages reach faster than those that are not urgent and have less priority.

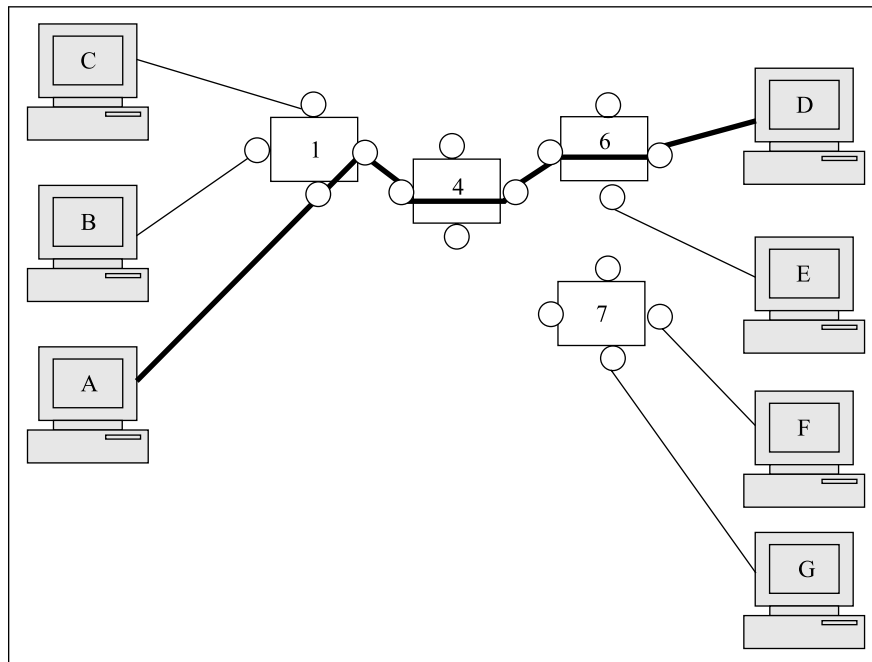


Fig. 7.11 *Example of circuit switch between two computers*

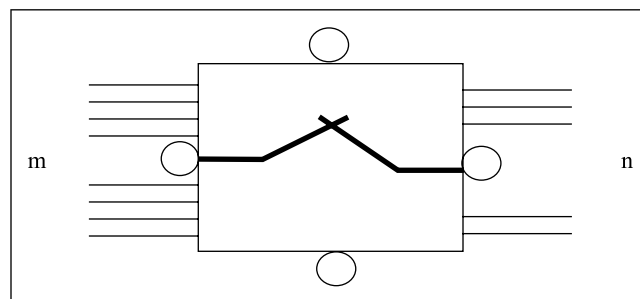


Fig. 7.12 *Circuit switch*

7.7.2 Packet Switching

Considering all the limitations of circuit switching, **packet switching** has emerged as the standard switching technology for computer-to-computer communications, and therefore, is used by most communication protocols such as X.25, TCP/IP, Frame Relay, ATM, etc. Unlike circuit switching, in packet switching, data are transmitted as discrete blocks, called **packets**, which are of potentially variable length. The underlying network mandates the maximum size of data, called **packet size** or **packet length**, which can be transmitted at a given time. Each packet contains data to be transferred, and also the control information such as the sender's address and the destination's address. Packets also help in recovering from erroneous transmission quicker and more easily. This is because, in case of packet switching, only the packets in error need to be retransmitted.

Packet switching can be classified into two types as shown in Fig. 7.13.

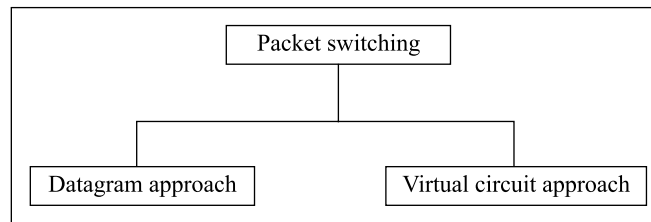


Fig. 7.13 Types of packet switching

Let us discuss these two approaches of packet switching now.

Datagram Approach

In the **datagram approach** of packet switching, each packet is considered a totally independent packet from all others. Even when there are multiple packets sent by the same source to the same destination for the same message, each packet is independent of all other packets from the point of view of the underlying network, and actually can follow different routes or path.

Figure 7.14 illustrates packet switching. Here, computer A is sending four packets to another computer D. These four packets belong to the same original message, but travel via different routes, and also can arrive at the destination D in a different order than how the source A sent them. Therefore, the destination node needs to have the buffer memory to store all the packets and at the end, and resequence them to form the original message. It is obvious that each packet must have a header containing the source and destination address, the packet number, the CRC, etc. The reason that the packets travel via different routes is that the routing decisions are taken for every packet separately, each time at every node, as the packets move from one node to the next. As the network condition and congestion at different nodes/links differs every second, different packets may choose different routes based on the situation at that time. For instance, at switch X, the routing table may suggest that packet 1 should be sent to the switch Y if it has to be sent from A to D. However, just before sending packet 2, the routing table may be updated with the new routing table due to the latest congestion conditions at different switches/nodes. Therefore, packet 2 may travel to switch Z from X to reach its destination D.

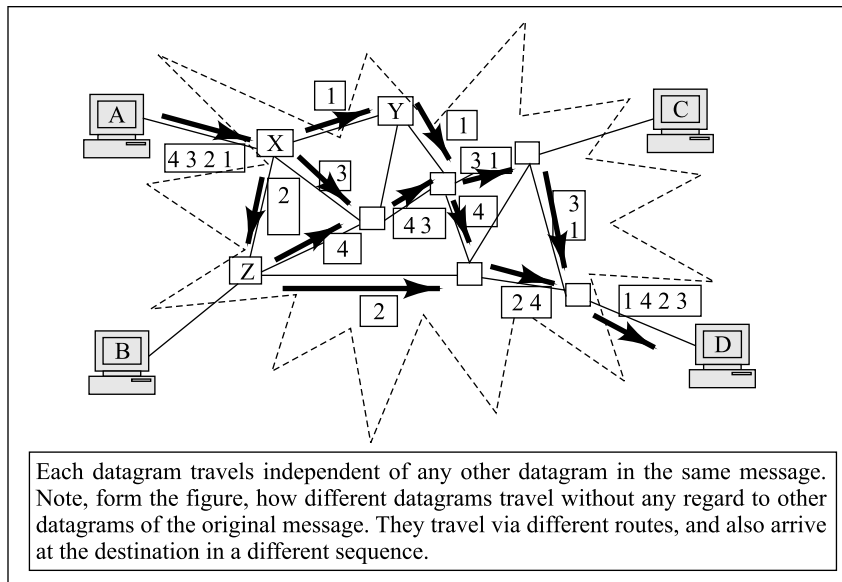


Fig. 7.14 Datagram approach

Virtual Circuit Approach

The **virtual circuit approach** is different from the packet switching approach. In this case, all packets belonging to the same message take the same route from the source to the destination. A single route is chosen between the source and the destination *before* the actual data transfer takes place based on the network/congestion conditions in the beginning. A unique **Virtual Circuit number (VC number)** is allocated to that path between the source and the destination. Each node on that path maintains just two entries, viz., VC number and the address of the next node to which the packet must be forwarded in order to reach the destination. During actual communication (data transfer), this same route is used for complete duration of that communication. For this, the header of each packet traveling from the source carries a virtual circuit number (instead of the full source and destination addresses), which is the unique identification for the virtual circuit already established between the source and the destination, as we have seen. At each node, this virtual circuit number is used to do the table search to find out the next node to which the packet is to be forwarded. This reduces a lot of overheads in terms of the header size, the routing table size at different routers/switches as well as the routing table search times.

Note that all packets travel by the same route and in the same sequence, thus obviating the need for re-sequencing at the destination. This also means that if the protocol follows an acknowledgement scheme, packet $n + 1$ will not be sent to the next switch until the previous switch is sure that the packet n has reached the next switch correctly.

Figure 7.15 illustrates how a virtual circuit approach works. Here, computer A wants to send a message to computer D. As you can see, there is more than one possible route, which can be used for this. However, one route is finalized during the *connection establishment* phase. Once this route is fixed, all the packets in that message are routed via the same route, even though alternate routes are available, and in fact, may even be more efficient at some time phases during the communication. This is the *data transfer* phase. Finally, the virtual connection is released during the *connection release* phase.

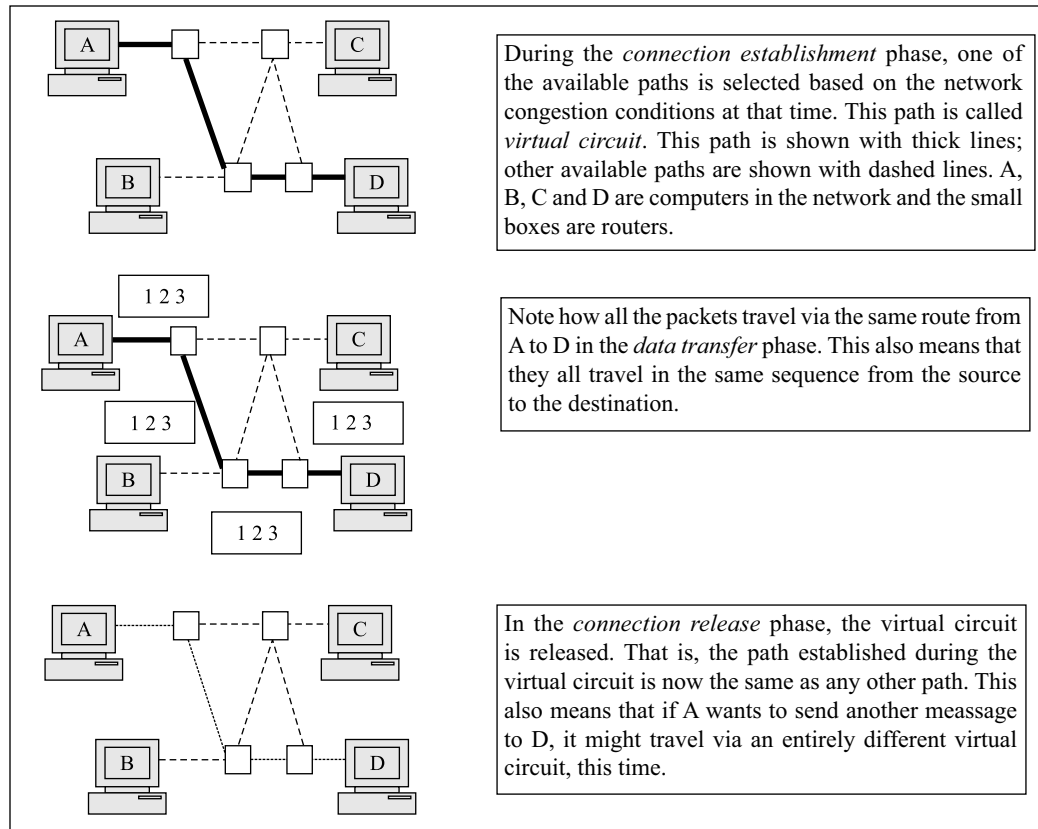


Fig. 7.15 Virtual circuit approach

Differences between Datagram Approach and Virtual Circuit Approach

Figure 7.16 summarizes the differences between the datagram approach and the virtual circuit approach.

Issue	Datagram	Virtual circuit
Connection set-up	Not required	Required
Addressing information	Every packet contains the full source and destination addresses	Every packet contains a short VC number
Routing	Each packet may travel via a different route	All packets follow the same route
Order	No assumptions about the ordering of packets at the destination can be made	Packets arrive at the destination in the same order as they were sent
Bandwidth availability	Dynamic	Fixed
Potential for bandwidth wastage	Yes	No
Store-and-forward mechanism	Yes	No
Billing	Per datagram	Per minute

Fig. 7.16 Differences between datagram approach and virtual circuit approach

7.7.3 Message Switching

Message switching is better known as the **store-and-forward** approach for an entire message (packet switching is store-and-forward at the packet level). In this method, a computer receives a message, stores it on its disk until the appropriate route is free, and then sends it along that route. Since there is no direct link between the source and the destination, the store-and-forward approach is considered as a switching technique. The main difference between packet switching and message switching is that in case of packet switching, the computer stores the packet to be forwarded on its route in its main memory until it can be forwarded. In contrast, message switching specifies that the message to be forwarded should be held by the computer on its disk before it can forward it. The message switching approach is now replaced by message queue technologies such as Microsoft Message Queue (MSMQ) and IBM's MQSeries. Figure 7.17 summarizes the message switching approach. Computer A wants to send a message to computer B. On its way, the message gets stored on the disks of two computers before it is forwarded.

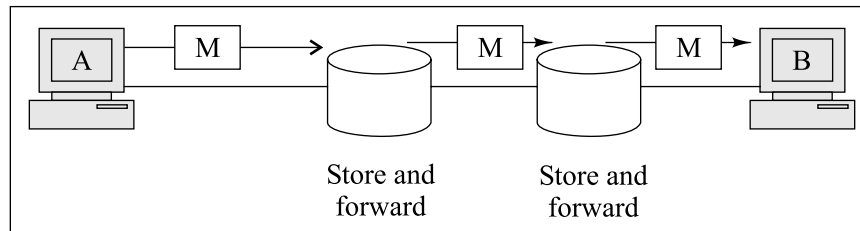


Fig. 7.17 *Message switching*

7.8 ROUTER AND ROUTING

A **router** is a device that connects two or more computer networks together. This allows two or more disparate computer networks to send data to each other. Figure 7.18 shows the idea.

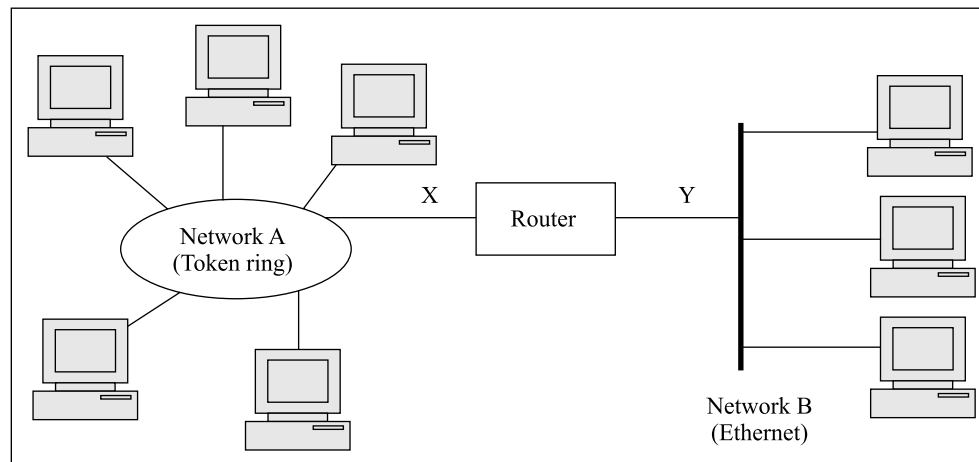


Fig. 7.18 *Router*

The figure shows a router connecting to two networks, viz, A (Token Ring) and B (Ethernet) at points X and Y, respectively. This means that the router must have two interfaces, i.e., one to interact with network A at point X, and the other to interact with network B at point Y. This enables it to send data between the two networks A and B. In fact, a router can connect more than two networks. In this case, it will have as many interfaces.

In the Internet, we have a number of networks wanting to communicate with each other. This also means that there would be so many routers as well. For instance, let us consider a network of computers as shown in Fig. 7.19.

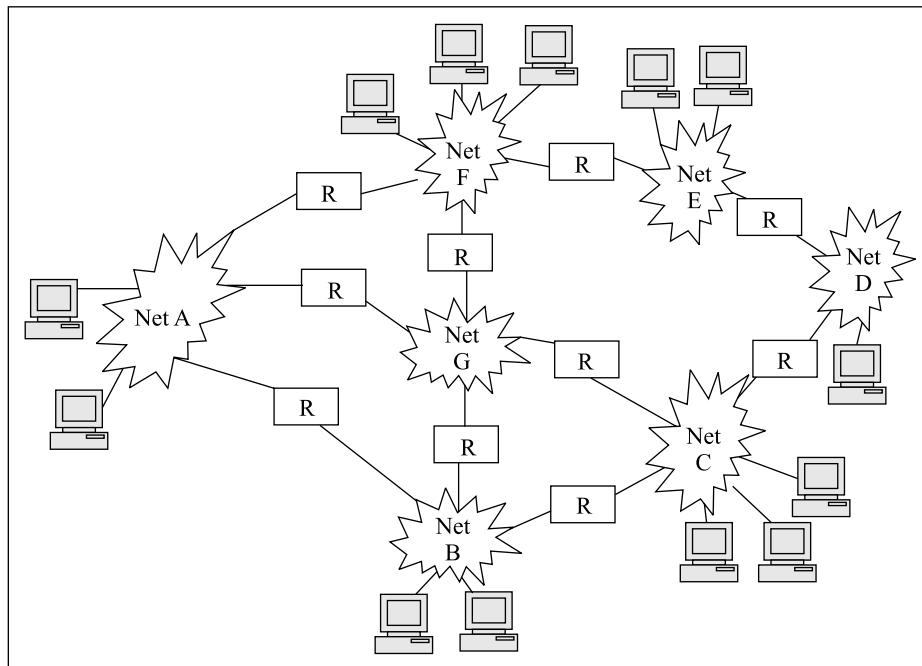


Fig. 7.19 Routers connecting many networks together

Figure 7.19 shows computer networks A through G connected together by a number of routers. Networks A through G could, of course, be of different topologies and sizes. That does not matter. The question now is how do routers help these networks interact with each other? For instance, suppose a computer on network A wants to send some data in the form of datagrams to a computer on network C. Note that these datagrams can travel as A-B-C or A-G-C, or in still other ways. How do we decide which route each datagram should take on its way from the source to the destination?

This decision of routing datagrams appropriately from the source to the destination must be taken in the software, and not the hardware, if we have to allow a seamless integration of networks with each other with the help of routers. For this, we have the concept of **routing algorithms**, which determine the routing decisions, i.e., how to forward a packet on to its next journey. We shall study a few popular routing algorithms in this chapter.

Factors Affecting Routing Algorithms

Three major factors that affect the decisions of routing algorithms are discussed below.

1. **Least-cost routing** – An attempt is always made by the routers to select the shortest possible path when sending packets forward. This is because, when a packet is routed, its *routing cost* is calculated. For this, a value is assigned to each link. The *routing cost* is then the aggregate value of all these links for the route taken by a particular packet. Note that the *routing cost* need not depend on the number of routes encountered, alone. For instance, suppose that a packet travels via a route that has four routers; and another packet travels via a route with three routers. Then it is possible that the route taken by the second packet costs more, even if the number of routes encountered (3) is less than the former (4). This could be because the links in the second route are more stable, and therefore, more costly. Thus, the routing cost depends on many factors, and not just the number of routes taken.
2. **Distributed routing** – In some routing protocols, once a path from the source to the destination is selected, the router sends all the packets for that destination via that same route as in the case of virtual circuit philosophy. Other routing protocols use a technique called **distributed routing**, wherein a router may choose a new route for each packet, even if all the packets belong to the same original message, as a result of the changes in the lengths of the links. That is, if a network A sends three packets to network E via routers, the first packet might travel through network B, the second through G and the third through F on their way to E. This is very much like the datagram philosophy.
3. **Packet lifetime** – Once a router decides about the path to be taken by a packet, it forwards the packet to the next router. In the datagram philosophy, the entire route is not predecided. It can change for each packet, and for each packet at a given router, the subsequent path may be recomputed. Therefore, the next router, may or may not use the same path for routing the packet forward, as was done for the previous packet. The second router is free to decide about a totally new path, if it feels that it is better because of network conditions or link availability, etc., at that time. This allows each router to have a very short and specific logic, which deals mainly with only the next step (called **hop**) of the packet, and not about its entire journey. However, this could also mean, at times, that a packet gets into a situation where it is being routed and re-routed across many routers again and again, because different routers could *think* differently, and could route the packet to create an endless loop.

Let us take an example with some hypothetical router and host names. For instance, suppose a router R1 believes that the shortest route from source A to a destination C is through a network B, and relays a packet accordingly. However, before B receives the packet, it learns that the link between B and C is broken. Therefore, B now decides that the best way to reach C is through A, and updates its **routing table** (we shall study this in later sections) accordingly. Therefore, when B receives the packet, it sends it back to A. Since A has not heard of the link problem between B and C, it sends it back to B, and so on. This means that the packet may not reach the ultimate destination at all. To resolve such situations, additional information is added to the header of a packet, which dictates how long a packet can live, called **packet lifetime**. This field contains the number of hops that are allowed for that packet, before a packet is considered to be lost, and therefore, destroyed at its current location. As a simple rule, as the packet travels through one or more routers, each router subtracts 1 from the value contained in this field. If it becomes zero, it means that the packet should be considered as lost, and the router destroys it accordingly. The job of concluding that the specific packet of the entire message did not reach the destination properly and therefore of requesting for its retransmission is left to the upper layer of the software (transport) running at the destination.

7.9 ROUTING ALGORITHMS

Having discussed the issues involved in routing decisions, let us take a look at the two major categories of routing algorithms used to calculate the shortest path between two routers, viz., **distance vector routing** and **link state routing**.

7.9.1 Distance Vector Routing

In the **distance vector routing** approach, each router shares its knowledge about the entire network conditions periodically with its neighboring routers. There are three important aspects to this.

1. **Knowledge about the whole network** – Each router periodically shares its own knowledge about the entire network with its neighboring routers. To start with, a router may have very little information about the network. At this point, other neighboring routers help it by sending it the information that they have. However, as the router participates more actively in the routing mechanism, it gathers a lot of information about the network, which it shares with other neighboring routers.
2. **Sharing information with neighboring routers** – A router shares the knowledge / information about the network only with those routers with which it has direct links. This is why we have been using the term *neighboring*.
3. **Periodic sharing of information** – Regardless of the changes in the network conditions, each router periodically sends the network information to its neighboring routers after the prespecified interval is elapsed, say, after every 30 seconds.

The distance vector routing scheme works extremely beautifully, using these simple ideas. To understand the actual working of distance vector routing, consider a sample network of networks (called the **Internet**) as shown in Fig. 7.20. There are seven networks numbered 1 to 7, connected to each other by six routers A through F. As we will notice, each router is connected to at least two networks, but it may also be connected to more than two networks, e.g., router A in the figure. It is obvious that the router will have to have as many *networks interface cords*.

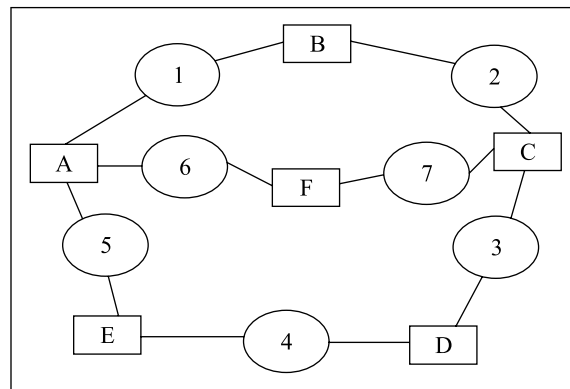


Fig. 7.20 Example – the Internet

Distance vector routing assumes that the cost of every link is 1. This makes its working simple. Thus, the transmission efficiency depends only on the number of links required to reach the destination. Therefore, distance vector routing depends only on hop count, and not on any other factors such as congestion level at every link or even the cost of every link.

Figure 7.21 shows the next step in the distance vector routing algorithm. Each router sends the information about the Internet only to its immediate neighbors. A question then remains, how do routers then know about the parts of the internet to which they are not directly connected? We shall answer this question soon.

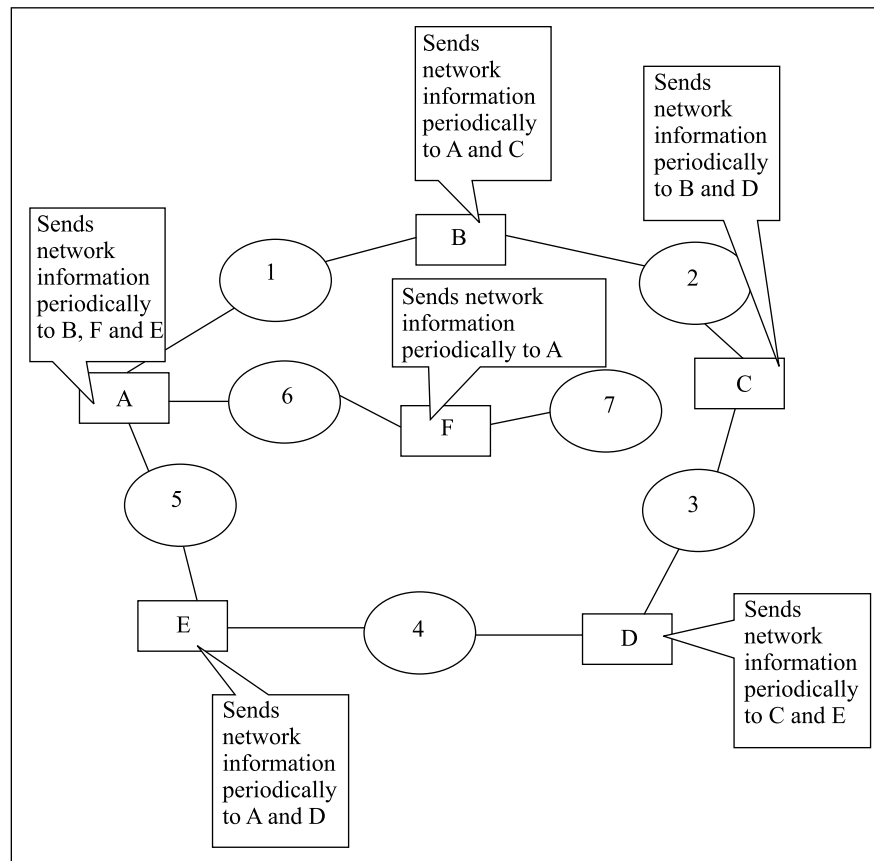


Fig. 7.21 Distance vector routing philosophy

As the figure shows, the routers periodically send the information that they have about the network to their respective neighbors. Thus, router A sends the information to neighboring routers B, F and E, for example. The neighbors update their knowledge with this information, add their own knowledge to it, and then send the complete information to their own neighbors (i.e., the neighbors of neighbors of the original router). When you consider that this creates a chain of information where each router sends its basic information, as well as what it receives from its neighbors to all other neighbors, you would realize that ultimately every router would obtain complete information about the whole network.

For storing information about the network, and keeping it updated, each router uses a **routing table**. The routing table has at least three columns, namely, the network id, the cost, and the id of the next router, which are described below:

1. **Network id** – This is the final destination of the packet.

2. **Cost** – This is the number of hops that a packet must take to reach the final destination.
3. **Next router** – This is the router to which the packet must be delivered, on its way to a particular final destination. The *network id* of the destination is mentioned in the network id field of the table.

For example, this table tells a router that to reach a network X via router Y, it costs Z. This is shown in Fig. 7.22, which shows the format for an empty distance vector routing table.

Network id	Cost	Next router
...
...
...
...
...

Fig. 7.22 Distance vector routing table format

When a router boots up, it has very little information. Since every router is connected to more than one networks, it knows the ids of those networks. Using just this piece of information, a router constructs the initial distance vector routing table, as shown in Fig. 7.23.

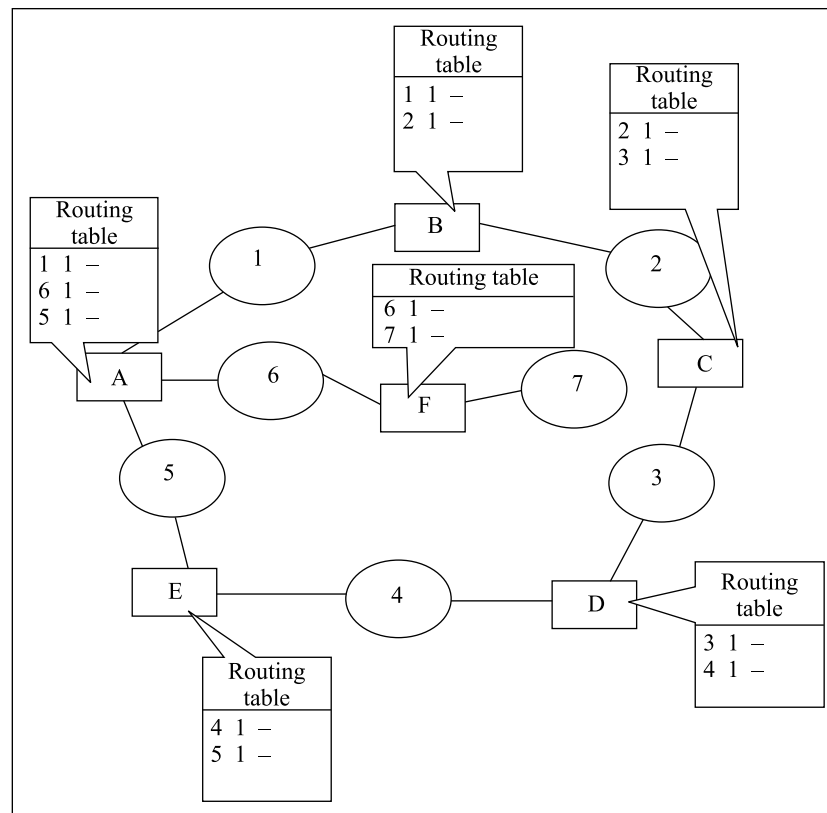


Fig. 7.23 Initial entries in the routing tables of various routers

Figure 7.23 shows the initial entries in the routing tables of the various routers in our sample network. The three columns correspond to the three columns of the table shown in Fig. 7.22. Thus, an examination of the routing table of router A tells us that it can send any packet to networks 1, 6 or 5 (entries in the first column), and that the cost for each such packet forwarding process is 1 (entries in the second column). Note that the third column, i.e., *next router* is empty. This is because at this stage the routers only have information about the networks to which they connect directly. They do not know anything about the *next hop*, i.e., about any other routers or networks in the Internet. They discover this information in stages. It first starts by identifying their neighboring routers.

Having constructed the initial routing table and having identified its neighboring routers, each router then sends its routing table to all its neighboring routers. That is, router A sends its routing table to neighboring routers B, F and E. Similarly, router B sends its routing table to its neighboring routers A and C, and so on.

Having received the routing table of router B, router A updates its own routing table to incorporate the details of the routing table of B. This happens as follows.

1. Router A takes the routing table of B, and adds 1 to the *next hop* field. This is because whatever is accessible to B directly is not accessible to A directly. For instance, network 2 is not directly accessible to A. But now A can access network 2 with the help of B, with one extra hop. That is, to reach network 2, router A has to first forward the packet to router B (first hop) and router B can then forward it to network 2 (second hop). Also, router A adds B as the *next hop* field in this table, because it can reach these new destinations (such as network 2) with the help of B. This is shown in Fig. 7.24. Notice that the table shows that in order to reach network 1, it will require two hops if it has to send it through the router B. However, this is not required, as network 1 is directly connected to router A. We will deal with this situation shortly.

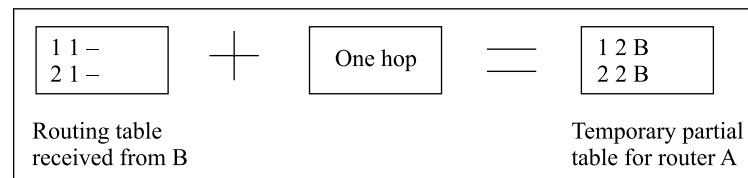


Fig. 7.24 Router A receives the routing table of B and creates a temporary table from it

2. Having created the temporary routing table as shown in Fig. 7.24, the router A now adds the contents of this temporary routing table to its own routing table and sorts it on *destination network id* to get a combined sorted table as shown in Fig. 7.25. The entries from the temporary table are marked for ease of identification.

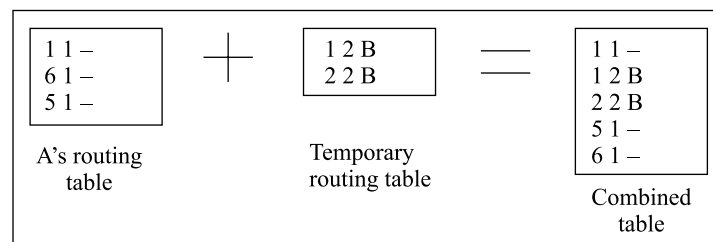


Fig. 7.25 Router A combines its routing table with the temporary table

- Now, router A removes the duplicate entries from the combined table to get the most optimized results as shown in Fig. 7.26. For this, it selects the shortest route when there are multiple ways of reaching a final destination.

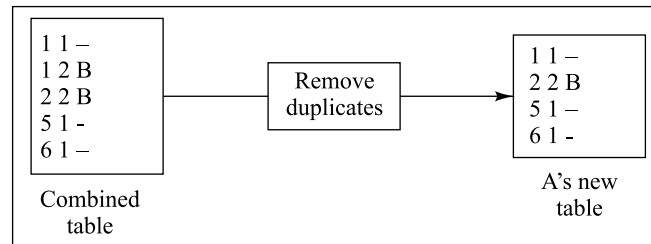


Fig. 7.26 A's new table

We have shown what happens when router B sends its routing table to router A. Note that router A has other neighbors, namely routers F and E. They would also send their own routing tables to router A. Router A would then send its updated router table to routers B, F and E. This would result into routers B, E and F updating their own tables, and so on. This process goes on, and on, and on! Finally, the routing tables of all the routers would get updated with the information for reaching every network on the Internet, as shown in Fig. 7.27 (of course, we have added the headings for readability, they would not be present in the actual tables).

We notice that to reach network 3 from router A, there are two paths. A-5-E-4-D-3 and A-1-B-2-C-3. Both require equal number of hops. We have chosen the first one through the router E, though the second one would have equally worked. This is because, the cost of each hop is assumed to be the same.

Using the final routing tables as shown in Fig. 7.27, routers can now forward any packet heading to the destination network. Of course, they would keep exchanging their routing tables periodically to take care of any networking problems by updating each other of these developments. Once the packet arrives at the network, then routing it to the final destination node on the same network is not very difficult. This is done using the physical addresses and the frame formats applicable to the destination networks as we have seen before.

7.9.2 Link State Routing

Unlike distance vector routing, in case of the **link state routing** algorithm, information is exchanged by a router about its neighborhood with *all other* routers in the Internet. Thus, the following points need to be remembered in the case of link state routing.

- Knowledge about the neighborhood** – Each router periodically shares its own knowledge about its neighborhood only, but it does it with all other routers in the Internet.
- Sharing information with all routers** – For sharing information, a process called **flooding** is used. What this means is simple. A router sends all the information about its neighbors, to all its neighbors. The neighbors forward this information to their own neighbors, and so on. In this process, a router will get back some of the information that it sent to its neighbors in the first place, but it has the intelligence to ignore it.

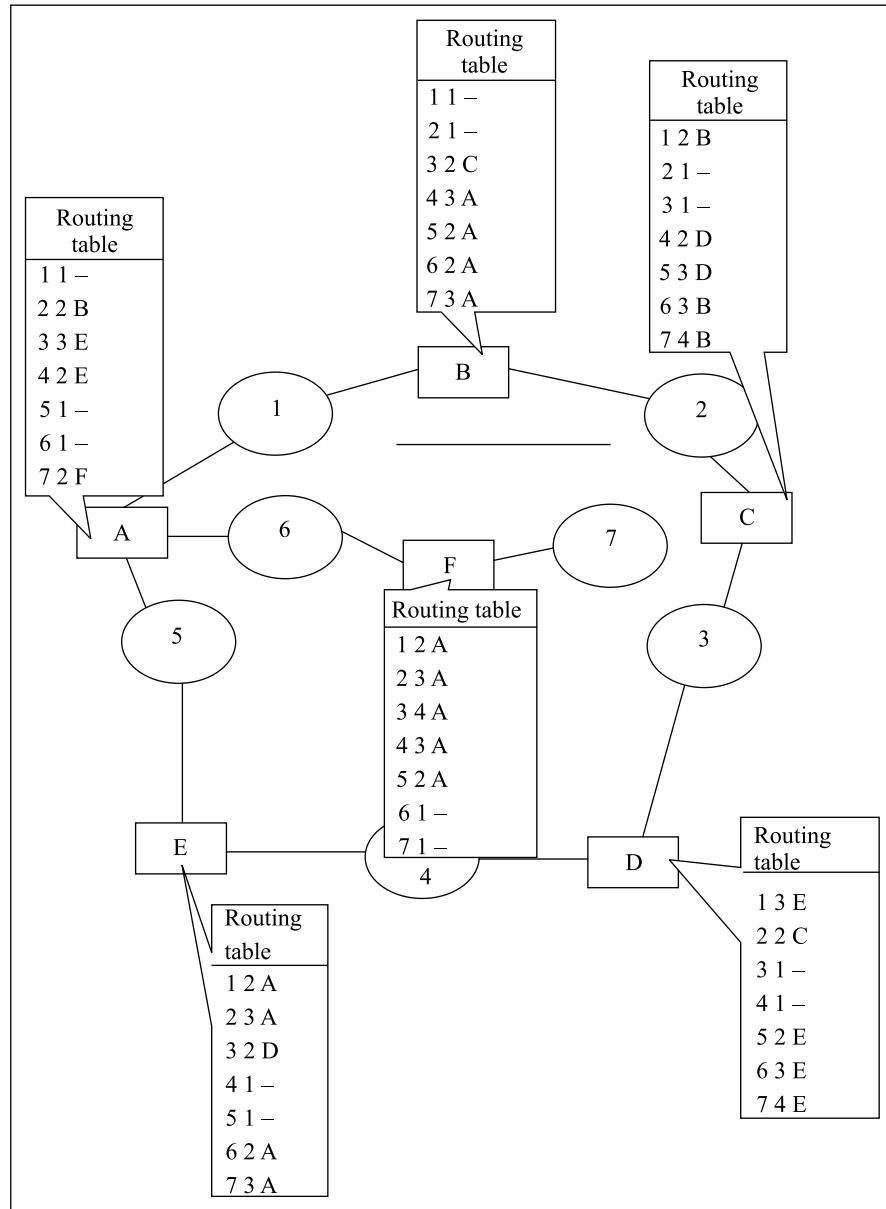


Fig. 7.27 Final routing tables for all routers

3. **Periodic sharing of information** – Regardless of the changes in the network conditions, each router periodically sends the network information to its neighboring routers after the prespecified interval is elapsed. Unlike distance vector routing, however, the interval here is much longer (say after every 30 minutes, as against 30 seconds of the distance vector routing).

We know that the cost of the packet routing is determined based on the hop count in the case of distance vector routing. In the case of link state routing, the cost is not directly based on the hop count. Instead, it is a weighted figure based on a number of factors, such as security levels, traffic and the state of the link. This also means that the cost of routing a packet between router A and network 1 could be different from that of routing a packet between router A and network 6, although both are just one hop away in Fig. 7.21. Note that the term *cost* has nothing to do with the transmission fees paid by the sender or the receiver, it is just a weighted average number used to determine which route is the fastest/shortest based on a number of factors discussed earlier.

We shall understand how link state routing works with the help of the same Internet example as shown in Fig. 7.21, which we had used for illustrating the concepts of distant vector routing. Initially, each router sends its knowledge only about its neighbors to every other router in the Internet. This is shown in Fig. 7.28.

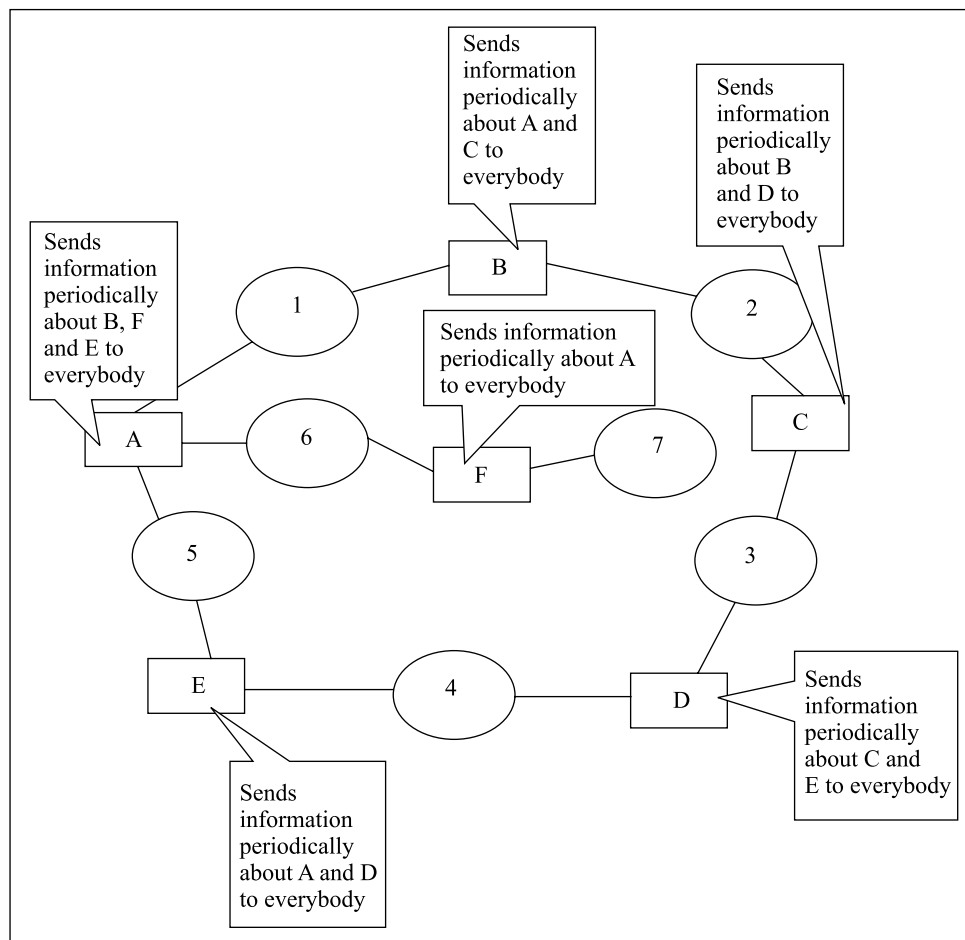


Fig. 7.28 Link state routing philosophy

Periodically, each router sends a very small **greeting packet** to each of its neighbors and expects a response back from the neighbor. If the neighbor reverts, the original router considers that the neighbor is *up and running*, and accordingly determines the cost based on the factors discussed earlier. Otherwise, the neighbor is considered to be in some error. Using this information, the original router then sends information about all its neighbors to the entire Internet in a process called flooding, as discussed earlier. For this, it sends a special packet called **Link State Packet (LSP)** to all other routers via its neighbors.

The general format of the LSP is shown in Fig. 7.29.

Advertiser	Network	Cost	Neighbor
...
...
...
...
...

Fig. 7.29 Link State Packet (LSP) format

The columns in the LSP are described below.

- **Advertiser** – It is the id of the router who is sending the LSP packet.
- **Network** – It is the id of the destination network to which this packet should go.
- **Cost** – It is the cost calculated based on different factors as discussed before.
- **Neighbor** – It is the id of the neighboring router about which this information is being sent in the LSP.

For example, a sample portion of the LSP (shown only for router A about its neighbors) could take the form as shown in Fig. 7.30.

Advertiser	Network	Cost	Neighbor
A	1	1	B
A	6	3	F
A	5	2	E

Fig. 7.30 LSP for router A

For example, the first row says that between router A (the first column) and router B (the fourth column), there is network 1 (the second column), and that the cost of going from router A to router B is 1 (the fourth column).

Every router receives every LSP packet, and uses it to create a local database called **link state database**. Thus, a link state database is a collection of all LSPs. Every router stores such a database on its disk, and uses it for routing packets. A sample link state database for our example Internet is shown in Fig. 7.31.

Advertiser	Network	Cost	Neighbor
A	1	1	B
A	6	3	F
A	5	2	E
B	1	4	A
B	2	2	C
C	2	5	B
C	3	2	D
D	3	5	C
D	4	3	E
E	5	3	A
E	4	2	D
F	6	2	A
F	7	3	—

Fig. 7.31  Link state database

Having constructed the link state database, each router executes an algorithm called **Dijkstra algorithm** to create its routing table. This algorithm considers the Internet as a graph, and finds the distance along a shortest path from a single node of the graph to all other nodes in the graph. Using this information, a routing table is created to compute the **shortest path**. This algorithm must be run for each routing table once. However, a detailed discussion of this is beyond the scope of this text.

7.9.3 Approaches to Routing

At a broad level, the routing tables can be constructed using one of the two approaches: **static routing** and **dynamic routing**.

1. **Static routing** – In this approach, when a router boots for the first time, it executes the routing algorithm and takes various decisions about forwarding packets via various routes to different destinations (i.e., builds a routing table). Once decided, the route cannot change for a particular destination in this scheme. This is a very simple approach. However, when there are network problems (e.g., a link is down or a router is congested, etc.), this approach is not very useful, as the routers cannot adapt to different situations of congestion, etc., and cannot forward the packets via the alternative routes, by updating the routing tables.
2. **Dynamic routing** – In this approach, the router executes the routing algorithm when it boots to take the initial routing decisions, and constructs the initial routing table exactly like the approach taken by static routing. However, this approach is adaptive. When a router senses that there are some changes in the network conditions (e.g., some node is down or some router is congested etc.), it re-examines the network conditions to see which routes are currently affected, and updates the routing tables accordingly. This means that the network changes are handled automatically. This is clearly a better, but more complex approach than static routing.

Note that these static and dynamic concepts are different from virtual circuit and datagram approaches, respectively. Static and dynamic routing are generic decisions, whereas virtual circuit and datagram approaches are related to only one transmission. Also, the former apply to routers, whereas the latter deal with an end-to-end connection (i.e., transmitting node-routers-destination node).

SUMMARY

Network topology defines how various nodes (also called hosts, just some terms for computers) are connected to one another. There are various ways in which the nodes can be interconnected. Accordingly, the topologies are called mesh, star, tree, ring, bus, and hybrid.

In mesh topology (also called complete topology), each node is connected to every other node by direct links. This is the simplest but most impractical topology as the number of connections required is huge. In star topology, there is a central switch called hub, to which all the nodes connect. Therefore, all transmissions pass through the hub. The trouble with this approach is that if the hub goes down, the entire network goes down. The tree topology is an extension of the star topology. Here, rather than a single hub, there are many hubs that connect to each other, and then to the main central hub. Nodes attach to the hubs. Bus topology uses multipoint philosophy. In this case, a long cable called bus forms the backbone to all the nodes. Hybrid topology uses two or more of the topologies. It is a combinational approach.

When multiple hosts are to be connected to each other to form a network, there can be two ways of achieving this, viz., connect every computer to every other computer, or connect some computers to a link and then connect those links. The second approach is clearly better, and paves way for switching, as happens in the case of telephone networks.

A switched network is made up of a number of interlinked nodes, called switches. A switch is a hardware (as well as software) device that allows a connection to be established between two or more devices, which are linked to it. Switching can be classified into circuit switching, packet switching, and message switching.

In case of circuit switching, a direct physical connection path is established between two computers. This is similar to a telephone call. Before the two computers can communicate, a dedicated circuit must be established between them. Since data transmission using computers is in bursts and unpredictable; circuit switching is not a suitable choice for computer communications. As long as the two computers communicate over this dedicated connection, no other computer can use this connection (i.e., at least the bandwidth that is allocated to this channel).

Unlike in a circuit switching, in packet switching, data are transmitted as discrete blocks called packets, which are of potentially variable length.

In the datagram approach of packet switching, each packet is considered as a totally independent packet from all others. Even when there are multiple packets sent by the same source to the same destination for the same message, each packet is independent of all other packets from the point of view of the underlying network, and therefore, can take different routes. This necessitates that the header of each packet contains the source and destination addresses, and also that the higher (transport) layer at the destination has to re-sequence the packets to form the original message.

The virtual circuit approach differs from the datagram approach in the way that all the packets belonging to the same message take the same route from the source to the destination. A single route is chosen between the source and the destination before the actual data transfer takes place. Therefore, all the packets reach the destination in sequence only. The packet header, therefore, needs to contain only the virtual circuit number.

In message switching, a computer receives a message, stores it on its disk until the appropriate route is free, and then sends it along that route.

A router is a device that connects two or more computer networks together. The number of routers increases with the increase in the number of networks and therefore, if we have to allow a seamless integration of networks with each other with the help of routers, then the decision of routing datagrams appropriately from the source to the destination must be taken in the software, and not the hardware. There are different types of routing algorithms, to serve this purpose. Routing algorithms help in determining the routing decisions, i.e., how to forward a packet on to its next journey, so that it reaches the destination quickly.

There are different factors like least-cost routing, distributed routing, and packet lifetime, which affect these routing algorithms. Least-cost routing is calculated based on the number of routers and also the cost and stability of the links the routers use. In the case of distributed routing, a router may choose a new route for each packet, even if all the packets belong to the same original message. This can give rise to situations, in which the packet may travel to and forth in a loop. To resolve such cases, some additional information called packet lifetime is added to the header of a packet, which dictates how long a packet can live. This field contains the number of hops that are allowed for that packet, before a packet is considered to be lost, and therefore, destroyed at its current location. Therefore, if a packet travels through a specific number of hops, it automatically gets destroyed. This avoids endless loops.

There are different routing algorithms like distance vector routing and link state routing. These two are different methods used to calculate the shortest, fastest and therefore least expensive path between two routers. There are two approaches with which the routing tables can be constructed. In static routing, once the path a packet can take between a source and destination is decided, it cannot be changed. This is not very useful in situations where there could be network changes, as this approach does not cater to this aspect. In case of dynamic routing, when a router senses that there are some network problems, it re-examines the network conditions to see which routes are currently affected, and updates the routing tables accordingly. Thus, the network changes are handled automatically.

KEY TERMS AND CONCEPTS

Active hub
Bus topology
Circuit switching
Complete topology
Datagram
Dijkstra algorithm
Distance vector routing

Distributed routing
Dynamic routing
Flooding
Greeting packet
Hub
Hybrid topology
Internet

Least-cost routing	Ring topology
Link state database	Router
Link State Packet (LSP)	Routing
Link state routing	Routing table
Media Access Control (MAC) protocol	Shortest path
Mesh topology	Star topology
Message switching	Static routing
Network topology	Store and forward
Packet	Switched network
Packet lifetime	Switching
Packet size	Tree topology
Packet switching	Virtual circuit
Passive hub	

QUESTIONS

True/False

1. The term network topology defines how various nodes are connected to a network.
2. Mesh topology is also called complete topology.
3. In case of star topology, the data always comes to the central node, which in turn, sends it to the appropriate node.
4. In ring topology, each node is directly connected to many other adjacent neighbors.
5. A switched network is made up of a number of interlinked nodes, called links.
6. Cell Relay and Frame Relay are switching methods.
7. In case of circuit switching, there is no direct physical connection path between two computers.
8. When a computer wants to communicate with another computer, a dedicated connection is established between them over the switches.
9. Packet switching is not very useful for computer-to-computer communications.
10. In packet switching, there is a concept of message priority, which means messages can be sent with different priorities.
11. A packet does not contain any other information than the data.
12. In case of message switching, the computer stores the packet to be forwarded on its route in its main memory until it can be forwarded.
13. Routing algorithms determine the route for data transfer between computers.
14. Routing cost is calculated based on the number of routes encountered during data transmission.
15. It is very well possible that a packet gets into a situation where it is being routed and re-routed across many paths, because different routers could *think* differently, and could route the packet to create an endless loop for that packet.
16. Each router periodically shares its own knowledge about the entire network with its neighboring routers.
17. Each router uses a routing table to keep the information about the networks and update it regularly.

18. In case of dynamic routing, once decided, the route cannot change for a particular destination.

Multiple-Choice Questions

1. In _____, each node is connected to every other node by direct links.
 - (a) ring topology
 - (b) tree topology
 - (c) mesh topology
 - (d) bus topology
2. The concept similar to a telephone network is used in _____.
 - (a) ring topology
 - (b) tree topology
 - (c) mesh topology
 - (d) star topology
3. In star topology, the central hub is called _____.
 - (a) active hub
 - (b) passive hub
 - (c) inactive hub
 - (d) live hub
4. In _____, if a node fails, the whole network cannot function.
 - (a) ring topology
 - (b) tree topology
 - (c) mesh topology
 - (d) star topology
5. _____ uses multipoint philosophy.
 - (a) ring topology
 - (b) bus topology
 - (c) mesh topology
 - (d) hybrid topology
6. In case of _____, a direct physical connection path is established between two computers.
 - (a) circuit switching
 - (b) packet switching
 - (c) message switching
 - (d) datagram approach
7. _____ is more suitable for human communications.
 - (a) circuit switching
 - (b) packet switching
 - (c) message switching
 - (d) datagram approach
8. _____ is more suitable for computer communications.
 - (a) circuit switching
 - (b) packet switching
 - (c) message switching
 - (d) None of the above
9. In the _____ approach, each packet is considered as a totally independent packet from all others.
 - (a) virtual circuit
 - (b) datagram
 - (c) circuit switching
 - (d) message switching
10. Connection set-up is _____ in case of virtual circuit approach.
 - (a) not required
 - (b) impossible
 - (c) optional
 - (d) mandatory
11. Message switching is also called _____ technique.
 - (a) get and hold
 - (b) store and release
 - (c) store and forward
 - (d) store and never release
12. A _____ can connect two different networks for enabling communication between them.
 - (a) connector
 - (b) wire
 - (c) joiner
 - (d) router

13. The _____ indicates how long a packet can live.
 (a) routing table (b) packet lifetime
 (c) packet data (d) router
14. Distance vector routing assumes that the cost of every link is _____.
 (a) 4 (b) 3
 (c) 2 (d) 1
15. The field that indicates where the packet should be forwarded next is _____.
 (a) network id (b) time to live
 (c) current computer's physical address (d) next router or next hop
16. Link state routing uses the technique of _____.
 (a) flooding (b) next hop
 (c) random numbers (d) timely delivery
17. In _____, once decided, the route cannot change for a particular destination.
 (a) dynamic routing (b) static routing
 (c) message passing (d) fixed routing

Detailed Questions

1. Discuss the differences between mesh topology and star topology.
2. Depict how tree topology looks and explain how it works.
3. Discuss merits and demerits of ring and bus topologies.
4. What is hybrid topology?
5. Discuss circuit switching in detail.
6. Discuss how packet switching is better over circuit switching for computer-to-computer communications.
7. Explain the two different approaches of packet switching.
8. What is message switching? How does it work?
9. What does routing mean and how does it take place?
10. Describe the different factors affecting routing algorithms.
11. What is distance vector routing?
12. How does distance vector routing work?
13. Describe the routing table structure.
14. How does routing table update take place?
15. Explain the link state routing algorithm.
16. How does distance link state routing work?
17. Discuss different approaches for routing.

8

Networking Protocols and OSI Model

8.0 INTRODUCTION

Protocol is nothing but a convention. We encounter this term quite often in newspapers describing, for example, a meeting between the leaders of two nations. To signify that “Everything is ok and the train can start” by a green flag is also a protocol. When we write a letter, we follow a certain protocol. The place where we write the address, the place where we adhere the stamp, the place where we write the name of the recipient, the way we begin with the pleasantries, the place and the way we write “Yours lovingly” or “Yours sincerely”, etc., all define a protocol.

Protocols can and normally have layers hidden in them, if we look into them a little carefully. A good example of this is human conversation, in general and over the telephone. In particular, Fig. 8.1 depicts these layers. We will take this example and describe the exact steps to learn about these layers. An interesting point is that we do this without knowing that we use protocols. While studying this, we will encounter a number of terms, which are also used in various computer networks.

We will assume that two persons X and Y want to have a conversation over the telephone about the world war and we will also assume that each one is taking down what the other one has to say. Thus, we will call this *World War* as an *idea*. Normally, the conversation takes place in terms of several messages from either end, hopefully one after the other. A message is a block of statements or sentences. A message could also consist of only one word such as *OK* or *yes*, denoting a **positive acknowledgement (ACK)** of what has been heard or received. A message could also mean a **negative acknowledgement (NAK)** or request for repeating such as *Come again* or, *Pardon me* or, *Can you repeat please*, etc. Remember that this can happen both ways. For instance, a typical conversation could be as follows.

- X: In World War II, the allied countries should have.... However, they did not do so because of the climate conditions. In addition, they did not have enough ammunition.
- Y: Yeah, I agree.
- X: Also, if you consider the factor of the atomic energy....
- Y: No, but, I think, there is another angle to it. If you consider the boundary between the two countries, it will be obvious. There is also a great book on the subject.
- X: Come again.
- Y: No, but I think there is another angle to it.

X: Yeah, but that is not the only factor...

Y: Could you repeat please?

X: ...

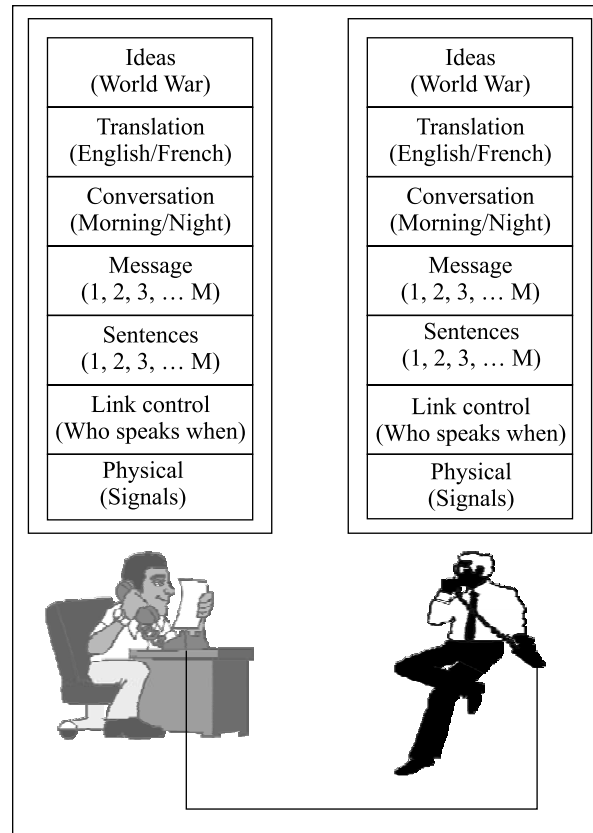


Fig. 8.1 Layers in human communication

Therefore at the level of *ideas*, both X and Y feel that they are discussing an idea such as *World War*. However, in reality the conversation consists of a number of messages from both sides as discussed before. Therefore, at a lower level, the view would be that a number of messages are sent at both ends. The protocol at this level decides what denotes an acknowledgement, what denotes a negative acknowledgement, etc., for the entire message.

A message could be too long. In this case, it may not be wise for X to speak for half an hour, only to receive a request for repeating the message in the end from Y. It is, therefore, prudent to send/receive positive or negative acknowledgements after each sentence in a message by *Yeah, Ok* or *Come again*, etc. A *sentence* is like a *packet* in the computer parlance. In this case also, one could decide a protocol to necessarily send a positive or negative acknowledgment after each sentence. If that is the case, the sender (the speaker) X will not proceed to the next statement until s/he hears some form of acknowledgment, or otherwise, and in fact, repeat the statement if s/he receives a negative acknowledgement before proceeding. An alternative to this would be a *timeout* strategy.

The speaker X would speak a sentence and wait for some time to hear any kind of acknowledgement. If s/he does not hear anything back, s/he assumes that the previous statement was not received properly, and therefore, repeats the sentence. A form of *sliding window* would mean speaking and acknowledging multiple sentences simultaneously may be 3 or 4 at a time. This is via media between acknowledging each sentence or the full message. We are not aware of this, but we actually follow all these protocols in daily conversations.

Apart from this **error control**, we also take care of **flow control**. This refers to speed mismatch between the speaker and the listener. If the speaker speaks too fast, the listener says *Go slow* or *Please wait* if s/he is taking down. In the world of computers, if the receiving computer is not fast enough, or if its memory buffer is full, which cannot hold any further data, it has to request the sender to wait. This is called *flow control*. Thus, the **data link control layer** is responsible for error control at the sentences level, and also flow control. This layer also decides who is going to speak, and when, by a convention, or in brief; and who has a control of the medium (in this case, the telephone line). This is called **media access control**. This function of media access control becomes necessary because the telephone line is shared between X and Y, and both can and usually do speak simultaneously, causing chaos. In fact, it can so happen that after a pause, thinking that the other party is waiting to hear from you, you may start speaking. However, exactly at the same time, the other party also can start speaking, thinking that you want the other party to speak. This results in a **collision**. The conversation gets mixed up normally, and both the parties realize about this collision and stop talking for a while. Hopefully, the parties will pause for different time intervals, thereby avoiding collision. Otherwise this process repeats. When to start speaking, how long to wait after the collision before restarting, etc., are typical conventions followed at this layer. These are the unwritten protocols of media access control that we follow in our everyday conversation.

In actual practice, we know that when we speak, the electrical signals in the telephone wires change. This is a **physical layer**. There must be a protocol here, too. What this level signifies is things such as how the telephone instruments are constructed, the way the telephone wires are manufactured and laid, the signal levels to denote engaged or busy tone, the signal level to generate a ring, the signal levels required to carry human voice, etc. This is a protocol at a physical level. Obviously, if a telephone and a refrigerator were connected at two ends of a wire, communication would be impossible.

8.1 PROTOCOLS IN COMPUTER COMMUNICATIONS

The same concept of protocols applies equally well to computer communications. To visualize this, let us imagine a network of computers as shown in Fig. 8.2.

Each computer is called a **node**. In distributed processing, different parts of databases/files can and normally do reside on different nodes as per the need. This necessitates transmitting files or messages from one node to the other as and when required. Let us assume that node A wants to transfer a file X to node D. Node A is not directly connected to node D. This is very common, because connecting every node to every other node would mean a huge amount of wiring.

This is the reason that the concept of **store and forward** is used in computer networks. First of all, a path is chosen. Let us say that it is A-F-G-D. Using this path, the node A sends the file to node F. The computer at F normally has to store this file in its memory buffer or on the disk. This storing is necessary, because the link F-G may be busy at this juncture, or node F may have

received a number of messages/files to be sent to other nodes (A, E or G) already and those could be waiting in a queue at node F. When the link F-G is free and ready for transmitting the file from F to G, node F actually transmits it to the node G. Thus, the node F *stores and forwards* the file from A to G. This process repeats until the file reaches the destination node D. This procedure demands that each node maintain a memory buffer to store the file, and some software, which controls the queuing of different messages and then transmitting them to the next node. This software also will have to take care of error and flow control functions in an error-free manner.

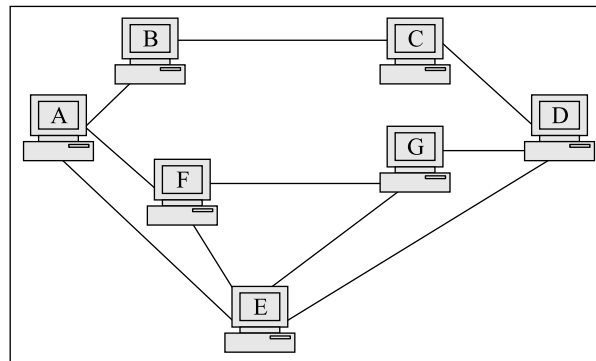


Fig. 8.2 A typical computer network

When the file/message is transmitted, both the nodes (source and destination) as well as all the intermediate nodes have to agree on some basic fundamentals. For example, what is a bit 1 and what is a bit 0? As we know, ultimately bit 0 and 1 correspond to some physical property (voltage level 0 = bit 0, voltage level 5 = bit 1, etc). If there is no understanding between the nodes, the bits could be completely misinterpreted. This understanding or protocol at the physical level is called a **physical layer**. It deals with things like what are bits 0 and 1, the communication modes (serial/parallel, simplex/half-duplex/duplex, synchronous/asynchronous, etc).

How does the *next* node find out whether the file or the message was received correctly or not? And also, how does that node react if it finds an error? There are several methods to detect an error in transmission, as discussed earlier. Obviously, we will need to compute the CRC for the whole file, append it with the data, recompute the CRC on the received data portion at the destination, and compare the received and computed CRC to ensure that they are the same.

There are many ways in which a positive or negative acknowledgement can be sent by the receiving node to the source node. If no error is detected, the receiving node can send a positive acknowledgement back, meaning that everything is OK. However, if an error is detected, the receiving node can either send a negative acknowledgement or choose not to send anything. The latter is called **time out**. In this method, the source node can wait for some time for a positive acknowledgement and having not received it in a specific time, conclude that the file has not been received OK at the destination and then send it again. This is a good method, excepting that when the source node starts sending the file again, the positive acknowledgement (*OK* message) from the receiving node could have been already half way to the source node. When this acknowledgement is received at the source node, it will be too late for the source node because the file/message would have been already sent twice to the destination node. There is normally a protocol to handle such a situation (e.g., the receiving node discards the second copy of the file). A surer way is to

definitely send either *OK* or *NOT OK* message back, and not to use the time out method, i.e., wait until either a positive or negative acknowledgement is received. However, this entails long waits because these messages themselves could take a long time to travel, due to heavy network traffic. The overall network efficiency in this case reduces, as the source node has to wait until it receives *some* acknowledgement.

All these functions of error detection, acknowledgements and retransmissions are clubbed under a name *error control*, and constitute an important part of the communications software, i.e., the **data link layer** in the OSI terminology, residing at every node; i.e., the source, destination as well as all the intermediate nodes, because the message has to reach correctly to the *next* node first, before it reaches to the destination node correctly. The data link layer also takes care of flow control, to take care of the speed mismatch between any two adjacent communicating computers. If the sending computer sends data too fast, it can get lost at the destination. The speeds, therefore, have to be continuously adjusted or monitored. This is called *flow control*.

If an error is detected, the entire file will have to be retransmitted. If the file size is large, the probability of an error is higher, as well as the time that it will take for retransmission. Also, the chances of error in retransmission are also higher. This is the reason that large messages (such as a file) are broken down in smaller chunks or blocks. These are called **packets** or **frames**. Another reason why data is sent in packets is when two pairs of computers want to use a shared transmission line. Imagine that computer A wants to send a big file of 10 MB to computer D by a route A-F-G-D. Also, at the same time, computer F wants to send a small file of 2 KB to computer G. Further suppose that the transmission of the big file over the link F-G starts momentarily ahead of the smaller file transmission over F-G. Assuming that only one pair of computers can use one transmission exclusively; the smaller transmission will have to wait for a long time before the bigger transmission gets over. Thus, a bigger transmission simply can hold up smaller transmissions, causing great injustice. Thus, it is better that each communication party break down their transmission into packets and takes turn to send down packets. Thus, both the files are broken down into packets first. At node F, a packet from the big file is followed by a packet from the small file, etc. This is called *Time Division Multiplexing*, as we have seen. At the other end (G), the smaller file is reassembled and used, whereas packets for the bigger file are separated, stored and forwarded to node D.

Obviously, every packet will have to have a header containing the source address, destination address, packet number and CRC. The destination address is used for *forwarding* or *routing* the packet to the *next* node, and ultimately to the final destination. The packet number helps in re-assembling the packets in case they reach the destination out of sequence. The CRC is used for the error control.

As we have discussed earlier, there are two ways in which the path can be chosen. One is the virtual circuit approach, and the other is the datagram approach. As we know, in a virtual circuit, the path is chosen in the beginning and all the packets belonging to the same message follow the same route. For instance, if a route A-F-G-D is chosen to send the file from A to D, all the packets of that file will traverse by the same route. At D, therefore, they will be received in the same order only, thereby avoiding the function of re-sequencing. This is because, even if packet 2 is received erroneously by node G from node F, node G will ask for its retransmission. Node F will then retransmit packet 2, and before sending packet 3, wait to ensure that node G has received packet 2 without any error. It will send packet 3 only after ensuring this. All this necessitates maintaining many buffers at different nodes for storing and forwarding the packets. As against this, in datagram,

the entire circuit is not predetermined. A packet is sent to the *next* node on the route, which is the *best* at that time, and will take the packet to the ultimate destination.

Choosing a path or *routing* is not a simple task by any stretch of imagination. Remember, each node is receiving many packets from different nodes to be temporarily stored and then forwarded to different nodes. For instance, node F in Fig. 8.2 can have packets received from A to be forwarded to E or G, or meant for it. It can also have packets received from E to be forwarded to A or to G, or to D via G, or packets meant for it. Node F can be receiving packets from node G meant for nodes A, E or for itself. In addition, node F itself will want to send various packets to different nodes. Therefore, the buffer of node F will contain all these packets. The source and destination addresses come handy in keeping track of these packets. You can imagine a buffer memory at node F, where all these packets are stored and then a scheduling algorithm picks them up one by one and sends or forwards them based on the destination node and the route chosen.

Now, to send the data from node A to node D, should it be sent via A-F-G-D or A-B-C-D or A-E-D- or A-F-E-D or A-F-G-E-D or A-F-E-G-D-? Apparently, A-E-D seems to be an obvious answer, as AED appears to be the shortest route. However, looks can be deceptive. Node E's buffer may be full at a given moment due to a message to be sent to node A from nodes G or D. If we follow a First Come First Serve (FCFS) method for forwarding messages, there will be a long wait before our message received from A will be forwarded to D.

This is an example of network congestion. These congestion levels have to be known before the route is chosen. Also, a path may be required to be chosen from any node to another. Therefore, this information about congestion or load on all the nodes and all the lines should be available at every node. Each node then has algorithms to choose the best path at that moment. We have studied routing algorithms earlier precisely for this reason. This, the **network layer** in the OSI parlance residing at every node, again is an important part of communications software.

Note that although we have shown the network to be consisting of only the computers called nodes, in real life, it is not so simple. Since these computers in a network are used for specialized purposes (such as running an application program or serving files on request), the job of routing packets from the sending computer to the receiving computer is handled by dedicated computers called routers. As we know, a router is a special computer that has the sole job of routing packets between the various computers on a network. It decides which packet to forward to which next node, so that it can ultimately reach the final destination. The necessary routing software runs inside the router to carry out this routing process. Therefore, although we have not shown for the sake of simplicity, in real life, we would have a number of routers connecting the various portions of a network to each other.

As we know, in case of the datagram approach, different packets belonging to a single message can travel by different routes. For a packet, a decision is taken about the *next* node to which it should be sent. For instance, at a given moment, the node F as well as the line A-F could have the least congestion (as compared to A-E and A-B). Therefore, the packet is sent via the route A-F. It takes a finite time for the packet to reach the node F, and then for the node F to check the CRC and send back the acknowledgement. Only after this, the node A decides to send the next packet. However, during this time interval, a number of packets could have arrived at node F from node E, to be forwarded to either A or G, or the ones meant for F itself. Therefore, the congestion at node F may have increased. Hence, the next packet could be sent by node A via the route A-E to be ultimately forwarded to D.

Therefore, different packets belonging to a message may not travel by a given predetermined route. In this case, it is possible that packet 3 may arrive before packet 2 at node D. This necessitates the function of re-sequencing and making sure that the entire message has been received without error. One could think of a CRC for the entire message level to be recomputed and matched before acknowledging the error-free receipt of the whole message. This packet consisting of the acknowledgement for the entire message will travel from the destination node to the source node. This function of ensuring in sequence and error-free receipt of the entire message and its acknowledgement retransmission is again a part of communication software, typically the *transport layer* in OSI parlance. It is clear that in case of the virtual circuit approach, there is a guarantee that packets will arrive at the destination in the order that they were sent, because, in this case, a route is (also called **Virtual Circuit Number – VCN**) is chosen in the beginning itself. It is used for all the packets belonging to that message. This is also why the packet in the virtual circuits does not require the full source and destination addresses. It only requires the Virtual Circuit Number (VCN). The routing tables maintained at the various nodes maintain the VCN and the *next node* entries. They are sufficient for routing. The datagram approach demands that the packet carry the source and destination node addresses, which can be utilized for routing, and finding the *next node* each time by using routing algorithms as studied earlier.

We will realize that there are two types of protocols. Some protocols are necessary between any two adjacent nodes and generally they operate at a packet level, i.e., they make sure that the next adjacent node receives a packet or frame correctly. In OSI parlance, **physical data link** and **network layers** are the layers that belong to this category. The other type of protocols is between the end points, i.e., the source node and the destination node (nodes A and D in this example). They make sure a **connection** is established between these two points, **sessions** started and terminated properly, messages (and not packets) are sent/received and acknowledged properly, and necessary data encryption/decryption or compression/decompression and code conversions/translations are done before handing the message over to the destination node. These are typically **transport**, **session**, **presentation**, and **application** layers in the OSI parlance. Figure 8.5 (see Section 8.2.1) depicts this.

Actually, communication software dealing with algorithm for error/flow control, routing, data compression, encryption, etc., could have been coded in one single program. However, such a program would have been difficult to code and maintain. It is for this reason that this function is divided into its logical parts or modules called layers. Using this concept, many manufacturers started coding their communication software in different number of layers. Thus, there was chaos.

Finally, the standards body ISO decided that there has to be a standard for this, so that different computers by different manufacturers could communicate with one another very smoothly. They came up with seven-layer architecture for this purpose. Regardless of the number of layers, all these functions described above have to be taken care of by any communication software, and this software has to reside at every node. Today, OSI has become a standard with which you can compare, though very few have actually implemented the OSI layers exactly as they are described in the standard. Therefore, OSI is actually a *reference model*. We will study it from this perspective.

8.2 THE OSI MODEL

8.2.1 Introduction

The OSI model is structured on seven layers as described in Fig. 8.3.

Layer Number	Layer Name
1 (Lowest) →	Physical
2	Data link
3	Network
4	Transport
5	Session
6	Presentation
7 (Highest) →	Application

Fig. 8.3  OSI layers

The usual manner in which these seven layers are represented is shown in Fig. 8.4.

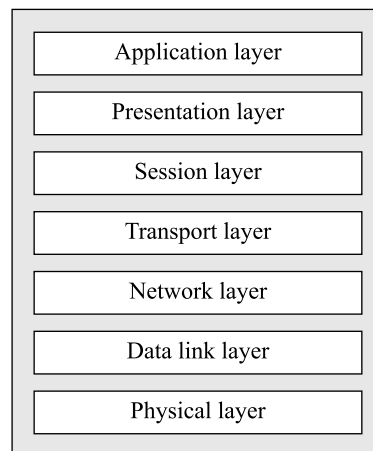



Fig. 8.4  OSI layers arranged in a hierarchy

Let us now study Fig. 8.5. Suppose host X wants to send a message to another host Y. This message would travel via a number of intermediate nodes. These intermediate nodes are concerned with the lowermost three OSI layers, i.e., physical, data link and network, as shown in Fig. 8.5. The other four layers are used by the sender (X) and the recipient (Y) only. Therefore, they are called end-to-end layers.

Note that within a host (either X or Y in this example), each layer calls upon the services of its lower layer. For instance, layer 7 uses the services provided by layer 6. Layer 6 in turn, uses the services of layer 5, and so on. Between X and Y, the communication appears to be taking place between the layers at the same level. This is called **virtual communication** or **virtual path** between X and Y. For instance, layer 7 on host X *thinks* that it is communicating directly with layer 7 on host Y. Similarly, layer 6 on host X and layer 6 on host Y have a *virtual communication* connection between them.

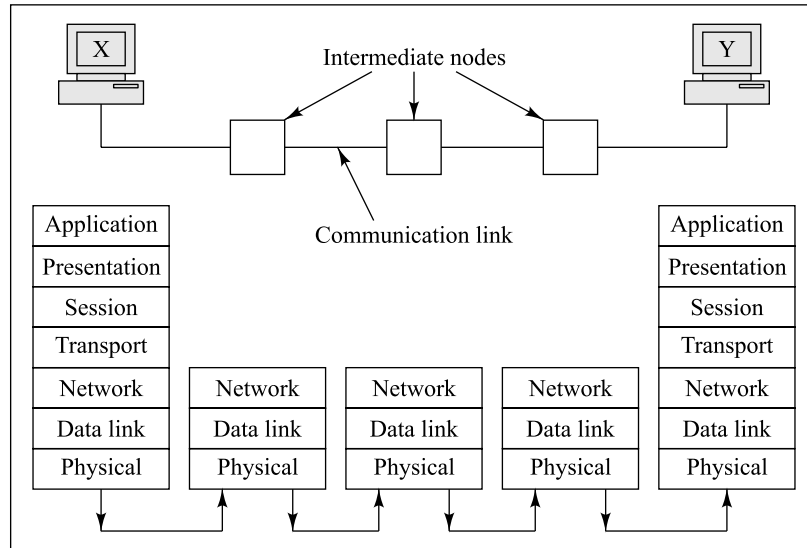


Fig. 8.5 Communication between hosts X and Y using the OSI layers

It is pointless keeping all the communication software functions in every node. Therefore, the functions of the first three layers are contained in a special computer called *router*. You could, now, construct a network of all routers, and imagine that the nodes are attached to the various routers as shown in Fig. 8.6, which is the same as Fig. 8.2, except that we employ routers.

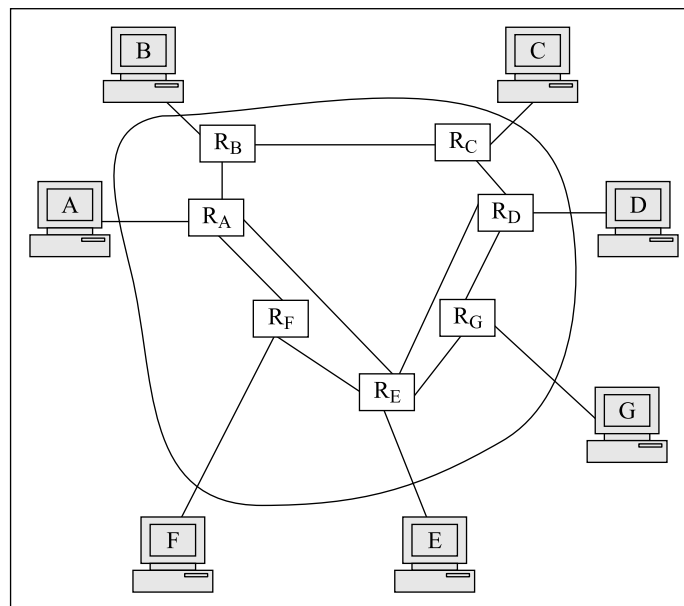


Fig. 8.6 Routers in a network

All that we said about data link layer functions, routing, etc., is still valid as we can see. When node A wants to send a message to node F, node A sends it to router R_A . After this, it gets through a specific route to get to router R_F , and then it reaches the node F.

8.2.2 Layered Organization

The application layer software running at the source node creates the data to be transmitted to the application layer software running at a destination node (remember *virtual path*?). It hands it over to the presentation layer at the source node. Each of the remaining OSI layers from this point onwards adds its own header to the frame as it moves from this layer (presentation layer) to the bottommost layer (the physical layer) at the source node. At the lowest physical layer, the data is transmitted as voltage pulses across the communication medium such as coaxial cables.

That means that, the application layer (layer 7) hands over the entire data to the presentation layer. Let us call this as *L7 data*, as shown in Fig. 8.7. After the presentation layer receives and processes this data, it adds its own header to the original data and sends it to the next layer in the hierarchy (i.e., the session layer). Therefore, from the sixth (presentation) layer to the fifth (session)

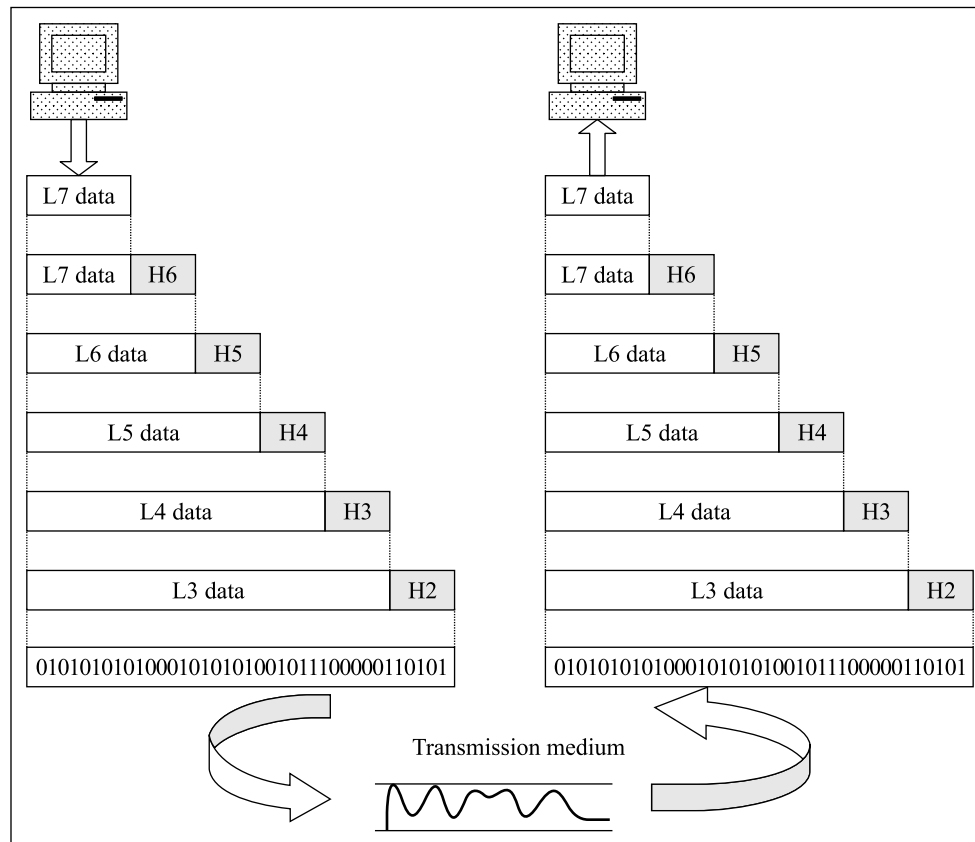


Fig. 8.7 Data exchange using OSI layers

layer, the data is sent as $L7 \text{ data} + H6$ as shown in Fig. 8.6, where $H6$ is the header added by the sixth (presentation) layer.

Now, for the fifth (session) layer, $L7 \text{ data} + H6$ is the input data (see Fig. 8.6). Let us call this together as $L6 \text{ data}$. When the fifth (session) layer sends this data to the next, i.e., the fourth (transport) layer, it sends the original data (which is $L6 \text{ data}$) plus its own header $H5$ together, i.e., $L6 \text{ data} + H5$, and so on. In the end, the original data ($L7$) and all the headers are sent across the physical medium.

Figure 8.7 illustrates this process.

8.3 OSI LAYER FUNCTIONS

8.3.1 Physical Layer

As shown in Fig. 8.8, the **physical layer** is concerned with sending raw bits between the source and destination nodes, which in this case are adjacent nodes. To do this, the source and the destination nodes have to agree on a number of factors such as what voltage constitutes a bit value 0, what voltage constitutes bit value 1, what is the bit interval (i.e., the bit rate), whether the communication is in only one or both the directions simultaneously (i.e., simplex, half duplex or full duplex), and so on. It also deals with the electrical and mechanical specifications of the cables, connectors, and interfaces such as RS 232-C, etc.

To summarize, the physical layer has to take into account the following factors:

1. **Signal encoding** – How are the bits 0 and 1 to be represented?
2. **Medium** – What is the medium used, and what are its properties?
3. **Bit synchronization** – Is the transmission asynchronous or synchronous?
4. **Transmission type** – Is the transmission serial or parallel?
5. **Transmission mode** – Is the transmission simplex, half-duplex or full-duplex?
6. **Topology** – What is the topology (mesh, star, ring, bus or hybrid) used?
7. **Multiplexing** – Is multiplexing used, and if so, what is its type (FDM, TDM)?
8. **Interface** – How are the two closely linked devices connected?
9. **Bandwidth** – Which of baseband or broadband communication is used?
10. **Signal type** – Are analog signals used, or digital?

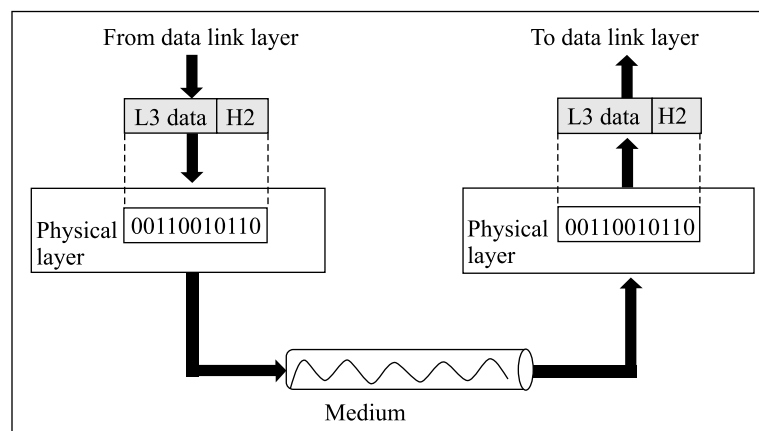


Fig. 8.8 Physical layer between adjacent nodes

8.3.2 Data Link Layer

The **data link layer** is responsible for transmitting a group of bits between the adjacent nodes. The group of bits is generally called *frame* or *packet*. The network layer passes a data unit to the data link layer. At this stage, the data link layer adds the header and trailer information to this, as shown in Fig. 8.9. This now becomes a data unit to be passed to the physical layer.

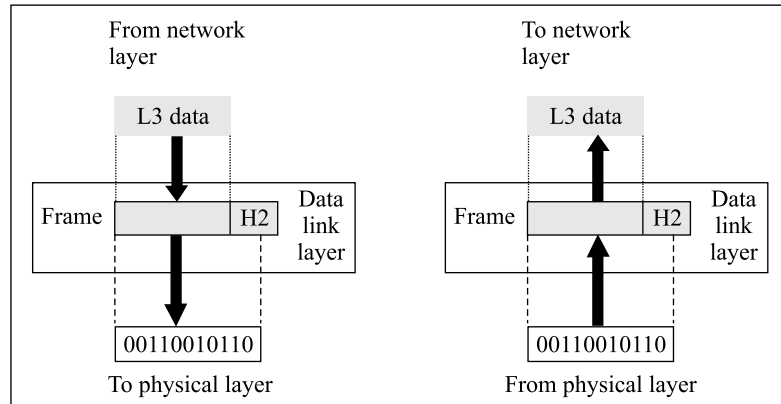


Fig. 8.9 Data link layer between adjacent nodes

The header (and trailer, which is not shown, but is instead assumed to be present) contain the addresses and other control information. The addresses at this level refer to the physical addresses of the adjacent nodes in the network, between which the frame is being sent. Thus, these addresses change as the frame travels from different nodes on a route from the source node to the destination node. The addresses of the end nodes, i.e., those of the source and destination nodes are already a part of the data unit transferred from the network layer to the data link layer. Therefore, it is not a part of the header and the trailer, added and deleted at the data link layer. Hence, they remain unchanged as the packet moves through different nodes from the source to the destination.

Let us illustrate this through an example. Let us refer to Fig. 8.2. Imagine that node A wants to send a packet to node D. If we use the datagram approach, in this case, the logical (i.e., IP) addresses of nodes A and D, say, ADDL (A) and ADDL (D) are the source and destination addresses. The data unit passed by the network layer to the data link layer will contain them. The data unit will look as it is shown in Fig. 8.10. Let us call this D_N .

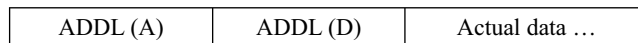


Fig. 8.10 Data unit at the network layer (D_N)

When this data unit (D_N) is passed from the network layer at node A to the data link layer at node A, the following happens:

1. The routing table is consulted, which mentions the *next node* to which the frame should be sent for a specific destination node, which is node D in this case. Let us imagine that the next node is F, based on the congestion conditions at that time, i.e., the path A–F is selected.
2. At this juncture, the data link layer at node A forms a data unit, say D_D , which looks as shown in Fig. 8.11. We will notice that D_D has encapsulated D_N and added the physical addresses of A and F (i.e., those of the NICs of A and F) as ADDP (A) and ADDP (F) to it.

ADDP (A)	ADDP (F)	D_N			Other Info
		ADDL (A)	ADDL (D)	Actual Data	

Fig. 8.11 Data unit at the data link layer (D_D) at node A

- Using the physical addresses of adjacent nodes A and F, the packet moves from node A to node F after performing the flow control functions, as discussed later (i.e., checking if node F is ready to accept a frame from A and at what data rate, etc.). Here, the packet is passed on from the data link layer to the network layer of node F after performing the error control function (i.e., verifying that the packet is error-free). Here, ADDP (A) and ADDP (F) are removed and D_N is recovered. Now, this D_N needs to be sent to the next hop to reach node D. For this, the final destination address, i.e., ADDL (D) is extracted from D_N . The frame now has to be sent from node F to node D.
- Again, the routing algorithm is performed at node F using ADDR (D) as the final destination, and the congestion conditions, etc., and a path is chosen. Let us say that the chosen path is FG.
- The network layer at node F passes D_N to the data link layer at node F. Here, the physical addresses of F and G are added to form the data unit at the data link layer at node F, as shown in Fig. 8.12.

ADDP (F)	ADDP (G)	D_N			Other Info
		ADDR (A)	ADDR (D)	Actual Data	

Fig. 8.12 Data unit at data link layer (D_D) at node F

- This continues until the data unit at data link layer D_D reaches node D. There again, the physical addresses are removed to get the original D_N , which is passed on to the network layer at node D. The network layer verifies ADDL (A) and ADDL (D), ensures that the packet is meant for itself, removes these addresses, and sends the actual data to the transport layer at node D.

The data link layer also performs the flow control function. Based on the speed of the CPUs, transmission, and buffer size and congestion condition, it is determined whether the frame/packet can be sent to the adjacent node, and if so, at what speed. If it can be sent, the node is ready to send the data. However, we have to make sure that the medium is free to carry the frame/packet.

If the connection is multipoint type (i.e., the medium is shared), then the problem of who should send how much data at what times, has to be solved. This problem typically arises in Local Area Networks (LANs), and is solved by the Media Access Control (MAC) protocol. Therefore, in LANs, the data link layer is split into two sublayers, as shown in Fig. 8.13. In this case, LLC takes care of normal data link layer functions, such as error control and flow control, etc.

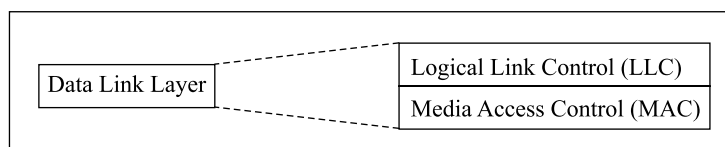


Fig. 8.13 Data link layer in LANs

In Wide Area Networks (WANs), where mostly point-to-point connections are used, this problem does not arise.

Thus, the data link layer performs the following functions:

1. **Addressing** – Headers and trailers are added, containing the physical addresses of the adjacent nodes, and removed upon successful delivery.
2. **Flow control** – This avoids overwriting on the receiver's buffer by regulating the amount of data that can be sent.
3. **Media Access Control (MAC)** – In LANs, it decides who can send data, when and how much.
4. **Synchronization** – Headers have bits, which tell the receiver when a frame is arriving. It also contains bits to synchronize its timing to know the bit interval to recognize the bit correctly. Trailers mark the end of a frame, apart from containing the error control bits.
5. **Error control** – It checks the CRC to ensure the correctness of the frame. If incorrect, it asks for retransmission. Again multiple schemes (positive acknowledgement, negative acknowledgement, go-back-n, sliding window, etc.) exist here.
6. **Node to node delivery** – Finally, it is responsible for error-free delivery of the entire frame/packet to the *next* adjacent node (node-to-node delivery).

8.3.3 Network Layer

The **network layer** is responsible for routing a packet within the subnet, i.e., from the source to the destination nodes across multiple nodes in the same network or across multiple networks. This layer ensures the successful delivery of a packet to the destination node. To perform this, it has to choose a route. As discussed before, a route could be chosen before sending all the packets belonging to the same message (virtual circuit) or it could be chosen for each packet at each node (datagram). This layer is also responsible for tackling any congestion problem at a node, when there are too many packets *stored* at a node to be *forwarded* to the next node. Where there is only one small network based on *broadcast* philosophy (e.g., a single Ethernet LAN), this layer is either absent or has very minimal functionality.

There are many private or public subnet operators who provide hardware links and the software consisting of physical, data link and network layers (e.g., X.25). They guarantee an error-free delivery of a packet to the destination at a charge. This layer has to carry out the **accounting function** to facilitate this billing based on how many packets are routed, when they are routed, etc. When packets are sent across national boundaries, the rates may change, thus making this accounting function complex.

A router can connect two networks with different protocols, packet lengths and formats. The network layer is responsible for the creation of a homogenous network by helping to overcome these problems.

At this layer, a header is added to a packet, which includes the source and destination addresses (logical addresses). These are not the same as the physical addresses between each pair of adjacent nodes at the data link layer as seen before. If we refer to Fig. 8.2 where we want to send a packet from A to D, addresses of nodes A and D (i.e., ADDL (A) and ADDL (D)) are these addresses, which are added to the actual data to form a data unit at the network layer (D_N). These addresses, and in fact, the whole of D_N remains unchanged throughout the journey of the packet from A to F

to G to D. Only physical addresses of the adjacent nodes keep getting added and removed, as the packet travels from A to F to G to D. Finally, at node D, after verifying the addresses, ADDL (A) and ADDL (D) are removed and the actual data is recovered and sent to the transport layer at node D, as shown in Fig. 8.14.

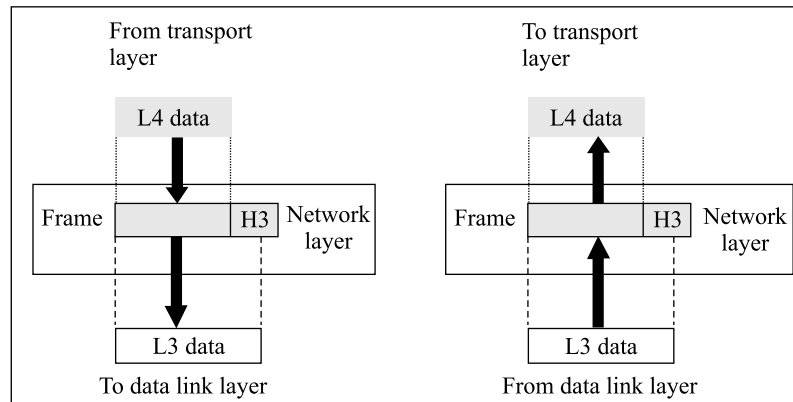


Fig. 8.14 Network layer between adjacent nodes

To summarize, the network layer performs the following functions:

1. **Routing** – As discussed earlier.
2. **Congestion control** – As discussed before.
3. **Logical addressing** – Source and destination logical addresses (e.g., IP addresses).
4. **Address transformations** – Interpreting logical addresses to get their physical equivalent (e.g., ARP protocol). We shall discuss this in detail in the next section of the book.
5. **Accounting and billing** – As discussed before.
6. **Source to destination error-free delivery** of a packet.

8.3.4 Transport Layer

Transport layer is the first end-to-end layer as shown in Fig. 8.5. All the lower layers were the protocols between the adjacent nodes. Therefore, a header at the transport layer contains information that helps in sending the message to the corresponding layer at the destination node, although the message broken into packets may travel through a number of intermediate nodes. As we know, each end node may be running several processes (may be for several users through several terminals). The transport layer ensures that the complete message arrives at the destination and in the proper order and is passed on to the proper application. The transport layer takes care of error control and flow control, both at the source and at the destination for the entire message, rather than only for a packet.

As we know, these days, a computer can run many applications at the same time. All these applications could need communication with the same or different remote computers at the same time. For example, suppose we have two computers A and B. Let us say A hosts a file server, in which B is interested. Similarly, suppose another messaging application on A wants to send a message to B. Since the two different applications want to communicate with their counterparts on remote computers at the same time, it is very essential that a communication channel must be

established not only between the two computers, but also between the respective applications on the two computers. This is the job of the transport layer. It enables communication between two applications residing on different computers.

The transport layer receives data from the session layer on the source computer, which needs to be sent across to the other computer. For this, the transport layer on the source computer breaks the data into smaller packets and gives them to the lower layer (network layer), from which it goes to still lower layers and finally gets transmitted to the destination computer. If the original data is to be recreated at the session layer of the destination computer, we would need some mechanism for identifying the sequence in which the data was fragmented into packets by the transport layer at the source computer. For this purpose, when it breaks the session layer data into packets, the transport layer of the source computer adds sequence numbers to the packets. Now, the transport layer at the destination can reassemble them to create the original data and present it to the session layer.

Figure 8.15 shows the relationship between transport layer and its two immediate neighbors.

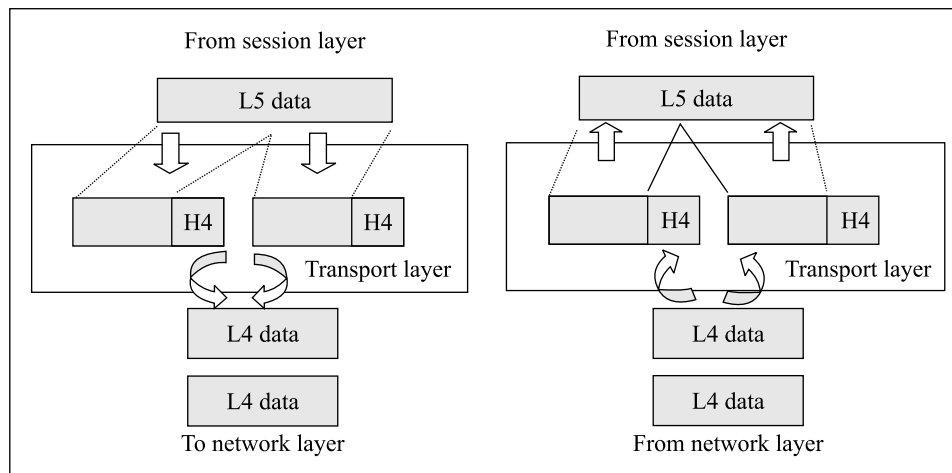


Fig. 8.15  *Transport layer*

The transport layer may also establish a logical connection between the source and the destination. A connection is a logical path that is associated with all the packets of a message, between the source and the destination. A connection consists of three phases, viz., establishment, data transfer and connection release. By using connections, the transport layer can perform the sequencing, error detection and correction in a better way.

To summarize, the responsibilities of the transport layer are as follows:

1. **Host-to-host message delivery** – Ensuring that all the packets of a message sent by a source node arrive at the intended destination.
2. **Application-to-application communication** – The transport layer enables communication between two applications running on different computers.
3. **Segmentation and re-assembly** – The transport layer breaks a message into packets, and numbers them by adding sequence numbers at the source; it uses the sequence numbers at the destination to reassemble the original message.

4. **Connection** – The transport layer might create a logical connection between the source and the destination for the duration of the complete message transfer for better control over the message transfer.

8.3.5 Session Layer

The main functions of the **session layer** are to establish, maintain and synchronize the interaction between two communicating hosts. It makes sure that a session once established is closed gracefully, and not abruptly. For example, suppose that a user wants to send a very big document consisting of 1000 pages to another user on a different computer. Suppose that after the first 105 pages have been sent, the connection between the two hosts is broken for some reason. A question now is, when the connection between the two hosts is restored after some time, must the transmission start all over again, i.e., from the first page? Or can the user start with the 106th page? These issues are the concerns of the session layer.

The session layer checks and establishes connections between the hosts of two different users. For this, the users might need to enter identification information such as login and password. Besides this, the session layer also decides things such as whether both users can send as well as receive data at the same time, or whether only one host can send and the other can receive, and so on (i.e., whether the communication is simplex, half duplex or full duplex).

Let us reiterate our earlier example of transmission of a very big document between two hosts. To avoid a complete retransmission from the first page, the session layer between the two hosts could create subsessions. After each subsession is over, a checkpoint can be taken. For instance, the session layers at the two hosts could decide that after a successful transmission of a set of every 10 pages, they would take a checkpoint. This means that if the connection breaks after the first 105 pages have been transmitted, after the connection is restored, the transmission would start from the 101st page. This is because the last checkpoint would have been taken after the 100th page was transmitted. The session layer is shown in Fig. 8.16.

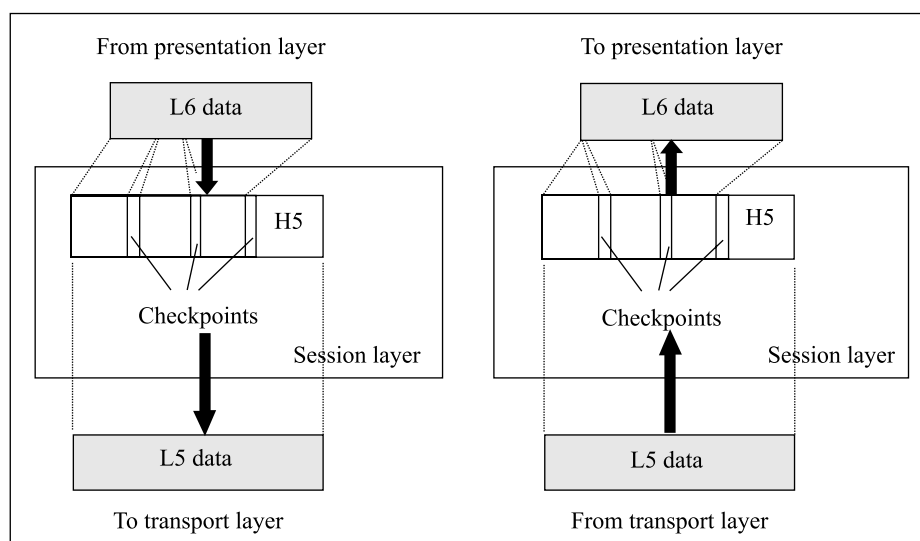


Fig. 8.16 Session layer

In some cases, the checkpointing may not be required at all, as the data being transmitted is trivial and small. Regardless of whether it is required or not, when the session layer receives data from the presentation layer, it adds a header to it, which among other things also contains information as to whether there is any checkpointing, and if there is, at what point.

To summarize, the responsibilities of the session layer are as follows:

1. **Sessions and subsessions** – The session layer divides a session into subsessions for avoiding retransmission of entire messages by adding the checkpointing feature.
2. **Synchronization** – The session layer decides the order in which data needs to be passed to the transport layer.
3. **Dialog control** – The session layer also decides which user/application sends data, and at what point of time, and whether the communication is simplex, half duplex or full duplex.
4. **Session closure** – The session layer ensures that the session between the hosts is closed gracefully.

8.3.6 Presentation Layer

When two hosts are communicating with each other, they might be using different coding standards and character sets for representing data internally. For instance, one host could be using ASCII code for character representation, whereas the other host could be using EBCDIC. The **presentation layer** is responsible for taking care of such differences. It is also responsible for (a) data encryption and decryption for security and (b) data compression and decompression, for more efficiency in data transmission. Figure 8.17 shows the responsibilities of the presentation layer.

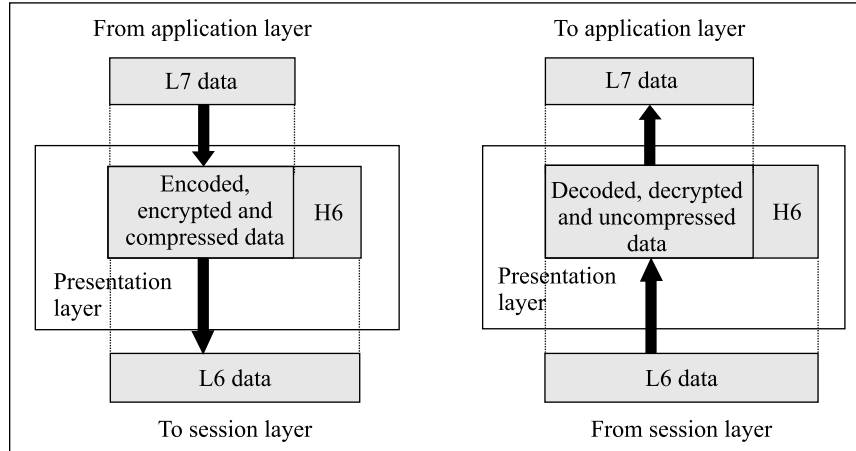


Fig. 8.17 Presentation layer

To summarize, the responsibilities of the presentation layer are as follows:

1. **Translation** – The translation between the sender's and the receiver's message formats is done by the presentation layer if the two formats are different.
2. **Encryption** – The presentation layer performs data encryption and decryption for security.
3. **Compression** – For efficient transmission, the presentation layer performs data compression before sending and decompression at the destination.

8.3.7 Application Layer

The **application layer**, the topmost layer in the OSI model, enables a user to access the network. The application programs using the network services also reside at this layer. This layer provides user interface for network applications such as remote log in (TELNET), World Wide Web (WWW), File Transfer Protocol (FTP), electronic mail (email), remote database access, etc. The users and application programs interact with a physical network at this layer. This should not be confused with the application system like accounting, purchasing, etc. If an accounting application requires an access to a remote database, or wants a file to be transferred, it will invoke the appropriate application layer (e.g., FTP). Thus, this layer can be considered as *consisting of the application* such as FTP, email, WWW, etc., which are the different ways in which one can access the network services. Thus, the application layer provides an abstract view of the layers underneath, and allows the users and applications to concentrate on their tasks, rather than worrying about lower level network protocols.

The conceptual position of the application layer is shown in Fig. 8.18.

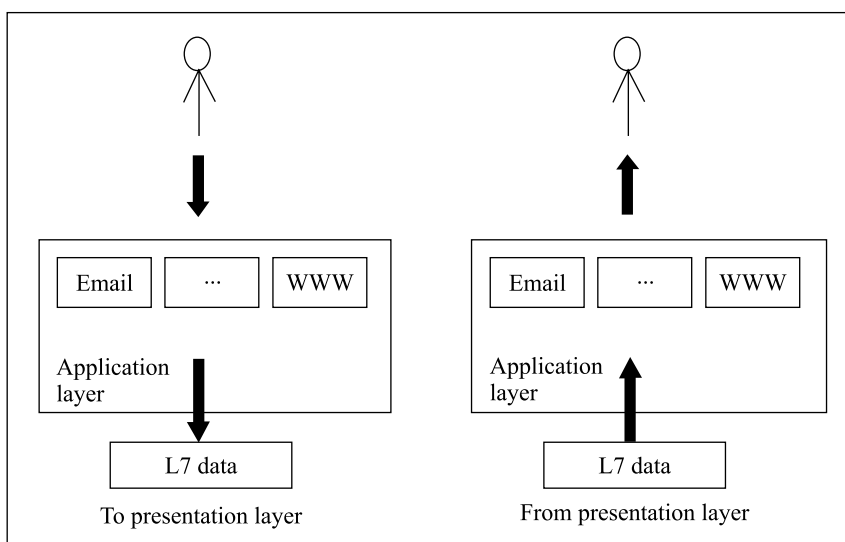


Fig. 8.18 Application layer

To summarize, the responsibilities of the application layer are as follows:

1. **Network abstraction** – The application layer provides an abstraction of the underlying network to an end user and an application.
2. **File access and transfer** – It allows a user to access, download or upload files from/to a remote host.
3. **Mail services** – It allows the users to use the mail services.
4. **Remote login** – It allows logging in a host, which is remote.
5. **World Wide Web (WWW)** – Accessing the webpages is also a part of this layer.

8.4 QUEUING THEORY AND M/M/1 QUEUES

In simple terms, **queuing theory** refers to the mathematical study of waiting queues or lines. It can be used to analyze various aspects related to the transmission or communication systems. Several performance-related aspects pertaining to a network can be computed by using the queuing theory, such as the average waiting time for a transmitter in a system. David Kindall suggested the notation of A/B/C to describe the various aspects pertaining to a queuing system model.

Here,

A = Inter-arrival time distribution

B = Service time distribution

C = Number of servers

For example, M/M/1 represents a single server with infinite capacity and infinite calling population. This kind of technique can be implemented, say, in a courier office that has only one employee. Here, the customer arrives, accomplishes her/his work by dealing with the single employee, and leaves. Using this technique, mathematicians can easily compute the time period for which the customer has to wait on average, etc.

SUMMARY

Protocol means convention. When computers need to communicate with each other either to exchange information or for sharing common resources, they must agree to and use a common protocol, so that the differences in their topologies, architectures, physical properties, and hardware characteristics are hidden from each other. In absence of a common protocol, different computers and computer networks would not be able to communicate with each other.

There are a number of requirements for data communication, such as data transmission, flow control, error control, routing, data compression, encryption, etc. It would be unwise to club all these features in a single piece of software as it would be too bulky. Therefore, all these features are logically subgrouped and then the subgroups are further grouped into groups called layers. The OSI model of communications protocols defines seven such layers as physical, data link, network, transport, session, presentation, and application. Each layer has an interface with its adjacent layers, and performs specific functions. Between two computers, the communication happens between their respective layers. Within a computer, the message flows between the seven layers of the OSI model. All this is hidden from an end user.

The physical layer is concerned with sending raw bits between the adjacent nodes, across the communication medium. To do this, the source and the destination nodes have to agree on a number of factors such as what voltage constitutes a bit value 0, what voltage constitutes bit value 1, what is the bit interval, whether the communication is in only one or both the directions simultaneously, and so on.

The data link layer is responsible for transmitting a group of bits between the adjacent nodes. The group of bits is generally called *frame* or *packet*. The network layer passes a data unit to the data link layer. The data link layer is then responsible for features such as flow control, media access and control, error control, etc.

The network layer is responsible for routing a packet within the subnet, i.e., from the source to the destination nodes across multiple nodes in the same network or across multiple networks. This layer ensures the successful delivery of a packet to the destination node.

The transport layer is responsible for host-to-host message delivery, application-to-application communication, segmentation and re-assembly, and logical connection management between the source and the destination.

The main functions of the session layer are to establish, maintain and synchronize the interaction between two communicating hosts. It makes sure that a session once established is closed gracefully, and not abruptly.

When two hosts are communicating with each other, they might be using different coding standards and character sets for representing data internally. The presentation layer is responsible to take care of such differences. It is also responsible for (a) data encryption and decryption for security and (b) data compression and decompression for more efficiency in data transmission.

The application layer, which is the topmost layer in the OSI model, enables a user to access the network. Application programs using network services also reside at this layer. This layer provides user interface for network applications such as remote file transfer, electronic mail, database access, World Wide Web, etc.

KEY TERMS AND CONCEPTS

Accounting function	Media Access Control (MAC)
Addressing	Medium
Address transformation	Multiplexing
Application layer	Negative acknowledgement (NAK)
Application to application communication	Network abstraction
Bandwidth	Network layer
Billing	Node
Bit synchronization	Node to node delivery
Collision	Packet
Compression	Physical layer
Congestion control	Positive acknowledgement (ACK)
Connection	Presentation layer
Data link layer	Protocol
Dialog control	Reassembly
Encryption	Remote login (TELNET)
Error control	Routing
File access and transfer	Segmentation
Flow control	Session
Frame	Session closure
Host to host delivery	Session layer
Interface	Signal encoding
Link control	Signal type
Logical address	Store and forward
Mail services	Subsession

Synchronization
Translation
Transport layer
Transmission mode
Transmission type

Time out
Topology
Virtual Circuit
Virtual communication
World Wide Web (WWW)

QUESTIONS

True/False

1. The submessage layer takes care of controlling information like how to send positive or negative acknowledgement.
2. Link control level is at a lower level than the submessage level.
3. Each node maintains a memory buffer to store a file and some software, which controls the transmission of the file to the *next* node in an error-free manner.
4. Every packet needs to have a header containing source address, destination address, packet number and CRC.
5. The OSI model consists of five different layers.
6. Network layer is responsible for routing a packet within the subnet.
7. Data link layer enables communication between two applications residing on different computers.
8. It is the transport layer, which decides whether both users can send as well as receive data at the same time or whether only one host can send and the other can receive.
9. The main functions of the presentation layer are to establish, maintain and synchronize the interaction between two communicating hosts.
10. The application layer provides an abstracted view of the layers underneath.
11. Things like translation, encryption and compression are taken care of by the presentation layer.
12. Application layer is responsible for network abstraction, file access and transfer, and mail services.

Multiple-Choice Questions

1. NAK is a _____ acknowledgement.
 - (a) positive
 - (b) negative
 - (c) neutral
 - (d) None of the above
2. The speed mismatch between the sender and the receiver is called _____.
 - (a) error control
 - (b) speed error
 - (c) flow control
 - (d) transmission control
3. In order that a bigger transmission does not overhaul a smaller one, the data is sent in the form of _____.
 - (a) boxes
 - (b) baskets
 - (c) groups
 - (d) packets
4. The _____ layer is the lowest layer in the OSI model.
 - (a) physical
 - (b) transport
 - (c) session
 - (d) application

5. The _____ layer is the topmost layer in the OSI model.
 - (a) physical
 - (b) transport
 - (c) session
 - (d) application
6. The intermediate nodes are concerned with the _____ layers only.
 - (a) top 3
 - (b) middle 3
 - (c) bottom 3
 - (d) topmost, middle and bottommost
7. The _____ layer is responsible for node-to-node delivery of packets.
 - (a) physical
 - (b) transport
 - (c) data link
 - (d) application
8. The _____ layer is responsible for routing packets within or across networks.
 - (a) physical
 - (b) network
 - (c) data link
 - (d) application
9. The _____ layer ensures the correct delivery of a complete message.
 - (a) data link
 - (b) transport
 - (c) session
 - (d) presentation
10. Encryption is handled by the _____ layer.
 - (a) data link
 - (b) transport
 - (c) session
 - (d) presentation
11. The WWW is a _____ layer protocol.
 - (a) physical
 - (b) application
 - (c) data link
 - (d) session
12. Data compression happens in the _____ layer.
 - (a) physical
 - (b) data link
 - (c) network
 - (d) presentation
13. The user is closest to the _____ layer.
 - (a) physical
 - (b) data link
 - (c) transport
 - (d) application
14. The user is farthest from the _____ layer.
 - (a) physical
 - (b) data link
 - (c) network
 - (d) presentation

Detailed Questions

1. Explain the term *protocol* in general.
2. Explain the different layers and their roles in protocols of computer communications.
3. Explain the different layers in the OSI model.
4. Explain the physical layer in the OSI model.
5. How does the data link layer in the OSI model work?
6. Discuss the role of the network layer in the OSI model.
7. How does the transport layer ensure that the complete message arrives at the destination, and in the proper order?
8. Explain how the session layer establishes, maintains and synchronizes the interaction between two communicating hosts.
9. Explain the role played by the presentation layer in handling different data.
10. Explain the topmost layer, i.e., the application layer in the OSI model.

9 Local Area Networks (LAN), Metropolitan Area Networks (MAN) and Wide Area Networks (WAN)

9.0 INTRODUCTION

There are basically three categories of computer networks. They are shown in Fig. 9.1.

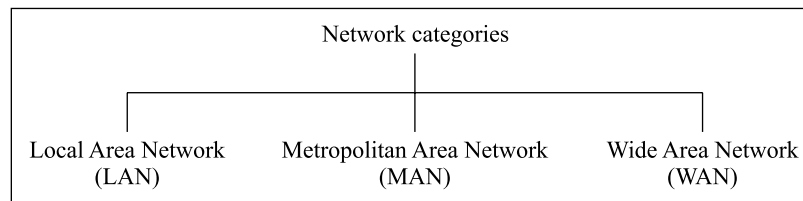


Fig. 9.1 Categories of computer networks

We will study LAN, MAN and WAN in this chapter.

- A **Local Area Network (LAN)** is generally a privately owned network within a single office, building or campus, covering a distance of a few kilometers. The main reason for designing a LAN is to share resources such as disks, printers, programs and data. It also enables the exchange of information. Classically, LANs had data rates of 4–16 Megabits per second (Mbps). Later, 100 Mbps LANs were introduced. Today, LANs with data rates of hundreds of Mbps are possible. LANs typically can use the star, bus or ring topology. The bus topology is popular in the **Ethernet** LANs and **Token Bus** LANs, and the ring topology is popular in the **Token Ring** LANs of IBM. A modified version of Token Ring is **Fiber Distributed Data Interface (FDDI)**. Out of these, Ethernet and Token Ring are the most popular LANs.
- A **Metropolitan Area Network (MAN)** is a network that is designed to cover an entire city. As we have seen, organizations create smaller networks called Local Area Networks (LANs). LANs are privately owned networks within the premises of an organization. However, suppose that an organization wants to connect the computers in its three city offices to each other. In such a case, the organization cannot obviously lay a private network all around the city. If every organization decides to do that, there would be cables everywhere on public land! Instead, the way out is to allow these organizations to utilize the services of existing telephone networks. This approach is called creating a MAN.
- A **Wide Area Network (WAN)** is huge as compared to a LAN or a MAN. A WAN spans across city, state, country or even continent boundaries. For instance, a WAN could be made

up of a LAN in India, another LAN in the US and a third LAN in Japan, all connected to each other to form a big network of networks. The technical specifications of WAN differ from that of a LAN, although in principle, a WAN looks like a *very big LAN*. Consequently, we must study WAN in detail.

9.1 LOCAL AREA NETWORKS (LAN)

LANs usually broadcast their message to all hosts on that LAN. This is because, in the case of LANs, all the hosts share a single transmission medium (wire). The address in the packet or frame enables the destination host to receive that packet, while all the other hosts ignore it. Broadcast networks can be **static** or **dynamic**.

In the static method, each host is given a fixed time slice to send the information. This is quite like the Time Division Multiplexing (TDM) method. If a host does not have anything to send, that time slice is wasted. That is why this method is not very popular.

In the dynamic method, a host can send a frame any time. Thus, if two hosts send a frame at the same time, the two frames could collide with each other. We need some arbitration to solve this problem. A protocol called **Media Access Control (MAC)** performs this job and decides which node can access the medium and when. It is a part of the data link layer in the OSI model.

The dynamic method can be further subdivided in two categories – **centralized** and **decentralized** channel allocation methods. In the centralized method, there is a single entity, for example, a bus arbitration unit, which decides who should send the data next. This is typically a master-slave method. A host wanting to send data can explicitly request this unit for the permission. Alternatively, the master can *poll* on all the slave units to find out whether any one of them has anything to send. In the star topology, the hub can play the role of the master. However, if the arbitration unit goes down, the entire network goes down. In addition, this method is not very efficient either.

In the decentralized method, the arbitration is done more democratically. You do not require any external arbitrator for this. This method is more efficient, and therefore, more popular. However, this requires certain discipline or protocol to be followed by all the hosts. We shall study this approach when we discuss the types of LAN.

The decentralized method can be implemented in two major ways:

1. Carrier sensing – Ethernet using bus topology.
2. Token passing – Token Bus using bus topology uses this method. Token Ring using ring topology and FDDI using a modified version of Token Ring topology uses this method.

We shall now study only the three major and popular implementations, viz., Ethernet (bus), Token Ring and FDDI (token passing).

9.2 ETHERNET

9.2.1 Introduction

Ethernet is the name of a popular packet-switching LAN technology. Ethernet was invented at Xerox PARC in the early 1970s. Ethernet was standardized in 1978. It is an extremely popular LAN technology used by several thousand local area networks around the world.

Ethernet uses a single coaxial cable as the transport medium. All hosts in the Ethernet LAN connect to this cable. A sample Ethernet using the bus architecture is shown in Fig. 9.2.

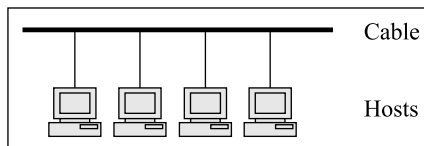


Fig. 9.2 Hosts connected to a cable in an Ethernet

A device called **transceiver** (abbreviation for transmitter and receiver) is used to establish the connection between a computer and the Ethernet. Physically, a small hole is made in the outer layer of the coaxial cable, so that a transceiver can attach to the medium actually carrying the signals. This is shown in Fig. 9.3.

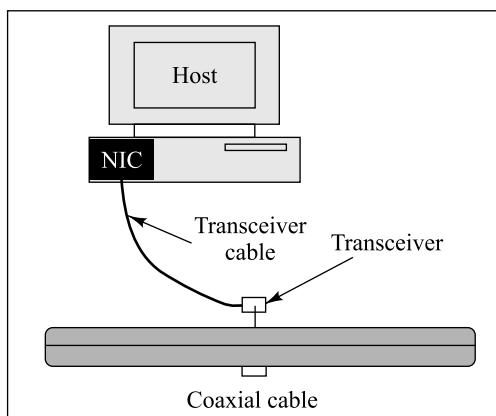


Fig. 9.3 Transceiver connects a host to the cable

The transceiver is responsible for sensing voltages on the cable for interpreting the signals. The transceiver contains analog circuits for interfacing with the cable, and digital circuits for interfacing with the host. Based on the signals on the cable, the transceiver is able to determine when another host is using the cable, i.e., whether the cable is busy. Each host's **Network Interface Card (NIC)** controls the operation of its transceiver using the network software inside the host.

At any point of time, the Ethernet bus can be in any of the following three states:

1. The bus is idle, i.e., no host is sending/receiving any message.
2. The bus carries a legitimate signal, i.e., it is busy.
3. The bus carries an erratic signal generated by a collision, which we will discuss later.

The transceiver constantly *listens* to the bus, i.e., measures the signal level to determine the state of the bus and then decides the course of action, as discussed later.

However, the transceiver does not connect to the host directly. Instead, it connects to a Network Interface Card (NIC), a small card plugged on the motherboard of the host, and functions like a small computer. It has a small CPU, memory and a limited instruction set. It performs all the network-related functions on behalf of the host, such as validating an incoming frame by checking its CRC

to ensure that it is not an erroneous frame, etc. Each NIC bears a unique **hardware address** or **physical address** that identifies a host uniquely. The hardware address is guaranteed to be unique all over the world. The manufacturer of the NIC has a chunk of free addresses, from which it assigns one address to every NIC.

Thus, if the NIC of one computer is replaced with another (because of reasons such as hardware failure or the NIC itself becoming damaged), the hardware address of the computer changes to the address of the new NIC. This is like a SIM card in a mobile phone.

From the point of view of the operating system of the host, the NIC is like a device that accepts from the host, the instructions to transfer the data from/to the host to/from the cable (medium). On receiving these instructions, it controls the transceiver for carrying them out, generates and interrupts when the given task is over, and reports back the status of the task to the host. It performs all these tasks without using the host's CPU, thus freeing the main CPU of all the network-related issues like the Direct Memory Access (DMA) controller does. The logical architecture of a NIC is shown in Fig. 9.4. As shown, it contains an interface to the LAN, and also contains a processor and memory, which are used during the transmission and acceptance of the data. The figure shows NIC as a separate entity. In actual practice, it fits into the host's motherboard. We have shown a logical, conceptual picture.

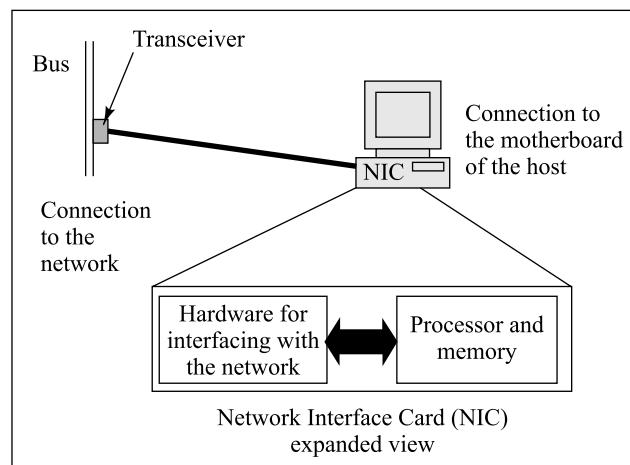


Fig. 9.4 Logical organization of the NIC

Following steps are carried out while transmitting a file/message by a host to another host on the bus:

1. The message is broken into different frames by a higher-level protocol. Each frame has a header and the actual data. The header contains the source address (i.e., the physical address of the NIC) of the computer sending the message, the destination address (i.e., the physical address of the NIC) of the computer receiving the message, and a few other fields like CRC, etc.
2. The frame is sent to the NIC of the source host for transmitting to the destination host. This frame is stored in the memory of the NIC.
3. The NIC now checks the status of the bus with the help of the transceiver and waits endlessly until it finds that the status of the bus is *idle*, i.e., nobody is sending/receiving any message.

4. The NIC sends the data bit by bit when it finds that the bus is idle. The hardware of the NIC computes and inserts the CRC in the header of the frame while transmitting the frame.
5. Let us imagine that two hosts want to use the bus to send the data to their respective destinations. If these two hosts check the status of the bus simultaneously, find it idle and send their respective frames at the same time, there will be a collision. There is a specific way that the Ethernet protocol handles this and recovers from it, which we will learn later.
6. If there is no collision, the frame along with the header starts traveling to all the nodes. After very small finite time, the signal will be all over the bus, as the signal speed is very high. At this juncture, the bus will be *busy* and any other host checking the status of the bus will be told that the bus is busy. Therefore, all the other hosts will have to wait. This is how collision will be avoided.
7. The transceiver of every host receives the signal values from the bus and converts them into bits and sends them bit by bit to the NIC of the host.
8. The bits are stored in the memory of the NIC of the host to form a frame.
9. The NIC of the host compares the destination address in the frame with its own hardware address. It accepts the frame if the two match. Otherwise, the frame is discarded.
10. If the frame is meant for itself (i.e., the addresses match), the NIC of the destination host computes the CRC on the bits received again, and compares it with the one received from the header of the frame. A mismatch indicates an error. The NIC discards such a frame. There is no provision for either a positive or negative acknowledgement in Ethernet. The higher layers have to take care of the missing/erroneous frames.
11. The error-free frames are passed by the NIC to the host at the destination, where they are assembled into a complete message and passed onto the higher layers for further processing.

From this discussion, it should be clear that the transceiver is a relatively simple device, but the NIC is not. The NIC includes a complicated piece of software and hardware.

9.2.2 Properties of Ethernet

The Ethernet uses the bus topology and has a transmission speed of 10 Mbps. It has the following properties:

1. **Broadcast network** – Ethernet is a broadcast network because the transceiver of every host receives every transmission from any host on the network. This is shown in Fig. 9.5. A transceiver does not have any intelligence built in. It simply accepts all the bits of a frame traveling across the Ethernet, and hands them over to its NIC one by one. It is the NIC, which decides if the frame is of any relevance to itself by matching the *destination address* field in the frame with the address of itself, and accordingly either accepts it or discards or ignores it. But the point is that every message travels across the entire cable and all NICs receive it, but only the NIC of the correct destination accepts it, and forwards it to its host computer.
2. **Best-effort delivery** – The underlying hardware does not provide any information to the source or the sending host if the frame sent by it has successfully reached the destination. For instance, suppose the destination host is down for some reason. Then the sender would not know that this is the case, and would assume that its transmission actually reached the destination. If the upper layers of the software running on the host put together all the frames thus received, the host will find some frames missing or wrongly received. It will have to request the source for retransmission. The point is the Ethernet hardware or software does

not take care of this. This will be detected only when the higher layers of network protocols at the destination node try to assemble the entire message.

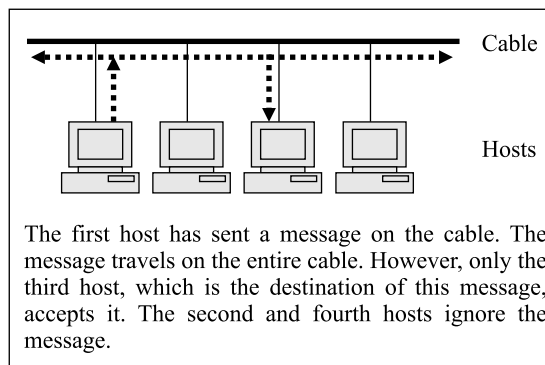


Fig. 9.5 Ethernet is a broadcast network

3. **Decentralized access control** – The control to the transmission cable is distributed. That is, in case of Ethernet, there is no single centralized authority that dictates if a host can transmit data. Instead, the Ethernet uses the approach of **Carrier Sense Multiple Access with Collision Detect (CSMA/CD)** as discussed next.

9.2.3 Carrier Sense Multiple Access with Collision Detect (CSMA/CD)

The idea behind CSMA/CD is quite simple. Multiple hosts can access the Ethernet bus simultaneously through their transceivers and can determine if it is idle by looking for the presence/absence of a carrier wave on the bus. For this reason, it is called Carrier Sense Multiple Access (CSMA). When a host has a frame waiting to be transmitted, its transceiver checks the Ethernet cable to see if another host has already sent some data. That is, it performs a carrier sensing. When the host determines that no other host is using the Ethernet, i.e., the cable is idle, it transmits its own data on to the Ethernet. Of course, it can send only a limited amount of data, so that other hosts also get a chance for data transmission.

When the transceiver of a host begins transmission on the Ethernet, the signal does not immediately reach all parts of the network. Instead, it travels at a speed of about 80% of the speed of light across the Ethernet, which takes some finite time, even if it is very small. Therefore, until the signal reaches another host, that host continues to believe that the cable is idle. Thus, it is quite possible that two transceivers believe that the Ethernet is free for transmission, and can transmit data exactly at the same time. When this happens, the electrical signals of the two transmissions intermingle, and neither remains a meaningful transmission. Such incidents are termed as **collisions**, which produce peculiar erratic signals, which the transceivers of both the sending hosts can detect.

To resolve collisions, while a host transmits the data, the transceiver of that host continues to *listen* to the Ethernet bus to see if a collision has occurred. If it has, the transceiver informs its NIC about it. The NIC stops further transmission and waits for some time before it asks the transceiver to retransmit the data. However, as soon as it detects a collision, it generates a specific jamming signal across the bus, informing all the nodes of the collision. The NIC of the other node trying to send some data also detects the jamming signal and backs off. Now, both the nodes wanting to send

some data wait for a while. The question is how long should they wait? If they wait for the same time, a collision will result again!

The Ethernet standard specifies a **binary exponential back-off policy**, whereby a sender waits for a random time after a first collision, twice as long if a retransmission also results into collision, four times as long if the re-retransmission also results in a collision, and so on. Note that different hosts can choose different random times for waiting. This ensures that if collisions occur, the hosts recover from them quickly. By doubling the waiting period and by making it random (thus different for every host on the Ethernet), chances of another collision after the first one are not very high. For instance, let us imagine that there are three nodes, N1, N2 and N3. Let us imagine that N1 and N2 send the data simultaneously, which results into a collision. Now, N1 waits for time t_1 (randomly chosen by an algorithm at N1) and N2 waits for t_2 (again randomly chosen by an algorithm at N2, and therefore, unlikely to be the same as t_1). After time = t_1 , N1 starts checking the cable status. It may find it busy. It waits until it gets free, and sends a frame. Now let us suppose that N3 also sends a frame at the same time. Again both back off. N1 now waits for time = $2t_1$, whereas N3 waits for time = t_3 (again randomly chosen at N3). In the meanwhile, N2 tries to send a frame after time = t_2 . Hopefully, it finds the cable free, and this time it succeeds in sending the frame. This example should clarify the binary back-off algorithm.

This is shown in Fig. 9.6.

9.2.4 Ethernet Addresses

We had briefly mentioned about hardware addresses of the Network Interface Card (NIC). In the case of Ethernet, a physical address is 48-bit long, and is called **Ethernet address**. An Ethernet address is always unique. The Ethernet address is hardcoded on the NIC. Thus, if we replace the NIC of a computer, its Ethernet address would change. In fact, precisely because of this reason, the higher-level network protocols do not depend on Ethernet address, and instead, use a computer address created in software.

The NIC uses the Ethernet address of a host to determine whether a frame received from the transceiver is addressed to its host or not. Since every NIC receives every frame traveling on the Ethernet regardless of whether it is addressed to that host, the NIC must perform this filtering to accept only those packets that are meant for its host, and discard others. Thus, the host itself is saved from being overwhelmed with a lot of unwanted frames.

9.2.5 Ethernet Frame

The Ethernet is a data link layer connection between hosts. Therefore, the unit of data exchanged by hosts over the Ethernet is called a frame, rather than a packet. An Ethernet frame has a format as shown in Fig. 9.7(a).

Let us describe the various fields in an Ethernet frame.

- **Preamble** – The preamble contains 8 bytes or 64 bits of alternating 0s and 1s to help the receiving hosts synchronize.
- **Destination address** – The 6-byte or 48-bit address of the destination to which the frame is addressed is contained in this field. This is the hardware NIC address that we have studied before. On receiving this frame, the NIC of the destination compares it with its own hardware address and if it matches, accepts it, or else rejects it.

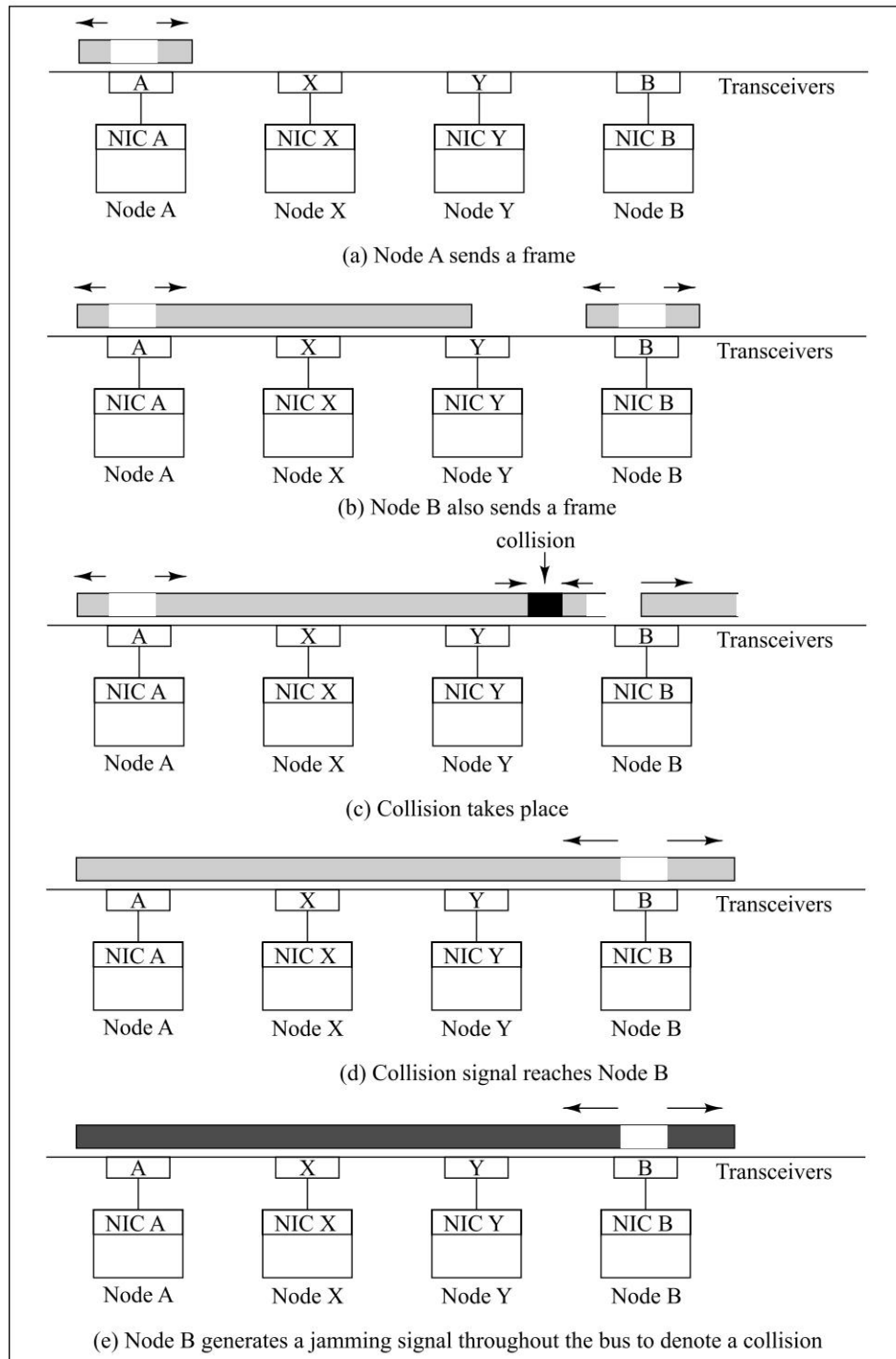


Fig. 9.6  CSMA/CD

Preamble	Destination address	Source address	Frame type	Frame data	CRC
8 bytes	6 bytes	6 bytes	2 bytes	64–1500 bytes	4 bytes

Fig. 9.7(a)  Ethernet frame format

- **Source address** – The NIC of the sending host inserts the hardware NIC address of the sender in this 6-byte or 48-bit field.
- **Frame type** – This field identifies the type of data carried in the frame, thus allowing for multiple higher-level protocols to use Ethernet as the physical medium. The operating system of the destination computer uses this field to determine how to process a given frame.
- **Frame data** – This field contains the actual data of the frame, which can be of variable length.
- **CRC** – This 4-byte or 32-bit field helps the destination NIC to detect transmission errors. The hardware of the NIC of the sending computer computes the CRC of the data and updates this field. The NIC hardware of the destination calculates its own CRC using the received data and compares it with the contents of this field. If they do not match, the destination realizes that a transmission error has occurred and takes an appropriate action, which could simply be discarding the frame. It could, however, inform the higher layer software at the destination node, which may (depending upon the accuracy required in the application) request the source node for retransmission of these erroneous frames.

In summary, the whole operation can be described as follows.

1. The source node software breaks up the data to be sent (the message) into multiple chunks, and sends it to its NIC.
2. The NIC of the source adds the fields such as source address, the destination address, the preamble, etc.
3. The NIC of the source node computes the CRC of the message and prepares a frame. It instructs the transceiver to *listen* to the bus, and look for the *idle* state of the bus.
4. The transceiver finds that the bus is *idle*, and starts sending a frame bit by bit.
5. While it sends the full frame, it continues to *listen* to the bus to see if there is any collision. If it detects it, it generates a jamming signal to let all the nodes know about the collision. After this, the binary exponential back-off algorithm takes over. Finally, at some stage, there is no collision.
6. The frame travels from node to node. The NIC of each node receives the entire frame in its memory, matches the *destination address* in the frame with its own to see whether it is meant for that host. If it is not, it ignores it. If it is, it checks the CRC and if correct, stores it in its memory.
7. It forwards all the frames of the message to the higher layers of the software at the destination node, which reconstructs the original message and uses it.

9.3 VIRTUAL LAN (VLAN)

When the Ethernet technology was first developed, no consideration was given to the logical organization of computers connecting together to form a network. The simple arrangement was to have a central hub connecting all the computers on the floor, and then all such hubs connecting together one large switch. If different groups of people needed different functionalities from this network, it was not possible. This was because only physical segregation on the basis of different

floors/hubs was possible. Logical segregation of computers connected to the network was not possible.

However, later on the picture looked somewhat liked as shown in Fig. 9.7(b). With the introduction of hubs and switches, it was now possible that LANs be configured as per the actual needs. In other words, instead of organizing a LAN in the strict physical sense, it was now possible to organize it on logical grounds. This is possible because the wires or connectors emerging from various computers can be inserted into different hubs, depending on the need. For example, in the Ethernet drawn in our diagram, it is not very difficult to make a change so that a computer currently connected to the first hub instead gets connected to the second hub. Note that this is just a logical change, not a physical one. The computer stays where it is. The hubs and the switch also stay where they are; only the connector's ends move.

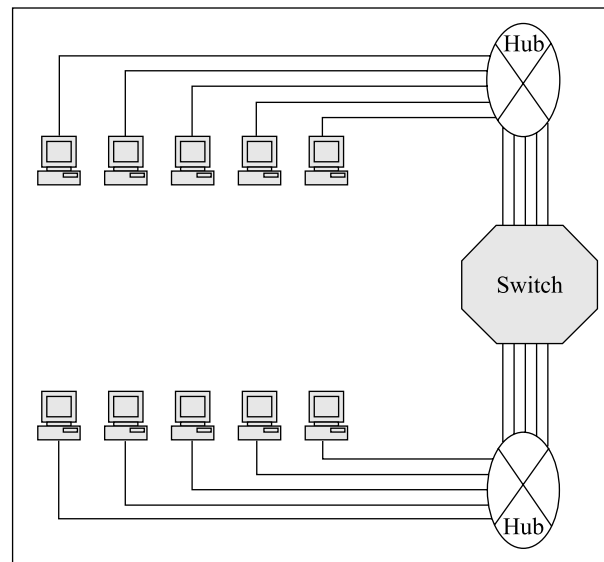


Fig. 9.7(b) Ethernet using hub and switch organization

To enforce appropriate user-level and group-level policies, the system administrator needs to know who is on which part of the LAN. So we cannot also say that this issue is immaterial. Also, depending on the different loads, only one part of the LAN may be heavily used and the other may remain reasonably free. Also, when broadcast (send to all) or multi-cast (send to a select few) messages are sent, it is important to know which all computers such messages should go to. As a reason, we can see that knowing which part of a LAN the computers belong to is vital.

Now if administrators have to pull the network cables in and out to move computers from one hub to another depending on changing requirements, it would be a very tedious process. Imagine a building containing thousands or more computers with dozens of hubs and switches, where a person's computer has to move from one hub to another. What chaos it would cause if this change is physical, i.e., taking the cable out from the current hub and moving it to the new hub.

Consequently, to achieve all these objectives, the technology of **virtual LAN (VLAN)** was conceived. These networks need special switches, which are VLAN-aware. For setting up a VLAN,

the administrator needs to decide roughly how many computers should connect to a VLAN-switch, and also how many such VLAN-switches are necessary. Administrators then use configuration tables to signify the current state of a VLAN. These tables contain entries that tell us which VLANs can be accessed by which network lines or ports. These are grouped in such a way that broadcasting or multicasting is possible by adjusting the entries in this table.

VLAN can be created by using one of the following three methods:

1. Work at the IP address level
2. Work at the physical address level
3. Work at the network port level

Every frame that is sent out now contains information about the VLAN tags also. In other words, every frame itself identifies whether in addition to the regular delivery, it needs to be sent to a few other computers belonging to a particular VLAN. That is, should the frame also be delivered to all the members of a VLAN on the basis of their IP or physical addresses or network ports? Since additional information had to be added for this purpose, the Ethernet frame size was increased from 1518 bytes to 1522 bytes. Technically, the VLAN standard is called IEEE 802.1Q.

9.4 FAST AND GIGABIT ETHERNET

Basic Ethernet transmission takes place at the speed of 10 Mbps. This speed is quite good. However, with emerging changes in computer hardware and software, an attempt was made to try and improve on this. Better hardware and software meant that while keeping the basic Ethernet specifications intact, technologists could work on making the Ethernet really fast. These improvements resulted in the creation of a set of modified Ethernet standards, which are collectively known as **fast Ethernet**. Fast Ethernet provides speeds in the range of about 100 Mbps. This concept was born in the year 1995. The most common of the fast Ethernet standards is called **100BASE-TX**. This standard is supported by a majority of hardware vendors. Fast Ethernet remained popular for about 3 years. Subsequently, it was almost toppled by **gigabit Ethernet**.

In the following paragraphs, we shall take a brief look at both of these more advanced Ethernet standards.

Fast Ethernet

The fast Ethernet technology is conceptually made up of two parts, viz., the **Media Access Controller (MAC)**, and the **Physical Layer Interface (PHY)**. In general, a layer called **Media Independent Interface (MII)** is sandwiched between the MAC and PHY. Usually the MII is found inside a single IC or as a join between two ICs. Of course, it is not mandatory that the interface between MAC and PHY is always an MII. In that sense, it is optional. However, because of MII, in theory, the transmission rate of fast Ethernet is limited to 100 Mbps.

When people speak about the 100BASE-T Ethernet, they refer to a collection of Ethernet networks that are created using the twisted-pair copper cables. Hence, depending on the variations between the various twisted-pair copper cables, we also have various variations of the 100BASE-T standard. Some of the common ones are as follows:

- 100BASE-TX – Provides 100 Mbps data rate over a two-pair cable of good grade.

- 100BASE-T4 – Provides 100 Mbps data rate over a four-pair cable of good grade. However, this standard is now obsolete.
- 100BASE-T2 – Another standard that provides 100 Mbps data rate over a two-pair cable of good grade. This is also obsolete now.

Fast Ethernet networks based on the 100BASE-T standard can span across a distance of 100 meters. Because the other two types of 100BASE-T networks are obsolete, the one that is used most often in real life is the 100BASE-TX network. 100BASE-TX network provides full duplex transmission capabilities.

Also, although originally it was advertised that fast Ethernet networks would work on the existing wiring of the basic Ethernet networks, it was not to be. Due to several technical problems, existing Ethernet networks had to be rewired at least to some extent for making them compliant with the fast Ethernet standards.

With the introduction optical fibers that started replacing twisted copper wires, new standards for fast Ethernet emerged. We can summarize them in the following manner:

- 100BASE-FX: This kind of fast Ethernet using optical fibers has two fibers: one for sending data, and the other for receiving it. Hence, we can use the full-duplex mode. However, to ensure that collisions are detected, it is also used in the half-duplex mode. In that case, the maximum length of the fiber is restricted to 400 meters.
- 100BASE-SX: This type of fast Ethernet technology uses a technique that is cheaper than what is used in 100BASE-FX. As a result, it can also support distances shorter in length as compared to the 100BASE-FX technology. Hence, 100BASE-SX is an alternative to 100BASE-FX when cost is the main factor.
- 100BASE-BX: This kind of fast Ethernet uses a single strand of optical fiber. A special multiplexer is used to split the single signal into transmission signal and receiver signal.
- 100BASE-LX10: This type of optical fiber uses two single-mode fibers.

Gigabit Ethernet

Gigabit Ethernet technology is actually a collection of various standards. These allow transmissions at a rate of about one gigabit per second. Gigabit Ethernet supports both the half-duplex and full-duplex modes. However, in practice, the full-duplex mode is more common. As Ethernet transitioned to fast Ethernet, transmission speeds increased from 10 Mbps to 100 Mbps. The next logical step after fast Ethernet was gigabit Ethernet, which provides transmission speeds of 1 Giga bits per second (i.e., 1 Gbps), or 1000 Mbps.

Initially, the standard developed for gigabit Ethernet in the year 1998 was called **IEEE 802.3z**. This was based on the use of optical fiber as the medium of communication. Later on, other names for the standard emerged, taking cues from fast Ethernet. As a result, these standards were collectively called 1000BASE-X. Note that fast Ethernet was 100BASE-X. In the case of gigabit Ethernet, variations of -X were -CX, -SX, -LX, or -ZX. The original IEEE 802.3z specification also had addendums in the form of two more specifications, namely **IEEE 802.3ab** (using unshielded twisted pair, i.e. UTP) and **IEEE 802.3ah** (which uses optical fiber again).

While initially gigabit Ethernet was used mainly as a backbone for large networks, it also started to gain popularity at the desktop level. Consequently, these days, many computers come with NICs that are compliant with gigabit Ethernet. Now, gigabit networks are also likely to be replaced, as we have 10 Gbps networks emerging.

The following table summarizes the various categories of gigabit Ethernet networks:

Table 9.1 *Gigabit Ethernet standards*

Standard	Specifications
1000BASE-CX	This standard supports a distance of about 25 meters.
1000BASE-SX	This standard uses multimode fiber, and supports a distance of about 220 to 550 meters.
1000BASE-LX	This standard uses multimode fiber, and supports a distance of 550 meters.
1000BASE-LX10	This standard uses single-mode fiber, and supports a distance of about 5 kilometers.
1000BASE-ZX	This standard uses single-mode fiber, and supports a distance of about 70 kilometers.
1000BASE-BX10	This standard uses single-mode fiber, and supports a distance of about 10 kilometers.
1000BASE-T	This standard uses twisted-pair copper wire, and supports a distance of about 100 meters.
1000BASE-TX	This standard uses twisted-pair copper wire, and supports a distance of about 100 meters.

9.5 TOKEN RING

Unlike the Ethernet that uses a bus topology, the Token Ring network is based on the ring topology. IBM's version of the Token Ring network became so popular that the Token Ring architecture itself is referred to as *IBM's Token Ring*, many times. A Token Ring network employs a mechanism called **token passing**, as we shall study shortly.

9.5.1 Basics of Token Ring

All hosts on a Token Ring share the same physical medium, just as the hosts on Ethernet do, and therefore, we need to address this issue of Media Access Control (MAC). However, in the case of Token Ring, the hosts are arranged to form a circular ring. When a host on the ring wants to transmit data, it cannot send it immediately. It must wait for a permission to do so. However, once a host gets the permission for data transmission, it is guaranteed that no other host would be allowed to transmit data at the same time. Thus, a host has an exclusive control over the transmission medium when it is about to transmit data, having obtained a right to do so.

The sending computer transmits a frame, which travels across the ring. This means that each host on the ring has to accept it, check the destination address and if it is not meant for it, forward it along. Only the actual destination after comparing the destination address in the frame with its own (i.e., its NIC's) makes a copy of it (i.e., accepts it), while the other hosts do not make a copy of it. In either case, every host forwards it to the next host on the ring anyway, so that the frame actually comes back to the sender. However, at the destination node, before it forwards the frame, that host checks the CRC to ensure that there are no errors and then not only accepts the frame, but also changes a flag bit in the frame to indicate the receipt of a correct frame. The frame, therefore,

comes back to the sender after covering the entire ring. At this stage, the sender can check the flag bit to verify if the frame was received by the destination successfully, or if there were any errors during transmission. This is shown in Fig. 9.8. This is how the acknowledgement scheme is implemented in Token Ring unlike Ethernet, where there is no such scheme.

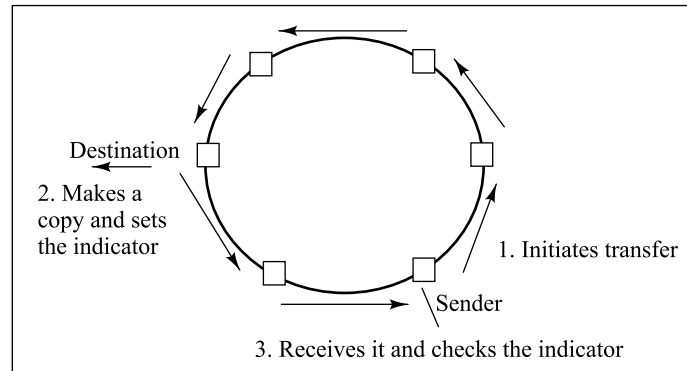


Fig. 9.8 Token Ring

9.5.2 Medium Access Control (MAC)

The crucial point to understand is how does a host know that the medium is free, and that it can send a frame? We have said that unlike Ethernet, where any host can transmit data any time, in the case of Token Ring, a host gets exclusive access to the medium when it is transferring data, and that no other host can perform any transmissions. How is this achieved?

Token Ring does not employ CSMA/CD. Rather, the Token Ring hardware ensures that the permission for data transmission is granted to each host on the Token Ring, in turn. The hardware facilitates this coordination. For this, the hardware uses a special three-byte frame called **token**. A token contains a bit pattern that is completely different from any other data frame, so that the token and data frames can be easily distinguished from each other. The Token Ring hardware makes sure that always there is one and only one token frame on the medium, which keeps circulating over the ring from one host to the next continuously.

The token is the permission for data transmission. Thus, before sending a frame, a host must wait for the token frame to arrive. Once a host receives the token frame, it knows that it has an exclusive access to the transmission medium. Therefore, it temporarily removes the token frame from the transmission medium and sends its data frame to the transmission medium. Once the data frame comes back to it by completing its full journey, the transmitting host then sends back the token frame onto the medium. When the token is withdrawn from the ring, no other host can send any data frame as a data frame can be sent by a host only after receiving the token frame. This takes care of media access in a shared medium. This is shown in Fig. 9.9.

The circulating mechanism of the token frame ensures that every host is given an equal chance for data transmission. Also, if a host has two or more frames to transmit, it must send one frame at a time. This means that small transmissions do not have to wait for long because of other big transmissions. In short, this is an interesting implementation of time division multiplexing.

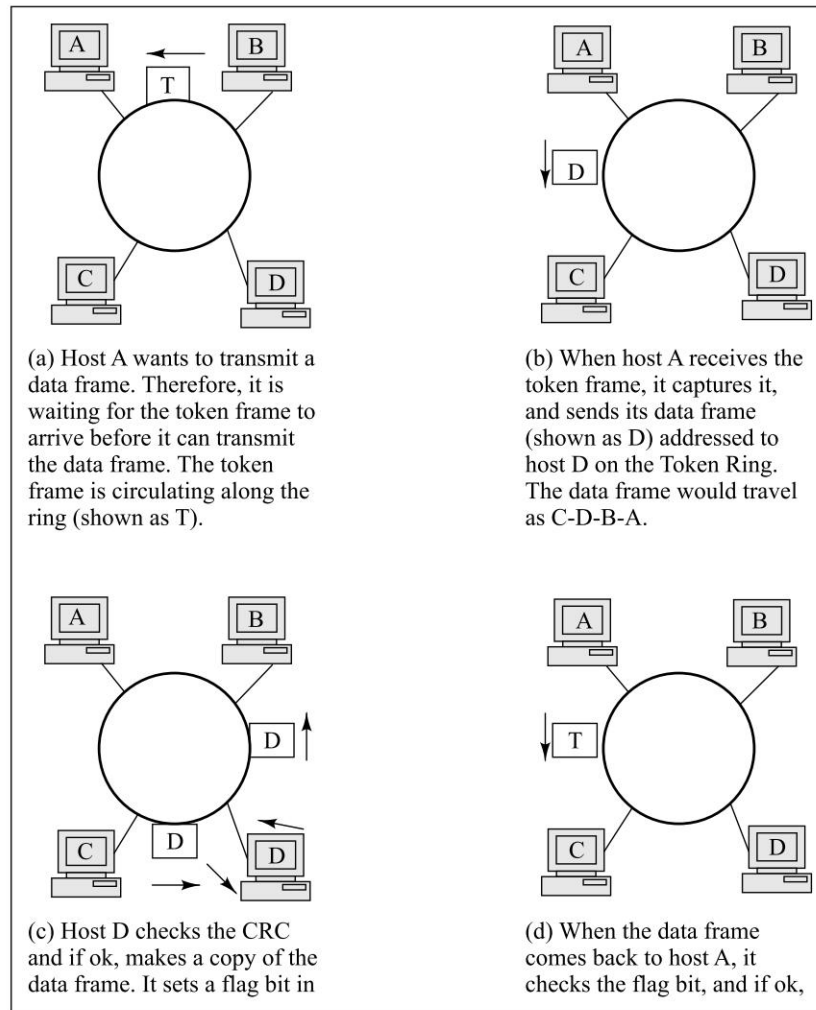


Fig. 9.9 Token passing mechanism used for enforcing Media Access Control (MAC)

In the extreme case, when no host has any data to transmit, the token frame keeps circulating continuously, without any host using it to gain access to the medium.

9.5.3 Addressing Mechanism

The Token Ring addressing mechanism is very similar to that of the Ethernet, with the 48-bit NIC address being used for all transmissions.

9.5.4 Properties of Token Ring

Token Ring networks demonstrate the following properties.

1. **Data rate** – Token Ring supports data rates up to 10 Mbps. The original specification was 4 Mbps. However, it has increased over the years, because of the improvements in the hardware.

2. **Transmission medium** – The ring in the Token Ring network consists of a series of shielded twisted-pair wire sections that link to their immediate neighbors. The output port of one host is connected to the input port of the next, to create a **unidirectional** traffic flow. The output from the final host is connected back to the input of the first host to complete the ring.

9.5.5 Token Ring Frame

The Token Ring specification describes three frame formats, viz., **data**, **token** and **abort**. These formats are discussed one by one.

1. **Data frame** – The data frame of the Token Ring has a format shown in Fig. 9.10.

Preamble	Destination Address	Source Address	Frame Data	CRC	ED	FS
3 bytes	6 bytes	6 bytes	Up to 4500 bytes	4 bytes	1 byte	1 byte

Fig. 9.10 Token Ring data frame format

The fields in the data frame of a Token Ring are as follows:

- **Preamble** – This field internally contains three subfields, each consisting of one byte. The preamble is used for synchronization purposes. One of the flags in this field also indicates that it is a data frame, and not a token frame or an abort frame.
- **Destination address** – The 6-byte or 48-bit address of the NIC of the destination, to which the frame is addressed, is contained in this field.
- **Source address** – The NIC of the sending host adds the NIC address of the sender to this 6-byte or 48-bit field.
- **Frame data** – This field contains the actual data of the frame, which, again can be of variable length, up to a maximum of 4,500 bytes.
- **CRC** – This 32-bit field helps the source and the destination NIC to detect transmission errors. The sender computes the CRC of the data and updates this field. The destination calculates its own CRC based on the received frame and compares it with the contents of this field. If they do not match, the destination realizes that a transmission error has occurred and takes an appropriate action, which could simply be discarding the frame. As we shall see, in case the two CRCs do not match, the destination does not set the *FS* bit (see below) to indicate that there was a transmission error. The sender notes this fact and has to send the frame again. Recall that the sender receives back the frame, which it had transmitted. Therefore, it can recalculate the CRC and do verification on her/his part as well.
- **ED (End Delimiter)** – This one-byte field signifies that the sender's data and control information ends here.
- **FS (Frame Status)** – This one-byte field is the last one in the data frame. The receiver sets it when it receives the frame correctly (i.e., after checking the CRC) to indicate that it was received by it correctly. It serves the purpose of an acknowledgement implicitly (although it is not just an acknowledgement, as we shall see). This is because the same frame then goes back to the sender, who checks this flag to verify if the frame was received by the destination correctly. One byte is occupied by this field as it is more than a mere acknowledgement. It contains two bits to indicate that the (destination) address was recognized and the data was captured (by the destination) correctly. This takes care of the situations where the destination is not present or powered on, as well as those where the destination is present but the frame

cannot be copied (may be because its buffer is full). These two bits occur twice in this byte to increase reliability. Other four bits remain unused.

2. **Token frame** – The format of the token frame is shown in Fig. 9.11.


Start Delimiter (SD)	Access Control (AC)	End Delimiter (ED)
1 byte	1 byte	1 byte

Fig. 9.11  *Token Ring token frame format*

The token frame acts like a placeholder. The fields of this frame serve the following purposes:

- **Start Delimiter (SD)** – This field signifies to a host that a frame is coming.
 - **Access Control (AC)** – This field indicates to the host that the arriving frame is a token frame.
 - **End Delimiter (ED)** – This field signifies to a host the end of the token frame.
3. **Abort frame:** This frame does not contain any meaningful information. It is used by a sender to abort an ongoing transmission for whatever reasons, or by the network monitor to purge old, unwanted frames in case of network errors. Its format is as shown in Fig. 9.12.

Start Delimiter (SD)	End Delimiter (ED)
1 byte	1 byte

Fig. 9.12  *Abort frame format*

These fields have been described earlier:

- **Start Delimiter (SD)** – This field signifies to a host that a frame is coming.
- **End Delimiter (ED)** – This field signifies to a host the end of the abort frame.

9.6 FIBER DISTRIBUTED DATA INTERFACE (FDDI)

9.6.1 Introduction

The **Fiber Distributed Data Interface (FDDI)** network architecture is a LAN protocol standardized by ANSI and other organizations. It supports data transmission rates of up to 100 Mbps, and is an alternative to Ethernet and Token Ring architectures. Originally, FDDI was developed using optical fiber as the transmission medium because only optical fiber could support data rates of 100 Mbps. These days, even copper wires can support such rates, and the copper version of FDDI is called **Copper Distributed Data Interface (CDDI)**. However, we shall concentrate on FDDI.

FDDI uses glass fibers for data transmission, as mentioned before, and therefore, encodes data bits in the form of pulses of light.

9.6.2 Properties of FDDI

The main properties of FDDI can be summarized as follows.

- **Token passing for Media Access Control** – Like the Token Ring protocol, FDDI also uses the concept of a token frame to regulate medium access. The same principles of token frame apply here. FDDI is also a ring-like structure where the network medium starts from a computer, passes through all the hosts in the network, and ends back at the original host.

- **Self-healing mechanisms** – The hardware in FDDI provides mechanisms for detecting and correcting problems on its own, as we shall study.

9.6.3 Operation of FDDI

FDDI operates exactly like Token Ring, with one difference. Token Ring employs a single wire through all the hosts in the network, whereas FDDI employs two. That is, FDDI hardware uses two independent rings to connect to every host, as shown in Fig. 9.13. The figure shows the FDDI network in two different forms, just to signify the ways in which it is depicted in literature.

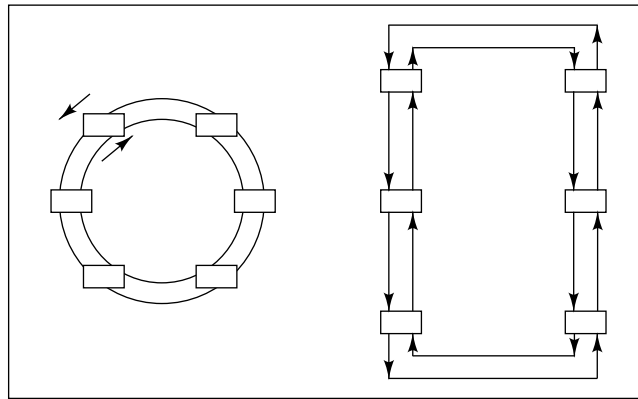


Fig. 9.13 FDDI

Normally, FDDI uses only one ring for data transmission. That is, it works like a Token Ring network. The NIC of each host examines all the frames that circulate around the ring, comparing the destination address in the frame with its own. Like the Token Ring approach, a host keeps a copy of the frame only if the two addresses match, else it simply forwards it along the ring. What is the need of the second ring, then?

9.6.4 Self-healing Mechanism

The self-healing mechanism of the FDDI network is made possible by the second ring. When a network error occurs, or a host is down, the NIC of a host realizes that it cannot communicate with its neighboring host. In such a case, the NIC uses the second ring, which is used as a backup for such failures, for data transmission. This is called **loopback**.

Figure 9.14 shows one such instance where a host is temporarily down for some hardware failure. In such a case, to bypass that host without affecting the rest of the network, the second (backup) ring is used as shown.

The purpose of the second backup ring should now be clear. Whenever the first ring fails, or a host on the ring fails, the second ring is used to create another closed loop, bypassing the failed host to create a new closed loop between all the other hosts using the loopback mechanism, while the fault ring can be repaired.

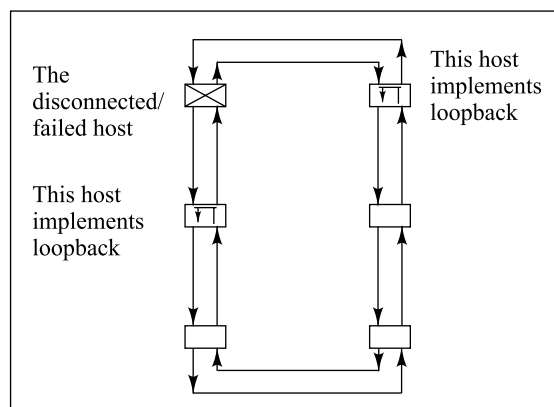


Fig. 9.14 The loopback mechanism using the second ring in case of network problems

9.6.5 FDDI Frame

The FDDI specification describes the frame format, which is very similar to the Token Ring data frame format with a few changes. Let us discuss the FDDI frame format, which is shown in Fig. 9.15.

Preamble	Destination address	Source address	Frame data	CRC	ED	FS
2 bytes	6 bytes	6 bytes	Up to 4500 bytes	4 bytes	1 byte	1 byte

Fig. 9.15 FDDI data frame format

The fields in the data frame of FDDI are as follows:

1. **Preamble** – This field internally contains two subfields, each consisting of one byte. The preamble is used for synchronization purposes.
2. **Destination Address** – The 6-byte or 48-bit address of the NIC of the destination, to which the frame is addressed, is contained in this field.
3. **Source Address** – The NIC of the sending host adds the NIC address of the sender to this 6-byte or 48-bit field.
4. **Frame Data** – This field contains the actual data of the frame. This is also of variable length, up to a maximum of 4,500 bytes.
5. **CRC** – This 32-bit field helps the source and the destination NIC to detect transmission errors. The sender computes the CRC of the data and updates this field. The destination calculates its own CRC based on the received frame and compares it with the contents of this field. If they do not match, the destination realizes that a transmission error has occurred and takes appropriate action, which could simply be discarding the frame. As before, the *FS* field is used for retransmission. Recall that the sender receives back the frame, which it had transmitted. Therefore, it can recalculate the CRC and perform verification on her/his part as well.
6. **ED (End Delimiter)** – This one-byte field signifies that the sender's data and control information ends here.

7. **FS (Frame Status)** – This one-byte field is the last in the data frame. The receiver sets it when it receives the frame to indicate that it was received by it. It serves the same purpose as explained in the case of Token Ring.

9.7 COMPARISON OF ETHERNET, TOKEN RING AND FDDI

Let us summarize the features of the three LAN technologies discussed in this chapter, namely Ethernet, Token Ring and FDDI in a tabular form, as shown in Fig. 9.16.

Network	Access mechanism	Address length	Data rate	Error control (Acknowledgement)
Ethernet	CSMA/CD	48 bits	1–10 Mbps	No
Token Ring	Token passing	48 bits	10–16 Mbps	Yes
FDDI	Token passing	48 bits	100 Mbps	Yes

Fig. 9.16 Comparison of the LAN technologies

9.8 METROPOLITAN AREA NETWORK (MAN)

MAN can be of two types as shown in Fig. 9.17. These two types are **Distributed Queue Dual Bus (DQDB)** and **Switched Multimegabit Data Services (SMDS)**. These services are provided by telephone companies.

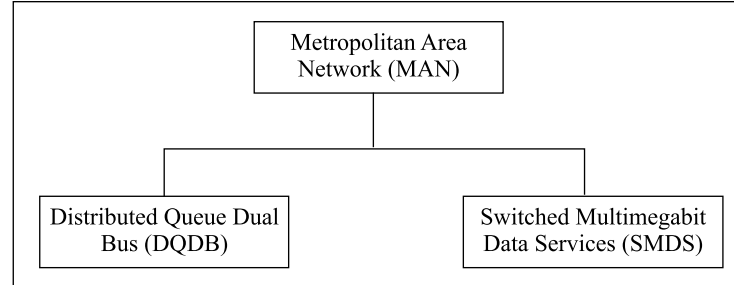


Fig. 9.17 MAN types

In the following sections, we shall discuss these two types.

9.9 DISTRIBUTED QUEUE DUAL BUS (DQDB)

9.9.1 Basics of DQDB

The *Distributed Queue Dual Bus (DQDB)* protocol is a **dual bus** configuration. This means that each host in the network connects to two backbone network lines. The hosts get an access to the transmission medium with an approach that is different from LANs. As we know, in case of Ethernet LANs, the access to the transmission medium is obtained using CSMA/CD; while in case of token ring, it is using token passing. However, in case of DQDB, it is done using a mechanism called **distributed queue**. Hence the name *Distributed Queue Dual Bus (DQDB)*.

Figure 9.18 shows sample DQDB architecture with two unidirectional buses, called bus A and bus B. In the figure, five hosts numbered 1 to 5 connect to these buses. Each bus connects to the hosts on their input and output ports.

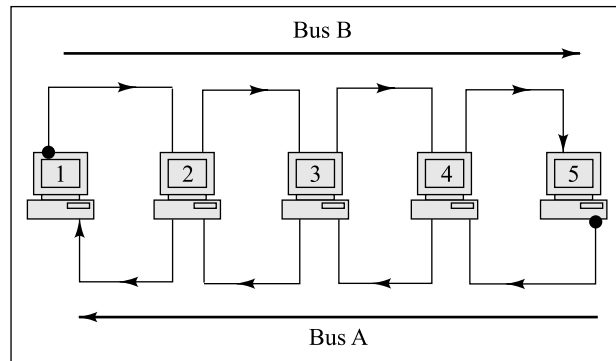


Fig. 9.18 DQDB architecture

9.9.2 DQDB Traffic

As Fig. 9.18 shows, each bus on DQDB supports traffic in only one direction. The two buses that make up DQDB facilitate traffic in the opposite directions. In Fig. 9.18, bus A allows traffic in the direction from host 5 to host 1, whereas bus B allows traffic from host 1 to host 5 in the opposite direction.

The relationships between hosts are expressed in relation to the direction of traffic on the two buses. This architecture gives rise to an interesting set of properties, called **upstream** and **downstream**. From host 3's point of view, hosts 1 and 2 are *downstream* on bus A, whereas hosts 4 and 5 are *upstream* on bus B. By the same logic, for host 5, there are no upstream hosts on bus B, but there are four downstream hosts on bus A. Because of this, host 5 is considered as the head of bus A, and host 1 as its end. By a similar reasoning, host 1 has no downstream stations on bus B, but has four upstream stations on bus A. So, host 1 is the head of bus B, and host 5 is its end.

Data flows on each bus in the form of 53-byte **transmission slots**. These slots are not packets; rather, they are like boxes or containers, which can contain packets. The heads generate slots. Therefore, station 5 on bus A and station 1 on bus B generate empty slots for use, respectively. The data transmission speed depends on how many slots are generated per second. Different possible data rates exist today. An empty slot travels along the bus until a host drops data into that slot and the destination receives it. However, a question comes up here. Since two buses exist, which of them should the sender of the data choose? For this, there is a rule that says that the sending host must choose a bus for which the destination is downstream. If we think deeper, this is quite logical. After all, the slots in a bus travel from the head of the bus to the last host. On that bus, the slots move towards the next downstream host. Therefore, to send data, a sending host must select the bus whose traffic is flowing towards its destination.

Let us illustrate this with example. Figure 9.19 depicts a situation where host 4 sends data to host 2. Let us now think which bus should host 4 select. Should it be bus A or bus B? If we go back to Fig. 9.18, we will realize that the downstream from host 4 to host 2 is on bus A. Therefore, host 4 must choose a slot on bus A.

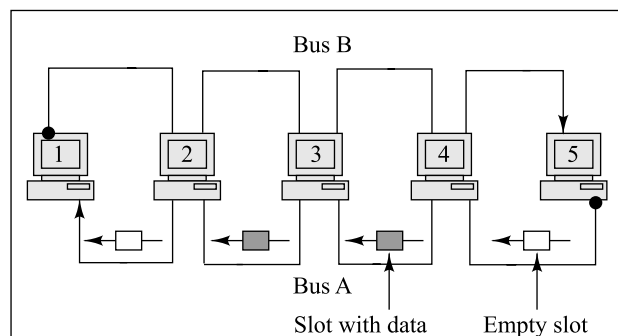


Fig. 9.19 Data transmission in DQDB

As shown in the figure, the transmission from host 4 to host 2 is done as follows:

1. The head of bus A, i.e., host 5, creates an empty transmission slot.
2. When the empty transmission slot reaches host 4, host 4 drops its data into that slot. Host 4 also signifies the destination address as *host 2*.
3. The slot moves along bus A to host 3. Host 3 examines the destination address field in the slot, and realizes that the data in the slot is not meant for it. Therefore, it does not read the data contents of the slot, and instead, lets the slot move along bus A.
4. When the slot reaches host 2, host 2 realizes that the slot is destined for it, by comparing the contents of the destination address field of the slot with its own address. Therefore, it copies the data contents of the slot, marks the slot as *read*, and allows it to move along bus A.
5. The slot reaches the end host (host 1) of bus A, where the slot gets absorbed.

If host 1 sends some data to host 3, it should be clear now that it has to go via bus B. It would follow the same steps as described above, and we shall not discuss them.

There is only one minor variation of this scheme. When the destination host is also the end host, the end host does not need to mark the slot as *read*, because it is the last host, anyway. This means that regardless of whether it marks it as *read* or not, the slot is going to be absorbed by that same host. So, if the destination host is the end host, the *read* flag is not set in the slot.

9.9.3 Medium Access

In DQDB, a host reserves the slot before transmitting its data. The slot reservation is made on the bus that is upstream for the host. The data transmission takes place on the downstream. For example, considering again that host 4 wants to send data to host 2 on bus A, host 4 sets a *reservation bit* in a slot on bus B, which is carrying data traffic in the opposite direction. Since this slot visits all the hosts on bus B, all hosts on bus B come to know that host 4 wants to reserve a slot on bus A.

The reservations made by all hosts are stored in a First In First Out (FIFO) queue structure. That is, the reservations are to be processed in the order that they were made. Each host on the network maintains a copy each of two such queues, i.e., one for each bus. Thus, at any point of time, every host knows how many reservations are pending to be served from the two queues. When a host receives a slot containing a *reservation request*, it adds a new element to the end of the queue to indicate that a new request has to be added to the pending reservation request list. On the other hand,

the moment a host receives an empty slot, it realizes that one of the reservations (from the list of reservation requests in the queues) is being served. To see which node's reservation is being served, it can examine the tail element of its reservation queue. There are two possibilities here:

1. If the reservation request belongs to another host on the bus, the host simply removes the tail element (i.e., the earliest reservation request made, as of now) from its list to record the fact that one of the pending reservations is now served.
2. If the reservation request being served as made by the host itself, it means that the host can transmit data now. Therefore, apart from removing the tail element from the queue, it adds the data to be transmitted to the empty slot.

9.10 SWITCHED MULTIMEGABIT DATA SERVICES (SMDS)

9.10.1 SMDS Basics

Switched Multimegabit Data Services (SMDS) is a high-speed MAN technology. SMDS is aimed at fulfilling the needs of organizations that have a number of LANs spread across different locations in a city. It is a packet-switched datagram service for high-speed MAN data transmissions. SMDS is provided by common carriers. Since it is a switched technology, the subscribers of SMDS pay only for the time that they use its services.

9.10.2 Medium Access

Bell Communications Research developed the SMDS specifications in the 1980s. A LAN can be connected to an SMDS network by a router. This is shown in Fig. 9.20.

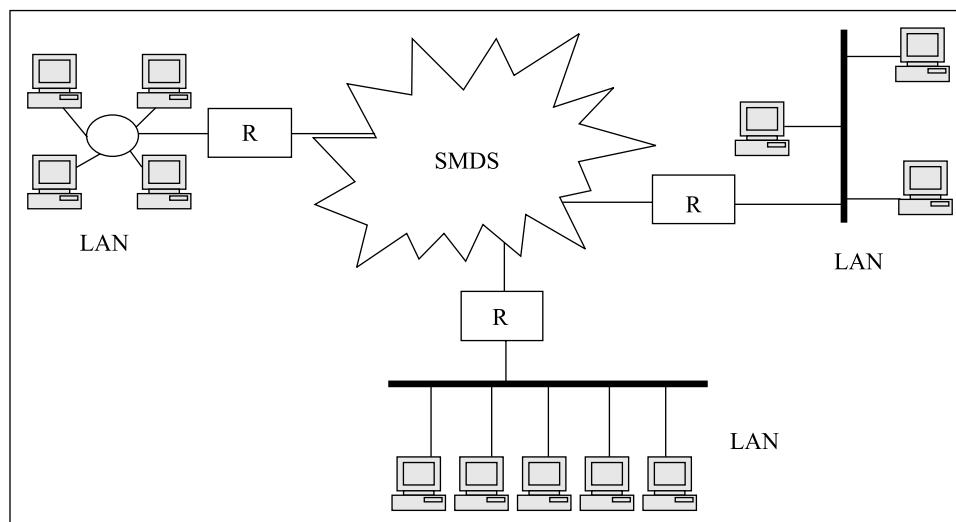


Fig. 9.20 *SMDS*

As Fig. 9.20 shows, many LANs are connected together to form a SMDS MAN. The SMDS network acts like a high-speed LAN backbone, which allows packets from any LAN to travel to any other LAN on the SMDS.

The connection is based on the dual bus technology, similar to what it is in DQDB. At the LAN end, we have a router, and at the SMDS end, we have a switch. This switch belongs to a telephone company's switching office, which, in turn, routes the LAN traffic to the other LAN(s) of the organization, which are connected to the SMDS service. This is shown in Fig. 9.21.

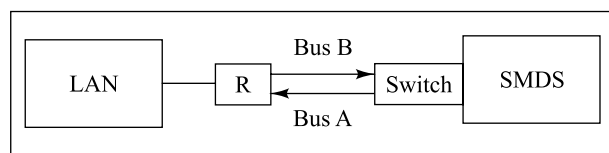


Fig. 9.21 The dual bus connection between LAN and SMDS

Such an arrangement avoids a direct one-to-one connection between all LANs of an organization. For instance, suppose an organization has four LANs numbered 1 to 4 in different parts of a city. If we have to have a connection from LAN 1 to LANs 2, 3, 4, and similar connections from LANs 2, 3 and 4 to each other, it would mean a lot of wires everywhere. Instead, SMDS uses the existing telephone company's switching infrastructure to route the traffic. It also means that a packet from one LAN to another could travel via many switches of telephone companies before reaching the destination.

SMDS uses a small header and allows for each data packet to contain up to 9188 bytes. It is a connectionless service wherein a host connected to a SMDS network can send a packet to any destination any time. Mandating that each packet must contain the destination address facilitates this.

9.10.3 SMDS Billing

In general, telephone company services are suitable for continuous traffic. If organizations use leased lines from telephone companies to connect their LANs, it might be wasteful since the LAN-to-LAN communication is likely to be in the form of short bursts. That is, there would be a lot of data for a short while, and then none at all for some time. Therefore, a leased line may not be cost-effective, since the billing would not be based on actual usage, and instead, would be a fixed charge regardless of its usage. Therefore, a packet-switching technology such as SMDS is more cost-effective for connecting LANs at different locations together. As we have mentioned before, the billing is done on the basis of the actual usage, in this case.

9.11 WIDE AREA NETWORK (WAN)

We have discussed Local Area Networks (LAN) and Metropolitan Area Networks (MAN). As mentioned earlier, a LAN can span a single building or campus. A MAN, on the other hand, can span a single city. Finally, a **Wide Area Network (WAN)** can span across cities, countries or even continents. The following question arises at this stage: since technologies are available that can extend LANs beyond their normal size to span across large distances with the help of satellites, would such a LAN not be considered as a WAN? The answer is negative. Even if a LAN is extended beyond a building or a campus, technological limitations prevent it from supporting a very large number of computers at very high speeds. On the other hand, a WAN can be scaled easily. That is, a WAN can support new sites with the same ease as it supports the existing sites.

For instance, suppose an organization whose base is in USA, decides to open an office in India. They can very well create a LAN between these two offices. However, if they open a third office in, say, Australia, the third office may not connect to the existing set-up with the same efficiency. In contrast, if WAN technology is used to first connect the offices in USA and India, and then connect the Australian office to that WAN, it can happen quite easily. Thus, *scalability* is the factor that distinguishes a WAN from other network types.

An obvious question now is: What is so special about a WAN that facilitates this scalability? It is the concept of *switching*. We have already discussed switching in detail. As we know, switching allows many computers to connect to many switches, rather than to each other. When computers are connected to each other to form a network, such architecture is called point-to-point architecture. Obviously as more and more computers are added to such a point-to-point set-up, it can become very complex. In many cases, there would be an upper limit on the number of computers that can be connected directly to each other. Instead, if more switches are added to which the additional computers can attach, the network size can be expanded arbitrarily. This is what happens in case of a WAN.

The basic hardware device in a WAN is called a packet switch. The name is due to the fact that this device moves (switches) packets from one connection to another. A packet switch is actually a computer that has its own processor, memory and I/O devices. All these peripherals are required for accepting and appropriately routing packets. Figure 9.22 shows the conceptual block diagram of a packet switch.

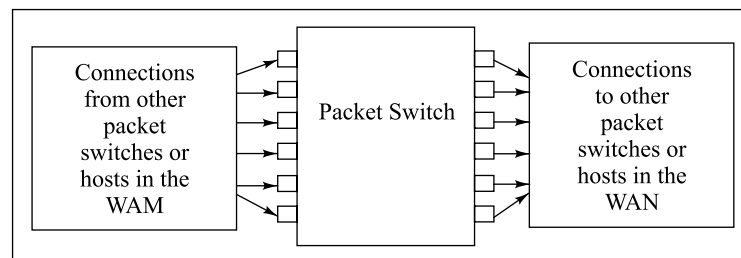


Fig. 9.22 Packet switch

9.12 WAN ARCHITECTURE

Put simply, when we connect a number of packet switches together, a WAN is formed. A packet switch has multiple I/O ports. Thus, it can connect to a many different computers or other switches to form a variety of topologies. Figure 9.23 depicts a WAN formed by four switches and 15 computers connected together as shown.

Figure 9.23 also signifies another important point – a WAN need not be a symmetric network. There can be any number of packet switches, and the packet switches connect to each other and the hosts. For instance, packet switch 1 connects to only three hosts, whereas packet switch 3 connects to 5 hosts. The packet switches are connected to each other using very high-speed lines, such as T1. The hosts local to a particular packet switch connect to that packet switch.

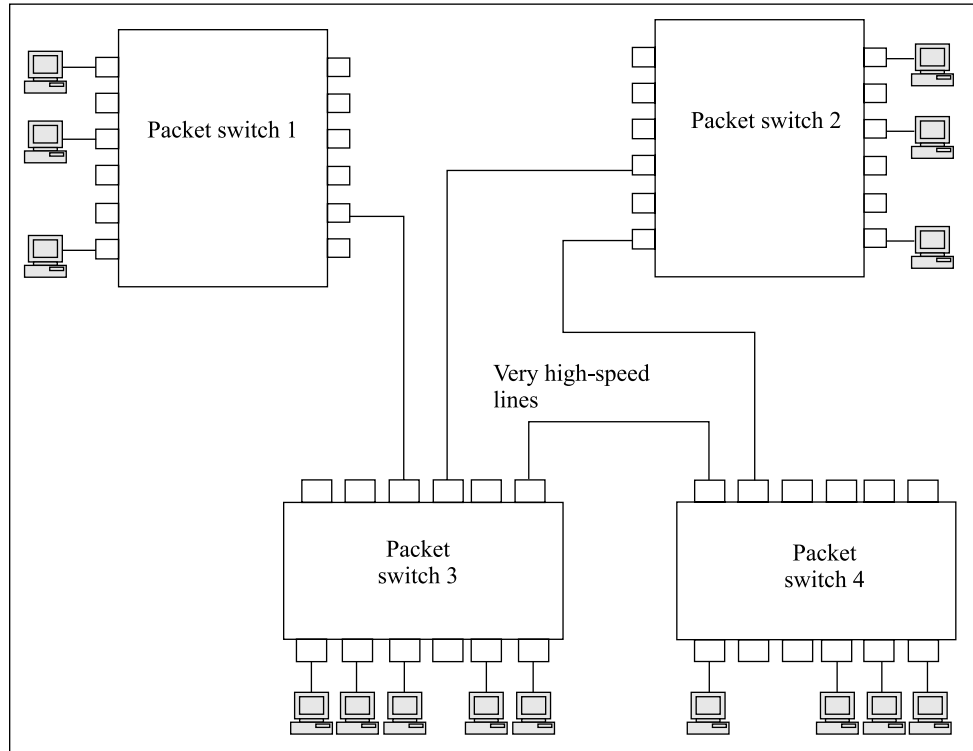


Fig. 9.23 Packet switches and computers forming a WAN

9.13 WAN TRANSMISSION MECHANISM

We know that at a time, only a pair of computers can communicate with each other in case of LANs. This is because all the computers in a LAN share a common transmission medium. However, in a WAN, the transmission medium is not shared, as we have seen. Therefore, any computer can transmit data at any time. To support this idea, WAN transmission mechanism is based on the packet *store-and-forward* concept. For this to be possible, a packet switch must have sufficient amount of buffer memory to store packets temporarily before they can be forwarded to the destination along with the chosen route.

From the viewpoint of a packet switch, the *store* operation occurs when a packet arrives from a host or another packet switch. When a packet arrives, the input device of the switch accepts the packet, stores it in the memory of the switch, and informs the processor of the switch by using the interrupt mechanism.

The processor of the switch examines the packet and decides where it needs to be forwarded. Accordingly, it performs the *forward* operation by invoking the services of the appropriate output device.

This store-and-forward philosophy allows a switch to store packets temporarily in its memory until the route over which they have to be forwarded is free. This also means that even if a route

is very busy, the switch can hold on to the packets until the route becomes free, so that the switch can transmit the held packets over that route.

In turn, this is why we said that any host could send any data at any time in case of a WAN, unlike what happens in case of a LAN. As we have discussed before, in case of a LAN, because all the hosts share the transmission medium, an access mechanism needs to be in place (such as CSMA/CD or token passing), that allows only one host to transmit at any point of time. Such Media Access Control (MAC) is not necessary in WANs, as the medium is not shared, and therefore, a number of messages could be traveling across the WAN to/from different switches/hosts at any time.

9.14 WAN ADDRESSING

To allow efficient routing of packets, the addressing mechanism in the case of WANs is slightly different from that used in LANs. As in LANs, every host in a WAN has a physical address. However, there is one difference. WAN addressing is hierarchical. The address of a host on a WAN is composed of two parts, viz., the first part identifies the switch to which the host connects (**switch number**), and the second part uniquely identifies a host attached to that switch (**host number**). For instance, if the address of a host is (1, 5), it means that the host is connected to switch number 1, and on that switch, it is host number 5. This is shown in Fig. 9.24.

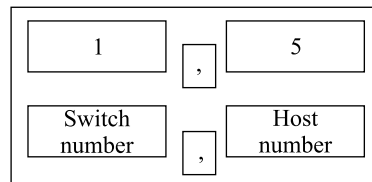


Fig. 9.24 Hierarchical addressing in WAN, consisting of switch number and host number

Taking this concept a bit further, Fig. 9.25 shows the hierarchical addressing scheme on a portion of a WAN. This portion of a WAN consists of two switches, each having two hosts attached to it.

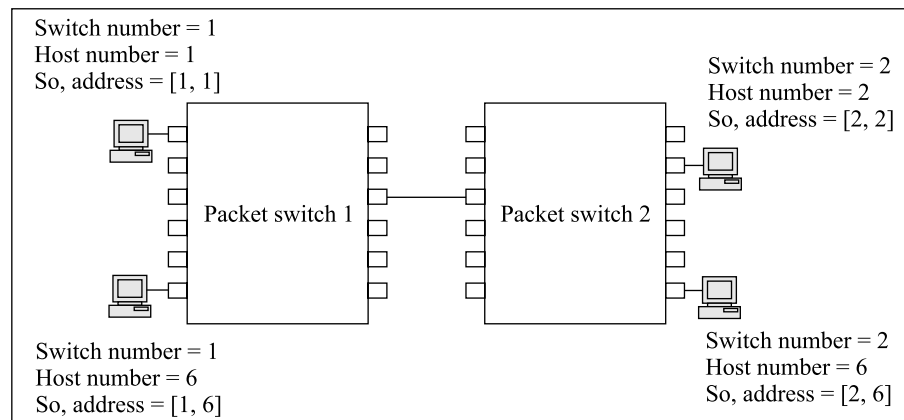


Fig. 9.25 Hierarchical addressing example in a WAN

Note how the switch number remains the same for all the hosts on the same switch, whereas the host number changes. The host number is unique only within a switch. Therefore, although we have a host with host number as 6 on both switches (numbered 1 and 2), the overall address consisting of switch number plus host number makes the complete address unique, namely [1,6] and [2,6]. Thus, any host can be uniquely identified in a WAN easily.

Although from an external perspective, the address of a host is represented as a combination of switch number and host number, internally the address is stored as a series of bits. The applications are not even aware of the external and hierarchical nature of this addressing.

9.15 PACKET FORWARDING

9.15.1 Introduction

How does a switch decide where and how to forward packets so that they reach the desired destination? For this, a switch must select an outgoing path over which the packet must be forwarded. It examines the destination address field of the packet for this purpose. Now, there are the following two possibilities.

1. If the *switch number* in the destination address field of the packet is the same as its own number, the switch knows that the destination host is directly attached to it, and therefore, simply forwards the packet to the appropriate host, based on the *host number* value.
2. If the *switch number* in the destination address field of the packet is not the same as its own number, the switch forwards it to another switch over its high-speed connection with that switch. For this, it uses the concept similar to that of a routing table. We shall discuss this concept in the following sections discussing WANs.

9.15.2 Next-hop Table

To decide where (i.e., to which switch) to forward the packet next, each switch on the WAN maintains a table called the **next-hop table**. Clearly, with the existence of so many hosts in the WAN, a switch does not keep information about the complete path required to reach every possible host on the WAN. Instead, it just maintains an entry about the next place (called *next hop*) a packet should take to eventually reach the destination.

To illustrate an idea, let us take a simple example. Suppose a passenger is traveling (a) from Delhi to Mumbai, (b) from Mumbai to London, (c) from London to New York, and finally (d) from New York to Atlanta. In her/his entire journey, the final destination remains constant, i.e., Atlanta. However, in each case, the *next hop* changes. Firstly, when going from Delhi to Mumbai, the *next hop* is Mumbai. When the passenger takes the Mumbai to London flight, the *next hop* is London. When the passenger takes the London to New York flight, the *next hop* is now New York. Finally, when the customer takes the flight from New York to Atlanta, the *next hop* is now Atlanta. Thus, the final destination remains the same throughout. However, the *next hop* keeps changing at every stage.

The same concept is used in the case of the packet forwarding mechanism, which uses the next-hop table. Figure 9.26 (a) shows a sample WAN structure where we have three switches numbered 1, 2 and 3. Each switch has two hosts connected to it. Figure 9.26 (b) shows the next-hop table for switch 2, which is explained later.

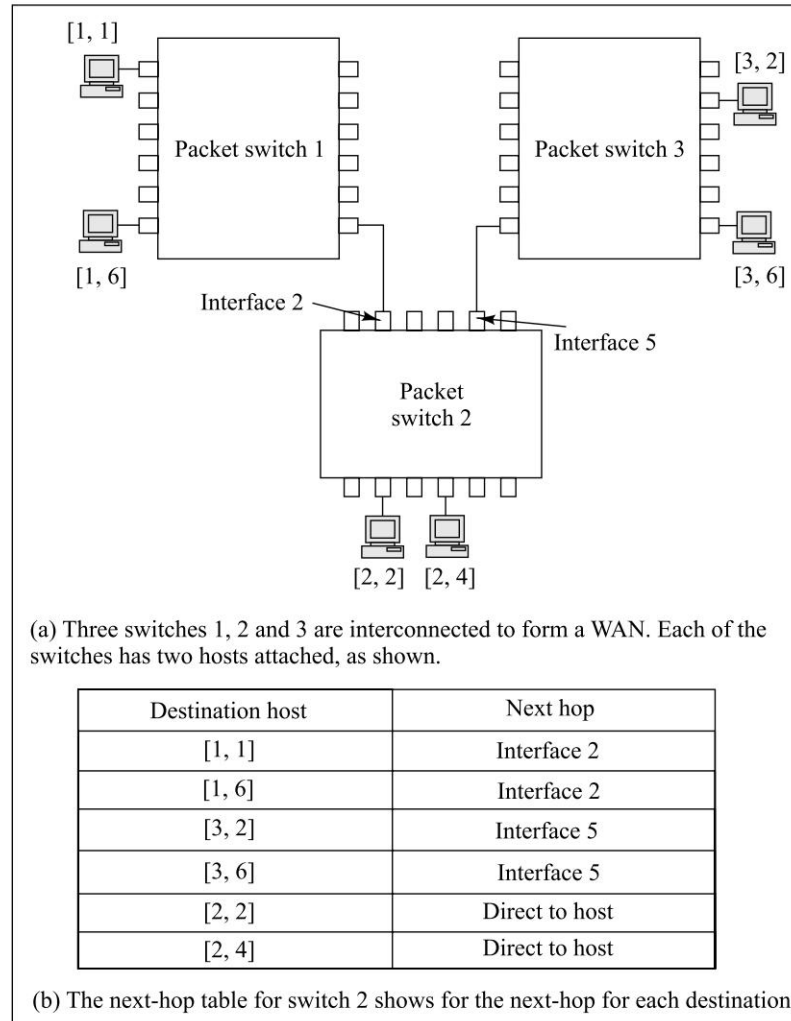


Fig. 9.26 Sample WAN and next-hop table

As shown in Fig. 9.26 (b), the next-hop table for switch 2, each entry in the table consists of two columns, viz., destination and the next hop required to move one step closer to that destination. For instance, the table indicates that to reach host with address [1, 6], the next hop that must be taken is *Interface 2*. The term *interface* signifies one of the outward paths on that switch, and has no meaning outside the context of a switch.

When forwarding a packet, the switch finds out the destination address of the packet from the packet header, looks for a corresponding entry for that destination in the *next-hop table*, and uses that entry to forward the packet to the next hop specified there. Therefore, if a packet arrives at switch 2 with the destination address as [3, 2], the switch 2 searches its *next-hop table*, locates an entry for destination host as [3, 2], and realizes that the packet should be forwarded to interface 5 (refer to Fig. 9.26 for reference). In contrast, if switch 2 receives a packet with the destination address

as [2, 4], switch 2 realizes that the host is attached to it directly. Therefore, it simply forwards it to host [2, 4]; rather than forwarding it to either interface 2 or 5. This is how packets get forwarded from one switch to another, until they reach their final destination.

9.16 NEXT-HOP TABLES AND ROUTING

You would now realize that the philosophy of *next-hop table* is actually familiar to us. We have already studied it when we studied the routing algorithms. *Next-hop tables* are more commonly known as *routing tables*. The process of using routing tables for forwarding packets to the appropriate destination one hop at a time is called *routing*.

It can be observed that the two-part hierarchical addressing consisting of the switch number and the host number is very useful in another way. The routing tables do not need to have the complete address specified for every host on the WAN. This is because all the hosts on the same switch are going to follow the same route. For instance, from the perspective of switch 2, two different packets having the destination address as [1, 1] and [1, 6] respectively must be given to interface 2 for the next hop. It does not matter that their host numbers are different. What matters is the fact that the destination switch number (1) is the same in both cases. Therefore, it is wasteful to maintain an entry for every destination host on every switch in a routing table. Instead, the routing table can be shortened to simply have the destination switch number and the corresponding next hop entry, as shown in Fig. 9.27.

Destination switch	Next hop
1	Interface 2
3	Interface 5
2	Direct to host

Fig. 9.27 Shortened routing table

When we remember that a WAN can consist of many switches, each having to maintain such a routing table, and using it for searching the next hop for every incoming packet, we will realize that shortening the routing table is a very significant improvement. Firstly, the switches have to maintain smaller tables, and secondly, because the number of entries is now much less, the searching is also faster at the time of actually forwarding a packet to the next hop.

When there are many switches, normally there are multiple routes possible to reach the same destination from the same source. For instance, in Fig. 9.23, you can reach switch 4 from switch 1 via switch 2 or switch 3. Depending upon the bandwidth of both these connections and congestion levels at these switches, a *route* is chosen. If switch 2 is chosen as an intermediate switch, then it will be mentioned as the *next hop* for a packet to be sent to switch 4 from switch 1. Therefore, this value will be stored in the routing table of switch 1. As the number of switches increases, routing becomes more complex. We already have discussed the routing algorithms, and therefore, need not be repeated here.

X.25 is a very important packet switching protocol used in WANs. It covers the first three layers of OSI, viz., physical, data link and network. The function of X.25 is to deliver a packet from any node to any other node connected to the X.25 network, so long as the packet size and format conform to the standard. Thus, it does routing, moves packets between adjacent nodes, and controls error and

flow, etc., to make sure that the packet reaches the destination. These days, **Frame Relay** technique is becoming more popular than X.25. We will talk about both in subsequent chapters.

9.17 PURE AND SLOTTED ALOHA

ALOHA was devised in the 1970s by Norman Abramson and his colleagues at the University of Hawaii. ALOHA is a channel allocation protocol. That is, it is used to determine how access should be granted to a medium shared by many hosts. There are two types of schemes in ALOHA, viz., **pure ALOHA** and **slotted ALOHA**.

Pure ALOHA

The basic idea of ALOHA is pretty straightforward. It allows users to transmit whenever they have data to be sent. Like CSMA/CD, collisions can and do happen in such a scheme. However, there is a feedback mechanism, which allows the sender to know if the sent frame was transmitted successfully, or whether it was destroyed because of a collision. Like CSMA/CD, in the case of a collision, the sender waits for a random time before retransmitting the frame. The randomness in the waiting time minimizes the scope for yet another collision. However, another collision cannot be completely ruled out. So, if another collision occurs, the sender waits for some more random time, and makes another attempt at data transmission. This can continue until the sender is successful in the transmission.

Each user on an ALOHA system is in one of the two possible modes: waiting or typing. Initially, suppose that all the users are in the typing state. When a line is typed, the user stops typing, and transmits the line. Now, the user has moved to the waiting state. Depending on whether the user's line was transmitted successfully (i.e., a collision (did not) occur), the user remains in the waiting state, or moves back to the typing state. This simple scheme of ALOHA is called pure ALOHA.

Slotted ALOHA

In 1972, Roberts published an improvement to the ALOHA standard, which is called slotted ALOHA. In this scheme, the capacity of an ALOHA system is effectively doubled. Here, the overall time is divided into several time slices or intervals. Each interval corresponds to one frame. Thus, the clock is tied to frame transmission. This means that a user cannot transmit data arbitrarily now. The user can transmit only when the interval for the next frame starts. This approach mandates that all the users agree with the slot boundaries.

The result of this change in the scheme of things is that we do not have to worry about transmissions that take place within the time interval needed for the transmission of one frame. We do not need to think about two consecutive frames. This is because collisions can now occur only during a particular time slot, but not across two different time slots.

Slotted ALOHA is a synchronous system, which divides the available time into various slots. The slot size is equal to the time it takes for one packet to be transmitted. This means that whenever a station is ready with a frame, it must wait until the next slot is available for transmission. It cannot start transmissions arbitrarily.

SUMMARY

A Local Area Network (LAN) usually serves the purpose of a small area, such as an office or a university to link all its computers. LANs have been extremely popular for many years for connecting computers and sharing resources over limited distances. LANs can be static, wherein each host on the LAN gets a fixed time slot to transmit. However, the newer approach of dynamic LANs is more popular as it allows any host to transmit at any time.

Because LANs use shared medium, a question arises as to which host should send data and when? This is solved democratically by the *Media Access and Control (MAC)* protocol, which is somewhat in between the physical and data link layers.

Ethernet, Token Ring and FDDI are the three most popular types of LANs. Token Ring and FDDI are pretty similar to each other using token passing method for MAC. Ethernet uses CSMA/CD for MAC.

A transceiver is used to establish the connection between a computer and the Ethernet bus. The transceiver connects to a Network Interface Card (NIC), which is on the motherboard of the host computer. The transceiver is responsible for sensing when the Ethernet network is available for transmission. The NIC is a small computer that relieves the CPU of the host from data transmission issues.

Ethernet employs the mechanism of CSMA/CD for data transmission. Here, any host can transmit at any time when its transceiver senses that the transmission medium is free. However, if two hosts transmit at the same time, a collision occurs. After this, the colliding hosts wait for random times. These random times are determined by the binary exponential back-off policy, so that the chances of subsequent collisions are reduced.

A Token Ring network does not allow random transmissions. Instead, before transmission, a host must wait for a special frame called *token frame*. Only when a host receives a token frame that it can transmit data. For this, the host captures the token, transmits the data frame and waits until the data frame travels over the entire Token Ring and comes back, at which point it releases the token frame. Now, another host can capture the token frame and use it for its data transmission. In any case, each host on the Token Ring receives the data frame. All but the one for which it is destined, simply ignore the data frame. The interested host makes a copy of it.

FDDI is a special case of Token Ring. The main difference between Token Ring and FDDI is that FDDI employs a dual ring, rather than a single ring. This is useful when a host in the network goes down, or when the primary ring fails. In such an event, the secondary ring can be used for data transmission over the FDDI network. Thus, it serves the purpose of a transmission backup.

Metropolitan Area Networks (MANs) are mainly useful for connecting hosts in the same city. Since a LAN has physical limitations, a MAN can be useful for creating bigger networks.

Distributed Queue Dual Bus (DQDB) is an example of MAN. It uses the mechanism of a dual queue. There are two buses connecting all the computers on a DQDB network. Each bus allows traffic in a single direction only. To transmit data, the sending host must select one of the two buses. A host reserves the slot before transmitting its data. At any point of time, every host knows how many reservations are pending to be served.

Switched Multimegabit Data Services (SMDS) is a high-speed MAN technology. SMDS is aimed at fulfilling the needs of the organizations that have a number of LANs spread across different locations in a city. Many such LANs are connected together to form a SMDS MAN. It is a connectionless service wherein a host connected to a SMDS network can send a packet to any destination any time. A packet-switching technology such as SMDS is more cost-effective for connecting LANs at different locations together.

A Wide Area Network (WAN) is the biggest of LAN, MAN and WAN. It can span across cities, countries and continents. The main difference between a WAN and a LAN/MAN is that a WAN is easily scalable. It achieves this with the help of the concept of switching. WAN allows different computers at a location to connect to a switch. Many such switches connect to each other to form a WAN. As the existing switches become insufficient, new switches can be added easily.

Each host on a WAN has a two-part address. The first part of the address identifies the switch number, and the second part identifies the host number on that switch. Depending on the switch number portion of the address in a packet, it is determined whether the packet is to be delivered locally or not. For this, each switch also maintains a next-hop table. This table maintains a next-hop entry for each host on the WAN. Using the table, a switch decides where to forward an incoming packet.

Maintaining a list of switch numbers and their corresponding next-hop entries, rather than maintaining one row for each host can further optimize the next-hop table. Since in a WAN, there can be a number of switches connecting many hosts over long distances, such an optimization is very useful.

ALOHA is a channel allocation protocol. There are two types of schemes in ALOHA, viz., pure ALOHA and slotted ALOHA.

KEY TERMS AND CONCEPTS

ALOHA	Fiber Distributed Data Interface (FDDI)
Binary exponential back-off policy	First In First Out (FIFO)
Carrier Sense Multiple Access with Collision Detect (CSMA/CD)	Frame data
Centralized dynamic LAN	Frame relay
Collision	Frame Status (FS)
CRC	Frame type
Decentralized dynamic LAN	Hardware address
Destination address	Host number
Distributed queue	Local Area Network (LAN)
Distributed Queue Dual Bus (DQDB)	Loopback
Downstream	Media Access Control (MAC)
Dual bus	Metropolitan Area Network (MAN)
Dynamic LAN	Network Interface Card (NIC)
End Delimiter (ED)	Next-hop table
Ethernet	Physical address
Ethernet address	Preamble
Fast Ethernet	Pure ALOHA
	Self healing mechanism

Slotted ALOHA	Token Ring
Source address	Transceiver
Static LAN	Transmission slot
Switch number	Unidirectional traffic
Switched Multimegabit Data Services (SMDS)	Upstream
Token bus	Wide Area Network (WAN)
Token frame	X.25

QUESTIONS

True/False

1. A LAN is usually within the boundary of a private building.
2. Internet is a LAN.
3. Ethernet uses a single coaxial cable as the transport medium.
4. Ethernet is not a broadcast network.
5. It is simply not possible in an Ethernet network that two hosts transmit frames at the same time.
6. An Ethernet address consists of 40 bits.
7. An Ethernet address is always unique.
8. In case of Token Ring, the hosts are arranged to form a circular ring.
9. The self-healing mechanism of the FDDI network is made possible by the second ring.
10. Ethernet has an error control mechanism.
11. A Metropolitan Area Network (MAN) is a network that is designed to cover an entire city.
12. The Distributed Queue Dual Bus (DQDB) protocol is a single bus configuration.
13. Each bus on DQDB supports traffic in only one direction.
14. In DQDB, if the destination host is the end host, the *read* flag is not set in the slot.
15. The reservations made by all hosts are stored in a First In First Out (FIFO) queue structure in DQDB.
16. SMDS is a packet-switched datagram service.
17. The connection in SMDS is based on the dual bus technology.
18. A Wide Area Network (WAN) can span across cities, countries or continents.
19. Scalability makes WAN very important.
20. Switching is trivial in WANs.
21. WAN uses two-part addressing.
22. Internally, a host address on the WAN is stored as a series of bits.
23. There cannot be any optimization in the next-hop table.

Multiple-Choice Questions

1. A _____ spans the largest distance among the category of computer networks.
 - (a) LAN
 - (b) MAN
 - (c) WAN
 - (d) Ethernet
2. In _____, the transmission media is shared between all the computers on a network.
 - (a) LAN
 - (b) MAN
 - (c) WAN
 - (d) X.25


3. A transceiver connects a _____ to _____.
 - (a) computer, computer
 - (b) network, network
 - (c) computer, Ethernet
 - (d) Ethernet, Ethernet
4. The _____ performs all network-related work on a computer connected to a network.
 - (a) transceiver
 - (b) CPU
 - (c) hard disk
 - (d) NIC
5. _____ helps Ethernet recover from simultaneous transmissions causing errors.
 - (a) CSMA/CD
 - (b) CSMA
 - (c) CD
 - (d) Collision
6. An Ethernet address can be _____.
 - (a) unique
 - (b) duplicated
 - (c) optional
 - (d) never duplicated
7. In Token Ring, a special packet containing a _____ goes around the network.
 - (a) data
 - (b) header
 - (c) token
 - (d) bit
8. To protect against fiber cut, FDDI has _____ sets of fibers.
 - (a) 0
 - (b) 1
 - (c) 2
 - (d) 3
9. Each bus in DQDB allows traffic in _____ direction(s).
 - (a) 0
 - (b) 1
 - (c) 2
 - (d) 3
10. In DQDB, the reservations made by all hosts are stored in a _____ queue structure.
 - (a) LIFO
 - (b) LILO
 - (c) FIFO
 - (d) FILO
11. Subscribers of _____ pay only for the time that they use its services.
 - (a) SMDS
 - (b) DQDB
 - (c) Frame Relay
 - (d) Token Ring
12. The basic hardware device in a WAN is called a _____.
 - (a) circuit
 - (b) circuit switch
 - (c) packet switch
 - (d) ring
13. In a _____, the transmission medium is not shared.
 - (a) LAN
 - (b) MAN
 - (c) WAN
 - (d) None of the above
14. WAN addressing consists of a _____ and a _____.
 - (a) switch number, switch number
 - (b) switch number, host number
 - (c) host number, switch number
 - (d) host number, host number
15. Each switch on a WAN maintains a _____ for forwarding packets.
 - (a) array
 - (b) hash table
 - (c) route
 - (d) next-hop table

Detailed Questions

1. What is the difference between static and dynamic LANs? Why are static LANs not as popular as dynamic LANs?
2. Describe the purpose of a transceiver and a Network Interface Card.
3. Discuss the main properties of an Ethernet network.

4. What is CSMA/CD? How does it work?
5. Why is binary exponential back-off policy important for Ethernet?
6. Describe the main fields in an Ethernet frame header.
7. How does a Token Ring network work? In what ways is it different from Ethernet?
8. Describe the main fields in a Token Ring frame header.
9. Why FDDI is called a 'self-healing' type of network?
10. Discuss how DQDB is different from an Ethernet or a token ring network.
11. What are *upstream* and *downstream*? What is their significance to DQDB?
12. Discuss how a host transmits data and how it travels over a DQDB network.
13. Explain how the concept of slot reservation works in DQDB.
14. Describe how SMDS avoids direct LAN-to-LAN connections.
15. How is billing done in case of SMDS?
16. What makes a WAN different from a LAN and a MAN?
17. How does a WAN achieve scalability?
18. Discuss a typical WAN architecture.
19. How does store-and-forward transmission work in a WAN?
20. Discuss WAN addressing mechanism. How is it useful for data transmission?
21. How does a switch decide which packet is local to it and which is not?
22. Explain the concept of the next-hop table used in WANs. How is it further improved?
23. What is ALOHA? Explain its types.

10 Medium Access Sublayer and ISDN



10.0 INTRODUCTION

The traditional analog telephone network is based on circuit switching and uses twisted-pair wires for voice transmissions. The same telephone network is also used for data transmission. Of course, due to the inherent limitations of the analog telephone network, transmitting computer data over telephone lines is not the best proposition. However, given the amount of twisted-pairs all over the world, and the kinds of investments companies have made in the telephone system infrastructure, it is unthinkable of suddenly replacing the telephone system with something modern, such as optical fiber.

To meet the challenges of reusing the existing twisted-pair wires, and yet supporting voice and data transmissions, as well as modern applications such as fire alarms through a telecommunications network, work began towards an integrated network that could cater to all these requirements. Most significantly, this had to be built on top of the existing infrastructure.

The **Integrated Services Digital Network (ISDN)** is an attempt in this direction. It must, however, be noted that doubts have surfaced recently regarding its usefulness, as it offers limited bandwidth (which is definitely higher than the current data rates, but less than what newer applications demand). Considering this, Internet browsing has become the focus area of ISDN. Comments as ISDN stands for *It Still Does Nothing*, etc., have been made by people on a lighter note.

10.1 STATIC AND DYNAMIC CHANNEL ALLOCATION

The term **channel allocation** is quite significant in any data transmission system. The term channel is an alternate name for the transmission medium. Hence, when we speak about channel allocation, we refer to the allocation of the capacity of the transmission medium (wired or wireless) to the computers that want to transmit data. For instance, in LAN, the problem of channel allocation is restricted to the bandwidth of the LAN cable. We need to decide how we want to allocate the capacity of the LAN cable to all the competing hosts. Similarly, this concept can be applied to any other network, including wireless networks. The basic problems remain the same. What range of the spectrum is going to be made available to the various transmitting hosts, how long would they be allowed to transmit, would they need to wait for someone else to free the medium, would this transmission be in a strict sequence of hosts, can more than two hosts transmit at the same time, and so on.

The aim of any channel allocation system is to achieve maximum throughput, i.e., reduce in efficiencies, but at the same time ensuring that no interferences between competing hosts occurs. In other words, the transmission of two or more hosts should not get garbled, and we should not end up getting mixed and non-understandable signals. At the same time, we also want the overall output to be as high as possible. In other words, we want the maximum utilization of the available bandwidth spectrum. These two basic principles drive the concept of channel allocation in most situations.

We can broadly divide the channel allocation schemes into the following three categories:

1. **Static channel allocation** – The **static channel allocation** scheme, also called **fixed channel allocation** scheme, is quite simple. The available bandwidth is permanently divided into predetermined fixed channels. Every channel has a predesignated bandwidth. The amount of the bandwidth and the number of channels both remain unchanged right throughout. Hence, this is a static allocation mode. The obvious advantage of such a scheme is its simplicity. Because the allocation of channels is fixed, there is no overhead in terms of observing and handling allocation changes that are dynamic in nature. Allocation of channels is done only once, right at the beginning, and this allocation remains unchanged subsequently. Of course, while this leads to more predictability and also reduction in maintenance overheads, it also causes problems of low throughput. Bandwidth is wasted many times because of imperfect allocations done in the beginning. However, even when we realize that some of the allocations are wrong and perhaps are going to cause an underutilization of the overall bandwidth, little can be done to solve this problem. In other words, the wastage in this kind of scheme is high. This type of channel allocation is not very efficient.
2. **Dynamic channel allocation** – In the **dynamic channel allocation** scheme, we do not depend on the initial one-time allocation of bandwidth to all the hosts. In other words, while the initial allocation of channels is done, that does not remain static. The allocation is subject to change. Depending on the transmission needs of the various stations, the available bandwidth is allocated to those hosts who need it. As a result, the overall system throughput in this case is always likely to be better. Hosts that transmit more are going to get more opportunities to send traffic, while hosts that do not intend to transmit are going to get lesser transmission slots. Of course, this also means that this allocation process has to be managed dynamically in a very efficient manner for true benefits to be realized. As a result, while this process increases the overall throughput, it also leads to more administrative overheads. The way this scheme works is that all the channels are marked *available* at the beginning. As soon as a host requests for a channel, the algorithm decides whether to make a particular channel available to that host. If this is indeed the case, that channel is now marked as *used* and it is not allocated to any other host that is also requesting for bandwidth. Once the initial host frees the channel, it is again marked as *available* and then it can be allocated to any of the requesting hosts. Hence, we can also consider that the central authority maintains a pool of *available* and *used* channels. Depending on the status, a channel belongs to one of these pools. Also, how to decide when to allocate a channel to a host is a tricky question, which may not have a definitive answer. It depends on the cost, priority, time duration that a channel is required for, and so on. So, extremely complicated algorithms may have to be used to deal with this situation.
3. **Hybrid channel allocation** – In the **hybrid channel allocation** scheme, we combine the features of the static and dynamic channel allocation modes. In other words, we can keep a

part of the spectrum under the control of the static channel allocation scheme, and the rest under the control of the dynamic channel allocation scheme.

We can compare the static and dynamic channel allocation schemes as shown in Table 10.1.

Table 10.1 *Differences between static and dynamic channel allocation*

Point of difference	Static channel allocation	Dynamic channel allocation
Performance	Better under heavy traffic	Better under low/moderate traffic
Flexibility in channel allocation	Nil	High
Suitability	Suitable for large networks	Suitable for small/medium sized networks
Overall flexibility	Low	High
Suggested application	Long-duration voice calls	Voice calls of short duration, Data transmission
Management overheads	Low	High
Call/Transmission set-up delay	Low	High
Signaling load	Low	High
Control	Centralized	Centralized or distributed

10.2 MEDIUM ACCESS CONTROL (MAC) SUBLAYER

In the OSI model, the problem of channel allocation is handled by the **Medium Access Control (MAC)** sublayer. This layer is considered as a part of the data link layer, and is considered to be located below the Logical Link Control (LLC) sublayer. Ethernet and many other LAN technologies make extensive use of MAC. In general, the MAC layer is connectionless and unreliable. In other words, it provides support for independent frames to be delivered from a source to a destination without creating any connection between them beforehand. Of course, sometimes, the LLC layer undergoes certain changes so that the MAC layer provides reliable service instead of a connectionless and unreliable service.

We can depict the MAC sublayer as shown in Fig. 10.1.

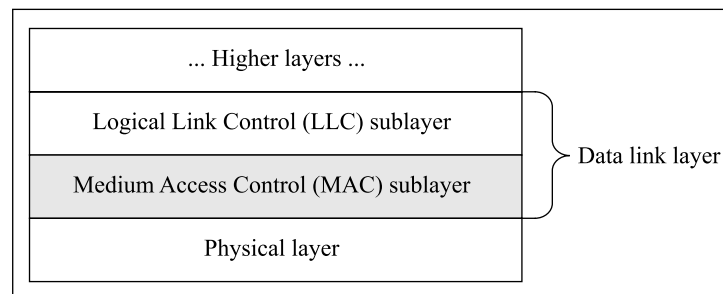


Fig. 10.1 *Medium Access Control (MAC) sublayer*

The MAC sublayer takes on the responsibilities of addressing and channel access control mechanisms. In other words, the MAC sublayer provides a **physical address** to every host on the network. This is also why many times the physical address is called **MAC address**. This physical address is static. It is a unique serial number hard coded onto the Network Interface Card (NIC) of a computer. For example, we know that in Ethernet networks, this is a 48-bit address.

The MAC sublayer also performs the functions of allocating channels to hosts whenever they want to transmit and claiming them back when the transmission needs are fulfilled. Several strategies can be used for this purpose. For instance, in Ethernet networks, the CSMA/CD mechanism is used, whereby hosts are allowed to transmit as and when they want to. If collisions happen, a recovery is attempted by using the exponential back-off algorithm. In other types of networks, different strategies for network access and channel allocation may be followed, depending on what works best in a given situation.

10.3 MAC IN LAN AND WAN

The MAC sublayer serves the same purposes in a LAN and a WAN. That is, this sublayer performs its expected functions of providing a physical address (MAC address) to every host on the network and it also takes up the responsibility of channel allocation. However, due to the basic differences between a LAN and a WAN, the way this works in these different networks and the challenges we encounter are quite different. In the case of a LAN, all the computers are in close proximity (e.g., inside a building). Hence, the transmission medium in the case of a LAN is generally shared. In other words, it is a bus network. However, in the case of a WAN, there is no question of all the hosts sharing the same transmission medium. This is because the hosts are distributed across the world, possibly in different countries. As a result, it cannot be a bus network. It is a distributed network.

The differences between a LAN and a WAN lead us to a situation where the addressing as well as channel allocation mechanisms are greatly different. For instance, because all the hosts on a LAN are clearly identifiable, the medium access mechanism can assume that all the hosts are directly attached to the transmission medium. Therefore, while designing the medium access control mechanism, we can presume that if any host transmits, all the other hosts would come to know about that transmission immediately. This is because all the hosts would have been attached to the same, single transmission medium, and therefore, they would have come to know about the transmission from another host. In other words, we can assume here that all the hosts can sense the physical medium directly. This is why Ethernet, which is a LAN-based protocol for medium access, uses the CSMA/CD mechanism. Here, every host senses the transmission medium first, and only if it finds the medium free, it transmits. Also, if it detects a collision, it stops. However, in the case of a WAN, clearly this mechanism is out of question. Hosts are separated from each other by large distances. Hence, there is no question of a common shared medium. Naturally, there is no concept of sharing a medium, sensing whether it is free, and then transmitting. Instead, all the transmissions must happen without having to sense the medium.

Consequently, there are significant differences between the MAC sublayers of a LAN and a WAN. The addressing in a LAN is simple, with a physical address being assigned to every host. However, because a WAN is made up of a network of several LANs where several hosts are on an individual LAN, the situation is different. We need a two-part logical address for every host on a WAN. This two-part logical address consists of a network number and a host number. These are separated with a colon. Hence, a WAN logical address looks like A:B, where A is the network part

and B is the host part. Of course, every host also has a physical address, which is used within the LAN for final delivery of frames. But from the point of view of the WAN, the physical address is immaterial. WAN makes use of the two-part logical address.

10.4 CLASSIFICATION AND STUDY OF MAC SUBLAYER PROTOCOLS/ COLLISIONS

Based on the conflicting requirements of the MAC sublayer in LAN and WAN, we can classify the MAC sublayer protocol into various categories. Let us discuss some of the prominent ones:

1. **Collision versus collision-free MAC protocols** – If we have a bus network like Ethernet, all the hosts share the bus. Because the transmission medium is shared, theoretically all hosts can transmit at the same time. This keeps happening in Ethernet. However, to reduce the chances of this kind of conflict, every host first checks the transmission medium (bus) before transmitting. Regardless, this can lead to collisions since two or more hosts can possibly find the transmission medium free and they can transmit at the same time. Hence, to handle these kinds of problems, we must have mechanisms built into the algorithm itself. Hence, we have the CSMA/CD mechanism to deal with this situation. However, in Token Ring networks, the possibility of collisions do not exist, since without obtaining a token, a host is not allowed to transmit. As a result, while Ethernet is a collision network requiring the possibility of handling collisions by way of CSMA/CD, the Token Ring network does not need this feature. Hence, it is a collision-free network.
2. **Collision-aware versus Collision-unaware networks** – This possibility exists when we compare a wired network such as Ethernet with a wireless network such as 802.11 (Wi-Fi). This is because in a wired network such as Ethernet, there are clear chances of collisions. Hence, we must be prepared to handle them. However, in the case of wireless networks such as 802.11, it is very difficult to even know about collisions. This is because hosts cannot detect the presence of other hosts in their vicinity very easily. Moreover, hosts can possibly move abruptly in and out. As a result, hosts in a wireless network would find it very difficult to handle and overcome collisions. Hence, such networks need to be so created that collisions are avoided in the first place.
3. **Fixed-address versus Variable-address networks** – In the case of wired networks, we generally have computers whose logical and physical addresses do not change that frequently. However, in the case of wireless networks or even wireless hosts in a wired network, hosts move from one location to the other along with their users (e.g., users take their laptops from one desk to another). In such cases, the MAC protocol has to be ready to support this transition abruptly. Hence, appropriate address resolution mechanisms are required to be in place for this purpose.
4. **Contention versus contention-free networks** – There are various situations where regulating the traffic is useful. For example, in a cellular network, time slots are allocated to users, so that the effective bandwidth is utilized in an effective manner. There are also some situations where such a regulation puts artificial controls on the overall transmission, and the usage of the overall bandwidth is not effective. To contrast between such conflicting requirements, the MAC sublayer needs to make appropriate adjustments. For example, in the pure ALOHA protocol, no provisions are kept for acquiring bandwidth before transmitting. Any host can transmit at any time. However, in the case of slotted ALOHA, time slots are created, and hosts are allowed to transmit only at the beginning of the slot. Such decisions need to be

taken depending on the situation. Sometimes, we may want to use a hybrid solution to cater to various needs.

10.5 ISDN AND ITS BACKGROUND

10.5.1 Analog Communication

The traditional telecommunications network was designed to carry voice traffic. Since human voice needs a small amount of bandwidth (usually 4 KHz at the most), the telecommunications network consisted of copper wires that were analog in nature and could carry voice traffic quite easily. It was also the reason why the bandwidth of copper wires was restricted to 4 KHz. However, as computer-to-computer data communications became more common, people started realizing that there was a big mismatch between what was *required* (facilities to carry digital data) and what was *available* (wires that could carry analog signals at a maximum frequency of 4 KHz). Since the telecommunication network existed everywhere, there was no choice for the technologists but to reuse the analog telephone lines for data transmission as well. As we know, modems are used to modulate the digital data arriving from the computer at the sender's end into analog signals, which are then carried over by the traditional analog telephone network (starting with the telephone company's Central Office or CO) to the receiver's end, where another modem performs a demodulation and hands over the digital data to the receiving computer. This is shown in Fig. 10.2.

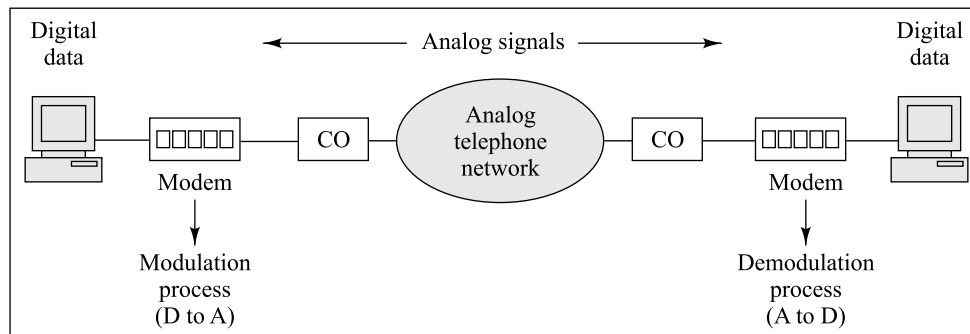


Fig. 10.2 Analog telephone network carrying digital data

To increase data rates over the traditional copper wires, techniques such as Quadrature Amplitude Modulation (QAM) were developed, which would alter more than one properties of the analog signal (such as amplitude and phase) to encode multiple bits per signal change. However, this would still be within the basic limitation of 4 KHz. Since the newer technologies need not obey this restriction of 4 KHz (because they do not use dial up modems for modulation, but instead, have their own modern modulation techniques), these technologies can utilize the bandwidth of the copper wires beyond the 4 KHz limit, and provide higher data rates. This is also why ISDN can provide higher data rates over the existing copper wires.

Of course, the analog transmission suits voice traffic, which is analog in nature, anyway. Therefore, the analog signals coming from a telephone do not require any kind of transformation before they can travel across a telephone network. In contrast, computer data is first modulated by a modem. As a result, the state of the voice/data connection between a user and the telephone network looked as shown in Fig. 10.3 for a number of years.

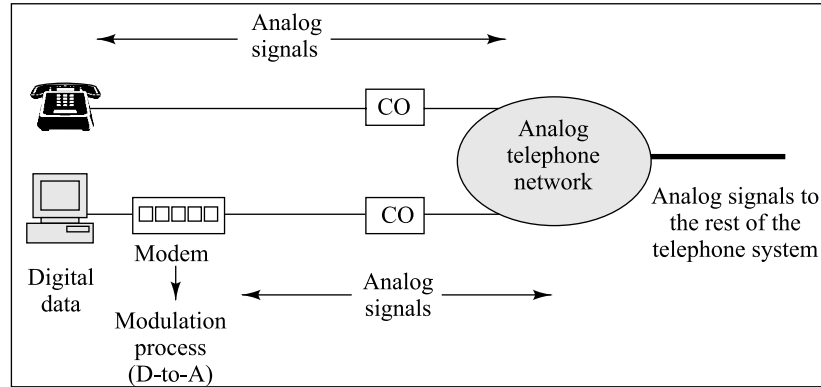


Fig. 10.3 Voice and data communications over analog telephone network

One of the primary aims of ISDN is to have a single wire coming out from the telephone company's CO to these devices.

10.5.2 Digital Communication

With technological progress in digital communications, telephone companies started providing digital communication backbones. This means that the communication between the Central Office (CO) and the rest of the telephone network became digital. To achieve this, the incoming analog signals at the CO were transformed into digital pulses by sampling techniques such as Pulse Code Modulation (PCM), which we have studied earlier. However, the communication between an end user and the CO continued to be analog. This was because there were huge investments in analog transmission technology between the CO and the end users (called the *last mile*). It was not easy to replace copper wires between the CO and the end user's premises overnight, simply due to the number of users who were connected in this fashion (millions of them).

Thus, we had a hybrid communications framework with analog communication between end user premises and the CO, and digital communication between the CO and the rest of the telephone network, as shown in Fig. 10.4.

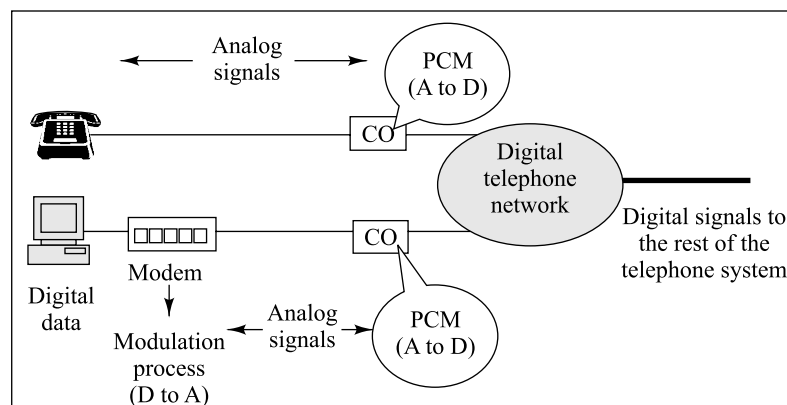


Fig. 10.4 Digital backbones, analog last mile

10.5.3 Integrated Services Digital Network (ISDN)

Even today, most of the telecommunications system is organized in a hybrid form (partly analog, partly digital) as shown earlier. Although this is not the best way to deal with communications; costs and other factors such as huge investments in analog transmission equipment and technology mean that this situation will not change rapidly. Nevertheless, in 1984, the world's major telephone companies came together and thought about an idea to digitize the whole chain of communication, including the last mile, but without changing the underlying copper wires. They thought about a new fully digital circuit-switched telephone system by early 2000. This system is called **Integrated Services Digital Network (ISDN)**.

ISDN aims to *integrate* voice and non-voice (primarily data) services together in *digital* form. Hence the name is ISDN. ISDN is already in place in many countries, and its usage is rising slowly. A conceptual view of ISDN is shown in Fig. 10.5.

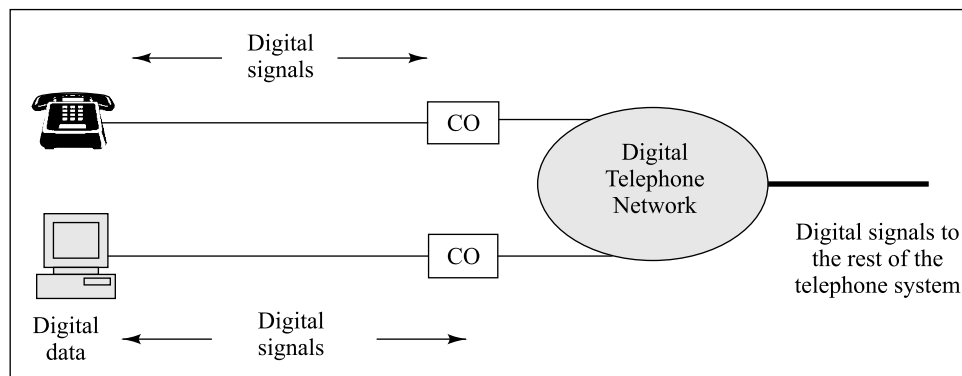


Fig. 10.5 Integrated Services Digital Network (ISDN)

It is thought that the main services provided by ISDN will be in terms of voice traffic. However, other enhancements are envisaged. ISDN can provide features that are highly desirable in Customer Relationships Management (CRM), based on features such as caller identification. Using this, the person receiving the call will know the telephone number of the caller, as in mobile telephony. ISDN will use this telephone number of the caller as the key to automatically read the customer's information from a database, and present it on to the operator's screen even before the operator can answer the call. This would enable an operator to greet the customer by name, know her/his background, and perhaps, her/his problem/service log to serve the customer better. Other voice services such as call forwarding; conference calls, etc., are enhanced in case of ISDN. In the area of non-voice applications, online burglar and smoke detector alarms can automatically call the police, fire brigade, etc., and provide address details using ISDN.

10.6 ISDN ARCHITECTURE

10.6.1 Digital Bit Pipe

The fundamental concept in ISDN is the **digital bit pipe**. This is a conceptual pipe through which bits flow between the end user and the CO (usually called **ISDN exchange** in the ISDN parlance). The key point about this bit pipe is that it is bidirectional. Bits can originate at the customer premises

or at the ISDN exchange. Also, it does not matter whether the bits correspond to the contents of a telephone call, a fax message, an email sent through a computer, or anything else.

The digital bit pipe supports time division multiplexing techniques to provide for multiple independent channels. The format of the bit stream and the multiplexing techniques are defined in the ISDN architecture. Early in the development of the ISDN standards, it was recognized that there would be two primary sets of users of ISDN services, viz., home users and business users. The bandwidth needs of home users are far smaller as compared to those of business users. Therefore, two separate standards were developed to address the two. A low bandwidth standard was defined for home users, and a high bandwidth standard (that supported multiple channels, each of which was identical to the home user channel) was developed for business users. Additionally, businesses can have multiple such bit pipes in case they need bandwidth more than that provided by a single *business bit pipe* (i.e., high bandwidth standard).

10.6.2 ISDN Channel Types

From a subscriber's point of view, ISDN offers three separate digital channels. These channels are designated as B, D and H. Each **Bearer (B) channel** carries digitized voice, data or compressed video up to the maximum rate of 64 Kbps. The **Data (D) channel** is used for signaling/controlling the B channels, and it operates at a data rate of 16/64 Kbps. The name *data* is actually a misnomer as the D channel is designed to carry only signal controlling information. In simple terms, a subscriber requests a service (such as a phone call by dialing a number, or sending a request for a webpage) over the D channel, and gets that service on a B channel. A **Hybrid (H) channel** offers data rates in the form of 384, 1536 or 1920 Kbps by combining multiple B and D channels. This is summarized in Fig. 10.6.

Channel	Data rate in Kbps
Bearer (B)	64
Data (D)	16/64
Hybrid (H)	384/1536/1920

Fig. 10.6 ISDN channel data rates

Let us study these channel types in detail.

Bearer (B) Channels

As we discussed, a B channel is defined at a rate of 64 Kbps. It is the basic user channel for carrying any kind of digital information in full-duplex mode with a maximum transmission rate of 64 Kbps. As we know, voice signals digitized using PCM also require 64-Kbps bandwidth (8,000 samples per second with 8 bits per sample). Thus, we can consider the B channel to be the replacement for the traditional analog phone system, called **Plain Old Telephone System (POTS)**. In some cases, if the transmission rates are higher than 64 Kbps, two B channels can be combined to allow a maximum transmission rate of 128 Kbps. However, one point must be stated. One B channel is destined to a single recipient only. It is not designed for demultiplexing a data stream midway to separate different transmissions and divert them to more than one destination.

Data (D) Channels

The main job of a D channel is to carry controlling signals (such as establishing a call, ringing, call interrupt, synchronization, etc.) for allowing the B channels to carry actual data. In traditional communication systems, the same interface carries data as well as controlling information (called **in-band signaling**). ISDN is different for separating these controlling tasks from the actual data transmission, and dedicating a separate channel for the carrying the control signals (called **out-band signaling**). There is a new protocol called **Signaling System Number 7 (SS7)**, that can be used over various digital circuit-switched networks to provide for out-band signaling (also called common-channel signaling). SS7 takes care of network intelligence functionalities (such as routing and delivery of the D channel messages (i.e., control messages)).

The subscriber uses the D channel for connecting to the network, and then for securing the B channel connection. After this, the subscriber uses the B channel to send and receive actual data. Conceptually, a D channel is similar to a telephone operator who assists you in making a telephone call. You pick up the phone and tell the operator which number to call. The operator makes the call and hands the line over to you once the call is successfully established. A D channel works in exactly the same fashion. If no signaling is required, the D channel itself can be used to carry data, such as in case of videotex, teletex, emergency services (alarm systems), etc.

Hybrid (H) Channels

Used for applications such as videoconferencing, which require higher bandwidth, H channel provides data rates of 384/1536/1920 Kbps. H channels simply offer higher bandwidth to the users. They can be used to subdivide the channel as per the user's own TDM scheme, or can simply be used as high-speed lines for applications such as faxes with high speed, high-speed data/audio, etc. Simply stated, H channels can be used in the place of B channels if higher bandwidths are desired.

10.7 ISDN INTERFACES

There are two ways in which ISDN can be used. These two interfaces supported by ISDN are **Basic Rate Interface (BRI)** mainly used by home users and **Primary Rate Interface (PRI)**, mainly used by corporate customers. We shall now study them one by one.

10.7.1 Basic Rate Interface (BRI)

The Basic Rate Interface (BRI) specifies a digital bit pipe that contains two B channels and one D channel (16 Kbps). The 2B+D channels are called *Basic Rate Interface (BRI)* in the ISDN terminology. Internally, ISDN uses a modified form of time division multiplexing technique to facilitate an illusion of multiple channels of data traveling over the same pair of wires. This is shown in Fig. 10.7.

As we can see, two B channels require 64 Kbps each. Additionally, the D channel demands 16 Kbps. Moreover, the BRI service itself needs another 48 Kbps for operating overhead. Therefore, a BRI ISDN interface needs a total of 192 Kbps bandwidth. Conceptually, we can consider a BRI service as a large pipe that contains three smaller pipes, i.e., two B channels and one D channel.

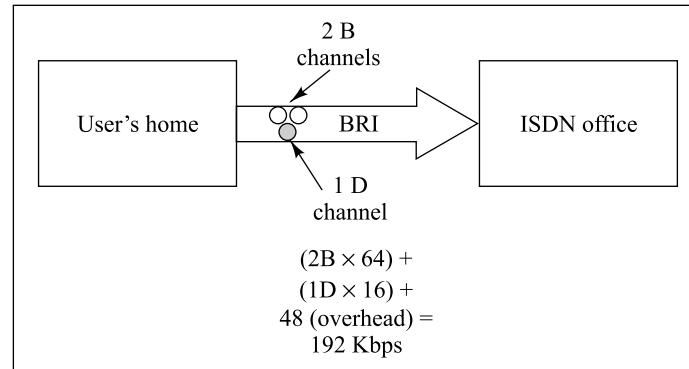


Fig. 10.7 Basic Rate Interface (BRI)

A BRI channel is suitable for residential and small office subscribers. For instance, a home user could use one B channel for browsing the Internet, and use the other line for making telephone calls at the same time. Of course, the subscriber can also combine the two B channels to have a faster connection to the Internet, instead, or may be for a videoconference.

10.7.2 Primary Rate Interface (PRI)

In comparison to the Basic Rate Interface (BRI), the *Primary Rate Interface (PRI)* has huge capacity. A PRI has 23 B channels instead of just two B channels. It also has the D channel for control signals. However, the D channel in case of a PRI is 64 Kbps. This makes the PRI a 23B+D channel. This gives a total bandwidth of 1.536 Mbps. The PRI service itself needs another 8 Kbps for its overheads. Thus, the total capacity of a PRI is 1.544 Mbps. Thus, a PRI service is a very big digital pipe consisting of 23 B channels and 1 D channel, as shown in Fig. 10.8.

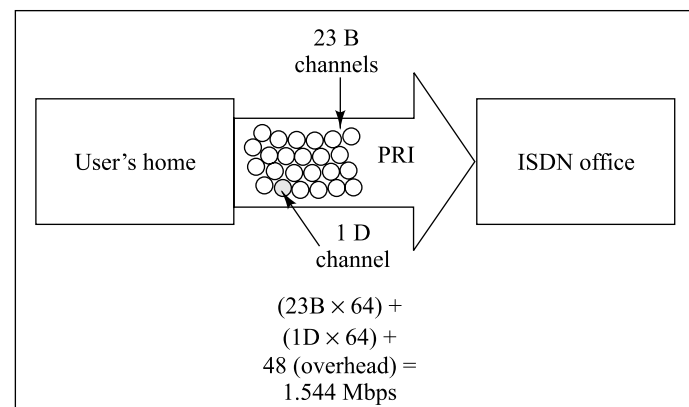


Fig. 10.8 Primary Rate Interface (PRI)

Since a PRI service contains 23 B channels, it means that it supports up to 23 full-duplex connections at the same time. The 23B+D design was invented keeping in mind, the existing telephone networks around the world. In the US, standards for digital telephone circuits were given

names that consist of the letter *T* followed by a number. For instance, a T-1 telephone line has a capacity of 1.544 Mbps. Therefore, a PRI works well with a T1 line in the US. In Europe, the PRI is changed to have a 30B+D service (i.e., 30 B channels and a D channel). This equates the European telephone standard of 2.048 Mbps, as used by an E-1 line.

10.8 FUNCTIONAL GROUPING

10.8.1 Introduction

The devices (such as computers, telephone machines) used to access ISDN are grouped to form **functional groupings**. A functional grouping clubs various devices, depending on their functionalities. For instance, some modern digital devices such as digital phones are ISDN-compatible, which means that they understand ISDN signaling and protocols. However, many other devices, such as our normal telephone instruments or computers do not understand ISDN signaling and protocols on their own. Therefore, they need some additional hardware and software pieces to make them ISDN-compatible. Consequently, such different devices are better kept in separate logical groups for ease of reference and understanding. This is similar to how a protocol model such OSI defines what each layer should do, and what interface it should provide to the adjacent layers. Internally, protocols within each layer might be different, but all work with each other as long as they conform to the OSI standard.

Similarly, here, the subscribers are free to select devices from different functional groupings, depending on their requirements. Since within each group, the devices must conform to the functional grouping standards, this poses no incompatibility issues. The functional groupings used at the user's premises are of three major types, viz., **Network Terminations (NT1 and NT2)**, **Terminal Equipment (TE1 and TE2)** and **Terminal Adapters. (TA)**

These functional groupings are shown in Fig. 10.9, and discussed thereafter.

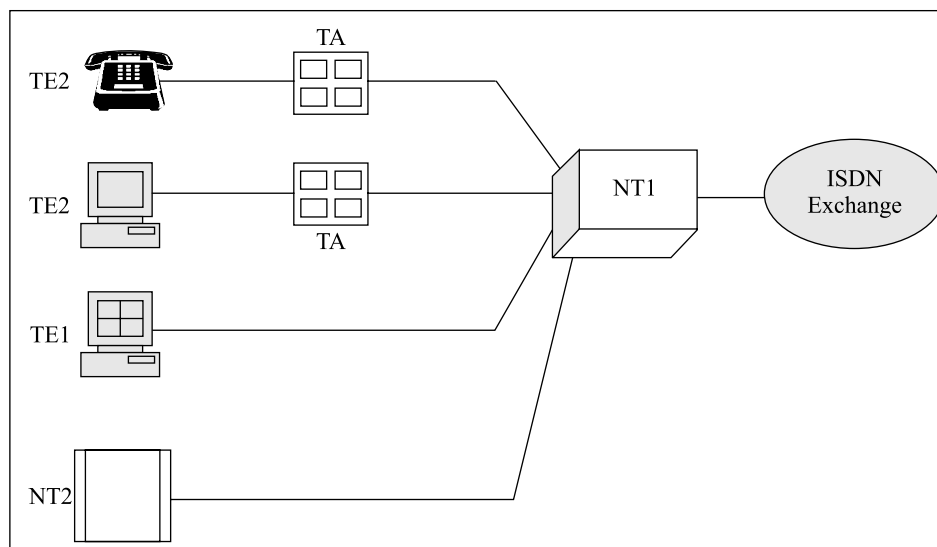


Fig. 10.9 Functional groupings

10.8.2 Terminal Equipments

Terminal Equipment 1 (TE1)

The **Terminal Equipment 1 (TE1)** is similar to the Data Transmission Equipment (DTE) concept. TE1 represents a device that supports ISDN standards. It is digital subscriber equipment such as digital telephone, digital fax machine, digital voice and data terminal. Therefore, it can be directly connected to NT1 as shown in the figure above.

Terminal Equipment 2 (TE2)

Devices such as our normal telephone instruments, computers (workstations/hosts), etc., do not *understand* ISDN. That is, they are not compatible with the ISDN standards, signaling and protocol mechanisms. Such devices belong to the functional group **Terminal Equipment 2 (TE2)**. As a result, TE2 devices cannot be used with ISDN directly. Rather, they need some additional hardware and software to function with the ISDN technology. As the figure shows, the additional hardware and software pieces are bundled inside another functional grouping called TA (which we shall study). Therefore, it cannot be directly connected to NT1 as shown in the figure above.

10.8.3 Network Terminations

Network Termination 1 (NT1)

The ISDN exchange places a **Network Termination 1 (NT1)** device inside the user's premises. The NT1 device is connected to the ISDN exchange (which is several kilometers away) using the existing twisted pair wires that are used for the telephone networks. The NT1 box contains a connector to which up to eight ISDN devices (such as telephones, computers, alarms, etc.) can be connected similar to the way a LAN is formed.

From a user's point of view, NT1 is the boundary of ISDN. Thus, a NT1 device controls the physical and electric aspects of ISDN in the user's premises and connects the user's devices to the digital bit pipe. This is similar to the physical layer of the OSI model.

Network Termination 2 (NT2)

For large businesses, another piece of equipment called the **Network Termination 2 (NT2)** is required. Such businesses usually need to support more than one telephone conversations at the same time. The intelligent NT2 device acts like a **Private Branch eXchange (PBX)**. It coordinates the transmissions from multiple incoming links and multiplexes them, so that they can be sent across to the ISDN exchange by the NT1 device. NT2 generally maps to the layers 2 and 3 of the OSI model.

10.8.4 Terminal Adapter (TA)

A **Terminal Adapter (TA)** acts like a converter. It transforms information received from non-ISDN devices into ISDN format, and vice versa. Clearly, since TE2 devices are non-ISDN, TA is used in conjunction with them. TA also resides in the user premises.

10.9 REFERENCE POINTS

A **reference point** defines the interface between two functional groupings. Thus, a reference point indicates how devices from two functional groupings should be connected, how they will exchange data, etc.

Figure 10.10 tabulates the reference points between the various functional groupings.

Reference Point	Description
R	Defines the connection between a TE2 and TA.
S	Defines the connection between a TE or TA and NT1 or NT2.
T	Defines the connection between a NT2 and NT1.
U	Defines the connection between a NT1 and ISDN exchange.

Fig. 10.10 Reference points

Figure 10.11 shows the reference points in ISDN connections.

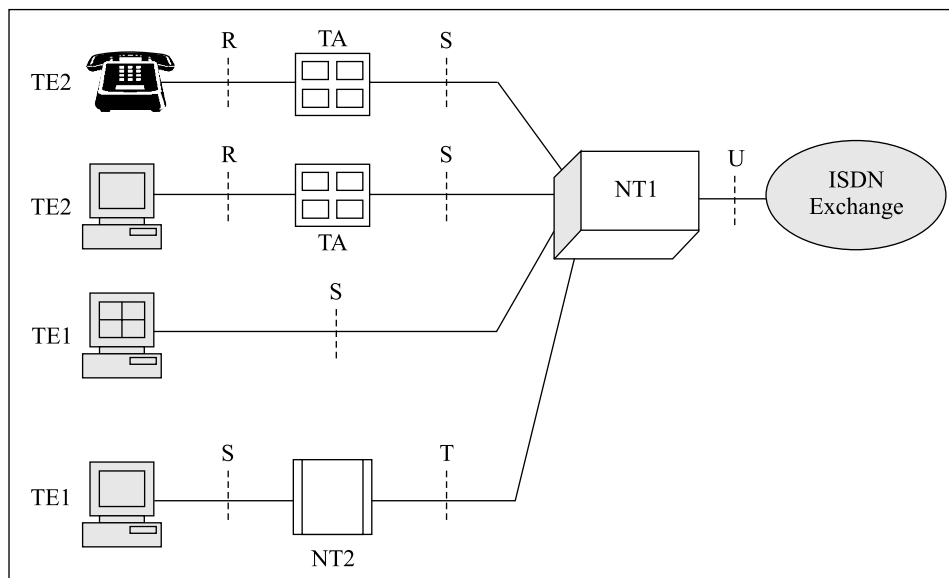


Fig. 10.11 Reference points in ISDN connections

10.10 ISDN PROTOCOL ARCHITECTURE

10.10.1 Introduction

The OSI model is a generic framework used to describe protocol stacks of specific implementations. However, ISDN protocol stack is complex, and cannot be described using the standard OSI model. Instead, a variation of the OSI model is used. Why is this required? Recall that ISDN defines two

main types of channels: a B-channel is used for user-to-user information exchange (data), whereas a D-channel is used for user-to-network communication (signaling). A single OSI model cannot accommodate both, since they internally use different protocols.

Moreover, since one of the major goals of ISDN is to integrate different kinds of information and services (such as computer data, telephone calls and fax messages), the management of these different functions also needs specialized handling, which is not needed in case of other simpler protocols. Thus, we have three sets of requirements, one for user information, the second for control signaling, and the third one for management functions.

Considering these aspects, ISDN is classified into three protocol blocks or planes, namely the **user protocol block (or user plane)**, **control protocol block (or control plane)** and **management protocol block (or management plane)**. Conceptually, these planes are shown in Fig. 10.12.

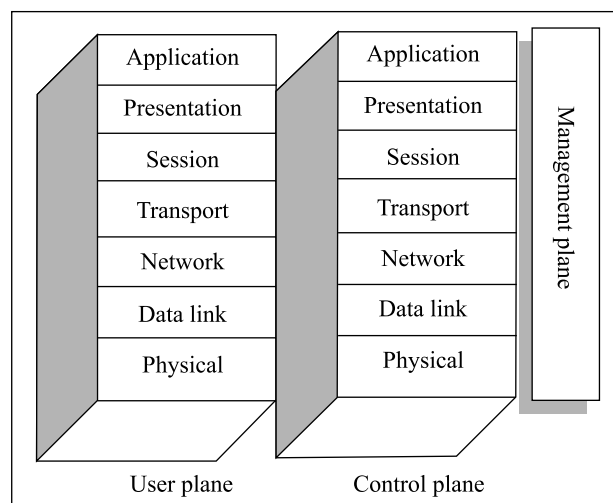


Fig. 10.12 *ISDN protocol layers*

The user plane deals with transfer of user information. The control plane is responsible for ISDN signaling. We shall discuss these two aspects subsequently. The management plane cuts across all the ISDN layers and is used for network management functions (such as remotely managing a network), and is out of the scope of this text.

Having got a general idea of the ISDN protocol layers, let us now think which protocols exist in which of these layers. To simplify our understanding, we shall consider the B channel (user plane) and the D channel (control plane) as two separate logical blocks.

- The physical layer protocols for both the B channel and the D channel are the same (BRI or PRI, as discussed earlier).
- At the data link layer, the B channel contains protocols such as frame relay (which we shall study in detail in a subsequent chapter) and LAPB. Here, the D channel contains the LAPD protocol.
- At the network layer, the B channel contains the X.25 protocol (which we shall study in detail in a subsequent chapter), whereas the D channel can also use X.25, and additionally, Q.391 (call control) protocol.

These protocols are shown diagrammatically in Fig. 10.13.

Transport, Session, Presentation, Application	Not specific to ISDN	Not specific to ISDN
Network	X.25	Q.391, X.25
Data link	Frame Relay, LAPB	LAPD
Physical	BRI and PRI	
	B channel	D channel

Fig 10.13 ISDN protocols in the B and D channels

10.10.2 Physical Layer

ISDN protocols come in two variations at the physical layer, viz., the Basic Rate Interface (BRI) and Primary Rate Interface (PRI).

Physical Layer – BRI

A BRI interface consists of 2 B channels, and 1 D channel, together constituting a 2B+D channel. A subscriber can connect to a BRI interface using the R, S or U reference points. Each of the reference points specify how many wires are to be used, what are the electrical specifications of the wires, etc.

The BRI frame format is shown in Fig. 10.14. The B channel consists of two subchannels, B1 and B2. Each of B1 and B2 is 8 bits and occurs twice, making a total of 16-bit B1 and 16-bit B2. Thus, overall, the B channel contains 32 bits. The D channel contains 4 bits. There are 12 bits reserved as overhead bits. Their positions are shown with check boxes in the figure. Note that we have not shown all the 12 bits distinctly, but have simply shown how the 12 overhead bits occupy 9 positions by forming groups. For instance, there are two overhead bits before the first B1, etc. Therefore, all 12 overhead bits are not shown separately. Instead, we have simply shown as to *where* they are located in the frame. Thus, a BRI frame contains 48 bits overall.

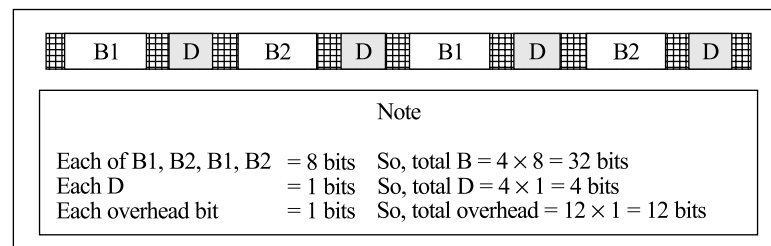


Fig. 10.14 BRI frame format

Now, if we send 4000 such frames in one second, we will send the following bits:

$$\begin{array}{lll}
 (B_1 + B_2 + B_1 + B_2) \text{ 4000 times} & = (32 \text{ bits of one B channel}) \text{ 4000 times} & = 128 \text{ Kb} \\
 (D) \text{ 4000 times} & = (4 \text{ bits of D channel}) \text{ 4000 times} & = 16 \text{ Kb} \\
 (12 \text{ bit overhead}) \text{ 4000 times} & = (12 \text{ bits of overhead}) \text{ 4000 times} & = 48 \text{ Kb}
 \end{array}$$

$$\text{Total (in 1 second)} = 192 \text{ Kb}$$

This explains why the BRI rate is 192 Kbps/second. Usually, the B channel capacity of 128 is written as 2B (where each B = 64 Kbps). This makes BRI a 2B + D channel.

Physical Layer – PRI

As we know, the PRI interface supports 23 B channels and 1 D channel. Thus, a PRI frame contains 23 B channels and 1 D channel. The PRI frame format is identical to the T-1 frame format, and is shown in Fig. 10.15.

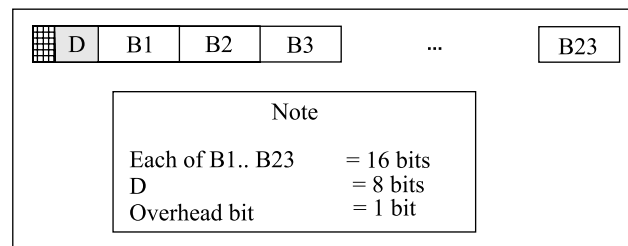


Fig. 10.15 PRI frame format

As we can see, the PRI frame format is considerably simpler than the BRI frame format. The bit size of each PRI frame is 193 bits (23 B channels, each of 8 bits + 1 D channel of 8 bits + 1 overhead bit). PRI frames are sampled at a rate of 8000 per second. Thus, we have an effective transmission rate of (193 bits per frame (i.e., sample) \times 8000 samples per second =) 1.544 Mbps.

10.10.3 Data Link Layer

At the data link layer, the B channel consists of the Frame Relay and LAPB protocols, which we shall study subsequently. The D channel consists of the **Link Access Procedure for the D channel (LAPD)** protocol, which we shall discuss here.

The LAPD frame format is shown in Fig. 10.16. Of the fields shown, only the address field is significant from the point of view of ISDN. Other fields are similar to those in any other data link layer frame.

As the figure shows, the address field of a LAPD frame consists of two bytes.

- The first byte contains six bits of **Service Access Point Identifier (SAPI)**, one bit to indicate if this is a command (C) or Response (R), and the last bit indicating whether the address is complete or is continuing in the next byte (0 means it will continue in the next byte, a discussion of which is beyond the scope of the current text). The SAPI field is useful for the network layer service (e.g., to answer the question: *What is the purpose of the D channel here?*).

- The second byte contains a field called **Terminal Equipment Identifier (TEI)**. It is used to uniquely identify a TE. Since this field consists of seven bits, it can identify different 128 TEs.

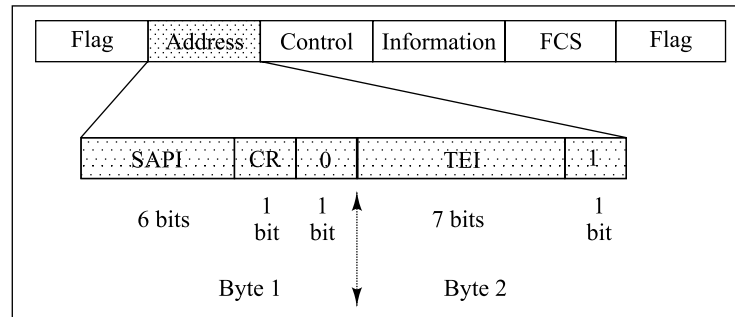


Fig. 10.16 LAPD frame format

10.10.4 Network Layer

The B channel uses the X.25 protocol, which we shall study in detail subsequently. The D channel also supports X.25, but additionally it also works with the Q.391 protocol. We have discussed the LAPD frame format earlier. One of the fields in the LAPD frame is *Information*. This *Information* field actually encapsulates the network layer packet. At the network layer, the ISDN packet is called a **message**. Thus, a *message* field in the network layer is transformed into *information* field at the data link layer. The message field contains many subfields, as shown in Fig. 10.17.

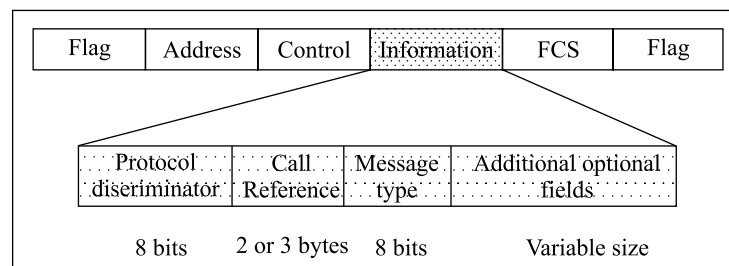


Fig. 10.17 Subfields of the message field

Let us discuss the four subfields of the *message* field.

- Protocol discriminator** – This field identifies the protocol in use.
- Call Reference** – This field shows the sequence number of the call.
- Message Type** – This one-byte field identifies the purpose of the message. A message can be used for call establishment, call information, call clearing and miscellaneous purposes.
- Additional optional fields** – These fields carry details that are specific to a particular call. For example, these fields contain the ISDN addresses of the sender and the receiver (discussed next), routing details, protocol details (circuit switching or X.25 or ATM or Frame Relay, etc.). This is followed by the actual data contents.

10.10.5 ISDN Addressing

In our normal telephone system, calls are placed on the basis of the telephone numbers. The caller's telephone number is used for charging the call amount, and the called party's telephone number is used to actually place the call. In order to facilitate this, every telephone subscriber has a unique telephone number. Since ISDN is based on the concept of the telephone network, its numbering (i.e., addressing) scheme is also based on that of the worldwide telephone network. However, there are a few differences. Most notable of them is that ISDN must cater to not only telephones, but also to other devices such as computers. Therefore, the addressing scheme must cater for this.

The ISDN address structure is shown in Fig. 10.18.

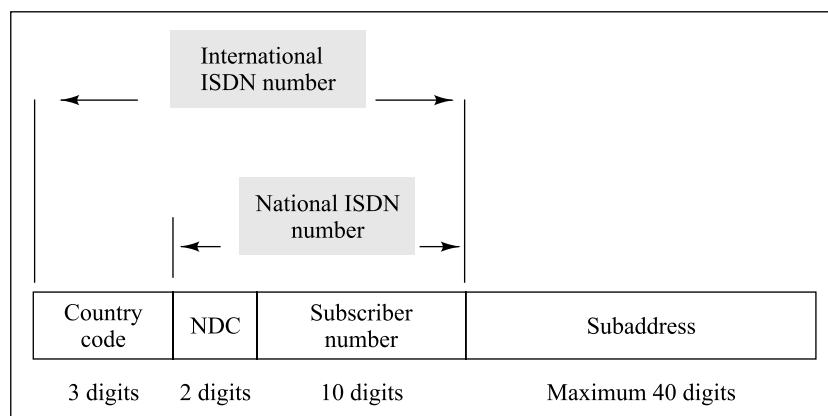


Fig. 10.18 *ISDN address*

The various fields of the ISDN address are described in Fig. 10.19.

Address field	Description
Country code	This three-digit field describes the destination country code of the ISDN number.
National Destination Code (NDC)	If more than one ISDN service providers serve the subscribers within a single country, this two-digit field identifies one of them uniquely.
Subscriber number	This ten-digit field identifies the subscriber's ISDN number. Usually, it contains a three-digit area code and a seven-digit ISDN phone number.
Subaddress	This field can contain up to 40 digits, and contains additional information.

Fig. 10.19 *ISDN address fields*

Let us consider an example. Suppose an ISDN subscriber dials a number 453-716-8919-788 to make a call within the country on the same ISDN network. Since the calling and the called persons are in the same country and are on the same network, there is no need for the country code and the NDC. Therefore, the first ten digits simply identify the called person's subscriber number. The last three digits represent the extension number of the called person (subaddress).

We mentioned that ISDN addressing works for telephones as well as computers. The way ISDN telephone addresses work should be clear from the above discussion. What is done for ISDN addressing with reference to computers? For this, ISDN addresses are assigned to the S reference points (Refer to Fig. 10.10). As we can see from that figure, the S reference point identifies a unique ISDN terminal. Therefore, once we assign an address to this reference point, we are actually able to identify an ISDN terminal uniquely.

10.11 NARROWBAND-ISDN (N-ISDN) AND BROADBAND ISDN (B-ISDN)

ISDN was primarily developed for higher bandwidths to the home and business users. However, soon it was realized that the BRI and PRI maximum transmission rates of 128 Kbps and 1.544 Mbps were not good enough for some high-end applications such as television transmissions, High Definition Television (HDTV), etc. Also, the basic BRI and PRI services are not sufficient for carrying too many parallel transmissions. Consequently, the development of the next generation of ISDN services, called **Broadband ISDN (B-ISDN)** is in progress. The original, plain vanilla version of ISDN is now called **Narrowband ISDN (N-ISDN)**. B-ISDN is expected to provide data rates of up to 600 Mbps, about 400 times faster than the PRI data rate.

However, B-ISDN requires optical fibers to be placed even in the last mile (i.e., from a telephone company's CO to its subscribers' homes). Clearly, this will take a lot of investment and involves many technical questions that must be solved.

The B-ISDN technology is closely related with the Asynchronous Transfer Mode (ATM) technology, which we shall study in a separate chapter. This has remarkable consequences. ATM is a packet switching technology. ISDN began as an extension of the traditional circuit-switching telephone networks. However, with progression towards B-ISDN, there is a transformation from circuit switching to packet switching. Of course, this does not mean that B-ISDN will not support circuit-mode applications.

SUMMARY

The Medium Access Control (MAC) sublayer is a part of the data link layer, and performs many useful functions in any computer network.

The traditional communication between two computers is over the voice network. It is analog and offers small bandwidth. Due to its limitations, digital backbones were soon in place. Slowly, a movement towards an all-digital network came about. The outcome of this research is Integrated Services Digital Network (ISDN).

ISDN offers an integrated digital service for all kinds of traffic, most notably, voice and data. ISDN uses the existing twisted-pair wires. There are three basic ISDN channels, viz., B, D and H. ISDN offers data rates in two different categories for business and home users. Home users generally go for Basic Rate Interface (BRI), which offers them 2B+D data rate (usually 192 Kbps). Business users opt for Primary Rate Interface (PRI), which supports 23B+D data rates (usually 1.544 Mbps).

For ease of implementation, ISDN devices are classified using functional groupings. Each functional group contains devices that are either ISDN compatible, ISDN incompatible, or make

incompatible ISDN devices compatible. Reference points demark the various functional groupings. ISDN protocol layers contain *planes*. ISDN works in the physical, data link and network layers of the OSI model.

ISDN addresses are identified using a combination of country code, national destination code, subscriber number, and subaddress.

The current transmission rates offered by ISDN may not be good enough for modern multimedia applications. The B-ISDN technology is likely to overcome these drawbacks.

KEY TERMS AND CONCEPTS

Basic Rate Interface (BRI)	Management protocol block
Bearer (B) channel	Medium Access Control (MAC)
Broadband ISDN (B-ISDN)	Narrowband ISDN (B-ISDN)
Channel allocation	Network Termination 1 (NT1)
Control Plane	Network Termination 2 (NT2)
Control protocol block	Out-band signaling
Data (D) channel	Physical address
Digital bit pipe	Plain Old Telephone System (POTS)
Dynamic channel allocation	Primary Rate Interface (PRI)
Fixed channel allocation	Private Branch eXchange (PBX)
Functional groupings	Reference point
Hybrid (H) channel	Service Access Point Identifier (SAPI)
Hybrid channel allocation	Static channel allocation
In-band signaling	Terminal Adapter (TA)
Integrated Services Digital Network (ISDN)	Terminal Equipment 1 (TE1)
ISDN exchange	Terminal Equipment 2 (TE2)
Link Access Procedure for the D channel (LAPD)	Terminal Equipment Identifier (TEI)
MAC address	User plane
Management plane	User protocol block

QUESTIONS

True/False

1. ISDN needs optical fiber.
2. ISDN is a digital technology.
3. Digital networks began with the *last mile*.
4. Each B channel consists of 64 Kbps.
5. BRI has higher data rate than PRI.
6. BRI contains three B channels.
7. PRI contains 23 B channels.
8. PRI offers 1.544 Kbps bandwidth.
9. TA is required in case of TE2 devices.

10. NT1 is the end point of an ISDN connection from a user's point of view.
11. NT1 is equivalent to a PBX.
12. The S reference point lies between TE2 and TA.
13. The control plane appears across all ISDN layers.
14. The SAPI field identifies the purpose of a D channel.
15. The TEI field can identify up to 128 TEs.
16. The subscriber's ISDN address consists of 10 digits.
17. After B-ISDN emerged, the current ISDN is called C-ISDN.

Multiple-Choice Questions

1. ISDN stands for _____ Services Digital Network.
 - (a) International
 - (b) Internet
 - (c) Integrated
 - (d) Interoperable
2. The *last mile* usually consists of _____.
 - (a) twisted-pair wires
 - (b) optical fibers
 - (c) coaxial cables
 - (d) radio transmission
3. The fundamental concept in ISDN is _____.
 - (a) analog signals
 - (b) digital bit pipe
 - (c) voice communications
 - (d) computer communications
4. A bearer channel contains _____.
 - (a) control information
 - (b) management functions
 - (c) miscellaneous functions
 - (d) user data/information
5. The data channel contains _____.
 - (a) control information
 - (b) management functions
 - (c) miscellaneous functions
 - (d) user data/information
6. The B channel provides a data rate of _____.
 - (a) 1.544 Kbps
 - (b) 16 Kbps
 - (c) 64 Kbps
 - (d) 128 Kbps
7. The BRI consists of _____.
 - (a) 2B + D
 - (b) B + D
 - (c) B + 2D
 - (d) 2B + 2D
8. The PRI consists of _____.
 - (a) 23B + 2D
 - (b) B + 23D
 - (c) 23B + D
 - (d) 23B + 23D
9. _____ do(es) not need a TA.
 - (a) TE1
 - (b) TE2
 - (c) Both
 - (d) None of the above
10. _____ lie(s) inside the user's premises.
 - (a) NT1
 - (b) NT2
 - (c) TE1
 - (d) All of the above
11. A _____ defines the interface between two functional groupings.
 - (a) divider point
 - (b) line segment
 - (c) reference point
 - (d) logical block

12. _____ lies between NT1 and ISDN exchange.
 (a) R (b) S
 (c) T (d) U
13. ISDN is classified into _____ protocol blocks.
 (a) 2 (b) 3
 (c) 4 (d) 5
14. The data link layer of the control block supports _____.
 (a) Ethernet (b) HDLC
 (c) Frame Relay (d) FDDI
15. The _____ field indicates the purpose of the call.
 (a) message name (b) message address
 (c) message type (d) message header
16. The country code in ISDN address consists of _____ digits.
 (a) 3 (b) 4
 (c) 2 (d) 0

Detailed Questions

1. Explain MAC. What is its purpose?
2. Contrast between static and dynamic channel allocation mechanisms.
3. Discuss the development of ISDN from the days of analog telephone networks.
4. What is *last mile*?
5. What are B, D and H channels? What are their capacities and purposes?
6. What are inband and outband signaling methods?
7. What is BRI? How is its architecture defined?
8. In what respects is PRI different from BRI?
9. Who and why subscribe to BRI and PRI?
10. What are Network Terminals? Discuss NT1 and NT2.
11. Why is a terminal adapter required?
12. What is the difference between TE1 and TE2?
13. What is a reference point? What are its significances?
14. Describe the frame formats for BRI and PRI.
15. Discuss the data link layer protocols in ISDN.
16. How many countries can be defined using ISDN address? Why?
17. What are the advantages and limitations of ISDN?

11

X.25 Protocol

11.0 INTRODUCTION

X.25 is a packet switching protocol used in a Wide Area Network (WAN). X.25 was developed in the mid-1970s by ITU-T. X.25 describes how a computer can be connected to a WAN, and how the communication between that computer and the rest of the WAN can take place. It defines standards for establishing, maintaining and terminating connections between them.

X.25 works at the bottommost three layers of the OSI model—the physical layer, the data link layer and the network layer.

1. At the physical layer, X.25 defines a protocol called **X.21**, which is specific to X.25. Conceptually, this is similar to other physical layer protocols, and defines voltage levels, etc.
2. At the data link layer (which is called **frame layer** in X.25 parlance), we have a bit-oriented protocol called **Link Access Procedure Balanced (LAPB)**. LAPB is a subset of the popular **High-level Data Link Control (HDLC)** protocol. It defines the frame formats, etc., as we shall study. This layer is responsible for flow control and error control between adjacent nodes.
3. At the network layer (which is called **packet layer** in X.25), there is a protocol called **Packet Layer Protocol (PLP)**. PLP is responsible for establishing the connection, transferring the data and terminating the connections between the end parties. This protocol sets up the virtual circuits (discussed subsequently) between the sender and the receiver. This layer is also responsible for end-to-end flow control and error control (normally, this is done by the transport layer, which is missing in X.25).

11.1 UNDERSTANDING HOW X.25 WORKS

The X.25 protocol can be best understood with the help of an analogy. We shall take a contrived example solely for the purpose of understanding, even if this seems a bit artificial at some places.

Let us assume that we have two offices of an organization (XYZ Corp. Limited), which are geographically apart—say, one is in Delhi, and the other one in New York. Suppose the Delhi office needs to send a 100-page document using the postal service to the New York office. However, for some strange reason, the document cannot be sent in one envelope. Instead, every page has to be sent separately in its own envelope. Then the following steps will take place.

1. One person (say, A) at the Delhi office will take each page from the document and will keep a copy of each of the 100 pages with her/him.
2. A will write the recipient's address (New York office address), sender's address (Delhi office address) and a sequence number (e.g., Page 1 of 100, Page 2 of 100, etc.) on each page. Such a page is shown in Fig. 11.1.

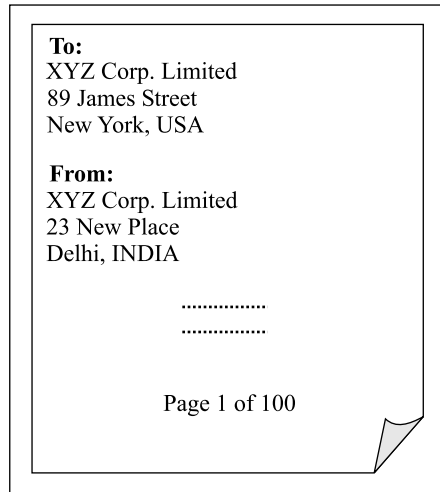


Fig. 11.1 *Sample page*

3. A will then put each such page inside a separate envelope and seal it.
4. However, A cannot simply put the address of the New York office and a stamp on this envelope and post it. This is because a rule of XYZ Corporation says that each envelope has to travel via a number of branches of XYZ Corporation that are located in the route between Delhi and New York. At each of these branch offices, the envelope has to be opened, its contents are to be verified by the local officer, and only if they are found accurate, it can be forwarded to the next branch office in the route. Otherwise, it must be discarded. To facilitate such a scheme, A writes the address of the next office in the route (in this case, Singapore), as the recipient's address on the envelope. This is shown in Fig. 11.2.

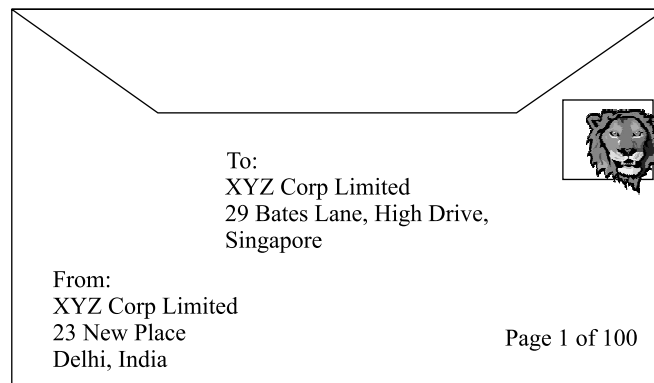


Fig. 11.2 *Sample envelope*

5. This scheme is such that as each envelope travels through a series of branches of XYZ Corporation, at each stage, a number of steps will happen. Let us discuss these steps, assuming that our clerk A (in this case, from Delhi) has sent the 100 envelopes one after the other in the fashion described earlier, to the clerk (named B) in the next branch (in this case, Singapore).
- (a) B will open each of the arriving envelopes, and see if its contents are intact or whether they have been damaged for any reason (as shown in Fig. 11.2). If the contents have become illegible, B shall call up A and request her/him to re-send that envelope with a good copy. Figure 11.3 shows the example of a damaged page, page number 27. Of course, the contents are not shown proportional due to lack of space.

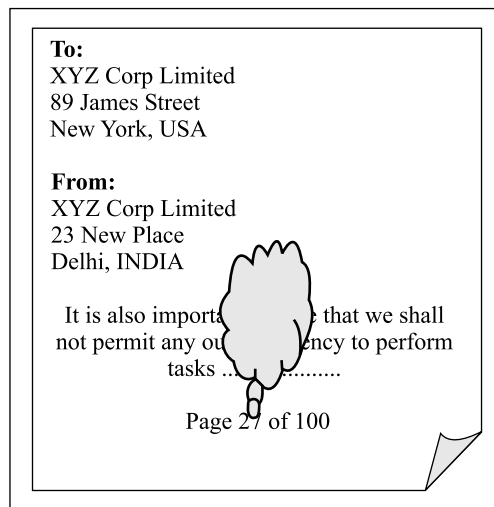


Fig. 11.3 *Damaged (illegible) page*

- (b) If the contents are found legible, B shall first create a copy of the page (the reason for this is explained subsequently), put the original page inside a fresh envelope, shall write the address of the next branch office as the *To:* address on the envelope, seal it, and send it to its next branch office. B will also send an acknowledgement back to A to indicate that s/he has successfully received the page.
- (c) Only at this juncture, A will be assured that the delivery of the page from her/his end was successful. At this point, A destroys the copy of that page, which it had maintained with her/him.

Let us draw a flow chart for the steps performed by a clerk such as B as shown in Fig. 11.4 to summarize the above steps.

6. We can imagine that when the next clerk, C, receives the envelopes from B, s/he will perform exactly the same checks as done by B. Only when s/he is satisfied that the correct envelope in sequence has arrived, C will read open it, read the contents of the page, make sure that they are legible, make a copy of it, put the original page in a new envelope, seal it, write the address of the next branch office as the *To: address*, and it to the next branch office. S/he will also send an acknowledgement back to B, after which, B will destroy her/his copy of

the page. If C finds any problem during this process, s/he shall call up B and request her/him to re-send a copy of the envelope, as it was damaged in some way.

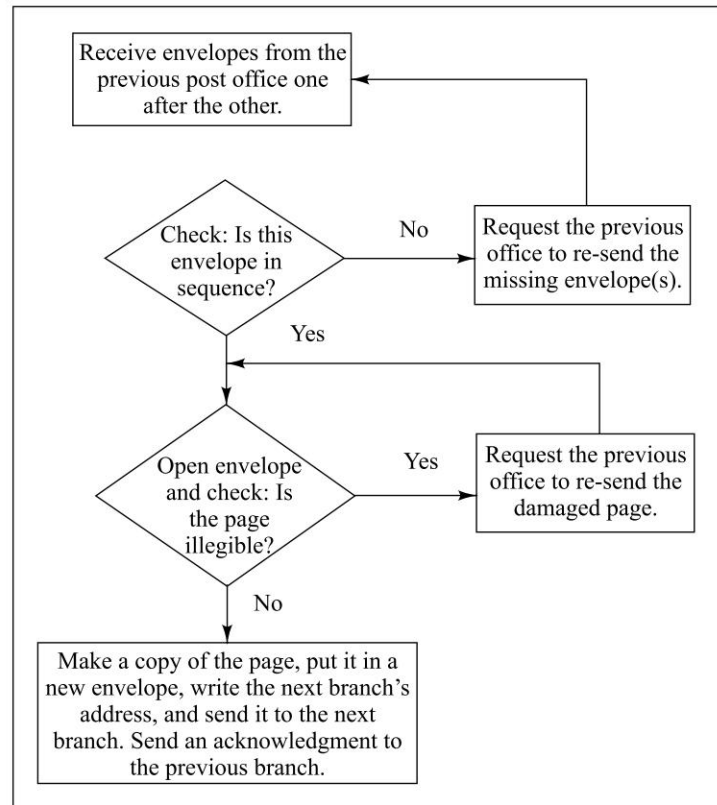


Fig. 11.4 *Process flow for a branch office clerk*

7. This process shall continue with the next branch offices, up and including the last destination, which is the New York office of XYZ Corp. Limited. Here, all the 100 pages will be received, using the process described earlier. The process will stop here.

By now, you must be wondering why we are describing this communication in such great detail. Well, the X.25 protocol works more or less in the same manner. The basic concepts of delivery, error checking, retransmissions, etc., are very similar to the way they are implemented in X.25.

11.2 CHARACTERISTICS OF X.25

The X.25 standard describes the communication protocols including delivery, error control, flow control, routing, etc., between a computer (called **Data Transmission Equipment** or **DTE**) and a network end point (usually a router or a gateway, also called **Data Communications Equipment** or **DCE**). Conceptually, this can be equated to the protocol on the local loop between a telephone and the local Central Office (CO). This logical equivalence is shown in Fig. 11.5.

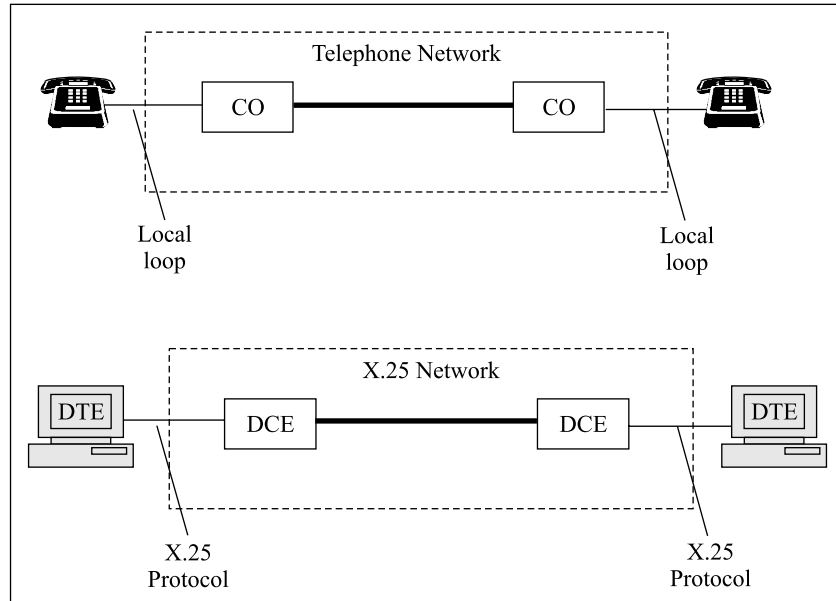


Fig. 11.5 Analogy between telephone local loop and X.25 protocol

Furthermore, X.25 has the following characteristics:

1. **Packet network** – X.25 is a packet network. This means that the data to be sent is broken into packets at the source, and the packets are sent to the destination, one by one.
2. **Protocol layers** – X.25 standard defines the communication protocols between a DTE and a DCE at three layers, viz., physical, data link and network. This is shown in Fig. 11.6.
3. **Types of Services** – X.25 supports two types of services, namely, **Permanent Virtual Circuit (PVC)** and **Switched Virtual Circuit (SVC)**. Let us discuss these in brief, although we have visited these terms earlier.

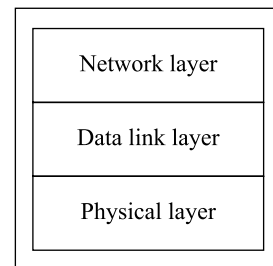


Fig. 11.6 X.25 layers

- (a) PVCs and SVCs both use packet switching, but set up a connection-oriented path through the network before the data is sent. A PVC is *provisioned*, which means that someone places the configuration information on non-volatile storage so although it takes a long time to set up, the PVC survives power failures as well. Like a telephone call, an SVC is created on demand—a computer requests an SVC, sends data, and then terminates the SVC.
- (b) Leased lines are used in circuit switching. A PVC is similar to the packet switching version of a leased line. In this method, the same virtual circuit is provided between two users on a continuous basis. Thus, a PVC is always *on*. There is only one time set-up and installation (in hardware/software), which marks the connections once and for all. Thus, all packets between two users connected via a PVC always travel via the same path, without needing any connection establishment/termination.

- (c) SVC is similar to a dial-up connection using circuit switching. A user requests for a virtual circuit, waits for the SVC to be established, and then starts using it. After the transmission is over, the user explicitly closes the SVC. Once a SVC is established, all the packets during the course of that transmission take the same route from the source and the destination. However, this is true only while the SVC is active. Once terminated, the sender, the receiver, and the network have no knowledge of the SVC. For a new transmission, a fresh SVC must be established.

11.3 PACKET FORMAT

Let us now discuss the X.25 packet format (i.e., the LAPB packet format, since the LAPB protocol is used in the data link layer). Actually, X.25 defines a number of packet formats. The packet formats differ depending on what they are supposed to do. For instance, different packet formats exist for data, control, status indication, and diagnostics. The different packet types are identified by their (different) headers. Every X.25 packet must contain at least three octets in the header portion – there can be more. We shall first examine what these three minimum octets contain, as shown in Fig. 11.7.

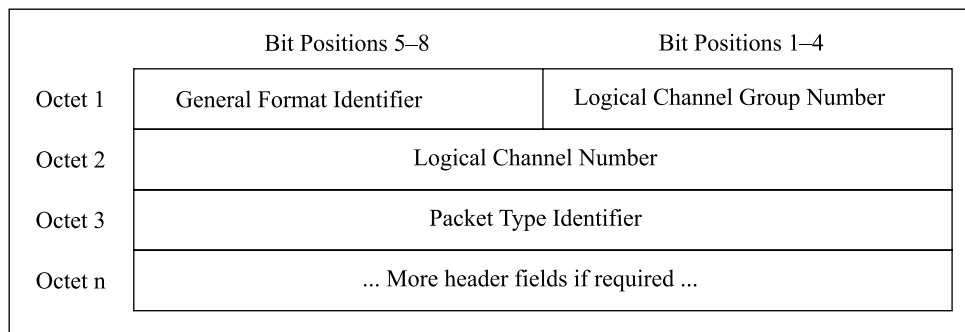



Fig. 11.7  X.25 Packet header format

There can be 16 **logical channel groups** (because it is a 4-bit field as shown in the figure) between two X.25 DTEs, each containing up to 255 **logical channels** (because it is a 8-bit field as shown in the figure), thus providing for a maximum of 4,096 logical connections per DTE. The first four bits in the first octet of the header identify a logical channel group, whereas the entire second octet identifies a logical channel number within a logical channel group.

Logical channel group + **Logical Channel Number (LCN)** together identify a logical connection (or virtual circuit) uniquely from all other logical connections.

The **General Format Identifier (GFI)** contains miscellaneous fields, which we shall not discuss. The **Packet Type Identifier (PTI)** defines the type of the packet, as discussed in the next section.

11.3.1 Data Packets

If a packet contains a 0 in bit number 1 of octet number 3 (i.e., the PTI field), it is a **data packet**. A data packet generally has three octets in the header portion. There can be a maximum of 128 octets after the header octets. In case of a data packet, the first and the second octets of the header field are unchanged. However, the third octet contains some special values, as illustrated in Fig. 11.8.

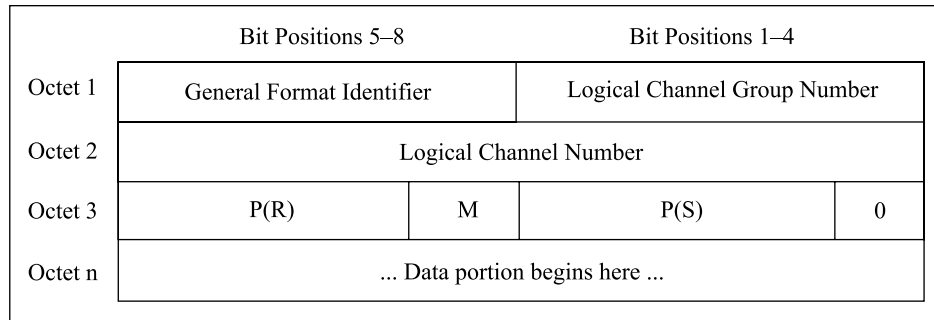


Fig. 11.8 *Format of a X.25 data packet*

Figure 11.8 shows the standard format of a X.25 data packet. As the figure shows, the first bit in the third octet indicates that this is a data packet. Also, there are three fields shown here, which need explanation.

The M (*More*) single-bit field indicates whether the receiver should expect more data from the sender or no. If this field contains a 1, it means that there is more data to follow.

P(S) and P(R) are data packet counters. Since each of these is a 3-bit field, each one can contain a value between 0 and 7 in decimal (i.e., 000 to 111 in binary). The P(S) field is called the **send sequence number**. The sender increases the value of P(S) every time it sends out a data packet. Thus, the P(S) of each outgoing data packet is one more than that of the preceding packet on the same virtual circuit (modulo 8 or modulo 128). The **receiver sequence number** or P(R) contains a number that indicates the number of the next packet expected from the other end of a virtual circuit.

11.3.2 Control Packets

Basic Control Packets

The other category of X.25 packets is **control packets**. In order to begin and end data transmission using data packets, the sender and the receiver must agree on a set of rules that will be used in the communication between them. The control packets help in setting up a X.25 virtual circuit to begin transmission, maintaining it while it is in use and tearing it apart when the communication in a session is over.

There are many control packet types in X.25. To best understand them, we can take an analogy of a telephone call. Let us first describe what happens during the establishment of a telephone call vis-à-vis the set-up of a X.25 connection as shown in Fig. 11.9.

As the figure shows, there are seven types of control packets. Note that an event occurring at one side of the virtual circuit is paired with another event on the other side. For instance, a *Call Request* packet on the sender's side results into an *Incoming Call* packet on the other side. Let us see how these packets are used in a real-life X.25 call as shown in Fig. 11.10. It shows the complete flow of a Virtual Circuit between two DTEs.

Telephone Network	X.25 Packet Network
Sender makes a call (Dial a number)	Sender sends Call Request packet
Receiver hears the phone ringing	Receiver receives Incoming Call packet
Receiver picks up the ringing phone	Receiver sends Call Accepted packet
Receiver hears <i>Hello</i> from the sender	Sender receives Call Connected packet
Sender wants to end conversation	Sender sends Clear Request packet
Receiver comes to know of end of conversation request	Receiver receives Clear Indication packet
Hang up	Receiver sends Clear Confirmation packet (with causes as indicated in other fields of the packet)

Fig. 11.9 *Analogy between a phone call establishment and a X.25 connection set-up*

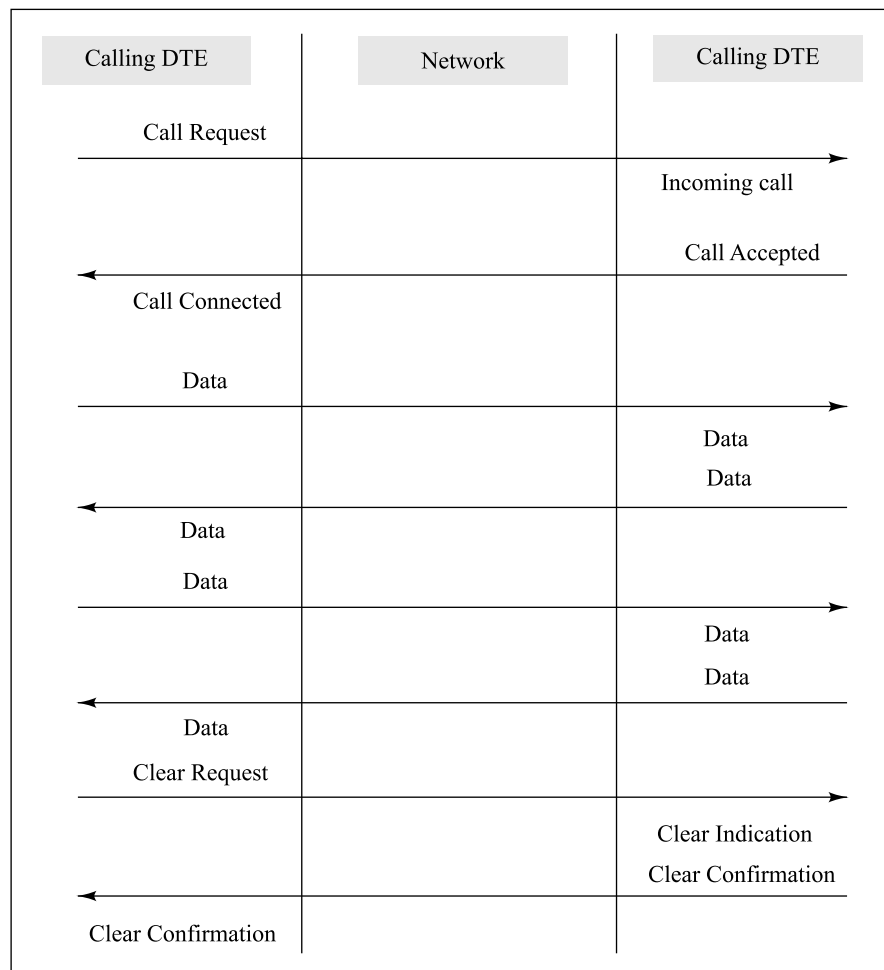



Fig. 11.10 *Flow in a virtual call*

How does a control packet get differentiated from a data packet? Well, the third octet in the X.25 packet header distinguishes not only a control packet from a data packet, but also distinguishes one type of control packet from another. Figure 11.11 shows the different control packet types along with the corresponding values in octet 3.

Sender DTE to DCE	Receiver DCE to DTE	Value in Octet 3
Call Request	Incoming Call	0 0 0 0 1 0 1 1
Call Accepted	Call Connected	0 0 0 0 1 1 1 1
Clear Request	Clear Indication	0 0 0 1 0 0 1 1
Clear Confirmation	Clear Confirmation	0 0 0 1 0 1 1 1

Fig. 11.11  How control packets are identified

Control Packets for Flow Control and Error Control

X.25 protocol implements the principles of Flow Control and Error Control using special control packets not covered above.

1. Flow Control

If one side of a virtual circuit has no data to send, it sends a **Receive Ready (RR)** packet to the other end, indicating that it is ready to receive more packets. The RR packet also contains the next packet number that it expects to receive on this virtual circuit.

The **Receive Not Ready (RNR)** control packet acknowledges the previous packets, but indicates to the sender that the receiver is not in a position to accept any more packets currently. When the sender receives a RNR packet from the receiver, it notes this fact and stops transmission from its side. When the other side is ready to accept packets again, it can send a RR packet to the sender, indicating that it can now start transmission again.

2. Error Control

X.25 protocol achieves error control using the concept of go-back-n. If a receiver senses an error, it sends back a **Reject (REJ)** control packet back to the sender with the packet number. When the sender receives the REJ packet, it realizes that something was wrong in the transmission, and resends the specified packet in error as well as all subsequent packets.

11.4 X.25 OPERATION

Having taken a detailed look at the technical aspects of X.25, let us understand how X.25 works. For this, consider that there are two hosts X and Y among others in a WAN. Suppose host X wants to send a packet to host Y. Further, let us assume that there are nine packet switches named A to I in this WAN, which use X.25 for packet switching. Let us understand how this packet transmission works.

There are two broad-level processes here. The first stage happens at the data link layer, and the other happens at the network layer. At the data link layer, every switch sends an acknowledgement of the data packet back to its immediately previous neighbor. Figure 11.12 shows the idea, which is explained step by step after the figure.

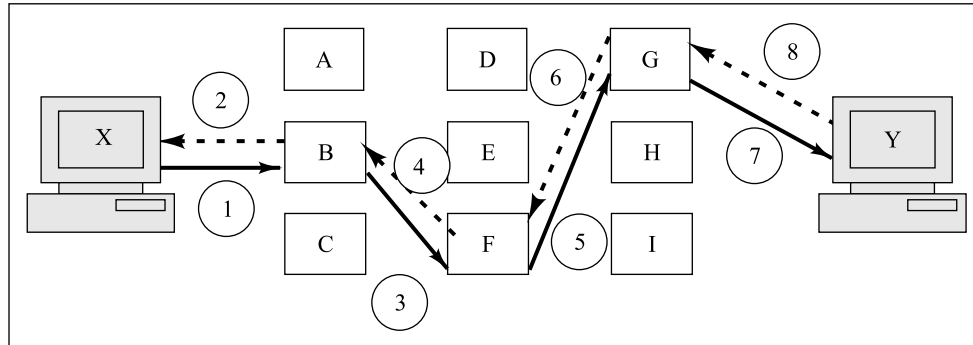


Fig. 11.12 Example of X.25 packet transmission – Data link layer

The second stage happens at the network layer. Here, the final destination (Y) sends an acknowledgment back to its immediately previous switch (G). This switch (G) then sends back an *acknowledgement of the acknowledgement*. This process then continues for all the previous switches. This is shown in Fig. 11.13.

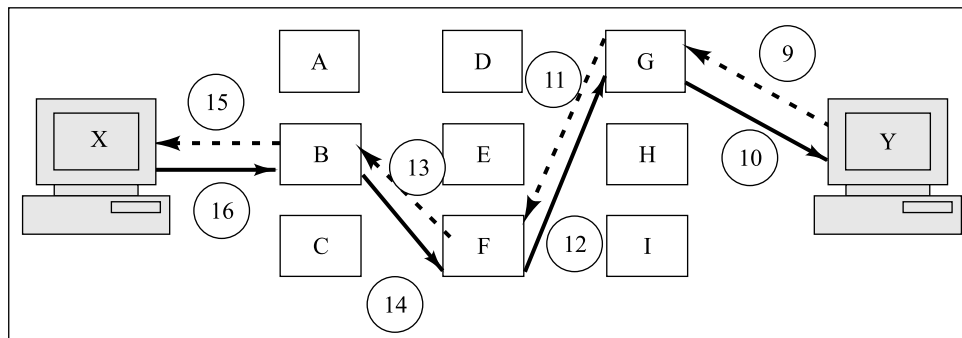


Fig. 11.13 Example of X.25 packet transmission (continued) – Network layer

As shown in the figure, the packet transmission using X.25 between X and Y at the data link layer takes place in the following manner.

1. A virtual circuit is created between X and Y, using the appropriate protocols. In this case, the virtual circuit is over the path X-B-F-G-Y. The sender host X sends a packet to the next switch on the virtual circuit, which is switch B in this case. Host X retains a copy of the packet until it receives an acknowledgement (see Step 2 below) from switch B.
2. Switch B has to now forward the packet received from host X to the next switch, so that it can reach the destination Y. However, before it can do so, the X.25 standard specified that the switch B must send an acknowledgement back to the original sender, i.e., host X. Therefore, switch B sends an acknowledgement for the packet received in Step 1 back to host X after verifying the CRC and ensuring that there is no error. At this juncture, host X destroys the copy of the original packet.
3. In this step, switch B forwards the packet to the next switch on the virtual circuit, which is switch F, in this case. Switch B retains a copy of the packet until it receives an acknowledgement packet from switch F.

4. Switch F now sends an acknowledgement back to switch B. At this juncture, switch B destroys the copy of the original packet.
5. Switch F now forwards the packet to the next switch, which happens to be switch G. Switch F also retains a copy of the packet until it receives an acknowledgement packet from switch G.
6. Switch G now sends an acknowledgement back to switch F. At this juncture, switch F destroys the copy of the original packet.
7. Switch G now forwards the packet to the final destination, i.e., host Y. Switch G also retains a copy of the packet until it receives an acknowledgement packet from host Y.
8. Host Y now sends an acknowledgement back to switch G. At this juncture, switch G destroys the copy of the original packet.

The network layer acknowledgement process works as shown in Fig. 11.13. As shown, this consumes another 8 steps, making the whole data transmission of just one packet a 16-step process (8 steps at the data link layer + 8 steps at the network layer). We shall not describe Steps 9 through 16 in detail—they should be fairly obvious by now. At a broad level, the final destination (Y) sends an acknowledgement back to the previously immediate switch (G), which acknowledges the acknowledgement. This is shown in Steps 9 and 10. This switch then sends back an acknowledgement to its previous switch (F), which acknowledges the acknowledgement (Steps 10 and 11), and so on. The acknowledgement also travels by the same path, as X.25 is based on virtual circuit, where the routing and path determination are done at the beginning of the transmission.

We would realize that it takes a considerable amount of time to send the acknowledgements back for every hop. Also, each host/switch retains a copy of the packet until it receives an acknowledgement from the *next hop* (hence the name *store-and-forward*).

Since computer networks were not as reliable in the early days as they are today, this sort of elaborate acknowledgement mechanism was required in X.25. However, these days, technology has improved immensely, which means that computer networks are very reliable, and transmission errors are rare. Therefore, **Frame Relay** has emerged as an improvement over X.25 in this regard, as we shall study later.

11.5 CCITT X.21

The **X.21** standard (also called **X21** standard) was devised in the 1970s by the International Telecommunications Union. This standard provides a digital signaling interface to the communication systems used by various companies and their carrier systems (i.e., telecom operators). The specifications included the basic interfacing techniques, error control, signaling, call control features, etc. The transmission rate offered by X.21 is between 10 Kbps and 10 Mbps. The X.21 standard was primarily used in Europe and Japan, and not so much in the US.

X.21 is a synchronous protocol. This means that the sender and the receiver must have an agreement before the actual communication between them starts. For this, special signals are used. The ready state for a DTE is indicated by a series of 1 bits on the T pin of the circuit. Similarly, the ready state for a DCE is a continuous transmission of 1 bits on the R pin of the circuit. It uses circuit-switching technology. Hence, connection establishment is essential before the data transfer phase.

SUMMARY

X.25 is an old WAN switching technology. It was developed when communication infrastructure was not up to the mark. To handle transmission errors that were frequent, X.25 incorporates an elaborate flow control and error control mechanisms. Every X.25 switch performs data link layer error checking, and the end nodes perform network layer end-to-end error checking.

X.25 supports the concept of virtual circuits, and uses both PVC and SVC. It defines separate packet formats for carrying data and control information. In data link layer, these packets are based on the LAPB protocol. The logical channel group + logical channel number together identify a logical connection (or virtual circuit) uniquely from all other logical connections.

A number of control packets help in setting up, maintaining and closing X.25 connections. The receiver can indicate to the sender in case of network congestions by sending special packets. Since X.25 provides detailed flow control and error control mechanisms, it is slow. Modern transmission media and switching technology do not require such detailed error handling. Frame Relay is a modern switching technology that removes the drawbacks of X.25.

KEY TERMS AND CONCEPTS

Control packet	Packet layer
Data Communication Equipment (DCE)	Packet Layer Protocol (PLP)
Data packet	Permanent Virtual Circuit (PVC)
Data Transmission Equipment (DTE)	Receiver sequence number
Frame layer	Send sequence number
High-level Data Link Control (HDLC)	Switched Virtual Circuit (SVC)
Link Access Procedure Balanced (LAPB)	X.21
Logical channels	X21
Logical channel groups	X.25
Logical Channel Number (LCN)	

QUESTIONS**True/False**

1. One of the X.25 layers is the data link layer.
2. Transport layer is also supported by X.25.
3. X.25 contains elaborate flow control and error control mechanisms.
4. In the data link layer, X.25 supports the LAPB protocol.
5. X.25 does not provide support for virtual circuits.
6. Logical channel group + Logical Channel Number (LCN) together identify a logical connection.
7. A data packet generally has four octets in the header portion.
8. The P(R) field indicates the number of the next packet expected from the other end of a virtual circuit.
9. Sender sends Clear Request packet is equivalent to beginning a telephone call.
10. If a receiver senses an error, it sends back a REJ packet back to the sender.

Multiple-Choice Questions

1. X.25 is based on the _____ technique.
 - (a) message switching
 - (b) circuit switching
 - (c) virtual circuit
 - (d) datagram approach
2. The end-to-end error checking is implemented in X.25 in the _____ layer.
 - (a) network
 - (b) transport
 - (c) physical
 - (d) data link
3. The adjacent nodes perform error control in X.25 in the _____ layer.
 - (a) network
 - (b) transport
 - (c) physical
 - (d) data link
4. The PLP packet is found in the _____ layer.
 - (a) network
 - (b) transport
 - (c) physical
 - (d) data link
5. PVC is _____.
 - (a) created every time
 - (b) never created
 - (c) created only once
 - (d) None of the above
6. The maximum value for a logical channel group is _____.
 - (a) 16
 - (b) 32
 - (c) 255
 - (d) 1024
7. The maximum value for a logical channel is _____.
 - (a) 16
 - (b) 32
 - (c) 255
 - (d) 1024
8. The sender increments the value of the _____ field every time it sends a packet.
 - (a) P(R)
 - (b) P(S)
 - (c) M
 - (d) LCN
9. The receiver's equivalent of a Call Request packet is _____.
 - (a) Call Accepted
 - (b) Incoming Call
 - (c) Call Connected
 - (d) Call Received
10. The _____ error control mechanism is used in X.25.
 - (a) go-back-n
 - (b) sliding window
 - (c) checksum
 - (d) CRC

Detailed Questions

1. Describe error checking in X.25, relating it to a real-life example.
2. Discuss the X.25 layers at a broad level.
3. Discuss leased line, PVC and SVC.
4. How is the LAPB frame organized?
5. What is the significance of the logical channel group and the logical channel number fields?
6. How are the various control packet types equivalent to messages in a telephone conversation?
7. How is flow control implemented in X.25?
8. What happens in the data link layer and the network layer during a X.25 transmission?
9. Why is elaborate error checking of X.25 considered wasteful these days?

12

Frame Relay and Congestion Control

12.0 INTRODUCTION

We have discussed the X.25 protocol in great detail previously. Although X.25 is still a widely used protocol, it is generally perceived as an obsolete protocol. **Frame Relay** has become quite popular in the last few years, instead. Like X.25, Frame Relay is also a WAN protocol. However, it is vastly different from X.25 in many respects, as we shall see. Frame Relay is perceived as a fast packet switching technology. Users were demanding faster data rates, capabilities of handling bursty transmissions (or bandwidth-on-demand) and lower costs by way of lower overheads.

As we noted, X.25 operates in the bottommost three layers of the OSI model, namely, the physical layer, the data link layer and the network layer. Frame Relay, however, does not operate in the network layer, and restricts its operations only to the bottommost two layers, namely, the physical layer and the data link layer, as shown in Fig. 12.1.

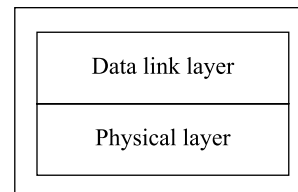


Fig. 12.1 *Frame Relay layers*

12.1 THE NEED FOR FRAME RELAY

Frame Relay gained popularity due to a number of reasons. Let us discuss some of the chief reasons that are attributed to the dramatic increase in the demand for Frame Relay services.

12.1.1 Higher Data Rates

Earlier, many organizations used to make use of the WAN technology such as leased lines or X.25 protocol to connect their geographically dispersed computers. These days most organizations have LAN connectivity. These LANs are geographically apart from each other, and organizations always want to be able to connect them to each other to form a WAN. This WAN connectivity had to be very fast to match the LAN speeds, so that it did not become the *weakest link*, and therefore, a bottleneck. This can be achieved in two main ways, namely, by using T-lines (also called leased lines) or by using Frame Relay. What are the pros and cons of these approaches?

T-lines offer high data rates. For instance, a T-1 line offers a data rate of 1.544 Mbps. However, a T-line is point to point. This means that if we want to connect multiple LANs to each other, it would necessitate a point-to-point T-line between every LAN pair. Thus, to connect six LANs, we would need $[6 \times (6 - 1) / 2]$ i.e. $[6 \times 5 / 2]$ i.e., 15 T-lines (in general, to connect n LANs, we would need $n \times (n - 1) / 2$ T-lines). This is shown in Fig. 12.2.

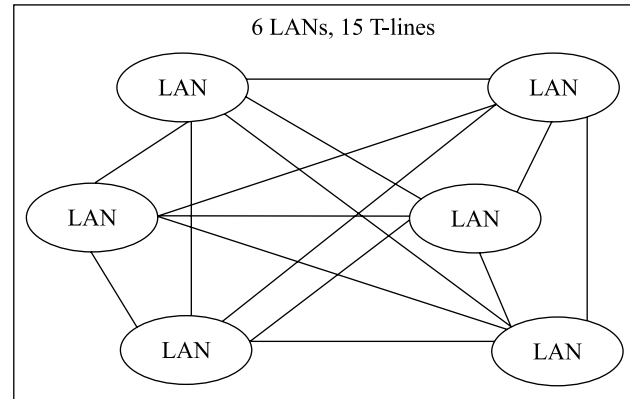


Fig. 12.2 Six LANs connected point-to-point using 15 T-lines

Also, once a T-line is obtained from the carrier, even if it is not used, one has to pay for it. Thus, the payment is not on the actual usage basis. Finally, the leased T-lines may be fully utilized some times, and little used at other times, depending on the usage. However, there is no scope for optimization in terms of increasing the data rates at peak hours, and reducing them when the transmission requirements are low.

In contrast, Frame Relay services will need only six T-lines to connect six LANs, as shown in Fig. 12.3. The Frame Relay network is shown as a cloud, which we shall explain in subsequent sections.

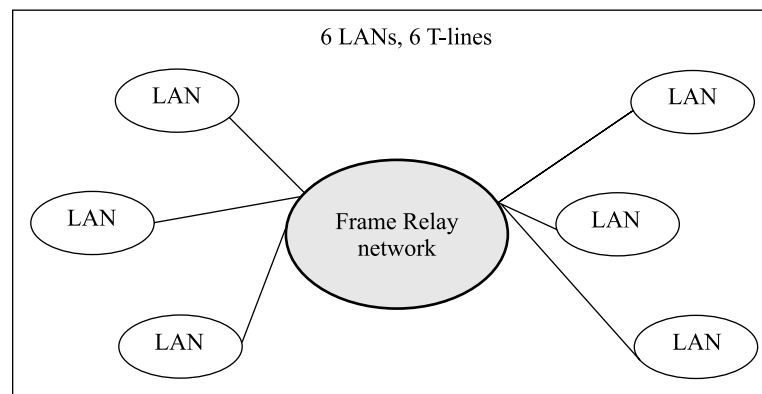


Fig. 12.3 Six LANs connected using a Frame Relay network

Interestingly, although the initial Frame Relay networks were built on T-1 data rate capabilities (i.e., 1.544 Mbps), they are now capable of handling up to T-3 transmissions (i.e., 44.376 Mbps).

12.1.2 Bursty Data (Bandwidth-on-demand)

A T-line offers constant data rate. That is, the T-line is leased from a carrier on the assumption that the transmission rate requirements would be constant all the time. In case of data transmission, this assumption is particularly incorrect. A user might type a few keystrokes, then go for a cup of coffee, come back after half an hour, download a file containing three images, requiring a very large bandwidth, and so on. These kind of unpredictable requirements are called **bursty data** requirements, or **bandwidth-on-demand**, i.e., as and when a user requires it.

For instance, assume that a user has access to a T-1 line (i.e., data rate of 1.544 Mbps). However, as illustrated earlier, the user might send data at 5 Mbps for 3 seconds, 0 Mbps (i.e., no data) for 2 seconds, and then again 10.44 Mbps for 5 seconds. Note that in 10 seconds, the user has transmitted 15.44 Megabit data, that is, the bandwidth requirement over 10 seconds is 1.5444 Mbps. The trouble is that this requirement is always fluctuating. Suddenly, the user wants more bandwidth, and the next moment, the user simply ignores the available bandwidth. A constant data rate offered by a T-1 line is of no use here.

To tackle this problem, Frame Relay supports a minimum average data rate, which can be exceeded whenever desired.

12.1.3 Lower Overheads

We have discussed the elaborate flow control and error control (error-checking and correction) mechanisms in X.25 networks. The X.25 protocol was devised at a time when transmission media, switches and routers were not very reliable. As we know, X.25 provides error control in both the data link layer and the network layer.

- Data link layer – Before a X.25 switch forwards a packet to its next switch, it keeps a copy of the packet. The next switch performs various kinds of checks to ensure that the packet arrived error-free, and sends back an acknowledgement to the original switch. Only after it receives an acknowledgement that this switch destroys the copy of the packet. This continues through the complete journey of the packet from the source to the final destination.
- Network layer – In addition, the network layer of the original source keeps a copy of the packet until it receives an acknowledgement from the network layer of the final destination.

This is shown in Figs 12.4 and 12.5. Figure 12.4 shows the data link layer operations, whereas Fig. 12.5 shows the network layer operations. Here, a source computer X wants to send a packet to a destination computer Y. A through I indicate the X.25 switches that route the packet from computer X to Y. As we can see, just to send one packet from X to Y, 16 steps are required—8 each in the data link layer (Fig. 12.4) and network layer (Fig. 12.5).

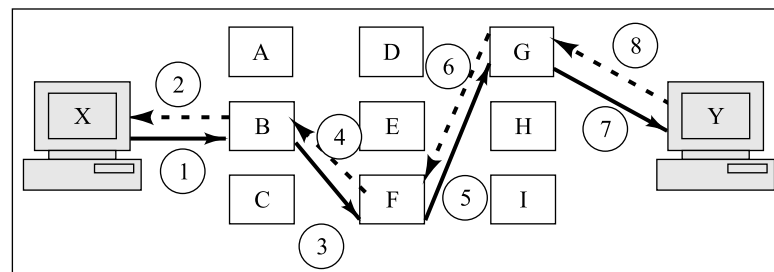


Fig. 12.4 Data link layer error control in X.25

First, let us observe the data transmission and acknowledgement mechanism at the data link layer. As we can see, computer X sends a data packet to switch B, which sends back an acknowledgement to computer X. Switch B forwards the packet to switch F, which sends back an acknowledgement to switch B, and so on. In each case, the computer/switch keeps a copy of the packet, forwards the packet to the next hop, and waits for an acknowledgement before destroying its copy of the packet.

Figure 12.5 shows the network layer acknowledgements. Computer Y sends an acknowledgement back to switch G (Step 9). Switch G *acknowledges the acknowledgement* (Step 10). Switch G then sends an acknowledgement back to switch F (Step 11), which is acknowledged by switch F (Step 12), and so on.

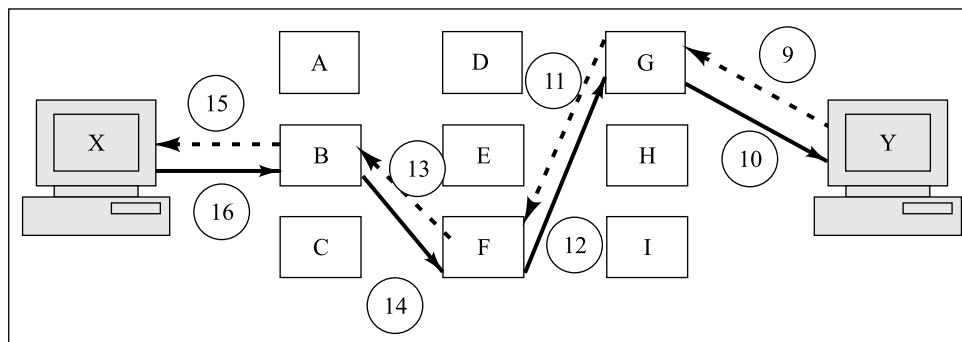


Fig. 12.5 Network layer error control in X.25

However, with rapid improvement in transmission hardware technology in the last few decades, elaborate flow control and error control can be highly undesirable these days. The WAN need not spend too much of time and bandwidth figuring out if the transmission was error-free, but instead, can concentrate on achieving higher data rates. Additionally, the fact that each node must keep a copy of the frame in its storage during the time it waits for an acknowledgement causes another traffic overhead and further reduction in overall data transmission rates.

These days, fiber optics cables are generally used for connecting LANs to WANs. Fiber optic cables are far less susceptible to noise as compared to copper wires. Thus, the chances of transmission errors have gone down dramatically. The elaborate error control mechanism employed by X.25 is thus not only undesired but is also a hindrance.

Frame Relay takes a different approach than X.25. It does not perform error checking at the data link layer. Instead, it leaves it to the network and transport layers. This means that it is out of the scope of Frame Relay (because Frame Relay itself operates only at the bottommost two layers, namely, the physical layer and the data link layer).

Let us consider the same example that we used for describing X.25 transmissions. We have two hosts X and Y. Host X wants to send a packet to host Y. There are 9 switches named A to I on the path between the hosts X and Y. Let us now understand how the packet transmission would work in the case of Frame Relay, as shown in Fig. 12.6 and explained thereafter.

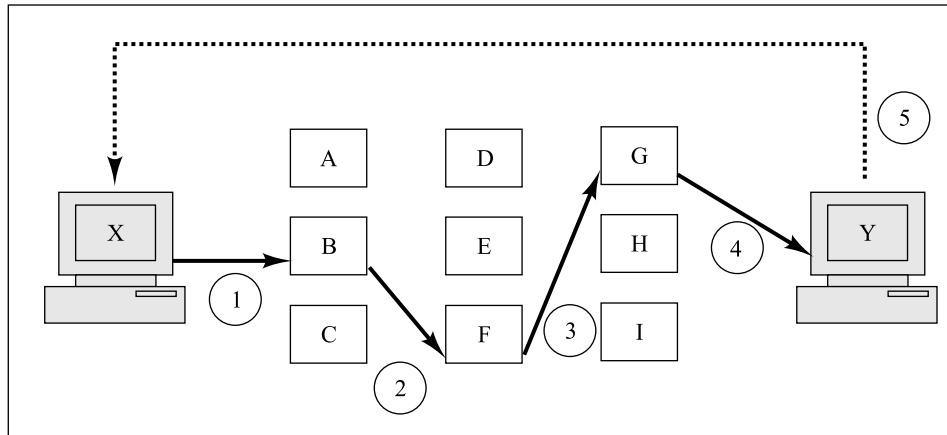


Fig. 12.6 Frame Relay switching example

The steps are as given below.

1. A virtual circuit is established between hosts X and Y. Let us assume that the virtual circuit takes the path X-B-F-G-Y. Host X sends the packet to the next switch in the virtual circuit. In this case, that is switch B.
2. Switch B neither carries out any error checking (using CRC, etc.) nor does it send an acknowledgement back to host X, unlike what happens in case of X.25. Here, switch B simply forward the packet to the *next hop*. In this case, it is switch F.
3. Similarly, switch F forwards the packet to its *next hop*, i.e., switch G.
4. Switch G forwards the packet to the *next hop*, which is the end destination (i.e., host Y).
5. Host Y now checks for any errors and sends an acknowledgement or a request for retransmission back to host X. This acknowledgement also travels from host Y to host X via the same route. Therefore, this route is not shown in the figure.

As we can see, the main difference between X.25 switching and Frame Relay switching is the acknowledgment mechanism. In case of X.25, it is very elaborate in the sense that every host/switch sends an acknowledgement back to its immediately previous hop. However, in the case of Frame Relay, a host/switch does not expect, and therefore, does not wait for an acknowledgement from its previous hop. Instead, it assumes that it would be delivered correctly to the next hop. Also, after a host/switch sends a packet to its *next hop*, it forgets about the transmission and does not retain a copy of the packet unlike what happens in case of X.25. This frees valuable memory resources of the hosts and switches.

Thus, the intermediate error checking is eliminated in case of Frame Relay. It is left only to the end points of the transmission. This means that host X would start a timer as soon as it forwards the packet to switch B. It would retain a copy of the packet. It would now expect to receive an acknowledgement for the packet from host Y within a specific time limit set in the timer. If the acknowledgement arrives before the timer expires, it means that the transmission was successful. Therefore, host X would remove the copy of the packet that it had retained. However, if the timer expires, it means that the acknowledgement did not arrive in the expected interval. Therefore, host X would retransmit the packet to switch B. It is needless to say that if the acknowledgement

sent by node Y was delayed in reaching node X causing this retransmission, node Y will have to discard this duplicate copy.

This explains why X.25 transmission speed is just 64 Kbps, whereas Frame Relay transmission speed is as high as 2 Mbps. From the above discussion, we can describe the Frame Relay philosophy as ‘*Given the quality of the transmission system, stop all point-to-point error detection and correction, and let the end nodes worry about it*’.

Let us summarize these concepts by distinguishing between X.25 and Frame Relay, as shown in Fig. 12.7.

Characteristic	X.25	Frame Relay
Virtual circuit establishment	Network layer	Data link layer
Flow control, error control	Data link layer as well as network layer	Not applicable
Transmission rate	Fixed	Bursty
Multiplexing	Network layer	Data link layer

Fig. 12.7 X.25 versus Frame Relay

12.2 HOW FRAME RELAY WORKS

12.2.1 Introduction

The Frame Relay technology can be used as a low-cost, high-speed communications WAN infrastructure for connecting LANs that require support for bursty traffic. Like X.25, Frame Relay provides support for Permanent Virtual Circuits (PVC) as well as Switched Virtual Circuits (SVC). An organization can decide to go for either of these, depending on the data transmission requirements and the amount to be invested in setting this infrastructure up.

Conceptually, a Frame Relay network can be depicted as shown in Fig. 12.8.

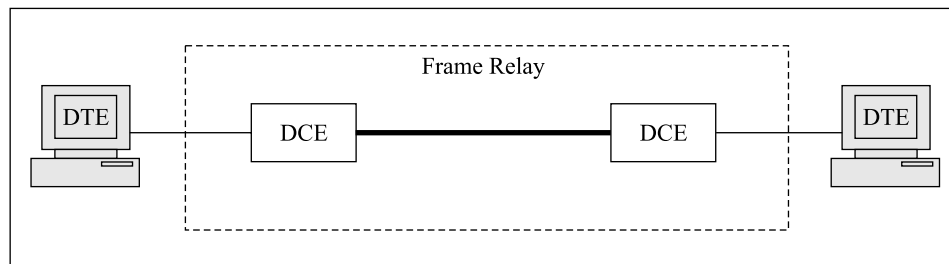


Fig. 12.8 Frame Relay – A conceptual view

As we have mentioned, the Frame Relay philosophy is also based on virtual circuits. However, virtual circuits are created and used in the data link layer, and not in the network layer (unlike X.25). In Frame Relay, a virtual circuit is identified by a number, called **Data Link Connection Identifier (DLCI)**. When a virtual circuit is established, a DTE is provided with a DLCI that it can use for communication with the remote DTE. This is shown in Fig. 12.9.

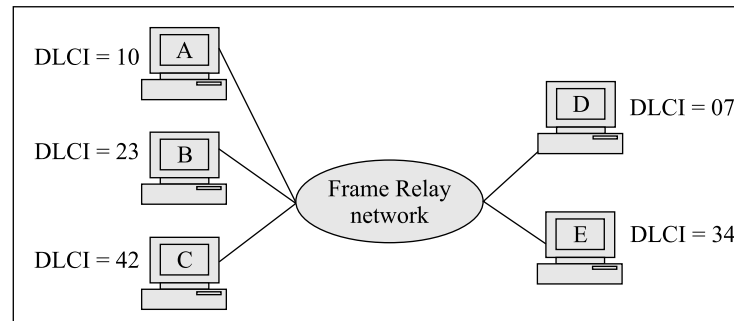


Fig. 12.9 *Frame Relay network and DLCI*

How are DLCI values used in actual transmission? Suppose a virtual circuit has been established between computers C and D using the DLCI values as shown in the figure. When computer C wants to send a packet to computer D, it uses DLCI = 42. Similarly, if computer D wants to send a packet to computer C, it uses DLCI = 07.

The assignment of DLCI can be permanent (in case of PVC) or on a per-connection basis (SVC). Frame Relay was initially supported only PVC connections. However, it now supports the SVC type as well. In case of PVC, the virtual circuit is already set up for Frame Relay. But in the case of SVC, Frame Relay needs the services of another higher-level protocol, at the network layer, to establish a SVC (since Frame Relay operates only at the physical layer and the data link layer).

12.2.2 Frame Relay Switching

Frame Relay technology uses the concept of switching to route packets from the source to the destination. As we have noted, there are one or more switches (specialized computers that are able to forward packets based on the addresses they contain) between the source and the destination.

Figure 12.10 shows a conceptual Frame Relay network. Here, we have four Frame Relay switches that connect multiple computers together to form a Frame Relay network. Note that we have shown very few computers connecting to the switches, just to keep things simple. In reality, the number of switches would be far less than the number of computers that they connect together.

How does a Frame Relay switch forward/route packets? Where is the intelligence to be able to do this? In order to be able to route packets from the source to the destination, via zero or more other Frame Relay switches, a Frame Relay switch maintains a table. This table tells a Frame Relay switch how to forward packets.

Figure 12.11 shows another Frame Relay network to illustrate the concept. Here, we have shown a Frame Relay switch that has six incoming interfaces and six outgoing interfaces. For simplicity, we have shown just one incoming interface (number 2) and two outgoing interfaces (number 10 and 12) as active. As we can see, the incoming interface (number 2) has received two packets, with DLCI values = 78 and 121. The Frame Relay switch maintains a table as shown in the second half of the figure to determine that a packet arriving at interface 2 with DLCI = 78 should be routed to interface 10 with DLCI = 37. Similarly, the Frame Relay switch also determines that the second packet with DLCI = 121, which has arrived at interface 2 should be routed to interface 12 with DLCI = 147.

This is how a Frame Relay switch effects forwarding of packets from incoming interfaces to the appropriate outgoing interfaces. Since a Frame Relay switch can be connected to another Frame Relay switch, the DLCI number is required even on the outgoing path, as it becomes the incoming DLCI number for that next switch. That next switch requires it for its routing decisions.

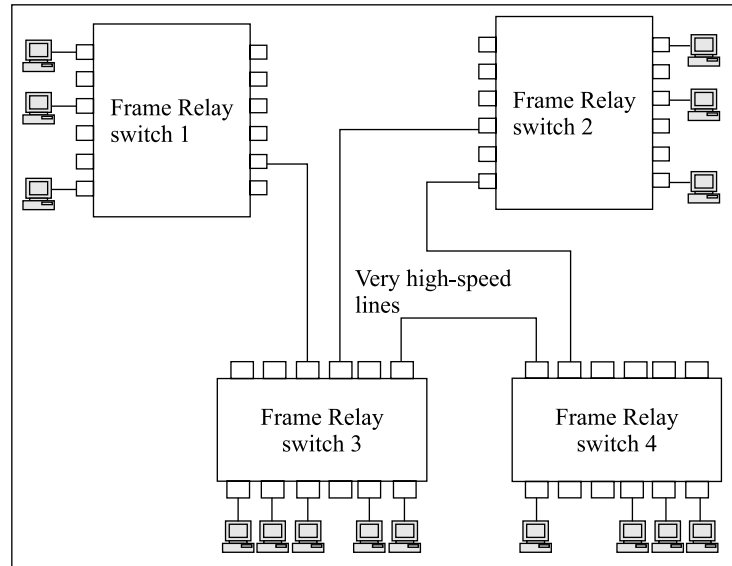


Fig. 12.10 Frame Relay switches

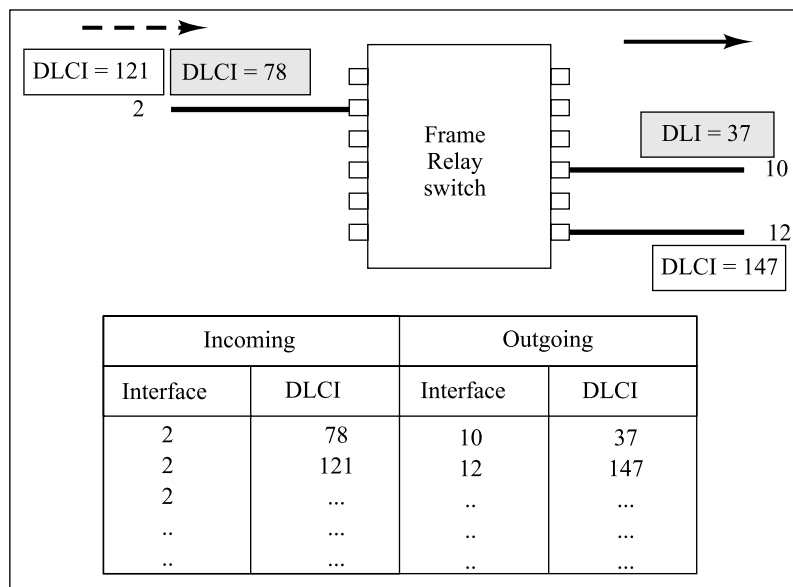


Fig. 12.11 Frame Relay routing concept

12.3 FRAME RELAY FRAME FORMAT

The Frame Relay protocol uses a physical frame format that is similar to other data link layer protocol frame formats. However, the most significant aspect of the frame format in case of Frame Relay is the absence of a control field. Thus, unlike X.25, Frame Relay does not have separate frame types (data frames and control frames). Since there are no control frames, there are no frame sequence numbers. Consequently, there is no flow control or error control mechanism.

The Frame Relay frame format is shown in Fig. 12.12.

Flag	Address	Information	FCS	Flag
1 Octet	2-4 Octets	Variable size	2 Octets	1 Octet

Fig. 12.12 Frame Relay frame format

The *Flags* and *Frame Check Sequence (FCS)* fields are not special to Frame Relay, and need not be discussed. The *Information* field contains the actual data that is being sent.

The *Address* field can be of 2 (default), 3, or 4 octets. It contains many subfields, as shown in Fig. 12.13, and discussed thereafter.

DLCI	C/R	EA	DLCI	FECN	BECN	DE	EA
6 bits	1 bit	1 bit	4 bits	1 bit	1 bit	1 bit	1 bit

Fig. 12.13 Frame Relay address format

Let us discuss the subfields of the address portion now.

- As we can imagine, the address field contains the **Data Link Connection Identifier (DLCI)** that is 10, 17 or 24 bits long. As we know, the DLCI field is equivalent to the X.25 virtual circuit identifier. That is, a DLCI allows multiple logical Frame Relay connections to be multiplexed over a single physical channel. The DLCI is split up into two portions, viz., a 6-bit portion and a 4-bit portion.
- The **Command/Response (C/R)** flag is not used by Frame Relay, and is provided for upper layers to identify a frame as either a command or a response.
- The **Extended Address (EA)** subfield indicates whether or not the current byte is the last byte of the address field. If it is 1, it signifies the end of address; else it indicates more address bits are to follow.
- The **Forward Explicit Congestion Notification (FECN)** and **Backward Explicit Congestion Notification (BECN)** bits are used to deal with network congestion problems, as discussed subsequently.
- The **Discard Eligibility (DE)** bit indicates the frame priority. In case of network overloads, the Frame Relay network can discard frames that have $DE = 0$ (which indicates a lower priority frame). Frames with $DE = 1$ are high priority frames, and should be discarded only as a last resort. The sender or a Frame Relay switch can set the DE bit.

12.4 CONGESTION CONTROL

12.4.1 Introduction

In simple terms, a network faces the problem of **congestion** if the users of the network send data at a rate that is faster than the network can handle. Every switch on a network has a finite processing speed and memory buffer to hold packets temporarily before they can be forwarded. If these resources are too small as compared to the incoming packets, then the network would become quite slow. The concept of flow control is used to avoid or deal with congestion. This achieves **congestion control**.

In X.25, flow control is performed both at the data link layer (node-to-node) and network layer (end-to-end). However, as we know, network layer is absent in case of Frame Relay. Therefore, there is no concept of end-to-end flow control in case of Frame Relay. Additionally, the data link layer in Frame Relay performs no flow control, either. Therefore, there is a high amount of likelihood of congestion in case of Frame Relay networks. For this, the principle of **congestion avoidance** is used. In order to achieve this, two bits in the Frame Relay frame are used, which warn the source or the destination of congestion possibilities.

12.4.2 BECN

The **Backward Explicit Congestion Notification (BECN)** bit informs the sender of congestion in a Frame Relay network. One question might arise here. Since the packets originate *from* the sender, how can we send information about network congestion to the sender? This can be achieved in the following two ways:

1. Either the receiver (i.e., Y in this case) can send the BECN bit as part of one of the acknowledgements.
2. One of the Frame Relay switches can send a special packet containing BECN to the sender. This is illustrated in Fig. 12.14.

In either case, the sender reduces its transmission rate in response to a BECN packet.

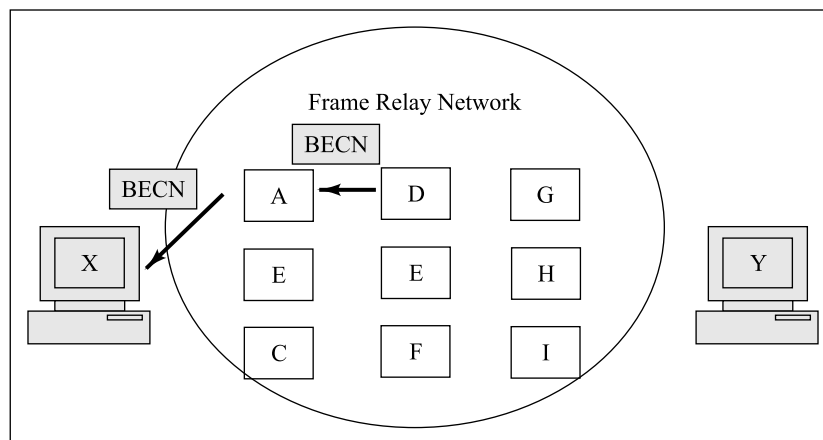


Fig. 12.14 BECN

12.4.3 FECN

In contrast to BECN, which is used to warn the sender (i.e., X in this case) of network congestion, a **Forward Explicit Congestion Notification (FECN)** bit is used to warn the receiver (i.e., Y in this case) of network congestion, as shown in Fig. 12.15. Switch D sets this bit in a frame that is sent through it to the receiver. Alternatively, it can send a special additional frame with the BECN bit set.

One pertinent question is since the receiver is receiving frames from the Frame Relay network, can it do anything to reduce the congestion? Interestingly, we can observe that the receiver can, in fact, handle this by delaying acknowledgements that are to be sent to the sender at the network layer. Of course, this mechanism is outside of the scope of Frame Relay. However, Frame Relay simply informs the receiver of network congestion. It is up to the receiver to use an appropriate mechanism at the network layer to indicate to the sender that it cannot accept any more data at this speed. This would help reduce the congestion.

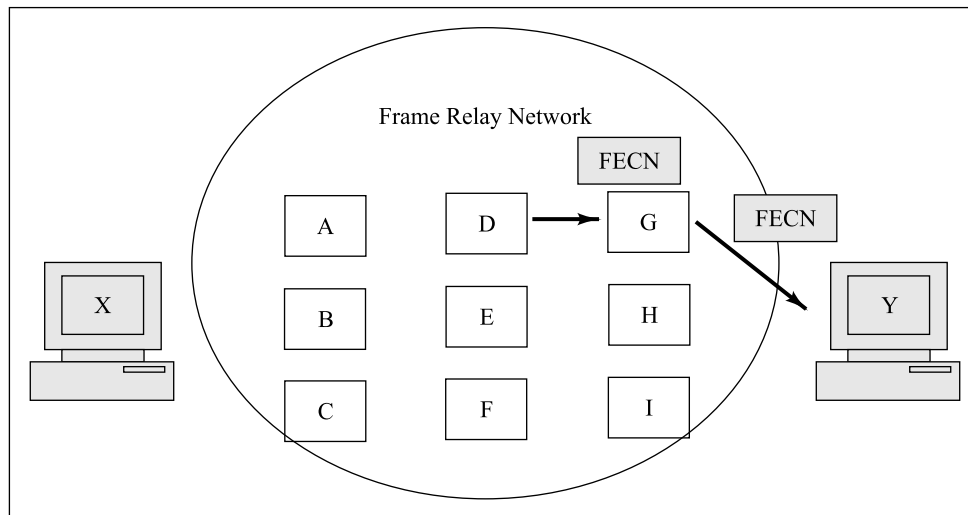


Fig. 12.15 **FECN**

12.5 CONGESTION CONTROL ALGORITHMS

12.5.1 Leaky Bucket Algorithm

The **leaky bucket algorithm** is used in various data transmission techniques to ensure conformance to expected levels of bandwidth and throughput. Whenever there are variations in the amount of data being transmitted, causing bursts of traffic, this needs to be monitored and controlled. Leaky bucket algorithm helps in this respect.

The name of the algorithm comes from the idea of a leaky bucket that has a hole at its bottom. As a result, water from the bucket keeps going out at a fixed rate until it is completely empty. Of course, more water can be added to the bucket, but if too much of water is added, it would cause an overflow as the capacity of the bucket is exhausted. This is shown in Fig. 12.16.

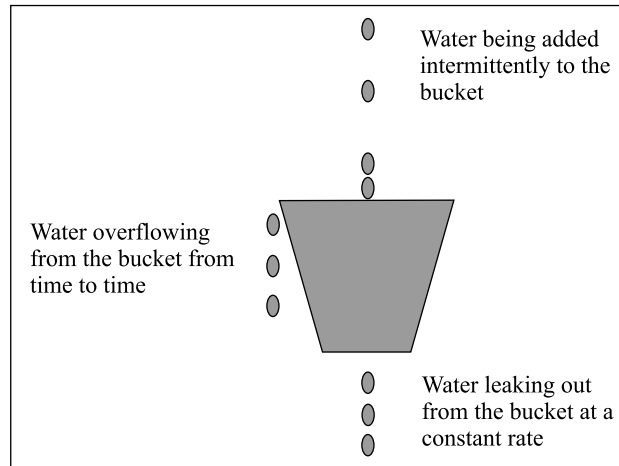


Fig. 12.16 *Leaky bucket algorithm*

The way this concept is used is as follows.

Every user on a network has a counter associated with her/him. Whenever the user sends a packet, the value of the counter is incremented by one. The value of the counter is also decreased automatically at a fixed pre-determined frequency. However, if the value of the counter exceeds a predefined threshold value, then it is considered that the user is sending too many packets, and that packet is discarded. To ensure that this scheme does justice to the connected hosts, the hosts can decide the threshold value of counter decrement. This is based on the average bandwidth made available to that user. Here, the counter serves the purpose of the bucket of our earlier example. The counter is a meter that puts a check on the rate of flow of packets. Thus we can state the broad principles of the leaky bucket algorithm as

1. Each virtual connection or user has a bucket of fixed capacity in the form of a counter. It leaks at a fixed, predefined rate.
2. If the bucket is empty, obviously it cannot leak anymore. Hence, it stops leaking.
3. It should be possible to add a specific amount of water to the bucket (i.e., packets to the connection).
4. If adding water to the bucket (i.e., adding packets to the connection) causes the bucket to overflow, the packet is considered not to conform to the algorithm. Hence, the water in the bucket is left unchanged.

The leaky bucket algorithm can be used as a meter or a counter in network traffic management. For example, in an ATM network, we have the concept of a virtual channel or virtual path. Here, we can use the leaky bucket algorithm to measure the burstiness and bandwidth of these channels against the expected limits that are set initially. The channels that do not conform to the set limits can be penalized.

12.5.2 Token Bucket Algorithm

The **token bucket algorithm** is also used in various data transmission techniques to ensure conformance to expected rates of data transmission. In other words, it helps monitor the amount of

data that can be injected into a network. Data bursts, i.e., sudden packets of data are allowed. Here, the concept of a **token bucket** is used. This is a conceptual or abstract bucket.

A token bucket contains tokens, another abstract concept. One token corresponds to one or more bytes of data. How many tokens are needed to transmit how many bytes is decided beforehand. When a packet is sent, appropriate number of tokens is removed from the bucket. When the bucket contains tokens, traffic is allowed to flow. However, if the bucket is empty, which means that it does not have any tokens, then traffic is not allowed. Hence, in other words, traffic is allowed up to the peak burst rate, provided there are tokens left in the bucket. We can state the broad principles of the token bucket algorithm as follows:

1. After every $1/r$ seconds, a new token is added to the token bucket.
2. The capacity of the bucket to hold tokens is equal to b . After this, when the bucket is full and if a new token arrives, then it is discarded.
3. When a packet containing n bytes arrives in the network, n tokens are removed from the bucket. The packet is then sent to the network.
4. However, if less than n tokens are available in the bucket, no tokens get removed from the bucket. The packet is considered to violate the rules of the algorithm, and is not delivered.

12.6 TRAFFIC CONTROL

Although taking action to deal with congestion control is out of the scope of the Frame Relay protocols at the physical layer and the data link layer, a Frame Relay network must know when to set the BECN and FECN bits in the first place. On what basis can a Frame Relay switch decide to set these bits so as to indicate to the sender or the receiver of congestion conditions? For this, four attributes are used, which are **access rate**, **committed burst size**, **Committed Information Rate (CIR)** and **excess burst size**. These are summarized in Fig. 12.17.

Attribute	Explanation
Access rate	A predefined maximum access rate (in bits per second) allows a user to utilize data rate less than or up to that value.
Committed burst size	The committed burst size is the amount of data that a user can send in a given time period. This is guaranteed by a Frame Relay installation. For example, we can have a committed burst size of 600 Kb in 5 seconds. This means that the user can send up to 600 Kb in a 5-second time period. Note that this is not data rate per second, but per group of seconds. Thus, during a 5-second period, the user may send 300 Kb in the first second, then may not send anything in the next two seconds, and send another 300 Kb in the remaining two seconds, totaling 600 Kb in five seconds.
Committed Information Rate (CIR)	The Committed Information Rate (CIR) is conceptually similar to the committed burst rate, with a difference. The CIR is always expressed with respect to one second, not a group of seconds. Thus, CIR is expressed in bps.
Excess burst size	This is the additional data rate per group of seconds that the Frame Relay service guarantees, on top of the committed burst size. Note that this is possible only if there is no network congestion.

Fig. 12.17  *Traffic control measures in Frame Relay*

12.7 FRAME RELAY ASSEMBLER/DISASSEMBLER (FRAD)

Many times, a Frame Relay network can be a part of a bigger overall network that is not in conformance with Frame Relay. That is, Frame Relay networks can co-exist with other networks that use different protocols such as X.25, PPP, etc. To handle these disparate protocols, the Frame Relay protocol defines a device called **Frame Relay Assembler/Disassembler (FRAD)**.

In simple terms, a FRAD works conceptually similar to a dial-up modem. We know that a modem accepts digital pulses from a sending computer and transforms them into analog signals so that they can be carried over the analog telephone network. At the receiving computer, the modem performs the reverse task of converting analog signals back into digital data that a computer can understand and accept.

A FRAD can similarly accept frames from networks that implement different protocols such as X.25, ATM, PPP, etc., and transform them into Frame Relay frames at the incoming end. At the outgoing end, it can accept frames in the Frame Relay format, and convert them into frames of these protocols. This is shown in Fig. 12.18. A FRAD can be implemented as a separate hardware device, or it can be made a part of a Frame Relay switch. A detailed discussion of FRAD is outside of the scope of this text.

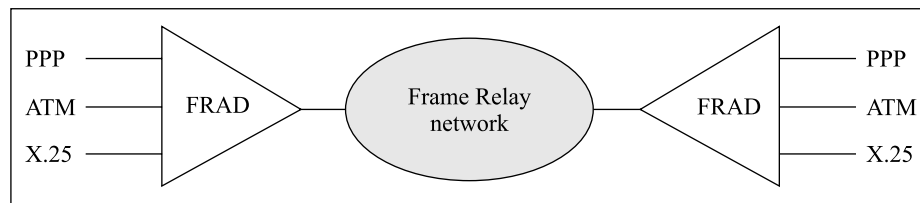


Fig. 12.18 *Frame Relay Assembler/Disassembler (FRAD)*

12.8 OTHER FEATURES

Frame Relay protocol can be used to carry voice traffic. For this, Pulse Code Modulation (PCM) is used to first digitize the analog voice signals, and data compression techniques are used to reduce the data size. When Frame Relay carries voice data, it is called **Voice Over Frame Relay (VOFR)**. Clearly, voice traffic carried by VOFR does not offer the best possible service—it is best carried using circuit switching. However, Frame Relay has made at least a provision in case such a service is required.

Another recent development in the context of Frame Relay is the **Local Management Protocol (LMP)**. Since Frame Relay was initially based solely on PVC functionality, there was not much to worry about in terms of handling real-time problems as in case of SVC. Since SVC is now supported, LMP is required. A detailed discussion of LMP is outside of the scope of this text.

SUMMARY

Frame Relay is a modern switching technology. Frame Relay can be considered as the improved version of the old X.25 protocol. The X.25 protocol contains elaborate flow control and error control mechanisms, which have become unnecessary with the ever-improving transmission and switching

technologies. Frame Relay does not provide any flow control or error control, and works only in the physical layer and the data link layer of the OSI model.

Initially developed on the T-1 line framework (data rate of 1.544 Mbps), Frame Relay now supports T-3 lines (44.376 Mbps). Frame Relay works well with the concept of bursty data. For switching and multiplexing, Frame Relay uses the concept of virtual circuits, and supports both PVC and SVC mechanisms. The Data Link Connection Identifier (DLCI) field identifies a virtual circuit.

Frame Relay switches maintain tables that help them route frames to the next hop. The DLCI values are used to identify the incoming and outgoing virtual circuits. Frame Relay defines a frame format at the data link layer similar to other data link layer protocols and does not contain any control fields. It uses two special bits for congestion control, which send a congestion warning either to the sender or the receiver. To know when to set these special bits, Frame Relay performs traffic control using four data rates that are predefined for a given Frame Relay installation.

A special device called Frame Relay Assembler/Disassembler (FRAD) allows a Frame Relay network to interoperate with other networks such as X.25, ATM, etc. FRAD transforms frames from Frame Relay to the respective protocol frame formats, and vice versa.

KEY TERMS AND CONCEPTS

Access rate	Data Link Connection Identifier (DLCI)
Backward Explicit Congestion Notification (BECN)	Excess burst size
Bandwidth-on-demand	Frame Relay
Bursty data	Frame Relay Assembler/Disassembler (FRAD)
Committed burst size	Forward Explicit Congestion Notification (FECN)
Committed Information Rate (CIR)	Leaky bucket algorithm
Congestion	Local Management Protocol (LMI)
Congestion avoidance	Token bucket algorithm
Congestion control	Voice Over Frame Relay (VOFR)

QUESTIONS

True/False

1. Frame Relay operates in the bottommost three layers of the OSI model.
2. A T-line is point-to-point.
3. Bursty data is the same thing as bandwidth-on-demand.
4. Frame Relay does not have elaborate error checking.
5. X.25 offers faster data rates than Frame Relay.
6. Frame Relay provides for detailed flow control.
7. Frame Relay does not support the SVC concept.
8. DLCI is a global identifier.
9. Frame Relay has data frames as well as control frames.
10. The C/R flag is not used by Frame Relay.

11. The BECN flag is the same as the FECN flag in terms of its basic intention.
12. The CIR is always expressed with respect to one second, not a group of seconds.
13. FRAD is useful for protocol conversion.

Multiple-Choice Questions

1. Frame Relay works in the physical and _____ layers.
 - (a) network
 - (b) data link
 - (c) transport
 - (d) application
2. Frame Relay services will need _____ T-lines to connect n LANs.
 - (a) $n \times (n - 1)$
 - (b) $2n$
 - (c) $n / 2$
 - (d) n
3. Frame Relay data rate is _____ X.25.
 - (a) faster than
 - (b) slower than
 - (c) same as
 - (d) slower than or equal to
4. Multiplexing in X.25 is done in the _____.
 - (a) physical layer
 - (b) data link layer
 - (c) network layer
 - (d) All of the above
5. Multiplexing in Frame Relay is done in the _____.
 - (a) physical layer
 - (b) data link layer
 - (c) network layer
 - (d) All of the above
6. In Frame Relay, a virtual circuit is identified by a number called _____.
 - (a) circuit number
 - (b) DLCI
 - (c) connection id
 - (d) circuit id
7. Frame Relay maintains _____ for routing frames.
 - (a) arrays
 - (b) databases
 - (c) messages
 - (d) tables
8. The address field in Frame Relay contains _____ octates.
 - (a) 3–6
 - (b) 1–3
 - (c) 2–4
 - (d) 1–4
9. The BECN flag is for the _____ to know of network congestion problems.
 - (a) sender
 - (b) receiver
 - (c) network
 - (d) switch
10. The FECN flag is for the _____ to know of network congestion problems.
 - (a) sender
 - (b) receiver
 - (c) network
 - (d) switch
11. The _____ is the amount of data that a user can send in a given time period.
 - (a) access rate
 - (b) committed burst size
 - (c) committed information rate
 - (d) excess burst size
12. _____ is the non-committed, additional data rate.
 - (a) access rate
 - (b) committed burst size
 - (c) committed information rate
 - (d) excess burst size
13. Frame Relay uses _____ for protocol conversions.
 - (a) FRAD
 - (b) packets
 - (c) frames
 - (d) LAPB

Detailed Questions



1. Why is Frame Relay faster than X.25?
2. What is the reason for less error checking in Frame Relay as compared to X.25?
3. How is Frame Relay different from a simple T-line?
4. Explain the concept of bursty data with an example.
5. What is DLCI? How is it useful?
6. How do Frame Relay switches perform routing of frames?
7. What is congestion control? How is it achieved in Frame Relay?
8. Explain the DE bit.
9. What are the four key attributes of traffic control in Frame Relay? How are they inter-related?
10. What is the significance of FRAD?

13 Asynchronous Transfer Mode (ATM)

13.0 INTRODUCTION

We have discussed the three key long-distance communication technologies, namely ISDN (also called *Narrowband ISDN* or *N-ISDN*), X.25 and Frame Relay. Just as X.25 and ISDN were the drivers behind the development of Frame Relay, Broadband ISDN (B-ISDN) was the starting point for **Asynchronous Transfer Mode (ATM)**. ATM is one of the most ambitious technologies ever dreamt of, simply because it aims at providing a one-stop solution to all voice and non-voice transmission requirements.

The word *asynchronous* stands out here, especially because ATM has emerged out of B-ISDN. B-ISDN itself is based on N-ISDN, which uses a synchronous Time Division Multiplexing (TDM) technique. Thus, it is natural to expect that B-ISDN, and therefore, ATM, would follow suit and be synchronous technologies. In reality, the opposite is true. ATM, as the name suggests, is *asynchronous*. Why do we have this case? Why is synchronous TDM not so attractive? Let us understand this with an example as shown in Fig. 13.1.

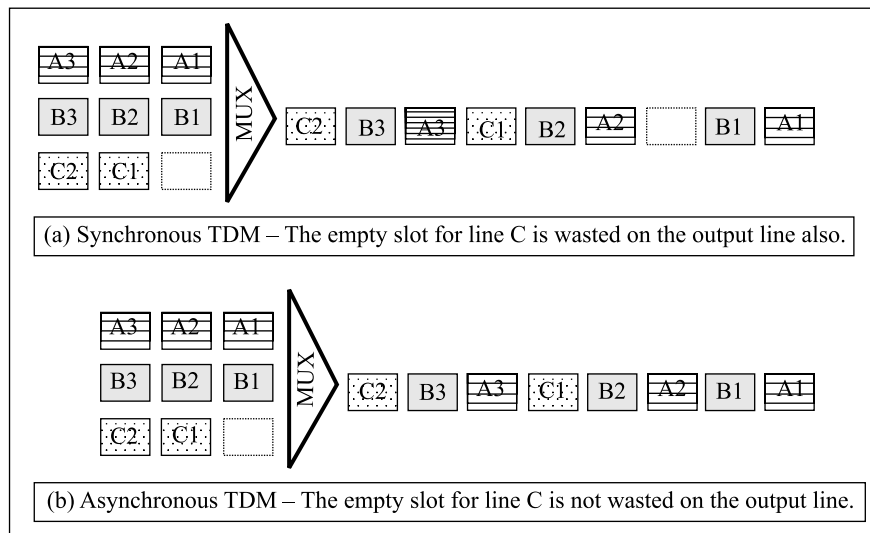


Fig. 13.1 Synchronous versus asynchronous TDM

Figure 13.1 differentiates synchronous and asynchronous TDM. The figure shows three input lines A, B and C, which are multiplexed to produce output on a single output line. In both cases (synchronous TDM and asynchronous TDM), lines A and B have three packets to send. However, line C has just two packets to send.

Figure 13.1(a) shows what happens in case of synchronous TDM. Note that the empty slot for line C is kept empty in the output slot as well. That is, since multiplexing is strictly happening in turn, even if an input line does not have any data to send, the multiplexer still allocates it a time slot, which is wasted.

Similarly, Fig. 13.1(b) shows what happens in case of asynchronous TDM. Note that the empty slot for line C is *not* kept empty in the output slot. Instead, the next available packet from another input line (i.e., A, in this case) is selected for transmission. Thus, no slot is wasted. No transmission takes place on a slot if and only if none of the input lines has any transmissions to make.

Technically, there are two key reasons behind the growing interest for asynchronous TDM.

1. Synchronous TDM does not provide flexibility in its switching approach to meet the needs of different applications. Remember that ISDN was created to integrate multiple (diverse) applications/interfaces. These different applications can and do demand different transmission rates. The standard B and D channels are acceptable for N-ISDN, but are not sufficient to handle all of B-ISDN requirements. Most data applications are bursty, whereas most voice and video applications have consistent transmission rate demands. Circuit switching approach suits voice and video, but not data, whereas, data transmission is better done with packet switching, as discussed earlier.
2. B-ISDN intends to support multiple high transmission rates (several Megabits). That is, some of the B-ISDN applications demand constant transmission rates, and others expect a support for variable transmission rates. Making such transmissions with diverse demands synchronous would mean that the switching system would be extremely complicated. This is because it would need the ability to handle varying high transmission rates. Switches should be capable of handling varying data rates. In case of N-ISDN, the transmission rate is just 64 Kbps. So, the switches need not have too much of intelligence.

These concerns led to the development of ATM. ATM is actually similar to Frame Relay in concept. Like Frame Relay, ATM assumes that the underlying transmission system is reliable. Unlike X.25, which provides for elaborate transmission flow control and error control, ATM is a fast packet transmission technology that works with the digital transmission systems extremely well.

13.1 OVERVIEW OF ATM

As mentioned earlier, ATM is a data link layer packet network. Like X.25 and Frame Relay, ATM supports multiplexing of multiple logical connections over a single physical channel. The information over a logical ATM connection flows in the form of **cells**. A cell is a fixed-size packet. Because of this nomenclature, ATM itself is sometimes called **Cell Relay**. Like Frame Relay (and unlike X.25), ATM does not provide any flow control or error control at the data link layer.

The designers of ATM have ensured backward compatibility. This means that ATM can serve as a LAN or a WAN backbone, without requiring any major hardware replacement. Thus, organizations are encouraged to opt for ATM without making any losses to their existing investments. The designers

have provided mapping mechanisms that allow frame and packet formats of other protocols to be transformed into ATM cells.

ATM is a modern WAN switching technique devised by telephone companies to deliver voice, data, and even video. ATM is designed with a very ambitious goal of providing a *universal service*, which would eventually replace all the existing and diverse network technologies. Till that happens, however, it must support the diverse networks.

At a very broad level, an ATM network uses switches. Multiple hosts connect to each switch, as shown in Fig. 13.2. Here, five hosts are connected to an ATM switch.

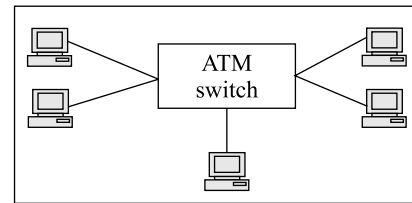


Fig. 13.2 ATM switch

As Fig. 13.2 shows, multiple hosts connect to the central ATM switch to create a star topology. This also means that in the case of ATM, the communication is between the two hosts via the switch. Recall that this is in contrast to a bus network, where the signal transmitted by a host on the bus reaches all the other hosts on that bus. This also means that ATM does not depend on the connections between the hosts, unlike a bus network. Even if a host on the ATM network goes down, or the connection between a host and the switch breaks, only that host is disabled without affecting the other hosts on the ATM network.

Each ATM switch connects to many other switches or hosts. This means that an ATM network is highly scalable. As more and more hosts are added to a network, more switches can be brought in to the ATM network to support these additional hosts.

ATM is designed to offer extremely high data rates. For instance, a typical ATM connection between a host and a switch can offer data transmission rate of 155 Mbps. Of course, to support this, ATM uses optical fiber, rather than copper wires, as the transmission medium. Like FDDI, the connection between a host and a switch consists of a pair of optical fiber cables. However, the reason for doing this in case of ATM is different. Because ATM is designed to be a full-duplex data transmission network; it uses two optical fibers – one for each direction of data transmission. Note that in the case of FDDI, it is done for a different reason – for ensuring that if one route goes down, the other can be used. This is conceptually shown in Fig. 13.3.

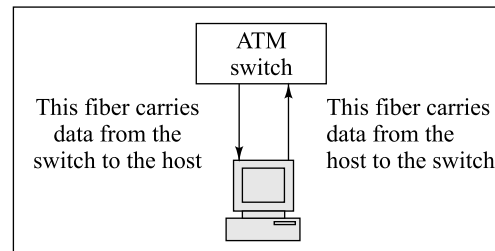


Fig. 13.3 ATM connection using two optical fibers

The challenges for ATM are many. Because it is devised to carry computer, voice, audio and video data, it must make an attempt to serve these conflicting and varying needs in an efficient manner. For instance, voice traffic (in digitized form, e.g., telephone conversations) must be communicated very fast, even if there are minor errors in transmission. On the other hand, computer data needs a very high amount of accuracy, even if the transmission speed suffers somewhat as a result of this. In contrast, video demands a totally different requirement to be satisfied, i.e., very high bandwidth. If

video is delivered at a slow speed, human eyes may not be fooled to believe that it is motion, which is actually required. To create the illusion of motion (called animation), human eyes must view pictures at the rate of 30 per second or more. If the ATM network does not deliver these pictures that constitute video, at this rate, then the video would be jerky.

Let us study how ATM attempts to meet all these challenges.

13.2 PACKET SIZE

The key issue in ATM networks is the packet size. What packet size is optimum? This has always been a matter of great debate over several decades in communications technology, even before ATM came up. Let us review this debate in brief.

13.2.1 Large Packet Size

The general conclusion is that it is better to have as large a packet size as possible. This is because, as we know, each packet contains a header that contains the information about the source and the destination addresses, CRC, etc. If we have smaller packets, each packet would have to have this additional overhead of packet headers anyway. Instead, if we have a bigger packet, we will still need only one header of the same size, thereby reducing the overhead percentage.

Consequently, computer networks have packet sizes in the order of a few Kb. However, as we know, the Pulse Code Modulation (PCM) method used for digitizing analog voice signals usually samples the incoming voice signal once every 125 microseconds (i.e., 8000 times a second), each sample consisting of 8 bits, thus requiring a speed of $8000 \text{ samples} \times 8 \text{ bits per sample} = 64 \text{ Kbps}$. Therefore, if we use standard packet sizes (e.g., 8 Kb per packet), to fill such a packet, we must wait for enough voice signals to arrive. When many such voice signals are accumulated, we would send them together. This means that the transmission would be delayed, thus causing a noticeable delay between the times a person speaks and the other person hears it.

For instance, let us say that out of 8 Kb, only one byte is remaining. When the person speaking starts her/his next sentence, the first word itself will fill this up. Now, the first packet will have to be sent through various switches to the destination, while the remaining portion of her/his sentence and subsequent conversation would have to be digitized and stored in the next packet. Additionally, if the scheme employs low speed links and acknowledgements at every stage, this would be a very painful exercise.

Variable Length Packets

To avoid the problems associated with large packet sizes, some networks use variable-length packets. This means that voice traffic can be sent in smaller packets, whereas other data can be sent in the form of bigger packets. This would seem to work well, and it does. However, this approach also has one drawback, which is complexity. The more the varying packet sizes, the more is the additional complexity for the underlying network, as it has to cater to the needs of varying packet sizes, and therefore, formats and their interpretations.

Another problem with varying packet sizes is that small packets could face injustice from big packets. For instance, suppose that two roads meet at a bridge. Just before your car reaches the bridge, a huge parade consisting of about 100 cars reaches the bridge. Because the parade arrives

there first, you must wait until all the cars in the parade cross the bridge. In a similar situation, if a network allows varying length packets, there would be situations when a small packet must wait for large packets to be transmitted first, before its turn. For example, Fig. 13.4 shows a multiplexer that accepts packets from many source routes, and sends it to a single outbound route. As the figure depicts, because a large packet P arrives before three small packets X, Y and Z at the multiplexer, packets X, Y and Z must wait until the whole of packet P is transmitted. If these packets contain digitized voice or video, it will not be heard or seen properly.

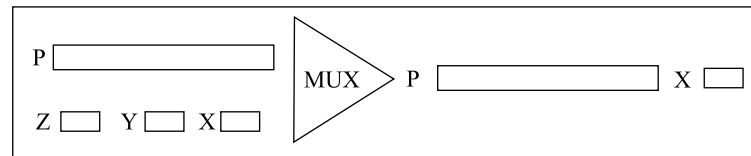


Fig. 13.4 Small packets must wait for large packets in case of variable length packets

The ATM Solution

Due to the reasons outlined above, the packet size cannot be too high or variable in ATM networks, as it is intended to serve as a data as well as voice transmission technology.

To achieve higher data rates and to avoid transmission delays, ATM uses a fixed packet size of 53 octates (an octate, like a byte, consists of 8 bits). Each 53-octate ATM packet is called a **cell**. Each cell contains a header consisting of 5 octates and data consisting of 48 octates. This is shown at a broad level in Fig. 13.5.

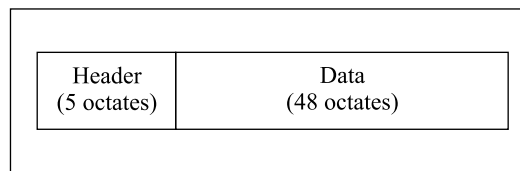


Fig. 13.5 An ATM cell consists of 53 octates

During the standardization of ATM, a conflict arose within the CCITT regarding the size of an ATM cell. The US wanted 64 octate cells because it was felt optimal for the US networks. The Europeans and Japanese wanted 32 octate cells, because it was optimal for them. In the end 48 octates was chosen as a compromise. Therefore, adding the 5 octates of header to the 48 octates of data gives 53 octates in total.

In ATM, even if the input is received in the form of large or variable length packets, all these packets are converted into equal cells, each of 53 octates. Thus, there is no question of large packets blocking the way of small packets. Using Time Division Multiplexing (TDM), these packets would be delivered in turn. Figure 13.6 shows the advantage of using fixed-size small cells. Contrast this with Fig. 13.4. However, the whole operation is so fast that the receiver receives these cells as if all these cells comprising one large packet are sent over a leased line with a marginal difference. At least, this is the idea.

As shown in Fig. 13.6, the large incoming packet P is split up into three smaller packets called A, B and C before transmission. Therefore, the multiplexer now takes turns to transmit packets from the two transmission lines. As a result, the packets X, Y and Z on the second transmission line now get a fair deal—they need not wait for the (original) large packet P to be transmitted completely. They can be transmitted in parallel with the split up packets A, B and C.

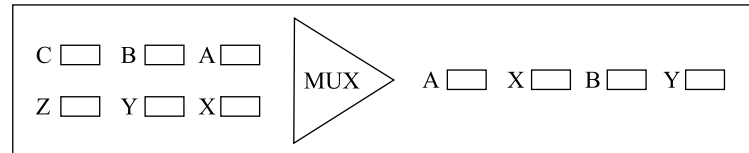


Fig. 13.6 Small fixed-size cells ensure a better justice for all routes

Moreover, different applications can specify their requirements by using a *service class*. For example, in the case of voice and video transmissions, where the transmission rate should be consistent, we can use the service class of **Constant Bit rate (CBR)**. In contrast, computer applications can use the service class **Available Bit Rate (ABR)**, which means send data as and when available, without waiting for a packet to be filled completely. We shall discuss these subsequently.

13.3 VIRTUAL CIRCUITS IN ATM

The advantages of using small fix-sized cells can be lost quickly if the cells are not routed efficiently. For instance, if the cells travel via different paths from the source to the destination (as it happens in case of packet switching), the destination would have to reassemble the original message from the cells that arrive via these different paths, resequence them, and check for missing or duplicate cells. These kinds of overheads can quickly negate the advantages offered by cell networks.

To avoid this, ATM networks use the concept of virtual circuits. As we have studied earlier, this approach means that all the cells in a single message would travel via a fixed, predefined route. Consequently, this means that the cells would arrive at the destination in the same order as the one in which they were sent originally by the sender. Therefore, there is no need for resequencing at the destination.

Let us understand this with an example. Figure 13.7 illustrates how a virtual circuit approach works. Here, computer A wants to send a message to computer D. As you can see, there is more than one possible route, which can be used for this. However, one route is finalized during the *connection establishment* phase, taking into account the congestion conditions at various nodes as well as the data rates available on various links joining these nodes on different routes. The best possible route is chosen, using a routing algorithm. Once this route is fixed, all the packets in that message are routed via the same route, even though alternate routes are available. This is the *data transfer* phase. Finally, the virtual connection is released during the *connection release* phase. Therefore, each cell contains a header and data portion. The header contains the route identification of the route chosen. Also, at every switch, a table is maintained, which provides the *next hop* the cell should be routed to, for a given route. This table is generated during the connection establishment phase at all the switches on the chosen route and deleted on releasing the connection. This reduces the header size, because, you no longer need to mention the full source and destination addresses.

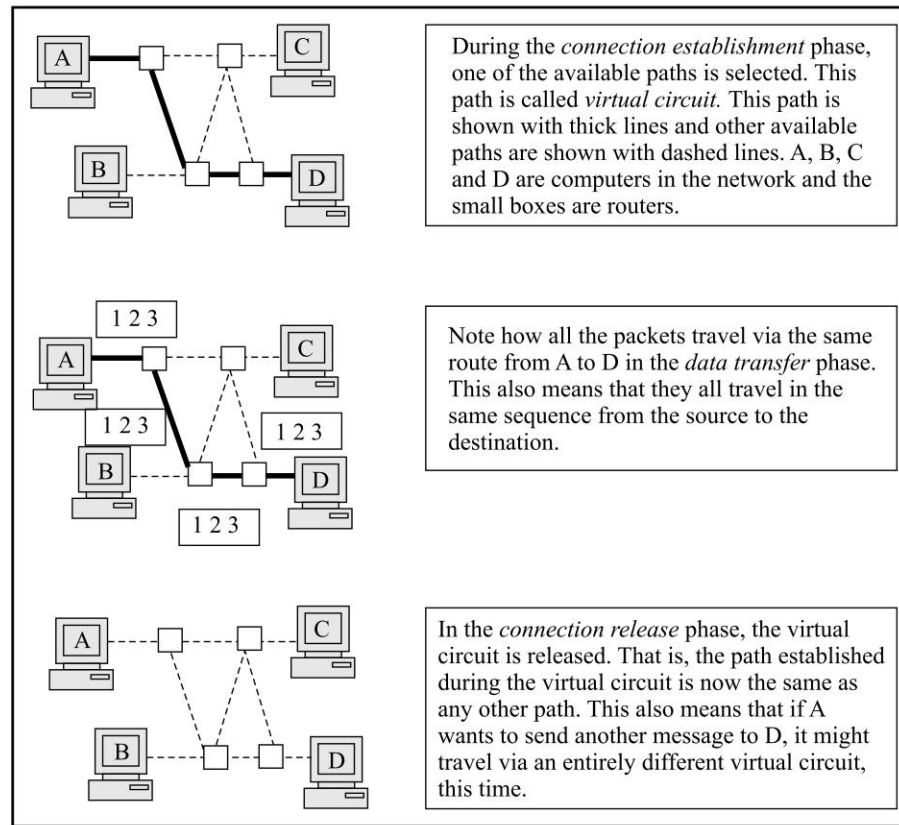


Fig. 13.7 Virtual circuit approach

ATM uses the same philosophy. Once a message is split up into 53-octate cells, each cell travels via the same route from the source to the destination.

Each cell header contains a field called **Virtual Path Identifier (VPI)**. The VPI identifies a virtual path uniquely. It is a short 12-bit number, which is generated when the connection is established. Thus, when a virtual circuit is first created, it is given a unique VPI, and that VPI is then used in the rest of the data transmission.

There is another identifier used, called **Virtual Circuit Identifier (VCI)**. The VPI identifies a path, whereas the VCI identifies an individual circuit or connection on that path. VCI is required in addition to VPI because there could be multiple circuits, connections or channels multiplexed on the same path. The Virtual Path can be viewed as a trunk that carries multiple circuits between the same switches.

A simple analogy to understand this is to think of an ATM connection as one or more highways connecting two cities (a computer and a switch, or a switch and a switch). Each virtual path is such a highway. Virtual circuits can be thought of as lanes on that highway.

This is shown in Fig. 13.8.

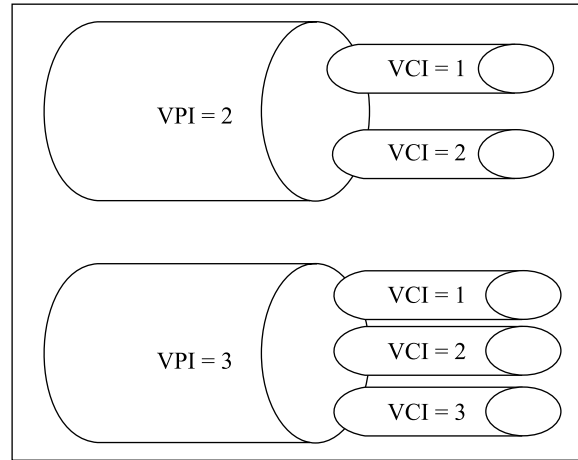


Fig. 13.8 VPI and VCI

The VCI consists of 16 bits. Thus, we have

$$\text{Connection identifier} = \text{VPI} + \text{VCI} = 12 \text{ bits} + 16 \text{ bits} = 28 \text{ bits}$$

Each cell would carry this 28-bit connection identifier. This is significantly smaller than the ATM host addresses, which can be as big as consisting of 160 bits. This makes the cell headers small, and thus reduces the overheads.

The virtual circuits can be preconfigured in the hardware, in which case, it is called **Permanent Virtual Circuit (PVC)**, or they can be stored in RAM and updated as and when required, in an approach called **Switched Virtual Circuit (SVC)**.

PVCs and SVCs both use packet switching, but set up a connection-oriented path through the network before the data is sent.

- PVC is *provisioned*, which means that someone places the configuration information on non-volatile storage, so although it takes a long time to set up, the PVC survives power failures as well. Thus, when using PVC, the VPI and VCI values are set up beforehand in the tables maintained by ATM switches, and they are used during transmission straightaway.
- Like a telephone call, an SVC is created on demand—a computer requests an SVC, sends data, and then terminates the SVC. To establish SVC, ATM uses higher layer protocols such as B-ISDN or IP. These protocols create a SVC between the two end points on demand, and ATM uses that SVC for transmission during that session.

13.4 ATM CELLS

ATM works with two cell formats, which differ from each other slightly. These two formats are (a) **User-Network Interface (UNI)** and (b) **Network-Network Interface (NNI)**. The user access devices connect to the ATM switches through a UNI interface. The switches are connected with other switches using a NNI. This is shown in Fig. 13.9.

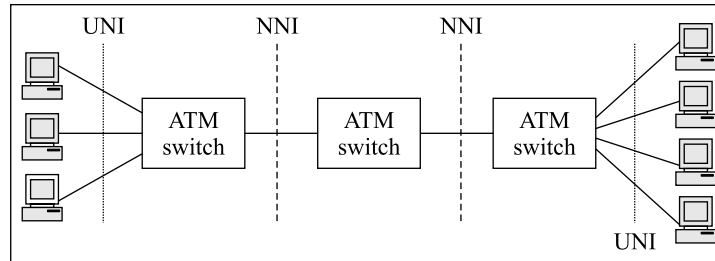


Fig. 13.9 UNI and NNI

Since the cell size is fixed at 53 octates, both UNI and NNI use 53-octate cells (48 octates for the data and 5 octates for the header).

Figures 13.10 and 13.11 show the cell formats for UNI and NNI. We shall explain the acronyms after the figures.

GFC	VPI	VCI	PT	R	CLP	HEC	User data
4 bits	8 bits	16 bits	2 bits	1 bit	1 bit	8 bits	48 octates

Fig. 13.10 UNI cell format

VPI	VCI	PT	R	CLP	HEC	User data
12 bits	16 bits	2 bits	1 bit	1 bit	8 bits	48 octates

Fig. 13.11 NNI cell format

As the above figures show, the UNI and NNI cell formats differ only in the way the first 12 bits are defined.

- In case of UNI, the first four bits are reserved for GFC, and the remaining 8 bits are reserved for VPI.
- In case of NNI, the first 12 bits collectively denote the VPI.

Figure 13.12 explains the various fields of the ATM cell formats.

13.5 SWITCHING

Let us consider a sample ATM network to understand how ATM switches can route ATM cells from the source to the destination. There are two types of ATM switches, viz., **VP switch** (that routes packets based only on the VPI values) and **VPC switch** (that routes packets based on VPI as well as VCI values).

13.5.1 VP Switch

In Fig. 13.13, we have shown an ATM VP switch that has six incoming interfaces and six outgoing interfaces. For simplicity, we have shown just one incoming interface (number 2) and two outgoing interfaces (number 10 and 12) as active. As we can see, the incoming interface (number 2) has received two packets, with VPI values = 78 and 121. The ATM switch maintains a table as shown

in the second half of the figure to determine that a packet arriving at interface 2 with VPI = 78 should be routed to interface 10 with VPI = 37. Similarly, the ATM switch also determines that the second packet with VPI = 121, which has arrived at interface 2 should be routed to interface 12 with VPI = 147. This is how an ATM switch effects forwarding of packets from incoming interfaces to the appropriate outgoing interfaces.

Field	Long Form	Meaning/Purpose
GFC	Generic Flow Control	The four-bit GFC field is used for flow control operations at the UNI level. It was felt at the time of the development of ATM that this field is not necessary in the case of NNI, so it is absent in NNI. Instead, in NNI, these bits are added to the VPI.
VPI	Virtual Path Identifier	As explained earlier, this field explains the virtual path used for the given transmission.
VCI	Virtual Channel Identifier	As explained earlier, this field explains the virtual channel within a virtual path used for the given transmission.
PT	Payload Type	This field identifies if the ATM cell contains user information or network information.
R	Reserved	This field is not used currently.
CLP	Cell Loss Priority	This field guides the ATM network when congestion occurs. If this field contains 0, it means that this cell has a very high priority, and must not be ignored. If this field contains 1, it means that in the event of heavy congestion, this cell can be discarded.
HEC	Header Error Control	This 8-bit field contains error control information.
User data	—	This field contains the actual data to be transmitted.

Fig. 13.12 *Fields in an ATM cell*

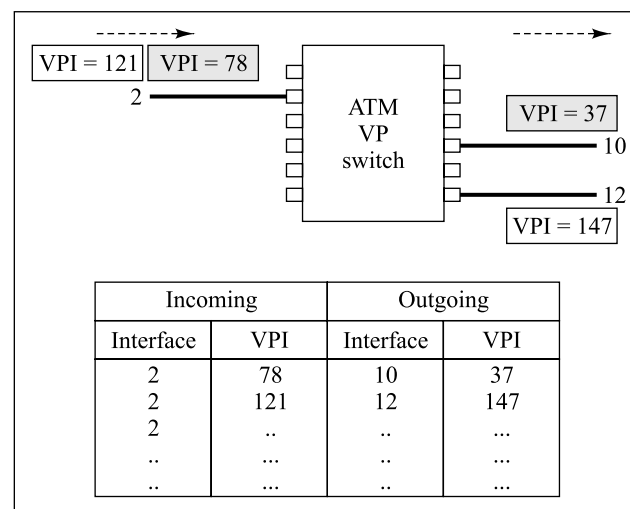


Fig. 13.13 *ATM routing concept using a VP switch*

13.5.2 VPC Switch

The ATM VPC switch deals with the VPI as well as VCI values for routing packets. We shall consider the same example we used to illustrate a VP switch, but shall also include the VC values now, as shown in Fig. 13.14.

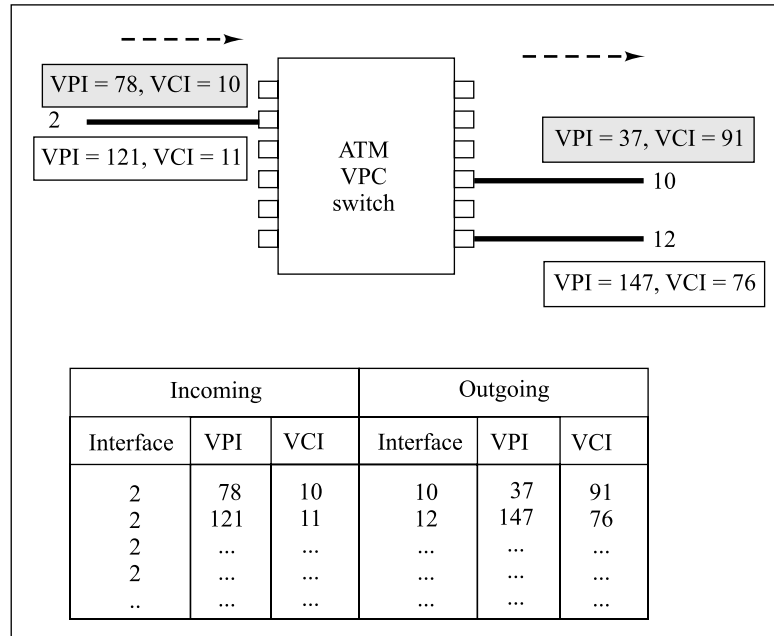


Fig. 13.14 ATM routing concept using a VP switch

As before, we have an ATM switch – this time a VPC switch – that has also six incoming interfaces and six outgoing interfaces. For simplicity, we have shown just one incoming interface (number 2) and two outgoing interfaces (number 10 and 12) as active. As we can see, the incoming interface (number 2) has received two packets, with (VPI = 78, VCI = 10) and (VPI = 121, VCI = 11). The ATM VPC switch maintains a table as shown in the second half of the figure to determine that a packet arriving at interface 2 with (VPI = 78, VCI = 10) should be routed to interface 10 with (VPI = 37, VCI = 91). Similarly, the ATM switch also determines that the second packet with (VPI = 121, VCI = 11), which has arrived at interface 2 should be routed to interface 12 with (VPI = 147, VCI = 76). This is how an ATM VPC switch affects forwarding of packets from incoming interfaces to the appropriate outgoing interfaces.

Why does ATM deploy two types of switches? This is used for enabling hierarchical routing of packets. Most ATM switches in an ATM network are VP switches, as they need to deal only with VPIs for routing decisions. However, the switches that connect to the network boundaries are VPC switches, as they need to deal with both VPIs and VCIs.

13.6 ATM LAYERS

ATM protocol consists of three layers, as shown in Fig. 13.15. These layers are Physical Layer, **ATM Layer** and **Application Adaptation Layer (AAL)**.

Let us discuss these layers now.

13.6.1 Physical Layer

Like all other networking protocols, the ATM Physical Layer deals with issues related to transmission media, bit transmissions, electrical/optical interfaces and encoding techniques. This enables transformation of ATM cells into raw bits, which can be transmitted across a transmission medium (usually a wire). The transmission medium is preferably optical fiber, although it can also be a twisted-pair wire or coaxial cable. ATM does not rule any one of these out. However, twisted-pair wire is generally not used, as it cannot support the high data rates demanded by B-ISDN.

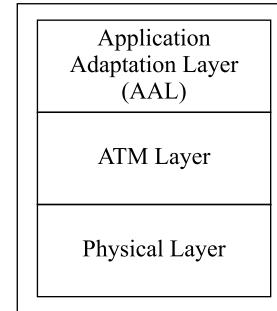


Fig. 13.15 *ATM layers*

13.6.2 ATM Data Link Layer

The ATM layer (also called ATM data link layer) deals with cell routing, switching, multiplexing and traffic management. The core area of the ATM layer is the definition of the ATM cell, which is a 53-octate unit, as we have discussed in detail. The ATM cell is defined in the ATM layer. The ATM layer accepts 48-byte AAL segments from the AAL layer, and adds the 5-byte cell header to transform the segment into a 53-byte ATM cell, that can be delivered to the physical layer. This is shown in Fig. 13.16.

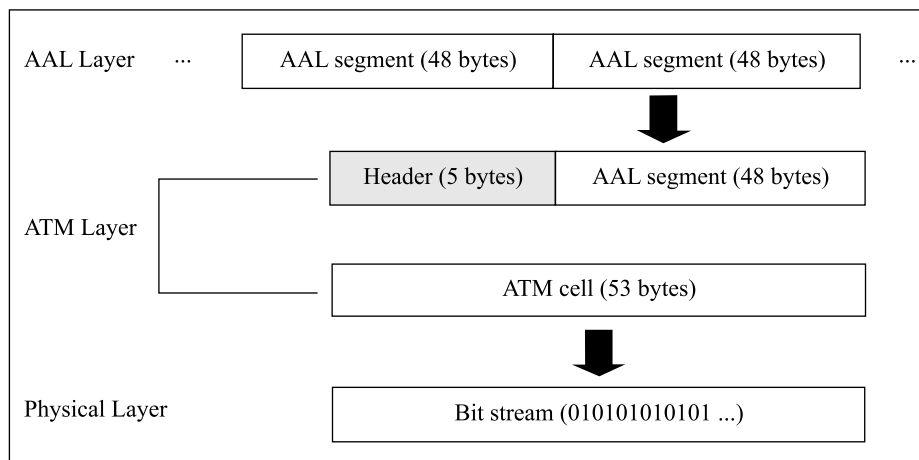


Fig. 13.16 *Formation of an ATM cell*

We have already discussed the format of the ATM cell in detail earlier. Therefore, we shall not discuss it here again.


13.6.3 Application Adaptation Layer (AAL)

As we have noted before, the **Application Adaptation Layer (AAL)** facilitates an interaction between the existing networks and ATM. At the incoming end, the AAL layer accepts transmission frames from the higher layers in the protocol model and transforms them into ATM cells. The incoming frames can be circuit/packet switched, can have fixed/variable data rate, and can represent voice/data/other transmission types. At the outgoing end, the AAL transforms fixed-size ATM cells into appropriate protocol frames.

13.6.4 Data Types and Categories

To facilitate the handling of different data types, ATM defines four different data types. That is, an AAL layer can take one of these types, and consequently, a different cell format for each one of them. The ITU-T has defined another data type to make the total number of types five. Figure 13.17 tabulates these types.

Data type	Description
Constant Bit Rate (CBR)	This type depicts data coming from or going to applications that produce or consume it at a fairly constant rate. In other words, transmission delays are not tolerable in this data type. Telephone calls, other voice transmissions and video are examples of CBR data.
Variable Bit Rate (VBR)	VBR data refers to data that has variable transmission rates. Notably, the bit rate changes from one part of the transmission to another, but has predefined parameters as to how much these can vary. Examples of VBR data are compressed data, voice and video.
Connection-oriented packet data	This type of data refers to computer-generated data that is carried by virtual circuits and internally uses packets to transmit data. Protocols such as X.25 and TCP are examples of this kind.
Connectionless packet data	This data type considers data coming from connectionless packet networks (i.e., the ones using datagram approach), such as the IP protocol.
Simple and Efficient Adaptation Layer (SEAL)	Defined by ITU-T, SEAL is a generic data type that refers to point-to-point, and never multipoint or internetwork-based transmissions.

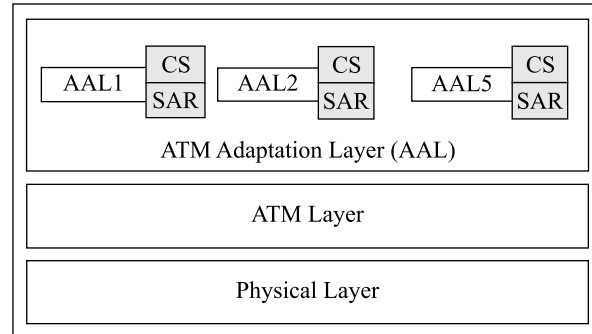
Fig. 13.17  Data types defined by AAL

To handle these five data types, the AAL defines five different categories, called AAL1 through AAL5. After some reconsideration, it has been decided to merge AAL3 and AAL4, as there is too much of overlap between them, and form a combined category called AAL3/4. It is also debated whether AAL2 should be dropped or combined with another category.

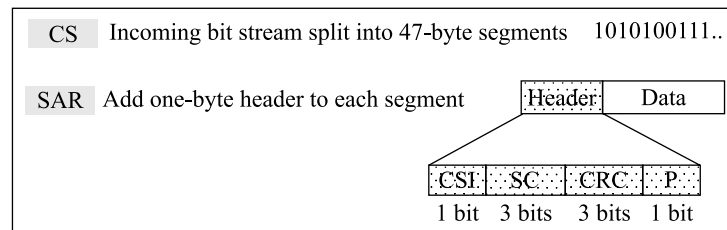
Furthermore, each category (AAL1 through AAL5) contains two sublayers, based on the functionality. The two sublayers are called **Convergence Sublayer (CS)** and **Segmentation And Reassembly (SAR)**. Thus, when we describe each of the categories AAL1 through AAL5, we shall discuss the aspects related to its CS and SAR. Conceptually, CS and SAR are shown in Fig. 13.18.

13.6.5 AAL1

As we have mentioned before, **AAL1** supports applications that transfer information at a constant bit rate. Examples of such data transmissions are voice and video. AAL1 allows ATM to work efficiently with the existing digital telephone networks such as DS-3 or E-1.

**Fig. 13.18** *CS and SAR in AAL*

The CS sublayer here divides the incoming bit stream into 47-byte segments, and passes them to the SAR sublayer. The SAR sublayer adds one byte of header, making this a 48-byte data unit, as shown in Fig. 13.19. This is given to the ATM layer below, which encapsulates it into a cell.

**Fig. 13.19** *AAL1 operation*

The subfields in the one-byte header appended by SAR to the CS segment are described in Fig. 13.20.

Header field	Description
Convergence Sublayer Identifier (CSI)	This bit is used for signaling purposes that are not yet specified.
Sequence Count (SC)	This field is used for end-to-end flow control and error control. It defines the modulo 8 sequence number of the segment.
Cyclic Redundancy Check (CRC)	The CRC field serves the same error-detection purposes, as in the case of other protocols.
Parity (P)	The parity field serves the same error-detection purposes, as in the case of other protocols.

Fig. 13.20 *AAL1 header subfields*

13.6.6 AAL2

AAL2 supports variable bit rates (such as compressed voice, video and data). The exact specifications of how AAL2 will achieve this have not been released.

The CS sublayer here divides the incoming bit stream into 45-byte segments, and passes them to the SAR sublayer. The SAR sublayer adds one byte of header and two bytes of trailer, making

this a 48-byte data unit, as shown in Fig. 13.21. This is given to the ATM layer below, which encapsulates it into a cell.

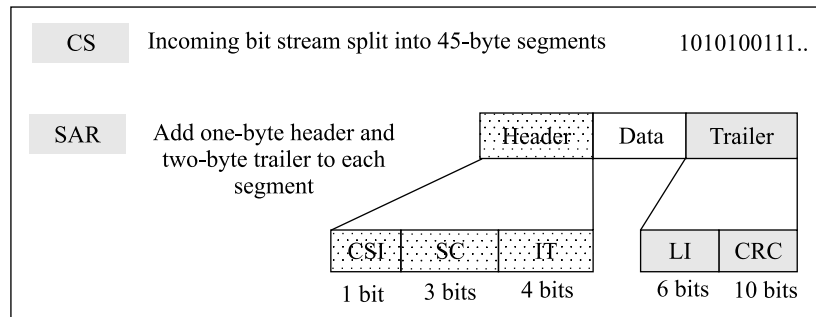


Fig. 13.21 AAL2 operation

The subfields in the one-byte header and two-byte trailer appended by SAR to the CS segment are described in Fig. 13.22.

Header/Trailer field	Description
Convergence Sublayer Identifier (CSI)	This bit is used for signaling purposes that are not yet specified.
Sequence Count (SC)	This field is used for end-to-end flow control and error control. It defines the modulo 8 sequence number of the segment.
Information Type (IT)	This field identifies if this is the start, middle or end of data.
Length Indicator (LI)	This field is useful for the last segment of a message to indicate how much portion of this segment is data and how much is the padding (if any).
Cyclic Redundancy Check (CRC)	The CRC field serves the same error-detection and error-detection purposes, as in the case of other protocols.

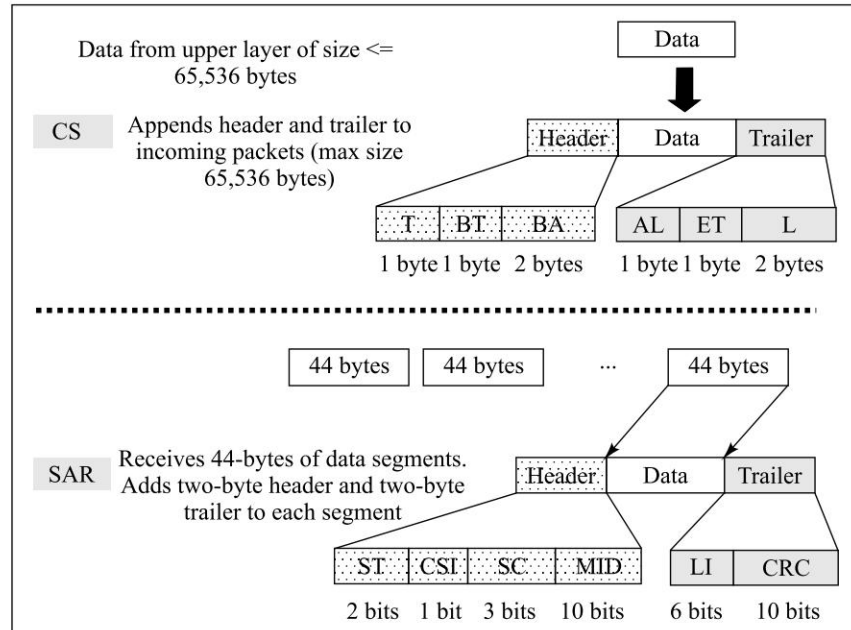
Fig. 13.22 AAL2 header and trailer subfields

13.6.7 AAL3/4

AAL3 and AAL4 were initially conceived to respectively support connection-oriented and connectionless data. However, it was soon realized that the basic issues with the two are the same, and therefore, the two categories were merged to form a single category AAL3/4.

The CS sublayer accepts a data packet with a maximum length of 65536 bytes from an upper layer protocol, such as Frame Relay or SMDS. It then adds a header and trailer to indicate the start and the end of the message, and only for the last frame, how much is the data and how much is the padding. It then splits these packets into 44-byte segments, and hands them over to the SAR layer. Interestingly, the CS does not append the header and trailer to every 44-byte segment – it adds them to the beginning and end of only the original segment. Thus, the middle portions of data are passed as they originally were. This ensures that the integrity of the original message is retained, and also keeps the overhead of headers and trailers to a minimum.

The SAR sublayer accepts 44-byte segments from the CS sublayer, and appends a two-byte header and a two-byte trailer to each frame and passes such 48-bit frames. This is shown in Fig. 13.23. The ATM layer receives these frames and encapsulates each one into a 48-byte cell.

**Fig. 13.23** AAL3/4 operation

The subfields in the CS and SAR headers/trailers are described in Fig. 13.24.

Header/Trailer field	Description
Type (T)	This one-bit field is the legacy from the old AAL3 category, and is not used here.
Begin Tag (BT)	This one-byte field indicates the beginning of data for the receiver's synchronization purposes.
Buffer Allocation (BA)	This two-byte field informs the receiver how big its buffer should be for receiving data.
Alignment (AL)	This one-byte field is not used for any special purposes.
End Tag (ET)	This one-byte field indicates the ending of data for the receiver's synchronization purposes.
Length (L)	This two-byte field indicates the length of the CS data unit.
Segment Type (ST)	This field identifies if this is the start, middle or end of data.
Convergence Sublayer Identifier (CSI)	This bit is used for signaling purposes that are not yet specified.
Sequence Count (SC)	This three-bit field is used for end-to-end flow control and error control. It defines the modulo 8 sequence number of the segment.
Multiplexing Identification (MID)	This 10-bit field identifies the cells coming from different sources, which are multiplexed on the same virtual circuit.
Length Indicator (LI)	This field is useful for the last segment of a message to indicate how much portion of this segment is data and how much is the padding (if any).
Cyclic Redundancy Check (CRC)	The CRC field serves the same error-detection and error-detection purposes, as in the case of other protocols.

Fig. 13.24 AAL3/4 header and trailer subfields

13.6.8 AAL5

The AAL3/4 provides services for flow control and error control that are quite comprehensive. However, all applications do not demand these services. For instance, LAN backbones, which are point-to-point, do not need such facilities. In such situations, the AAL5 is useful. It assumes that all the cells belonging to a single transmission (i.e., a single message) travel sequentially in order, and that the upper layers provide the necessary services for flow control and error control. Therefore, AAL5 does not provide any addressing, ordering or other header information either with the CS or with the SAR. Hence it is named Simple and Efficient Adaptation Layer (SEAL).

The CS sublayer accepts a data packet of size 65,536 bytes or less, and adds an 8-byte trailer, as shown in Fig. 13.25. The CS unit then breaks the whole packet into 48-byte segments, and gives them to the SAR sublayer. The SAR sublayer does not perform any additional processing, and instead, hands over the segments to the ATM layer, where they are encapsulated into 48-byte cells.

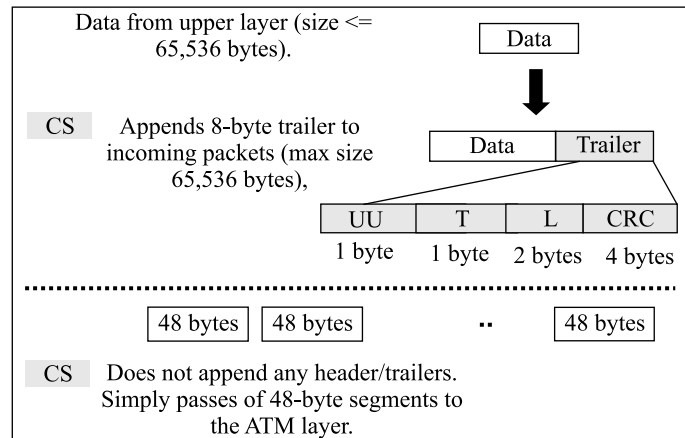


Fig. 13.25 AAL5 operation

The subfields in the CS trailer are described in Fig. 13.26.

Trailer field	Description
User to User ID (UU)	This one-byte field is not used currently.
Type (T)	This one-bit field is not used currently.
Length (L)	This two-byte field indicates the length of the CS data unit.
Cyclic Redundancy Check (CRC)	The CRC field serves the same error-detection and error-detection purposes, as in the case of other protocols.

Fig. 13.26 AAL5 header and trailer subfields

13.7 MISCELLANEOUS TOPICS

13.7.1 Service Classes

The ATM Forum defines four types of service classes, viz., **Constant Bit Rate (CBR)**, **Variable Bit Rate (VBR)**, **Available Bit rate (ABR)** and **Unspecified Bit Rate (UBR)**. They are summarized in Fig. 13.27.

Service class	Description
CBR	Similar to a T-line, this class is useful for customers needing real-time audio or video services.
VBR	Classified into VBR-Real Time (VBR-RT) and VBR-Non-Real Time (VBR-NRT) , this class is used for applications that use compression techniques. VBR-RT applications require real-time services, whereas VBR-NRT services are useful for applications that do not demand real-time services.
ABR	Suitable for applications that are bursty in nature, ABR is the minimum guaranteed rate.
UBR	UBR does not guarantee anything – it is a best-effort delivery mechanism.

Fig. 13.27  *ATM Service Classes*

13.7.2 Quality of Service (QoS)

The Quality of Service (QoS) defines parameters related to the performance of an ATM network. An ATM user can request for these parameters. Each service class is associated with one or more attributes. The attributes can be classified into two major categories as user-related attributes and network-related attributes.

Figure 13.28 summarizes user-related attributes. These attributes are related to the end user in the sense that they define how fast a user wants to send/receive data. These attributes are negotiated and defined at the time of the contract between the user and the network service provider.

Attribute	Description
Sustained Cell Rate (SCR)	This is the average cell rate over a period of time, which could be more or less the actual transmission rates, as long as the average is maintained.
Peak Cell Rate (PCR)	This is the maximum transmission rate at a point of time.
Minimum Cell Rate (MCR)	This is the minimum cell rate that the network guarantees a user.
Cell Variation Delay Tolerance (CVDT)	This is a unit of measuring the changes in cell transmission times (i.e., what is the maximum and minimum delay between the delivery of any two cells).



Fig. 13.28  *User-related QoS attributes*

Figure 13.29 summarizes network-related attributes. These attributes define the characteristics of a network.

Attribute	Description
Cell Loss Ratio (CLR)	This attribute defines the fraction of the cells lost/delivered too late during transmission.
Cell Transfer Delay (CTD)	This is the average time required for a cell to travel from the source to the destination.
Cell Delay Variation (CDV)	This is the difference between the maximum and minimum values of CTD.
Cell Error Ratio (CER)	This parameter defines the fraction of cells that contained errors.

Fig. 13.29  *Network-related QoS attributes*

13.7.3 Characteristics of ATM

Let us summarize the characteristics of the ATM technology as shown in Fig. 13.30.

Characteristic	Details
Asynchronous	As the name suggests, ATM is an asynchronous network. This means that the source and the destination do not have to be coordinated before transmission begins. This also means that each cell in a transmission can travel independently without regard to other cells in the same transmission.
Virtual circuit approach	Although asynchronous, because ATM uses the virtual circuit approach, the transmission becomes connection-oriented. That is, a connection (virtual circuit) is first established before any cells can be transmitted.
Order of cells	Because all cells in a single transmission take the same route, they always arrive in the same order as the one in which they were sent, at the destination.
Equal delays	Even if there are network problems, the delays apply to all the transmissions equally. This is because different transmissions must stick to the 53-octate cell format. Therefore, there is no scope for more delays for one transmission and less delay for others.
Automatic error detection and correction	ATM does not provide elaborate retransmission mechanisms, unlike other networks for the following two reasons: (a) Today's transmission media are very reliable and the scope for transmission errors is very little. (b) ATM uses small, 53-octate cells. Because of these two reasons, it is proved that even when there are errors in ATM transmissions, they are usually one-bit errors, which means that they can be detected and corrected easily.

Fig. 13.30 ATM characteristics

SUMMARY

Asynchronous Transmission Mode (ATM) is an extremely ambitious transmission network that is expected to replace most of the existing networking protocols. ATM uses small packets, called cells for transmission. It is based on the asynchronous Time Division Multiplexing (TDM) technique and works best when the underlying transmission medium is optical fiber (although it can work with twisted-pair wire and coaxial cable). This allows fast packet transmission. Interestingly, although ATM is based on B-ISDN, it is asynchronous in nature.

ATM uses the concept of virtual circuits that are established between the two end points before transmission begins. Like X.25 and Frame Relay, the virtual circuits can be of the PVC or SVC types. A virtual circuit in ATM is identified by the combination of Virtual Path Identifier (VPI) and Virtual Circuit Identifier (VCI).

ATM works with two interfaces, which differ from each other slightly, and are called User-Network Interface (UNI) and Network-Network Interface (NNI). Each ATM cell contains 5 bytes of header information and 48 bytes of data, making it a 53-byte cell. There are two types of ATM switches. A VP switch routes cells based on the VPI values. A VPC switch routes cells based on the VPI as well as VCI values.

ATM protocol has three layers, viz., physical, ATM and AAL. The physical layer defines the hardware level transmission details. The ATM layer contains details regarding the ATM cell format

and switching information. The AAL layer is useful for transforming non-ATM frame formats into ATM cells, and vice versa. It allows ATM to interoperate with other protocol suites. AAL contains five sublayers called AAL1 through AAL5, each of which serves a different type of application. AAL sublayers 3 and 4 have been combined into a single sublayer called AAL3/4. ATM uses service classes and Quality of Service (QoS) parameters to define user and network attributes, which are then used for actual ATM operation.

KEY TERMS AND CONCEPTS

Application Adaptation Layer (AAL)	Segmentation And Reassembly (SAR)
Asynchronous Transfer Mode (ATM)	Simple and Efficient Adaptation Layer (SEAL)
ATM Layer	Unspecified Bit Rate (UBR)
Available Bit rate (CBR)	User-Network Interface (UNI)
Cell	Variable Bit Rate (VBR)
Cell Relay	Virtual Circuit Identifier (VCI)
Constant Bit rate (CBR)	Virtual Path Identifier (VPI)
Convergence Sublayer (CS)	VP switch
Network-Network Interface (NNI)	VPC switch
Quality of Service (QoS)	

QUESTIONS

True/False

1. ATM supports synchronous TDM.
2. Asynchronous TDM is more optimized than synchronous TDM.
3. ATM uses fixed size cells.
4. Each ATM cell contains 48 bytes of data and 8 bytes of header information.
5. ATM uses VPI and VCI for routing.
6. ATM does not need virtual circuits for operations.
7. ATM does not employ the datagram approach.
8. Variable-sized cells are a better approach.
9. ATM has two types of switches.
10. VPI uses 12 bits for a NNI connection.
11. AAL is used by ATM to interoperate with other protocols.
12. ATM cells are never multiplexed.
13. AAL1 and 2 have been combined.
14. AAL2 supports variable bit rate.
15. AAL3/4 provides for flow control and error control.
16. CBR is suitable for bursty traffic.
17. PCR is the maximum cell transmission rate.
18. CDV is the difference between the maximum and minimum values of CTD.

Multiple-Choice Questions

1. In the context of data transmission, ATM is the acronym for _____.
 - (a) Automatic Transmission Mechanism
 - (b) Automatic Teller Machine
 - (c) Asynchronous Transfer Mode
 - (d) Asynchronous Telecommunications Model
2. ATM is a _____ technology.
 - (a) synchronous TDM
 - (b) asynchronous TDM
 - (c) statistical TDM
 - (d) None of the above
3. ATM needs _____ sets of optical fiber cables.
 - (a) 2
 - (b) 3
 - (c) 4
 - (d) None of the above
4. ATM uses the _____ approach.
 - (a) packet switching
 - (b) datagram
 - (c) circuit switching
 - (d) virtual circuit
5. The _____ ATM layer is used for protocol transformations.
 - (a) physical
 - (b) ATM
 - (c) AAL
 - (d) data link
6. The _____ layer produces a 48-byte cell.
 - (a) physical
 - (b) ATM
 - (c) AAL
 - (d) data link
7. The output of the _____ layer is a 53-byte cell.
 - (a) physical
 - (b) ATM
 - (c) AAL
 - (d) data link
8. The UNI connection uses _____ bits to indicate the VPI field.
 - (a) 8
 - (b) 10
 - (c) 12
 - (d) 14
9. The NNI connection uses _____ bits to indicate the VPI field.
 - (a) 8
 - (b) 10
 - (c) 12
 - (d) 14
10. The _____ field in an ATM cell indicates if it is ok to drop that cell.
 - (a) VPI
 - (b) VCI
 - (c) CLP
 - (d) GFC
11. AAL1 is useful for _____.
 - (a) constant bit rate
 - (b) variable bit rate
 - (c) connection-oriented packet data
 - (d) connectionless packet data
12. AAL2 is useful for _____.
 - (a) constant bit rate
 - (b) variable bit rate
 - (c) connection-oriented packet data
 - (d) connectionless packet data
13. AAL2 is useful for _____.
 - (a) variable bit rate
 - (b) connection-oriented packet data
 - (c) connectionless packet data
 - (d) connection-oriented packet data and connectionless packet data

14. _____ is similar to a T-1 line.
 (a) CBR (b) VBR
 (c) ABR (d) UBR
15. _____ is suitable for bursty traffic.
 (a) CBR (b) VBR
 (c) ABR (d) UBR
16. _____ is the minimum cell rate that the network guarantees a user.
 (a) SCR (b) PCR
 (c) MCR (d) None of the above
17. _____ is the average time required for a cell to travel from the source to the destination.
 (a) CLR (b) CTV
 (c) CTD (d) CER

Detailed Questions

1. Discuss synchronous and asynchronous TDM.
2. Why are ATM cells of a fixed size?
3. Why are ATM cells of a small size?
4. What are the three layers in the ATM protocol?
5. What is the purpose of the ATM layer?
6. Discuss the significance of the AAL layer.
7. How are the VPI and VCI fields useful?
8. What are the two main switching mechanisms in ATM?
9. Describe the contents of an ATM cell.
10. What are UNI and NNI?
11. Discuss the purpose of the AAL1 layer.
12. How is AAL2 different from AAL1?
13. What are the situations where AAL3/4 is useful?
14. Why is AAL5 required?
15. What are service classes? List the different service classes.
16. Define the various QoS parameters related to the user and the network.

14 Wireless Communication

14.0 OVERVIEW OF WIRELESS NETWORKS

Wireless networks have been around for a number of years in various forms. For example, in the earlier days, ships used light pulses (light switched *on* and *off*) using the Morse code for signaling purposes. From these primitive wireless signals, we have moved to very sophisticated mobile phone technologies now. They are very modern and rich.

From today's view point, we can classify wireless networks into four categories, as shown in Fig. 14.1.

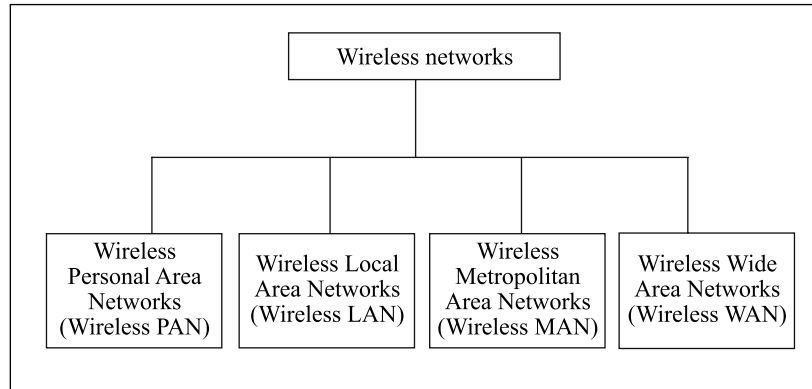


Fig. 14.1 Types of wireless networks

Let us have a brief idea of these four types of wireless networks, as shown in Table 14.1.

Let us discuss these types of wireless networks in brief.

1. **Wireless Personal Area Networks (Wireless PAN)** – A **Wireless Personal Area Network (Wireless PAN)** has a very short range of about 50 feet. It can be used for connecting devices and peripherals to each other in a personal area. The data rates of such networks are also quite small (usually 2 Mbps). Such networks are used for replacing wires in small areas. Bluetooth is an example of such a technology.

Table 14.1 *Types of wireless networks*

Type of wireless network	Purpose	Strength/ Capacity	Example Network	Applications
Wireless Personal Area Networks (Wireless PAN)	Within the reach of a person	Low to moderate	Bluetooth – IEEE 802.15	Replace cables for devices and peripherals
Wireless Local Area Networks (Wireless LAN)	Within the reach of a building or small distance	High	WiFi - IEEE 802.11	Extend wireless networks with wireless access
Wireless Metropolitan Area Networks (Wireless MAN)	Within the reach of a city	High	WiMAX – IEEE 802.16	Fixed wireless communication between home/business and the Internet
Wireless Wide Area Networks (Wireless WAN)	Available worldwide	Low	Mobile phones – 2G, 2.5G, 3G	Mobile phone communication from anywhere

2. **Wireless Local Area Networks (Wireless LAN)** – A **Wireless Local Area Network (Wireless LAN)** can be used to replace a wired LAN inside a building. Typical usages of such networks are to allow people to use the local network services or connect to the Internet, etc., without using wires. Such a network can be used as an addition to or as a replacement of traditional wired Ethernet. Usually, the highest data rates these networks provide are in the range of 54 Mbps. Some other popular places where such networks are created are the hot spots created by Internet/network service providers at train stations, airports, cafeterias, etc. The popular WiFi or IEEE 802.11 networks are examples of these networks.
3. **Wireless Metropolitan Area Networks (Wireless MAN)** – A **Wireless Metropolitan Area Network (Wireless MAN)** is used to allow people access between two fixed points or one fixed point and one mobile device. For example, we may have wireless connectivity between two branches of a bank located in the same city. Another example of wireless MAN is someone accessing the Internet from home using a wireless data card.
4. **Wireless Wide Area Networks (Wireless WAN)** – A **Wireless Wide Area Network (Wireless WAN)** covers large areas, for example, a country or a continent, or more as well. Usually such networks are very slow with maximum data access rates in kilobyte terms. Such networks can be used by people for accessing the Internet, emails, and corporate applications when on the move. Mobile phones based on technologies called 2G, 2.5G, and 3G use such networks.

14.1 IEEE STANDARDS FOR LAN, MAN, AND WAN – 802.1, 802.2, 802.3, 802.4, 802.5, 802.11

IEEE has come up with several standards for wired and wireless networks. They are usually referred to by their ‘numbers’, such as 802.1, 802.2, etc. We will briefly cover these standards.

IEEE 802.1

The **IEEE 802.1** standard covers areas pertaining to LAN/MAN architecture, their internetworking, link security, overall network management, and protocol layers above the MAC and LLC sublayers.

The IEEE 802.1 standard is actually a set of many standards, viz., IEEE 802.1B, 802.1D, 802.1E, and so on. Many of these standards became obsolete over a period of time, and were either withdrawn or merged with other standards. Some standards are of course, current. For instance, IEEE 802.1AE-2006 standard pertaining to MAC security is still active.

IEEE 802.2

The **IEEE 802.2** standard defines the Logical Link Control (LLC) sublayer of the data link layer of the OSI protocol stack. LLC is the upper half of the data link layer. Therefore, the job of the LLC is to provide a uniform interface to the layer above data link layer, i.e., to the network layer. The other sublayer in data link layer is the MAC layer, which is hardware-dependent (e.g., Ethernet, 802.11, Token Ring, etc).

The IEEE 802.2 standard provides for the following three kinds of modes:

- **Type 1** – This mode of operation is an unacknowledged connectionless mode. It provides for unicast or point-to-point transmissions, multicast transmissions (i.e., to multiple hosts on the same network), or broadcast transmissions (i.e., to all hosts on the same network). Since this is unacknowledged and connectionless service, there is no guarantee that frames sent using this mode arrive in sequence at the destination. The sender does not even know if those frames reached one or more destination.
- **Type 2** – This mode is connection-oriented. It means that before frames are sent by the sender, they are numbered in a sequence. Hence, the receiver is also guaranteed to receive these frames in the same sequence. Also, no frames can be lost.
- **Type 3** – This is a mixture of the earlier modes. It is an acknowledged connectionless service. It can be used in point-to-point, i.e., unicast mode only. Here, frames sent by the sender are not numbered, but they are acknowledged by the receiver. It means that frames can arrive out of order at the receiver, but at least the sender would come to know about this fact.

IEEE 802.3

The **IEEE 802.3** standard defines the physical layer and the Medium Access Control (MAC) sublayer of the data link layer in the case of wired Ethernet networks. This is obviously with reference to a LAN. We have already looked at several standards in this category. For example, the 1000BASE-T standard is a gigabit Ethernet over twisted pair at the rate of 1 Gbps. It is covered under the 802.3ab standard.

IEEE 802.4

The **IEEE 802.4** standard covers Token Bus. Token Bus has been discussed earlier. To summarize, Token Bus is nothing but an implementation of a Token Ring protocol, over a virtual ring over a coaxial cable. The principle of the network is quite simple. A token keeps getting circulated over the network. Only the host that possesses the token has a right to transmit. Of course, if a host possessing the token does not have anything to transmit, it simply forwards the token, i.e., the right to transmit, to the next host. For this to be possible, each host needs to know the address of its immediate neighbors. Protocols are designed to handle this, a new addition of hosts, as well as disconnections. All of this comes under the IEEE 802.4 standard.

IEEE 802.5

The **IEEE 802.5** standard is nothing but the Token Ring mechanism, as discussed earlier in the book. The Token Ring standard is based on the idea of a circulating token. A host that possesses the token can transmit, others cannot. This avoids contentions and collisions in the network. A host that does not possess the token must wait even if it has data to be sent out. A host that gets the token either can send a frame and forward the token to the next host. If it has nothing to send, it simply forwards the token to the next host.

IEEE 802.11

The **IEEE 802.11** set of standards covers Wireless Local Area Networks (WLAN). This covers the modulation of input signals in the wireless medium, i.e., air. Several variations of the basic protocol were released, such as IEEE 802.11a, IEEE 802.b, etc. Of these, IEEE 802.11b and IEEE 802.11g have been most successful. Because of restrictions on cryptography, the security in these versions was not very high. This was improved in IEEE 802.11i.

14.2 INFRARED COMMUNICATION

Infrared communication (IR) is an example of wireless communication. However, it is limited to very simple applications and suffers from several disadvantages, mainly very small bandwidth and distances that it can support. Infrared communication works in the **micrometer** range, which is 1 to 430 THz. The term *infrared* comes from the fact that red color has the longest wavelength amongst the colors in visible light. However, the wavelength of infrared is longer than that of red color, and hence the frequency of infrared communications is smaller than that of the red color. Hence, we have the term ‘below red color’ or ‘infrared’.

Infrared communication is used by military for surveillance, for vision in the darkness at night, tracking objects, etc. We use infrared communication whenever we use our remote controls to operate television sets, DVD players, etc. Weather forecasting and astronomy also make use of infrared communication.

Infrared communication can be divided into five categories, as shown below.

Type	Details
Near-infrared	This type of communication can be seen in fiber optic cables, night vision goggles, etc.
Short-wavelength infrared	This is used in long-distance communication.
Mid-wavelength infrared	This type of communication is used in launching guided missiles.
Long-wavelength infrared	Without using an external light source as the sun, this type of infrared communication facilitates taking images of an object using passive thermal technology.
Far infrared	This is used in laser systems.

14.3 BLUETOOTH

14.3.1 Basic Concepts

We encounter wires everywhere. The computer is connected to a mouse with a wire. The modem is connected to a computer using a wire. The telephone network is connected to a modem using a wire, and so on. This is quite cumbersome. There are too many wires everywhere, and this eats up a lot of space as well as creates confusions. Hence, work was on to see if we could somehow get rid of at least some of these wires. This resulted progress on a number of standards in this direction. Of these, **Bluetooth** perhaps happens to be the most popular one. More importantly, it has survived the test of time, and is now used to connect devices to one another at short distances (e.g., to connect a mouse to a computer without a wire). It is defined as an IEEE standard (**IEEE 802.15**).

The conceptual block diagram of the usage of Bluetooth can be drawn as shown in Fig. 14.2. Here, we have a main device (a laptop) using Bluetooth to communicate with its mouse, printer, and two mobile phones. Of course, no wires are needed.

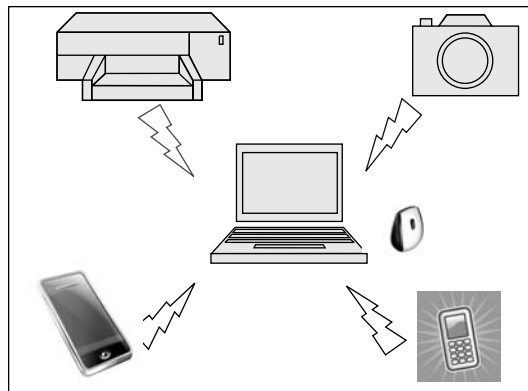


Fig. 14.2 Typical Bluetooth network

14.3.2 Bluetooth Usage Scenario

A practical example of where Bluetooth can be used would greatly help. We will describe this case with all the Bluetooth jargon included, and explain those jargons in the subsequent sections. So, we may want to revisit this example to establish the context of that jargon after we actually study it.

A person is walking into a hotel lobby that has wireless network enabled. After reaching there, s/he wants to check if any new emails are waiting to be read in her/his inbox. S/he has a laptop (or a PDA), but no wired Internet connection. But s/he has Bluetooth enabled on the laptop. S/he selects a menu or the email icon. Bluetooth devices are so programmed that as soon as they enter a new location, they automatically start an inquiry process to figure out if any wireless access points are in vicinity. For this purpose, the person's laptop sends an inquiry, to which the wireless infrastructure in the hotel responds. Now, the person's laptop sends a **paging request** to the hotel's wireless network. As a result of this paging request, the wireless network absorbs this new laptop into its existing network and in effect synchronizes the laptop with the access point. This means that their **clock offset** and **frequency hop** are synchronized. A protocol called *LMP* now establishes a

link with the wireless access point. Because the user intends to send/receive e-mails, an *ACL* link is used. The following steps take place for this purpose:

1. **Service discovery** – The *LMP* uses the **Service Discovery Protocol (SDP)** to discover the services offered by the access point. For example, it checks whether the email service is supported by this access point.
2. **L2CAP channel** – The user's laptop now creates an **L2CAP** channel to communicate with the access point. Either L2CAP channel can be directly used by the email application or an alternative is to run a different protocol such as **RFCOMM** on top of it. In the latter case, existing applications developed for serial ports can be run without any changes using Bluetooth.
3. **Security** – If the access point requires authentication/encryption, it sends a **security request for pairing**. The user needs to know the PIN to respond successfully. We should note that the PIN is not sent over the wireless channel. What is done instead is that another key is generated based on the PIN, and that is sent out. Encryption can also be optionally enabled.
4. **PPP** – Dial-up networking uses the **Point To Point (PPP)** protocol for serial communication over a telephone line. If we use the same protocol here, the same application will now be able to send PPP packets over RFCOMM. Hence, it would work like a serial port. This means that the user can access any applications that s/he can normally access using the Internet, because now the TCP/IP protocol stack can execute in the upper layers.

14.3.3 Piconet and Scatternet

Bluetooth technology is based on the principle that whenever any two devices want to communicate in a wireless fashion, one of them should be the **master** (also called **primary**) and the other should be the **slave** (also called **secondary**). The term *master* does not indicate any special privileges. Instead, it indicates that the master device can decide what Bluetooth clock should be used and how the frequency of communication be used (we shall talk more about it shortly). One master device can have multiple slaves. The master can communicate with all the slaves in a round-robin fashion (i.e., one after the other). Two slaves cannot communicate with each other directly. Only a master-slave pair can communicate with each other. Hence, if two slaves want to communicate with each other, they must form a separate Bluetooth network, and one of them must assume the role of the master.

The group of one master device and its slaves forms what is called a **piconet**. This is shown in Fig. 14.3.

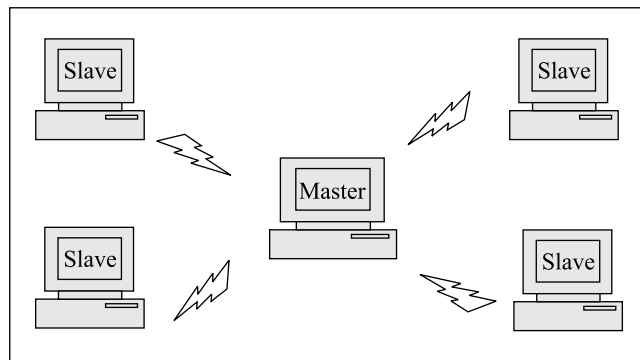


Fig. 14.3  *Piconet*

Multiple piconets connected together create a **scatternet**. This is shown in Fig. 14.4.

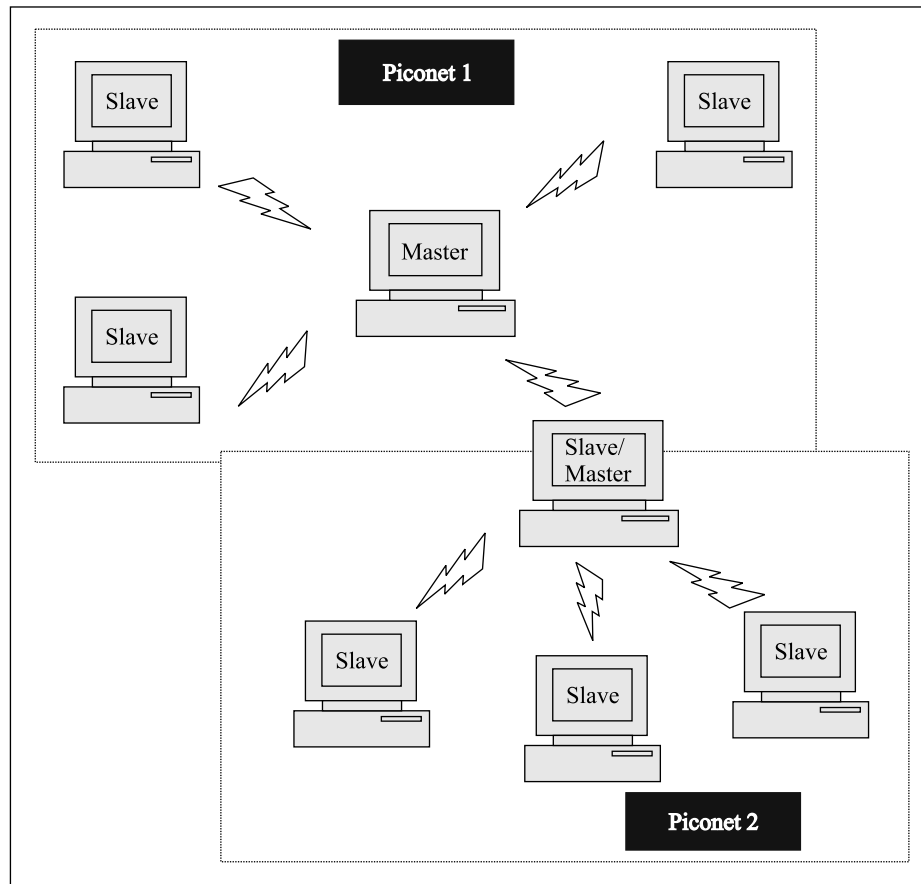


Fig. 14.4 Two piconets forming a scatternet

We should note the following points in this regard:

- A piconet can consist of up to eight devices in a range of ten meters. As mentioned earlier, one of the devices is a master, and the rest are slaves. The slaves synchronize their operations with the master. This means that when the clock pulse of the master device ticks, exactly at the same time, the clock pulses of all the slave devices must also click. Then we say that these slaves' devices are in synchronization with the master device.
- When two or more piconets create a scatternet, two things are of interest with reference to the device shown as *Slave/Master* in Fig. 14.4:
 - A single device can be in two piconets simultaneously. This means that this device is in the overall scatternet.
 - A master device in one piconet can be a slave device in another piconet. For example, in our diagram, the device belonging to both piconets is a slave in piconet 1, but the master in piconet 2.

Piconets are created in an ad-hoc manner, i.e., as and when needed. They are also destroyed whenever their need is over.

14.3.4 Bluetooth Protocol Stack

The Bluetooth protocol stack looks as shown in Fig. 14.5. We have not shown all the complexities in the protocol stack to keep things simple.

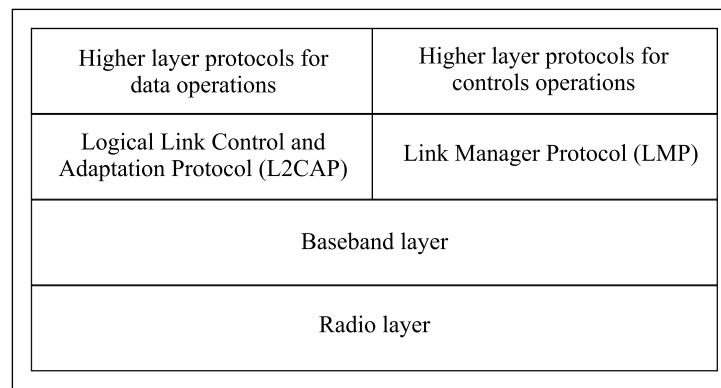


Fig. 14.5 *Bluetooth protocol stack*

Let us have quick overview of the protocols in the stack before we discuss each one of them in detail. As is the case with most protocol stacks, it is easier to examine the protocols from the bottom up.

- **Radio layer** – This layer is similar to the physical layer in other protocol stacks, such as TCP/IP. This layer deals with the air interface (i.e., radio signals) in terms of the frequency to be used, how the frequency hopping should happen, what sort of modulation scheme should be used, how much of power should be consumed, etc.
- **Baseband layer** – This layer is conceptually similar to Medium Access Control (MAC) in other protocols. Here, the issues addressed are connection establishment in a piconet, addressing mechanisms, frame formats, timing of events, etc.
- **Link Manager Protocol (LMP)** – This layer performs the function of setting up of the link between two devices, maintaining it, and other functions such as authentication, encryption, and negotiation of frame sizes.
- **Logical Link Control and Adaptation Protocol (L2CAP)** – This layer allows the higher layer protocols to link up with the baseband layer. It ensures that both connection-oriented as well as connectionless modes are supported.

The layers above the LMP and L2CAP can be any traditional protocol layers (such as IP-TCP, etc.). We shall now discuss these layers in detail.

14.3.5 Radio Layer

As the name suggests, the **radio layer** is one that is responsible for transmission and receipt of radio signals (i.e., wireless signals) between Bluetooth devices. Two Bluetooth devices share a single physical channel for communication. Therefore, they must be tuned to the same radio frequency at

the same time, and they must also be within a nominal range of each other. As mentioned earlier, Bluetooth uses the frequency band of 2.45 GHz. This frequency range or band is reserved for Industrial, Scientific, and Medical (ISM) applications. Like the physical layer of the OSI model, the radio layer here is responsible for the actual transmission of information, but not for deciding when to start or stop the transmission.

The transmission band used by Bluetooth is shared with a number of other devices, such as Wireless LAN (i.e., WiFi) based on the IEEE 802.11 standard, waves generated by microwave ovens, cordless phones, sodium vapour lamps, and so on. As a result, there is a lot of traffic and possibility of collisions of Bluetooth traffic with traffic from these devices. Consequently, the designers of Bluetooth had to devise several mechanisms so as to ensure that Bluetooth traffic does not clash with traffic from these other devices, sharing the frequency spectrum. Hence, they came up with three ideas, viz., frequency hopping, adaptive power control, and short data packets.

Frequency Hopping Spread Spectrum (FHSS)

Imagine that one second is divided into 1600 slots and that a device using Bluetooth can transmit in any of these slots. In other words, there are 1,600 opportunities for transmission for every device per second. That is, we can think of 1,600 opportunities being available for transmission in one second. This concept is shown in Fig. 14.6.

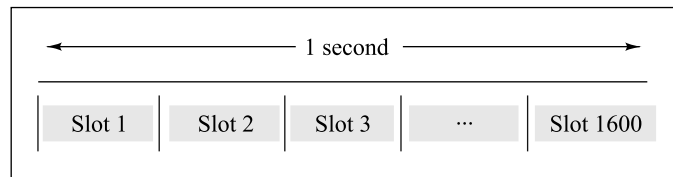


Fig. 14.6 Concept of time slots in Bluetooth

Since one slot occupies $1/1600^{\text{th}}$ of a second, the duration of one slot is 0.000625 seconds, i.e., 625 microseconds (or 625 μs).

Bluetooth uses the concept of **frequency hopping**. In simple terms, this means that the modulation frequency at which the transmission happens keeps changing (i.e., *hopping*). Astonishingly, the frequency change or hop happens for each of these 1600 slots. In other words, if device 1 is sending some data to device 2 in one second, device 1 would use 1600 different frequencies during one second, one after the other.

This technique is called **Frequency Hopping Spread Spectrum (FHSS)**, and it ensures that when two devices in two different piconets transmit, they do not use the same frequency, thus avoiding a possible collision. Also, because the transmission from the same device keeps happening on changing frequencies, even if the first of those transmissions fails, say, because that frequency is blocked, the next one is likely to go through because it would happen on a different frequency.

Since a device changes the modulation frequency 1,600 times a second, it uses that frequency only for 625 microseconds (because $1/1600 = 0.000625$).

14.3.6 Baseband Layer

We can compare the baseband layer with the Medium Access Control (MAC) sublayer in the case of a Local Area Network (LAN). As mentioned earlier, the total bandwidth available is divided into 79 physical channels. Each channel has a bandwidth of 1 MHz.

Usage of Physical Channels

Let us take the example of Bluetooth as implemented in the US, Europe, and many other countries. We will begin with the initial frequency of 2.4 GHz as the starting point. The specifications say that some initial space should be left unused, and that a device that wants to start transmission should start at the frequency of 2.402 GHz. The second channel should use the next available frequency for transmission, which is 2.403 GHz. The third channel should use a frequency of 2.404 GHz, and so on. This is shown in Fig. 14.7.

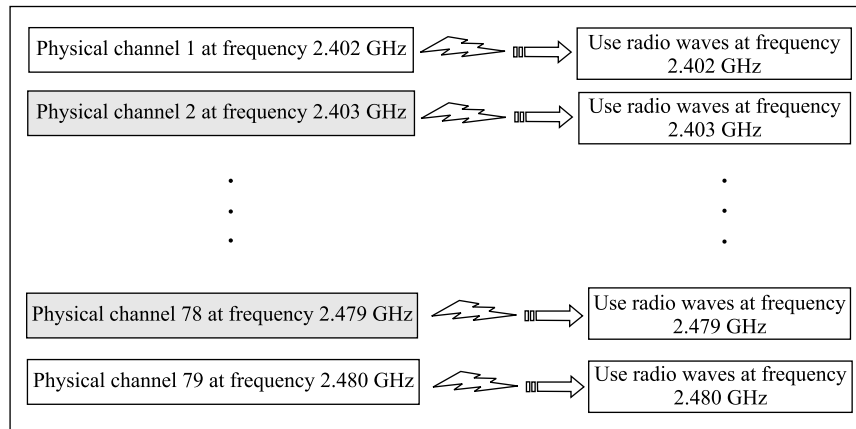


Fig. 14.7 Physical channels concept

In general, we can say that the n th channel will use the frequency of $2.402 + n$ GHz, where n varies from 0 to 78. How is this relevant? Let us take an example as shown in Fig. 14.8.

Let us consider two piconets: piconet 1 and piconet 2. Piconet 1 contains four devices, say, device 1 to device 4, and piconet 2 contains three devices, say, device 5 to device 7.

Let us talk about piconet 1.

1. Suppose that in piconet 1, device 1 (the master) wants to send some data to device 2.
2. Suppose that device 1 starts sending data with the initial frequency of 2.402 GHz. Device 2 would also tune to this frequency.
3. After 625 μ s elapse, both device 1 and device 2 tune (i.e., *hop*) and would change their carrier frequency based on a random-like (or pseudo-random) algorithm, so that they use a different physical channel. What would this new frequency be, is something that cannot be predicted. Device 1, which is assuming the role of the master, would decide this.
4. This would continue for the entire duration of transmission between device 1 and device 2.

Now let us talk about piconet 2.

1. Suppose that in piconet 2, device 7 (the master) wants to send some data to device 6.
2. Suppose that device 7 starts sending data with a different physical channel that has the initial frequency of 2.409 GHz. Device 2 would also tune to this frequency.
3. After 625 μ s elapse, both device 7 and device 6 tune (i.e., *hop*) and would change their carrier frequency based on a random-like (or pseudo-random) algorithm, so that they use a different physical channel. Note that this would still be different from the hop done by devices on piconet 1, and hence a collision is unlikely.

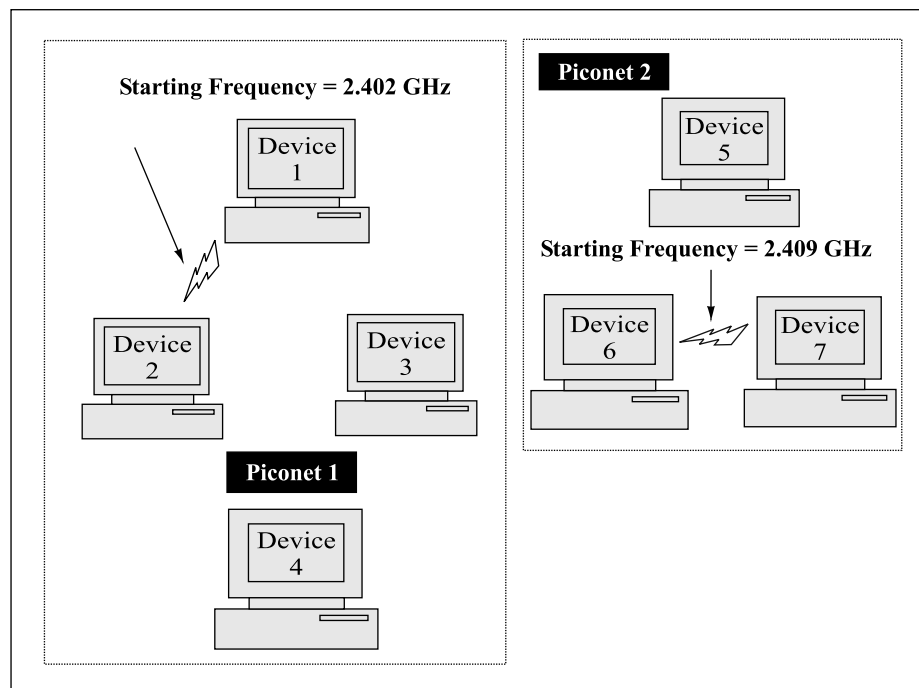


Fig. 14.8 Usage of different physical channels

Frames and Slots

We need to get certain basic definitions right before proceeding.

- **Frame** is a unit of data exchange (i.e., it contains the data to be sent from one device to another, along with the control information).
- **Slot** is the time allocated for the duration of 625 micro seconds, as discussed earlier.

In other words, data is sent by one device to another in the form of frames. One frame can be sent inside 1, 3 or 5 slots. The simplest case is if that the frame fits inside one slot. However, if a frame is bigger than the capacity of a single slot, the frame can use either 3 or 5 slots. If a slot contains a frame that fits inside that single slot, the hopping frequency is 625 microseconds. However, if a slot contains a frame that does not fit into one slot, i.e., a slot contains only a part of a larger frame, then the hopping happens after all the 3 or 5 slots (depending on the situation) related to that frame are completed.

Also, if a frame occupies multiple slots (i.e., 3 or 5), then while the sender sends the entire frame in these 3 or 5 slots, the receiver must wait. In other words, in such a case, the entire transmission happens from the sender's side to the receiver's side over multiple slots, while the receiver just keeps receiving this transmission.

Conceptually, this is shown in Fig. 14.9.

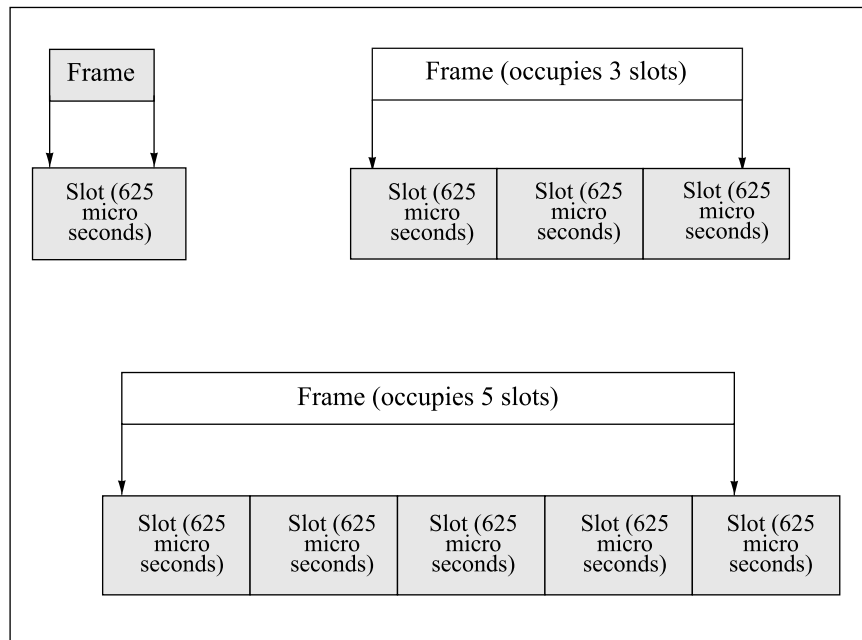


Fig. 14.9 *Frames and slots*

We know that each physical channel has a bandwidth of 1 MHz. In Bluetooth, roughly one bit is sent per Hertz. Therefore, 1 million bits can be sent in one million Hertz, leading to a data transmission rate of 1 Mbps. However, plenty of the available bandwidth is consumed by overheads.

Each slot lasts for 625 microseconds. However, frequency hopping means that out of the 625 micro seconds, about 259 microseconds are wasted per hop during the lifetime of a slot. Thus, for a 1-slot frame, roughly 366 of the 625 bits are available (because $625 - 259 = 366$). Of these, 126 bits consume access code and header. This leaves just 240 bits of actual contents per slot.

14.4 802.11 WIRELESS LAN

14.4.1 Basic Concepts

The term **Wireless Local Area Network (Wireless LAN or W-LAN)** is now very popular. However, even more popular is its other name, which is **Wireless Fidelity (Wi-Fi)**. We know that **Ethernet** is the most popular wired LAN technology. Similarly, standards are in use for wireless LAN as well.

In wired LAN, every computer has a **Network Interface Card (NIC)** and a **transmitter–receiver (transceiver)** with which it communicates with the external world. Same is the case with computers

connected to a wireless LAN. Here, the NIC works with a wireless (i.e., radio waves-based) modem and an antenna for the actual transmission. Therefore, the transceiver roughly maps to the radio wave modem and antenna. Usually, there is an antenna on the ceiling of the floor where the wireless LAN is created. Every computer on the wireless network communicates with this antenna via radio waves in the air. The **IEEE 802.11** standard has been devised for wireless LAN technology. The standard is based on the principles of the wired LAN (Ethernet) standard **IEEE 802.3**, with exceptions made, as appropriate, because we now talk about a wireless network.

The overall view of the 802.11 protocol suite is shown in Fig. 14.10.

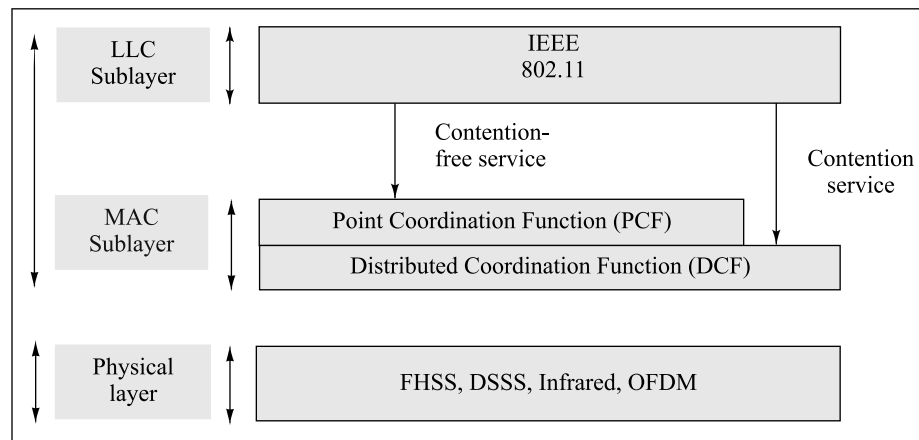


Fig. 14.10 802.11 protocol stack

14.4.2 Challenges in Wireless LAN

Wireless networks and standards pose several challenges, right from naming to everything else, which can be summarized as follows:

1. What name should be given to this standard was not clear. Earlier standards were named/numbered as 802.1, 802.2 ... Hence, this standard got the name 802.11.
2. There was a challenge to find frequency range for wireless LANs that would work universally, across the world. This was because if someone takes her/his laptop from one place to another (say, to another country), the basic specifications of wireless networking cannot be expected to be completely different.
3. Since radio signals have a limited range, it was necessary to find frequencies that would allow communication for shorter distances, but not for so short distances such as done in the case of infrared or Bluetooth.
4. It was important to devise mechanisms to protect users' privacy. This means that good security mechanisms had to be built in the specifications.
5. While we discuss all other issues, limited battery life of laptops and other devices such as mobile phones that use the wireless LAN technology needs to be kept in mind. Otherwise, heavy specifications may demand too much of processing power, draining the laptops and other devices.

6. The specifications had to ensure that whatever happens, the end result has to be economically viable for the users of the wireless networks and also for the companies that set them up in the first place.
7. Finally, some people were asking if wireless networking can cause cancer. Hence, the *suspicious* frequency ranges had to be avoided at any cost.

14.4.3 Access Point (AP)

The origins of wireless LANs are quite simple. As wireless technology started becoming popular, people thought of using laptops in offices, with an ability to move around and yet being connected to the network at all times. These days, people want to connect to networks and to the Internet via their mobile phone. This led to the development of various standards for wireless LAN. Like the way it happens at the beginning of any new technology, many incompatible standards got developed for wireless LAN, too. And hence an effort was made to standardize and unify all of them. This effort concluded in the IEEE 802.11 specification.

In general, a wireless LAN has an **Access Point (AP)** (also called a **base station**), which is the main controller of the network, like a switch in an Ethernet network. Of course, a wireless network can also be constructed *without* an access point. In that case, the wireless network is called an **ad-hoc network** or **peer-to-peer network**.

For now, we shall concentrate on the network involving an access point, an idea that is captured in Fig. 14.11.

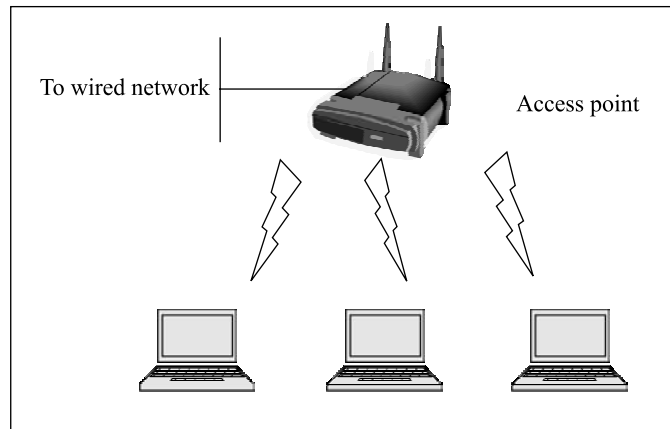


Fig. 14.11 Access Point (AP) concept

The access point is nothing but a wireless router. It has a transceiver and connects on one side to the wireless devices and on the other to a wired network such as Ethernet. When installed at home, it usually connects to wireless devices on one side and to a cable modem or a DSL modem on the other side. Computers that need to connect to a wireless LAN need to have a wireless NIC. At any given point of time, a wireless device or host must belong to only one access point. Otherwise, there would be confusion. Because of this arrangement, other access points know to which base station any host belongs to at any moment, and redirect traffic meant for that host to that specific access point.

Within the 802.11 standard, there are further complications. Variants have emerged, and they are incompatible with each other in some ways or the other. These variants are known by adding an alphabet suffix to the standard name, and hence we call them as 802.11a, 802.11b... 802.11g. They differ in the frequency range used as well as the speed that is supported. Table 14.2 shows the popular variations of the 802.11 standard.

Table 14.2 Popular variations of the 802.11 standard

Variation	Speed	Frequency used	Details
802.11	1–2 Mbps	2.4 GHz	Original standard, launched in 1997, not in practical use anymore
802.11a	Up to 54 Mbps	5 GHz	Higher frequency, less penetration, less susceptible to interference
802.11b	Up to 11 Mbps	2.4 GHz	Popular earlier, now getting replaced by 802.11g
802.11g	Up to 54 Mbps	2.4 GHz	Usually compatible with 802.11B equipment

What is interesting to note here is that the frequency of transmission used by the most popular 802.11g variation is the same as that of 802.11b (i.e., 2.4 GHz), but its speed of transmission is the same as that of the 802.11a standard (i.e., 54 Mbps). This is done so that 802.11g networks are compatible with 802.11b. The reason being 802.11g was supposed to replace 802.11b. However, there was no point in replacing 802.11b if among other improvements the speed offered by 802.11g was not also higher. And hence, while retaining the frequency, the speed of transmission was increased as much as possible.

14.4.4 IEEE 802.11 Medium Access Control (MAC) Layer

A wireless network is made up of Access Points (APs) or base stations placed at strategic locations around a building. The base stations themselves are interconnected using copper wire or optical fiber. Hence, some authors compare every room in the building with a cell in a cellular network, and the whole building to the entire cellular system. Bandwidth supported, as we have mentioned earlier, is between 11 and 54 Mbps.

From the transmission point of view, in the case of wired networks, it is an acceptable thought that after a frame is sent, it reaches the destination without any problems in majority of the cases. However, this cannot be said about wireless networks. Interference, noise, and multipath fading are some of the most common reasons of failure. Hence, instead of relying on the underlying medium, wireless LAN takes a different approach. Elaborate acknowledgement mechanisms are built-in to achieve better coordination and guarantee of delivery. Here, the receiving host must send a positive acknowledgment (ACK) when it receives a frame correctly. Also, the complete package of *DATA* and *ACK* frames is together considered as a logical transaction. In other words, when host A sends a *DATA* frame to host B, host B must respond with an *ACK*. While both *DATA* and *ACK* operations of these two devices are completed, other hosts in the wireless network cannot transmit. They are *locked out*. This concept is shown in Fig. 14.12.

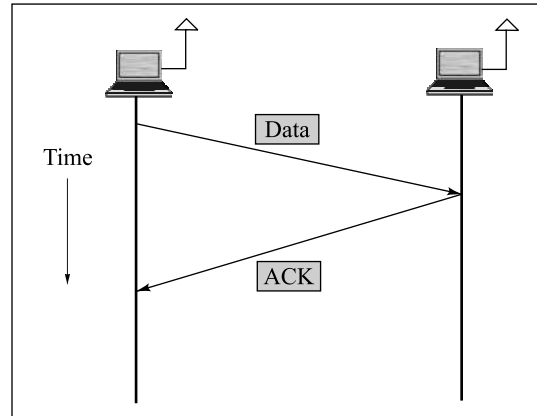


Fig. 14.12 DATA and ACK example

14.4.5 Medium Access

We have discussed earlier that CSMA/CD cannot be used in wireless LAN. The idea of listening to the medium to judge if it is idle and then transmitting does not work here. There are two interesting problems that prevent the usage of this mechanism, as illustrated in Fig. 14.13.

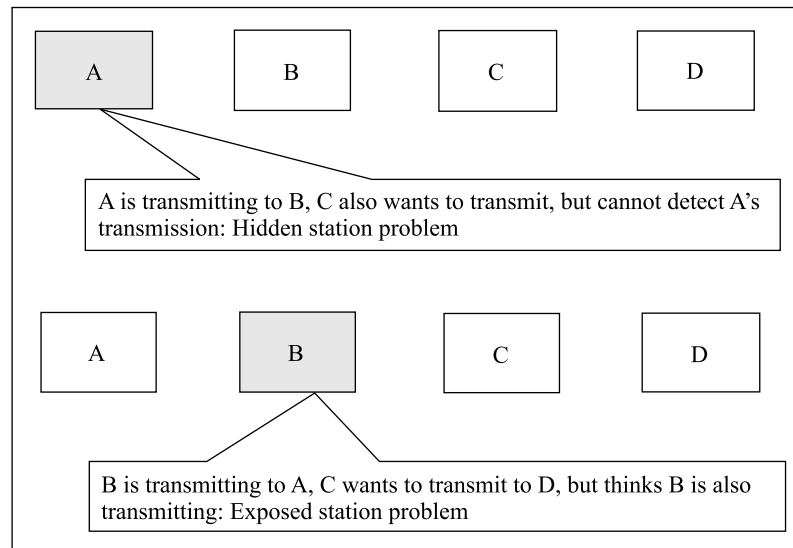


Fig. 14.13 Hidden station problem and exposed station problem

Let us understand these two problems now.

- Hidden station problem** – The problem of a host (same as *station*) not being able to detect the presence and therefore the transmission of another host is called **hidden station problem**. In the diagram (upper part), host C wants to transmit to host B, but cannot detect the fact that host A is already transmitting to host B. In this way, host A is *hidden* from host C.

- **Exposed station problem** – Sometimes, the opposite of what we discussed earlier happens. A host thinks that the medium is in use, when it is actually free. This conclusion leads the host to behave that it must wait, and therefore, does not transmit, effectively wasting bandwidth. This problem is called **exposed station problem**. In the diagram (lower part), host B is transmitting to host A. At the same time, host C wants to send some data to host D. This communication (C–D) has actually nothing to do with the currently ongoing transmission (B–A). But C does not know this – and waits for no reason. In other words, B and C are unnecessarily *exposed* to each other.

This leads us to the following very interesting observations.

1. In a wired LAN, the whole emphasis is on whether the sender can send. In other words, the CSMA/CD technique worries about interference at the sender's end. If there is already a transmission happening, the sender waits.
2. We cannot replicate this CSMA/CD model in the case of a wireless LAN, for the reasons and problems outlined above.
3. Alternatively, we can say that the point of importance in the case of a wireless LAN is *not* the possible interference at the sender's side, but it is the possible interference at the receiver's end. As long as the receiver can receive whatever the sender sends, that is all that matters. Interference at the sender's end may or may not matter.
4. Therefore, technically, in a wireless LAN, multiple concurrent transmissions can happen if the destination in each case is different, and if various destinations are all out of range of one another.

This brings us to the point about the two types of carrier sensing, as shown in Fig. 14.14.

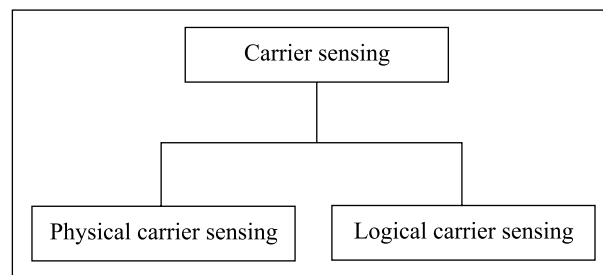


Fig. 14.14 Types of carrier sensing

Let us understand the types of carrier sensing now.

- **Physical carrier sensing** – The concept of **physical carrier sensing** refers to the idea that the physical layer is responsible for carrier sensing. In other words, the mechanisms required to check whether the physical medium (in this case, air) is free from any competing transmissions (in this case, radio signals corresponding to the frames sent by other hosts in the same wireless network) needs to be built into the physical layer. This means that the transceivers that need to be built into the wireless hosts need the capabilities of both transmitting and receiving, taking into account the *hidden station problem*. This is not impossible, but is certainly very difficult and quite expensive.

- **Logical carrier sensing** – In **logical carrier sensing** also called **virtual carrier sensing**, there is no need to build any mechanisms for physically checking if the transmission medium is idle. Instead, this job is performed logically. This is done at the data link layer. Although we shall discuss this in detail shortly, the basic idea can be summarized as follows:

1. Whenever any host wants to send a data frame to another host on the wireless LAN, it sets a timer value that is communicated to all the hosts on that network. This timer value indicates the time period for which the host likely to occupy the transmission medium. In other words, it tells all the other hosts in the wireless LAN to wait for an amount of time equal to the timer value set.
2. All the other hosts in the wireless LAN note the value of the timer and do not transmit anything until the timer expires.
3. The sender is expected to complete its transmission within this timer duration.

Therefore, as we can see, there is a logical carrier sensing here. In other words, no other host is permitted to transmit based on the value of the timer field, which is an implicit lock put on it.

Quite clearly, it is far more beneficial and simpler to use logical carrier sensing mechanism in the case of a wireless LAN, which is what is done in practice.

Wireless networks also suffer from the problem of **near and far terminals**. In wired networks, the physical location of a host does not matter. All hosts are treated as equal. However, in the case of a wireless network, the signals are sent in air, like human voice when speaking. We know that these signals, like human voice, get increasingly weak or distorted with the distance that they travel. Hence, the hosts that are *near* to the sender have an advantage as compared to the hosts that are *far away*.

14.4.6 MAC Sublayers

The 802.11 MAC layer is composed of two sublayers, as listed in Fig. 14.15.

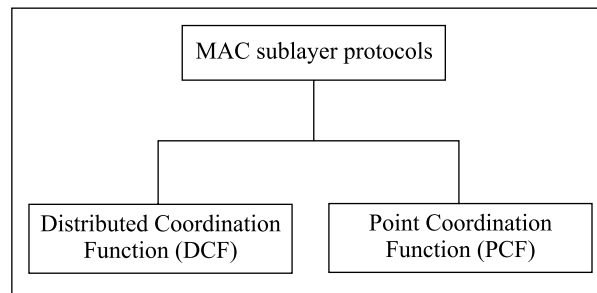


Fig. 14.15 MAC sublayers in 802.11 wireless LAN

1. **Distributed Coordination Function (DCF)** – The **Distributed Coordination Function (DCF)** mechanism is the most popular mechanism for medium access in the case of wireless LANs. DCF provides a **contention-oriented service**.
2. **Point Coordination Function (PCF)** – The **Point Coordination Function (PCF)** mechanism is not very common. It provides a **contention-free service**.

SUMMARY

IEEE has standardized a number of operations and mechanisms pertaining to both wired and wireless networks. They are known by the 'IEEE 802' names, for example, IEEE 802.11 is for wireless Local Area Networks. Within wireless networks, we can have personal, local, metropolitan, and wide area networks.

The smallest of all possible wireless networks of any practical use is Bluetooth. Bluetooth allows two or more devices to communicate with each other at short distances. Examples of Bluetooth communication are a wireless mouse connecting to a computer or data being sent between two mobile phones, etc. Here, one of the two devices assumes the role of a master, and the other device becomes a slave. Various configurations are possible.

The 802.11 standard for Wireless LANs (also called Wi-Fi) specifies what should happen inside a network of computers inside a building. Such a network does not use wires. Although wired and wireless LANs look similar, there are several technical and architectural differences between the two. In Wireless LANs, there is no guarantee that hosts would remain stationary and also that they would recognize their surroundings. Hence, a number of challenges emerge, which need to be resolved.

KEY TERMS AND CONCEPTS

Access Point (AP)	Logical carrier sensing
Ad-hoc network	Logical Link Control and Adaptation Protocol (L2CAP)
Base station	Master
Baseband layer	Near and far terminals
Bluetooth	Network Interface Card (NIC)
Clock offset	Paging request
Contention-free service	Pairing
Contention-oriented service	Peer-to-peer network
Distributed Coordination Function (DCF)	Physical carrier sensing
Exposed station problem	Piconet
Frame	Point Coordination Function (PCF)
Frequency hop	Point To Point (PPP)
Frequency hopping	Primary
Frequency Hopping Spread Spectrum (FHSS)	Radio layer
Hidden station problem	RFCOMM
IEEE 802.1	Scatternet
IEEE 802.11	Secondary
IEEE 802.11	Security request
IEEE 802.2	Service Discovery Protocol (SDP)
IEEE 802.3	Slave
IEEE 802.4	Slot
IEEE 802.5	Transmitter-receiver (Transceiver)
Link Manager Protocol (LMP)	

Virtual carrier sensing	Wireless Personal Area Networks (Wireless PAN)
Wireless Local Area Network (Wireless LAN or W-LAN)	Wireless Wide Area Networks (Wireless WAN)
Wireless Local Area Networks (Wireless LAN)	
Wireless Metropolitan Area Networks (Wireless MAN)	

QUESTIONS

True/False

1. Infrared communication is used in Wi-Fi.
2. The IEEE 802.1 standard covers Wi-Fi.
3. The IEEE 802.3 standard is a specification for Ethernet.
4. Ethernet is an example of a wireless network.
5. Bluetooth is a PAN.
6. Bluetooth can replace Ethernet.
7. In Bluetooth, we can have m:n communication.
8. We can have a piconet in Bluetooth.
9. Wi-Fi is another name for 802.11 networks.
10. Hidden station problem is a wired network problem.
11. Exposed station problem is a wireless network problem.
12. Wireless networks are not possible in LAN.

Multiple-Choice Questions

1. The standard for Ethernet is called _____.
 (a) IEEE 802.11 (b) IEEE 802.1
 (c) IEEE 802.4 (d) IEEE 802.3
2. The standard for Bluetooth is called _____.
 (a) IEEE 802.15 (b) IEEE 802.11
 (c) IEEE 802.1 (d) IEEE 802.2
3. The standard for wireless LAN is called _____.
 (a) IEEE 802.15 (b) IEEE 802.11
 (c) IEEE 802.2 (d) IEEE 802.5
4. Bluetooth is an example of _____.
 (a) WAN (b) MAN
 (c) LAN (d) PAN
5. WiMax is an example of _____.
 (a) PAN (b) LAN
 (c) MAN (d) WAN
6. In Bluetooth, the _____ protocol is used to discover the services offered by an Access Point (AP).
 (a) SDP (b) L2CAP
 (c) PPP (d) RFCOMM

7. A piconet consists of up to _____ devices in a range of 10 meters.
(a) 4 (b) 6
(c) 8 (d) 10
8. The bottommost layer in Bluetooth protocol stack is the _____.
(a) radio layer (b) baseband layer
(c) L2CAP (d) LMP
9. The problem in which host thinks that the medium is in use, when it is actually free is called _____.
(a) near terminal (b) far terminal
(c) hidden station problem (d) exposed station problem
10. A situation where a host thinks that it must wait, and therefore, does not transmit effectively wasting bandwidth is called _____.
(a) near terminal (b) far terminal
(c) hidden station problem (d) exposed station problem

Detailed Questions

1. Write a note on IEEE 802 standards.
2. Describe any two of IEEE 802.1, IEEE 802.2, IEEE 802.3, and IEEE 802.4.
3. What is Bluetooth?
4. Explain practical usage of Bluetooth.
5. What is a piconet and scatternet?
6. What are a master and a slave in Bluetooth?
7. What is an AP in an IEEE 802.11 network?
8. Explain how communication happens in a WLAN.
9. What is the exposed station problem in a wireless LAN?
10. What is the hidden station problem in a wireless LAN?

15 Internetworking Concepts, Devices, Internet Basics, History and Architecture



15.0 INTRODUCTION

In the previous chapters, we have studied the basic principles of computer networking. Let us now study another extremely important concept of connecting many such computer networks together. This is called **internetworking**. A network of computer networks is called an **internetwork** or simply, **internet** (note the lowercase *i*). The worldwide Internet (note the uppercase *I*) is an example of the internetworking technology. The Internet, as we have seen, is a huge network of computer networks. The following sections describe the motivations behind such a technology, as well as how it actually works.

When two or more devices have to be connected for sharing data or resources or exchanging messages, we call it networking. When two networks need to be connected for the same purpose, we call it internetworking. The main difference between networking and internetworking is that in case of networking all the devices are compatible with each other (e.g., hosts in a LAN); it may or may not be the case in internetworking. When we want to connect two or more networks to form an internetwork, it is quite possible that the networks are incompatible with each other in many respects. For instance, we might want to connect an Ethernet LAN with a Token Ring LAN and a WAN. All the three types of networks are quite different from each other. They differ in terms of their topologies, signaling, transmission mechanism as well as wiring, etc. Therefore, the challenge in internetworking is more in terms of handling these incompatibilities and bringing all the incompatible networks to a common platform.

In this chapter, we shall discuss various connecting devices that are required to facilitate networking and internetworking. These devices form the backbones of any network or internetwork (abbreviated as internet, which is different from the worldwide network of networks, i.e., the Internet: note the difference in case of the letter).

The Internet has been acknowledged as one of the greatest developments of the 20th century. In fact, people talk about the Internet in the same way as the revolutionary inventions such as electricity and the printing press among others. The Internet is here to stay even if the dot coms have perished. In this chapter, we shall look at the fundamentals of the Internet technology. More specifically, we shall study how the Internet is organized and how it works. We shall also take a look at the historical perspective of the Internet.

We shall first study the basic concepts behind the Internet and then see how the different components of the Internet work. The Internet is basically the world's largest network of computer networks. Many different kinds of applications run over the Internet. We shall discuss those in detail. The **Transmission Control Protocol/Internet Protocol (TCP/IP)** protocol is the backbone of the Internet. We shall see how it works. Finally, we will study what happens when a user is **browsing** the Internet.

15.1 WHY INTERNETWORKING?

The main reason for having an internet is that each computer network is designed with a specific task in mind. For example, a LAN is typically used to connect computers in a smaller area (such as an office) and it provides fast communication between these computers. On the other hand, WAN technologies are used for communication over longer distances. As a result, networks become specialized entities. Moreover, a large organization having diversifying needs has multiple networks. In many cases, these networks do not use the same technology in terms of the hardware as well as communication protocols.

Consequently, a computer can only communicate with other computers attached to the same network. As more and more organizations had multiple computer networks in the 1970s, this became a major issue. Computer networks became small islands! In many cases, an employee had to physically move for using computers connected to different networks. For example, to print a document, the employee would need to use a computer that was connected to a print server. Similarly for accessing a file on another network, the employee had to use a computer on that network, and so on. Clearly, this was a nuisance. This affected productivity, as people did not like to move around for performing trivial tasks.

As a result, the concept of **universal service** came into being. In simple terms, it means that there is no dependence on the underlying physical technology, or on the fact that there were many separate physical networks. Like a telephone network, people wanted a single computer network in their organization. A user should be able to print a document or send a message to any other user from her/his computer, without having to use a separate computer on another network for each such task. For this to be possible, all computer networks should be connected together. This means that there should be a network of physically separate networks. This forms the basis of internetworking.

15.2 PROBLEMS IN INTERNETWORKING

It is fine to think of a network of computer networks or an internet in theory. However, one must also remember that organizations invest so much when they build computer networks in terms of cost as well as infrastructure (cabling, providing space in the building for it, etc.). Therefore, they would want to reuse their existing infrastructure rather than creating everything from the scratch. However, there are problems in this. Electrical as well as software incompatibility makes it impossible to form a network merely by interconnecting wires from two networks. For example, one network could represent a binary 0 by -5 volts, whereas another network could represent it by $+5$ volts. Similarly, one network could use a packet size of, say, 128 bytes, whereas another could use 256-byte packets. The method of acknowledgement or error detection/recovery could also be entirely different. There could be many more such differences like routing algorithms, etc.

Thus, any two networks cannot directly communicate with each other by just connecting a wire between them. Since there are many incompatible networking technologies, the problem becomes more acute. An organization could have many networks of different types. This means that there is a large amount of disagreement between the networks in terms of signaling, data representation and error detection, recovery, etc. Therefore, the concept of universal service through internetworking is not simple to achieve, although it is highly desirable.

15.3 DEALING WITH INCOMPATIBILITY ISSUES

In spite of the problems mentioned earlier, computer scientists have found out a mechanism by which computer networks can be connected together to form an internet. The incompatibility issues are addressed in two respects the following text.

15.3.1 Hardware Issues

At the hardware level, some additional hardware is used to connect physically distinct computer networks. This hardware component is most commonly a router. A router is a special-purpose computer that is used specifically for internetworking purposes. A router has a processor (CPU) and memory like any other computer. However, it has more than one I/O interface that allows it to connect to multiple computer networks. From a network's point of view, connecting to a router is not extraordinary in any way. A network connects to a router in the same way as it connects to any other computer. A router connects two or more computer networks, as shown in Fig. 15.1.

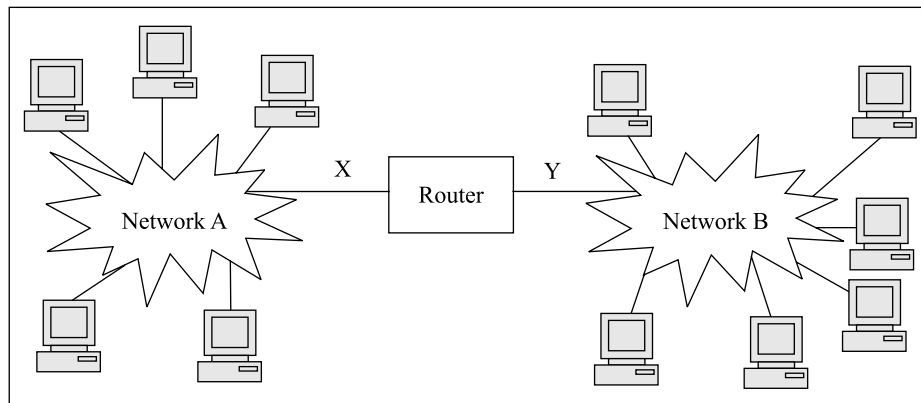


Fig. 15.1 *A router connects two or more computer networks together*

A network has many computers or nodes attached to it. Therefore, an address of a node or a computer could be treated as network id + node id. Each node has a Network Interface Card (NIC), which has this address hardcoded into it. If a router is treated as yet another computer by the network, it means that the router basically has two addresses – one for each network, at points X and Y as shown in the figure.

The router is a special computer that has two Network Interface Cards (NICs), which connect to these two networks. These two NICs correspond to the two physical addresses that the router has.

The most important point in this discussion is that a router can connect incompatible networks. That is, networks A and B in the figure could be both LANs of the same or different types, both WANs of the same or different types; or one of them could be a LAN and the other a WAN, etc. A router has the capability to connect them together. How is this possible? For this, a router has the necessary hardware (NIC for each type of network) as well as software (protocols) that make it possible. Moreover, even if both A and B in the figure are of the same category, say LANs, they could internally use different technology (one could use Ethernet and another could use FDDI). The router handles all these incompatibilities as well. Again, this is possible because of the hardware and software contained by a router. The point is, A and B in the figure could be arbitrary networks. However, the router would still be able to interconnect them.

Interestingly, the Internet (note the uppercase I) looks as shown in Fig. 15.2.

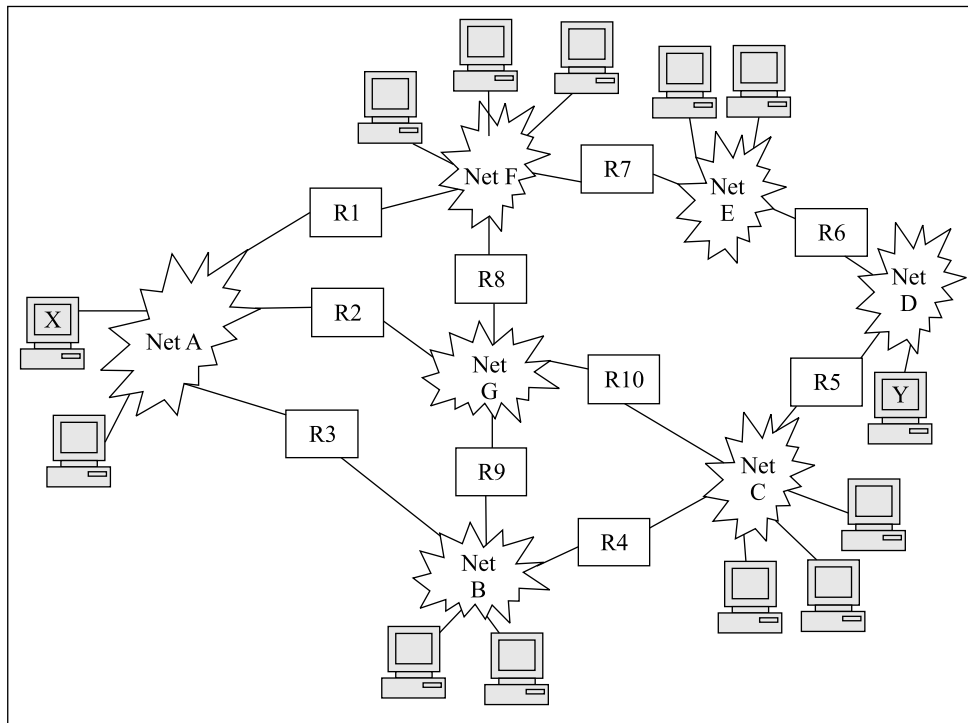


Fig. 15.2 A portion of the Internet

The figure shows seven networks connected by ten routers. Network A could be an Ethernet, network B could be an FDDI, and network C could be a Token Ring, whereas network G could be a WAN. A router connects two networks through two NICs that are contained by each such router.

If computer X on network A wants to send a message to computer Y on network D, the message can be sent in different routes or paths such as

1. X – Net A – R2 – Net G – R10 – Net C – R5 – Net D – Y
2. X – Net A – R1 – Net F – R7 – Net E – R6 – Net D – Y
3. X – Net A – R3 – Net B – R4 – Net C – R5 – Net D – Y

Many more routes also exist.

The router is responsible for routing the packets to the destination. To do this, the software computes the routing algorithm, and based on this, each router stores the routing table, which states for each destination, the next hop, to which the packet is to be sent. It is for this reason that the router is supposed to act at the network layer of the OSI model. It neither examines the contents of the packet, nor tries to interpret them. Figure 15.3 shows this.

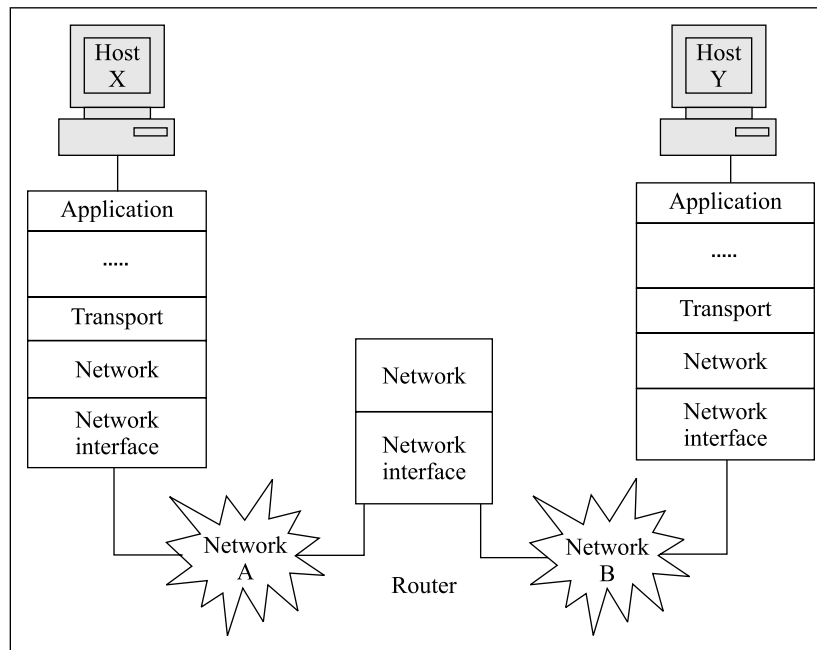


Fig. 15.3 Router is at the network layer of the OSI model

15.3.2 Software Issues

At the software level, routers must agree about the way in which information from the source computer on one network would be transmitted to the destination computer on a different network. Since this information is likely to travel via one or more routers, there must be a pre-specified standard to which all routers must conform. This task is not easy. Packet formats and addressing mechanisms used by the underlying networks may not be the same. Does the router actually perform the conversion and re-conversion of the packets corresponding to the different network formats? Though not impossible, this approach is very difficult and cumbersome. What is done is to define a standard packet format in which the sender breaks down the original message. We will study this later. Therefore, some networking protocols are required that can standardize communication between incompatible networks. Only then, the concept of universal service can be truly realized. In the case of all Internet communications, the TCP/IP suite of protocols makes this possible.

The basic idea is that TCP/IP defines a packet size, routing algorithms, error control methods, etc., universally. Let us refer to Fig. 15.2 again. If node X wants to send some message to node Y by route number 1 given above (X – Net A – R2 – Net G – R10 – Net C – R5 – Net D – Y), the following happens; imagining that Net A is Ethernet and Net G is Token Ring:

1. The message is broken down into the packets as per the TCP/IP protocol. Each packet has the source and destination addresses of X and Y.
2. Each packet is broken down or inserted into the Ethernet frame. Ethernet frame can be carried only on the Ethernet network (in this case, Net A). The TCP/IP packet along with its final source/destination addresses (of X and Y) is enclosed within an Ethernet frame, which has additional source and destination addresses, which are physical addresses on the same network (of X and R2 as both are on Net A). After this, the CRC is computed and appended to the Ethernet frame.
3. Both, node X as well as R2 are on Net A, which is Ethernet. Thus, the frame travels from X to R2 using CSMA/CD, using the Ethernet source/destination addresses of X and R2 .
4. At R2, the CRC is checked, the Ethernet header dropped, and the original TCP/IP packet recovered. It contains the final source and destination addresses of X and Y.
5. From the destination address, routing algorithm is used to find out the *next hop*, which is R10, in this case. We know that both R2 and R10 are on the Token Ring network Net G.
6. Net G is a Token Ring. Therefore, R2, which knows this fact, puts this TCP/IP packet as data in the Token Ring frame format after adding the header, etc. Here also, the TCP/IP packet, which contains the final addresses of X and Y is encapsulated in the Token Ring frame, which has additional source and destination addresses of R2 and R10, respectively for transporting the packet from R2 to R10 on the Token Ring, etc.
7. Like before, R2 as well as R10 are on Token Ring using the Token Ring source/destination addresses of R2 and R10. Thus, the packet reaches R10, etc.
8. This process repeats until the packet reaches Y. At Y, the header is removed to get the original TCP/IP packet. The destination address is verified and the packet is stored.
9. After all the packets are received at Y, the TCP/IP at Y ensures the error free receipt of all packets of the message and then passes it on to the application layer at Y.

This is how TCP/IP solves the problem of connecting heterogeneous networks seamlessly.

15.4 A VIRTUAL NETWORK

The Internet software makes it appear that there is a single, seamless system of communication to which many computers are attached. The internal details of many real, actual networks connecting together to form it are hidden, and instead, it appears to be a single, large network. Every computer on the Internet has an address assigned to it. This is like the postal address assigned to a home. Using this address, any user can send packets to any other computer on the Internet. The users of the Internet do not have to be bothered about the internal structure of the physical networks, their interconnection, routing decisions, or the presence of routers themselves. Thus, an illusion of a **virtual network** is created. This is an abstracted view presented to a common user, who is not interested in knowing the internal organization of the communication system. For example, a telephone user simply wants to dial someone's number and talk with that person instead of knowing how the signaling system works or how many telephone exchanges exist in the system and how they function. Similarly, an Internet user is merely interested in communicating with another user of the Internet, using the computer address of the other user, or s/he is interested in using the services on that computer.

The concept of a virtual network is very important. It ensures that different computer networks can not only be connected together, but also be looked upon and used as a single network. This

forms the basis of the biggest network of networks, i.e., the Internet. This concept is illustrated in Fig. 15.4. The figure shows the illusion of a single, large virtual network corresponding to the real network (shown in Fig. 15.2).

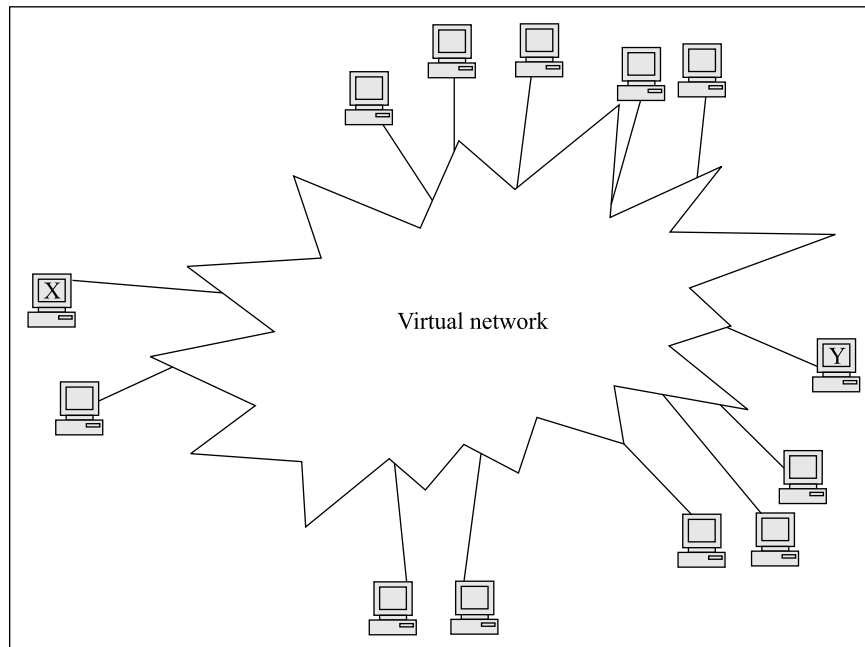


Fig. 15.4 *The Internet is a virtual network of computer networks*

15.5 INTERNETWORKING DEVICES

At a high level, the **connecting devices** can be classified into **networking devices** and **inter-networking devices**. Each of them has another level of classification, as shown in Fig. 15.5. We have discussed routers in brief in the previous chapters.

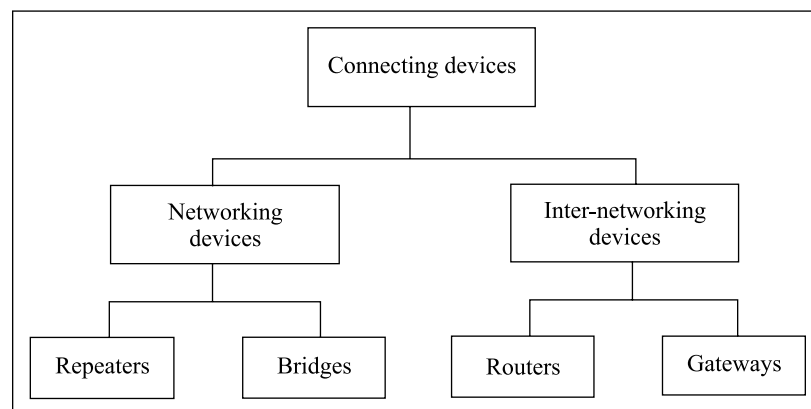


Fig. 15.5 *Connecting devices*

Let us summarize these devices first as shown in Fig. 15.6, before we take a detailed look at each of them.

Device	Purpose	Present in which OSI Layer
Repeaters	Electrical specifications of a signal	Physical
Bridges	Addressing protocols	Data link
Routers	Internetworking between compatible networks	Network
Gateways	Translation services between incompatible networks	All

Fig. 15.6 Summary of networking devices

Note that in each of the last three cases, the device is present in the layer mentioned in the table, as well as one level below it. That is, a bridge is present in the data link layer as well as the physical layer. A repeater is already at the lowest OSI layer (i.e., the physical layer), and therefore, it is present in that layer only.

15.6 REPEATERS

We have discussed **repeaters** earlier. A repeater, also called a **regenerator**, is an electronic device, which simply regenerates a signal. It works at the physical layer of the OSI protocol as shown in Fig. 15.7. Signals traveling across a physical wire travel some distance before they become weak (in a process called attenuation), or get corrupted as they get interfered with other signals/noise. This means that the integrity of the data that the signal carries, is in danger. A repeater receives such a signal, which is likely to become weak or corrupted, and regenerates it. For instance, let us assume that a computer works on a convention that 5 volts represent 1 and 0 volts represent 0. If the signal becomes weak or distorted and the voltage becomes 4.5, the repeater has the *intelligence* to realize that it is still a bit 1 and therefore, it can regenerate the bit (i.e., 5 volts). That is, the repeater simply recreates the bit pattern of the signal, and puts this regenerated signal back on to the transmission medium. In effect, the original signal is *created* once again.

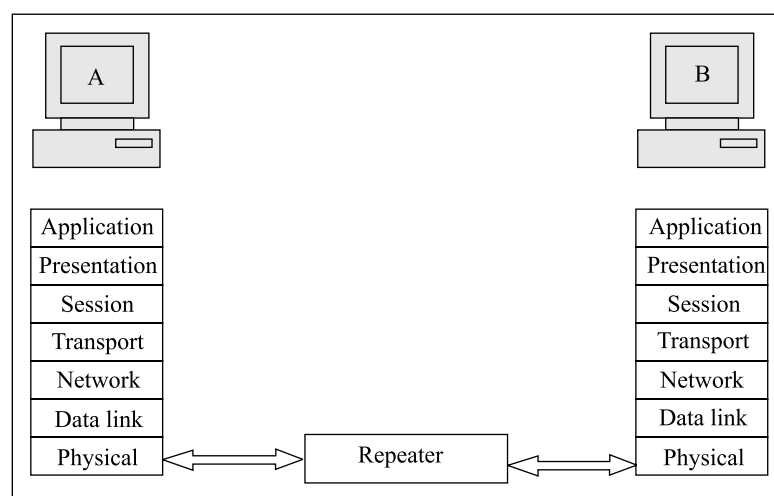


Fig. 15.7 Repeater at the physical layer

We would realize that a repeater allows extending a network beyond the physical boundaries, otherwise imposed by the data transmission media. Note that a repeater does not anyway change the data that is being transmitted, or the characteristics of a network. The only responsibility of a repeater is to take a stream of bits, in the form of a signal, regenerate it so that the signal is accurate now, and send it forward. It does not perform any intelligent function.

For instance, in the sample network (LAN) shown in Fig. 15.8, host A wants to send a packet containing the bit stream 01100110 to host D. Note that the two hosts are on the same LAN, but on different portions of the LAN. By the time the signal sent by host A can reach host D, it becomes very weak. Therefore, host D may not be able to get it in the form of the original signal. Instead, the bits could change to say 01100111 before the signal reaches host D. Of course, at a higher level, the error-control functions would detect and correct such an anomaly. However, even before this can happen, at the lowest level, the repeater simply prevents it from occurring by taking the input signal corresponding to bits 01100110 sent by host A, simply regenerating it to create a signal with the same bit format and the original signal strength, and sending it forward.

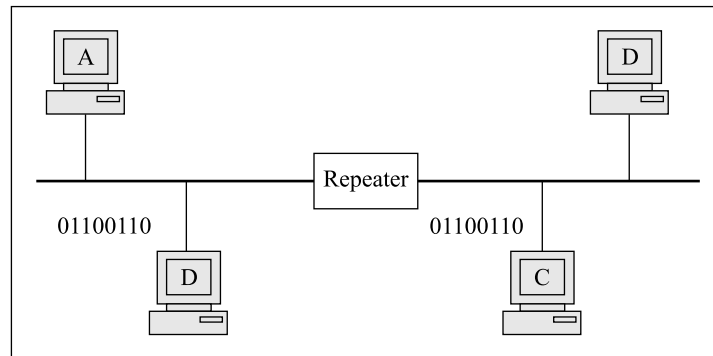


Fig. 15.8 Repeater regenerating a signal

People sometimes confuse between repeaters and amplifiers. However, they are different. An amplifier is used for analog signals. In analog signals, it is impossible to separate the original signal and the noise. An amplifier, therefore, amplifies an original signal as well as the noise in the signal, as it cannot differentiate between the two. On the other hand, a repeater knows that the signal has to be identified as either 0 or 1 only. Therefore, it does not amplify the incoming signal – it regenerates it in the original bit pattern. Since a signal must reach a repeater before it becomes too weak to be unidentifiable, the placement of repeaters is an important concern. A signal must reach a repeater before too much noise is introduced in the signal. Otherwise, the noise can change the bits in the signal (i.e., the voltage corresponding to the bit values), and therefore, corrupt it. After corruption, if a repeater regenerates it, incorrect data would be forwarded by the repeater.

15.7 BRIDGES

15.7.1 Introduction

A **bridge** is a computer that has its own processor, memory and two NIC cards to connect to two portions of a network. A bridge does not run application programs, and instead, facilitates host-to-host communication within a network. It operates at the physical as well as data link layers of the OSI protocol hierarchy. This is shown in Fig. 15.9.

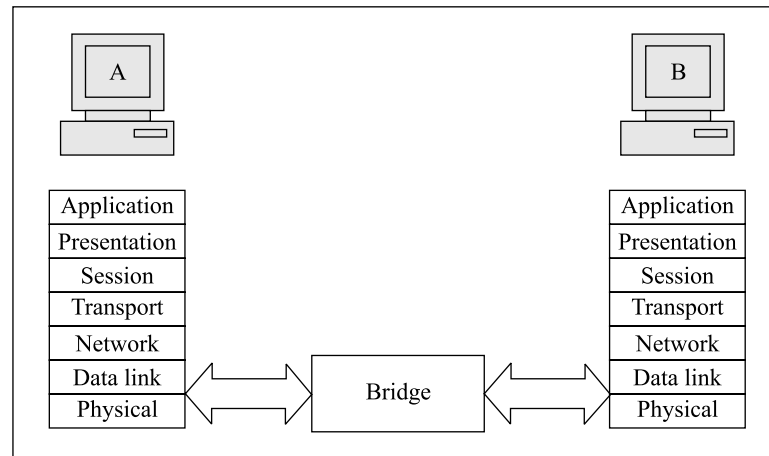


Fig. 15.9 Bridge at the last two OSI layers

The main idea of using a bridge is to divide a big network into smaller subnetworks, called **segments**. This is shown in Fig. 15.10. Here, the bridge splits the entire network into two segments, shown with dotted lines. We have also shown two repeaters, which we shall disregard for the current scope of discussion. Due to the bridge, the two segments act as a part of the single network.

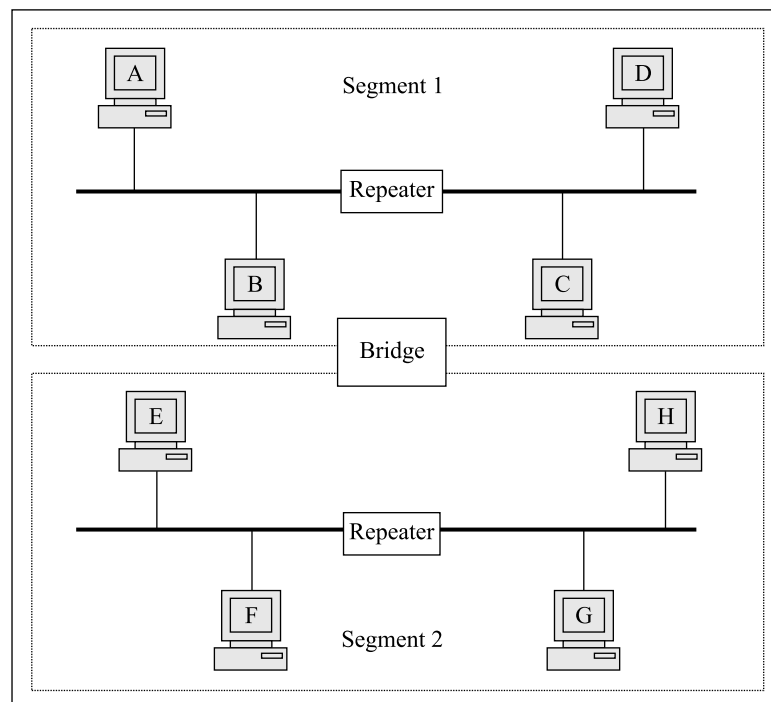


Fig. 15.10 Bridge connecting two segments

15.7.2 Functions of a Bridge

At a broad level, a bridge might appear to be the same as a repeater. After all, a bridge enables the communication between smaller segments of a network. However, a bridge is more intelligent than a repeater, as discussed below.

The main advantage of a bridge is that it sends the data frames only to the concerned segment, thus preventing excess traffic. For example, suppose we have a network consisting of four segments numbered 1 to 4. If a host on segment 1 sends a frame destined for another host on segment 3, the bridge forwards the frame only to segment 3, and not to segments 2 and 4, thus blocking unwanted data traffic.

Let us illustrate this with an example network shown earlier in Fig. 15.10. Suppose in our sample network, host A wants to send a frame to host D. Then, the bridge does not allow the frame to enter the lower segment. Instead, the frame is directly relayed to host D. Of course, the repeater might regenerate the frame as shown in Fig. 15.11.

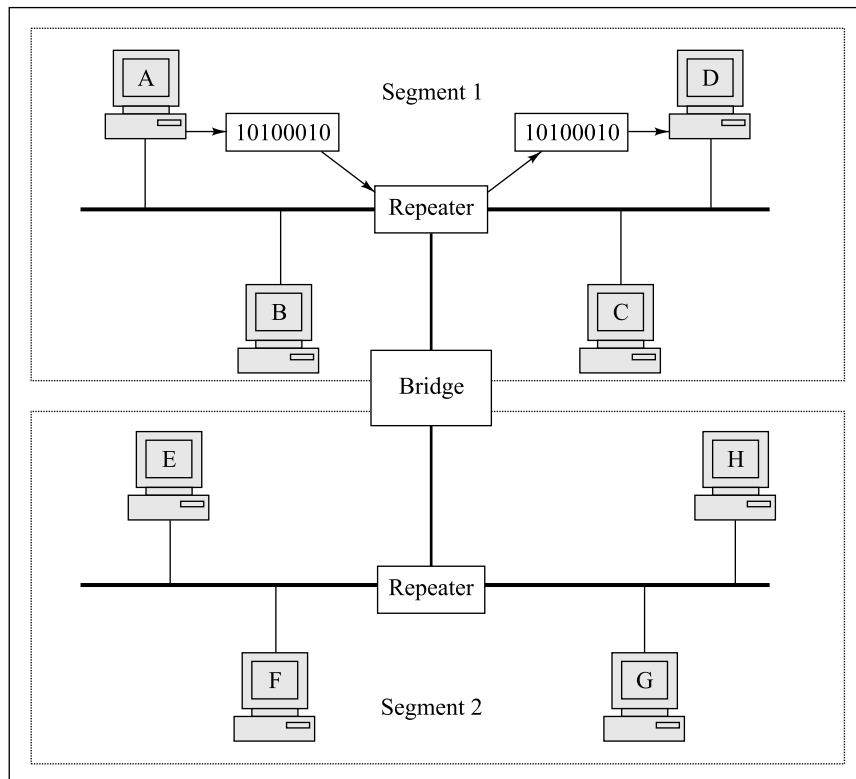


Fig. 15.11 *A bridge minimizes unwanted traffic*

By forwarding frames only to the segment where the destination host resides, a bridge serves the following purposes:

1. Unwanted traffic is minimized, thus network congestion can also be minimized to the maximum extent possible.

2. Busy links or links in error can be identified and isolated, so that the traffic does not go to them.
3. Security features or access controls (e.g., a host on segment can send frames to another host on network C but not to a host on network B) can be implemented.

Since bridges operate at the physical layer, they know the physical addresses of the different hosts on the network. Bridges can also take on the role of repeaters in addition to network segmenting. Thus, a bridge can not only regenerate an incoming frame, but also forward this regenerated copy of the frame to only the concerned segment, to which the destination host is attached. In such cases, the repeaters can be done away with.

15.7.3 Types of Bridges

As we have learned, a bridge forwards frames to only that segment of a network to which the destination host is attached. But how does it know to which segment is the destination host attached? For instance, in Fig. 15.10, if node A sends something to node D, the bridge should know that node D is not on the lower segment (2) and therefore, block that frame from entering the lower segment (2). On the other hand, if node A wants to send something to node G, it should pass it to the lower segment (2). How does it perform this filtering function?

For this, a bridge maintains a table of host addresses versus the segment numbers to which they belong. For the sample network and segments shown in Fig. 15.10, we can have a simple table used by the bridge as shown in Fig. 15.12. Note that instead of showing the 48-bit physical addresses, we have shown the host ids for ease of reference.

Host address	Segment number
A	1
B	1
C	1
D	1
E	2
F	2
G	2
H	2

Fig. 15.12 Host address to segment mapping

Bridges are classified into three categories based on (a) how they create this mapping table between host addresses and their corresponding segment numbers, and (b) how many segments are connected by one bridge. These three types of bridges are shown in Fig. 15.13.

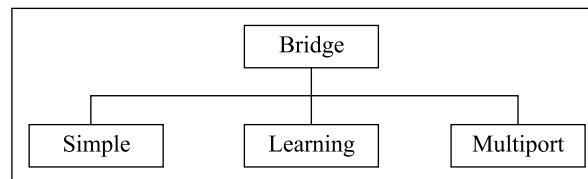


Fig. 15.13 Types of bridges


Let us discuss these three types of bridges now.

1. **Simple bridge** – This is a very primitive type of bridge. A **simple bridge** connects two segments. Therefore, it maintains a table of host addresses versus segment numbers mapping for the two segments. This table has to be entered by an operator manually by doing data entry of all the host addresses and their segment numbers. Whenever a new host is added, or an existing host is replaced or deleted, the table has to be updated again. For these reasons, simple bridges are the cheapest, but also have a lot of scope for error due to manual intervention.
2. **Learning bridge** – A **learning bridge**, also called an **adaptive bridge**, does not have to be programmed manually unlike a simple bridge. Instead, it performs its own bridging functions. How does it perform this function? For building the host address to segment number mapping table, a learning bridge examines the source and destination addresses inside the received frames, and uses them to create the table. Therefore, when a bridge receives a frame from a host, it examines its table to check if the address of the sending host is available in the table. If not, it adds it to the table along with its segment number. Then it looks at the destination address to see if it is available in its mapping table. If it is available, the bridge knows on which segment the destination host is located. Therefore, it delivers the frame to that segment. If the destination address, and therefore the segment number of the destination address are not available in its mapping table, the bridge sends the frame to all the segments to which it is connected.

Consequently, with the first packet transmitted by each host, the bridge learns the segment number for that host, and therefore, it creates an entry for that host in its mapping table containing the host address and its segment number. Over a period of time, the bridge constructs the complete mapping between the hosts and their segment numbers for the entire network. Since the bridge continues checking and updating its mapping table all the time, even if new hosts are added, existing hosts are removed or their NICs replaced, it does not matter! The bridge *learns* about these changes and adapts to them automatically.

Let us understand how a learning bridge creates its mapping table, with reference to Fig. 15.10. Suppose that the hosts on the network shown in Fig. 15.10 are just about to start transmissions for the first time. Note how the bridge first builds and then updates its mapping table, as shown in Fig. 15.14 for the given sequence of transmission.

Frame sent by host	Frame sent to host	Entry in the <i>Host address</i> column of the bridge's mapping table	Entry in the <i>Segment id</i> column of the bridge's mapping table
A	D	A	1
D	C	D	1
A	B	–	–
B	C	B	1
H	C	H	2
E	G	E	2
F	E	F	2
G	B	G	2
C	E	C	1

Fig. 15.14  A learning bridge building a mapping table

The last two columns of Fig. 15.14 show the mapping table of the bridge. Each of the rows indicates the process of updating the mapping table. For example, in the very first case, host A sends a frame to host D. The bridge receives the frame, examines the source address and realizes that it does not have an entry for the source (A) in its mapping table. Therefore, it creates an entry for the host A as the last two columns of the first row signify. In the same manner, all the other updates can be easily understood. The third row is different and interesting. Here, host A has sent a frame to host B. However, since the mapping table of the bridge already has an entry for A, the bridge does not add it again to its mapping table.

3. **Multipoint bridge** – A **multipoint bridge** is a special case of either a simple or a learning bridge. When a simple or a learning bridge connects more than two network segments, it is called a *multipoint bridge*.

15.7.4 Bridging Across Longer Distances

Bridges can be used to connect networks that span across long distances. There are two approaches for this. In the first approach, a leased serial line is used to connect the two distant networks. In the second approach, satellite links are used for distant communication between the two networks. The former is less expensive, and is therefore, more popular. However, satellites can be used if the distances vary greatly. For this to be possible, each of the networks needs to have the bridge hardware. The bridge hardware keeps listening to the traffic, and learns about the local addresses. It then uses this information to prevent forwarding of frames destined for local addresses across the leased line or satellite.

The motivation for this is that the leased lines or satellites have limited bandwidth. Therefore, it is very important to keep the traffic across the leased lines to a minimum. Additionally, the bridge hardware also performs the buffering functionality. This is because the frames arriving from the locally attached network would arrive faster at the bridge, whereas the distant network may not be able to accept the frames at such fast a rate, because of the limited bandwidth. Consequently, the bridge has to temporarily buffer the frames before it forwards them across the leased line or satellite.

15.8 ROUTERS

15.8.1 Introduction

A **router** operates at the physical, data link and network layers of the OSI model as shown in Fig. 15.15. A router is termed as an *intelligent* device. Therefore, its capabilities are much more than those of a repeater or a bridge. A router is useful for interconnecting two or more networks. These networks can be heterogeneous, which means that they can differ in their physical characteristics such as frame size, transmission rates, topologies, addressing, etc. Thus, if a router has to connect such different networks, it has to consider all these issues. A router has to determine the best possible transmission path, among the several available.

15.8.2 How Does a Router Work?

The concept of a router can be illustrated with the help of Fig. 15.16. As shown in the figure, there is a Token Ring network A, and an Ethernet network B based on bus architecture. A router connects to the Token Ring at point X, and to the Ethernet network at point Y. Since the same router connects to the two networks at these points, it means that this router must have two NICs. That is, the router

is capable of working as a special host on a Token Ring as well as the Ethernet network, in this case. Similarly, a router can connect other combinations of networks, such as an Ethernet with a FDDI, a Token Ring with a FDDI and with an Ethernet, and so on. The point is that, each of the router's NIC is specific to one network type to which it connects. In this example, the NIC at point X is a Token Ring NIC, whereas the NIC at point Y is an Ethernet NIC.

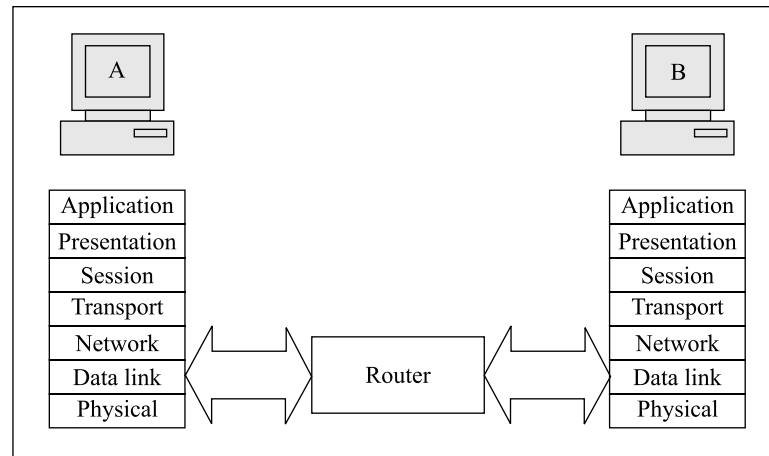


Fig. 15.15 Router at the last three OSI layers

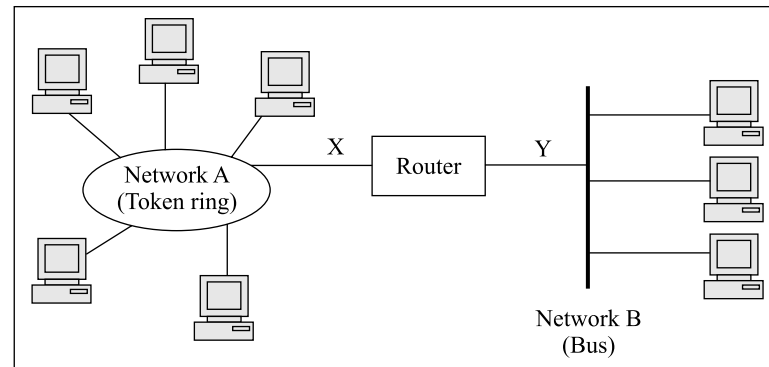


Fig. 15.16 Router connecting a Token Ring and a bus

Going a step further, we can connect multiple networks with the help of more than one router. For example, suppose we have an Ethernet, a X.25 network and a Token Ring as shown in Fig. 15.16. Then we can connect the Ethernet to the X.25 network using a router R1. This means that the router R1 must have two NIC interfaces, one catering to Ethernet and the other to X.25. Similarly, router R2 connects the X.25 network to a Token Ring. This means that the router R2 must also have two NIC interfaces, one for X.25 and the other for Token Ring. This is shown in Fig. 15.17. We can imagine that using more routers, we can connect any number of homogeneous or heterogeneous networks together to form an internetwork (or internet). The Internet (note the upper case), which is the popular network of networks is an example of an internetwork.

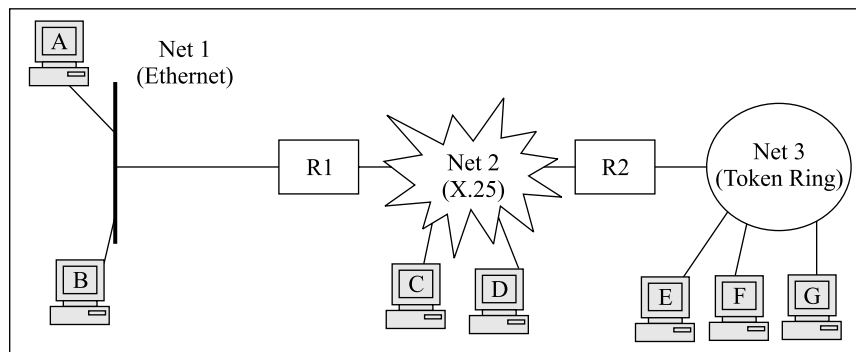


Fig. 15.17 Two routers connecting three networks together

15.8.3 Issues Involved in Routing

Having two NIC interfaces is fine. However, two major questions remain unanswered. They are related to the *frame format* and *physical addresses*, as discussed below.

Suppose host A on network 1 sends a frame for host G on network 3. As we can see from Fig. 15.17, the frame must pass through routers R1 and R2 before it can reach host G. However, the *frame format* and the *physical addressing mechanisms* used by network 1 (Ethernet) and network 3 (Token Ring) would be different.

1. Therefore, what would happen if host A attempts to send an Ethernet frame to host G? Because first the frame would reach router R1, which is also connected to network 1 (i.e., Ethernet), it would understand the format of the Ethernet frame, and because it has to now forward the frame to router R2, which is a X.25 network, router R1 would have to reformat the frame to X.25 format and send it to router R2. Router R2 can then transform the frame to the Token Ring frame format, and hand it over to host G, which is local to it. However, this reformatting is extremely complex. This is because it does not involve only converting the frame from one format to the other. It also involves mimicking the other protocol – including acknowledgement (Token Ring provides for it, but the Ethernet does not), priorities, etc. Can there be a better solution?
2. Similarly, what would host A put in the *destination address* field of the frame that it wants to send to host G? Should it put the physical address of host G in this field? In this case, it might work correctly, as both Ethernet and Token Ring use 48-bit physical addresses. However, what if host A wanted to send a frame to host D, instead of G? In this case, the address sizes of host A (Ethernet) and host D (X.25) would differ. Therefore, using physical addresses of hosts on other networks can be dangerous on an internet. How do we resolve this issue?

To resolve such issues, the network layer proposes two solutions as follows:

1. To resolve the issue of different frame formats, we should use a single *logical frame format*, which is independent of the physical networks. That is, we should have a common logical frame format, which does not depend on whether the source or the destination is an Ethernet or Token Ring, or any other network. Similarly, the intermediate networks can also be different from the source and the destination networks (as happens in our case, for example, where the source is Ethernet (network 1), the intermediate network is a X.25 network (network 2)

and the destination is a Token Ring (network 3)). We can then use that logical frame format universally, regardless of the underlying network types.

Therefore, the sender A must encapsulate this logical frame into an Ethernet frame and give it to its local router R1. The router R1 must extract the original logical frame from this Ethernet frame, transform it into a X.25 frame format by adding the appropriate X.25 headers, which is compatible with the next network via which it has to move forward, and send it to router R2. Router R2 should then extract the logical frame out from the X.25 frame, transform it into a Token ring frame by adding the appropriate Token ring headers, and give it to host G. At node G again, the original logical frame is extracted. When all such frames (i.e., the entire message) arrive at node G, it can be handed over to the upper layers at node G for processing.

We shall examine this process in detail in a separate chapter later, when we discuss TCP/IP.

2. To resolve the issue of different address formats, a **universal address** or **logical address** or **network-layer address** can be used across all networks. This logical address would be independent of all the underlying physical addresses, and their formats. Therefore, when the sender wants to send a frame to host G, it would put the logical address of host G in the *destination address* field. Using this logical destination address and the logical addresses of the intermediate nodes, routing will be done using which, the frame can move forward from host A to router R1, from router R1 to router R2, and from router R2 to host G. At various stages, the logical addresses would need to be translated to their equivalent physical addresses, because physical networks understand physical and not logical addresses.

We shall examine this process also later.

In both the cases, routers play a very significant role. Of course, apart from these, the routers have to find the most optimal path for routing frames. We have already discussed this concept earlier.

15.9 GATEWAYS

As shown in Fig. 15.18, a **gateway** operates at all the seven layers of the OSI model.

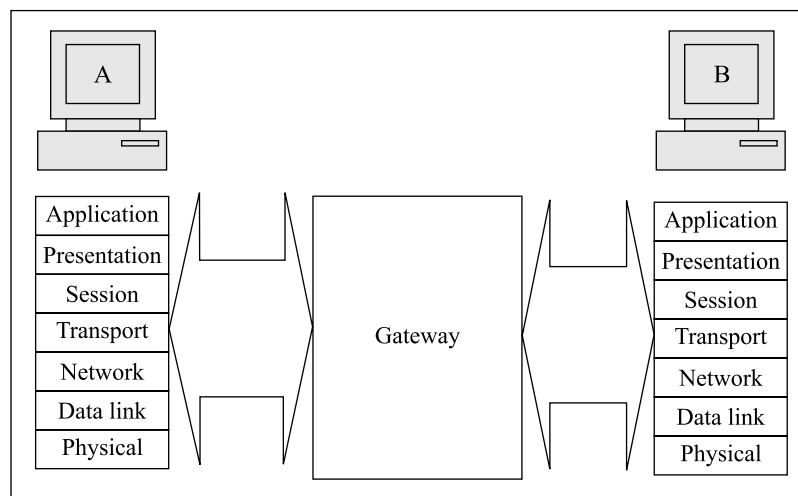


Fig. 15.18 Gateway at all the OSI layers

As we know, a router can forward packets across different network types (e.g., Ethernet, Token Ring, X.25, FDDI, etc). However, all these dissimilar networks must use a common transmission protocol (such as TCP/IP or AppleTalk) for communication. If they are not using the same protocol, a router would not be able to forward packets from one network to another. On the other hand, at a still higher level, a gateway can forward packets across different networks that may also use different protocols. That is, if network A is a Token Ring network using TCP/IP and network B is a Novell Netware network, a gateway can relay frames between the two.

This means that a gateway has to not only have the ability of translating between different frame formats, but also different protocols. The gateway is aware of, and can work with, the protocols used by each network connected to a router, and therefore, it can translate from one to the other. In certain situations, the only changes required are to the frame header. In other cases, the gateway must take care of differing frame sizes, data rates, formats, acknowledgement schemes, priority schemes, etc. That means that the task of the gateway is very tough.

Clearly, a gateway is a very powerful computer as compared to a bridge or a router. It is typically used to connect huge and incompatible networks.

15.10 A BRIEF HISTORY OF THE INTERNET

15.10.1 Introduction

Although the Internet seems to have become extremely popular over the last decade or so, it has a 40-year long history. The motives behind the creation of the Internet were two-fold, as given below:

1. Researchers wanted to communicate with each other and share their research papers and documents.
2. The US military system wanted a strong communications infrastructure to withstand any nuclear attack by the erstwhile Soviet Union. The idea was that even if both the countries were completely destroyed by the World War, the important American scientists and diplomats could hide in the underground bunkers and still communicate with each other to reconstruct America ahead of Soviet Union and therefore, win the World War that would follow.

These developments date back to the 1960s. This necessitated a large decentralized network of computers within the United States. In 1961, Baran first introduced the concept of **store-and-forward packet switching**. We have studied this in the earlier chapters. These concepts were very useful in the development of the Internet.

15.10.2 ARPAnet

Baran's original ideas did not attract publicity for quite some time. Actually similar ideas were put forward by Zimmerman in France. Baran's ideas were firstly used by the **Advanced Research Project Agency (ARPA)** of the US Department of Defense. They sponsored a network of computers called **ARPAnet**, which was developed with the aim of sharing of scattered time-sharing systems. This made sharing of long-distance telephone lines possible, which were quite expensive. ARPAnet was first proposed in 1966 and was actually built by a few researchers in 1969. This was the pioneering effort in actually practicing concepts of wide-area packet switching networks, decentralized routing, flow control and many applications such as **TELNET** (which allows a user to log in to a computer from remote distance) and **FTP (File Transfer Protocol)**, which are used even today.

Once ARPAnet started becoming popular, people thought of connecting other networks to ARPAnet, thus creating a network of computer networks, or the *internetwork*. This led to the ideas of shared packet format, routing and addressing schemes. The important point is, throughout these developments, care was taken to ensure that the network of networks should be kept as decentralized as possible. The concept of a *router* was defined at this stage. As we have discussed, a router is a computer that is mainly used for transferring data packets between computer networks. Also, IP protocol was so designed that no assumptions were made regarding the underlying transmission medium (such as twisted pair) or the frame format used by a specific network (e.g., Ethernet). It was kept independent of the transmission medium and the specific network hardware or software.

Other networks such as CSNET and NEARnet were developed using the same philosophy as of ARPAnet. Soon, many such networks started developing. Because of the inherent design, these networks could all be connected together. By the early 1980s, ten such networks had formed what is now called the *Internet*.

15.10.3 The World Wide Web (WWW)

The **World Wide Web (WWW)** is an application that runs on the Internet and is one of the most popular of all the Internet applications. The WWW was first developed with a very simple intention, i.e., to enable document sharing between researchers and scientists that were located at physically different places. In 1989, Tim Berners-LEE at the Conseil Européen pour la Recherche Nucleaire (CERN), now known as the European Laboratory for Particle Physics, started the WWW project. His goals were in the following two areas:

1. Developing ways of linking documents that were not stored on the same computer, but were scattered across many different physical locations (i.e., hyperlinks).
2. Enabling users to work together – a process called **collaborative authoring**. After over a decade's growth, the first goal has been successfully met. However, the second is still not complete.

Interestingly, as soon as the basic outline of the WWW was ready, CERN published the software code for the general public. This attracted programmers very much to the WWW. This concept of **open source code** or **freeware** (unlike **proprietary source code**, which is not made available to the general public), meant that people could not only experiment with the WWW software, but also add new functionalities and extend them to new heights. Soon, hundreds of programmers from the National Center for Supercomputing Applications (NCSA) at the University of Illinois started working on the initial WWW software development. Making use of the basic code developed at CERN, NCSA came up with the first **Web browser** with graphical user interface called **Mosaic** in 1993. A browser is simply a program running on a **client computer** that retrieves and allows viewing a document stored on a remote **server computer** (we shall soon study the meaning of client and server computers).

Key members of the teams that developed Mosaic and the original **Web server** software [A Web server is a computer program that waits for requests from remote clients (i.e., Web browsers) for documents stored on the server computer, retrieves them, and sends these back to the clients] at CERN came together to form a company called Netscape Communications. They developed Netscape Navigator, a new browser in December 1994. This browser was the first commercial Web browser for the WWW that included many new features, the chief among them being security, thus enabling commercial transactions over the WWW in the years to follow. Over the next few years,

the WWW grew from experimentation to a truly commercial project. This can be judged from the value ratings of Netscape Communications. Netscape Communications, a company with two-year history with almost no revenue, went public in August 1995. It was initially valued at \$1.1 billion, a figure that rose almost five times in the next four months.

To avoid proprietary influences, the WWW project shifted from CERN to Massachusetts Institute of Technology (MIT) in 1995 and is now called the **W3 Consortium**. This Consortium coordinates the development of WWW standards and ensures uniformity and minimum duplication of efforts.

15.11 GROWTH OF THE INTERNET

While ARPA was working on the Internet research project, the Unix operating system was also taking the computing world by storm. A group of computer researchers developed Unix in the early 1970s at the Bell Labs. Bell Labs allowed universities to have copies of Unix for teaching and research purposes. To encourage its portability, Bell Labs provided the source code of the Unix operating system to the students. This meant that the students could try it out in a variety of ways to see if it worked and could also modify it to make it better. A group of students at the University of California at Berkley wrote application programs and modified the Unix operating system to have network capabilities (e.g., sending a message to another computer, accessing a file stored on a remote computer, etc.). This version of Unix, later called **BSD Unix** (Berkley Software Distribution) became very popular.

ARPA noticed that BSD was now a well-known entity. They signed a deal with Berkley researchers under which the BSD Unix now incorporated TCP/IP protocol in the operating system itself. Thus, with the next version of BSD Unix, people got TCP/IP software almost free. Although few Universities who bought BSD Unix had connection to the Internet, they usually had their own Local Area Networks (LANs). They now started using TCP/IP for LANs. Later on, the same concept was applied to the Internet. Thus, TCP/IP first entered the LANs at the universities, and from there, to other networks.

By the early 1980s, the Internet had become reliable. The main interconnection at this stage was between academic and research sites. The Internet had demonstrated that the basic internetworking principles were quite sound. This convinced the US military. They now connected their computers to the Internet using TCP/IP software. In 1982, the US military decided to use the Internet as its primary computer communication medium. At the start of 1983, ARPANET and the concerned military networks stopped running old communication software and made TCP/IP the de facto standard.

Before the US military started switching its computers to use TCP/IP, there were about 200 computers connected to the Internet. In 1984, this number almost doubled. Other US government agencies such as Department of Defense (DOD) and National Aeronautics and Space Administration (NASA) also got themselves connected to the Internet. Meanwhile, the cold war suddenly ended. The Internet came out of the secret world of the military and became open to the general public and businesses. Since then, the number of computers connected to the Internet kept almost doubling every year. In 1990 there were approximately 2,90,000 computers connected to the Internet. By 1997, this number reached 1,61,46,000. At the time of going to the press, it is estimated that every three seconds a new computer connects to the Internet. Figure 15.19 shows the number of computers connected to the Internet from 1983 to 2001.

Year	Approximate number of computers connected to the Internet
1995	16,00,000
1996	36,00,000
1997	70,00,000
1998	147,00,000
1999	248,00,000
2000	359,00,000
2001	479,00,000
2002	569,00,000
2003	608,00,000
2004	757,00,000
2005	939,00,000
2006	1,043,00,000
2007	1,173,00,000
2008	1,463,00,000
2009	1,669,00,000
2010	1,996,00,000

Fig. 15.19 The growth of the Internet

This is shown graphically in Fig. 15.20.

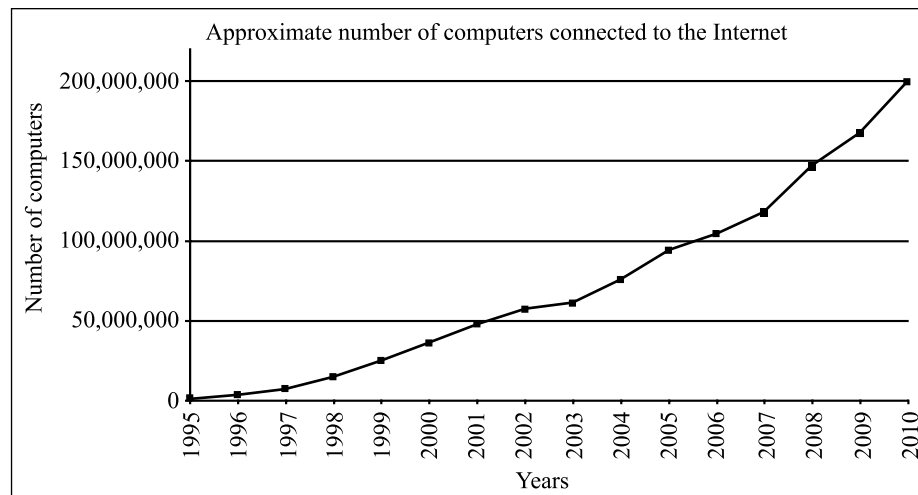


Fig. 15.20 The growth of the Internet

15.12 INTERNET TOPOLOGY

The Internet is a worldwide network of computer networks. It is organized to form a hierarchy that makes it very simple to understand and operate. Let us look at the different parts of this hierarchy, which is shown in Fig. 15.21.

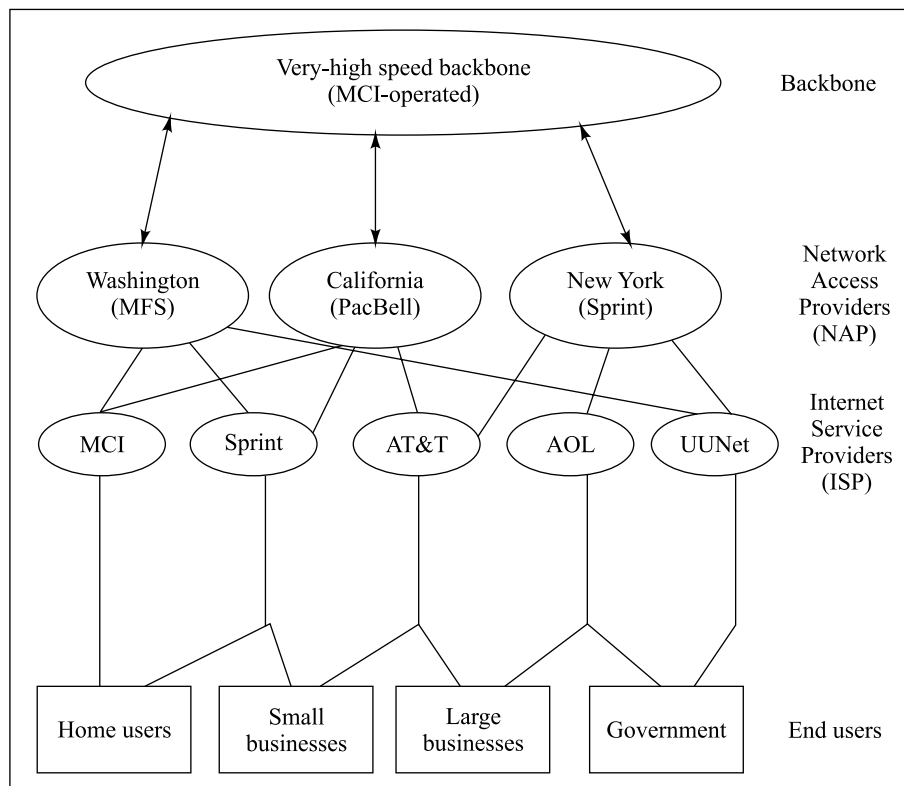


Fig. 15.21 The Internet topology

At the very top, there is a very high-speed backbone operated by MCI and at the other extreme end, there are users and businesses. There are intermediate layers of **Network Access Providers (NAP)** and **Internet Service Providers (ISP)** as shown in the figure.

Let us now see how a user gets connected to the Internet.

1. A home user dials into an **Internet Service Provider (ISP)**, usually using a twisted-pair telephone connection using a modem. Thus, the business of ISP is essentially buying the bandwidth in wholesale and selling it in retail at profit. Of course, in order to win in the competitive market, many ISPs provide additional services such as Web site designing, Web site hosting etc.
2. The ISP, in turn, routes the call to a **Network Access Provider (NAP)**. For instance, AOL has its own network of several computers spread all over, connected via a number of routers. This network was set up for allowing communication between users, which were connected to and which were subscribed for AOL only. For this, AOL set up its network and enabled

users from homes or offices to be connected to the AOL network for a fee. Thus, two users connected to AOL could send emails using AOL's network and the email software. In a sense, AOL was competing with the Internet. However, considering the growth and the popularity of the Internet, AOL decided to cooperate with the Internet. It means that anybody connected on the AOL network should now be able to access the Internet or send an email to the one who is not on the AOL network, but who is on the Internet. Now, the same AOL has become an ISP. Thus, a message is accepted by this network at a specific access point from the ISP and carried through the network as shown in Fig. 15.22. At another access point, it is transferred to the network of the NAP at a specific access point.

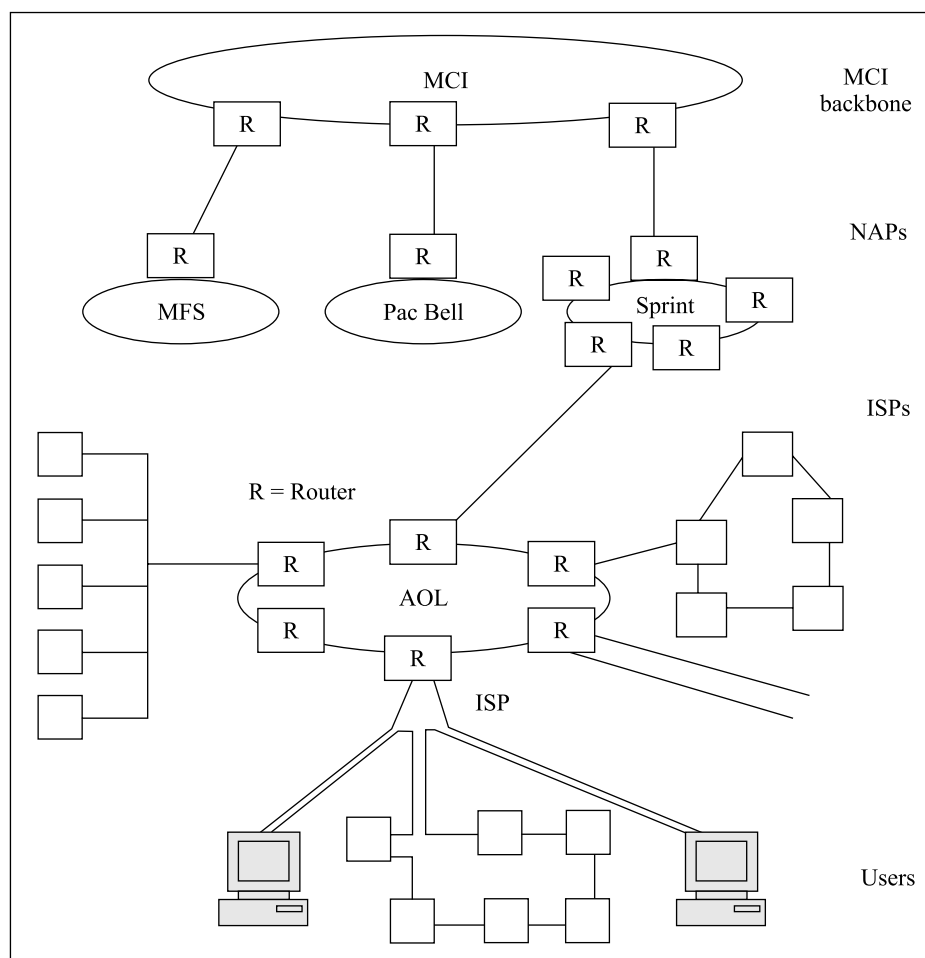


Fig. 15.22 A detailed look at NAP and ISP

An important aspect is to connect these NAPs to one another to allow sending messages from one computer to any other. Earlier, each NAP used to be connected to every other. This complicated the process. Finally, a high-speed backbone was conceived and operated by MCI. On this backbone, specific points were identified as **Network Access Points** (again NAPs). Any

Network Access Provider had to connect to one of the (ideally the closest) Network Access Points on the backbone.

Instead of AOL, if we can imagine that any other ISP, the general principles would still apply. A home user or an office dials into the ISP; the ISP connects to one of the Network Access Providers, which in turn, connects to the high-speed backbone at a convenient Network Access Point. This completes the circuit. Now, the user can send/receive any message to anyone else connected to the Internet.

15.13 INTERNAL ARCHITECTURE OF AN ISP

15.13.1 Introduction

Having discussed the hierarchical arrangement of the various aspects of the Internet, let us now focus our attention to understanding the internal architecture of an ISP. At the fundamental level, an ISP has to essentially provide for the feature of Internet access to its subscribers. This is a compulsory feature of an ISP. Optionally, it might also provide some other services such as Web page hosting and email access for its subscribers; and also its own ISP portal. Each of these services necessitates additional hardware/software in the ISP premises. Let us discuss these additional services provided by an ISP, and their requirements.

1. **Web page hosting** – The ISP might allow its subscribers to create their own Web pages. For this, the ISP can allocate a certain amount of disk space per user, and let the user create a set of Web pages there. Any Internet user should be able to browse these Web pages like any other Web page on the Internet. Obviously, for this to be possible, apart from the disk space, a Web server is required in the ISP premises to host the subscribers' Web pages, and issue them to any user (Web client/browser) on request.
2. **Email access** – Almost all ISPs provide an email facility for their subscribers. This means that the ISP must host a server to send and receive mails on behalf of its subscribers. Such a server is called an **SMTP server**, as we shall study later. (Do not worry about the name at this juncture.)
3. **ISP portal** – As a value-added service, most ISPs host their own portal. An ISP portal is more or less similar to any other portal such as Yahoo, on the Internet. The ISP portal provides news, weather information, technical updates, sports updates, chat facilities, and so on. For this, the ISP needs to set up Web servers that host these contents and create the necessary Web pages to show them.

15.13.2 High-level Architecture of an ISP

Considering these points, let us draw a high-level conceptual view of the typical architecture of an ISP, as shown in Fig. 15.23.

The ISP buys a number of **IP addresses** for a number of its subscribers, depending on its estimate as to how many users are likely to make an attempt to connect to the Internet. As we shall see in later chapters, an IP address is used to identify a computer on the Internet uniquely. Thus, when a user connects to an ISP, the ISP provides one of the IP addresses from its list, which is currently unused, so that the user's connection with the ISP can be uniquely identified. When the user breaks its connection with the Internet (and therefore, the ISP), its IP address is added back to the list of free IP addresses, so that the ISP can now allocate that IP address to another user.

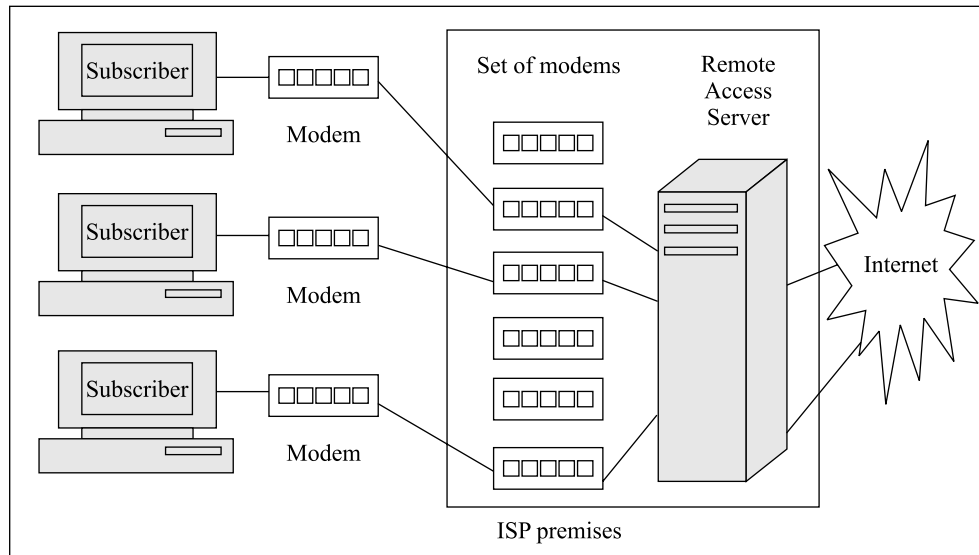


Fig. 15.23 The logical architecture of an ISP

Figure 15.23 shows three users connected to the Internet via the ISP. At this stage, we are not going into the details of the additional services provided by the ISP. Instead, we shall look at the basics of a common subscriber-ISP connection. Let us discuss this step by step.

1. When a subscriber obtains (*buys*) an Internet connection from an ISP, the subscriber gets a user id, password and a telephone number to dial into the ISP. The ISP stores the user id and the password in a database maintained by the ISP on a **Remote Access Server (RAS)** shown in Fig. 15.23. When the subscriber wishes to connect to the Internet, using these settings, a client **dial-up program** at the subscriber's end dials the number provided by the ISP via the **modem**. In simple terms, it passes the telephone number of the ISP to the modem and instructs it to call that number.
2. The subscriber's modem uses the underlying telephone connection to call the ISP's number.
3. This call is like any telephone call between two parties. The difference in this case is that the subscriber's modem tries to establish a connection with one of the modems at the ISP's premises. Since many subscribers can attempt to connect to the same ISP at the same time, an ISP usually obtains many telephone connections from the telephone company. Of course, it maps a single *hunting line number* or a *board number* to all its other telephone number. Therefore, although all the subscribers dial the same number, actually, the call might be routed to another number of the ISP without the subscriber noticing it. In any case, assuming that not all telephone lines of the ISP are already busy, the call placed by the subscriber reaches one of the modems of the ISP. Thus, a dedicated telephone connection is established between the modems of the subscriber and the ISP.
4. The ISP's modem responds back to the subscriber's modem with an acknowledgement signal.
5. The subscriber's modem passes the acknowledgement message to the dial-up program of the subscriber. This is an indication to the application program at the subscriber's end that the ISP is ready to accept the connection from the subscriber. Of course, for this, the subscriber must first authenticate himself, as described next.

6. The dial-up program realizes that it is time for user authentication. Therefore, it prompts for the user id and password from the user and having got it, sends the same via the same path already created, to the ISP. Of course, it has to hand this information over to the modem, which sends it across the telephone connection to the ISP's modem.
7. The ISP's modem receives this information, and passes it on to the *Remote Access Server (RAS)*. The RAS maintains a database of the valid user ids and their passwords. It consults that database to check whether the user id and password sent by the subscriber are valid. If they are valid, it sends a *valid* message back to the subscriber's modem via its own modem. Otherwise it sends an *error* message back to the subscriber's modem.
8. Assuming that the subscriber's user id and password were authenticated successfully, the subscriber is virtually connected to the Internet through the ISP, the Network Access Provider and the high-speed backbone, as discussed earlier.
9. The subscriber can now access the Internet services such as HTTP, FTP, email, and so on. For this, the ISP now acts as a mediator that takes the subscriber's requests (which are in the form of IP packets as we will see later), and forwards them on to the rest of the Internet. Similarly, it sends the responses received from the Internet, back to the subscriber.

Note that we have kept the discussion at a simplistic level, and have not really gone into the details of the Remote Access Server (RAS) and other facilities at the ISP's premises. Let us now look at these internal aspects of the ISP's infrastructure as shown in Fig. 15.24.

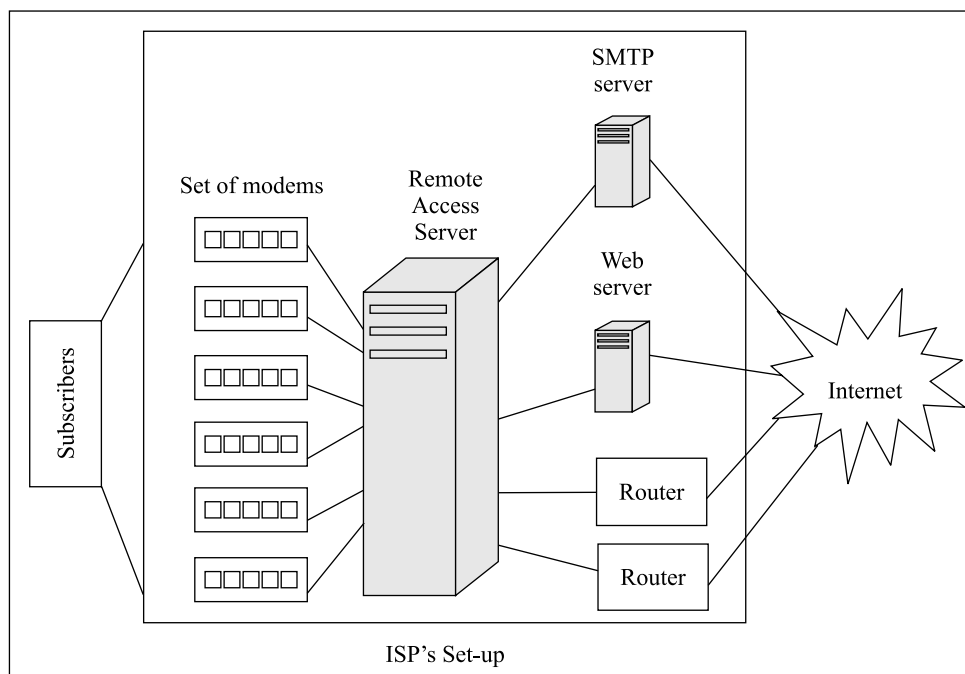


Fig. 15.24 The internal architecture of an ISP

As Fig. 15.24 shows, and as we have discussed previously, a subscriber of an ISP connects to one of the telephone lines of the ISP via a modem. The ISP's modem then routes the subscriber to the Remote Access Server (RAS) to authenticate her/him. Once the user is authenticated, he is as good as connected to the Internet. After this, the user can send requests for Web pages, send/receive emails or files, etc. The user's interaction with the Internet is coordinated by RAS, as shown in the figure.

In the figure, we have shown an **SMTP server** at the top. The ISP needs this server for storing the subscribers' **emails** until the time the subscriber connects to the Internet. Why is this required? Let us understand this. As we discussed, when a user subscribes to the services of an ISP, the ISP optionally allows the user to create an email id and use it for sending/receiving emails. For instance, let us assume that a user Lisa Johnson subscribes to the services of the famous ISP – AOL. Then, AOL might allow Lisa to create an **email id** such as *Lisa.Johnson@aol.com*. This simply means that Lisa can now send or receive emails using this email id. However, Lisa would not be connected to the Internet all the time. Suppose she connects to the Internet for an hour every day to do some browsing or for sending and receiving emails etc.

What happens if somebody sends an email to Lisa when she is not connected to the Internet? The email would be lost. To avoid such situations, the ISP (AOL) sets up an SMTP server, and creates an account (i.e., the disk space and the folders, mailboxes in that disk space) for Lisa on its server. It then keeps its SMTP server *on* and connected to the Internet all the time. Therefore, anybody can send emails to Lisa any time regardless of whether she is connected to the Internet. AOL's SMTP server would receive the emails on her behalf, and store them in an account created for her. When Lisa connects to the Internet the next time, and wants to check if any new emails have arrived, her browser connects to this SMTP server at AOL and retrieves the emails from her account on that SMTP server. At this juncture, she can transfer her emails to her PC/disk and/or delete these messages from the SMTP server. Of course, this interaction is facilitated by RAS. We shall study this in detail when we discuss how email works. However, the main point is that since the Internet users are not always connected to the Internet, an ISP stores emails on their behalf on the SMTP server temporarily.

The second server shown in the figure is the Web server. As we had discussed in the optional features provided by an ISP, this Web server can serve two purposes. Firstly, it can be used by the ISP to set up a portal. Secondly, this Web server can be used to store the Web pages created by the subscribers. Of course, in an ideal architecture, the Web pages created by the subscribers and those belonging to the ISP's portal would be hosted on different Web servers for reasons of security and maintenance.

Finally, the RAS connects to a couple of routers. In practice, there could be many more routers here. These routers essentially route the user's requests for Web pages, etc., to the Internet. For instance, suppose a user wants to connect to the Hotmail Web site. Then, there is no point for the RAS to redirect this request either to the SMTP server or the Web server shown in the figure for a simple reason that Hotmail's Web server is outside of the boundaries of the ISP. Therefore, the ISP must route this request to the outside Internet. The routers shown here perform precisely the same job. Many such routers ensure that the ISP is not overloaded with requests, and that the subscribers get a good response/throughput. Of course, just having many routers would not serve the purpose. In addition to this, the ISP has high-speed large bandwidth connections from these routers to the rest of the Internet (i.e., NAP).

SUMMARY

When multiple computers are connected to each other, a computer network is formed. When multiple computer networks are connected to each other, it becomes an internetwork, or an internet.

Internetworking is a highly desirable feature whereby multiple discrete computer networks can be interconnected. This allows for vast sharing of information and resources across physical boundaries. However, internetworking is not easy. Different types of networks have different physical characteristics such as varying signaling mechanisms, different addressing techniques, etc. The goal of internetworking is to address these differences and create a seamless internetwork.

A router is a special-purpose computer that can connect multiple computer networks. For this, a router has an interface (NIC) to each of the networks that it connects to. When a packet arrives at a router for forwarding it to another host/network, the router examines the destination address contained in the packet and forwards the packet on its appropriate route after reformatting the packet to suit the network to which it is being sent.

The Internet is a virtual network of computer networks. The term *virtual* arises because it is actually a network of a number of networks, which differ in their hardware and software characteristics, and yet work seamlessly with each other. At the backbone of this are the routers that connect these different types of networks to each other. This is similar to how different telephone systems in the world work happily with each other without the end user worrying about the differences in them.

Networking and internetworking devices are used respectively for connecting computers and networks together. Depending on the needs and the sophistication required, there are two main categories, viz., networking devices and internetworking devices. Networking devices include repeaters and bridges. Internetworking devices can be classified into routers and gateways.

A repeater is an electronic device, which simply regenerates a bit stream. It works at the physical layer of the OSI protocol. It has no other intelligence built in.

A bridge is a computer that has its own processor, memory and two NIC cards to connect to two portions of a network. A bridge does not run application programs, and instead, facilitates host-to-host communication within a network. It operates at the physical as well as data link layers of the OSI protocol hierarchy. Bridges help isolating network problems to a portion of a network. They also help in extending the network by adding segments to them.

A router is an *intelligent* device. Its capabilities are much more than those of a repeater or a bridge. A router is useful for interconnecting two or more networks. These networks can be heterogeneous, which means that they can differ in their physical characteristics such as frame size, transmission rates, topologies, addressing, etc.

The most powerful device is a gateway. A gateway can forward packets across different networks that may also use different protocols. That is, if network A is a Token Ring network using TCP/IP and network B is a Novell Netware network, a gateway can relay frames between the two.

The Internet is one of the most significant developments of the 20th century. The development of the Internet can be attributed to two main factors: (a) The need for electronic document exchanges, as felt by the scientist community, and (b) The need for a decentralized network, as felt by the US government to withstand a possible attack by the then Soviet Union.

At the heart of the Internet are the TCP/IP suite of protocols and store-and-forward packet switching. The Internet provides for a variety of applications such as File Transfer Protocol (FTP) for transferring files, TELNET for remotely accessing a computer, Electronic mail (email) and the most popular of all, the World Wide Web (WWW).

The WWW model, like all other Internet applications, is based on the concept of client-server computing. The Web server is a computer that hosts and serves Web pages, upon requests from the Web browsers, which are the clients.

The Internet has grown from a few hundred computers to several millions in less than twenty years. The Internet is now growing exponentially.

At the heart of the Internet is a worldwide backbone, to which many Network Access Providers (NAPs) attach. Each NAP, in turn, serves many Internet Service Providers (ISPs). A home user, or an office, then connects to one such ISP. This hierarchy allows for an almost unlimited growth of the Internet.

An ISP typically purchases bandwidth in wholesale and distributes it to its users in retail. Apart from that, ISPs these days also allow additional facilities for their users.

KEY TERMS AND CONCEPTS

Adaptive Bridge	Modem
Advanced Research Project Agency (ARPA)	Network Access Point (NAP)
ARPAnet	Network Access Provider (NAP)
BSD Unix	Networking devices
Bridge	Network-layer address
Browsing	Open source code
Client computer	Proprietary source code
Collaborative authoring	Remote Access Server (RAS)
Connecting devices	Regenerator
Dial-up program	Repeater
Email	Router
Email id	Segment
Email service	Server computer
File Transfer Protocol (FTP)	Simple bridge
Freeware	SMTP server
Gateway	Store and forward packet switching
Internet	TELNET
Internet Service Provider (ISP)	Transmission Control Protocol/Internet Protocol (TCP/IP)
IP addresses	Universal address
ISP portal	Universal service
internet	Virtual network
Internetwork	Web browser
Internetworking	Web page hosting
Internetworking devices	Web server
Learning Bridge	W3 Consortium
Logical address	World Wide Web (WWW)
Mosaic	

QUESTIONS

True/False

1. Internet and internet are different names for the same concept.
2. Any two networks can communicate with each other by just connecting a wire between them.
3. There are only software issues when connecting networks to each other.
4. A router is a special-purpose computer that is used specifically for internetworking purposes.
5. Just one address is sufficient for a router.
6. A router acts at the network layer of the OSI model.
7. The Internet is called a virtual network.
8. When two or more networks are connected for sharing data or resources or exchanging messages and resources it is called internetworking.
9. The two or more networks forming an internetwork must be compatible with each other; else the connection is not possible.
10. A repeater receives a signal, which is likely to become weak or corrupted, regenerates it and puts it back on to the transmission medium.
11. A bridge is a computer that has its own processor, memory and two NIC cards to connect to two portions of a network.
12. A bridge also runs application programs, along with facilitating host-to-host communication within a network.
13. A bridge operates at the physical as well as data link layers of the OSI protocol hierarchy.
14. A repeater is more intelligent than a bridge.
15. Bridges know the physical addresses of the different hosts on the network.
16. The repeaters help us to identify busy links or links in error.
17. A simple bridge is also called an adaptive bridge.
18. A router is considered to be very intelligent. It has capabilities, which are much more than those of a repeater or a bridge.
19. All the dissimilar networks must use a common transmission protocol for communication.
20. A gateway is typically used to connect huge but compatible networks.
21. Baran first introduced the concept of store and forward packet switching.
22. At the very top of the Internet hierarchy, there is a very high speed backbone operated by MCI.
23. The business of ISP is buying the bandwidth in wholesale and selling it in retail at profit.
24. The Internet is not at all based on TCP/IP.
25. An ISP needs an SMTP server for storing the subscribers' emails until the time the subscriber connects to the Internet.

Multiple-Choice Questions

1. In internetworking, the two or more networks that connect with each other _____
 incompatible with one another.

(a) have to be	(b) may be
(c) must not be	(d) None of the above

2. Usually, a _____ is used for internetworking purposes.
 - (a) host
 - (b) wire
 - (c) router
 - (d) joiner
3. A router must have at least _____ NICs.
 - (a) 2
 - (b) 3
 - (c) 4
 - (d) 5
4. There are _____ incompatibility issues when forming an internet out of networks.
 - (a) both hardware and software
 - (b) only hardware
 - (c) only software
 - (d) software but not hardware
5. A bridge is a _____ device.
 - (a) networking
 - (b) connecting
 - (c) internetworking
 - (d) routing
6. A _____ is the simplest of all networking/internetworking devices.
 - (a) repeater
 - (b) bridge
 - (c) router
 - (d) gateway
7. Generally, a _____ is used to divide a network into segments.
 - (a) repeater
 - (b) bridge
 - (c) router
 - (d) gateway
8. A _____ builds its mapping table as and when it gets more information about the network.
 - (a) simple bridge
 - (b) repeater
 - (c) adaptive bridge
 - (d) regenerator
9. A logical address is _____ the physical address.
 - (a) the same as
 - (b) tightly coupled with
 - (c) sometimes related to
 - (d) completely unrelated to
10. A _____ can understand multiple networking protocols.
 - (a) repeater
 - (b) bridge
 - (c) router
 - (d) gateway
11. _____ first introduced the idea of store and forward packet switching.
 - (a) Baran
 - (b) Barak
 - (c) Daran
 - (d) Daran
12. A _____ waits for requests from a _____.
 - (a) Web browser, Web server
 - (b) client, server
 - (c) Web server, Web browser
 - (d) None of the above
13. A home user dials into _____.
 - (a) ISP
 - (b) NAP
 - (c) backbone
 - (d) router
14. The _____ maintains details of user ids and passwords at the ISP.
 - (a) router
 - (b) file
 - (c) FTP server
 - (d) RAS server

Detailed Questions

1. Discuss the motives for internetworking.
2. What is universal service?
3. What are the various broad-level issues in internetworking?

4. Discuss the hardware issues in internetworking.
5. How does a router facilitate interconnection between two or more networks?
6. What are the software issues in internetworking?
7. What is internetworking? Give a summary of internetworking devices.
8. Explain the role played by the repeater at the physical layer.
9. What is a bridge? Explain its functions.
10. What are the types of bridges? Explain the simple bridge.
11. Explain how learning and multiport bridges work.
12. What is a router? How does it work?
13. Discuss the issues involved in routing.
14. How does a gateway work?
15. Discuss in brief the history of the Internet.
16. Explain how the Internet grew to the huge numbers as they stand today.
17. Explain the Internet hierarchy in brief.
18. What is the fundamental service provided by an ISP? What are the optional features it provides?
19. Discuss how a user gets connected to the Internet using an ISP.

16 Ways of Accessing the Internet

16.0 INTRODUCTION

These days, the Internet can be accessed in a number of ways. Initially, the only way to access the Internet was by obtaining a telephone connection, and then purchasing an account with an Internet Service Provider (ISP). The telephone company would carry the data between the user's premises and the ISP's office (called *last mile*) as shown in Fig. 16.1. From this point onwards, the ISP would connect the user to the Internet.

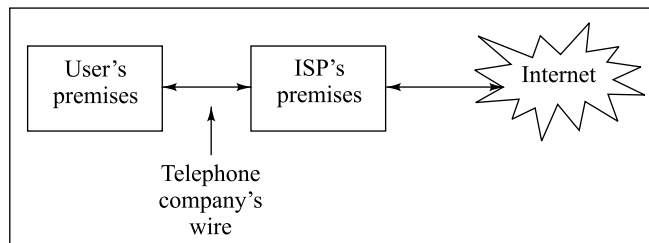


Fig. 16.1 Basic Internet access method

Although this basic framework has not changed a lot, the ways of connecting to the ISP from a user's premises itself have changed dramatically. Technological progress has meant that now there are many ways in which one can connect to the ISP for different needs. In the following sections, we shall discuss these alternatives such as dial-up access, ADSL, cable modems, and leased lines. We have already studied another access option, ISDN.

However, one fact remains. Although there are a number of possibilities for connecting a user's computer to the Internet, all such methods still use the services of an ISP. Therefore, before we study the modern Internet access technologies, we have studied how the architecture of an ISP looks from the inside, in the previous chapter.

16.1 DIAL-UP ACCESS FOR AN INDIVIDUAL USER

16.1.1 Introduction

As we have noted, the Internet backbone and computers connected to it can be accessed in a number of ways. The simplest of them is to directly access the Internet using a high-level Internet protocol

called TELNET, which we shall study later. Protocols such as TELNET are an integral part of the TCP/IP suite of protocols. In such a case, the computer that wants to access the Internet should be connected to the network or to a server, which is connected to the Internet – thus, providing remote access. Clearly, in such a case, the client computer needs to use the TCP/IP suite of protocols for connecting to the server and therefore, access the Internet via the server. But at the same time, for using TELNET, the user has to know how to use the TELNET program, which means a few cryptic commands.

However, the home users want to connect to the Internet *directly* without wanting to remember the cryptic syntax of programs such as TELNET. The term *direct* is used here to mean that they want to connect to the Internet through an Internet Service Provider (ISP), which provides a simple interface over the TELNET program for the end user. Here, the serial mode of data transmission is used. The client computer connects to one of the servers at the ISP through a serial telephone line using a modem. Initially, the client has to log on to the ISP's server, using the user id and password provided. Once the identification is over, the client can connect to the Internet through the ISP's server. Here, the communication between the client and the ISP takes place using two data link layer protocols, viz., the **Point-to-point protocol (PPP)** or **Serial Line Internet Protocol (SLIP)**. These two protocols allow a home user to dial into an ISP over the physical telephone line, by performing the data link layer functions.

In the case of PPP and SLIP, the connection is called **dial up**. This is because the client first *dials in* at the server computer of the ISP and then gets connected to the Internet. The ISP server is already connected to the Internet using TCP/IP. In TCP/IP, each computer connected to the Internet is assigned a unique 32-bit address, called **IP address**, which we will study later. When the ISP starts its business, it *buys* a number of IP addresses on the Internet based on the number of users. It then uses this pool of IP addresses to allocate or deallocate to/from the various users at run time as users log on/off to/from the Internet through the ISP. When the user dials into the Internet through the ISP, the ISP server then allocates a unique IP address temporarily to the client computer. This IP address is determined at the run-time by selecting one from the pool of IP addresses available with the ISP server. Once the session is over, the IP address is returned to the pool of IP addresses at the server. Therefore, it is quite possible that the IP address assigned to a client computer is the same across different Internet sessions. Every time a client computer wants to start a new Internet session by dialing up the ISP server, the ISP server assigns it an IP address that is available at that moment. The ISP server acts as an intermediate point of contact for all the clients. The ISP server takes on the job of sending and receiving IP packets to and from the client.

Thus, using PPP or SLIP, the communication between a client and the rest of the Internet happens in such a manner as if the client computer is directly connected to the Internet. This is shown in Fig. 16.2.

Let us discuss the two protocols in brief. As we have mentioned, these protocols reside in the data link layer.

16.1.2 SLIP

As we know, the main data carrying mechanism in the Internet world is IP packets. Any dial-up connection that is expected to carry IP packets needs to be configured in such a manner that it understands IP packets and can transmit them. To cater to this need, 3COM developed a new mechanism, called the **Serial Line Internet Protocol (SLIP)**. SLIP is not a standard protocol for

carrying IP packets between a home user and the ISP server, over a serial line. However, it has become popular because of its simplicity.

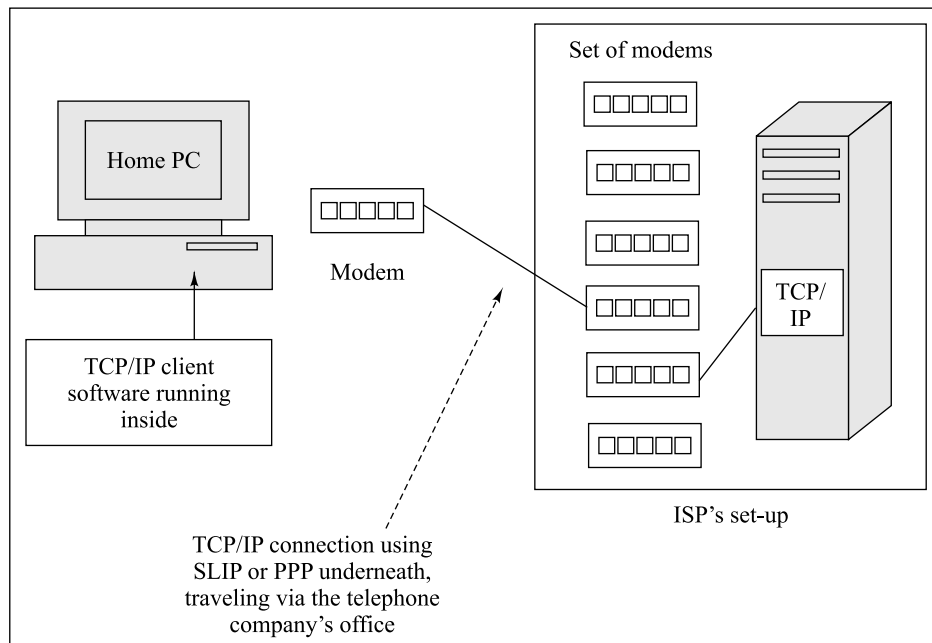


Fig. 16.2 *SLIP/PPP*

SLIP simply takes IP packets from a source and sends it to the destination at the other end, without providing for addressing mechanisms, error control or data compression. This makes it very straightforward to implement. SLIP does not worry about establishing or maintaining the connection between the client and the ISP server. It simply creates a frame over each IP packet and sends these frames. The role of SLIP begins only when there is a proper connection between the two modems. Since SLIP does not have any addressing mechanisms, the client must know the IP address of the ISP server before it connects to it.

However, the simplicity of SLIP also means it is not rich in features. This paved way for PPP.

16.1.3 PPP

The Point-to-Point Protocol (PPP) is another example of the evolving nature of the Internet. PPP was born out of sheer need. However, unlike SLIP, PPP has become a standard Internet protocol for remote access using dial-up connections. The basic functionality of PPP is pretty similar to that of SLIP. It passes IP packets in the form of frames between a client and an ISP server. However, it does more than this. PPP is actually a set of protocols, as described below.

1. Firstly, the **Line Control Protocol (LCP)** is responsible for establishing, maintaining and terminating the connection between the two end points (home user and the ISP). Both the end points must reach an agreement before the communication can start. LCP is used for this purpose. LCP takes up the job of setting up the link between the client and the ISP server.

Thus, the first thing in a PPP session is the exchange of LCP protocols between the client and the ISP server to make sure that the communication link is reliable and functioning. LCP verifies these facts and gives a *go ahead* signal.

2. The second phase is **authentication**. For this purpose, **Password Authentication Protocol (PAP)** is used. Here, the user must establish a proof of identity, so that s/he can use the services of the ISP. Firstly, the user identification (usually user id and password) information is sent to the ISP. At the ISP's end, the PAP software either accepts or denies the connection from the user after the user id and password are checked against the ISP's user database.
3. Once this is done, PPP on the client side sends out a **Network Control Protocol (NCP)** packet. This NCP packet tells the ISP server what kind of traffic is to be passed over this PPP link. This is required in situations where variations of IP (such as IPX and AppleTalk) are used.
4. Finally, the **IP Control Protocol (IPCP)** takes over. Here, actual IP packets are now exchanged. IPCP establishes and terminates a network layer connection between the home user and the ISP.

In conclusion, we can say that although SLIP is much simpler to implement, it also provides no additional services. For these reasons, generally PPP is preferred over it for serial line Internet access.

16.2 LEASED LINES

A large number of medium and large organizations generally need a large amount of bandwidth for connecting to the Internet, because the number of users is very high. For instance, suppose 100 users in an office need Internet access at the same time. In such situations, the simplest option of obtaining 100 dial-up accounts from an ISP is not very attractive. This is because having 100 ISP accounts is not good enough. To connect these 100 users to the ISP, the organization would also need 100 telephone lines. Clearly, this is not acceptable for the reasons of cost and maintenance. Also, the moment a new user is added, a new Internet connection and a new telephone line would be required for that user.

Consequently, telephone companies and ISPs have come up with the option of offering more bandwidth from their premises, and then let the organization divide it internally the way it wants. For this, an ISP provides an option of leasing lines to these kinds of organizations. In simple terms, a **leased line** can be thought of as a very thick pipe connecting the office of an organization with the Internet via the ISP. A medium to big organization obtains a digital line from an ISP for a fixed charge per month, regardless of its actual use. That is, the organization may or may not use the complete bandwidth of the leased line, but it would still pay a fixed charge to the ISP. In return, the organization gets a larger bandwidth from the ISP, which can be shared by multiple users at the same time.

16.3 DIGITAL SUBSCRIBER LINE (DSL)

16.3.1 Introduction

Normally, we connect to the Internet through a regular modem, or through a local area network connection in our office that connects to the ISP via a leased line, or through a **cable modem** (discussed later). More and more people, however, are connecting to the Internet through a **Digital**

Subscriber Line (DSL) connection, which is a very high-speed connection using the same wires as a regular telephone line. DSL is of many types such as ADSL, RADSL, HDSL, VDSL, SDSL, etc. We shall discuss only ADSL here.

A standard telephone connection usually consists of a pair of copper wires, bundled together, that the phone company installs in your home. The pair of copper wires actually has a lot of room (i.e., spare bandwidth) for carrying more data than your phone conversations. The wires are capable of handling a much greater bandwidth (i.e., the range of frequencies than that required for carrying voice). The DSL technology exploits this extra capacity of a normal telephone line to carry information on the line without disturbing the line's ability to carry conversations. The entire plan is based on matching particular frequencies to specific tasks.

A normal telephone line, also called the **Plain Old Telephone Service (POTS)**, limits the frequencies that the switches, telephones and other equipment can carry. Typical human voice has a range of frequencies from 0 to 20000 Hz. The very high and very low frequencies are rarely used (except in hi-fidelity or hi-fi equipment). Ordinary human conversation can be captured quite well without loss of appreciable quality if we capture frequencies between 300 and 3300 Hz, i.e., the bandwidth of 3 KHz. Normally, there are guard bands at both ends making the bandwidth required for the human conversation as much as 4 KHz. Actually, the telephone wires have the potential of handling frequencies up to several million hertz in most cases. Unfortunately, only one user uses the entire twisted wire pair between the home and the telephone exchange of the telephone company (called the *last mile*). Multiplexing is thus not possible here (though it is used heavily between exchanges). The use of such a small portion of the wire's total bandwidth for carrying normal conversations was due to historical reasons. The telephone system has been in place for about a century. By limiting the frequencies carried over the wires, the telephone system can pack many such wires into a very small space without worrying about interference from one wire causing problems to another. However, modern telephone systems use digital signaling. Therefore, there is very little scope for interference between the signals of two or more adjacent wires. The modern equipment that uses digital signaling, rather than analog signaling, can utilize much more of the telephone line's capacity. DSL is based on the same principles.

16.3.2 DSL Basics

For using the DSL services, a user or an Internet customer needs to have a **DSL modem** (discussed later). When the customer connects to a normal telephone line with the help of a DSL modem rather than a standard modem, the telephone line is called a Digital Subscriber Line (DSL). Actually, it is the same old POTS line. A DSL modem is a misnomer in the sense that it does not convert a digital signal into an analog one, unlike a normal modem. Instead, the signal originating from the customer's end travels via the DSL modem as a digital signal through the telephone line. For this, the available range of frequencies on a telephone line is divided into different slots. Frequency modulation technique is used to send digital signals.

How does DSL divide the available range of frequencies in a normal telephone line into many slots? This division is based the assumption that most Internet customers browse or download much more information than they send, or upload. For instance, a customer's Web browser would send requests that consist of a few bytes, using the HTTP protocol. However, in response, it would typically receive millions of bytes of data (especially because these days the Internet documents are no longer only static textual HTML documents, but instead contain multimedia components such as

images, pictures, video, etc.). Under this assumption, if the connection speed from the Internet to the customer (called *downstream*) is 3–4 times faster than the connection speed from the customer back to the Internet (called *upstream*), then the customer will see the most benefit, most of the time. Because the upstream and downstream bandwidths differ, this technology is also called **Asymmetric DSL (ADSL)**. However, it must be pointed out that this is not always beneficial for the end customer. For a customer who uses the Internet for browsing Web pages, downloading information and purchasing items, it works well. However, if the customer is actually a business unit, which is selling products online (and therefore, needs to send catalog information using the upstream), ADSL would not be a good solution, as the customer would actually send more data than it receives.

There are two competing and incompatible standards for DSL. The early standard based on a simpler technology was called the **Carrierless Amplitude Phase (CAP)**. CAP was used on many of the early installations of DSL. The new and official American National Standards Institute (ANSI) standard for DSL is a system called **Discrete MultiTone (DMT)**. According to the DSL equipment manufacturers, most installations follow the DMT standard. Let us briefly discuss these two standards.

- **Carrierless Amplitude Phase (CAP)**

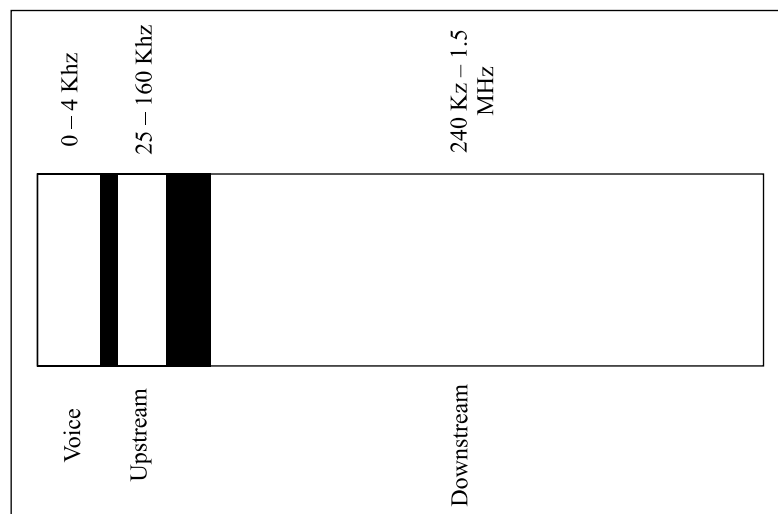


Fig. 16.3 *Carrierless Amplitude Phase (CAP)*

The Carrierless Amplitude Phase (CAP) system divides the total number of signals on the telephone line into three distinct bands as shown in Fig. 16.3.

1. Voice conversations are carried in the 0–4 KHz (kilohertz) band, as they are in all POTS circuits.
2. The upstream channel (from the customer back to the telephone office) is carried in a band between 25–160 KHz.
3. The downstream channel (from the telephone office to the customer) begins at 240 KHz and goes up to a limit that changes, depending on quite a few conditions such as the line length, the noise, the number of customers in a particular telephone company switch, etc. In any case, it has a maximum limit of about 1.5 MHz (megahertz).

Since the three channels are widely separated, the possibility of interference between the channels on one line, or between the signals on different lines is minimized.

- **Discrete MultiTone (DMT)**

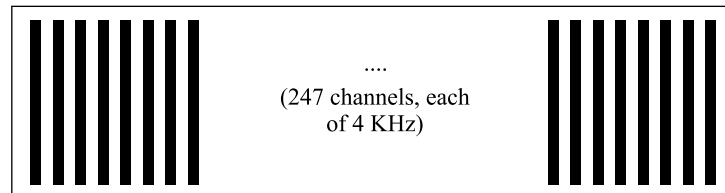


Fig. 16.4 *Discrete MultiTone (DMT)*

As shown in Fig. 16.4, like CAP, the Discrete MultiTone (DMT) system also divides the total signal capacity into separate channels. However, unlike CAP, it does not use two broad channels for upstream and downstream data. Instead, DMT divides the whole bandwidth into 247 distinct channels, each of 4 KHz capacity. Conceptually, it is like dividing the normal telephone line into 247 different 4 KHz lines and then attaching a modem to each one of them. A customer gets the equivalent of 247 modems connected to her/his computer at once.

Each of these channels is monitored constantly. If the quality is too bad, the signal is shifted to another channel. Thus, this system constantly shifts signals between different channels, searching for the best available channels for transmission and reception. Additionally, some of the lower channels (those starting at about 8 KHz) are used as bidirectional channels, for both upstream and downstream data transport. Monitoring and sorting out the information on the bidirectional channels, and keeping up with the quality of all 247 channels makes DMT more complex to implement than CAP, but at the same time gives it more flexibility on lines of differing quality.

16.3.3 DSL Requirements

For using the DSL technology, a unit similar to the normal modem is required at the customer's end. This unit is called a **DSL transceiver** or **DSL modem**. At the telephone company's end, we have a **DSL Access Multiplexer (DSLAM)**. Let us understand their functionalities.

- **DSL modem** – A DSL modem is the point where data from the customer's computer or network is connected to the telephone line. The DSL modem can connect to a customer's equipment in several fashions, although most residential installations use a USB or 10-baseT Ethernet connections. While most of the DSL modems sold by ISPs and telephone companies are simply DSL modems (as expected), similar device used by organizations may combine network routers, network switches and other networking equipment in the same unit.
- **DSLAM** - The DSLAM installed at the telephone company's end is the equipment that takes connections from many DSL customers and aggregates them onto a single, high-capacity connection, which is then routed via an Internet Service Provider (ISP) to the Internet. This is shown in Fig. 16.5.

A DSLAM is flexible in terms of supporting multiple types of DSL protocol and modulation (which we shall study shortly). The DSLAM provides one of the main differences between user service through DSL and through cable modems. Because cable modem customers generally share a network loop that runs through a neighborhood, additional customers usually mean

a lowered performance (we will study cable modems in the next section). In contrast, DSL provides a dedicated connection from each customer back to the DSLAM, meaning that customers won't see a performance decrease as new customers are added, unless, of course, the total number of customers begins to saturate the single, high-speed connection between the DSLAM and the ISP. At that point, an upgrade by the telephone company can provide a better performance to all the customers connected to that DSLAM.

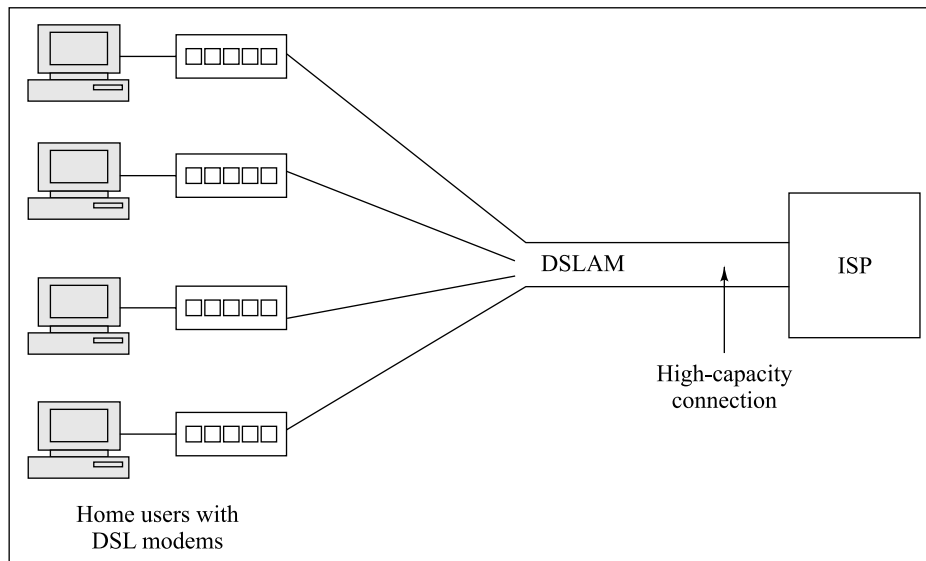


Fig. 16.5 DSL modems and DSLAM

16.3.4 Carrying Voice and Data using DSL

It must be pointed out that how much benefits a DSL customer gets, greatly depends on how far the customer is located from the central office (i.e., the telephone exchange) of the telephone company that provides the DSL service (called DSL service provider). DSL is, therefore, called a *distance-sensitive technology*. This means that as the length of the connection between the telephone company and the customer increases, the signal quality degrades more and more, and the connection speed also goes down. The maximum limit for DSL service is about a distance of 18,000 feet (5,460 meters) between the telephone company and the customer. However, for achieving higher connection speed and better quality of service, many DSL service providers place a limit lower than this, on the distances for the service. Consequently, at the extreme end of the distance limit, DSL customers may see connection speeds far below the promised maximums. On the other hand, the customers, which are closer to the DSL service provider, have the potential for enjoying very high connection speeds in future. For instance, the DSL technology can provide maximum downstream (Internet to customer) speeds of up to 8 megabits per second (Mbps) at a distance of about 6,000 feet (1,820 meters), and upstream speeds of up to 640 kilobits per second (kbps). In practice, the best speeds widely offered currently are 1.5 Mbps downstream, and 64–640 kbps upstream.

At this stage, the following two questions surface:

1. Why is the distance between a customer and the telephone office (i.e., DSL service provider) an issue in case of DSL?

2. If distance is a limitation for DSL, why is it not also a limitation for normal voice telephone calls?

We shall answer the second question first. The answer lies in small amplifiers called *loading coils* that the telephone company uses to boost voice signals. These loading coils are placed strategically one after the other at a distance of a few meters from each other by the telephone company, so that they boost/amplify the incoming voice signals as they pass along the wire. The aim is to enable them to remain strong throughout their journey, for the receiving end to be able to detect and interpret them. This is why there is no problem of distance in case of voice telephone signals. Then, why is distance an issue in case of DSL?

The trouble is that the loading coils used for boosting voice signals are incompatible with DSL signals. Therefore, if there is a voice coil in the loop between a customer's telephone and the telephone company's office, the customer would not be able to get a DSL access at all.

This is why although a DSL connection uses the same telephone wire as the underlying physical medium; it is functionally not the same as a voice connection. When a customer changes over from a normal telephone connection to a DSL connection, the voice connection actually becomes a part of the overall DSL connection, as we have noted earlier. This means that if there are any voice coils, they must be removed first.

It should be noted that ISDN is another technology for obtaining high data rates using the existing wiring. We have studied ISDN earlier, and would not discuss it further.

16.4 CABLE MODEMS

Cable television has become extremely popular all over the world. Various television software companies broadcast their programs such as serials to sports programs and live talk shows, which are brought to their subscriber's homes by cable service providers. Just as a telephone network employs twisted pair wires (or these days, optical fibers), the coaxial cable (also called broadband wire) runs from a cable company to each individual's house. The coaxial cable carries the necessary signals for all the channels that the cable company broadcasts. An individual's television set is tuned to those frequencies, thereby allowing the television set to *catch* those programs. Therefore, signals for the programs for all the channels actually arrive at the TV set in the house over the cable. When we change a channel using the remote control, we select the frequency bandwidth allocated to that program. Only those signals are then selected by our TV set and the program starts showing. Thus, the basic principle of FDM is used here.

Firstly, the capacity of a coaxial cable is huge. It is much larger than what is required for the TV programs. Therefore, the unused capacity can be utilized to carry Internet traffic. Also, the cable television networks are already in place in most parts of the world where the Internet is popular. So, new wiring is not required. For these two reasons, the thought of providing Internet access to a home user along with cable television access became popular. The overall architecture of Internet access via cable modems is shown in Fig. 16.6.

Let us understand how this works, step by step.

1. When a user wants to access the Internet, s/he invokes the computer's browser as usual. Internally, the browser's requests reach the Network Interface Card (NIC) of the computer.

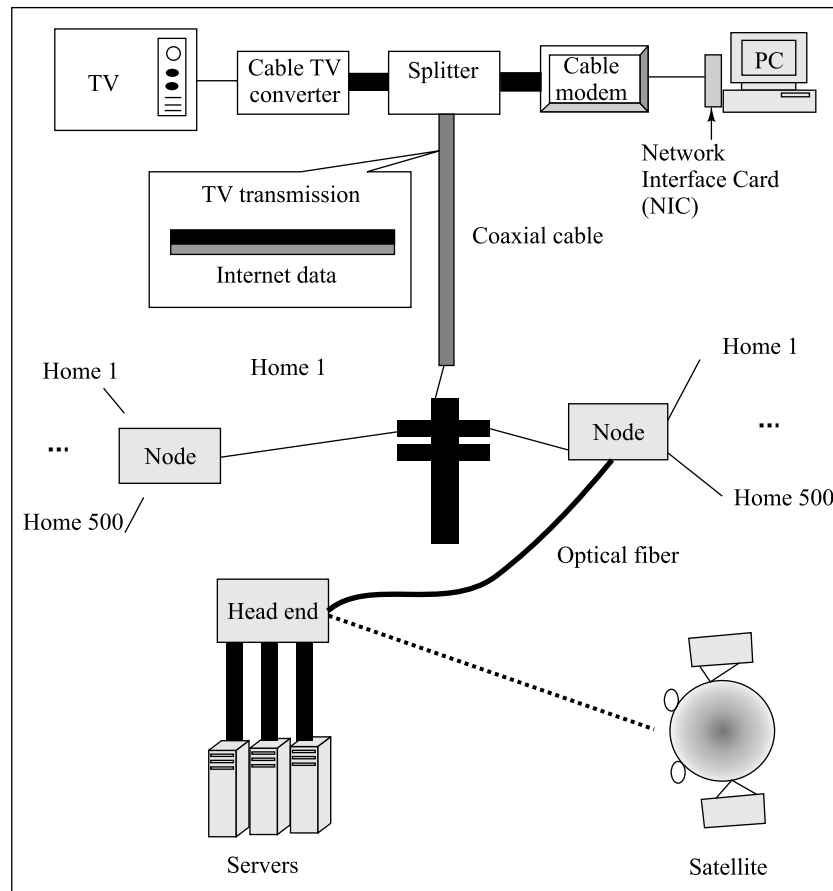


Fig. 16.6 Internet access by cable

2. As shown in Fig. 16.6, a device called **splitter** is fitted inside the premises of a cable TV user. As the name suggests, the splitter is a device that splits the signals inside a coaxial cable into two parts. One part of the signal is the usual television signal. However, the other part of the signal carries Internet data (i.e., IP packets in the analog way like the other TV signals are carried). These two separate parts of the signal arrive in separate wires. One wire goes to the television set as usual. The other wire carrying IP packets is connected to the **cable modem**. Like an ordinary modem, the basic job of a cable modem is to convert a computer's digital data into analog signals that the wire can carry, and vice versa. The cable modem then connects to the Network Interface Card (NIC) of a computer, as usual. The user can now use the same cable connection to view television as well as to access the Internet.
3. From the cable modem, the requests go to the coaxial cable. The coaxial cable carries both television and computer signals simultaneously. So, if you are also watching television at the same time, there would be no problems. The computer signals travel on a 6 MHz channel on the coaxial cable.

4. The cable company serves each town through a number of **central nodes** – the main point on the cable company’s network. Each node services about 500 customers as shown in the figure. The cable company divides each town into about 500 homes, all of which are located on a single node. All these homes connect to the same node.
5. Many such nodes are connected via high-speed optical fiber links into a single **head end** cable facility. You can roughly equate a node to an ISP and a head end to a NAP. A single head end typically connects about 4–10 nodes. The head end is responsible for the delivery of television programs and Internet access to the cable customers.
6. The head end, in turn, receives television signals from satellites and has Internet access via high-speed connections with other major NAPs. Thus, these television and Internet connections are responsible for providing cable customers with cable television and Internet access.
7. Additionally, just as an ISP hosts its own Web pages on a Web server or allows its subscribers to access emails by hosting a SMTP server, a head end might also host such servers for these additional services to its cable customers.

16.5 DTE-DCE INTERFACE

In any computer network, we speak of two kinds of equipments, viz., **Data Terminal Equipment (DTE)**, and **Data Circuit-terminating Equipment (DCE)**. In general, we can roughly equate computers with DTE and the actual data transmission equipment as DCE. Thus, a DTE generates data along with the required control characters, and passes them onto the DCE. The DCE converts this into a signal form that is appropriate for the transmission medium and sends it over the network. The process gets reversed at the receiver’s end. We can show this concept in a diagrammatic fashion as illustrated in Fig. 16.7.

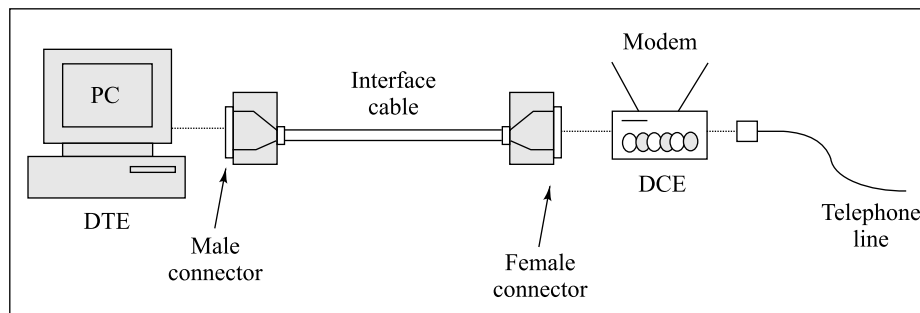


Fig. 16.7 DTE-DCE interface

The protocol that defines the interface between a DTE and a DCE is called **RS-232**. Initially, the RS-232 protocol was used for serial communication over telephone lines. Initially, the RS-232 standard specified a 25-pin connector. However, subsequently, it was realized that many of the pins were not redundant, and as a result, the number of pins in the connector was reduced to 9. Every pin has a specific meaning and purpose. There are two types of connectors, viz., male and female. A male connector can only attach to a female connector, and vice versa. Table 16.1 shows the meaning and purpose of every pin in a 9-pin connector.

Table 16.1 RS-232 9-pin connector DCE and DTE

Pin Number	DTE (Male connector)	DCE (Female connector)
1	Received line signal detect	Received line signal detect
2	Received data	Transmitted data
3	Transmitted data	Received data
4	DTE ready	DTE ready
5	Signal ground	Signal ground
6	DCE ready	DCE ready
7	Request to send	Clear to send
8	Clear to send	Request to send
9	DCE ready	DCE ready

Using RS-232, it is possible to send data at a rate of about 20 Kbps over a distance of up to 15 meters. Special control circuits are used to manage the connection between a DTE and a DCE. Now, the RS-232 standard itself is largely superseded by the **Universal Serial Bus (USB)** standard.

16.6 EIA RS-232 AND EIA RS-449 INTERFACE

The **Electronics Industries Association (EIA)** has come up with various standards for data communication. We have already discussed one such standard briefly earlier, which is called RS-232. The RS-232 standard includes the definition of the following features:

1. Characteristics of the electrical signals such as the voltage pulse levels, signal timing, rate of signaling, characteristics pertaining to short-circuiting, etc.
2. Responsibility of all the circuits in the interface connector
3. Interface circuits for a set of defined telecommunication and data communication applications
4. Mechanical and pin specifications for the connectors

However, the following points are excluded from the standard:

1. Mechanism used for character encoding such as ASCII, UNICODE, or EBCDIC
2. Error detection and data compression algorithms
3. Bit rates used for actual data transfers
4. Supplying power to external devices

RS-232 considers voltages between ± 3 and 15 volts as valid. Everything else is considered as an error. In general, the 1 bit is defined as a negative voltage value, and a 0 bit as a positive voltage value.

As mentioned earlier, as technologies progressed, support for RS-232 started getting deprecated and instead, the USB standard was favored. USB is faster, needs less power, and has easier connectors to work with. It is mainly used for connecting external devices to a computer as well as for the communication between a computer and its device drivers.

Like RS-232, **RS-449** is another specification for DTE-DCE communication. It also defines mechanical and functional characteristics of the interface between a DTE and a DCE. However, other standards superseded RS-449, and hence it was withdrawn and replaced by newer standards. Ideally, RS-449 should have replaced RS-232. But things did not turn out that way. RS-449 is a

high-speed digital interface. It uses a 37-pin connector. Specifications for each pin are defined like in any other standard. RS-449 uses two wires, which are twisted around one another. The difference of voltage between these two wires is measured for the purpose of understanding what data is being transmitted. The two signals on the wires are either called A and B or + and -. The connection is always between two A points or two B points, i.e., between two + points or two - points.

16.7 SONET/SDH – SYNCHRONOUS TRANSPORT SIGNALS

There are two popular protocols that allow multiplexing of multiple digital signals (bit streams) over optical fiber. The mechanism used could be lasers or Light Emitting Diodes (LED). These protocols are called **Synchronous Optical Networking (SONET)** and **Synchronous Digital Hierarchy (SDH)**. SONET and SDH essentially mean one and the same thing from a general point of view. However, for technical accuracy, we should mention that SDH is a superset of SONET. One more point we should note is that the bodies that have developed these standards are different. SONET was developed by ANSI, and SDH was developed by ITU-T.

SONET/SDH supports simultaneous transfer of signals from multiple different origins. Also, this is done within the boundaries of a single framing protocol. We must also clarify that SONET/SDH is not itself a communications protocol. It is more of a transport protocol. For example, SONET/SDH is used for transferring ATM cells. SONET/SDH is widely used across the world.

Interestingly, the structure of data inside a SONET/SDH frame is quite complex. There are headers between data items. Hence, different frames can have their own data rates and other complexities. The basic unit of transport in SDH (like a frame) is called **STM-1 (Synchronous Transport module, level 1)**. It offers data rates of 155.52 Mbps. In SONET, this unit is called **STS-3c (Synchronous Transport Signal 3 concatenated)**. In STS-1, each frame contains 810 octets. On the other hand, an STM-1 frame contains 2430 octets.

We can draw a block diagram for showing a typical SONET network, as illustrated in Fig. 16.8.

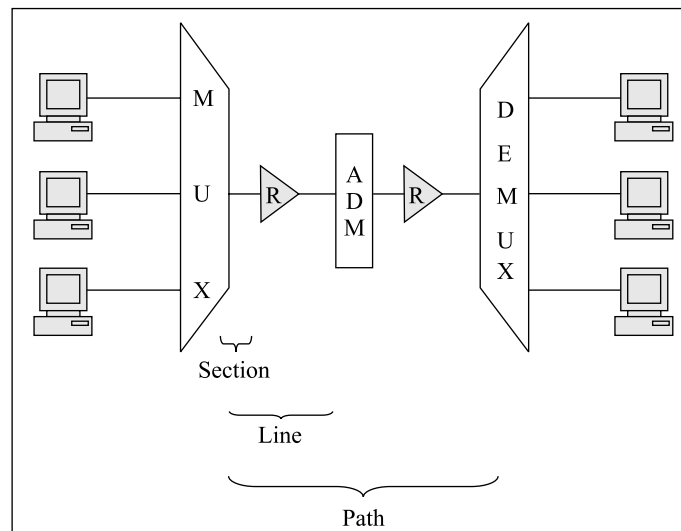


Fig. 16.8 SONET network

As we can see, a SONET network creates a multiplexed network of computers. The information gets de-multiplexed on the other side. We have also used certain terminologies such as section, line, and path. These would be explained shortly. Also, the device in the middle is called Add/Drop Multiplexer (ADM).

In packet-based networks such as Ethernet, a frame is made up of a header and a payload. First the header is transmitted, followed by the payload and the trailer, if there is one. In SONET, there are minor variations. The header is called *overhead*. It is not sent at the beginning of the transmission. Instead, the overhead is sent intermittently along with the payload. Hence, some part of the overhead is transmitted, followed by some part of the payload. This is again followed by a similar cycle many times.

The internal formats of the SONET frame and the SDH are different. The terminology used to describe them is also different. However, the implementation details for SONET and SDH are quite similar. As a result, these two technologies are nearly interchangeable from an implementation point of view.

16.8 SONET LAYERS—APPLICATIONS

SONET is considered to be made up of four layers, starting with the bottommost layer upwards, i.e., **Photonic layer**, **Section layer**, **Line layer**, and **Path layer**. Let us understand these layers in brief, as shown in Table 16.2.

Table 16.2 SONET layers

Layer	Description
Photonic layer	Specifies the electrical and optical interface for transporting data over an optical fiber. This layer performs the job of converting STS electrical pulses into optical light pulses on the sender's end, and the optical light pulses back into electrical pulses at the receiver's end. This layer directly transmits bits on the physical medium. From the input electric pulses, it creates appropriate optical signals. This layer is therefore responsible for low-level issues such as deciding the shape of the signal pulse, deciding how much of power is required, and wavelength-related issues.
Section layer	This layer transfers the STS frames over an optical fiber. Many people compare this layer with the data link layer in the OSI protocol suite. Framing and physical transfer of data are handled here. This layer also handles the functions of error monitoring, section maintenance, etc.
Line layer	The line layer is responsible for the synchronization and multiplexing features needed by the layer above, i.e., the path layer. This layer also provides for a feature called automatic protection switching, which makes use of additional spare capacity in case the primary allocation is inadequate. This layer ensures that the payload and overhead are successfully transmitted. This layer modifies certain bits in the overhead pertaining to quality control issues.
Path layer	This highest layer interfaces with the services such as ISDN, T1, etc., and maps them into SONET/SDH format. This layer is responsible for end-to-end communication, maintenance, and control.

We can compare the data link layer of the OSI model with the path, line, and section layers of the SONET protocol stack, and the physical layer of the OSI model with the photonic layer of the SONET protocol stack. This is shown in Fig. 16.9.

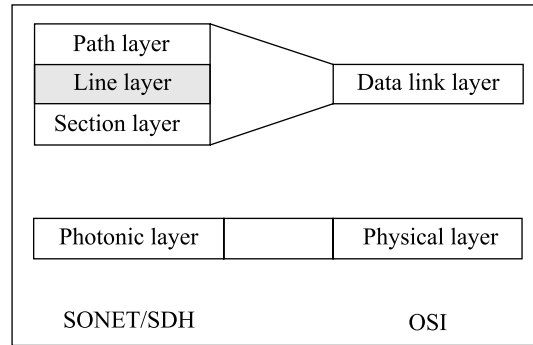


Fig. 16.9 *Equivalence of SONET/SDH and OSI protocol stack layers*

SONET signals are called STS-n signals. A SONET STS-n signal is transmitted at the rate of 8000 frames per second. Every byte in such a frame can carry one digitized voice channel. For example, the STS-1 channel sends 8000 frames per second. Each STS frame consists of $9 \times (1 \times 90) = 8100$ bytes, and because every byte is made up of 8 bits, the effective data rate is $8000 \times 9 \times (1 \times 90) \times 8 = 51.840$ Mbps. Similarly, for the STS-3 channel, the data rate would be $8000 \times 9 \times (3 \times 90) \times 8 = 155.52$ Mbps. Every frame takes 125 micro seconds to transfer.

SUMMARY

A home/office user can access the Internet in a number of ways. In an office, the whole network could be connected to the Internet through a single computer to which every computer in the network attaches.

The simplest mechanism for accessing the Internet is to have a dial-up account with an Internet Service Provider (ISP). This is the cheapest and simplest method for accessing the Internet. However, it is also the slowest. Two protocols that allow this are Point-to-point protocol (PPP) and Serial Line Internet Protocol (SLIP). PPP is modern and more popular of the two, as it allows for error checking.

Many organizations these days use the services of leased lines. Here, an organization leases a telephone line for a prefixed charge every month, regardless of the actual usage. This allows multiple users to share the single common line, as in the case of offices.

The Digital Subscriber Line (DSL) is a more recent phenomenon. In DSL, the existing wire set-up is reused to provide much higher data rates. This is made possible by using the full bandwidth of the copper telephone wire, unlike what is done in normal telephone conversations. This is a very useful facility for a home user. It is based on the assumption that the *downstream* (i.e., Internet to customer) traffic is much more than the *upstream* (i.e., customer to Internet). This assumption allows much higher download speeds than otherwise possible with an ordinary dial-up connection. DSL can be of two types, viz., Carrierless Amplitude Phase (CAP) and Discrete MultiTone (DMT).

These days, accessing the Internet using cable television is also very popular. Since the bandwidth of the coaxial cables used in television programming is much higher than what is actually used for cable television transmissions; the unused excess capacity can be used for carrying Internet traffic. A cable modem is required at the customer's end to receive and decipher Internet traffic signals. The transmission capacity of cable Internet is also much higher than the ordinary dial-up Internet access.

DTE and DCE are very common terms in the area of serial data transfer. Various standards emerged for this purpose, of which RS-232 became most popular for connecting analog transmission lines to digital data. SONET and SDH are examples of high-speed multiplexed optical fiber-based networks.

KEY TERMS AND CONCEPTS

Asymmetric DSL (ADSL)	Password Authentication Protocol (PAP)
Authentication	Path layer
Cable modem	Photonic layer
Carrierless Amplitude Phase (CAP)	Plain Old Telephone Service (POTS)
Central node	Point-to-point protocol (PPP)
Data Circuit-terminating Equipment (DCE)	RS-232
Data Terminal Equipment (DTE)	RS-449
Dial up	Section layer
Digital Subscriber Line (DSL)	Serial Line Internet Protocol (SLIP)
Discrete MultiTone (DMT)	Splitter
DSL Access Multiplexer (DSLAM)	STM-1 (Synchronous Transport module, level 1)
DSL modem	STS-3c (Synchronous Transport Signal 3 concatenated)
DSL transceiver	Synchronous Digital Hierarchy (SDH)
Electronics Industries Association (EIA)	Synchronous Optical Networking (SONET)
Head end	TELNET
IP Control Protocol (IPCP)	Universal Serial Bus (USB)
Leased line	
Line Control Protocol (LCP)	
Line layer	

QUESTIONS

True/False

1. SLIP is preferred to PPP.
2. SLIP is much simpler to implement than PPP.
3. A leased line can be thought of as a very thick pipe connecting the office of an organization to the Internet via the ISP.
4. DSL is a very high-speed connection that uses the same wires as a regular telephone line.
5. DMT uses two broad channels for upstream and downstream data.
6. The basic job of a cable modem is to convert a computer's digital data into analog signals that the wire can carry, and vice versa.
7. RS-232 is an example of serial data transmission.
8. Data originates from DCE.
9. SONET is a synchronous network.
10. Multiplexers are needed in SONET/SDH.

Multiple-Choice Questions

1. In SLIP/PPP, the connection is called _____.
 (a) dial left (b) dial right
 (c) dial up (d) dial down
2. SLIP is _____ as compared to PPP.
 (a) simpler (b) complex
 (c) more comprehensive (d) None of the above
3. _____ is responsible for setting up a connection between a user and the ISP.
 (a) NCP (b) PAP
 (c) LCP (d) SLIP
4. _____ tells what kind of traffic is to be passed over a PPP link.
 (a) NCP (b) PAP
 (c) LCP (d) SLIP
5. The DSL technology is based on exploiting the unused bandwidth of a _____ line.
 (a) telephone (b) cable
 (c) leased (d) electricity
6. ADSL technology is devised with a view that an Internet user _____ data.
 (a) uploads more, downloads less (b) uploads less, downloads more
 (c) uploads and downloads equal (d) does not upload or download
7. CAP logically divides the transmission line into _____ bands.
 (a) 5 (b) 4
 (c) 3 (d) 2
8. DMT divides the whole bandwidth into _____ distinct channels.
 (a) 500 (b) 300
 (c) 800 (d) 247
9. Cable signaling uses _____.
 (a) TDM (b) PCM
 (c) FDM (d) PAM
10. A _____ is used to divide the signals into television signals and Internet data.
 (a) splitter (b) divider
 (c) broker (d) divisor

Detailed Questions

1. What are SLIP and PPP? How are they different?
2. Explain the typical dial-up connection between a home user and an ISP.
3. What is a leased line? What purpose does it serve?
4. How can DSL provide faster access than POTS?
5. Discuss the two DSL standards.
6. Describe the requirements for DSL.
7. How can a user access Internet over the cable television? What are the requirements for this?
8. Discuss the concepts of DTE and DCE.
9. What is RS-232? Explain its pin configurations.
10. Write a note on the various layers in SONET/SDH.

TCP/IP

Part 1: Introduction to TCP/IP, IP, ARP, RARP and ICMP

17

17.0 INTRODUCTION

The **Transmission Control Protocol/Internet Protocol (TCP/IP)** suite of protocols forms the basis of Internetworking (note the uppcase, which means that we are referring to the worldwide network of computer networks). It is TCP/IP that creates an illusion of a virtual network when multiple computer networks are connected together. TCP/IP was developed in the early 1970s. Interestingly, the development of Local Area Networks and of course, the Internet, came about during the same time. Thus, TCP/IP grew with the Internet and because LANs also became popular soon; connecting LANs was one of the early goals of TCP/IP. In fact, when multiple networks with multiple frame/datagram formats and also multiple other algorithms (routing, error control, compression, etc.) are to be connected, there are two alternatives: (a) Protocol conversion, (b) A universal protocol with its frame/datagram size and other algorithms operating at every node in every network in addition to the existing protocols with algorithms of converting to/from that network's frame/datagram from/to the frame/datagram of the universal protocol. TCP/IP uses the latter philosophy.

We have seen the OSI model of network protocols. TCP/IP was developed before OSI. Therefore, the layers present in TCP/IP do not match exactly with those in OSI. Instead, the TCP/IP suite consists of three layers as shown in Fig. 17.1. Although TCP/IP does not define the lowermost two layers (data link layer and physical layer), we have shown them for the sake of completeness.

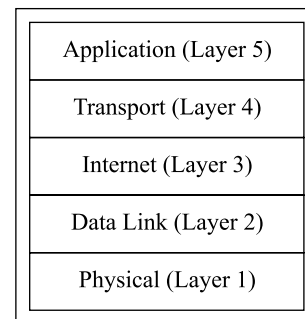


Fig. 17.1 TCP/IP Layers

Layer 1 – Physical Layer (Not a Part of TCP/IP)

The physical layer in TCP/IP is not different in any way to the physical layer of the OSI model. This layer deals with the hardware level, voltages, etc., and there is nothing significantly different here in case of TCP/IP.

Layer 2 – Data Link Layer (Not a Part of TCP/IP)

The data link layer is also very similar to other network models. This covers the **Media Access and Control (MAC)** strategies, i.e., who can send data and when, etc. This also deals with the frame formats (e.g., Ethernet) and so on.

Layer 3 – Internet Layer or Network Layer

The Internet layer is very important from the context of the communications over an internet or the Internet. This layer is concerned with the format of datagrams as defined in the **Internet Protocol (IP)** and also about the mechanism of forwarding datagrams from the source computer to the final destination via one or more routers. Thus, this layer is also responsible for actual routing of datagrams. This layer makes internetworking possible, and thus, creates an illusion of a virtual network. We shall study this in detail.

The IP portion of the TCP/IP suite deals with this layer. This layer follows a datagram philosophy. That is, it routes and forwards a datagram to the next hop, but is not responsible for the accurate and timely delivery of all the datagrams to the destination in a proper sequence. Other protocols in this layer are **Address Resolution Protocol (ARP)**, **Reverse Address Resolution Protocol (RARP)** and **Internet Control Message Protocol (ICMP)**.

Layer 4 – Transport Layer

There are two main protocols in this layer, viz., **Transmission Control Protocol (TCP)** and **User Datagram Protocol (UDP)**. TCP ensures that the communication between the sender and the receiver is reliable, error-free and in sequence. The IP layer sends individual datagrams through various routers, choosing a path for each datagram each time. Thus, different datagrams may reach the destination via different routes and may reach out of sequence. In addition, a datagram may not reach the destination correctly. The IP does not even check the CRC for the data in each datagram at each router. At the destination, the TCP software is responsible for checking the CRC, detecting any errors, reporting them, and acknowledging the correct delivery of datagrams. Finally, it also sequences all the datagrams that are received correctly, to form the original message. TCP uses a **sliding window** technique, so that multiple datagrams can be sent and acknowledged in one shot, instead of waiting for the acknowledgement of each datagram before sending the next one. As we shall see later, **UDP** is also used in this layer. However, UDP does not offer reliability. It is, therefore, way faster. Whenever slight variations as much as about speed are not a major consideration (i.e., in case of voice or video), you can use UDP. However, it is advisable to use TCP for sending data such as banking transactions.

Layer 5 – Application Layer

Like the OSI model, the application layer allows an end user to run various applications on the Internet and use the Internet in different ways. These applications (and various underlying protocols) are **File Transfer Protocol (FTP)**, **Trivial File Transfer Protocol (TFTP)**, **email (SMTP)**, **remote login (TELNET)**, and the **World Wide Web (HTTP)**. The application layer corresponds to layers 6 and 7 in the OSI model. The layer 5 of OSI, i.e., the session layer is not very important in the case of TCP/IP. Therefore, it is almost stripped or ignored.

17.1 TCP/IP BASICS

Before we discuss TCP/IP in detail, let us draw a diagram of the various subprotocols that together compose the TCP/IP software, as shown in Fig. 17.2 and Fig. 17.3. We shall discuss all these protocols in detail subsequently.

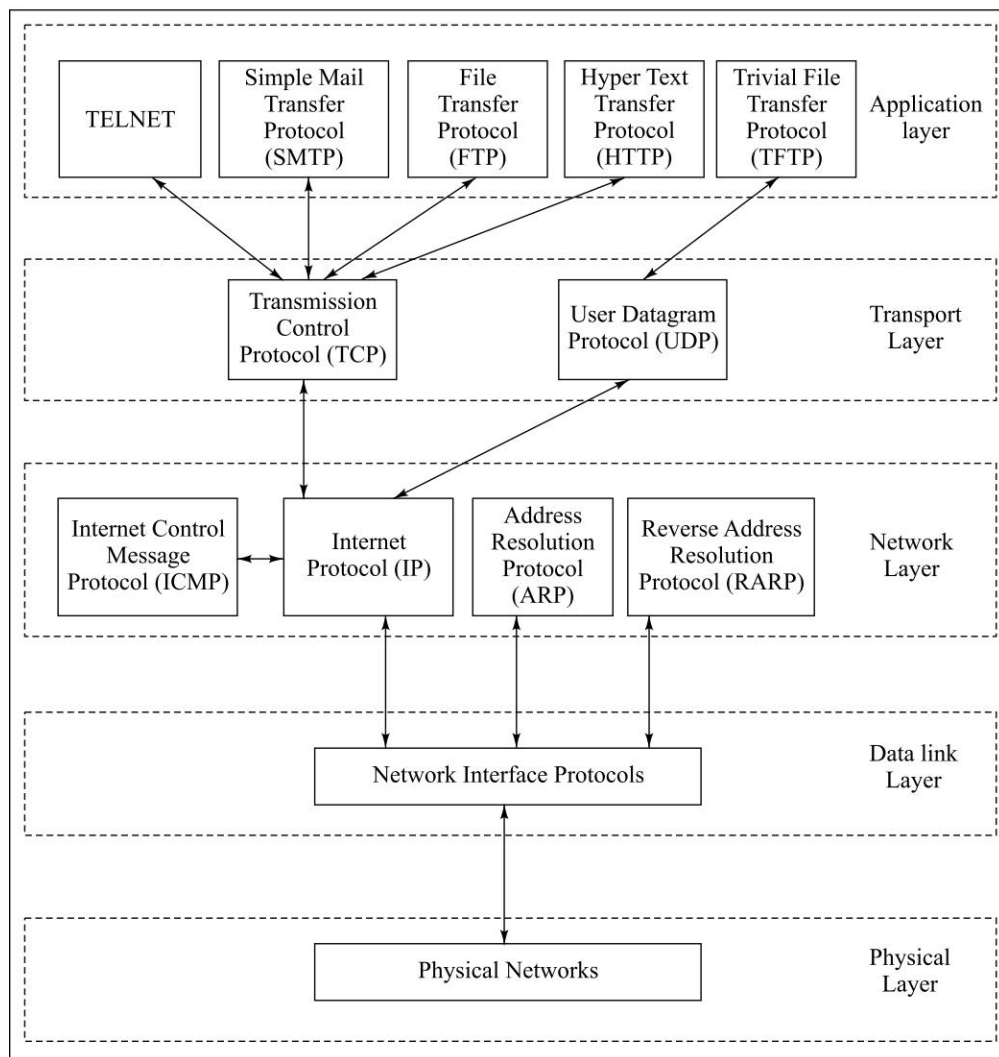


Fig. 17.2 Protocols in the TCP/IP suite at different layers

It is important to know the mapping between the OSI model and the TCP/IP model. That gives us a good understanding of where things fit in. Figure 17.3 shows the encapsulation of data units at different layers of the TCP/IP protocol suite in comparison with the OSI model. As the figure shows, the data unit initially created at the application layer is called a *message*. A message is then broken down into *user datagrams* – or just *datagrams* (also called *datagrams* or *segments*) by TCP or UDP in the transport layer. The datagrams are then encapsulated into *frames* by the data link layer. Remember, if for instance, Ethernet is an underlying network; the entire thing consisting of IP header + TCP header + datagram is treated as the data portion of the Ethernet frame to which Ethernet frame header is attached. This frame header contains the addresses of the source and the destination nodes, which are the physical addresses of the adjacent routers on the chosen path.

This is how the datagram travels from router to router using the underlying network. When the frame reaches the next router, the CRC is checked, the frame header dropped, the final source and destination addresses are checked in the IP/TCP headers, the next router is decided based on the path, and again the datagram is encapsulated in the frame format of the underlying network connecting those routers. When all the datagrams reach the destination node, TCP at that node puts all the datagrams together to form the original message.

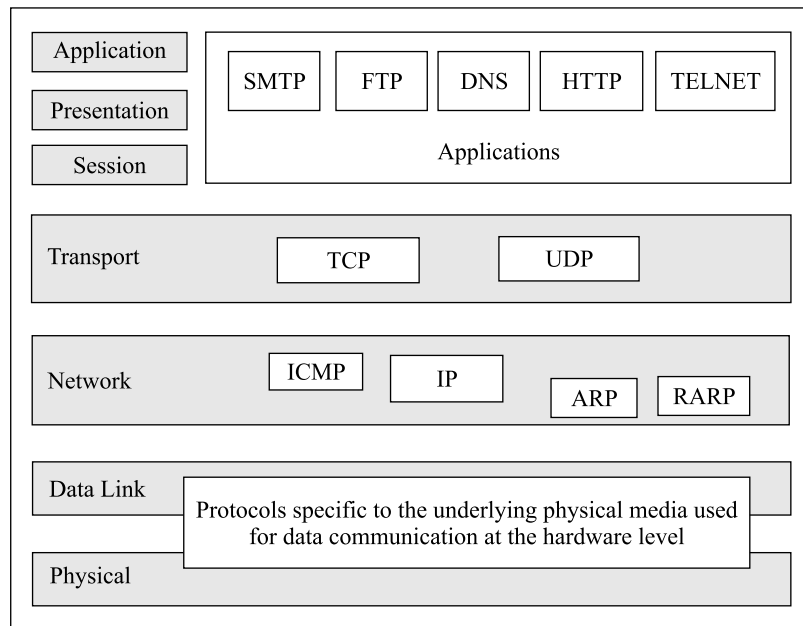


Fig. 17.3 Mapping of TCP/IP protocols with the OSI model

At a broad level, each datagram of a message travels from the source to the destination as shown in the figure. For simplicity, it is assumed that the message is very small and fits in one datagram/frame.

At the lowest level (physical layer), if a datagram is to be sent from node A to node E over the Ethernet shown in Fig. 17.5, it will ultimately be sent as an Ethernet frame by appending the frame header by node A. At node E, the header will be discarded and the original datagram will be retrieved.

Finally, the frames are sent as bits in the form of voltages by the physical layer.

17.2 WHY IP ADDRESSES?

How does the Internet Protocol (IP) know where a datagram comes from and where it has to be delivered? Recall our previous discussion of a virtual network. The primary goal of the Internet is to provide an abstracted view of the complexities involved inside it. For a common user, the Internet must appear as a single large network of computers, and the fact that it is internally a network of many incompatible networks must be hidden from her/him. At the same time, people dealing

with the networks that make up the Internet must be free to choose the hardware and networking technologies that suit their requirements, such as Ethernet, Token Ring, SNA, etc. This means that a common interface is required to bind the two views together, i.e., one of an end user of the Internet and the other of the people dealing with their own networks.

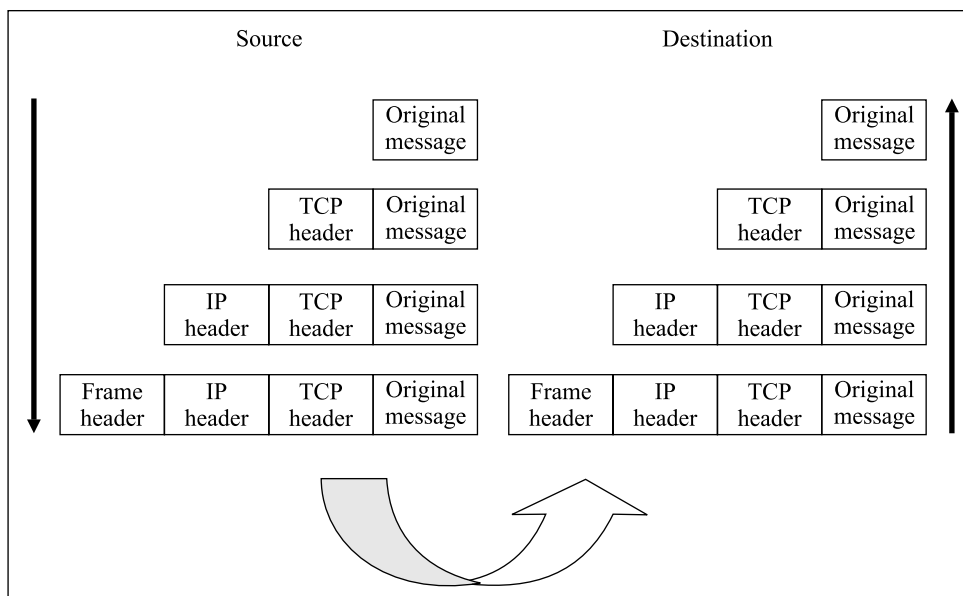


Fig. 17.4 Message transfer from the source to the destination at different layers

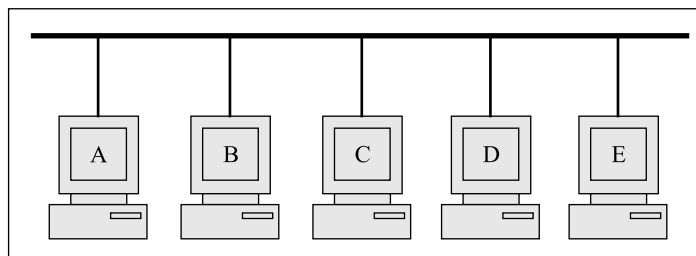


Fig. 17.5 Sample Ethernet

Consequently, identifying a computer over the Internet is a big challenge. Different networking technologies have different *physical addressing* mechanisms. What is the *physical address* or *hardware address* of a computer? There are three methods to assign the hardware address to a computer:

1. **Static addresses** – In this scheme, the physical address is hard coded in the Network Interface Card (NIC) of the computer. This address does not change, and is provided by the network hardware manufacturer.
2. **Configurable addresses** – In this case, the physical address is configured inside a computer when it is first installed at a site. Unlike a static address, which is decided and hard coded

by the hardware manufacturer, a configurable address allows the end customer to set up a physical address.

3. **Dynamic addresses** – In this scheme, every time a computer boots, a server computer dynamically assigns it a physical address. This also means that the physical address keeps on changing every time a computer is switched off and on. A pool of free physical addresses is used to identify free addresses out of them, and one such free address is assigned to the newly booting computer. Instead of using one of the pool of free addresses, the dynamic address can be generated as a random number at run time, and it is ensured that the same random number is not used as a physical address by any other computer in that network.

We shall not discuss these types further. However, it is essential to remember that static addresses are the simplest of the three, and therefore, are most popular as they need no manual interventions or dynamic processing. In any case, the point is that there is a unique physical address or hardware address for every computer on a network. This address gets stored on the NIC of that computer. The network must use that address for delivery of messages to that computer.

As we know, the NIC is simply an I/O interface on a computer that allows communication between the computer and all other computers on a given network. The NIC is a small computer having a small amount of memory, which is responsible for the communication between the computer itself and the network to which it is attached. This is shown in Fig.17.6.

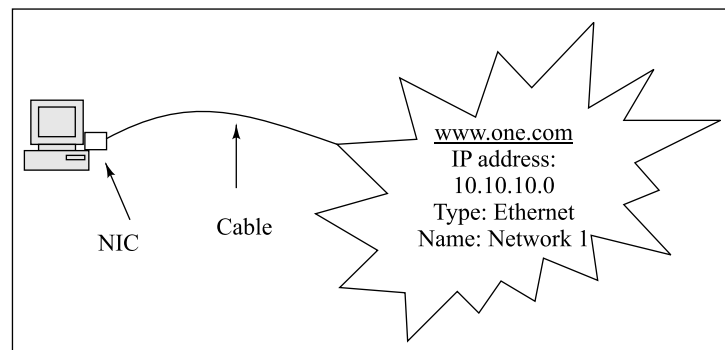


Fig. 17.6 *The Network Interface Card (NIC) is the interface between a host and the network to which it attaches*

Although the NIC is shown separately for ease of understanding, it actually fits in one of the slots of the motherboard of the computer, and therefore, is inside the computer. It is very similar to the way a hard disk fits in one of the slots of the computer's motherboard. The NIC appears like a thick card.

The main point is that the NIC is *between* a computer and the network to which it belongs. Thus, it acts as an interface between a computer and its network. When a computer wants to send a message to another computer on the same network, the following process occurs:

1. The computer sends a message to its NIC. The NIC is a small computer, as we have noted. Therefore, it also has its own memory. The NIC stores this message into its memory.
2. The NIC now breaks the message into frames or datagrams, as appropriate to the underlying network protocol (e.g., Ethernet, Token Ring, etc.).

3. The NIC inserts the source and destination *physical* addresses into the frame header. It also computes the CRC and inserts it into the appropriate field of the frame header.
4. The NIC waits until it gets control of the network medium (e.g., in Ethernet, when the bus is free, or in Token Ring, when it grabs the token, etc.). When it gets the control, it sends one or more frames onto the network.
5. Each frame then travels over the network and is received by the respective NICs of all the computers on the network. Each NIC compares its own physical address with that of the destination address contained in the frame. If there is a mismatch, it discards the frame. Thus, only the correct recipient's NIC accepts the frame, as its address matches with the destination address of the frame.
6. The correct recipient receives all the frames in the same fashion.
7. The recipient NIC computes the CRC for each frame, and matches it with that in the frame to check for error, and if acceptable, uses all these frames to reconstruct the original message after discarding the header, etc.
8. The NIC of the recipient computer now hands over the entire message to the recipient computer.
9. The physical address of a computer is predecided by the manufacturer and is always unique. However, the trouble is, the physical addressing scheme differs from one manufacturer to another. To give the appearance of a single and uniform Internet, all computers on the Internet must use a uniform addressing mechanism wherein no two addresses are the same. Since physical networks differ in terms of the address sizes and formats, physical addresses cannot be used. Therefore, IP cannot use the physical address of a computer. It must have a layer on top of the physical address.

17.3 LOGICAL ADDRESSES

To guarantee uniform addressing for all computers on the Internet, the IP software defines an addressing mechanism that does not depend on the underlying physical addresses. This is called **logical addressing**. It is very important to understand the difference between the physical and logical addresses of a computer. The physical address is hard coded on the NIC inside a computer and is thus a hardware entity, whereas a logical address is purely an abstraction and is thus a creation of the software. This abstraction permits computers over the Internet to think only in terms of the logical addresses. Thus, there is no dependence on the physical addresses, and therefore, on the underlying networking mechanisms.

When a computer wants to communicate with another computer on the Internet, it uses the logical address of the destination computer. It is not bothered with the physical address of the destination computer, and therefore, the size and format of the physical address. Thus, even if these two computers have totally different physical addressing mechanisms, they can easily communicate using their logical addresses. This makes communication so simple that sometimes people find it hard to believe that these are not the physical addresses of the computers, and are, in fact, purely logical entities. Of course, internally, we have to find out the physical address from the IP address of the node to actually transmit a frame over a network. This is achieved by a protocol called Address Resolution Protocol (ARP). The example in the next section will clarify this.

17.4 TCP/IP – AN EXAMPLE

17.4.1 Introduction

Let us illustrate the working of TCP/IP with the help of an example. Let us assume that there are three networks, viz., an Ethernet, a X.25 WAN and a Token Ring. Each one has a frame format that is recognized within that network. The sizes and the formats of these frames obviously are different. Figure 17.7 shows these networks connected by routers R1 and R2.

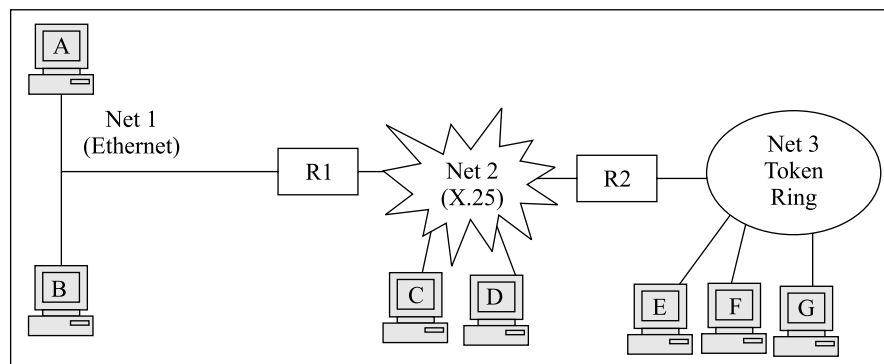


Fig. 17.7 Different networks in the Internet

If computer A of Net 1 wants to send a message (e.g., an email or a file) to computer G of Net 3, how can we achieve this? The networks and their frame formats are different. Ethernet frame, X.25 datagram and Token Ring frame formats are different as shown in Fig. 17.8 (shown not to the proportion).

Preamble (8 bytes)	Destination address (6 bytes)	Source address (6 bytes)	Frame type (6 bytes)	Data (46–1500 bytes) (2 bytes)	CRC (4 bytes)
(a) Ethernet frame format					
General format identifier (4 bytes)	Logical channel number identifier (12 bytes)	Receive number (4 bytes)	Send (4 bytes)	Data (Variable length)	
(b) X.25 datagram format					
Preamble (8 bytes) Field	Access Control Field (8 bytes)	Frame control (48 bytes)	Destination address (48 bytes)	Source address (48 bytes)	Data (Variable length) Other data (56 bytes)
(c) Token Ring frame format					

Fig. 17.8 Different frame formats

As you can see, there is a vast difference between not only the sizes of the three frame/datagram formats, but also the individual field lengths and contents. Now, in which format should the communication take place? This is where IP plays a major role, as we shall see.

17.4.2 IP Datagrams

The structure of the standard format, called an **IP datagram**, is shown in Fig. 17.9.

Version (4 bits)	HLEN (4 bits)	Service type (8 bits)	Total length (16 bits)	
Identification (16 bits)			Flags (3 bits)	Fragmentation offset (13 bits)
Time to live (8 bits)	Protocol (8 bits)		Header checksum (16 bits)	
Source IP address (32 bits)				
Destination IP address (32 bits)				
Data				
Options				

Fig. 17.9  IP datagram

An IP datagram is a variable-length datagram. A message can be broken down into multiple datagrams and a datagram in turn can be fragmented into different fragments, as we shall see. The datagram can contain a maximum of 65,536 bytes. A datagram is made up of two main parts, viz., the **header** and the **data**. The header consists of anywhere between 20 and 60 bytes and essentially contains information about the routing and delivery. The data portion contains the actual data to be sent to the recipient. The header is like an envelope, i.e., it contains information *about* the data. The data is analogous to the letter inside the envelope. Let us examine the fields of a datagram in brief.

1. **Version** – This field currently contains a value 4, which indicates **IP version 4 (IPv4)**. In future, this field would contain 6 when **IP version 6 (IPv6)** becomes the standard.
2. **Header Length (HLEN)** – Indicates the size of the header in a multiple of four-byte words. When the header size is 20 bytes as shown in the figure, the value of this field is 5 (because $5 \times 4 = 20$), and when the option field is at the maximum size, the value of HLEN is 15 (because $15 \times 4 = 60$).
3. **Service type** – This field is used to define service parameters such as the priority of the datagram and the level of reliability desired.
4. **Total length** – This field contains the total length of the IP datagram. Because it is two bytes long, an IP datagram cannot be more than 65,536 bytes ($2^{16} = 65,536$).
5. **Identification** – This field is used in the situations when a datagram is fragmented. As a datagram passes through different networks, it might be fragmented into smaller subdatagrams to match the physical datagram size of the underlying network. In these situations, the subdatagrams are sequenced using the identification field, so that the original datagram can be reconstructed from them.
6. **Flags** – This field corresponds to the earlier field (identification). It indicates whether a datagram can be fragmented in the first place; and if it can be fragmented, whether it is the first or the last fragment, or it can be a middle fragment, etc.

7. **Fragmentation offset** – If a datagram is fragmented, this field is useful. It is a pointer that indicates the offset of the data in the original datagram before fragmentation. This is useful when reconstructing a datagram from its fragments.
8. **Time to live** – We know that a datagram travels through one or more routers before reaching its final destination. In case of network problems, some of the routes to the final destination may not be available due to reasons such as hardware failure, link failure or congestion. In that case, the datagram may be sent through a different route. This can continue for a long time if the network problems are not resolved quickly. Soon, there could be many datagrams traveling in different directions through lengthy paths, trying to reach their destinations. This can create congestion and the routers may become too busy, thus bringing at least parts of the Internet to a virtual halt. In some cases, the datagrams can continue to travel in a loop in between, without reaching the final destination and in fact, coming back to the original sender. To avoid this, the datagram sender initializes this field (that is, *time to live*) to some value. As the datagram travels through routers, this field is decremented each time. If the value in this field becomes zero or negative, it is immediately discarded. No attempt is made to forward it to the next hop. This avoids a datagram traveling for an infinite amount of time through various routers, and therefore, helps avoid network congestion. After all the other datagrams have reached the destination, the TCP protocol operating at the destination will find out this missing datagram and will have to request for its retransmission. Thus, IP is not responsible for the error-free, timely and in-sequence delivery of the entire message; it is done by the TCP.
9. **Protocol** – This field identifies the transport protocol running on top of IP. After the datagram is constructed from its fragments, it has to be passed on to the upper layer software piece. This could be TCP or UDP. This field specifies which piece of software at the destination node the datagram should be passed on to.
10. **Source address** – This field contains the 32-bit IP address of the sender.
11. **Destination address** – This field contains the 32-bit IP address of the final destination.
12. **Options** – This field contains optional information such as routing details, timing, management, and alignment. For instance, it can store the information about the exact route that the datagram has taken. When it passes through a router, the router puts in its id, and optionally, also the time when it passed through that router, in one of the slots in this field. This helps tracing and fault detection of datagrams. However, most of the time, this space in this field is not sufficient for all these details, and therefore, it is not used very often.

All the computers on the Internet have to run the TCP/IP software stack and therefore, have to understand the IP datagram format. Also, all the routers need to have the IP software running on them. Hence, they have to understand the IP datagram format.

Each computer in the whole Internet has a 32-bit unique **IP address**, which consists of two parts, viz., **network id** and **host id** within that network. It is the network id portion of the IP address of the destination, which is used for routing the datagrams to the destination network through many other routers. To enable this, a routing table is used, which gives the address of the next hop for a specific destination. Each router uses a separate routing table. Once the datagram reaches the destination network, the *host id* portion of the destination IP address is used to reach the datagram to the destination computer on that network. Also, as we have noted, each computer attached to the network has to have a Network Interface Card (NIC) which understands and transmits frames corresponding to that network.

17.4.3 More on IP

As we have mentioned before, the TCP/IP protocol suite uses the IP protocol for transmission of datagrams between routers. IP has two important properties, viz., it is **unreliable** and it is **connectionless**. Let us understand what it means before we discuss our example in detail.

IP is Unreliable

When we say that IP is unreliable, it means that IP does not provide a guarantee that a datagram sent from a source computer definitely will arrive at the destination. In this sense, it is called a **best-effort delivery** mechanism.

For understanding this, consider what happens when you send a letter to somebody through the postal service. You write the letter, put it in a stamped envelope and drop it in a post box. The letter then travels through one or more post offices, depending on the destination (e.g., if it is in the same or different city, state and country) and finally reaches the destination. However, the postal service does not guarantee a delivery. All it says is that we will try our best to deliver this letter as soon as we can, to the recipient. The job of each post office is simple – it forwards the letter to the next appropriate destination, which may or may not be the final destination. For instance, if you send a letter from Boston to Houston, the Boston city post office will forward your letter to the Houston city post office along with all other letters destined for Houston. The Houston post office would forward the letter to the concerned area post office within Houston. This will happen as the letter passes through a number of intermediate post offices, which essentially act as exchanges or routers. Finally, the letter would be forwarded to the ultimate destination. However, at no point of time is any post office checking back to see if the letter is properly received by the next recipient. They all assume in good faith that it is happening that way.

IP works on a very similar principle. In internetworking terms, IP does not have any error-checking or tracking mechanisms. Therefore, IP's job is restricted to forwarding a datagram from its initial source onto the final destination through a series of routers. However, it does not in any way check to see if the datagram reached its next destination. IP assumes that the underlying medium is reliable and makes the best possible effort of delivery. There could be several reasons behind datagrams that are lost, such as bit errors during transmission, a congested router, disabled links, and so on. IP lets reliability become the responsibility of the transport layer (i.e., TCP), as we shall see later.

Just as the postal system has a concept of registered letters wherein each post office keeps a track of letters received from the various sources and heading for various destinations, the Internet has this capability in the form of TCP. TCP makes every datagram work like a registered letter to keep its track and ensure error-free delivery. We shall discuss TCP in detail later.

IP is Connectionless

Consider what happens when you make a phone call to someone. After you dial the number and the two of you start speaking, you are the only users of the communication channel. That means, even if none of you speak for some duration, the communication channel remains unused; it is not allocated to another telephone conversation between two other persons. This concept of *switched circuits* makes telephone communications connection-oriented. However, IP, like other networking protocols, does not assume any connection between the sender and the receiver of datagrams. Instead, each datagram sent by IP is considered to be an independent datagram and it does not

require any connection to be set up between the sender and the receiver before datagrams can be transmitted. Any computer can send datagrams to any other computer any time, with no regard to other computers on the Internet.

Clearly, if this has to work, a datagram must contain information about the sender and the receiver. Otherwise, the routers would not know where to forward the datagrams. Thus, a datagram has to contain not only data, but also additional information such as where the datagram originated from and where it is heading. Using the analogy from the postal department, a datagram is therefore called a *datagram*. Whereas in *virtual circuit* philosophy, a circuit, i.e., the entire path from the source to the destination is established before all datagrams in a message are routed, and which remains fixed throughout, this is not the case with datagrams, where for each datagram, the route is decided at the routing time. We have discussed this before in datagram switching. Therefore, IP is a *packet switching* technology.

17.4.4 Communication using TCP/IP

Now, the actual communication takes place as discussed below. We shall assume the network as shown in Fig. 17.6 for this example. We still assume that a user on node A wants to send a message (e.g., an email) to a user on node G. The following steps will be executed to accomplish this task.

1. The application layer running on node A (e.g., an email program) hands over the message to be transmitted to the TCP/IP layer running on node A.
2. The TCP layer breaks the message into smaller packets, and appends the TCP header to each one, as shown in Fig. 17.10. We will talk about this later when we discuss TCP in detail.

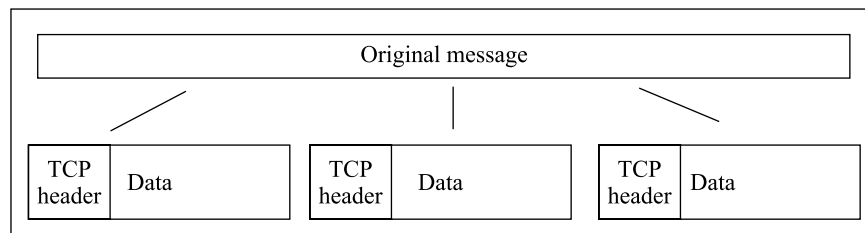


Fig. 17.10 TCP layer breaks down the original message into packets

3. The IP layer breaks each packet further into fragments if necessary, and appends the IP header to each one. We will assume for simplicity that the fragmentation is not necessary. At the IP level, the TCP header + datagram are treated together as data. This is the basic process of encapsulation. Thus, in this case, it will just append the IP header to the datagram for IP (i.e., the original datagram plus TCP header). This becomes the datagram for the lower layer (i.e., Ethernet). The datagram now looks as shown in Fig. 17.11.

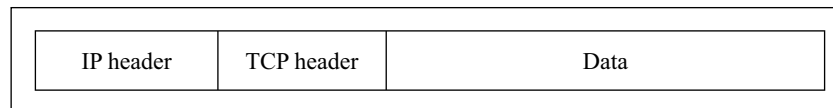


Fig. 17.11 IP header is added to the TCP datagram

The IP header contains the 32 bit source as well as destination addresses corresponding respectively to the source and destination computers.

4. Now, the whole IP datagram shown earlier is treated as *data* as far as Net 1 (which is an Ethernet) is concerned. The IP software at node A realizes from the network id portion of the destination IP address (contained in IP header) that it is not the same as the network id of the IP address of computer A. Therefore, it realizes that the destination computer is on a different network; and it simply has to hand the datagram over to router R1. As a general rule, every node has to hand over all datagrams that have a different network id than their own, in the destination address field, to a router, based on the routing algorithm. In this case, based on this algorithm, node A decides that it has to hand over the datagram to R1. Note that both A and R1 are on the same Ethernet network, and Ethernet only understands frames of certain format Here is where encapsulation comes handy.
5. The datagram now reaches the NIC of node A. Here, an Ethernet frame is formed as shown in Fig. 17.12. The data portion in the frame is the IP header + TCP header + datagram. The Ethernet header contains the 48-bit source and destination addresses. As we know, these are physical addresses given by the manufacturer to the NICs of the source (i.e., node A) and destination (i.e., router R1) computers. The problem is how do we get these physical addresses?

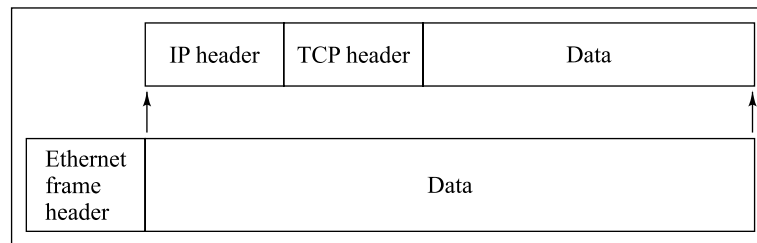


Fig. 17.12 Ethernet frame

6. The NIC at node A has to move 48 bit physical source and destination addresses into the Ethernet frame as well as compute the CRC. The source address is that of NIC of A itself, which the node A knows. The problem is to get the same for the destination, which is the NIC of router R1. To get this, an **Address Resolution Protocol (ARP)** is used. The Ethernet frame now is ready to be transmitted.
7. Now, the usual CSMA/CD protocol is used. The bus is continuously monitored. When it is idle, the frame is sent. If a collision is encountered, it is resent after a random wait and so on. Finally, the frame is on the bus. While it travels through various nodes, each node's NIC reads the frame, compares the destination address in the frame header with its own, and if it does not match, discards it.
8. Finally, when it matches with that of the router R1, it is stored in the memory of the router's NIC. The NIC of the router verifies the CRC and accepts the frame.
9. The NIC of the router removes the Ethernet frame header and obtains back the original IP datagram as shown in Fig. 17.13, and passes it on to the router.

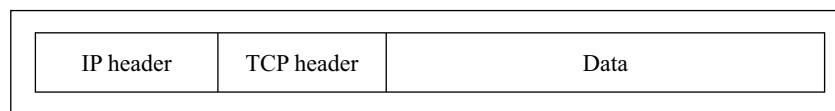


Fig. 17.13 Original IP datagram

10. Now, the router checks its routing table to learn that if the final destination is G, the next hop for this datagram is router R2. To use the routing table, the destination 32-bit IP address has to be extracted from the IP header (refer to Fig. 17.8), because the routing tables contain the destination IP address and the *next hop* to reach there. In this case, the *next hop* is router R2. The router R1 knows that there is a X.25 WAN connecting it and R2. In fact, R1 and R2 both are on this X.25 WAN (though R1 is also on the Ethernet LAN).
11. At this stage, the IP header + TCP header + data, all put together, is treated as *data* again for X.25, and a header for X.25 is generated and a datagram in the X.25 format is prepared. The datagram header contains the source and destination addresses, which are those of R1 and R2, respectively as our intention is to transport the datagram from R1 to R2, both on the X.25 network, wherein the datagram also now is in the X.25 format. Router R1 then releases the datagram on to the X.25 network using ARP as before. It looks as shown in Fig. 17.14.

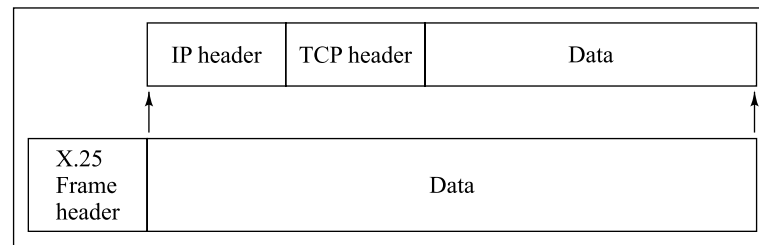


Fig. 17.14 X.25 frame

12. The X.25 network has its own ways of acknowledgement between adjacent nodes within the X.25 network, etc. The datagram travels through various nodes and ultimately reaches the R2 router.
13. R2 strips the X.25 datagram of its header to get the original IP datagram as shown in Fig. 17.15. R2 now extracts the source and destination 32-bit IP addresses from the IP header.

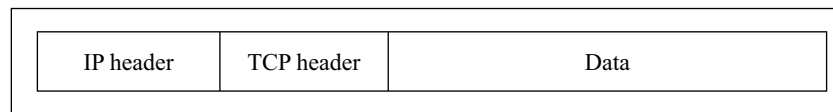


Fig. 17.15 Original IP datagram

14. R2 is also connected to the Token Ring network. Now, R2 compares the network id portion of the 32-bit destination IP address in the above datagram with its own and realizes that both are the same. As a consequence, it comes to know that computer G is local to it.
15. R2 now constructs a Token Ring frame out of this IP datagram by adding the Token Ring header, as shown in Fig. 17.16. The format shows 48-bit physical source and destination addresses. These are now inserted for those of R2 and G respectively; frame control bits are computed and the Token Ring frame is now ready to go. For this, again ARP is used.
16. R2 now directly delivers the Token Ring frame to computer G. G discards the Token Ring frame header to get the original IP datagram.
17. In this fashion, all the datagrams sent by computer A would reach computer G. The IP header is now removed (it has already served its purpose of transporting the IP packet.) And the datagrams only with TCP header are handed over to the TCP layer at node G. They may reach

out of sequence or even erroneously. The TCP software running in computer G is responsible for checking all this, acknowledging the correct receipt or requesting for retransmission. The intermediate routers do not check this. They are only responsible for routing. Therefore, we say that *IP is connectionless* but *TCP is connection-oriented*.

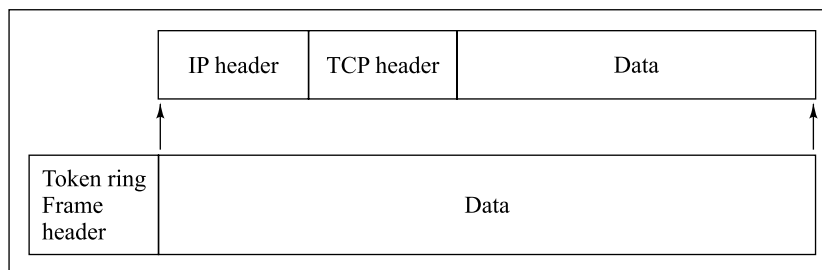


Fig. 17.16 Token Ring frame

18. Ultimately, all the datagrams are put in sequence. The original message is thus, reconstructed.
19. The message is now passed on to the appropriate application program such as email running on computer G. The TCP header contains the fields source and destination **socket numbers**. These socket numbers correspond to different application programs, of which many of them are standardized, which means that given a socket number, its corresponding application program is known. These socket numbers are used to deliver the datagrams to the correct application program on computer G. This is important because G could be a multiprogramming computer that is currently executing more than one application. So, it is essential that the correct application program receive these datagrams.
20. The program on computer G can process the message, e.g., it can inform the user of that computer that an email message has arrived. Thus, the user on computer G can then read the email message.

We will notice the beauty of TCP/IP. Nowhere are we doing exact frame format or protocol conversion. TCP/IP works on all the nodes and routers, but Ethernet works as it did before. It carries the frame in its usual fashion and follows the usual CSMA/CD protocol. The same is true about X.25 and Token Ring. The point is TCP/IP *fools* all these networks and still carries a message (email, file, Web page, etc.) from any node to any other node or any other network in the world; therein lays the beauty of TCP/IP.

17.5 THE CONCEPT OF IP ADDRESS AND IP DATAGRAM/PACKET

17.5.1 Introduction

On the Internet, the IP protocol defines the addressing mechanism to which all participating computers must conform. This standard specifies that each computer on the Internet be assigned a unique 32-bit number called **Internet Protocol address**, or in short, **IP address**. Each datagram traveling across the Internet contains the 32-bit address of the sender (source) as well as that of the recipient (destination). Therefore, to send any information, the sender must know the IP address of the recipient.

The IP address consists of three parts, viz., a **class**, a **prefix** (or **network number**) and a **suffix** (or **host number**). This is done to make routing efficient, as we shall see. We shall discuss the various types of classes in the next section. The prefix denotes the physical network to which the computer is attached. The suffix identifies an individual computer on that network. For this, each physical network on the Internet is assigned a unique **network number**; this is the prefix portion of the IP address. Since no two network numbers across the Internet can be the same, assigning network numbers must be done at the global level, to ensure that there is no clash with another network number. The **host number** denotes the suffix, as shown in Fig. 17.17. *Host* is just another name for a computer on the Internet. The assignment of host numbers can be done locally.

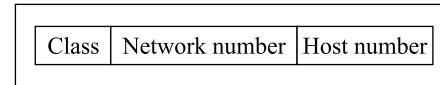


Fig. 17.17 Three parts of IP address

The network number is analogous to a street name and the host number is similar to a house number. For example, only one house can have number 56 on James Street. There is no confusion if we have a house number 56 on George Street in the neighborhood. Thus, across streets, the same house number can be repeated. But in a given street, it must be unique.

In this case, within a network (identified by network number), no two hosts can have the same host number. However, if their network numbers differ, the host numbers can be the same. For example, suppose the Internet consists of only three networks, each consisting of three computers each. Then, we can number the networks as 1, 2 and 3. Within each network, the hosts could be numbered in the same way, i.e., 1, 2 and 3. Therefore, logically, the address of the second computer on the third network would be 3.2, where 3 is the network number and 2 is the host number. The IP addressing mechanism works in a very similar fashion. Note that conceptually, this is very similar to the two-part WAN addressing that we had discussed earlier.

To summarize, IP addressing scheme ensures the following:

1. Each computer is assigned a unique address on the Internet.
2. Although network addressing is done globally (world-wide), host addressing (suffixing) can be carried out locally without needing to consult the global coordinators.

17.5.2 Who Decides the IP Addresses?

To ensure that no two IP addresses are ever the same, there has to be a central authority that issues the prefix or network number portion of the IP addresses. Suppose an organization wants to connect its network to the Internet. It has to approach one of the local Internet Service Providers (ISPs) for obtaining a unique IP address prefix. At the global level, the **Internet Assigned Number Authority (IANA)** allocates an IP address prefix to the ISP, which in turn, allocates the host numbers or suffixes to each different customer one by one. Thus, it is made sure that IP addresses are never duplicated.

Conceptually, we can consider that IANA is a *wholesaler* and an ISP is a *retailer* of IP addresses. The ISPs (retailers) purchase IP addresses from the IANA (*wholesaler*), and sell them to the individual customers.

17.5.3 Classes of IP Addresses

As we know, the designers had decided the following two things:

1. Use 32 bits for representing an IP address.
2. Divide an IP address into three parts, viz., a class, a prefix and a suffix.

The next question was to determine how many bits to reserve for the prefix (i.e., the network number) and how many for the suffix (i.e., the host number). Allocating more bits to one of suffix and prefix would have meant lesser bits to the other. If we allocate a large portion of the IP address to the network number, more networks could be addressed and therefore, accommodated on the Internet, but then the number of bits allocated for the host number would have been less, thus reducing the number of hosts that can be attached to each network. On the other hand, if the host number were allocated more bits, a large number of computers on less number of physical networks each would be allowed.

Since the Internet consists of many networks, some of which contain more hosts than the others, rather than favoring either of the schemes, the designers took a more prudent approach of making everybody happy, as shown in Fig. 17.18, and described thereafter.

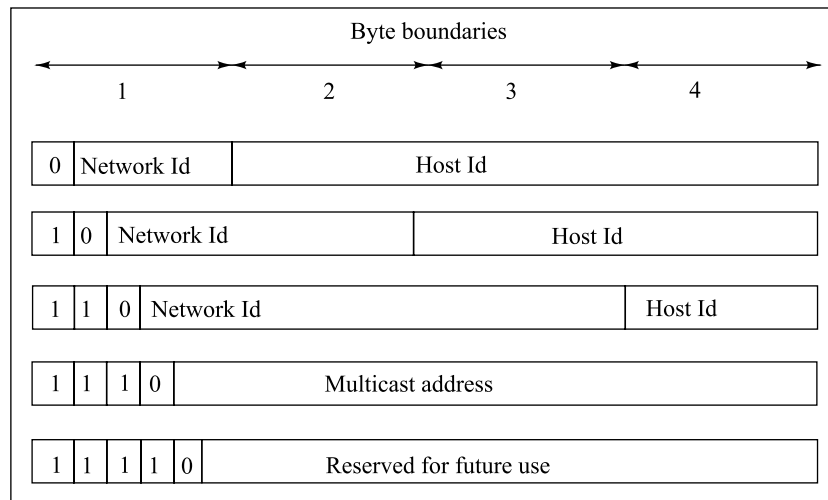


Fig. 17.18 *Classes of IP address*

As shown in the figure, the concept of a class was introduced. The IP address space is divided into three **primary classes** named A, B and C. In each of these classes, a different number of bits are reserved for the network number and the host number portions of an IP address. For example, class A allows 7 bits for the network number and 24 bits for the host number; thus allowing fewer networks to have a large number of hosts each. Class B is somewhere in between. On the other hand, class C reserves 21 bits for the network number and just 8 bits for the host number; thus useful for a large number of networks that have smaller number of hosts. This makes sure that a network having a large number of hosts can be accommodated in the IP addressing scheme with the same efficiency as a network that has very few hosts attached to it.

In addition to the three primary classes, there are two more classes named D and E, which serve special purpose. Class D is used for **multicasting**, which is used when a single message is to be sent to a group of computers. For this to be possible, a group of computers must share the common **multicast address**. Class E is not used as of now. It is reserved for future use.

How do we determine, given an IP address, which class it belongs to? The initial few bits in the IP address indicate this. If the first bit in the IP address is a 0, it must be an address belonging to class A. Similarly, if the first two bits are 10, it must be a class B address. If the first three bits are 110, it belongs to class C. Finally, if the first four bits are 1110, the IP address belongs to class D. When class E would come in use, the first five bits would indicate this fact by having a value of 11110. Excepting these bits reserved for the class, the remaining bits contain the IP address itself. These bits are divided into network number and host number using the philosophy described earlier.

How is this mechanism useful in practice? Simply put, there are two possibilities in case of data transmission in the form of IP datagrams from a source to a destination.

1. The source and the destination are on the same physical network. In this case, we call the flow of packets from the source to the destination as **direct delivery**. To determine this, the source can extract the *network number* portion of the destination host and compare it with the *network number* portion of itself. If the two match, the source knows that the destination host is on the same physical network. Therefore, it need not go to its router for delivery, and instead, can use the local network mechanism to deliver the packet as shown in Fig. 17.19.

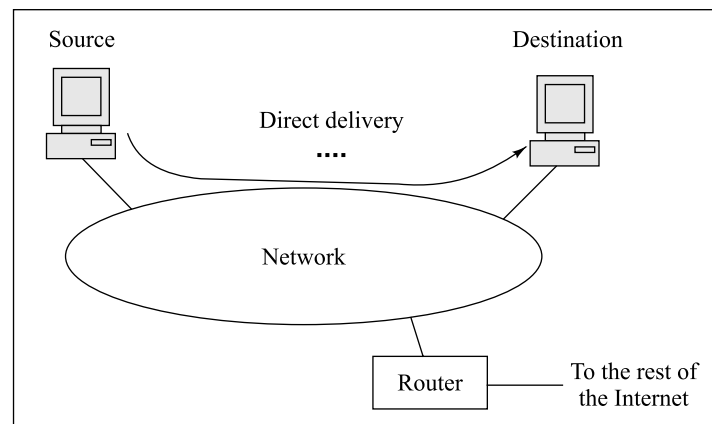


Fig 17.19 Direct delivery

2. The source and the destination are on different physical networks. In this case, we call the flow of packets from the source to the destination as **indirect delivery**. To determine this, the source can extract the *network number* portion of the destination host and compare it with the *network number* portion of itself. Since the two do not match, the source knows that the destination host is on a different physical network. Therefore, it needs to forward the packet to the router, which, in turn, forwards the packet to the destination (possibly via more routers, a possibility, which we shall ignore here). This is shown in Fig. 17.20.

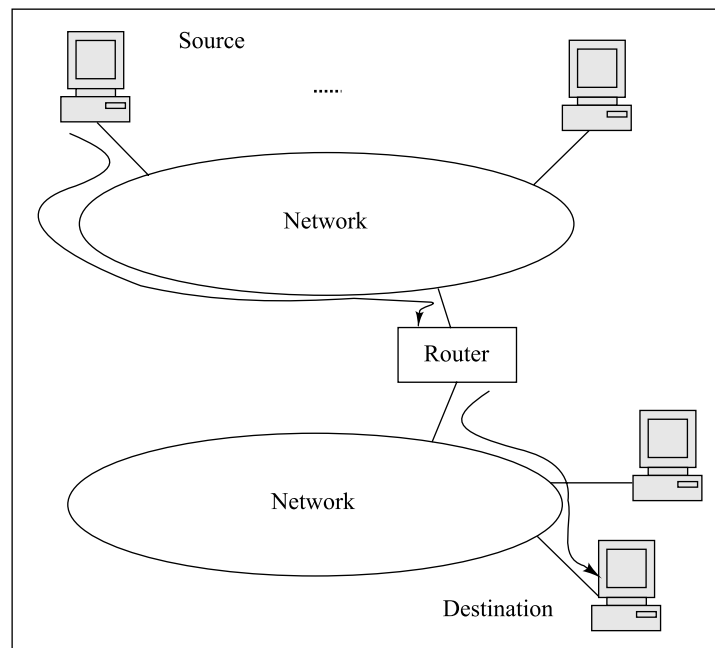


Fig. 17.20 Indirect delivery

Based on the above discussion, let us find out how many networks and hosts each class can serve, depending on the number of bits allocated for the network number and the host number. This is shown in Fig. 17.21.

Class	Prefix size (in bits)	Maximum number of networks	Suffix size (in bits)	Maximum number of hosts
A	7	128	24	1,67,77,216
B	14	16,384	16	65,536
C	21	20,97,152	8	256

Fig. 17.21 Maximum number of networks and hosts in each class

17.5.4 Dotted Decimal Notation

It is very difficult to talk about IP addresses as 32-bit binary numbers in regular communication. For example, an IP address could be

10000000 10000000 11111111 00000000

Clearly, it is just not possible to remember such addresses for us humans. Computers would, of course, work happily with this scheme. Therefore, the **dotted decimal notation** is used for our convenience. In simple terms, the 32 bits are divided into four **octets**. Octets are the same as bytes. Each octet, as the name suggests, contains eight bits. Each octet is then translated into its equivalent decimal value and all the four octets are written one after the other, separated by dots. Thus, the above address becomes

117.117.255.0

This can be shown diagrammatically in Fig. 17.22.

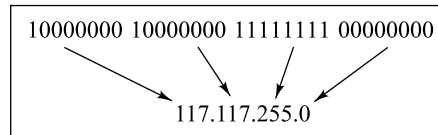


Fig. 17.22 *Equivalence between binary and dotted decimal notation*

Note that the lowest value that an octet can take is 0 in decimal (00000000 in binary) and the highest is decimal 255 (11111111 in binary). Consequently, IP addresses, when written in dotted decimal notation, range from 0.0.0.0 to 255.255.255.255. Thus, any valid IP address must fall in this range.

Another way to represent the classification is as shown in Fig. 17.23.

	Minimum value	Maximum value
Class A	<div>0.0.0.0</div> <div>Net id Host id</div>	<div>127.255.255.255</div> <div>Net id Host id</div>
Class B	<div>128.0.0.0</div> <div>Net id Host id</div>	<div>191.255.255.255</div> <div>Net id Host id</div>
Class C	<div>192.0.0.0</div> <div>Net id Host id</div>	<div>223.255.255.255</div> <div>Net id Host id</div>
Class D	<div>224.0.0.0</div> <div>Multicast address</div>	<div>239.255.255.255</div> <div>Multicast address</div>
Class E	<div>240.0.0.0</div> <div>Reserved</div>	<div>255.255.255.255</div> <div>Reserved</div>

Fig. 17.23 *IP address ranges in decimal notation system*

It should be mentioned that class A and B are already full. No new network can be assigned an address belonging to either of these categories. IP addresses belonging only to class C are still available. Therefore, whenever a company sets up a new LAN, which it wants to get connected

to the Internet, it is normally assigned a class C address. We shall discuss this limitation when we discuss **IPv6**.

17.5.5 Routers and IP Addresses

We have seen that a router is a special purpose computer that is mainly used for forwarding datagrams between computer networks over the Internet. This means that a router connects two or more networks, as we had discussed. As a result, a router needs to have as many IP addresses as the number of networks that it connects, usually at least two. Figure 17.24 shows an example. Here, router R1 connects two networks: an Ethernet (whose IP network number is 130.100.17.0, i.e., the full IP address of any host on that network would be 130.100.17.*, where * is the host number between 0 and 255) and a Token Ring (whose IP network number is 231.200.117.0, i.e., the full IP address of any host on that network would be 231.200.117.*, where * is the host number between 0 and 255). We will note that both of these are class C networks. Similarly, router R2 connects the same Token Ring network to a WAN whose IP address prefix is 87.12.0.0, which is a class A network. Note that the routers are assigned IP addresses for both the interfaces that they connect to. Thus, as shown in Fig. 17.24, router R1 has an IP address 130.100.17.7 on the Ethernet LAN, whereas it has an IP address of 231.200.117.19 on the Token Ring network. The same thing can be observed in R2. In general, if a router connects to n networks, it will have n IP addresses.

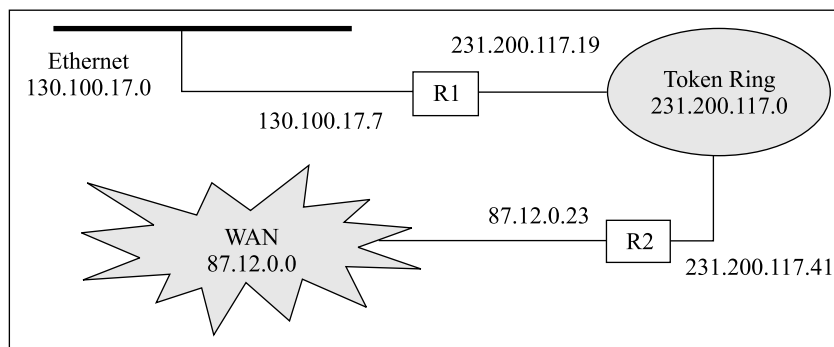


Fig. 17.24 A router has two or more IP addresses

17.5.6 Subnetting and Subnet Addressing

In order to make better utilization of the available IP address space, a technique called **subnetting** is used many times. Subnetting divides the main network into smaller **subnetworks (subnets)**, with each subnet getting its own IP addressing space. In other words, we can allow a network to be internally divided into multiple smaller networks, while giving the view of a single homogeneous network to the outside world.

Normally, an IP address is a two-level hierarchy, consisting of the Network id followed by the Host id. However, when we make use of subnets, the IP address has a three-level hierarchy, consisting of the Network id, Subnet id, and Host id. In other words, the main network is made up of several subnets, and each subnet, in turn, has several hosts attached to it. Every subnet functions like a full-fledged network, while hiding the internal complexities from the external world. This provides several advantages as given below:

1. Because an organization is likely to be physically divided into several groups, creating subnets to represent them makes sense. In other words, subnets give a better picture of the internal structure of the organization's computer network.
2. We can make better utilization of the available IP address space.
3. Because subnets are hidden from the external world, this does not add to any complexities from the point of view of the external world.
4. The ability to split an existing network into smaller networks reduces the demand for more and more IP addresses.
5. The routing tables work without any changes.

Of course, when subnetting is in operation, there are some changes to the routing process for the organization that has implemented subnetting. Normally, when a router receives a datagram, if it is indirect delivery, the router checks the network id in the destination address field of the datagram, and forwards the datagram appropriately. However, if this is direct delivery, then the router sends the datagram to the host matching the destination address field in the datagram. However, now the IP address consists of 3 parts: Network id, Subnet id, and Host id. As a result, the router must know how many bits are reserved for the Subnet id and how many remaining bits are allocated to the Host id.

To understand this better, let us take an example. Suppose that we have a class B network with IP address of 153.72.0.0. Now, we know that the first 16 bits are reserved for the network id part (153.72), and the remaining 16 bits can be used for the host id part. If we were to create subnetworks inside this network, we must decide how to divide the 16 bits reserved for the host id into subnet id and host id. This decision will depend on how many subnets we want to create. For example,

1. If we use 1 bit for the subnet id, then $2^1 = 2$ subnets would be possible. This would leave 15 bits for the host id part.
2. If we use 2 bits for the subnet id, then $2^2 = 4$ subnets would be possible. This would leave 14 bits for the host id part.
3. If we use 3 bits for the subnet id, then $2^3 = 8$ subnets would be possible. This would leave 13 bits for the host id part.

We can extend this concept further, but beyond 3 or 4 bits for the subnet id, things start becoming a bit meaningless in most situations. As we can see, we must strike a balance between allocating more bits to the subnet id and host id.

An example is shown in Fig. 17.25.

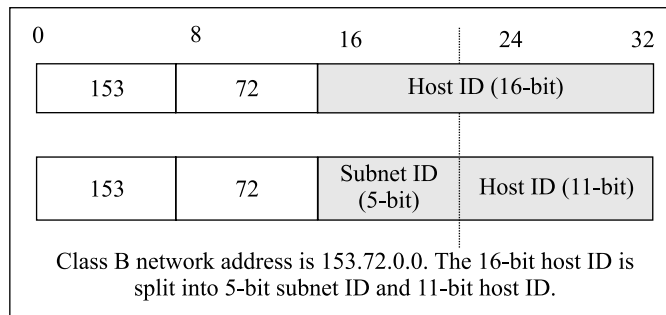



Fig. 17.25  *Subnetting example*

As we can see in Example, 5 of the 16 host address bits are reserved for the subnet id. As a result, only 11 bits are available for the host id.

Subnet addressing is based on the principles of first identifying the base address of each subnet. Then host addresses can be found out easily. For example, consider a network with IP address 211.77.20.0. Suppose we want to create 6 subnets in this network. How would the subnet addressing mechanism work?

First, let us express 211.77.20.0 in binary notation. Hence, we will have
11010011.01001101.00010100.00000000.

Because we are going to create 6 (i.e., 110 in binary) subnets, we must reserve 3 bits of the host id part for the subnet id. Hence, we can rewrite our binary IP address with the subnet id part underlined for clarity: 11010011.01001101.00010100.00000000. The underlined 3 bits denote the subnet id, and remaining 5 bits denote the host id. Therefore, we will have the actual IP addresses divided into various subnet blocks as follows:

- Subnet 1:** First host: 11010011.01001101.00010100.00000000 (i.e., 211.77.20.0),
Second host: 11010011.01001101.00010100.00000001 (i.e., 211.77.20.1),
Third host: 11010011.01001101.00010100.00000010 (i.e., 211.77.20.2),
Fourth host: 11010011.01001101.00010100.00000011 (i.e., 211.77.20.3),
Fifth host: 11010011.01001101.00010100.00000100, (i.e., 211.77.20.4), and so on.
- Subnet 2:** First host: 11010011.01001101.00010100.00100000 (i.e., 211.77.20.32),
Second host: 11010011.01001101.00010100.00100001 (i.e., 211.77.20.33),
Third host: 11010011.01001101.00010100.00100010 (i.e., 211.77.20.34),
Fourth host: 11010011.01001101.00010100.00100011 (i.e., 211.77.20.35),
Fifth host: 11010011.01001101.00010100.00100100 (i.e., 211.77.20.36), and so on.
- Subnet 3:** First host: 11010011.01001101.00010100.01000000 (i.e., 211.77.20.64),
Second host: 11010011.01001101.00010100.01000001 (i.e., 211.77.20.65),
Third host: 11010011.01001101.00010100.01000010 (i.e., 211.77.20.66),
Fourth host: 11010011.01001101.00010100.01000011 (i.e., 211.77.20.67),
Fifth host: 11010011.01001101.00010100.01000100 (i.e., 211.77.20.68), and so on.
- Subnet 6:** First host: 11010011.01001101.00010100.11000000 (i.e., 211.77.20.192),
Second host: 11010011.01001101.00010100.11000001 (i.e., 211.77.20.193),
Third host: 11010011.01001101.00010100.11000010 (i.e., 211.77.20.194),
Fourth host: 11010011.01001101.00010100.11000011 (i.e., 211.77.20.195),
Fifth host: 11010011.01001101.00010100.11000100, (i.e., 211.77.20.196), and so on.

As we can see, because we have reserved 5 bits per subnet for the host id part, each subnet can contain at the most 2^5 i.e., 32 hosts. This is also clear from the addressing details above.

17.5.7 IP Version 6 (IPv6)

As we have discussed, IP has been extremely successful in making the Internet a worldwide network that hides the complexities involved in its underlying networks, hardware changes and increases in scale, amazingly well. However, there is one major problem with the original design of IP. Like those involved in almost every other invention, the designers of IP failed to realize its immense future capabilities and popularity. They decided to use only 32 bits for the IP address.

At that time, it seemed to be a large number. However, due to the mind-boggling growth of the Internet over the last few years, the range of IP addresses is appearing to be too less. IP addresses are exhausting too fast, and soon there would simply be not enough IP addresses. We need more addressing space. Secondly, the Internet is being used for applications that few would have dreamt of even a few years ago. Real-time audio-video and collaborative technologies (analogous to virtual meeting among people, over the Internet) are becoming very popular. The current IP version 4 (IPv4) does not deal with these well because it does not know how to handle the complex addressing and routing mechanisms required for such applications.

The new practical version of IP, called **IP version 6 (IPv6)**, also known as **IP Next Generation (IPng)** deals with these issues. It retains all the good features of IPv4 and adds newer ones. Most importantly, it uses a 128-bits IP address. It is assumed (and more importantly, *hoped*) that IP address of this size would be sufficient for many more decades, until the time people realize that IPv6 is actually too small. Apart from this, IPv6 also has support for audio and video. It is expected that IPv4 would be phased out in the next few years and IPv6 would take over from it.

17.6 ADDRESS RESOLUTION PROTOCOL (ARP)

In the last chapter, we have seen the importance of IP addressing. In simple terms, it makes addressing on the Internet uniform. However, having only an IP address of a node is not good enough. There must be a process for obtaining the physical address of a computer based on its IP address, in order to be able to finally actually transmit the frame/datagram over the network, to which the node belongs. This process is called **address resolution**. This is required because at the hardware level, computers identify each other based on the physical addresses hard-coded on their Network Interface Cards (NICs). They neither know the relationship between the IP address prefix and the physical network, nor the relationship between an IP address suffix and a particular computer.

Networking hardware demands that a datagram should contain the physical address of the intended recipient. It has no clue about the IP addresses. To solve this problem, the **Address Resolution Protocol (ARP)** was developed. ARP takes the IP address of a host as input and gives its corresponding physical address as the output. This is shown in Fig. 17.26.

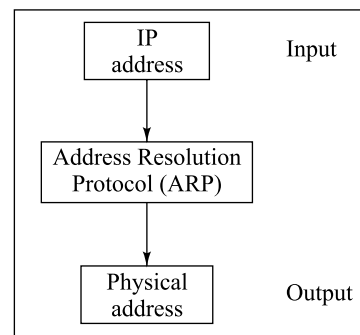


Fig. 17.26 Address Resolution Protocol (ARP)

There are three methods for obtaining the physical address based on an IP address. These three methods are as follows.

1. **Table lookup** – Here, the mapping information between IP addresses and their corresponding physical addresses is stored in a table in the computer's memory. The network software uses this table when it needs to find out a physical address, based on the IP address.
2. **Closed-form computation** – Using carefully computed algorithms, the IP addresses can be transformed into their corresponding physical addresses by performing some fundamental arithmetic and Boolean operations on them.

3. **Message exchange** – In this case, a message is broadcasted to all the computers on the network in the form *Are you the one whose IP address is X? If so, please send me your physical address*. The computer whose IP address matches X (the broadcasted IP address), sends a reply, and along with it, its physical address to the broadcasting computer. All other computers ignore the broadcast. This method is the simplest one and hence most popular, and we shall discuss it in detail.

17.6.1 ARP using Message Exchange

How message exchange works is simple. We assume that every host knows its IP address as well as physical address. It also knows the IP address of the destination where it wants to send a datagram. However, while sending a frame on a specific network such as Ethernet, the physical addresses of both the source and the destination have to be specified in the frame. Therefore, the sender node has to know the physical address of the destination. All it knows is its IP address.

Anytime a host or a router needs to find the physical address of another host on its network, it creates a special datagram called **ARP query datagram** that includes the IP address of the destination whose physical address is needed. This special datagram is then broadcasted over the network as shown in Fig. 17.27.

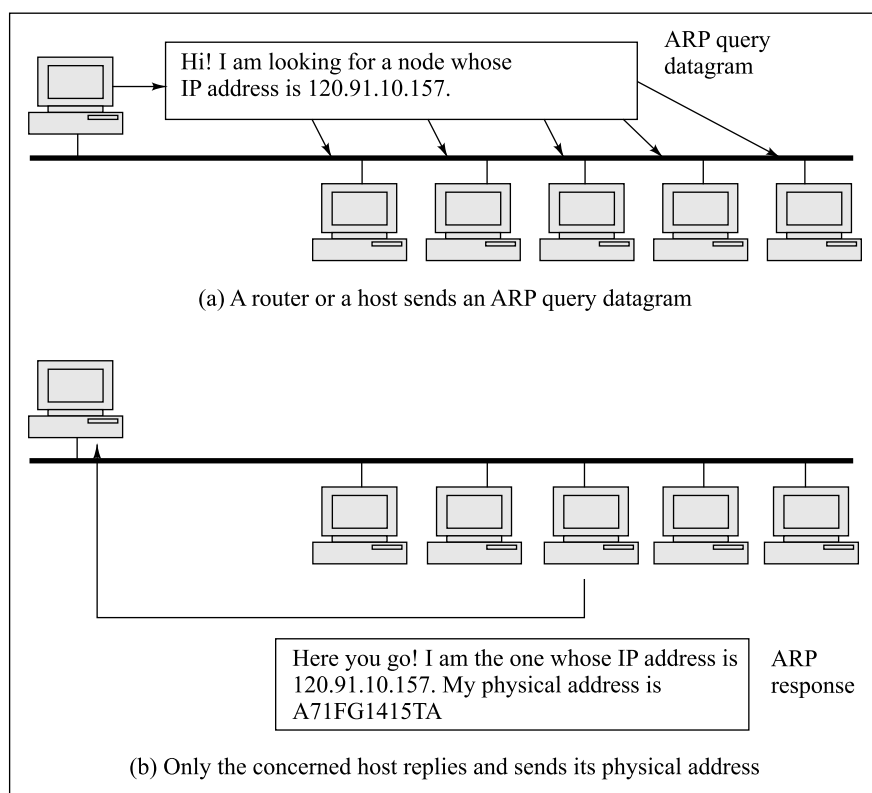


Fig. 17.27 Example of ARP

As shown in the figure every host on the network receives this datagram even if the destination physical address is absent. This is because this is a broadcast request, which means that the datagram should go to all the hosts in the network. Every host then checks the IP address of the destination mentioned in the ARP query datagram with its own. If it does not match, it discards it. However, if it matches, it sends back its physical address to the original node. Thus, only one of the hosts on the network would respond back to the ARP query datagram, as shown in the figure.

This whole process, datagram/response formats, etc., together constitute the Address Resolution Protocol (ARP).

17.6.2 ARP Caching

There is one interesting phenomenon associated with the ARP. It learns from past experiences. Once it knows the relationship between a particular IP address and a physical address, it stores this information in a table. This is called **ARP caching**. Thus, next time if a host wants to send a datagram to the same destination, it need not again broadcast the ARP query datagram to find the physical address associated with it. Instead, it first checks it in its ARP cache. Only if the IP address (and therefore, the physical address) of the destination host is not found in the table, it broadcasts the ARP query datagram.

17.6.3 ARP is Local

It is very important to understand that the address resolution performed by ARP is always local to a network. A computer can resolve another computer's address only if both the computers are on the same physical network. In other words, a computer cannot resolve the address of another computer if it is on a different physical network. The reasons for this should be obvious. Since physical networks are of different kinds and use different addressing mechanisms, a computer must not attempt to resolve another computer's address if they are on different physical networks. Let us understand this with an example as shown in Fig. 17.28.

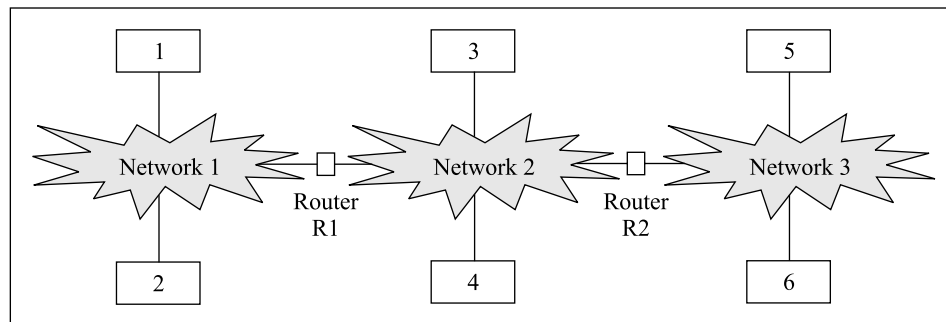


Fig. 17.28 *Address resolution is always local*

The figure shows computers 1 and 2 attached to the same network. Similarly, computers 3 and 4 are attached to another network. A third network has computers 5 and 6 attached to it. The networks are then connected together by routers R1 and R2. Suppose computer 2 wants to send a datagram to computer 5. This would work as follows:

1. Computer 2 realizes that it cannot resolve the address of computer 5. Instead, it knows that it has to forward the datagram to router R1. Therefore, it resolves R1's address using ARP, and sends the datagram to R1. Remember that R1 has two addresses corresponding to two networks that it connects to. This address resolution process would have given the physical address of the router on the same network (i.e., network 1).
2. The protocol software on router R1, in turn, realizes that it cannot reach computer 5, and therefore, cannot resolve its address. Therefore, it forwards the datagram to router R2. To achieve this, it uses the second address of itself (R1), i.e., the one on network 2. It now resolves the address of R2 using ARP. This is R2's address on network 2. With this address, it now forwards the datagram to R2.
3. Router R2 knows that computer 5 connects to the same physical network to which it is connected. Therefore, it resolves the address of computer 5 using ARP and forwards the datagram to computer 5.

From this, it should be clear that the protocol software resolves only the address for the next immediate hop, on the same network. It cannot resolve the address of a computer on another network.

17.7 REVERSE ADDRESS RESOLUTION PROTOCOL (RARP)

There is one more protocol in the ARP suite of protocols. The **Reverse Address Resolution Protocol (RARP)** is used to obtain the IP address of a host based on its physical address. That is, it performs a job that is exactly opposite to that of the ARP. An obvious question would be – is this really needed? After all, a host should have the IP address stored on its hard disk. However, there are situations when this is not the case. Firstly, a host may not have a hard disk at all (e.g., a diskless workstation). Secondly, when a new host is being connected to a network for the very first time, it does not know its IP address. Finally, a computer may be discarded and replaced by another computer, but the same network card could be reused. In all these situations, the computer does not know its own IP address.

RARP works in a very similar way to ARP, but in the exactly opposite direction, as shown in Fig. 17.29.

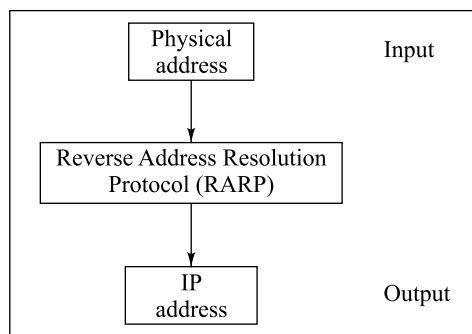


Fig. 17.29 Reverse Address Resolution Protocol (RARP)

In RARP, the host interested in knowing its IP address broadcasts an **RARP query datagram**. This datagram contains its physical address. Every other computer on the network receives the datagram. All the computers except a centralized computer (the server computer) ignore this datagram. However, the server recognizes this special kind of datagram and returns the broadcasting computer its IP address. The server contains a list of the physical addresses and their corresponding IP addresses for all diskless workstations. This is shown in Fig. 17.30.

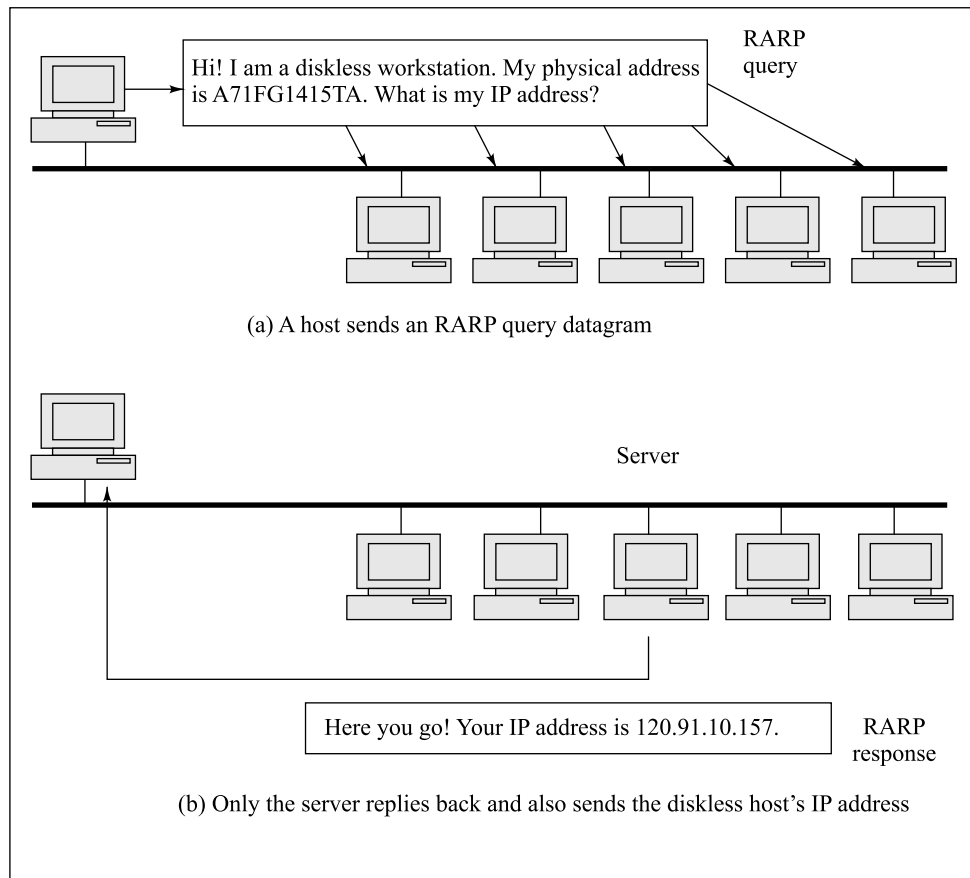


Fig. 17.30 Example of RARP

17.8 INTERNET CONTROL MESSAGE PROTOCOL (ICMP)

As we have studied previously, the Internet Protocol (IP) is a *connectionless, best-effort* data transport protocol.

By connectionless, we mean that IP sends each datagram without assuming that there is a connection between the source and the destination. Every datagram is viewed by IP as independent from all other datagrams. Of course, in order to send the datagram to the correct destination, the IP datagram header contains the destination address.

By best effort, we mean that IP makes the best effort of delivering a datagram from a source to a destination. However, it does not guarantee that the datagram would be delivered correctly. As we shall see, the correct delivery is guaranteed by the TCP protocol. It only means that the IP itself does not contain any error detection/acknowledgement/retransmission schemes for regular data datagrams. TCP, however, has all these facilities.

However, this does not mean that IP does not have any error control mechanisms at all. Although the issues of connection management between the source and the destination, correct delivery, etc., are handled by TCP, IP includes a protocol for reporting errors that can potentially bring down the Internet, at least temporarily. For example, suppose a router is receiving datagrams for forwarding at a rate that is too fast for it to handle. Or suppose that a host on the Internet is down, and not knowing this, another host is trying to send datagrams to that host repeatedly. When we consider that at the same time thousands of routers or hosts could potentially face these issues, their severe consequences can be grasped easily. Therefore, to avoid such disasters, the designers of the Internet have included the **Internet Control Message Protocol (ICMP)** that serves as an error reporting mechanism in such and similar cases.

There is another purpose for ICMP. Sometimes, a host needs to find out if a particular router is working or if it is down. ICMP facilitates these network management queries as well.

ICMP enables the detection and reporting of problems in the Internet. However it does not play any role in the correction of these problems; that is left to the hosts or routers. For instance, consider a very simple example as shown in Fig. 17.31. Router R connects two hosts A and B. We have deliberately kept it very simple. As shown, suppose that the wire between R and B is accidentally cut. Now, if host A sends any datagrams for host B, router R cannot transport them, as its connection with host B is lost. Therefore, the ICMP software (which is a part of the IP software, anyway) in router R takes over and informs host A that the destination (i.e., host B) is unreachable. However, it does not prevent A from sending more datagrams for B. Therefore, ICMP does not involve any error correction mechanisms.

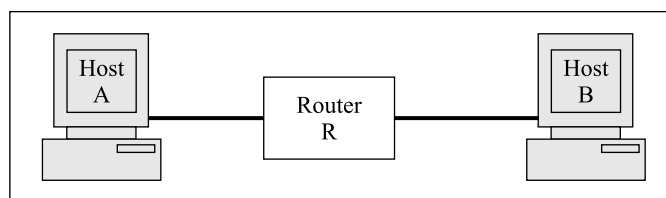


Fig. 17.31 Connection between a host and a router is lost

17.8.1 How ICMP Works

ICMP works by sending an error message for a specific reason to the concerned host. For instance, in our example of Fig. 17.30, the ICMP software on router R would send a message *Destination unreachable* to host A, when host A sends any datagrams destined for host B. Similarly, for other kinds of problems, different messages are used.

Let us consider a few examples of ICMP error messages.

1. **Destination unreachable** – We have already discussed this with reference to Fig. 17.30. Whenever a router realizes that a datagram cannot be delivered to its final destination, it sends

a *Destination unreachable* message back to the host that had sent the datagram originally. This message also includes a flag to indicate whether the destination host itself is unreachable, or whether the network containing the end destination is down.

2. **Source quench** – There are occasions when a router receives so many datagrams that it cannot handle them. That is, the number of datagrams arrived at a router could exhaust the size of its memory buffer, where it usually stores these datagrams temporarily before forwarding them to the next router/the final destination. The router cannot simply accept any more datagrams. In such situations, any more datagrams that the router receives must be discarded. However, if the router simply discards them, how would the senders know that there is a problem? The senders would have no clue. And, they could go on sending more datagrams to this router. In order to prevent such a situation, the router sends a *source quench* message to all the senders who are sending it more datagrams. This signals the hosts sending datagrams to the router, that they should not send any datagrams to that router now. Rather, they should wait for some time before transmitting more datagrams or before retransmitting the datagrams discarded by the router.
3. **Redirect** – When a host creates a datagram for sending it to a particular destination, it first sends it to the nearest router. The router then forwards it on to another router, or the end destination, if the end destination is directly reachable. However, during the journey of a datagram via one or more routers like this, it could happen that a router incorrectly receives a datagram, which is not on the path of the end destination. The datagram should, instead, go to another router. In such a case, the router that received the datagram incorrectly sends a *redirect* message to the host or network from where it received that datagram.

Figure 17.32 shows such an example. Here, host A wants to send a datagram to host B. We realize that the datagram should be first forwarded to the router R2, as both host A and router R2 are on the LAN shown in the figure. Thereafter, the router R2 should forward it to the host B, as both router R2 and the host B are on the WAN shown in the figure. Let us assume

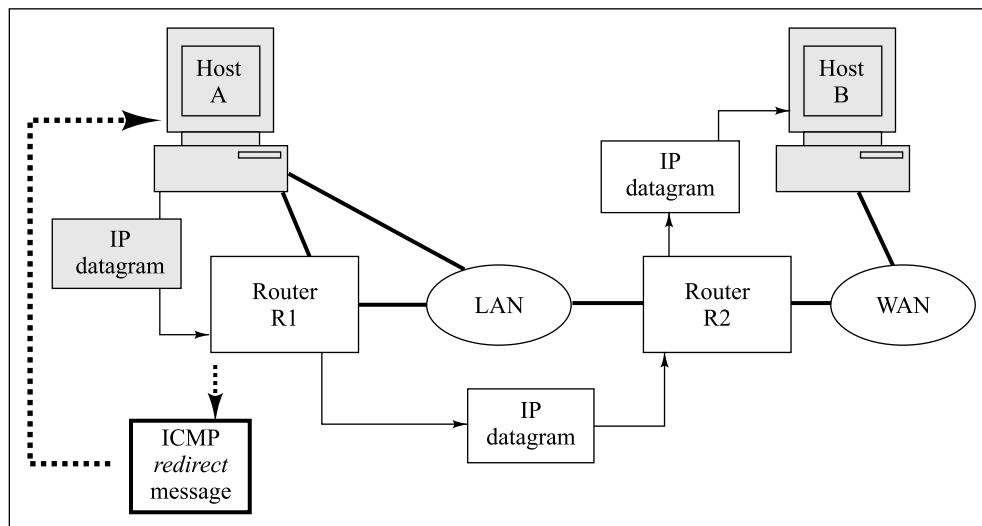


Fig. 17.32 Example of Redirect ICMP message

that, by mistake, the host A first sends the datagram to router R1. However, R1 is not directly on the route of host B. Router R1 realizes this, and forwards the datagram to the appropriate router R2 after consulting its routing table, which tells R1 that if you have to send a datagram from R1 to B, it will have to be sent to R2. At the same time, R1 sends a *redirect* message back to host A to ensure that host A updates its routing table and sends all datagrams destined for host B thereafter to router R2.

4. **Time exceeded** – We know that every IP datagram contains a field called *time to live*. This field is used to determine how long the datagram can live. This helps the Internet infrastructure in preventing datagrams from living and moving on for too long, especially when there are network problems. For instance, suppose that host A sends a datagram to host B and that the two hosts are separated by a number of intermediate routers. Initially, the host A sets this value based on the expected number of routers that the datagram is expected to pass through (may be a little more than that number). Then every time a datagram moves from A to B via these routers, the router reduces the amount of the field *time to live* of that datagram by a certain value before forwarding it to the next router. If a router receives this field with the value of *time to live* being zero, it means that the datagram must be discarded, as it is moving through too many routers. Therefore, the router discards this datagram. It then communicates this fact to the original sending host by a *time exceeded* ICMP message. The original host can then take an appropriate corrective action, such as choosing another router or waiting for some time before retransmission.

To avoid sending long text messages such as *destination unreachable*, ICMP actually maps these messages to error codes, and just sends the error code to the host. For instance, when a router has to send a *destination unreachable* message to a host, it sends an error code of 3. The number 3 corresponds to the *destination unreachable* message. This can be done by standardizing all error codes vis-à-vis their corresponding messages and making that table of codes and messages a part of the ICMP software. A few ICMP error codes and their corresponding messages for the ones discussed earlier are shown in Table 17.1.

Table 17.1 ICMP error codes and error messages

Error code	Error message
3	Destination unreachable
4	Source quench
5	Redirect
11	Time exceeded

17.8.2 Delivering ICMP Messages

Although ICMP is a network layer protocol similar to IP, its messages are not sent directly to the data link layer. Instead, it uses the services of IP, as we shall see. ICMP messages are made up of ICMP datagrams. Each ICMP datagram contains two fields, viz., an ICMP header and an ICMP data area. At a lower level, the complete ICMP datagram is then encapsulated inside the data portion of an IP datagram, and is delivered as an IP datagram. Of course, at the lowest level, the IP datagram would be encapsulated inside the data portion of a hardware frame. This is shown in Fig. 17.33.

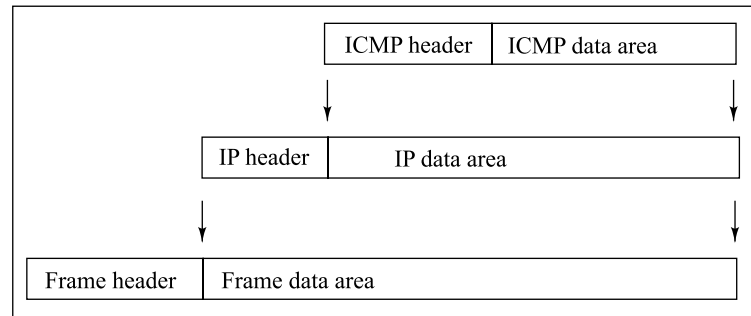


Fig. 17.33 ICMP datagram inside IP datagram; IP datagram inside hardware frame

As we have discussed, the ICMP message is sent back to the host, which should be informed about the problem. But how does a router know to which host it should send the ICMP error message? It is actually quite simple. An ICMP message is always sent back in response to a received IP datagram (for instance, when a router receives a datagram while experiencing *source quench*). This original IP datagram sent by the host contains the host's IP address in the field 'Source address' of that IP datagram. Therefore, the router wanting to send an ICMP message back to that host simply examines that datagram, extracts the 'Source address' of that datagram, and uses it as the destination address for its ICMP datagram. Conceptually, this is very similar to what happens when we click the 'Reply' button when using email. The email program automatically inserts the email address of the original sender in the 'To' field of our reply.

17.9 DATAGRAM FRAGMENTATION AND REASSEMBLY

As we have discussed, an IP datagram (also called a datagram) can travel through a number of different networks on its way to the ultimate destination. During this process, the IP datagram is encapsulated by the various routers into hardware frames. This is shown in Fig. 17.34.

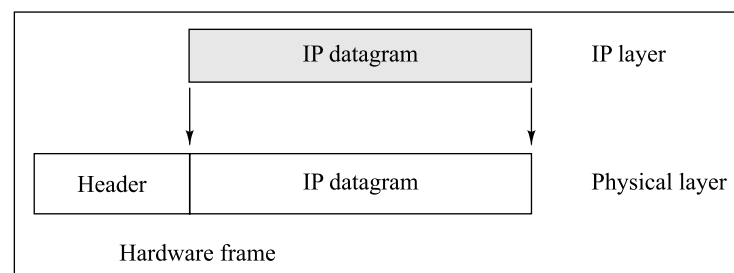


Fig. 17.34 IP datagram encapsulated into a hardware frame

As the figure shows, the entire IP datagram becomes the data portion inside the hardware frame. We have seen that the IP datagram itself consists of IP header + IP datagram, and IP datagram, in turn, consists of TCP header + TCP datagram. The IP header contains the 32-bit IP addresses of the source and the destination. From the IP layer of a host, the IP datagram is sent to the physical layer. At the physical layer, the network hardware creates a hardware frame that contains the frame header as usual, and encapsulates the entire IP datagram as its data portion. The frame header

contains the physical addresses of the source and the destination nodes on the same network (i.e., two adjacent nodes/routers on the same network in the whole route from one extreme source and destination nodes). After this, the host sends the hardware frame on its way to the next router. The router receives the hardware frame, extracts the IP datagram out of it, and creates another hardware frame out of it, by encapsulating the IP datagram into another hardware frame, based on the next network over which the datagram has to be sent.

17.9.1 Maximum Transmission Unit (MTU)

Every physical network topology has certain characteristics that are specific to the physical medium used in that network. For example, the data link layer protocol of different network types has its own hardware frame format. One of the most important characteristics of this format is the maximum amount of data that a frame can carry at a time, i.e., in a single frame. This amount of data is called **Maximum Transmission Unit (MTU)**. Because of the hardware and software differences, the MTU for two different network types is different. For instance, the MTU for a Token Ring is 17914 bytes (including the header). That is, any node in a Token Ring network can transmit up to a maximum of 17914 bytes at one time in a single frame. Similarly, the Ethernet MTU is 1500 bytes, whereas the MTU of a FDDI network is 4352 bytes. Thus, different network topologies put an upper limit on how many bytes can be transmitted as a single unit by any host connecting to that network. Table 17.2 shows a list of MTU values for some of the popular network topologies.

Table 17.2 *Different network topologies and their MTUs*

Network type (Topology)	Maximum Transmission Unit (MTU) in bytes
Token Ring	17914
Token Bus	8166
FDDI	4352
Ethernet	1500
Point-to-Point Protocol (PPP)	1500
Serial Line IP (SLIP)	1006
X.25 and ISDN	576

Coming back to our earlier example, at a higher level, data from one computer is sent to another over the Internet as IP datagrams. Since IP is a software abstraction aimed at resolving the hardware and software differences between incompatible network types, the size of an IP datagram must not depend on the underlying network type. It must always be the same, no matter what the underlying network type is. That is, the MTU for an IP datagram must be the same for all the different network types.

When the IP datagram format was finalized, the engineers decided to allow an IP datagram to carry the maximum number of bytes that any network type allows. This was to allow an IP datagram to be independent of all the underlying network types. That is, the maximum number of data bytes allowed in an IP datagram = Maximum MTU size of any network type. It was realized that the maximum MTU among all the various network types is 65535, allowed by a Hyperchannel (a network type like Ethernet/Token Ring). Therefore, the maximum data that an IP datagram can carry is also 65535. That is, the MTU for IP is always 65535 regardless of the underlying physical medium / network type.

17.9.2 Fragmentation

We had raised the following question earlier – *What happens if the size of the IP datagram is greater than the maximum size allowed by a networking technology for a hardware frame?*

We can now rephrase the question as – *How do we handle situations where the IP datagram contains more data than the MTU of a network type?*

In such a case, the original IP datagram is fragmented into more than one IP datagrams before encapsulating them inside the hardware frames, and then these frames are forwarded to the next router/end destination where they are reassembled into the original datagram.

When an IP datagram is fragmented into more than one IP datagram fragment, each of these fragments has its own header. Many of these header fields remain the same, while some others change. We shall study this later. However, we must also be aware of another possibility. If an IP datagram is fragmented into smaller fragments, encapsulated inside the hardware frames and sent forward, there could be a situation where the next network that it encounters has still smaller MTU size. In such a case, each of the fragments themselves is further fragmented, and the fragments at this second level are sent forward. We shall also examine this case with an example later. However, the point is that an IP datagram could be fragmented several times before it reaches the ultimate destination. This is shown in Fig. 17.35.

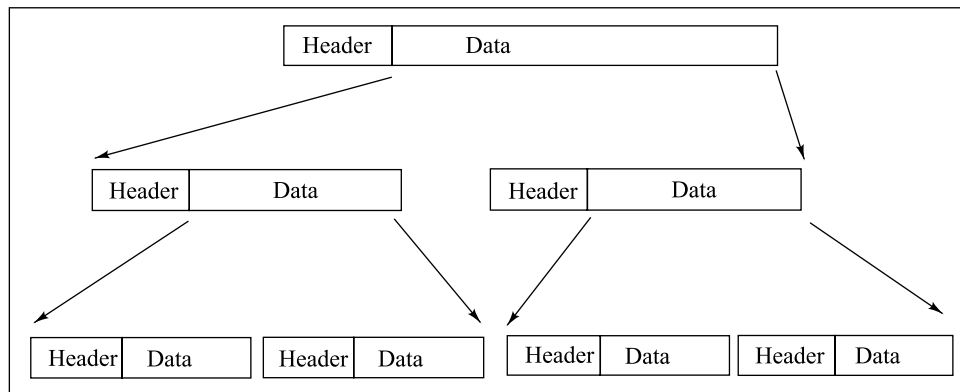



Fig. 17.35 Fragmenting an IP datagram and fragmenting the fragment

Fragmenting an IP datagram to accommodate it inside hardware frames is fine. However, once they reach the destination, these fragments must be reassembled properly to recreate original the IP datagram. This process of recreating an IP datagram from its fragments is called **reassembly**. The fragmentation can be performed by a source or an intermediate router, which is on the way of an IP datagram. However, the reassembly must be performed only by the final destination. This is because from the point of view of the intermediate routers and networks, every fragment is an independent IP datagram. There is no guarantee that these fragments travel in the same route. Therefore, any attempt to group them to recreate an IP datagram out of them at any intermediate points would not server any purpose. However, the final destination receives all these fragments regardless of the route that each one of them takes. Therefore, it is logical that the final destination alone is responsible for reassembly.

Fields Related to Fragmentation

Let us look at the format of IP datagram as shown in Fig. 17.36.

Version (4 bits)	HLEN (4 bits)	Service type (8 bits)	Total length (16 bits)	
Identification (16 bits)			Flags (3 bits)	Fragmentation offset (13 bits)
Time to live (8 bits)	Protocol (8 bits)		Header checksum (16 bits)	
Source IP address (32 bits)				
Destination IP address (32 bits)				
Data				

Fig. 17.36  *Format of IP datagram*

Let us now consider the fields of an IP datagram header that are important while fragmenting the datagram. The host or router fragmenting a datagram must be extremely careful with three fields of the IP datagram header during fragmentation. These three fields are identification, flags and fragmentation offset, as explained below.

1. **Identification** – This is a 16-bit field inside the header of an IP datagram. We know that each IP datagram header also has a *source IP address* field, which contains the IP address of the source, from where the IP datagram is originating. The combination of the *source IP address* and *identification* fields helps identify an IP datagram over the Internet uniquely. The IP software inside each computer on the Internet maintains a counter in its memory. Each time the host wants to send an IP datagram on to the Internet, the IP software increments this counter by one, and sets the value of the *identification* field equal to that of the counter. This ensures that the counter field, and therefore, the *identification* field, is always unique. This also helps in identifying an IP datagram over the Internet uniquely. Obviously, whenever an IP datagram is fragmented, each of the fragments must have the same value in the *identification* field as it was in the original IP datagram (but different *fragmentation offset*, as we shall learn later). Therefore, the host or router fragmenting an IP datagram must ensure that the value of the *identification* field is copied into all fragments of the datagram. This helps the final destination host to reassemble the fragments into a single datagram based on the common *identification* number. It knows that all the fragments that have the same *identification* number must be assembled into a single datagram.
2. **Flags** – This is a 3-bit field, as shown in Fig. 17.37.

Reserved	Do not fragment	More fragments
Bit 0	Bit 1	Bit 2

Fig. 17.37  *Flags field*

The first of the three bits is reserved.

The second bit is called *do not fragment* bit. If this bit is set to 1, it means that this IP datagram must not be fragmented. Therefore, if the MTU of the underlying network type is smaller than the IP datagram size, the host must discard, but must not fragment, this datagram. After discarding the datagram, the host sends an ICMP error message to the original source of the IP datagram from where the datagram came. However, if this bit contains a 0, it means that this IP datagram can be fragmented, if required.

The third bit in this field represents the *more fragments* bit. If it is set to 1, it means that this is a fragment, and moreover, it is not the last fragment of an IP datagram – there are more fragments of this IP datagram to follow. If this bit contains a 0, it means that this is the last fragment of an IP datagram.

3. **Fragmentation offset** – This 13-bit field indicates the relative position of the fragment with respect to the original, whole IP datagram. It indicates the starting byte position in the original IP datagram for that specific fragment. This field is used to indicate which parts of the original IP datagram can be found and in which fragments. This field indicates the starting position in terms of eight-byte blocks. For instance, suppose an IP datagram contains 256 bytes of data, of which, the first 248 bytes are stored in the first fragment and the rest 8 bytes in the second fragment. Then, the *fragmentation offset* for the first fragment would be 0; starting counting at 0, because it is the first fragment containing bytes 0 to 247. For the second fragment, we should expect that the *fragmentation offset* should contain 248. However, because the *fragmentation offset* is stored as a multiple of eight-byte blocks, this field for the second fragment contains 31 (because $248/8 = 31$). This also puts restriction on fragmentation. For instance, you could not fragment at random byte numbers.

Rules of Fragmentation

Before discussing an actual example of fragmentation, let us review the rules that a host/router performing the fragmentation process must follow when fragmenting an IP datagram.

1. The header portion of an IP datagram is not fragmented. Only the data portion is fragmented.
2. The datagram header sizes are not considered when calculating the fragment sizes. For instance, suppose there is an IP datagram of size 5000 bytes, of which 20 bytes are used for the IP header. Now, if this datagram is to be fragmented into two smaller fragments, only the data portion of 4980 bytes is fragmented into two fragments of 2490 bytes each. Each fragment would additionally have 20 bytes of header. This is shown in Fig. 17.38.
3. As shown in the figure, each resulting fragment has its own headers.
4. Fragmentation must occur on an eight-byte boundary. If an IP datagram contains 256 bytes of data, and only 250 bytes out of these can fit into a fragment, then only 248 bytes are actually stored in the first fragment because of this reason. The remaining eight bytes would be put into another fragment.

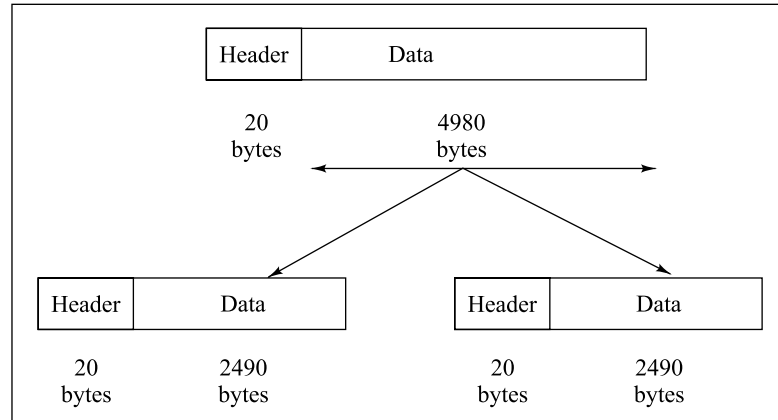


Fig. 17.38 Only the data portion of an IP datagram gets fragmented

17.10 COMPARISON OF OSI AND TCP/IP PROTOCOL SUITES

Now that we have a fair understanding of the OSI and TCP/IP protocol suites, let us do a quick comparison between the two in summary form. This is shown in Table 17.3.

Table 17.3 Comparison of OSI and TCP/IP protocol suites

Point	OSI	TCP/IP
Number of layers	7	3 main layers and then provision for data link and physical layers
Layers	Presence of session and presentation layers	Absence of session and presentation layers
Role of the application layer	Only user applications run here	In addition to user applications, work related to user session and presentation of information is done here
Conceptual clarity	Clearer and easier to understand	More complex than OSI
Practical usage	Low	Very high
Redundancy/Duplication of duties across different layers	High	Low
Development/Standardization	Done by OSI	Done by International Telegraph and Telephone Consultative Committee (CCITT)

SUMMARY

The Internet is a network of many heterogeneous computer networks. This means that physical addressing would not work on the Internet. Therefore, the concept of logical addresses is used on the Internet to make it a virtual network, which hides the complexities of the underlying different

heterogeneous physical networks. This logical address is the 32-bit IP address. However, at the lowest level, computers still must use physical addresses to communicate with computers on the same network. Therefore, we need a mechanism to convert a logical IP address to a physical address before the actual transmission can take place over a local network. To resolve this problem, the concept of address translation is used. The Address Resolution Protocol (ARP) is a mechanism that specifies the IP address of a computer, and gets back its physical address.

IP and ARP work together to deliver the IP datagrams. A router maintains a table to decide which destination addresses are directly reachable, and for which other addresses it has to forward the datagrams to another router.

A router is a special purpose computer that connects to two or more networks. This means that a router must have two or more IP addresses, one per network that it connects to. This allows a router to forward datagrams from one network to another.

TCP/IP consists of 3 layers, viz., Internet, Transport, and Application. The Internet layer is unique to TCP/IP. It includes a very significant protocol called Internet Protocol (IP). IP is the main protocol responsible for uniform host addressing, datagram formats and lengths and routing. The transport layer is responsible for end-to-end delivery of IP datagrams, and contains two main and widely differing protocols, viz., Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). The application layer sits on top of the transport layer, and allows the end users to do Web browsing (using HTTP), file transfers (using FTP) and send emails (using SMTP), etc. This runs only on the host.

The Internet Control Message Protocol (ICMP) is a error-reporting (but not error-correcting) protocol. Examples of ICMP messages are *destination unreachable*, *source quench*, *redirect*, *time exceeded*, etc.

This process of breaking down an IP datagram is called fragmentation. Of course, at the destination, all the fragments must be collected together and from them, the original IP datagram must be formed again.

KEY TERMS AND CONCEPTS

Address Resolution Protocol (ARP)	Fragmentation offset
Address binding	Header Length (HLEN)
ARP caching	Host id
ARP query datagram	Hyper Text Transfer Protocol (HTTP)
Best effort delivery	Identification
Class	Internet Protocol (IP)
Connectionless	IP address
Destination address	IP Next Generation (IPng)
Destination unreachable	IP version 4 (IPv4)
Dotted decimal notation	IP version 6 (IPv6)
Electronic mail (Email)	Internet Assigned Number Authority (IANA)
File Transfer Protocol (FTP)	Internet Control Message Protocol (ICMP)
Flags	Maximum Transmission Unit (MTU)
Fragmentation	Media Access and Control (MAC)

Network id	Subnet
Octet	Subnetwork
Options	Subnetting
Prefix	Suffix
Protocol	Time exceeded
RARP query datagram	Time to live
Reassembly	TELNET
Redirect	Total length
Remote login	Transmission Control Protocol (TCP)
Reverse Address Resolution Protocol (RARP)	Transmission Control Protocol/Internet Protocol (TCP/IP)
Service type	Trivial File Transfer protocol (TFTP)
Simple Mail Transfer Protocol (SMTP)	Unreliable
Sliding window	User Datagram Protocol (UDP)
Socket	World Wide Web
Source address	
Source quench	

QUESTIONS

True/False

1. There is a single method to assign the hardware address to a computer.
2. Configurable address does not change, and is provided by the network hardware manufacturer.
3. In case of a dynamic address, the physical address keeps on changing every time a computer is switched off and on.
4. The Network Interface Card fits in one of the slots of the motherboard of the computer, and therefore, is inside the computer.
5. Logical addressing is used for defining an addressing mechanism that depends on the underlying physical addresses.
6. When a computer wants to communicate with another computer on the Internet, it uses the IP address of the destination computer.
7. To send any information, the sender need not know the IP address of the recipient.
8. To ensure that no two IP addresses are ever the same, there is a central authority that issues the prefix or network number portion of the IP addresses.
9. To send a single message a group of computers, a group of computers must share the common multicast address.
10. A router needs to have as many IP addresses as the number of networks that it connects to, for forwarding datagrams between computer networks over the Internet.
11. TCP and UDP are a part of the transport layer.
12. The Internet layer follows a datagram philosophy.
13. UDP offers reliability.
14. An IP datagram is a fixed-length datagram.
15. If a datagram is fragmented, the *fragmentation offset* field is useful.
16. An IP address is made up of two parts, viz., network number and host number.
17. IP guarantees datagram delivery.

18. Obtaining the IP address of a host based on its physical address is called address translation.
19. ARP is used for address binding.
20. Address resolution is a process used for obtaining the physical address of a computer based on its IP address.
21. Computers identify each other based on the physical addresses hard-coded on their Network Interface Cards (NICs).
22. IP addresses can be transformed into their corresponding physical addresses by using a table lookup.
23. A computer can resolve another computer's address even if both the computers are on different physical networks.
24. There are situations when the computer does not know its own IP address.
25. IP is a connection-oriented protocol.
26. Internet Control Message Protocol (ICMP) serves as an error reporting as well as error-correcting mechanism.
27. *Source quench* message signals the hosts sending datagrams to the router, that they should not send any datagrams to that router now.
28. If a router receives this field with the value of *Time to live* being zero, it means that the datagram must be discarded, as it is moving through too many routers.
29. ICMP is a transport layer protocol.
30. The size of an IP datagram can be greater than the maximum size allowed for a network.
31. Maximum Transmission Unit for an IP datagram must be the same for all the different network types.
32. The process of recreating an IP datagram from its fragments is called reassembly.
33. The combination of the *source IP address* and *identification* fields helps identify an IP datagram over the Internet uniquely.
34. Whenever an IP datagram is fragmented, each of the fragments can have different value in the *identification* field.
35. The *flags* field in the IP datagram header is used to indicate which parts of the original IP datagram can be found in which fragments.

Multiple-Choice Questions

1. Layer 4 from bottom in TCP/IP is the _____.
 - (a) physical layer
 - (b) application layer
 - (c) transport layer
 - (d) internet layer
2. ARP lies in the _____.
 - (a) physical layer
 - (b) application layer
 - (c) transport layer
 - (d) internet layer
3. _____ does not offer reliable delivery mechanism.
 - (a) UDP
 - (b) TCP
 - (c) ARP
 - (d) FTP
4. IP address _____ the physical address.
 - (a) is the same as
 - (b) has no relation with
 - (c) means
 - (d) None of the above

5. Currently, the IP address has a size of _____ bits.
 - (a) 128
 - (b) 64
 - (c) 32
 - (d) 16
6. The _____ field helps routers in discarding packets that are likely to cause congestion.
 - (a) time to live
 - (b) options
 - (c) protocol
 - (d) fragmentation offset
7. IP makes a _____ of datagram delivery.
 - (a) worst effort
 - (b) guaranteed delivery
 - (c) best effort
 - (d) All of the above
8. In _____ scheme, the physical address is hard coded on the NIC of a computer.
 - (a) configurable addresses
 - (b) static addresses
 - (c) dynamic addresses
 - (d) None of the above
9. The _____ for all computers on the same physical network is the same.
 - (a) host id
 - (b) physical address
 - (c) IP address
 - (d) network id
10. If an IP address starts with a bit sequence of 110, it is a class _____ address.
 - (a) A
 - (b) B
 - (c) C
 - (d) D
11. The maximum number of class B networks is _____.
 - (a) 6480
 - (b) 12808
 - (c) 25610
 - (d) 16384
12. Given a _____ address, ARP helps find the corresponding _____ address.
 - (a) physical, logical
 - (b) physical, router
 - (c) logical, physical
 - (d) logical, router
13. When a datagram cannot be delivered to a destination, a router sends a message _____.
 - (a) source quench
 - (b) destination unreachable
 - (c) redirect
 - (d) time exceeded
14. The time-to-live field in an IP datagram is related to the ICMP message _____.
 - (a) source quench
 - (b) destination unreachable
 - (c) redirect
 - (d) time exceeded
15. The MTU of _____ is the highest.
 - (a) Token Ring
 - (b) Ethernet
 - (c) X.25
 - (d) PPP
16. The header portion of an IP datagram is _____.
 - (a) always fragmented
 - (b) fragmented if required
 - (c) never fragmented
 - (d) None of the above
17. The 'do not fragment' flag containing a zero indicates that the datagram _____.
 - (a) must be fragmented
 - (b) must not be fragmented
 - (c) is already fragmented
 - (d) may be fragmented
18. A zero in the 'more fragments' flag indicates that _____.
 - (a) this is the last fragment
 - (b) there are more fragments to follow
 - (c) this is the only datagram
 - (d) this is an Ethernet frame

Detailed Questions

1. What are the different methods of assigning physical address to a computer?
2. Explain the process of message transfer between two computers.
3. Describe the three parts of an IP address.
4. Explain how an IP address is designed to a host.
5. What is IP Version 6 (IPv6)? Why is it required?
6. Describe the various layers in TCP/IP in brief.
7. Explain how a message sent by an application on one host reaches the application on another host via one or more routers using TCP/IP.
8. Describe the various fields in the IP datagram header.
9. What is the purpose of the *time to live* field of the IP datagram header?
10. Why IP is called connectionless?
11. How does the Address Resolution Protocol (ARP) work?
12. Explain why ARP is local.
13. Explain the Reverse Address Resolution Protocol (RARP). Why is it required?
14. Discuss how IP and ARP work together.
15. Why IP is called a *best-effort delivery* protocol?
16. Explain the following ICMP messages: (a) Destination unreachable, (b) Source quench, and (c) Redirect.
17. What is the purpose of the field *time to live* in the IP datagram header?
18. How does ICMP software on a host know to which other host it should send an error message?
19. Explain the process involved in sending a datagram from one host to another host on a different network.
20. What is Maximum Transmission Unit (MTU)?
21. How the situation is handled using fragmentation where the IP datagram contains more data than the MTU of a network type?
22. Describe the three fields of the IP datagram header—identification, flags and fragmentation offset.
23. Explain the rules of fragmentation.
24. What is subnetting? Explain with an example.
25. Compare the OSI and TCP/IP protocol stacks.

TCP/IP

Part 2: TCP and UDP

18

18.0 INTRODUCTION

The transport layer runs on top of the Internet layer and is mainly concerned with the transport of packets from the source to the ultimate destination (i.e., **end-to-end delivery**). Unlike IP, which is involved in the routing decisions, the main function of the TCP protocol that runs in the transport layer is to ensure a correct delivery of packets between the end points. In TCP/IP, the transport layer is represented by two protocols, viz., TCP and UDP. Of the two, UDP is simpler. UDP is useful when speed of the delivery is critical; never mind the loss of a few datagrams such as in the transmission of digitized images, voice or video. TCP is useful when it is essential to make sure that datagrams travel without errors from the source to the destination, even if that makes the overall delivery slightly more time consuming. This is not only useful but also necessary in the transmission of business of scientific data. This is shown in Fig. 18.1.

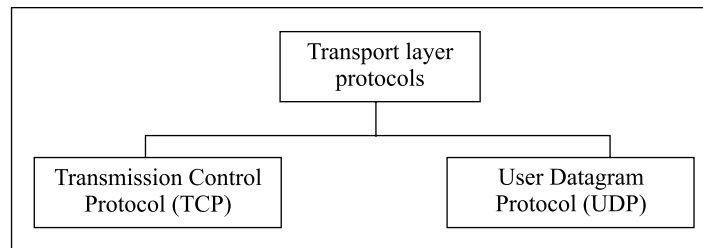


Fig. 18.1 *Transport layer protocols on the Internet*

In this chapter, we shall discuss TCP and UDP.

18.1 TCP BASICS

The Transmission Control Protocol (TCP) works extremely well with IP. Since the Internet uses packet switching technique, there could be congestion at times. TCP takes care of these situations and makes the Internet reliable. For example, if a router has too many packets, it would discard some of them. Consequently, they would not travel to the final destination and would get lost. TCP automatically checks for lost packets and handles this problem. Similarly, because the Internet offers alternate routes (through routers) for data to flow across it, packets may not arrive at the destination in the same order as they were sent. TCP handles this issue as well. It puts the packets back in

order. Similarly, if some packets are duplicated due to some hardware malfunction, TCP discards the duplicate packets. We shall discuss how TCP deals with these issues later.

18.2 FEATURES OF TCP

Let us list the main features offered by the TCP portion of the TCP/IP protocol suite. These are *reliability*, *point-to-point communication*, and *connection-oriented approach*.

1. **Reliability** – TCP ensures that any data sent by a sender finally arrives at the destination as it was sent. This means, there cannot be any data loss or change in the order of the data. Reliability at the transport layer (TCP) has four important aspects as shown in Fig. 18.2.

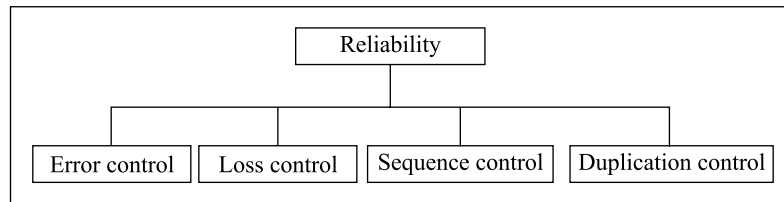


Fig. 18.2 Four aspects of reliability in transport layer delivery

Let us examine these four aspects in brief.

- (a) **Error control** – Data must reach the destination exactly as it was sent by the sender. We have studied mechanisms such as checksums and CRC that help achieve error control at the lower layer, the data link layer. However, TCP provides its own **error control** at a higher layer, the transport layer. Why is this additional error control at the transport layer necessary when the data link layer already takes care of it? The answer lies in the fact that error control at the data link layer ensures error-free delivery between two networks, however, it cannot guarantee this error-free delivery end-to-end (i.e., between the source and the destination). The reason behind this is that if a router that connects two networks introduces some errors, the data link layer does not catch them. The data link layer only ensures the network-to-network error control. This is shown in Fig. 18.3. This is the reason why TCP provides for its own error control mechanism.

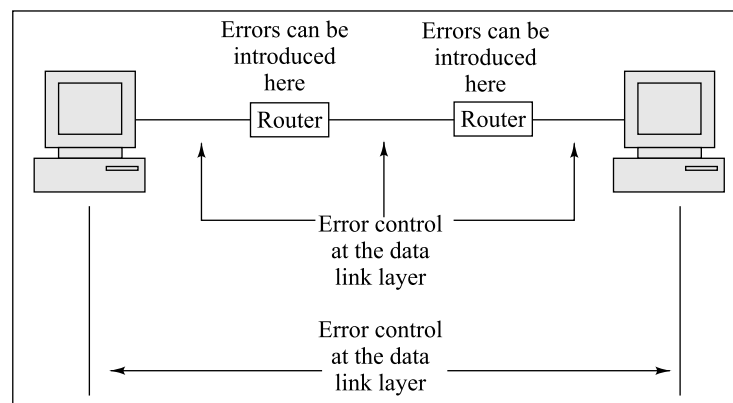


Fig. 18.3 Transport layer and data link layer error control

- (b) **Loss control** – It might happen that the source TCP software breaks the original message into three packets, and sends the three packets to the destination. As we know, the IP software is connectionless and does not guarantee delivery (*best effort*). It might happen that one of the three packets gets lost midway, and never reaches the destination. TCP should ensure that the destination knows about this, and requests for a retransmission of the missing packets. This is **loss control**.

How can this be achieved? It should be easy to imagine that TCP can number packets as 1, 2 and 3 when breaking the original message into packets, and can check if all the three have arrived correctly at the destination. This is shown in Fig. 18.4.

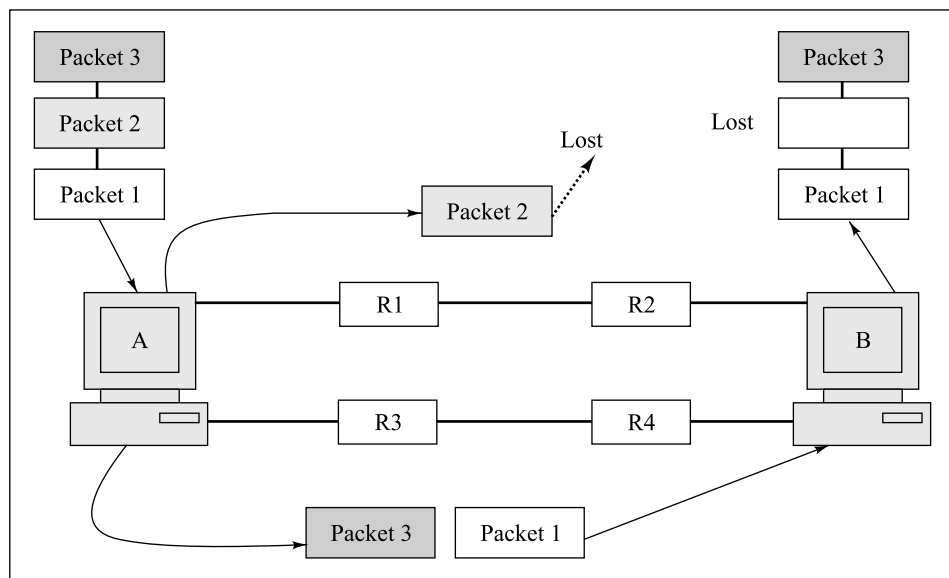


Fig. 18.4 Loss control

- (c) **Sequence control** – Since different packets of the same message can take different routes to reach the same destination, they could reach out of sequence.

For instance, suppose the sender host A sends three packets destined for another host B. Suppose that packet 1 and packet 3 travel via routers R3 and R4, whereas packet 2 travels via routers R1 and R2. It might very well happen that the receiver host B receives packets 1 and 3 first, and then packet 2 (because there are some congestion conditions on the route A-R1-R2-B). Thus, the sending sequence from A was packet 1, 2 and 3. The receiving sequence at host B is packet 1, 3, 2. This problem of is shown in Fig. 18.5. TCP deals with such a problem with the help of proper **sequence control** mechanisms.

- (d) **Duplication control** – **Duplication control** is somewhat opposite to *loss control*. In case of *loss control*, one or more *lost* packets are detected. In *duplication control*, one or more *duplicate* packets are detected. Since the same packet can arrive at the destination twice or more, the destination must have some mechanism to detect this. Thus, it can accept only the first packet, and reject all its (duplicate) copies.

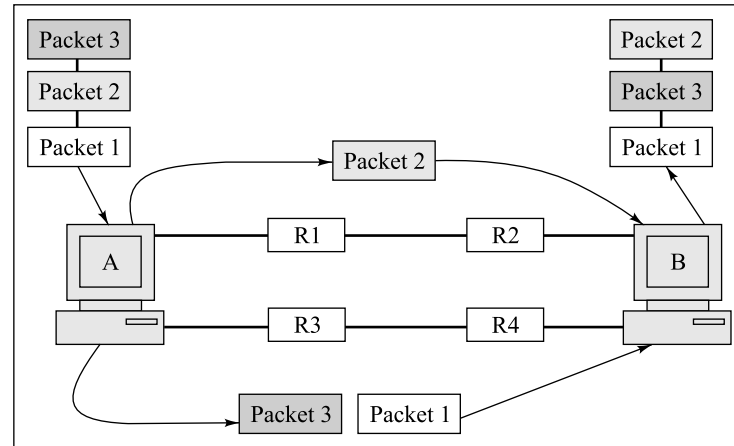


Fig. 18.5 Sequence control

Figure 18.6 shows an example of duplication. Here, packet 3 arrives at the destination host B two times. The TCP software at host B would detect this, and retain only one of them, discarding the other (redundant) copy.

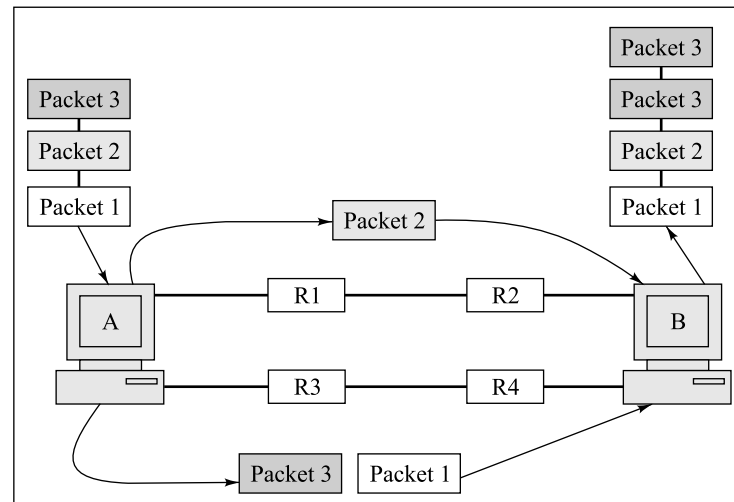


Fig. 18.6 Duplication control

2. **Point-to-Point communication** – Also called **port-to-port communication**, each TCP connection has exactly two end-points, viz., a source and a destination. Moreover, for the communicating parties, the internal details such as routing are irrelevant. They communicate as if there is a direct connection between them. Also, there is no confusion about who is sending data or who is receiving it, simply because only two computers are involved in a TCP connection. Also, this communication is *full duplex*, which means that both the participating computers can simultaneously send messages to the other.

3. **Connection-oriented** – TCP is connection oriented. The connections provided by TCP are called **virtual connections**. The term *virtual* is used because physically there is no direct connection between the computers, i.e., it is achieved in the software, rather than in the hardware. This means that a connection must be established between the two ends of a transmission before either can transmit data. Thus, an application must first request TCP for a connection to a destination and only when that connection is established can it perform the data transmission. It is very important to note that this is different from a *virtual circuit*. In a *virtual circuit*, the sender and the receiver agree upon a specific physical connection (path) to be used among all those possible, before transmission can start. This path defines the physical route of transmission (i.e., which routers the message shall pass through) for every message/packet. Thus, the entire transmission path is aware of a connection. However, in a TCP *virtual connection*, only the sender and the receiver are aware of the connection; the intermediate nodes and routers do not have a clue about this. From their perspective, it is simply a matter of passing the received packets forward via the best route possible. This route itself may (and actually does, many times) vary for every packet.

18.3 RELATIONSHIP BETWEEN TCP AND IP

It is interesting to know the relationship between TCP and IP. Each TCP message gets encapsulated in an IP datagram and then the datagram is sent across the Internet. When IP transports the datagram to the final destination, it informs the TCP software running on the final destination about it and hands the datagram over to TCP. IP does not bother about the contents of the message. It does not read them, and thus, does not attempt to interpret them in any manner. IP acts like the postal service that simply transfers datagrams between two computers. Therefore, from the view point of TCP, IP is simply a communication channel that connects computers at two endpoints. Thus, TCP views the entire Internet as a communication channel that can accept and deliver messages without altering or interpreting their contents. From the view point of IP, each TCP message is *some* data that needs to be transferred – *what* that data is, is of little consequence. This is shown in Fig. 18.7.

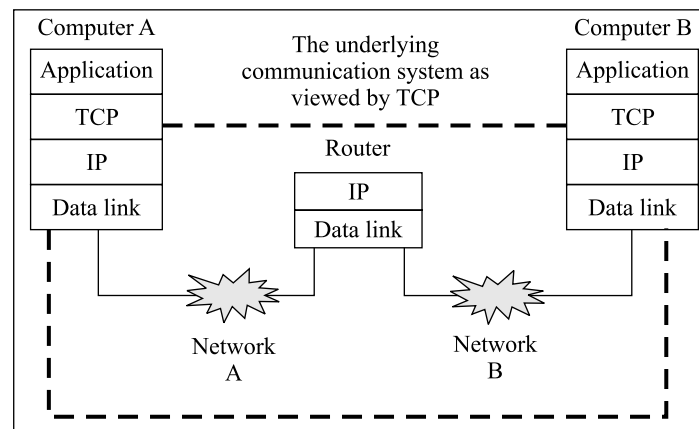


Fig. 18.7 TCP protocol's view of the Internet communication system

Conceptually, we can think that many applications (such as File Transfer Protocol (FTP), Remote login – TELNET, Email – SMTP, World Wide Web – HTTP, etc.) keep sending data to the TCP

software on a sending computer. The TCP software acts as a data multiplexer at the sender's end. The TCP software receives data from the applications, multiplexes them and gives it to the local IP software. That is, regardless of which application (FTP, TELNET, etc.) is sending data, TCP puts this data into TCP packets and gives it to IP. IP adds its own header to each such TCP packet to create an IP packet out of it, and from there, it is converted into a hardware frame and sent to the appropriate destination.

At the destination's end, the IP software receives this multiplexed data (i.e., an IP packet after removing the frame header) from its physical layer, and gives it to the local TCP software. The TCP software at the destination's end then demultiplexes the data (i.e., first removes the IP header to extract the TCP packet, and then removes the TCP header to get the original message) and gives it to the concerned application. This idea of multiplexing and demultiplexing is shown in Fig. 18.8.

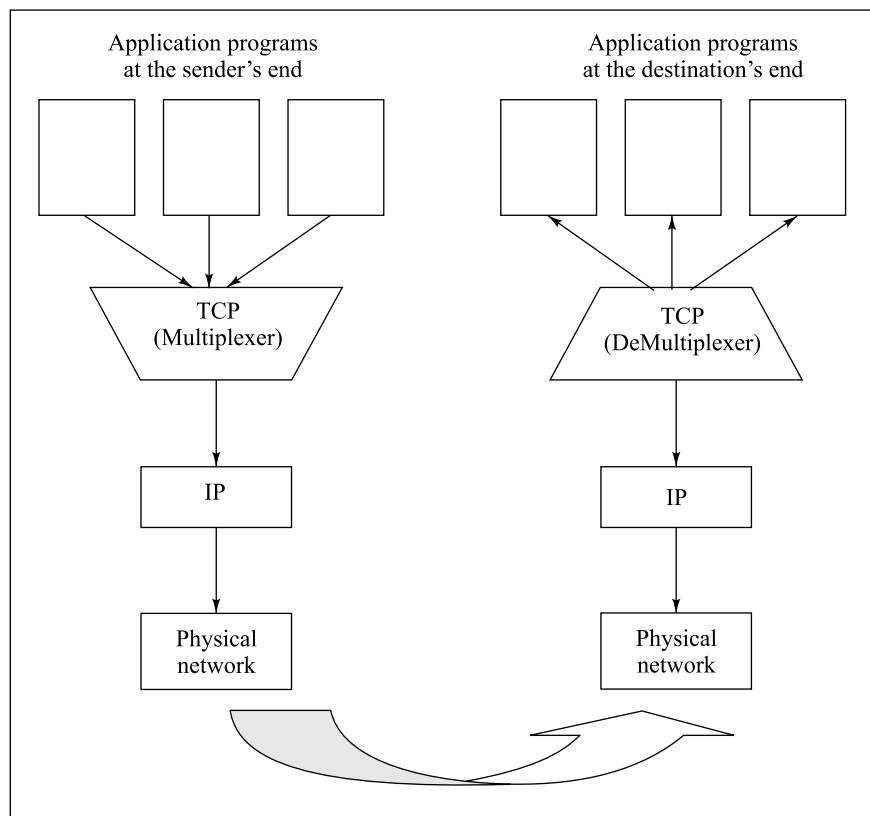


Fig. 18.8 TCP acts as a multiplexer/demultiplexer

18.4 PORTS AND SOCKETS

18.4.1 Ports

Applications running on different hosts communicate with TCP with the help of a concept called **ports**. A port is a 16-bit unique number allocated to a particular application. When an application

on one computer wants to open a TCP connection with another application on a remote computer, the concept of port comes handy. To understand why, let us use a simple analogy. Suppose a person A wants to call another person B, working in an office, over the phone. First A has to dial the phone number of B's office. After A does this, suppose an operator answers the call at the other end. Now, A must tell B's extension number (or name) to be able to connect to B. The operator would use this information (the extension number or name of B) to redirect the call to B.

In exactly the same fashion, suppose A is an application on a computer X, which wants to communicate with another application B on a remote computer Y. Now, the application A must know that it has to first reach computer Y, and then the application B on that computer. Why is just the destination computer's address (Y) not enough? This is because, at the same time, another application C on computer X might want to communicate with yet another application D on computer Y. If application A does not specify that it wants to communicate with application B, and instead just specifies that it wants to communicate with *some* application on computer Y, clearly, this would lead to chaos. How would, for example, computers X and Y know that application A wants to communicate with application B, and not D?

Figure 18.9 shows an example of using a port number in conjunction with an IP address. As shown in the figure, an application A running on computer X wants to communicate with another application B running on computer Y. Application A provides the IP address of computer Y

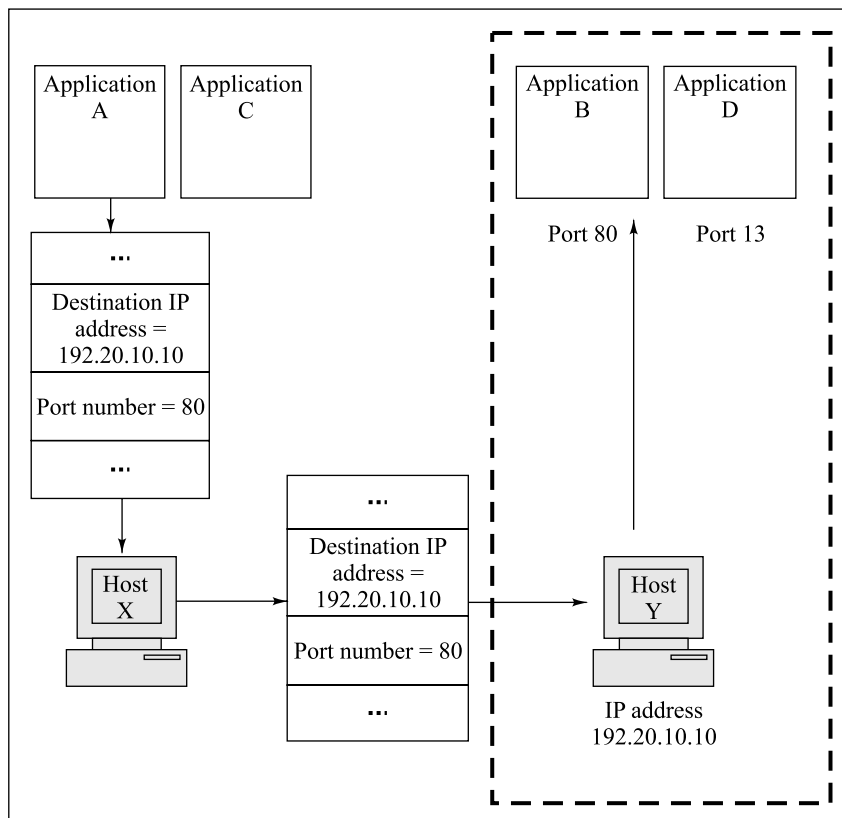


Fig. 18.9 Use of port numbers

(i.e., 192.20.10.10) and the port number corresponding to application B (i.e., 80) to computer X. Using the IP address, computer X contacts computer Y. At this point, computer Y uses the port number to redirect the request to application B.

This is the reason why, when an application wants to communicate with another application on a remote computer, it first opens a TCP connection (discussed in the next section) with the application on the remote computer using the IP address (like the telephone number) of the remote computer and the port number of the target application (like the extension of the person to speak with). Thus, the IP protocol enables communication between different two computers, whereas TCP enables the communication at a higher level than IP, i.e., between two applications on these different computers. This is shown in Fig. 18.10.

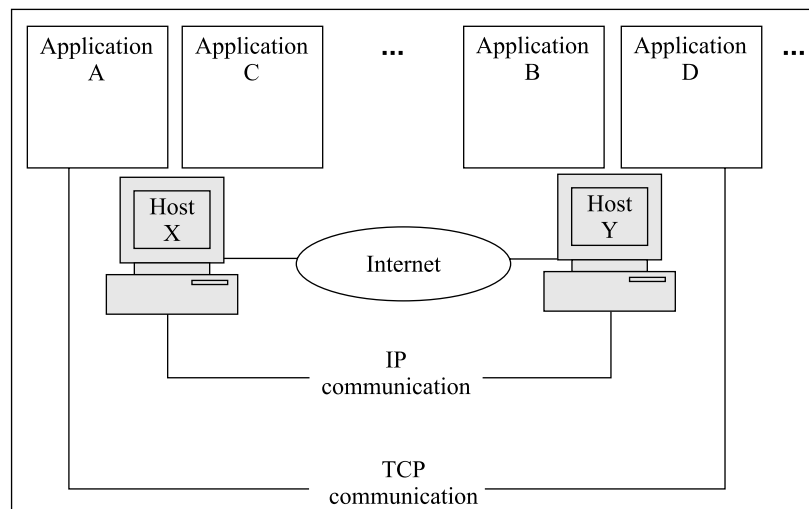


Fig. 18.10 Levels at which TCP and IP reside

As shown in the figure, application A on host X communicates with application B on host Y using the TCP protocol. As we shall study later, application A could be a *Web browser* on a *client's computer*, and application B could be a *Web server* serving the documents to the client's browser. This is the TCP perspective. From the IP protocol's perspective, however, which two applications on the two hosts are communicating is not significant. IP just knows that *some* applications on these two computers X and Y are communicating with each other. Thus, TCP enables application-to-application communication, whereas IP enables computer-to-computer communication.

However, if many applications on different computers are talking to many applications on other computers, on many links between the two nodes, it may be economical to combine (multiplex) the data between them and separate (demultiplex) at the appropriate nodes. This is achieved by the TCP software running on the various nodes.

18.4.2 Sockets

A port identifies a single application on a single computer. The term **socket address** or simply **socket** is used to identify the IP address and the port number concatenated together, as shown in Fig. 18.11.

For instance, port 80 on a computer 192.20.10.10 would be referred to as socket 192.20.10.10:80. Note that the port number is written after the IP address, with a colon separating them. This is shown in Fig. 18.12.

As we can imagine, **pair of sockets** identifies a TCP connection between two applications on two different hosts, because it specifies the end points of the connection in terms of IP addresses and port numbers, together. Thus, we have a unique combination of (Source IP address + Source port number + Destination IP address + Destination port number) to identify a TCP connection between any two hosts (typically a client and a server).

Normally, the server port numbers are called **well-known ports**. This is because a client (such as an Internet user) needs to know beforehand where (i.e., *on which port*) a particular application on the server is running. This would enable the client application to send a request to the server application (using the server's IP address and port number) to set up a TCP connection. If the client does not know whether the server is running its email software on port 100 or port 200, how can the client use either of them for requesting a connection?

It is similar to a situation where we have 1000 electrical plugs available (of which only one is working), and we do not know which one to use for plugging in a wire to glow a lamp. In the worst case, we might need to try 1000 times to establish the right connection! How nice it would be, instead, if someone says "Hey, use that plug number (in the case of TCP, *port number*) 723".

For this reason, at least the standard applications on a server are known to execute on specific ports, so that the client has no ambiguity while requesting for a TCP connection with the server. For instance, the HTTP protocol almost always runs on port 80 on the server, waiting to accept TCP connection requests from clients.

This is also the reason why multiple TCP connections between different applications, or even the same applications on the two hosts, are possible. Although the IP addresses of the two hosts would be the same in all the TCP connections, the port numbers would differ. When a client initiates a TCP connection, the TCP software on the client allocates an unused port number as the port number for this connection. Let us assume that this port number is 23. Further, let us assume that the client is sending an HTTP request (which means that the server's port number will be 80). Thus, from the client's point of view, the source port number is 23, and the destination port number is 80. When the server wants to send an HTTP response back to the client, the server shall now consider 80 as the source port number and 23 as the destination port number. This is shown in Fig. 18.13.

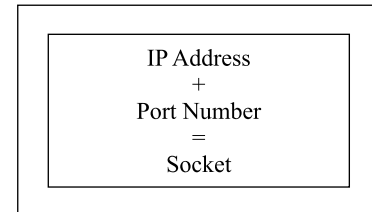


Fig. 18.11 Socket

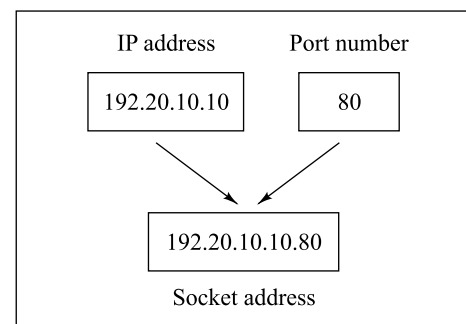


Fig. 18.12 Socket example

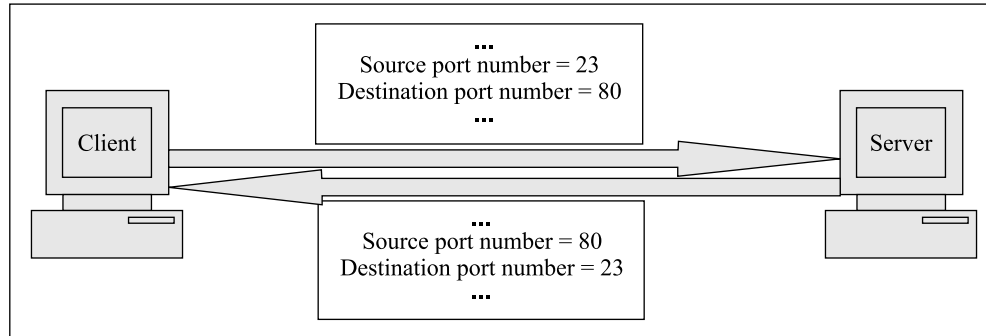


Fig. 18.13 Source and destination port numbers

18.5 CONNECTIONS – PASSIVE OPEN AND ACTIVE OPEN

Applications over the Internet communicate with each other using the connection (also called *virtual connection*) mechanism provided by TCP. We shall subsequently discuss how a TCP connection between two hosts is established and closed. These TCP connections are established and closed as and when required. That is, whenever an application X needs to communicate with another application Y on a different computer on the Internet, the application X requests the TCP software on its computer to establish a connection with the TCP software running on the computer where application Y is running. We have already discussed this in detail.

In this interaction model, one computer (called the *client*) always requests for a TCP connection to be established with the other (called the *server*). However, how does the server accept connections from one or more clients? For this, the TCP software on a server executes a process called **passive open**. In simple terms, this means that a server computer expects one or more clients to request it for establishing TCP connections with them. Therefore, the TCP software on a server waits endlessly for such connection requests from the clients. This is what passive open means. As a result of a passive open, a server computer allows TCP connection requests to be received, but not to be sent. This means that the server is interested only in accepting incoming TCP connection requests, and never in sending outgoing TCP connection requests. In contrast, a client always initiates a TCP connection request by sending such a request to the server. Therefore, the client is said to be in an **active open** mode.

Thus, when a publicly accessible server (e.g., a Web server) starts, it executes a passive open process, which means that it is ready to accept incoming TCP connections. A client (e.g., a Web browser) would then send an active open request for opening a TCP connection with that server. This is shown in Fig. 18.14. As shown in the figure, when a client needs to get a document from a remote server, the client issues an *active open* request to the local TCP software, providing it the IP address and TCP port of the destination server application. The TCP software on the client then uses this information to open a connection with the remote server.

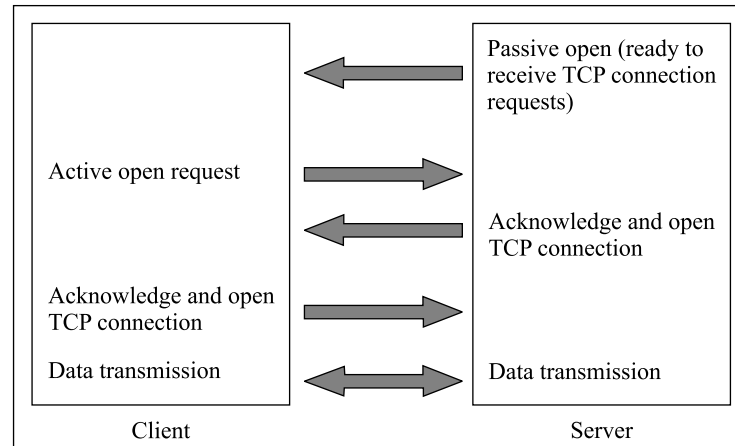


Fig. 18.14 Passive and active open processes

18.6 TCP CONNECTIONS

We have been saying that TCP is a connection-oriented protocol. Let us examine this statement in more detail now. To ensure that a connection is established between the two hosts that are interacting for a message transfer, TCP creates a logical connection between them. For this, TCP uses a technique called **three-way handshake**. This means that three messages are exchanged between the sender and the receiver of the message for establishing the connection. Only after this three-way handshake is successful that a connection between them is established. After this, they can exchange messages, assured that TCP would guarantee a reliable delivery of those. It has been proved that a three-way handshake is necessary and sufficient for establishing a successful connection between any two hosts. We shall not go into the details of this proof.

To be able to create a TCP connection between any two hosts, one of them must wait passively to accept active connection requests from the other host. In TCP/IP, it is the server who waits passively for connection requests. The client always initiates a connection request. Thus we can imagine that at a slightly lower level, the server would execute function calls such as *LISTEN* and *ACCEPT*. *LISTEN* would mean that the server is ready to listen to incoming TCP connection requests at a particular port, and *ACCEPT* would be invoked when a server is ready to accept a particular TCP connection request from a client, who has requested for it. On the other hand, the client would invoke a *CONNECT* request, specifying the IP address and port number of the server with which it wants to establish a TCP connection.

Conceptually, this is extremely similar to how telephone conversations between any two persons begin. This idea is shown in Fig.18.15.

If we replace the two humans speaking over the phone in the diagram with hosts using TCP virtual circuits for communication, we can easily understand the concepts such as *CONNECT*, *LISTEN* and *ACCEPT*.

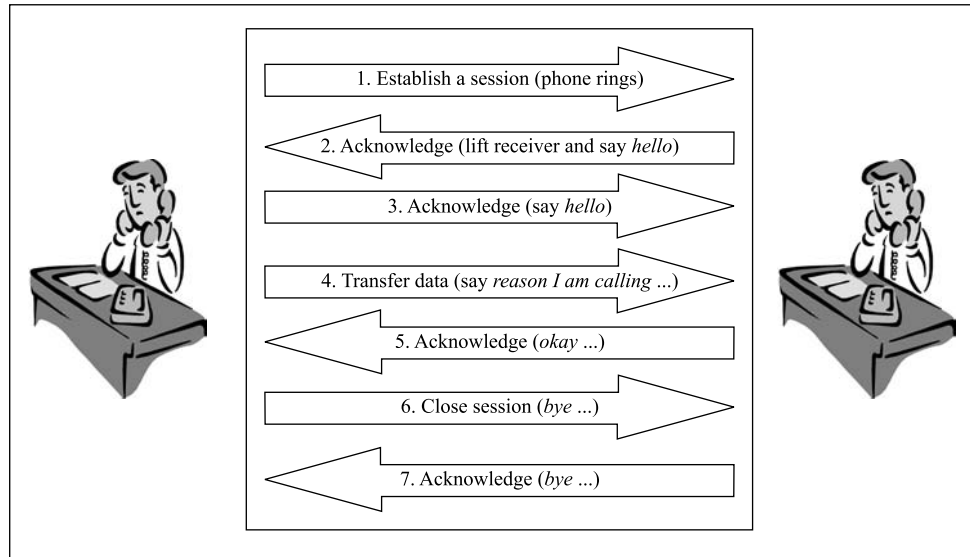


Fig. 18.15 Concept of a TCP connection with reference to a telephone call

18.7 WHAT MAKES TCP RELIABLE?

Before we understand how TCP achieves reliability, let us discuss why this is required in the first place. The main reason is that the underlying communication system is unreliable. This means that there is no guarantee that packets sent across the communication system would reach the ultimate destination, unless this is checked at the transport layer. Another problem is that the Internet layer has no means of checking duplicate packets and therefore, rejecting them.

For achieving reliability, the TCP software in the destination computer checks each packet upon arrival and sees if the same packet had arrived before. If so, it simply discards this duplicate packet. For detecting a packet loss, TCP employs the **acknowledgment** mechanism. In simple terms, whenever a packet arrives at a destination, the TCP software in the destination computer sends an acknowledgment back to the sending computer. If the sending computer does not receive this acknowledgment in a pre-specified time interval, it automatically resends the packet – a process called **packet retransmission**. For this, the sending computer sets a timer as soon as it sends a packet. If this timer expires before the acknowledgment arrives, the sending computer retransmits the packet. Let us discuss this in detail with an example as shown in Fig. 18.16.

Figure 18.16 shows how retransmission happens with the help of the acknowledgement mechanism. The sending computer sends a packet and waits for acknowledgement from the receiving computer in a stipulated time set using the timer. As can be seen in case of the second packet, the sending computer does not get any response from the receiving computer before the timer elapses. As a result, it retransmits the second packet. In case of packets 1 and 3, retransmission is not required because the acknowledgement from the receiving computer arrives at the sender's end before the timer expires.

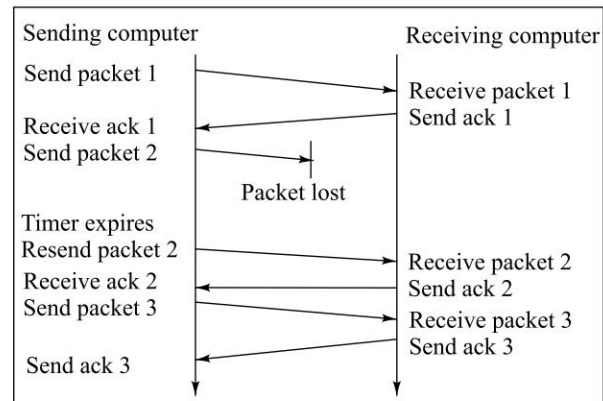


Fig. 18.16 Retransmission example

18.8 TCP PACKET FORMAT

Figure 18.17 shows the format of a TCP packet. A TCP packet, also called a **segment**, consists of a header of size 20 to 60 bytes, followed by the actual data. The header consists of 20 bytes, if the TCP packet does not contain any options. Otherwise, the header consists of 60 bytes. That is, a maximum of 40 bytes are reserved for options. Options can be used to convey additional information to the destination. However, we shall ignore them, as they are not very frequently used.

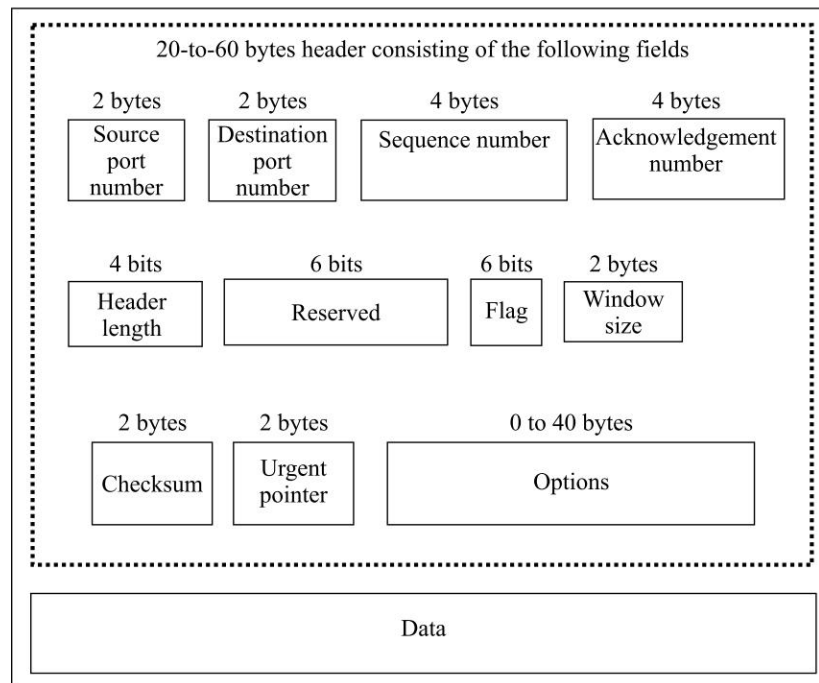


Fig. 18.17 TCP Packet format

Let us briefly discuss these header fields inside a TCP packet.

1. **Source port number** – This 2-byte number signifies the port number of the source computer, corresponding to the application that is sending this TCP packet.
2. **Destination port number** – This 2-byte number signifies the port number of the destination computer, corresponding to the application that is expected to receive this TCP packet.
3. **Sequence number** – This 4-byte field defines the number assigned to the first byte of the data portion contained in this TCP packet. As we know, TCP is a connection-oriented protocol. For ensuring continuous connectivity, each byte to be transmitted from the source to the destination is numbered in an increasing sequence. The sequence number field tells the destination host, which byte in this sequence comprises the first byte of the TCP packet. During the TCP connection establishment phase, the source as well as the destination generates different unique random numbers. For instance, if this random number is 3130 and the first TCP packet is carrying 2000 bytes of data, then the sequence number field for that packet would contain 3132 bytes (3130 and 3131 are used in connection establishment). The second segment would then have a sequence number of 5132 (3132 + 2000), and so on.
4. **Acknowledgement number** – If the destination host receives a packet with the sequence number X correctly, it sends $X + 1$ as the acknowledgement number back to the source. Thus, this 4-byte number defines the sequence number that the source of the TCP packet is expecting from the destination as a receipt of the correct delivery.
5. **Header length** – This 4-bit field specifies the number of four-byte words in the TCP header. As we know, the header length can be between 20 and 60 bytes. Therefore, the value of this field can be between 5 (because $5 \times 4 = 20$) and 15 (because $15 \times 4 = 60$).
6. **Reserved** – This 6-byte field is reserved for future and is not currently unused.
7. **Flag** – This 6-bit field defines six different control flags, each one of them occupying one bit. Out of the six flags, two are most important. The SYN flag indicates that the source wants to establish a connection with the destination. Therefore, this flag is used when a TCP connection is being established between two hosts. Similarly, the other flag of importance is the FIN flag. If the bit corresponding to this flag is set, then it means that the sender wants to terminate the current TCP connection.
8. **Window size** – This field determines the size of the sliding window that the other party must maintain.
9. **Checksum** – This 16-bit field contains the checksum for facilitating the error detection and correction.
10. **Urgent pointer** – This field is used in situations where some data in a TCP packet is more important or urgent than other data in the same TCP connection. However, a discussion of such situations is beyond the scope of the current text.

18.9 PERSISTENT TCP CONNECTIONS

A TCP connection is **non-persistent**. In simple terms, this means that a client requests for a TCP connection with the server. After server obliges, the client sends a request, the server sends a response, and closes the connection. If the client wants to make another request, it has to open a brand new TCP connection with the server. That is, the session does not *persist* beyond the lifetime of one request and one response.

There are situations when non-persistent connections are not desirable. As we shall study, on the Internet, a single Web page can contain text, images, audio and video. Suppose a client requests

a server for a Web page containing some text and two images. The images reside on the server as separate files, and therefore, normally, the client would have to open three *separate* TCP connections with the server for obtaining the complete Web page. This is shown in Fig. 18.18.

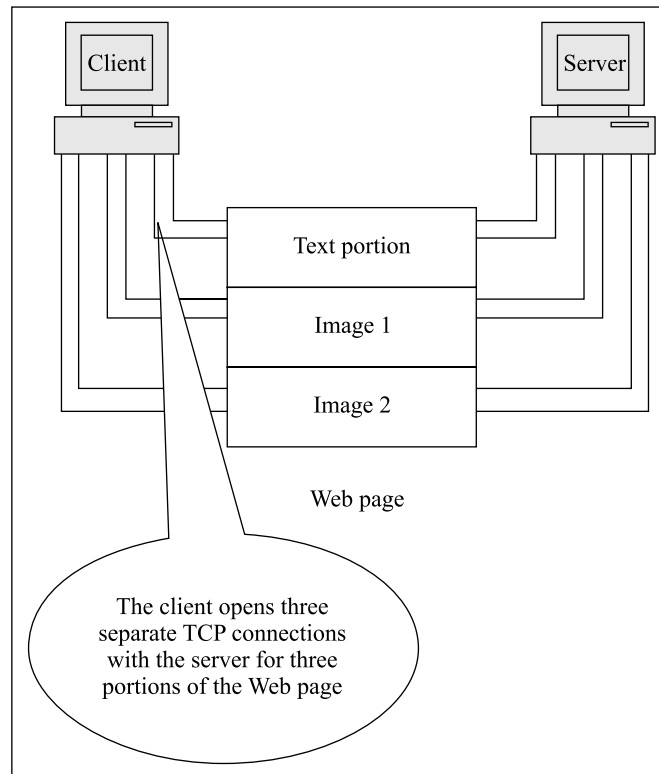


Fig. 18.18 Persistent connection

Instead, the new version of TCP/IP protocol suite allows for **persistent connections**. Here, the server serves the request, and does not terminate the connection. Instead, it keeps it open. Thus, the client can reuse the same connection for all forthcoming requests (such as images in a Web page). So, in our example, just one connection would suffice.

Persistent connections can be of two types. In persistent connections **without pipelining**, the client sends the next request within the same TCP connection to the server only after receiving a response to its previous request. In persistent connections **with pipelining**, the client can send multiple requests to the server within the same TCP connection without waiting for any responses from the server. Thus, multiple requests are possible in this case.

Persistent connections are expected to improve the throughput in case of Web pages that contain many non-textual data such as images, audio and video because a single TCP connection can suffice all portions of the Web page.

18.10 USER DATAGRAM PROTOCOL (UDP)

We know that the TCP/IP suite of protocols offers two protocols at the transport layer. The first is Transmission Control Protocol (TCP), which we have studied in detail. The other protocol in the transport layer is **User Datagram Protocol (UDP)**. We shall study UDP now. UDP is far simpler but less reliable than TCP.

UDP is a connectionless protocol, unlike TCP. UDP allows computers to send data without needing to establish a virtual connection. Since there is no error checking involved here, UDP is simpler than TCP. UDP does not provide for any acknowledgement, sequencing or reordering mechanisms. Thus, UDP packets may be lost, duplicated or arrive out of order at the destination. UDP contains very primitive means of error checking (such as checksums). The UDP packets are not numbered unlike TCP packets. Thus, even when multiple UDP packets are sent by the same source to the same destination, each packet is completely independent of all previous UDP packets. Since there is no connection between the sender and the destination, the path taken by a UDP packet cannot be predicted.

Thus, it is left to the application program that uses UDP to accept the full responsibility to handle issues such as reliability, including data loss, duplication, delay, and loss of connection. Clearly, this is not such a good idea for every application program to perform these checks, when a reliable data transport mechanism such as TCP is available. However, when the speed of delivery is more important than the reliability of data, UDP is preferred to TCP. For instance, in voice and video transmissions, it is all right to lose a few bits of information, than sending every bit correctly at the cost of transmission speed. In such a situation, UDP would be a better choice. In contrast, computer data transmission must be very reliable. Therefore, for transferring computer data such as files and messages, TCP is used.

18.11 UDP PACKET

Figure 18.19 shows the format of a UDP packet. UDP packets are also called **user datagrams**. Each UDP packet has a fixed 8-byte header that is subdivided into four fields, each of two bytes. This is followed by the actual data to be transmitted using this UDP packet, as the figure depicts.

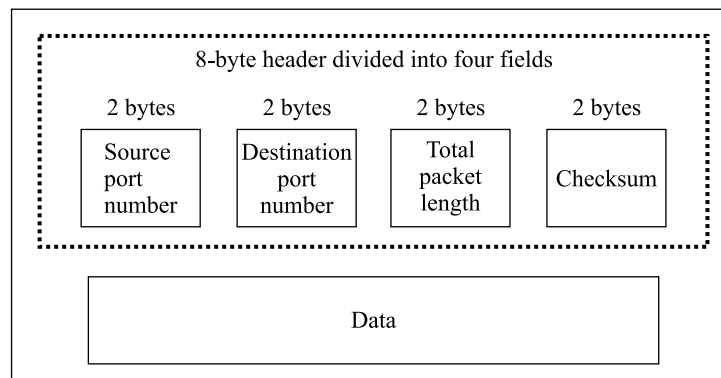


Fig. 18.19 *UDP packet format*

Let us briefly examine the fields in the UDP packet header.

1. **Source port number** – This is the port number corresponding to the application running on the source computer. Since it can take up to two bytes, it means that a source port number can be between 0 and 65,535.
2. **Destination port number** – This is the port number corresponding to the application running on the destination computer. Since it can also take up to two bytes, it means that a destination port number can also be between 0 and 65,535.
3. **Total Packet Length** – This 2-byte field defines the total length of the UDP packet, i.e., header plus data. Actually, this field is not required at all, because there is a similar *packet* length field in the IP packet, which encapsulates a UDP packet inside it before sending it to the destination. Therefore, the following equation is always true:

$$\text{UDP packet length} = \text{IP packet length} - \text{IP header length}$$
4. UDP packet length = IP packet length – IP header length
5. However, this field is retained in the UDP packet header as an additional check.
6. **Checksum** – This field is used for error detection and correction, as usual.

18.12 DIFFERENCES BETWEEN UDP AND TCP

We shall now understand the broad level difference between UDP and TCP with a simple example. Suppose three clients want to send some data to a server. Let us first understand how this can be done with the help of UDP. We shall then examine what happens in the case of TCP.

18.12.1 Using UDP for Data Transfer

In the case of UDP, a server is called **iterative**. It means that when a server is dealing with UDP requests, it processes only one request at a time. A client prepares a UDP request packet, encapsulates it inside an IP packet, and sends it to the server. The server processes the request, forms a UDP response packet and sends it back to the client. In the meanwhile, if more UDP requests arrive at the server, the server does not pay any attention to them. It completes servicing a UDP request before taking up any other UDP request. In order to ensure that the UDP requests received in the meantime are not lost, the server stores them in a queue of waiting UDP requests, and processes them one after the other. Note that the UDP requests could arrive from the same client or from different clients. In any case, they are strictly processed one after another in a sequence. Since UDP is connectionless, this is fine. This is shown in Fig. 18.20.

18.12.2 Using TCP for Data Transfer

In contrast to the UDP model, TCP works strictly on the basis of connections. That is, a connection (virtual connection) must be first established between a client and a server before they can send data to each other. As a result, if multiple clients attempt to use the same server at the same time, a separate connection is established for each client. Therefore, the server can process many client requests at the same time unlike what happens in UDP. For this reason, when using TCP, a server is said to be in **concurrent** mode. It means that a server can concurrently serve the requests of multiple clients at the same time, similar to the way a multiprogramming operating system executes many programs at the same time. A connection is established between the server and each client, and each such connection remains open until the entire data stream is processed.

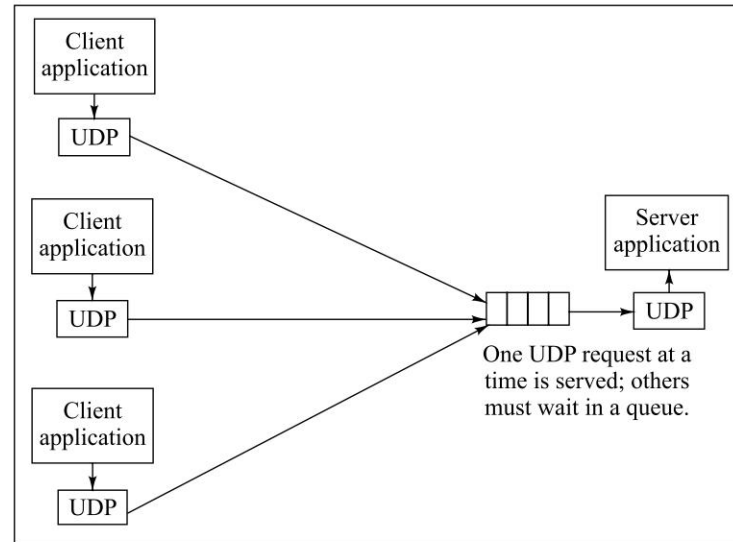


Fig. 18.20 UDP packets are queued

At the implementation level, the concept of parent and child processes is used. That is, when a request for a connection is received from a new client, the server creates a child process, and allocates it to the new client. The new client and the child server processes then communicate with each other. Thus, there is a separate connection between each client and a server child process. Once a parent server process creates a child process to serve the requests of a particular client, the parent process is free to accept more client requests, and create more child processes, as and when necessary. This is shown in Fig. 18.21.

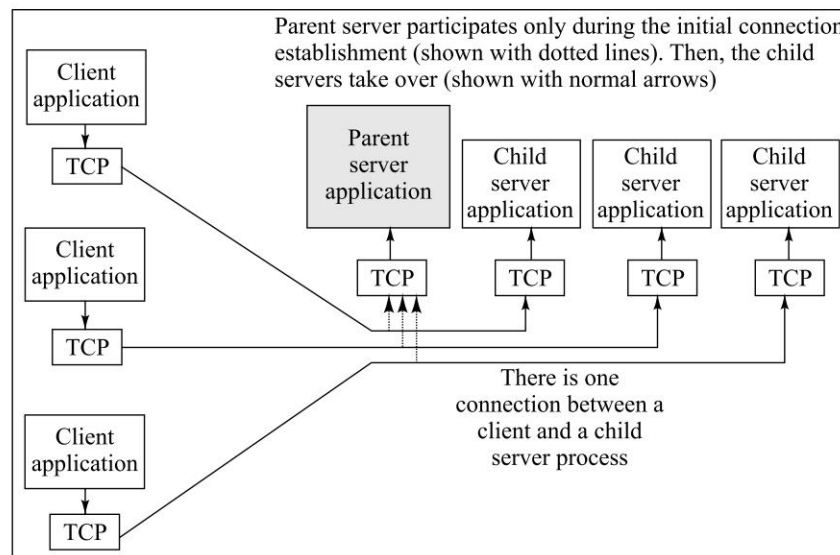


Fig. 18.21 TCP transmission

When the communication between a client and a server is over, the parent process kills that particular server child process.

SUMMARY

The Transmission Control Protocol (TCP) is one of the two transport layer protocols—the other is User Datagram Protocol (UDP). The main function of TCP is to ensure correct delivery of packets between the end-points. Since the lower layer protocol (IP) is a connectionless, best-effort delivery mechanism that does not worry about issues such as transmission impairments, loss of data and duplicate data, these features have been built into a higher layer protocol, i.e., TCP.

The main features offered by TCP are reliability, end-to-end communication and connection management. At the sender's end, TCP acts as a multiplexer that collects packets from a number of source applications. At the destination, TCP works as a demultiplexer by distributing the received packets to the appropriate destination applications.

To understand TCP, two concepts are crucial: port and socket. Applications running on different hosts communicate with TCP with the help of a concept called ports. A port is a 16-bit unique number allocated to a particular application. A socket is used to identify the IP address and the port number concatenated together. A pair of sockets identifies a unique TCP connection between two applications on two different hosts.

To allow connections, a server performs a passive open, thus waiting for connection requests. A client, on the other hand, sends a connection request to such a passively waiting server. The TCP software on the client and the server then establishes a connection between the two using a three-way handshake. A TCP connection is also closed using a three-way handshake.

TCP employs an acknowledgment mechanism to ensure correct delivery. If the sender does not receive an acknowledgement from the destination in a predetermined time interval, the sender resends the packet. TCP also uses the sliding window technique to ensure error control, so that the receiver is not overwhelmed with too many packets that it cannot accept/process. At the receiver's end, TCP reassembles the packets received, sequences them, removes duplicates if any, and constructs back the original message as was sent by the sender. A TCP packet contains a header consisting of 20 to 60 bytes. The rest of the packet contains actual data.

The User Datagram Protocol (UDP) is the simpler of the two protocols in the transport layer. Transmission Control Protocol (TCP) is a sophisticated protocol that provides a number of features for error control, flow control, accurate delivery, end-to-end communication, etc. UDP is a far simpler protocol that does not provide any of these features.

UDP is a connectionless protocol that does not create a virtual connection between the source and the destination. Instead, it delivers the packets as soon as they are received. UDP does not provide for any acknowledgement, sequencing or reordering mechanisms. Thus, if the transmission is in error, UDP would not be able to detect and therefore, repair it. The application program using the services of UDP must fulfill all these additional requirements.

In multimedia transmissions or voice transport, transmission speed is a major concern than accurate delivery of the message itself. The change of values of a few data bits is acceptable in such transmissions. UDP is a suitable candidate for such transmissions. UDP is never used for transmitting critical data.

A UDP packet contains an 8-byte header, followed by the actual data. Since UDP does not perform any error checking, it can do with such a small header. If multiple UDP packets arrive at a destination, all of them must wait in a queue before all their preceding packets are processed. Because of this, a UDP server is called iterative, which means that it strictly processes one packet at a time. TCP, on the other hand, is concurrent, because it processes multiple client requests simultaneously.

KEY TERMS AND CONCEPTS

Acknowledgement	Point-to-point communication
Active open	Port
Duplication control	Port-to-port communication
Concurrent	Segment
End-to-end delivery	Sequence control
Error control	Socket
Iterative	Three-way handshake
Loss control	Transport Layer Protocol (TCP)
Non-persistent connection	User datagram
Packet retransmission	User Datagram Protocol (UDP)
Pair of sockets	Well-known ports
Passive open	With pipelining
Persistent connection	Without pipelining

QUESTIONS

True/False

1. The transport layer runs on top of the Internet layer.
2. UDP guarantees a successful delivery of packets between two hosts.
3. TCP makes the Internet a reliable transport medium.
4. TCP acts as a multiplexer/demultiplexer.
5. A port is a 16-bit unique number allocated to a particular computer.
6. The term socket address or simply socket is used to identify the IP address and the port number concatenated together.
7. A server is always passive.
8. A client may or may not be active.
9. Opening and closing a TCP connection needs a three-way handshake.
10. The concept of *error control* defines the amount of data that a source can send before receiving an acknowledgement from the destination.
11. A TCP packet consists of a header of size 20 to 60 bytes.
12. The *SYN* flag is used when a TCP connection is being established or closed between two hosts.
13. UDP allows computers to send data without needing to establish a virtual connection.
14. The UDP packets are numbered.
15. The path taken by a UDP packet cannot be predicted.
16. When the speed of delivery is trivial than the reliability of data, UDP is preferred to TCP.
17. TCP is iterative.

Multiple-Choice Questions

1. Transport layer protocols are useful for ensuring _____ delivery.
 - (a) host to host
 - (b) host to router
 - (c) network to network
 - (d) end to end
2. _____ is a reliable delivery mechanism.
 - (a) IP
 - (b) TCP
 - (c) UDP
 - (d) ARP
3. Transport layer is _____ the data link layer.
 - (a) above
 - (b) below
 - (c) at the same layer as
 - (d) completely dependent on
4. When a packet is lost in transit, it should be handled by _____.
 - (a) sequence control
 - (b) error control
 - (c) loss control
 - (d) duplication control
5. When a single packet reaches the destination twice, it should be handled by _____.
 - (a) sequence control
 - (b) error control
 - (c) loss control
 - (d) duplication control
6. When packet 2 reaches packet 1 at the destination, it should be handled by _____.
 - (a) sequence control
 - (b) error control
 - (c) loss control
 - (d) duplication control
7. TCP uses the mechanism of _____.
 - (a) physical connections
 - (b) virtual connections
 - (c) circuit switching
 - (d) virtual circuits
8. Combination of _____ and _____ makes a socket.
 - (a) TCP address, IP address
 - (b) IP address, UDP address
 - (c) IP address, port number
 - (d) IP address, physical address
9. Well-known ports are generally required _____.
 - (a) only on the client
 - (b) on the client and the server
 - (c) on the client but not on the server
 - (d) on the server
10. The client does _____.
 - (a) active open
 - (b) passive open
 - (c) Both (a) and (b)
 - (d) None of the above
11. The mechanism of _____ is used if the acknowledgement is not received in a specified time interval.
 - (a) packet deletion
 - (b) packet retransmission
 - (c) packet holding
 - (d) packet caching
12. UDP is iterative because _____.
 - (a) it processes multiple requests at the same time
 - (b) it performs round-robin checks
 - (c) it processes only one request at a time
 - (d) it performs parallel processing
13. TCP is concurrent because _____.
 - (a) it processes multiple requests at the same time
 - (b) it performs round-robin checks
 - (c) it processes only one request at a time
 - (d) it performs parallel processing

Detailed Questions




1. Briefly discuss when to use TCP and when to use UDP.
2. Describe the broad-level features of TCP.
3. Discuss the idea of a port.
4. What is the difference between a port and a socket?
5. Why are sockets important?
6. Discuss the idea of *passive open* and *active open*.
7. How does the *three-way handshake* for creating a TCP connection work?
8. What factors make TCP reliable?
9. Describe in brief how sliding window works.
10. What is the purpose of the field *sequence number* inside a TCP packet header?
11. Describe the main fields of UDP packet header.
12. Why UDP is called iterative?
13. Explain why TCP is concurrent.
14. What is a persistent TCP connection? When is it useful?

TCP/IP

Part 3: DNS, Email, FTP and TFTP

19



19.0 INTRODUCTION

In the last few chapters, we have discussed various protocols in the network/Internet layer and the transport layer of the TCP/IP protocol suite. These protocols such as IP, ARP, TCP, and UDP are responsible for providing lower layer services such as delivery of packets from one host to another and optionally error-checking and retransmission, etc.

The network and transport layer protocols in TCP/IP would have no meaning without the application layer protocol services such as **Domain Name System (DNS)**, **Simple Mail Transfer Protocol (SMTP)** for **electronic mails (emails)**, **File Transfer Protocol (FTP)** and **Trivial File Transfer Protocol (TFTP)**. We shall study all these protocols that are akin to passengers in a transport system. Of course, these are not the only protocols in the application layer of TCP/IP suite of protocols. We shall discuss the remaining application layer protocols in TCP/IP later.

19.1 DOMAIN NAME SYSTEM (DNS)

19.1.1 Introduction

Although computers work at their best when dealing with numbers, humans feel quite at home, dealing with names, instead. For instance, you would certainly prefer if someone asks you to send a message to Jim's computer than telling you to send it to a computer whose IP address is 150.21.90.101 (Though this is much better than having to talk about an address as a string of 32 bits). Jim's computer might correspond to this IP address. But for you, it is far better to call it as Jim's computer, or even better, simply Jim! This simple idea of identifying computer networks and computers on those networks by some names is the basis for **domain names**.

A domain name is a name given to a network for ease of reference by humans. The term *domain* actually refers to a group of computers that are called by a single common name. Of course, somebody ultimately has to translate these domain names into IP addresses, because it is only these 32-bit IP addresses of computers that the TCP/IP or the Internet understands while sending or receiving any messages such as emails or files. This is conceptually shown in Fig. 19.1 where a computer's domain name is *John*, and its IP address is 150.21.90.101. The diagram shows the perspectives from both the point of views of a user and a network.

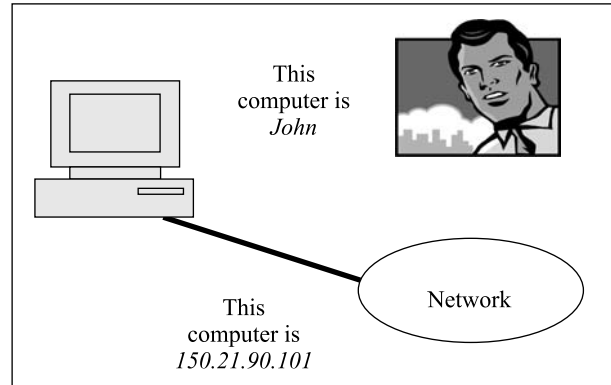


Fig. 19.1 Domain name and IP address are different representations of the same computer

People often name their computers with pride. In some organizations, naming computers is standardized. But in many other cases, computers are identified by the names of the people that use them, or by planet names. Jokingly, you could hear a comment such as *Barbara is down today*, which actually means Barbara's computer is not working for some reason. To summarize, we humans like to call computers by, well, names. There is only one problem here. Two computers in a network cannot have the same name. Otherwise, a computer cannot be identified uniquely. For this reason, it is necessary to ensure that the computers' names are always unique and that too globally, if we want to use them on the Internet.

In order to make computers' names unique, the Internet naming convention uses a simple and frequently used idea. Additional strings or suffixes are added to the names. The full name of a computer consists of its local name followed by a period and the organization's suffix. For example, if Diana works in IBM, her computer's name would be *Diana.IBM*. Of course, if there are two or more persons with the same name in an organization, another convention (e.g., *Diana1* and *Diana2* in this case) could be used.

We would realize that this is not good enough. The names of the organization themselves could be same or similar. For example, *john.techsystems* may not suffice, since there can be many organizations with the name *techsystems*. (Moreover, there could be many *Johns* in each of them). This means, having the organization's suffix to the local name is not adequate. As a result, another suffix is added to the computers' names after the organization's name. This indicates the type of the organization. For example, it could be a commercial organization, a non-profit making concern or a university. Depending on the type, this last suffix is added. Normally, this last suffix is three characters long. For example, *com* indicates a commercial organization, *edu* indicates a university and *net* indicates a network. As a result, Diana's computer would now become *Diana.IBM.com*. In general terms, all computers at IBM would have the last portion of their names as *IBM.com*. If an IBM university crops up tomorrow, it would not clash with *IBM.com*. Instead, it would become *IBM.edu*. This is shown in Fig. 19.2.

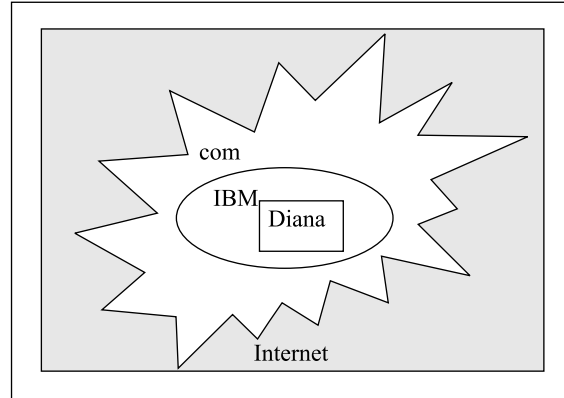


Fig. 19.2 Domain Name example

It must be mentioned that a computer's name on the Internet need not necessarily be made up of only three parts. Once the main portion of the name is allocated to an organization (e.g., IBM.com), the organization is free to add further subnames to computers. For instance, IBM's US division might choose to have a prefix of IBM-US.com or IBM.US.com to all their computers instead of IBM.com.

Initially, all domain names had to end with a three-character suffix such as *com* or *org*. However, as the Internet became more popular and widespread, people thought of adding country-specific prefixes to the domain names. These prefixes were two characters long. Examples of these suffixes are *in* for India, *uk* for England, *jp* for Japan and *de* for Germany. So, if the computer containing the information about the site (called **Web server**, which we shall study in detail later) were hosted in England, the prefix would not be *com*, instead it would be *co.uk*. For instance, BBC's site is *www.bbc.co.uk* and not *www.bbc.com*. Basically, it is decided on the physical location of the Web server as well as where the domain name is registered. However, any site in the US does not have a two-character suffix (such as *us*). All commercial domain names in the US end with *com* and not *co.us*. The reason for this is simple. The Internet was born in the US, and therefore, the US is taken as default. Remember that the country name is not written on the postal stamps in England, after all the postal system started there, and no one then thought that it would one day become so popular that all other countries would adopt it. In a similar way, people did not think that one day, **Web sites** (a term used to refer to the existence of an organization on the Internet, or the *Web*) would come up in so many different parts of the world. Therefore *com* meant US, at least, initially.

The generic domain names are shown in Fig. 19.3.

Domain name	Description
com	Commercial organization
edu	Educational institution
gov	Government institution
int	International organization
mil	Military group
net	Network support group
org	Non-profit organization

Fig. 19.3 Generic domain names

The proposed additional generic name labels are shown in Fig. 19.4.

Domain name	Description
arts	Cultural organization
firm	Business unit or firm
info	Information service provider
nom	Personal nomenclature
rec	Recreation or Entertainment group
store	Business offering goods/services
web	Web-related organization

Fig. 19.4 Proposed generic domain names

Thus, humans use domain names when referring to computers on the Internet, whereas computers work only with IP addresses, which are purely numeric. For instance, suppose while using the Internet, I want to send a message to my friend Pat who works in a company called Sunny Software Solutions. Therefore, there should be some screen on my computer that allows me to type `pat.sunny.com`. However, when I do so, for the Internet to understand this, clearly, there must be a mechanism to translate this computer name (*pat.sunny.com*) into its corresponding IP address (say, *120.10.71.93*). Only then, the correct computer can be contacted. This problem is shown in Fig. 19.5.

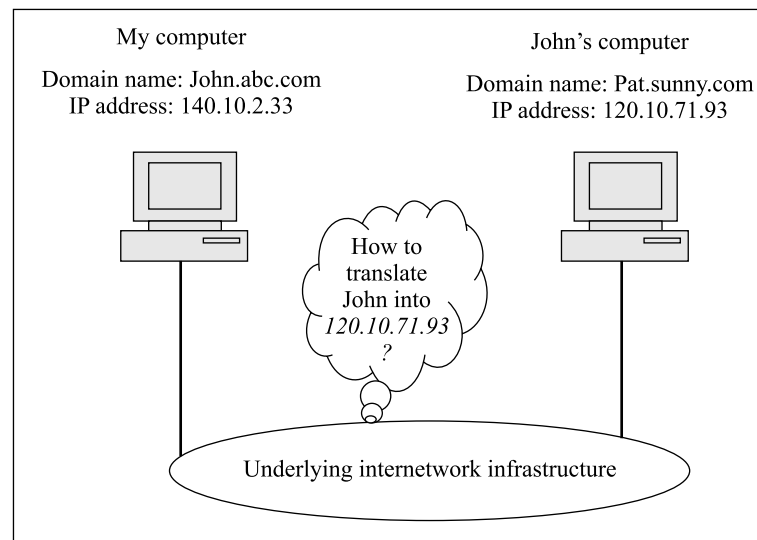


Fig. 19.5 How to translate domain names into IP addresses?


How is this achieved? We shall study this now.

19.1.2 Domain Name System (DNS)

In the early days of the Internet, all domain names (also called **host names**) and their associated IP addresses were recorded in a single file called *hosts.txt*. The Network Information Center (NIC)


in the US maintained this file. A portion of the hypothetical *hosts.txt* file is shown in Fig. 19.6 for conceptual understanding.

Host Name	IP address
John.abc.com	120.10.210.90
Pete.xyz.co.uk	131.90.120.71
Julie.pqr.com	171.92.10.89
...	...

Fig. 19.6  *Hosts.txt file – A logical view*

Every night, all the hosts attached to the Internet would obtain a copy of this file to refresh their domain name entries. As the Internet grew at a breathtaking pace, so did the size of this file. By the mid1980s, this file had become extremely huge. Therefore, it was now too large to copy it to all systems and almost impossible to keep it up to date. These problems of maintaining *hosts.txt* on a single server can be summarized as shown in Fig. 19.7.

Problem	Description
Traffic volumes	A single name server handling all domain name queries would make it very slow and lead to a lot of traffic from and to the single server.
Failure effects	If the single domain server fails, it would almost lead to the crash of the full Internet.
Delays	Since the centralized server might be <i>distant</i> for many clients (e.g., if it is located in the US, for someone making a request from New Zealand, it is too far). This would make the domain name requests-responses very slow.
Maintenance	Maintaining single file would be very difficult as new domain name entries keep coming, and some of the existing ones become obsolete. Also, controlling changes to this single file can become a nightmare.

Fig. 19.7  *Problems with a centralized domain name mechanism*

To solve this problem, the Internet **Domain Name System (DNS)** was developed as a **distributed database**. By distributed, we mean that the database containing the mapping between the domain names and IP addresses was scattered across different computers. This DNS is consulted whenever any message is to be sent to any computer on the Internet. It is simply a mapping of domain names versus IP addresses.

The DNS is based on the creation of a hierarchical domain-based naming architecture, which is implemented as a distributed database, as remarked earlier. In simple terms, it is used for mapping host names and email addresses to IP addresses.

Additionally, DNS allows the allocation of host names to be distributed amongst multiple naming authorities, rather than centralized at a single point, and also facilitates quicker retrievals. This makes the Internet a lot more democratic as compared to early days. We shall study this in the next section.

19.1.3 The DNS Name Space

Although the idea of assigning names to hosts seems novel and prudent, it is not an easy one to implement. Managing a pool of constantly changing names is not trivial. The postal system has to face a similar problem. It deals with it by requiring the sender to specify the country, state, city, street name and house number of the addressee. Using this hierarchy of information, distinguishing John Rosenberg of 13th North Avenue, New York, USA from the John Rosenberg of 13th North Avenue, Chelmsford, England, becomes easy. DNS uses the same principle.

The Internet is theoretically divided into hundreds of top-level domains. Each of these domains, in turn, has several hosts underneath. Also, each domain can be further subdivided into subdomains, which can be further classified into sub-subdomains, and so on. For instance, if you want to register a domain called *Honda* under the category *auto*, which is within *in* (for India); the full path for this domain would be *Honda.auto.in*. Similarly, from Fig. 19.8, it can be seen that *john.maths.oxford.edu* identifies the complete path for a computer under the domain *john*, which is under the domain *maths*, which is under the domain *oxford*, and which is finally under the domain *edu*.

This creates a tree-like structure as shown in Fig. 19.8. Note that a leaf represents a lowest-level domain that cannot be classified further (but contains hosts). The figure shows a hypothetical portion of the Internet.

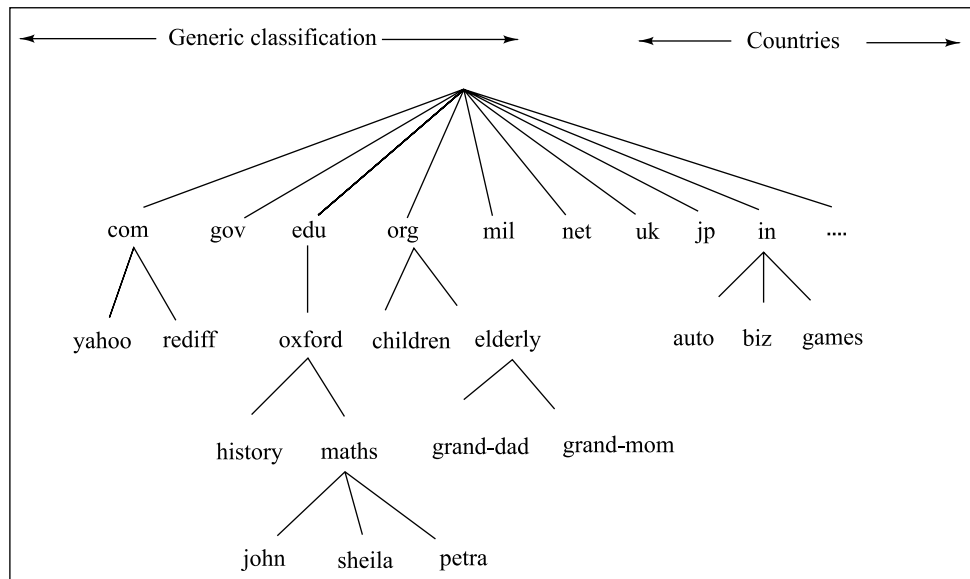


Fig. 19.8 A portion of the Internet's domain name space

The topmost domains are classified into two main categories, viz., **generic** (which means, the domains registered in the US) and **countries**. The generic domains are subclassified into categories such as **com** (commercial), **gov** (the US federal government), **edu** (educational), **org** (non-profit organizations), **mil** (the US military) and **net** (network providers). The country domains specify one entry for each country. For instance, **uk** (United Kingdom), **jp** (Japan), **in** (India) and so on.

Each domain is fully qualified by the path upward from it to the topmost (unnamed) root. The names within a full path are separated by a dot. Thus, Microsoft's Technology section could be named as `tech.microsoft.com`; whereas Sun Microsystems's downloads section could be named as `downloads.sun.com`. Domain names are case insensitive. Thus, `com` and `COM` is the same thing in a domain name. A full path name can be up to 255 characters long including the dots, and each component within it can be up to a maximum of 63 characters. Also, there could be many dots in a domain name within each component, which is in turn separated by dots.

19.1.4 DNS Server

Introduction

There is no doubt that we should have a central authority keeping track of the database of names in the topmost level domains such as `com`, `edu` and `net`. However, it is not prudent to centralize the database of all of the entries within the `com` domain. For example, IBM has hundreds of thousands of IP addresses and domain names. IBM would like to maintain its own **Domain Name System Server (DNS Server)**, also called just **Domain Name Server**, for the `IBM.com` domain. A domain name server is simply a computer that contains the database and the software for mapping between domain names and IP addresses. Similarly, India wants to govern the *in* top-level domain, and Australia wants to take care of the *au* domain, and so on. That is why DNS is a distributed database. IBM is totally responsible for maintaining the name server for `IBM.com`. It maintains the computers and the software (databases, etc.) that implement its portion of the DNS, and IBM can change the database for its own domain (*IBM.com*) whenever it wants to, simply because it owns its domain name servers. Similarly, for `IBM.au`, IBM can provide a cross-reference entry in its `IBM.com` domain, and take its responsibility.

Thus, every domain has a domain name server. It handles requests coming to computers owned by it and also maintains the various domain entries. This might surprise you. In fact, this is one of the most amazing facts about the Internet. The DNS is completely distributed throughout the world on millions of computers, and it is administered by millions of people, yet it appears to be a single, integrated worldwide database!

How the DNS Server Works

The DNS works in a manner similar to a telephone directory inquiry service. You dial up the inquiry service and ask for a person's telephone number, based on the name. If the person is local, the directory service immediately comes up with the answer. However, if the person happens to be staying in another state, the directory service either directs your call to that state's telephone directory inquiry service or asks you to call them. Furthermore, if the person is in another country, the directory service takes help of their international counterparts. This is very similar to the way a DNS server works. In case of the telephone directory service, you tell a person's name and ask for the telephone number. In case of DNS, you specify the domain name and ask for its corresponding IP address.

Basically, the DNS servers do two things tirelessly:

1. Accepting requests from programs for converting domain names into IP addresses
2. Accepting requests from other DNS servers to convert domain names into IP addresses

When such a request comes in, a DNS server has the following options:

1. It can supply the IP address because it already knows the IP address for the domain.
2. It can contact another DNS server and try to locate the IP address for the name requested. It may have to do this more than once. Every DNS server has an entry called *alternate DNS server*, which is the DNS server it should get in touch with for unresolved domains. The DNS hierarchy specifies how the chains between the various DNS servers should be established for this purpose. That discussion is beyond the scope of the current text.
3. It can simply say, “I do not know the IP address for the domain name you have requested, but here is the IP address for a name server that knows more than I do”. In other words, it suggests the name of another DNS server.
4. It can return an error message because the requested domain name is invalid or does not exist.

This is shown in Fig. 19.9.

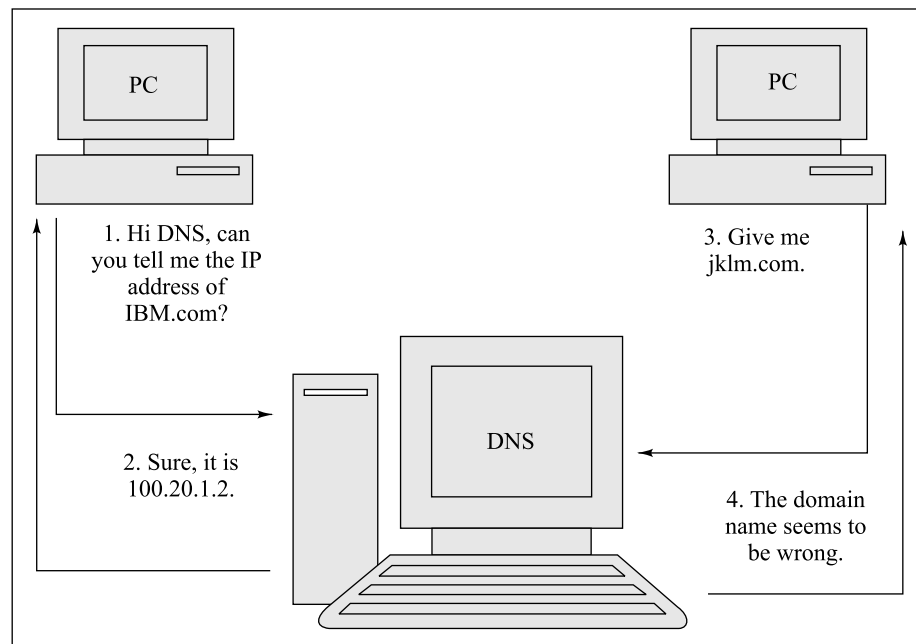


Fig. 19.9 Interactions between hosts and a DNS Server

As the figure shows, one host is interested in knowing the IP address of the server at *IBM.com*. For this purpose, it contacts its nearest DNS server. The DNS server looks at the list of domain names and their IP addresses. It finds an entry for the domain and sends it back to the client computer. However, when the DNS server receives another request from another computer for *jklm.com*, it replies back saying that such a domain name does not exist. As we know, for this, it may have to consult other DNS servers to see if they have any idea about this domain name or it may have to suggest the name of the DNS server that the host should contact itself.

For using DNS, an application program performs the following operations:

1. The application program interested in obtaining the IP address of another host on the Internet calls a library procedure called **resolver**, sending it the domain name for which the corresponding IP address is to be located. The resolver is an application program running on the host.
2. The resolver sends a UDP packet to the nearest DNS server (called the **local DNS server**).
3. The local DNS server looks up the domain name and returns the IP address to the resolver.
4. The resolver returns the IP address back to the calling application.

Using the IP address thus obtained, the calling application establishes a transport layer (TCP) connection with the destination or sends UDP packets, as appropriate. All this happens without the end user being aware of it. When you key in the domain name such as *honda.auto.com* to see its Web site, internally, the DNS is used to get the IP address and then the connection is established.

19.2 ELECTRONIC MAIL (EMAIL)

19.2.1 Introduction

Electronic mail (email) was created to allow two individuals to communicate using computers. In early days, the email technology allowed one person to type a message and then send it to another person over the Internet. It was like posting a card, except that the communication was electronic, instead of on paper.

These days, email facility allows many features such as the following:

1. Composing and sending/receiving a message
2. Storing/forwarding/deleting/replying to a message with normally expected facilities, such as carbon copy (CC), blind carbon copy (BCC), etc.
3. Sending a single message to more than one person
4. Sending text, voice, graphics, and video
5. Sending a message that interacts with other computer programs

The following are the best features of email:

1. The speed of email is almost equal to that of telephonic conversations.
2. The recording of the email messages in some form is like the postal system (which is even better than the telephone system).

Thus, email combines the best of the features of the telephone system and the postal system, and is yet very cheap. From the view point of users, email performs the following five functions:

1. **Composition** – The email system can provide features in addition to the basic text editor features such as automatic insertion of the receiver's email address when replying to a message.
2. **Transfer** – The email system takes upon itself the responsibility of moving the message from the sender to the receiver, by establishing connections between the two computers and transferring the message using TCP/IP.
3. **Reporting** – The sender needs to know whether the email message was successfully delivered to the receiver or it did not reach the receiver for whatever reason. The email system performs this reporting task as well.

4. **Displaying** – The email system displays the incoming messages in a special pop-up window, or informs the user in some way that an email message has arrived. The user can then open that message on the screen.
5. **Disposition** – This includes features such as forwarding, archiving, and deleting messages that have been dealt with. The user can decide what to do with such an email message, and instruct the email system accordingly.

There is a tremendous similarity between the postal system through which we send letters, and the email system. When we write a letter to someone, we put it in an envelope, write the intended recipient's name and the postal address on the envelope and drop it in a post box. The letter then goes via one or more interfaces, such as interstate or intercountry postal services. It also passes through various *nodes*, where sorting and forwarding of letters take place. Remember, the pin code comes handy for this. Finally, it arrives in the personal mailbox of the recipient. (Here, we imagine that each resident has a post mailbox near her/his house. We may also assume that the person checks the mailbox for any letters twice a day). This is shown in Fig. 19.10. Here, a person from New York wants to send a letter to her/his friend in Brighton (England).

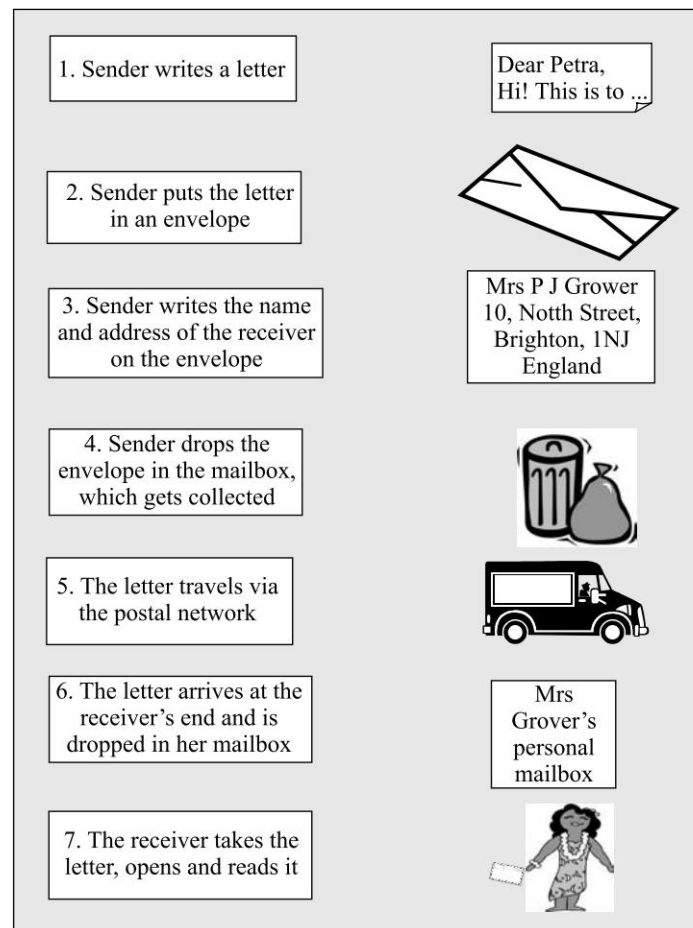


Fig. 19.10 Postal communication system used by humans

Email does not work a lot differently than this. The major difference between postal mail and email is the interface. The postal system has humans coordinating most of the communication (say, New York to London, and London to Brighton) in terms of moving the letter ahead, however, in case of email it is all handled by one or more intermediate routers, as studied earlier. Email uses TCP/IP as the underlying protocol. This means that when a person X writes a message to Y, it is broken down into packets according to the TCP/IP format, routed through various routers of the Internet and reassembled back into the complete email message at the destination before it is presented to Y for reading.

Interestingly, email first started with people sending files to each other. The convention followed was that when it was required to send an electronic message, the person would send a file instead, with the desired recipient's name written in the first line of the file. However, people soon discovered problems with this approach, some of which are as follows:

1. There was no provision for creating a message containing text, audio and video.
2. The sender did not receive any acknowledgement from the receiver, and therefore, did not know if the message indeed reached the receiver.
3. Sending the same message to a group of people was difficult through this approach. Examples of such situations are memos or meeting invitations sent to many people.
4. The user interface was poor. The user had to first invoke an editor, type the message into a file, close the editor, invoke the file transfer program, send the file, and close the file transfer program.
5. The messages did not have a predefined structure, making viewing or editing cumbersome.

Considering these problems, it was felt that a separate application was needed to handle electronic messaging.

19.2.2 The Mailbox

Just as we usually have our own personal mailbox outside the building for receiving postal mails, for receiving emails, we have an electronic **mailbox**. An email mailbox is just a storage area on the disk of the computer. This area is used for storing received emails. This is similar to the way a postal mailbox stores postal mails.

The postman arrives some time during the day to deliver postal mails. At that time, the recipient may not be at home. Therefore, the postman deposits the letters in the mailbox. We check our mailbox after returning home. Therefore, usually there is some gap between the time the mail is actually delivered in the box and the time it is actually opened. That is why this type of communication is called *asynchronous*, as opposed to the *synchronous* telephonic conversation, where the both parties are communicating at the same time. Similarly, when you write an email to somebody, that person may not have switched on her/his computer. Should this email reach that computer only to find that it cannot accept it? To solve this problem, another computer is given the responsibility of storing email messages before they are forwarded. This computer, along with the software is called **email server**. The email server is dedicated to her/his task of storing and distributing emails, but can, in theory, also perform other tasks. There is a mailbox (i.e., some disk space) on the email server computer for each client computer connected to it and wanting to use the email facility. That server has to be kept *on* constantly. When the user types in her/his email, it is sent from her/his computer to the email server of the sender, where it is stored first.

Similarly, all emails received for all the users connected to the email server are received and stored on this server first. The reason is that this email server is always *on*, even if the user (client) computers are shut *off*. When the client computer starts and connects to the server computer, the client can pick up the email from her/his mailbox on the server and either bring it onto her/his hard disk of the client PC, or just read it without bringing it to its own computer (i.e., download) and retain it or delete it. Thus, the user of the client computer can read all mails one by one and reply to them, delete them, forward them, etc. Therefore, in this regard, emails are similar to postal mails. They can be stored until the recipient wants to have a look at them. However, unlike postal mails, which takes days or even weeks to travel from the sender to the recipient, emails travel very fast, i.e., in a few minutes. In this aspect, emails are similar to telephone calls.

Thus, we will realize that there are two email servers that participate in any email communication as shown in Fig. 19.11.

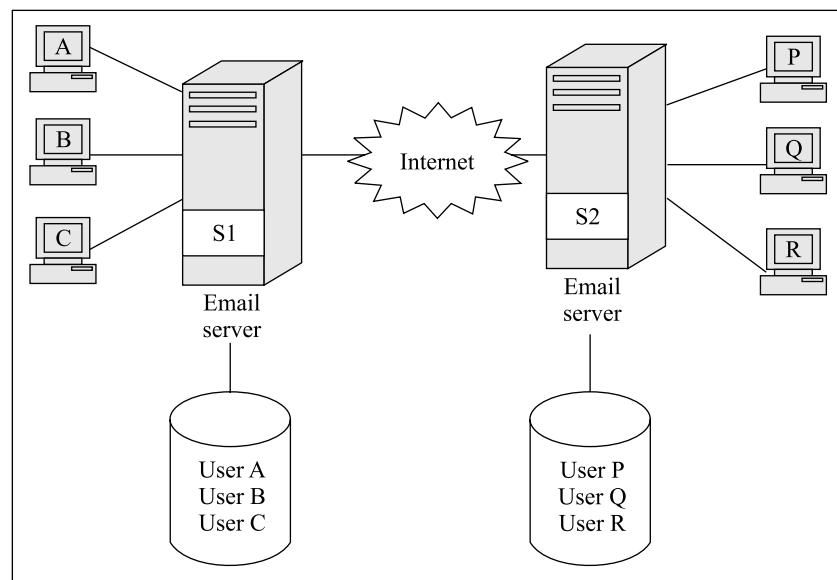


Fig. 19.11 Overview of the email system

When a user A wants to write an email to P, A creates a message on her/his PC and sends it. It is first stored on its email server (S1). From there, it travels through the Internet to the email server of P (i.e., S2). It is stored in the mailbox of P on the hard disk of S2. When P logs on, the PC is connected to her/his server (S2) and P is notified that there are new messages in her/his mailbox. P can then read them one by one, redirect them, delete them or transfer them to her/his local PC (i.e., download).

The email service provided by the Internet differs from other communication mechanisms in one more respect. This feature called **spooling**, allows a user to compose and send an email message even if her/his network is currently disconnected or the recipient is not currently connected to her/his end of the network. When an email message is sent, a copy of the email is placed in a storage area on the server's disk, called spool.

A spool is a queue of messages. The messages in a spool are sent on a *first come first searched* basis. That is, a background process on the email server periodically searches every message in a spool automatically after a specified time interval, and an attempt is made to send it to the intended recipient. For instance, the background process can attempt to send every message in a spool after every 30 seconds. If the message cannot be sent due to any reasons such as too many messages in the queue, the date and time when an attempt was made to send it is recorded. After a specified number of attempts or time interval, the message is removed from the spool and is returned back to the original sender with an appropriate error message. Until that time, the message remains in the spool. In other words, a message can be considered as delivered successfully only when both the client and the server conclude that the recipient has received the email message correctly. Till that time, copies of the email message are retained in both the sending spool and the receiving mailbox.

The postal system worldwide identifies the recipient using her/his unique postal address—usually some combination of city/zip code and street name and numbers, etc. In a similar fashion, an email is sent to a person using the person's **email address**. An email address is very similar to a postal address—it helps the email system to uniquely identify a particular recipient. We shall look at email addresses in more depth.

19.2.3 Sending and Receiving Email

The person sending a postal mail usually writes or types it and puts it in the envelope having the recipient's postal address. The software that enables the email system to run smoothly, i.e., the email software, has two parts. One that runs on the client (user's) PC called **email client software** and the other that runs on the email server, called **email server software**. For writing an email, the sender runs **email client software** on her/his computer. The email client software is a program that allows the user to compose an email and specify the intended recipient's email address. The composing part is very similar to simple word processing. It allows features such as simple text to be typed in, adjusting the spacing, paragraphs, margins, fonts and different ways of displaying characters (e.g., bold, italics, underlining, etc.). The email is composed using this software, which asks for the address of the recipient. The user then types it in. The email client software knows the sender's address anyway. Thus, a complete message with the sender's and the recipient's addresses is created and then sent across.

Using the recipient's email address, the email travels from the source to the email server of the source, and then to the recipient's email server; of course, through many routers. As we know, the underlying protocol used is again TCP/IP. That means that the bits in the contents of the email (text, image, etc.) are broken down into packets as per TCP/IP format and reassembled at the recipient's end. In between the nodes, the error/flow control and routing functions are performed as per the different protocols of different networks. The TCP/IP software running on the email server ensures the receipt of the complete email message. This server also has to have a part of the email server software, which manages the email boxes for different clients. After receiving the message, this software deposits it in the appropriate mailbox. When the recipient logs on to the server, the message is transferred to her/his computer. The recipient also has to have email client software application running on her/his computer. It is used to read the received email, and reply if necessary. The receiver can also forward the email received to other users of email anywhere on the Internet, or s/he can delete it. All this is done by using the email client software on the recipient's

computer. Obviously, as this software also allows replying to the message, it also has to have word processing capabilities.

Thus, the email software itself is divided into two parts, viz., client portion and server portion. The client portion allows you to compose a message, forward it, reply to a message, and also display a received message. The server portion essentially manages the mailbox to store the messages temporarily and deliver them when directed.

Each company normally installs an email server, using which, all the employees can communicate with each other, and also with the outside world. Alternatively, most of the ISPs provide the email service, which then have to take care of email server hardware and software. Apart from this, there are organizations like Yahoo, Hotmail, etc., who create a large pool of servers, with the server part of email software. Now, you can communicate with anyone freely on the Internet. Why does Yahoo do this? Because Yahoo feels that many people will subscribe to Yahoo, due to its free email service, and then while sending/receiving emails will also see the advertisements displayed. Yahoo, in turn, gets the money from the advertisers, who want to advertise on the Yahoo Web site, due to its large number of subscribers. It is exactly like a TV channel and their advertisements.

Having understood the basic concepts, let us look at the email message anatomy in more detail.

19.2.4 Email Anatomy

Here is a sample email message, as shown in Fig. 19.12.

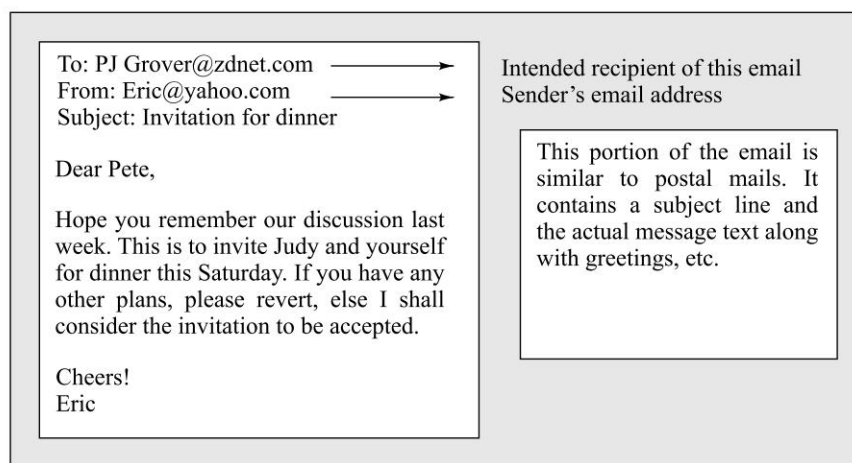


Fig. 19.12 *A sample email message*

Each electronic mailbox on the server has a unique email address. This consists of two parts, viz., the name of the user and the name of the domain. The @ symbol joins them to form the email address as shown in Fig. 19.13.

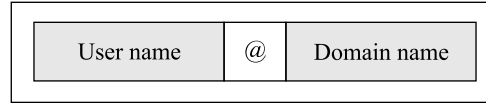


Fig. 19.13 *Email address format*

As we have seen before, the domain name usually identifies the organization or university of the user, like the street and city names. The user name is like the house number. For example, P J Grover works for an organization called zdnet in the above example (PJGrover@zdnet.com). Therefore, zdnet is the name of the organization and P J Grover is one of the users belonging to that organization. This is similar to writing the name of the person along with the house number and then the street name, city, etc., on the envelope. Note that the user name syntax is not very strict in many cases. If the email service is provided by an organization where the person is working (i.e., the email server hardware/software is hosted by the organization) itself, some standard is usually established (e.g., all email ids would be in the form name.surname@domainname). However, if the person subscribes to a free email service provider (such as Yahoo, Hotmail, USA.net, Rediffmail, etc.), s/he is free to choose the user name portion. Thus an email id can be as silly as shutup@hotmail.com, where shutup is a user name! This is possible because there are no naming standards in case of the email service providers.

There are a few organizations like AOL, who have set up their own network throughout the US. They have installed many email servers and routers, etc. You can become a member of AOL by applying to them and paying them some fees. You can then receive the email software required for the client machine (i.e., for sending/receiving messages having capabilities of word processing, forwarding, replying, etc.). You can now send an email to any other member of AOL family through the AOL network. But AOL initially was not a part of the Internet. In fact, it was competing against it in some sense. However, as the Internet grew in size, people from the AOL family wanted to send/receive messages to/from people on the Internet. If that was not possible, they started switching to the Internet. This is because the Internet had more number of users that you could reach out to. The problem was twofold. The AOL network was not connected to the Internet. The second problem was that the email formats and protocols followed within AOL and the Internet were quite different. Thus, only a physical connection would not have worked. In fact, many people were forced to have two separate accounts, i.e., one on AOL to communicate with people on that network, and the other for the Internet for people who were on the Internet. To avoid this, AOL itself got connected to the Internet where any message sent from anywhere on the AOL network could go through a gateway where the format was changed to suit the Internet and then it could reach anywhere on the Internet. What was true about AOL was true about many other networks, which have now got connected to the Internet.

Similarly, Yahoo is another network within the Internet with thousands of subscribers. All the people connected to Yahoo would have the domain name as yahoo.com. Thus, if the sender of our letter is Eric on the Yahoo domain, her/his full email address is Eric@yahoo.com. This is shown in Fig. 19.14.

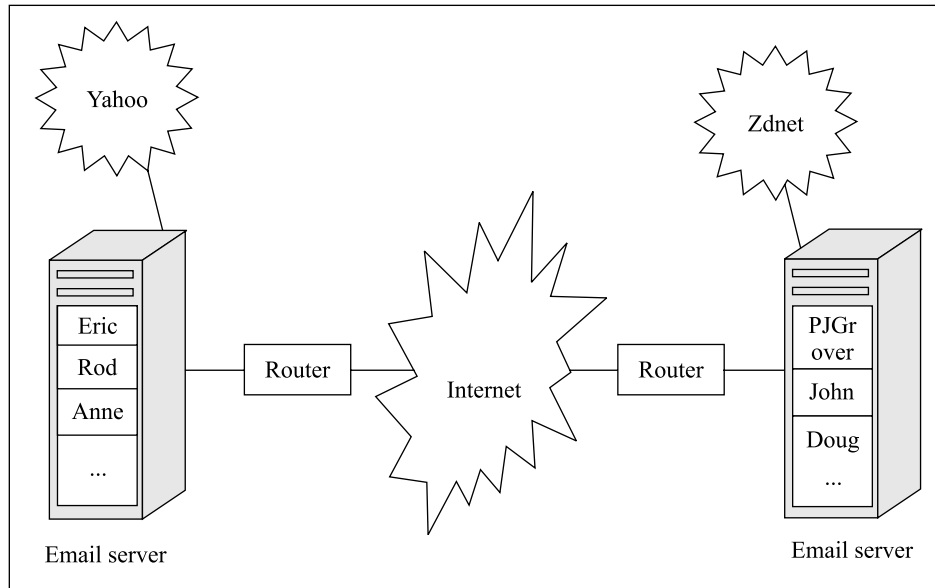


Fig. 19.14 *Concept of domains and email servers*

The corresponding block diagram for this is as shown in Fig. 19.15.

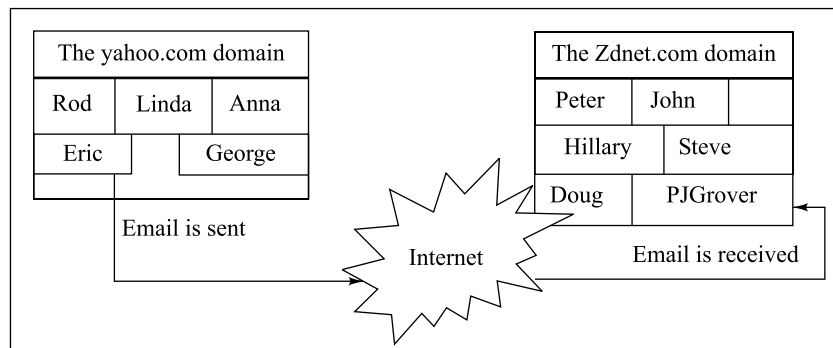
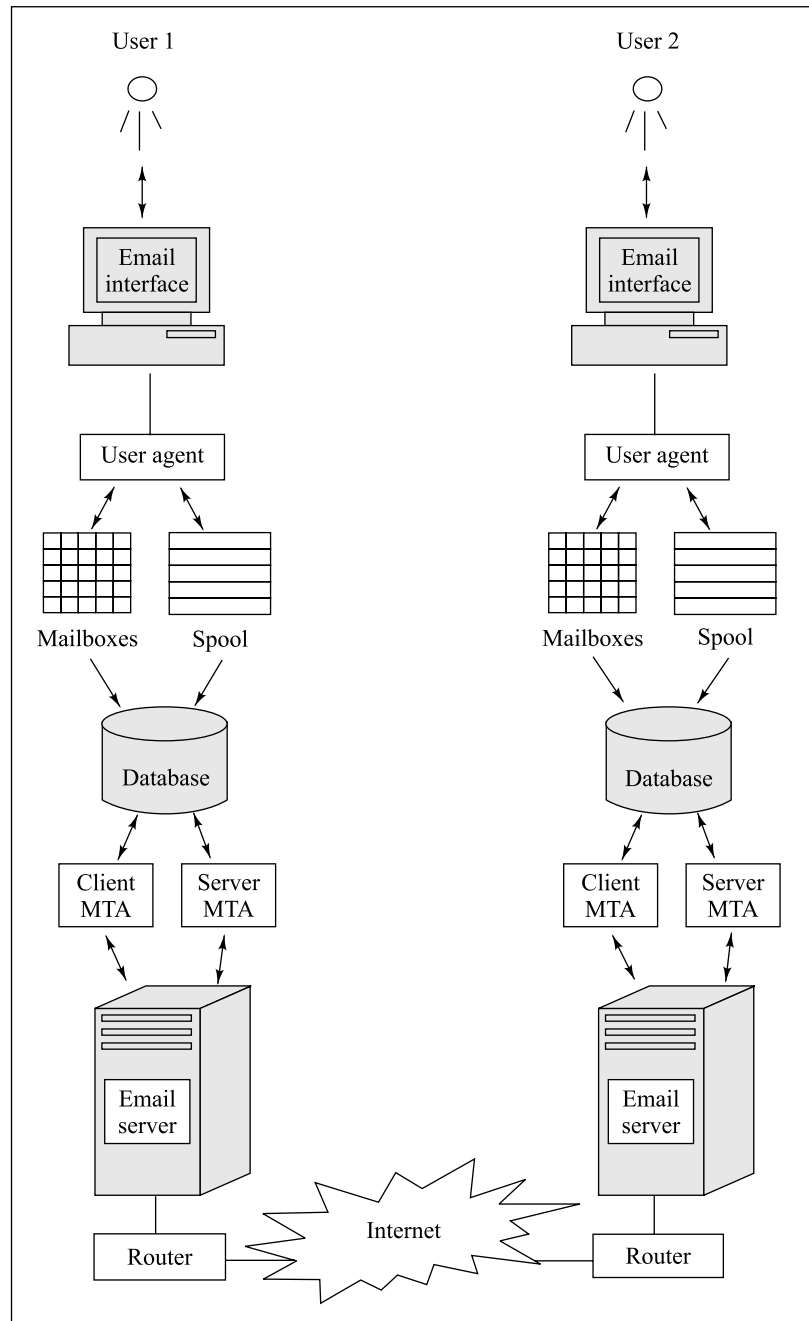


Fig. 19.15 *Domains and mailboxes on domains*

Several terms are used in the email technology. Let us understand them with the help of Fig. 19.16.

As shown in Fig. 19.16, there are many components of the email architecture, as briefly described below.

1. **User Agent (UA)** – The user agent is the user interface client email software (such as Microsoft Outlook Express, Lotus Notes, Netscape Mail, etc.) that provides the user facilitates for reading an email message by retrieving it from the server, composing an email message in a Word-processor like format, etc.

**Fig. 19.16** Email architecture

2. **Mailbox** – We have discussed mailboxes as well. There is one mailbox per user, which acts as the email storage system for that user.
3. **Spool** – We have already discussed spool. It allows storing of email messages sent by the user until they can be sent to the intended recipient.
4. **Mail Transfer Agent (MTA)** – The mail transfer agent is the interface between the email system and the local email server.

19.2.5 Email Transfer Protocols

For email messaging, every domain has an **email server** computer(s) set up, as we have seen. These email servers run protocol software that enable email communication. There are two main email protocols, viz., **POP** and **SMTP**. POP is concerned with the retrieval of an email message stored on a server computer, whereas SMTP is actually responsible for transmitting an email message between the sender and the recipient. Because both the email protocol software programs run on server computers, the server computers themselves are called **POP server** and **SMTP server**, respectively. Actually, a single server computer can host both SMTP and POP server programs. The two email server programs work in tandem, as we shall see.

POP Server

The **Post Office Protocol (POP)** provides a standard mechanism for retrieving emails from a remote server for a mail recipient. For instance, suppose that a home user X usually connects to the Internet using a dial-up connection to an ISP. Also suppose that another person Y has sent an email to X, when X is not connected to the Internet. Now, the email gets stored in the mailbox for user X provided by the ISP.

When X connects to the Internet next time and wants to see the new emails that have arrived for her/him since the last time s/he had connected to the Internet, s/he opens her/his email client program. That email client program on her/his computer in turn invokes a POP client, which contacts the POP server hosted by the ISP. The POP server then opens the mailbox for user X and sends the emails arrived for her/him to the POP client (i.e., to the user's computer).

In simple terms, any user who wants to receive emails but does not have a permanent connection to the Internet uses a POP client to pull emails from a POP server. The POP server consults the user's mailbox to perform this task. This is shown in Fig. 19.17. In the case of a company that provides an email access to all its employees, the SMTP and POP servers have to be hosted by the company itself, instead of the ISP.

Why is POP needed? We know that emails are stored on a centralized email server, i.e., the SMTP server. SMTP server expects the destination host (i.e., the email recipient) to be online all the time. Without this, it cannot create a TCP connection with it, and therefore, cannot forward the email message to the destination host. As we know, desktop computers are usually powered down when the business hours are over. Therefore, the solution to this problem is having a POP server. Whereas the SMTP server in an organization receives and stores all the incoming emails for any users in that organization, it is not used for transporting the emails to the end destination. Instead, the SMTP server forwards the incoming emails to the user's mailbox, and the POP server retrieves the emails from the respective email boxes of the users when requested for by the POP clients. When a user's POP client connects to the POP server, the POP server opens the mailbox for that user and sends the new emails to that user.

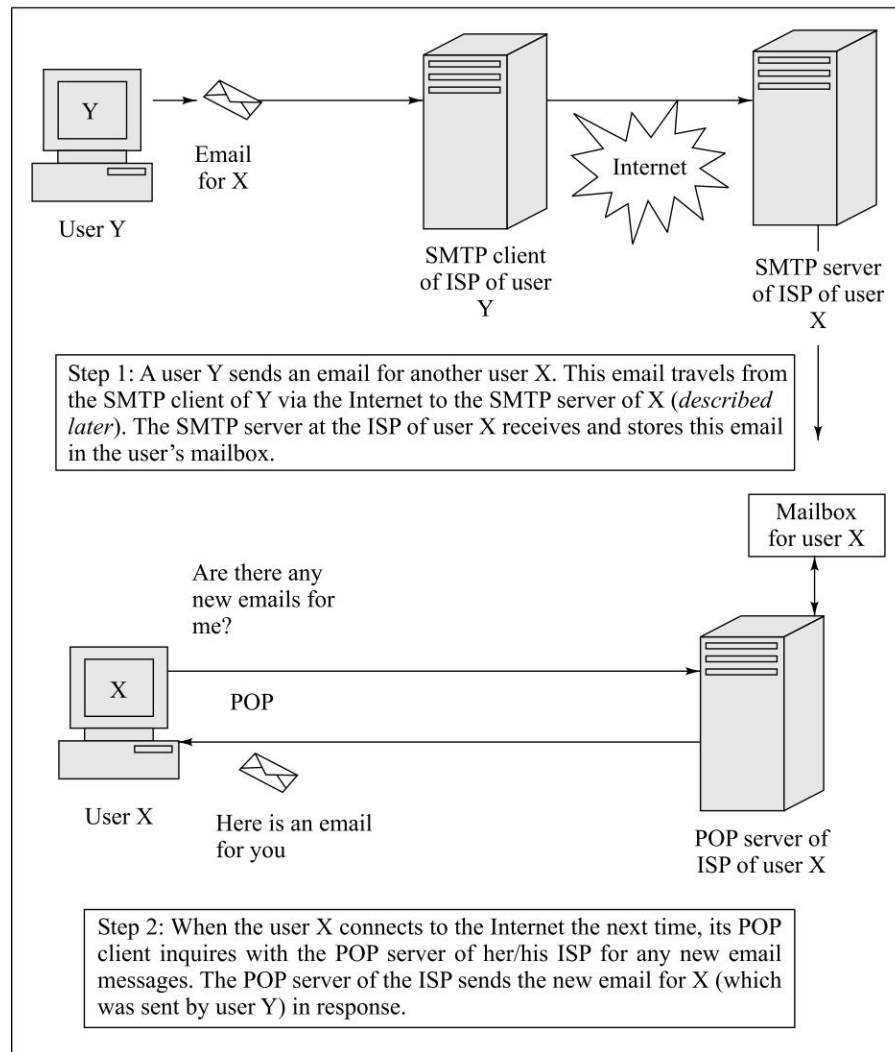


Fig. 19.17 SMTP and POP servers

In other words, it should not be expected that for a user to access her/his emails, either s/he should be connected to the Internet all the time or s/he would have to physically go where the SMTP server is kept and access her/his emails. To avoid this situation, the TCP/IP protocol suite includes a protocol, viz., POP, which allows a user's mailbox to be remotely accessed from the email server from the user's computer itself. The protocol provides a facility for a user's mailbox to reside on the email server and for accessing items in that mailbox from the user's computer using a connection between the user's computer (the client) and the email server. Of course, the user's email software is responsible for connecting to the POP server and for requesting unread mails in the first place. Based on this request, the POP server performs the operations described earlier.

For forwarding an email message to the actual recipient, a POP server simply maintains a collection of text files; one per user. The text file contains all the email messages for that user, one after the other. Therefore, when an email arrives for a user, it simply appends the email at the end of the text file for that user. This text file for a user is called mailbox. When a user (client) connects to the POP server, the POP server looks at the user name and then opens the corresponding text file for that user (which contains new and therefore, unread emails for that user). It then sends the contents of this text file, which are actually new email messages, to that user. The current version of POP is 3, and therefore, it is sometimes also called **POP3**.

SMTP Server

The **Simple Mail Transfer Protocol (SMTP)** actually transfers the email message from the SMTP server of the sender to the SMTP server of the recipient. Its main job is to carry the email message between the sender and the recipient. Of course, it uses the TCP/IP protocol underneath. That is, SMTP runs on top of TCP/IP (in the application layer). What it does is quite simple, as explained below.

1. At the sender's end, the SMTP server takes the message sent by a user's computer.
2. The SMTP server at the sender's end then transfers the message to the SMTP server of the recipient.
3. The SMTP server at the recipient's end then takes the email message and gives it to the POP server at the recipient's end.

SMTP is actually quite simple. The communication between a client and server using SMTP consists of human-understandable ASCII text. We shall first describe the steps and then list the actual iteration.

1. The process starts with the client sending a TCP connection request to the server.
2. The server sends back a *READY FOR MAIL* reply, indicating that it can accept an email message from the client.
3. The client then sends a *HELO* (abbreviation of *HELLO*) command to the server, and identifies itself.
4. The server then sends back an acknowledgement in the form of its own DNS name.
5. The client can now send one or more email messages to the server. The email transfer begins with a *MAIL* command that identifies the sender.
6. The recipient allocates buffers to store the incoming email message, and sends back an *OK* response to the client. The server also sends back a return code of 250, which essentially means *OK*. The reason both *OK* and a return code of 250 are sent back is to help both humans and application programs understand the server's intentions (humans prefer *OK*, application programs prefer a return code such as 250).
7. The client now sends the list of the intended recipients of the email message by one or more *RCPT* commands (one per recipient). The server must send back a *250 OK* or *550 No such user here* reply back to the client for each recipient.
8. After all *RCPT* commands, the client sends a *DATA* command, informing the server that the client is ready to start transmission of the email message.
9. The server responds back with a *354 Start mail input* message, indicating that it is ready to accept the email message. It also tells the client what identifiers it should send to signify that the message is over.


10. The client sends the email message and when it is over, sends the identifier provided by the server to indicate that its transmission is over.
11. The server sends back a *250 OK* response.
12. The client sends a *QUIT* command to the server.
13. The server sends back a *221 Service closing transmission channel* message, indicating that it is also closing its portion of the connection.

The actual interaction between the SMTP client and SMTP server as described above is shown in Fig. 19.18. Here, a user John at host yahoo.com sends an email to users Linda and Anna at host excite.com. The SMTP client software on the host yahoo.com establishes a TCP connection with the SMTP server software on the host excite.com (not shown in the figure). Thereafter, the exchange of messages happens as shown below (S indicates messages sent by the server to the client, and C indicates messages from the client to the server). Also note that the server has informed the client that it would like to see a <CR><LF><LF> identifier when the client's transmission is over. Accordingly, the client sends this when its transmission is over.

```

S: 220 excite.com Simple Mail Transfer Service Ready
C: HELO yahoo.com
S: 250 excite.com
C: MAIL FROM: <john@yahoo.com>
S: 250 OK
C: RCPT TO: <linda@excite.com>
S: 250 OK
C: RCPT TO: <anna@excite.com>
S: 250 OK
C: DATA
S: 354 Start mail input; end with <CR><LF><LF>
C: ... actual contents of the message ...
C: .....
C: .....
C: <CR><LF><LF>
S: 250 OK
C: QUIT
S: 221 excite.com Service closing transmission channel

```

Fig. 19.18  Example of an email message between the SMTP client and the SMTP server

Differences between SMTP and POP

Although both SMTP and POP servers communicate over the Internet, there are several differences between them, as follows:

1. The protocols themselves are different (SMTP and POP).
2. SMTP accepts a message from an arbitrary sender; whereas POP server allows only a user to access her/his mailbox after the user authenticates himself (using a user id and password).
3. SMTP server can transfer only email messages, whereas a POP server can also provide information about the mailbox contents (e.g., the number of unread mails, a list of sent mails etc.).

The Complete Journey of an Email Message

Let us now take a look at the complete journey of an email message. Combining the functionalities of the two email servers, we can draw Fig. 19.19.

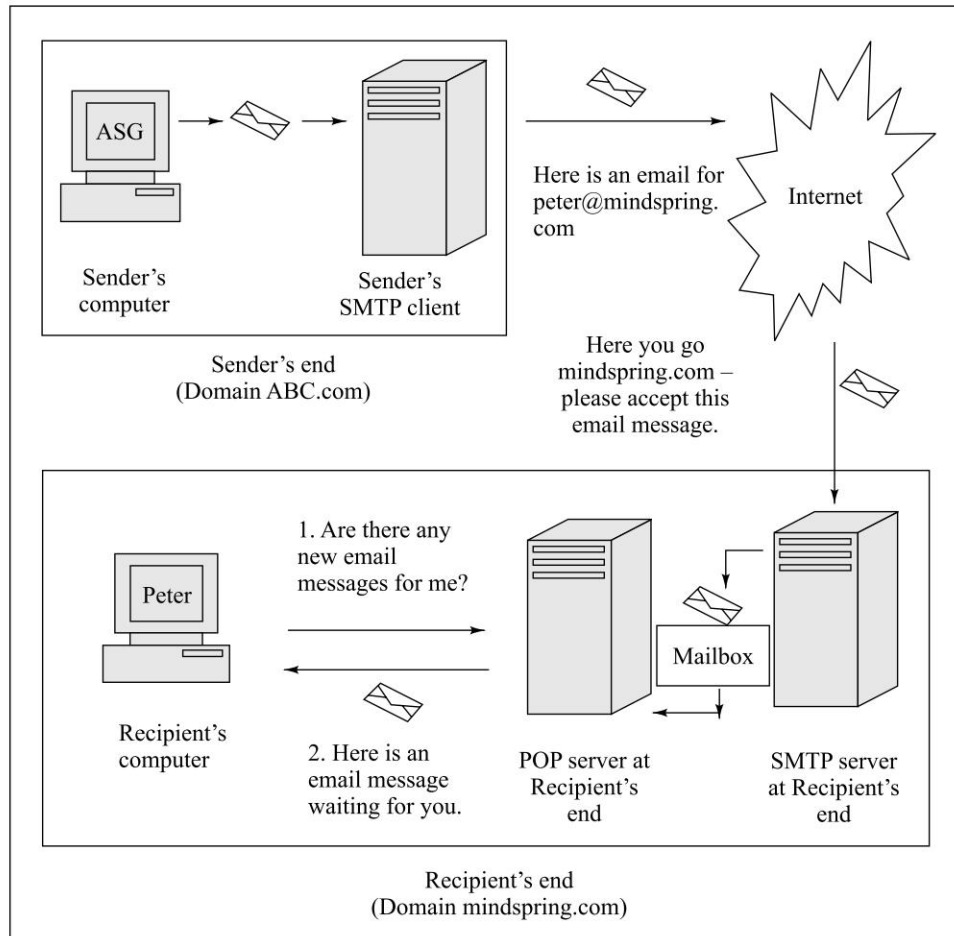


Fig. 19.19 Complete journey of an email message

Let us understand the whole process step by step.

1. Suppose I have an email account named ASG on the domain ABC.com. Therefore, my email account is ASG@ABC.com. I want to send an email message to Peter, whose email id is peter@mindspring.com. I compose the email message using an email client program on my computer and press the *send* button provided by the email program.
2. The SMTP client at ABC.com takes the recipient's email address and makes two parts of it, as (a) The recipient's name (peter) and (b) The domain name (mindspring.com). If the recipient were in the same domain as the sender, the recipient's domain name would also have been ABC.com. In that case, the SMTP client at ABC.com would simply hand the message over to

the SMTP server at ABC.com. But in this case, since the recipient's domain is not ABC.com, the SMTP client has to communicate with a different domain (that is, mindspring.com).

3. Using the DNS mechanism, the SMTP client at ABC.com requests for the IP address corresponding to the domain name mindspring.com. Either the DNS server knows it and therefore, does not need to make any further requests or it requests for it with other DNS servers, and finally obtains the IP address.
4. Having obtained that IP address, the SMTP client at ABC.com now connects with the SMTP server at mindspring.com using TCP/IP protocols. The port number for SMTP is predetermined. Therefore, now a TCP connection can be established.
5. Once the connection is established, the SMTP client at ABC.com sends the email message to the SMTP server at mindspring.com using TCP/IP (i.e., breaking the message into smaller packets, routing them, etc.). After the message is transferred, the SMTP client and the SMTP server break the connection. As we know, this is a virtual TCP connection. The message is sent as IP datagrams.
6. The SMTP sever at mindspring.com now stores the email message in the mailbox designated for Peter. This mailbox can be accessed by both SMTP as well as POP servers, as we have seen before.
7. When Peter connects to the POP server at his domain the next time, using the POP client inside its email software, it asks the POP server if there are any new email messages for him. In response, the POP server sends a copy of this email message to her/his computer. Thus, the email reaches the ultimate destination.

This scheme offers the following main benefit: The sender as well as the recipient need not be connected to the Internet all the time. The sender can compose the email message without needing to connect to the Internet. Only when the sender actually wants to send the message that s/he needs to connect to the Internet. The same thing happens with the recipient. The recipient connects to the Internet only for receiving email messages using POP. Once the emails are downloaded onto her/his computer from the POP server, s/he can disconnect. Since the emails are already on her/his computer, s/he need not contact the POP server for reading them.

19.2.6 IMAP Protocol

Usually, after downloading mails to her/his computer using POP, the user moves the email to an email folder depending on the subject or the sender (e.g., there could be one folder for all emails from friends, another folder for all emails containing jokes, etc). However, there are situations when the user wants these folders to be created on the server. Why is this required? Remember that POP downloads emails to the user's computer. Now suppose a user uses POP in the office to retrieve emails. What would happen if the user now wants to access these emails from home? Since the emails would be downloaded on the user's computer in the office, they would have been removed from the server. The user would not be able to access them from home even if s/he has a POP client installed at home.

To deal with such situations, the **Internet Mail Access Protocol (IMAP)** was introduced. IMAP allows creation of folders on a remote IMAP server, and allows them to be accessed from any IMAP client from anywhere. Clearly, IMAP server has to maintain folders/subfolders for all IMAP users. Therefore, it is more complex than POP.

To summarize, although the basic philosophy of IMAP is the same as that of POP, POP involves downloading of messages on to the client computer, whereas IMAP allows retention and manipulation of messages on the server.

19.2.7 Browser-based Emails

Many Web sites such as Hotmail and Yahoo among others allow users to access emails using a Web browser. These are called **browser-based emails**. Here, the *user agent* is a Web browser. Also, HTTP (and not POP/IMAP) is used for interaction between the browser and the email server (such as Hotmail or Yahoo). That is, when a user wants to send or view the received emails, the browser sends all such commands in the form of HTTP requests. Furthermore, when a Hotmail/Yahoo user sends an email, it travels over HTTP to the Hotmail/Yahoo server. From Hotmail/Yahoo server to the email recipient's email server, however, SMTP protocol is used as usual for email transfer.


Conceptually, the interaction between the browser and the Hotmail/Yahoo server is similar to IMAP. However, the underlying protocol is different (HTTP and not IMAP).

19.2.8 Multipurpose Internet Mail Extensions (MIME)

Traditional email systems are text-based, which means that a person can compose a text message using an editor and then send it over the Internet to another recipient. However, in the modern era, exchanging only text messages is not quite sufficient. People want to exchange multimedia files, documents in various arbitrary formats, etc. To cater to these needs, the **Multipurpose Internet Mail Extensions (MIME)** system extends the basic email system by permitting users to send binary files using the basic email system. A MIME email message contains a normal Internet text message along with some special headers and formatted sections of text. Each such section can hold an ASCII-encoded portion of data. Each section starts with an explanation as to how the data that follows should be interpreted or decoded at the recipient's end. The recipient's email system uses this explanation to decode the data.

Let us consider a simple example containing MIME header. Figure 19.20 shows an email message (after the sender has composed the email message and has attached a graphics file having .GIF extension). The figure shows the email message as it actually travels to the recipient. The content-Type MIME header shows the fact that the sender has attached a .GIF file to this message. The actual image would be sent as a part of the message after this, and would appear gibberish when viewed in the text form (because it would be the binary representation of the image). However, the recipient's email system shall recognize that this is a .GIF file, and as such, it should invoke an appropriate program that can read, interpret and display contents of a .GIF file.

```
From: Achyut Godbole <agodbole_2000@yahoo.com>  
To: Atul Kahate <akahate@indiatimes.com>  
Subject: Cover image for the book  
MIME-Version: 1.0  
Content-Type: image/gif  
  
<Actual image data in the binary form such as R019a0asdjas0 ...>
```

Fig. 19.20  *MIME Extensions to an email message*

One danger introduced by MIME is email viruses. Since before MIME, an email message could contain only ASCII text data, there was no scope for viruses being transmitted via emails. However, because anyone can send any binary file using MIME with email; it is quite possible that the binary file contains viruses. To prevent such a possibility or at least to reduce its impact, usually all modern anti-virus software programs automatically check incoming email attachments for existence of viruses, and either clean them and then open the attachment or directly delete the virus-infected attachment, and inform the user accordingly. Since a lot of information and messages are exchanged using emails, these days, viruses propagate via emails more than any other sources (such as floppy disks).

19.2.9 Email Privacy

As we have discussed, when one person sends an email to another, the email message can potentially travel through a number of intermediate routers and networks before it reaches the recipient. Consequently, there is a concern among email users about its privacy. What if the email message gets trapped on its way and is read by an unintended recipient? To resolve this issue, two main schemes are in use, viz., **Pretty Good Privacy (PGP)** and **Privacy Enhanced Mail (PEM)**. We shall discuss them in brief, as a detailed discussion of these techniques is beyond the scope of this text.

Pretty Good Privacy (PGP)

Phil Zimmermann is given credit for **Pretty Good Privacy (PGP)**. PGP is a complete email security package that provides privacy, authentication, digital signatures and compression. The PGP software and its complete source code of PGP are freely downloadable from the Internet. PGP versions for Windows, Unix and Macintosh operating systems are available. PGP uses the principles of public key encryption.

Privacy Enhanced Mail (PEM)

Privacy Enhanced Mail (PEM), unlike PGP, was the effort of a working group and not an individual. Messages sent with PEM are first translated into a canonical (common) form so that the same conventions about white spaces, tabs, carriage returns and linefeeds are used. This transformation ensures that the MTAs that sometimes modify messages because they do not understand certain characters are not allowed to do so here. Next, the same principles of public key encryption are used.

Unlike PGP, PEM does not support compression. The encryption in PGP is done using 128-bit keys. However, in PEM, this is done by using only 56-bit keys.

19.3 FILE TRANSFER PROTOCOL (FTP)

19.3.1 Introduction

We have seen how email works. However, there are situations when we want to receive or send a file from or to a remote computer. Emails are usually just short messages. Transferring files from one computer to another is quite different. A special software and set of rules called **File Transfer Protocol (FTP)** exists for this purpose. FTP is a high-level (application layer) protocol that is aimed at providing a very simple interface for any user of the Internet to transfer files. At a high level, a

user (the client) requests the FTP software to either retrieve from or upload a file to a remote server. We shall study how this works in detail.

Figure 19.21 shows at a broad level, how an FTP client can obtain a file ABC from an FTP server.

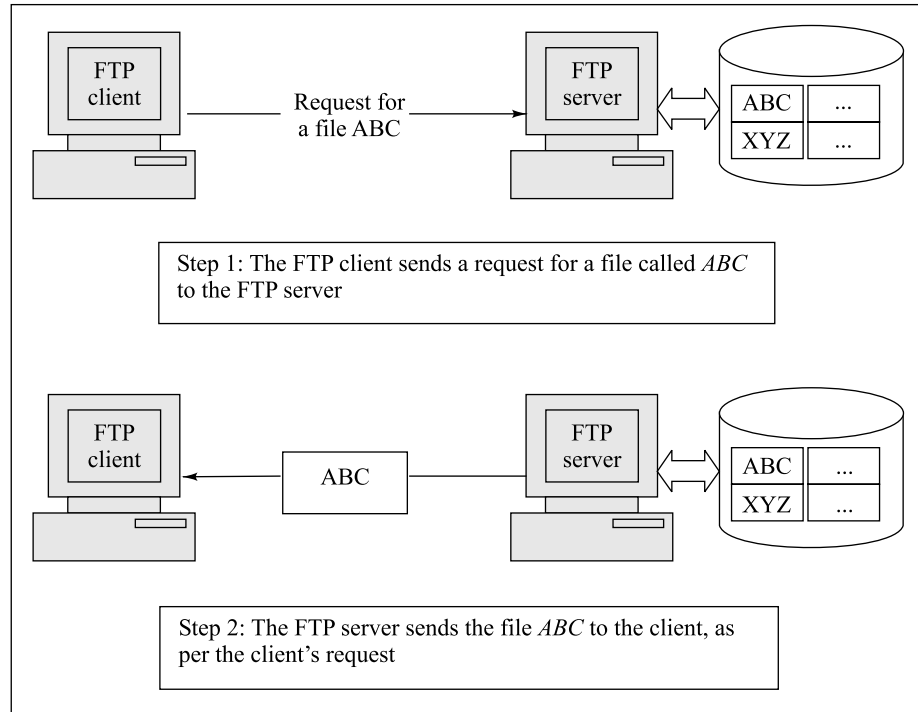


Fig. 19.21 A high level view of File Transfer Protocol (FTP)

For this, a user at the FTP client host might enter a command such as *GET ABC*, which means that the client is interested in obtaining a file called *ABC* from the specified server. FTP supports other commands such as *PUT*, *OPEN*, *CLOSE*, etc. The commands are self-explanatory.

19.3.2 Issues with File Transfers

Emails are meant for short message transfers. FTP is meant for file transfers. But that is not the sole reason why FTP was born in the first place. When a user wants to download a file from a remote server, several issues must be dealt with. First of all, the client must have the necessary authorizations to download that file. Secondly, the client and server computers could be different in terms of their hardware and/or operating systems. This means that they might represent and interpret data in different formats (e.g. floating point representation). Thirdly, an end user must not be concerned with these issues as long as s/he has the necessary access rights.

FTP provides a simple file transfer mechanism for the end user, and internally handles these complications without bothering her/him.

19.3.3 FTP Basics

Let us first discuss the user perspective of FTP. FTP presents the user with a prompt and allows entering various commands for accessing and downloading files that physically exist on a remote computer. After invoking an FTP application, the user identifies a remote computer and instructs FTP to establish a connection with it. FTP contacts the remote computer using the TCP/IP software. Once the connection is established, the user can choose to download a file from the remote computer or s/he can send a file from her/his computer to be stored on the remote computer.

However, FTP differs from other application layer protocols in one respect. All other application layer protocols use a single connection between a client and a server for intercommunication. However, FTP uses two connections between a client and a server. One connection is used for the actual file's data transfer, and the other is used for control information (commands and responses). This separation of data transfer and commands makes FTP more efficient. Internally, this means that FTP uses two TCP/IP connections between the client and the server. This basic model of FTP is shown in Fig. 19.22.

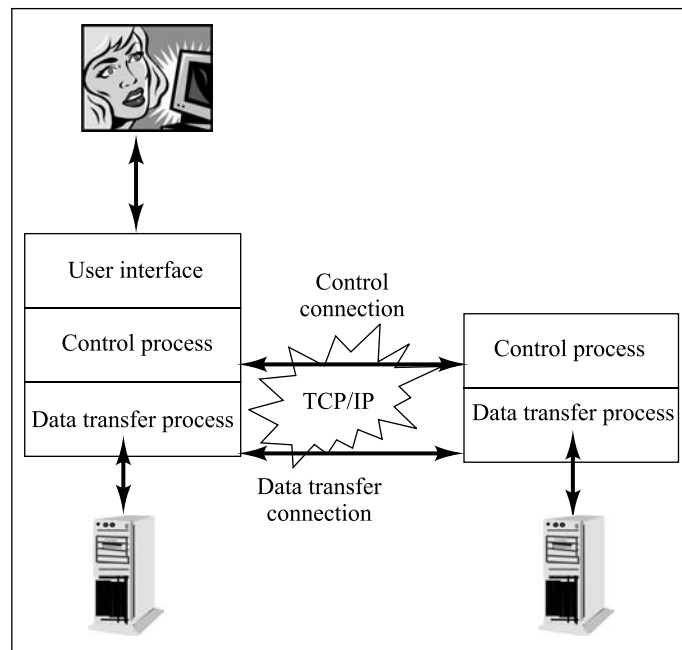


Fig. 19.22 Two connections are used in the FTP process

As shown in Fig. 19.22, the client has three components, viz., the user interface, the client control process and the client data transfer process. On the other hand, the server has just two components, viz., the server control process and the server data transfer process. Since there is no interaction required at the server's side exactly at the time of file transfer, the user interface component is not required at the server. The TCP **control connection** is made between the control processes of the client and the server. The TCP **data transfer connection** is made between the data transfer processes of the client and the server. While the data of the file is sent (in the form of IP packets and TCP/IP protocol) from the server to the client, the server keeps a track of how much data is sent (number of bytes sent, percentage of the total file size in bytes, etc.) and how much is remaining. It keeps

on sending this information simultaneously on the second connection, viz., the control connection. This is how, the control connection reassures the user downloading/uploading the file that the file transfer is going on successfully, by displaying messages about the number of bytes transferred so far, the number of bytes remaining to be transferred, the completion percentage, etc.

Note that if multiple files are to be transferred in a single FTP session, then the control connection between the client and the server must remain active throughout the entire FTP session. The data transfer connection is opened and closed for each file that is to be transferred. The data transfer connection opens every time the commands for transferring files are used, and it gets closed when the file transfer is complete.

19.3.4 FTP Connections

Let us understand how the control and data transfer connections are opened and closed by the client and the server during an FTP session.

Control Connection

The process of the creation of a control connection between a client and a server is pretty similar to the creation of other TCP connections between a client and a server. Specifically, the following two steps are involved here:

1. The server passively waits for a client (passive open). In other words, the server waits endlessly for accepting a TCP connection from one or more clients.
2. The client actively sends an open request to the server (active open). That is, the client always initiates the dialog with the server by sending a TCP connection request.

This is shown in Fig. 19.23.

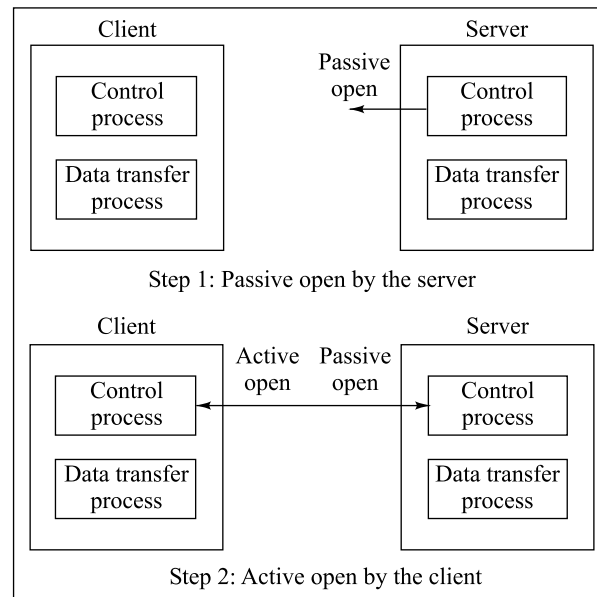


Fig. 19.23 Opening of the control connection between the client and the server

The opening of a control connection internally consists of the following steps:

1. The user on the client computer opens the FTP client software. The FTP client software is a program that prompts the user for the domain name/IP address of the server.
2. When the user enters these details, the FTP software on the client issues a *TCP connection* request to the underlying TCP software on the client. Of course, it provides the IP address of the server with which the connection is to be established.
3. The TCP software on the client computer then establishes a TCP connection between the client and the server using a three-way handshake as we have discussed previously in the section on TCP. Of course, internally it uses protocols such as IP and ARP for this as discussed many times before.
4. When a successful TCP connection is established between the client and the server, it means that an FTP server program is ready to serve the client's requests for file transfer. Note that the client can either download a file from the server or upload a file on to the server. This is when we say that the control connection between the client and the server is successfully established.

As we have noted earlier, the control connection is open throughout the FTP session.

Data Transfer Connection

The connection for data transfer, in turn, uses the control connection established previously. Note that unlike the control connection, which always starts with a passive open from the server, the data transfer connection is always first requested for by the client. Let us understand how the data transfer connection is opened.

The client issues a passive open command for the data transfer connection. This means that the client has opened a data transfer connection on a particular port number, say X, from its side.

The client uses the control connection established earlier, to send this port number (i.e., X) to the server.

The server receives the port number (X) from the client over the control connection, and invokes an open request for the data transfer connection on its side. This means that the server has also now opened a data transfer connection. This connection is always on port 20, which is the standard port for FTP on any Web server (not specifically shown in the figure).

This is shown in Fig. 19.24.

19.3.5 Client–Server Communication using FTP

Having opened the control and data transfer connections, the client and the server are now ready for transferring files. Note that the client and the server can use different operating systems, file formats, character sets, and file structures. FTP must resolve all these incompatibility issues. Let us now study how FTP achieves this using the control connection and the data transfer connection.

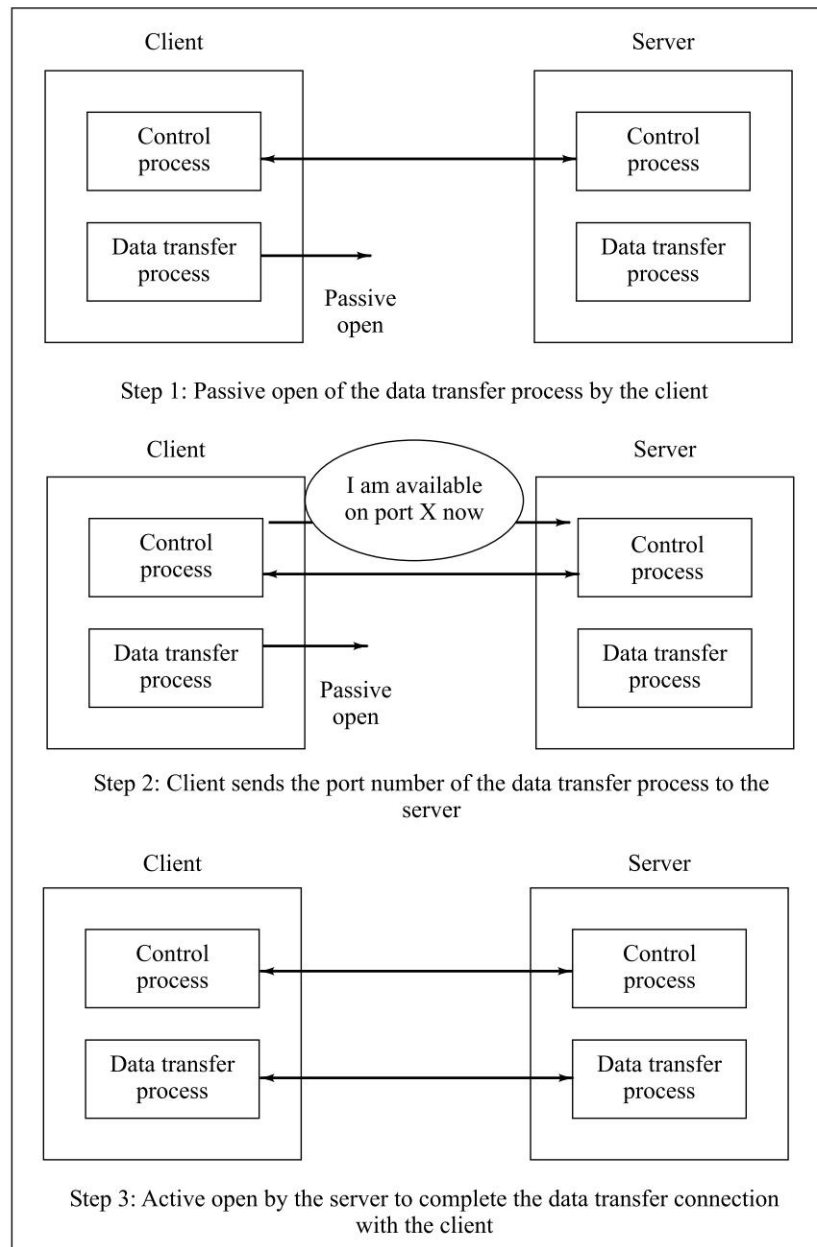


Fig. 19.24 Opening of the control connection between the client and the server

Control Connection

The control connection is pretty simple. Over the control connection the FTP communication consists of one request and one response. This request-response model is sufficient for FTP, since

the user sends one command to the FTP server at a time. This model of the control connection is shown in Fig. 19.25.

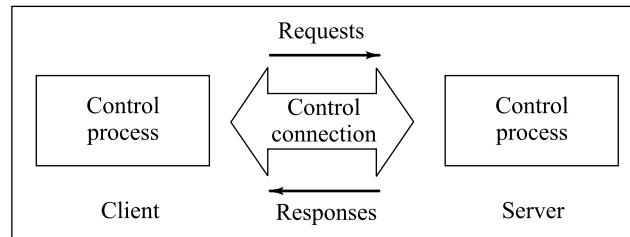


Fig. 19.25 Command processing using the control connection

The requests sent over the control connection are four-character commands such as QUIT (to log out of the system) ABOR (to abort the previous command), DELE (to delete a file), LIST (to view the directory structure), RETR (to retrieve a file from the server to the client), STOR (to upload a file from the client to the server), etc.

Data Transfer Connection

The data transfer connection is used to transfer files from the server to the client or from the client to the server, as shown in Fig. 19.26. As we have noted before, this is decided based on the commands that travel over the control connection.

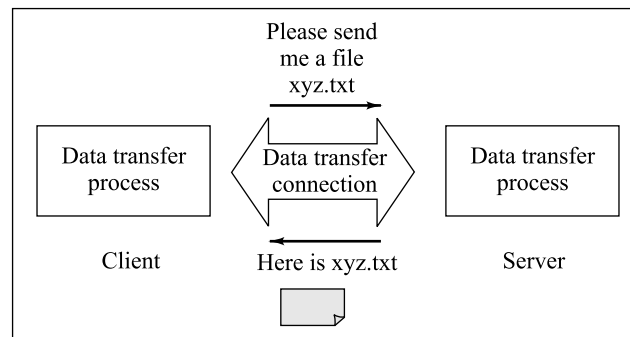


Fig. 19.26 File transfer using the data transfer connection

The sender must specify the following attributes of the file:

1. **Type of the file to be transferred** – The file to be transferred can be an ASCII, EBCDIC or **Image file**. If the file has to be transferred as ASCII or EBCDIC, the destination must be ready to accept it in that mode. If the file is to be transferred without any regard to its content, the third type is used. This third and last type, i.e., Image file is actually a misnomer. It has nothing to do with images. Actually, it signifies a binary file that is not interpreted by FTP in any manner, and is sent as it is. Compiled programs are examples of image files.
2. **The structure of the data** – FTP can transfer a file across a data transfer connection by interpreting its structure in the following ways:

- **Byte-oriented structure** – The file can be transmitted as a continuous stream of data (byte-oriented structure), wherein no structure for the file is assumed.
 - **Record-oriented structure** – The other option for the structure of the file being transferred is the record-oriented structure, where the file is divided into records and these records are then sent one by one.
3. **The transmission mode** – FTP can transfer a file by using one of the three transmission modes as described below:
- **Stream mode** – If the file is transmitted in stream mode, which is the default mode, data is delivered from FTP to TCP as a continuous stream of data. TCP is then responsible for splitting up the data into appropriate packets. If the data uses the byte-oriented structure (see earlier point), then no end-of-file character is needed. When the sender closes the TCP connection, the file transfer is considered to be complete. However, if the file follows the record-oriented structure, then each record will have an end-of-record character, and the file would have an end-of-file character at the end.
 - **Block mode** – Data can be delivered from FTP to TCP in terms of blocks. In this case, each data block follows a three-byte header. The first byte of the header is called block descriptor, whereas the remaining two bytes define the size of the block.
 - **Compressed mode** – If the file to be transferred is being compressed, it can be compressed before it is sent. Normally, the Run Length Encoding (RLE) compression method is used for compressing a file. This method replaces repetitive occurrences of a data block by the first occurrence only, and a count of how many times it repeats is stored along with it. For example, the most compressed data blocks in case of a text file are blank spaces, and those in a binary file are null characters.

This information is used by the FTP to resolve the heterogeneity problem. In any case, we must note that the data travels from the sender to the recipient as IP packets. That is, the file is broken down into TCP packets, and then into IP packets. The IP packets are then sent one by one by the sender to the recipient. We have discussed this earlier, and shall not elaborate it here.

WU-FTP, CuteFTP are some of the most commonly used FTP software implementations.

19.4 TRIVIAL FILE TRANSFER PROTOCOL (TFTP)

The **Trivial File Transfer Protocol (TFTP)** is a protocol used for transferring files between two computers, similar to the use of FTP. However, TFTP is different from FTP in one major respect. FTP uses the very reliable TCP as the underlying transport layer protocol, whereas TFTP uses the unreliable UDP protocol for data transport. Other minor differences between FTP and TFTP are that while FTP allows changing directory of the remote computer or to obtain a list of files in the directory of the remote computer, TFTP does not allow this. Also, there is no interactivity in TFTP. It is a protocol designed for purely transferring files.

There are situations where we need to simply copy a file without needing to use the sophisticated features provided by FTP. In such situations, TFTP is used. For example, when a diskless workstation or a router is booted, it is required to download the bootstrap and configuration files to that workstation or router. The device (workstation or router) in such a case simply needs to have the TFTP, UDP and IP software hard coded into its Read Only Memory (ROM). After receiving power, the device executes the code in ROM, which broadcasts a TFTP request across the network. A TFTP server

on that network then sends the necessary files to the device, so that it can boot. Not much of error checking and authentication is required here. TFTP is a suitable candidate for such situations.

TFTP does not allow for user authentication unlike FTP. Therefore, TFTP must not be used on computers where sensitive/confidential information is stored.

TFTP transfers data in fixed-size blocks of 512 bytes each. The recipient must acknowledge each such data block before the sender sends the next block. Thus, the sender sends data packets in the form of blocks and expects acknowledgement packets, whereas the recipient receives data blocks and sends acknowledgement packets. Either of them must time out and retransmit, if the expected data block or acknowledgement, as appropriate, does not arrive. Also, unlike FTP, there is no provision for resuming an aborted file transfer from its last point.

SUMMARY

Computers work best with numbers, humans don't. Humans prefer names. Every computer on the Internet has a unique IP address. Since the IP addresses used on the Internet to identify computers are in the numerical form, it would be very difficult for humans to remember the IP addresses of even a few computers. This difference between the perceptions of humans and computers is resolved by assigning names to computers. Thus, the name of a computer is mapped to its IP address.

The name given to a group of computers, that is, to a computer network, is called its domain name. For instance, yahoo is a domain name. Individual computers within a domain are named by the computer's name, then a dot, following the domain name. For instance, Jim's computer on yahoo would be jim.yahoo. At the highest level, domain names themselves are grouped under categories such as commercial organizations (com), non-profit organizations (org), and so on. Thus, the complete domain name for Jim's computer on the yahoo domain would be jim.yahoo.com.

Electronic mail (email) was created to allow two individuals to communicate using computers. Email combines the best features of a telephone call (immediate delivery) and a letter delivered via post/courier (an immediate response is not compulsory, but is always possible). Using email, a user can send messages to one or more other users at the same time, reply to messages, forward messages, etc. Email is perhaps the most widely known and used application on the Internet. It is also the cheapest and an environment friendly mechanism of communication. Nowadays, one can send photographs, images and videos along with email messages as attachments.

An email user needs to have an email client software (such as Outlook Express, Netscape mail, Eudora or Lotus Notes client) to compose or receive emails. The email client software is also called a User Agent (UA). Every email user has an email address in the form *User-name@domain-name*. This identifies any email user on the Internet uniquely.

Two main protocols exist for email transport. The Post Office Protocol (POP) is concerned with the retrieval of an email message stored on a server computer, whereas Simple Mail Transfer Protocol (SMTP) is actually responsible for transmitting an email message between the sender and the recipient. SMTP is the backbone of the actual email transmission. POP allows a user to access emails stored on the SMTP server, without having to visit the location of the remote SMTP server.

Multipurpose Internet Mail Extensions (MIME) allows an email to not only contain text data, but also any binary file such as an image, audio, video, documents in different formats. The MIME headers identify what type of data is being sent along with the text message.

Two email privacy standards are available, viz., Pretty Good Privacy (PGM) and Privacy Enhanced Mail (PEM). PEM is an Internet standard unlike PGP, which was mostly developed by an individual.

When big files need to be transferred from one computer to another remote computer over the Internet, email may not be suitable. In such situations, the File Transfer Protocol (FTP) is used to transfer files between the two computers. FTP, like email, is an application layer protocol that provides a very simple interface for a user to send/receive a file from a remote file server.

Unlike other applications at the application layer, FTP opens two connections between the client and the server computers. One connection (called data transfer connection) is used for the actual file transfer, while the other connection (called control connection) is used for exchanging control information. For instance, when a file is being transferred from the server to the client, the data transfer connection would be used to transfer the data, whereas the control connection would be used to signal to the user as to what percentage of the file transfer is completed.

A simpler version of FTP, called Trivial File Transfer Protocol (TFTP) also exists, but is used rarely. Unlike FTP, TFTP does not perform any validations or error control. This is because FTP uses the reliable transmission protocol (TCP), whereas TFTP uses the unreliable transmission protocol (UDP).

KEY TERMS AND CONCEPTS

Block mode	Mailbox
Browser-based emails	Mail Transfer Agent (MTA)
Byte-oriented structure	Multipurpose Internet Mail Extensions (MIME)
Compressed mode	Post Office Protocol (POP)
Control connection	Pretty Good Privacy (PGP)
Data transfer connection	Privacy Enhanced Mail (PEM)
Distributed database	Resolver
Domain name	Simple Mail Transfer Protocol (SMTP)
Domain Name System (DNS)	Spooling
DNS server	Stream mode
Electronic mail (Email)	Stream oriented structure
Email address	Transmission mode
Email client	Trivial File Transfer Protocol (TFTP)
Email server	UDP packet
File Transfer Protocol (FTP)	User agent
Host name	Web server
Internet Mail Access Protocol (IMAP)	Web site
Local DNS server	

QUESTIONS

True/False

1. A domain name is a name given to a network for ease of reference of humans.
2. Humans must interact with the Internet using IP addresses.

3. A computer's name on the Internet must be made up of only three parts.
4. The Domain Name System (DNS) is a distributed database.
5. A DNS server obtains the domain name of a computer when its IP address is provided to it.
6. An email mailbox is just a storage area on the disk of the computer.
7. Email is synchronous mode of communication.
8. The underlying protocol used in emails is not TCP/IP.
9. An email address takes the form *User-name#domain-name*.
10. POP is used for email transfer between the source and destination email servers.
11. IMAP downloads messages to the client computer.
12. FTP stands for Fast Transport Protocol.
13. FTP uses a single connection between the client and server computers.
14. FTP is more reliable than TFTP.
15. TFTP involves user authentication.

Multiple-Choice Questions

1. TFTP is an example of _____ layer protocol.
 - (a) application
 - (b) network
 - (c) transport
 - (d) physical
2. The hosts.txt file used to contain _____ and _____.
 - (a) IP address, physical address
 - (b) IP address, domain name
 - (c) Domain name, IP address
 - (d) Domain name, physical address
3. The com domain name refers to _____.
 - (a) common
 - (b) commercial
 - (c) computer
 - (d) None of the above
4. _____ is a storage area to store emails.
 - (a) Database
 - (b) File
 - (c) Mailbox
 - (d) Server
5. The sender of an email needs to have email _____ software.
 - (a) server
 - (b) mailbox
 - (c) sharing
 - (d) client
6. The _____ symbol is used to connect the user name and the domain name portions of an email id.
 - (a) &
 - (b) @
 - (c) *
 - (d) \$
7. In an email id, the prefix refers to the _____.
 - (a) domain name
 - (b) user name
 - (c) IP address
 - (d) None of the above
8. _____ is used to temporarily hold messages until they can be sent.
 - (a) Spool
 - (b) Mailbox
 - (c) MTA
 - (d) Client
9. _____ protocol is used to retrieve emails from a remote server.
 - (a) POP
 - (b) IP
 - (c) POP
 - (d) SMTP

10. _____ protocol is used for transferring mails over the Internet.
(a) POP (b) IP
(c) POP (d) SMTP
11. _____ allows non-text data to be sent along with an email message.
(a) PGP (b) MIME
(c) PEM (d) MTA
12. For transferring big files over the Internet, the _____ protocol is used.
(a) SMTP (b) POP
(c) HTTP (d) FTP
13. _____ uses UDP as the transport protocol.
(a) FTP (b) TFTP
(c) All of the above (d) None of the above
14. _____ uses TCP as the transport protocol.
(a) FTP (b) TFTP
(c) All of the above (d) None of the above
15. FTP uses the control connection for transferring _____.
(a) data (b) control information
(c) data and control information (d) All of the above

Detailed Questions

1. What is the need for additional suffixes such as *com*, *edu* and *gov*?
2. What is DNS? Why is it required?
3. Draw a conceptual view of the Internet domain name space.
4. Explain the significance of a DNS server.
5. When a DNS server receives a request, what are the possible actions that it can take?
6. What is a resolver?
7. What is the purpose of an email server?
8. Describe spooling in brief.
9. Why is an email client required?
10. Discuss email architecture in brief along with its main components.
11. Discuss POP and SMTP servers in brief. How is IMAP different from POP?
12. What is the purpose of FTP?
13. Discuss the FTP connection mechanism between the client and the server.
14. What are the specific purposes of the control connection?
15. What are the differences between FTP and TFTP?

TCP/IP

Part 4: WWW, HTTP and TELNET

20



20.0 INTRODUCTION

Apart from email, the most popular application running on the Internet is the **World Wide Web (WWW)**. It is so important that people often confuse it with the Internet itself. However, WWW is just an application such as email and File Transfer that uses the Internet for communications, i.e., TCP/IP as an underlying transport mechanism. Many companies have Internet **Web sites**. What is meant by a Web site? It is a collection of **Web pages** like a brochure (a collection of paper pages), except that the pages are stored digitally on the **Web server**. You could have a dedicated server computer in your company for storing these Web pages (or for *hosting the Web site, as it is popularly called*), or you could lease some disk space from a large server installed at your ISP's location.

In either case, the server has a piece of software running on it, which is actually the Web server software. However, commonly, the server computer itself is called the *Web server* (actually inaccurately in a strict sense). The function of the Web server hardware and software is to store the Web pages and transmit them to a client computer as and when it requests for the Web pages to be sent to them from the server, after locating them. The Web site address (also called **Uniform Resource Locator** or **URL**) will be that of the first page (also called **home page**) of your Web site installed on the server of your ISP. Internally, each Web page is a computer file stored on the disk of the server. The file contains *tags* written in a codified form, as we shall see, that decide how the file would look like when displayed on a computer's screen.

In your company's Web site, you can display information about your products, employees, policies or even the practices. It can, therefore, be used as a company news bulletin and it can be made more attractive using different colors, sound and animation (multimedia). A company can use it like a window in a shop where you display what you want to sell. On a Web site, not only what is to be sold is displayed; there can be Web sites which are dedicated to specialized tasks such as displaying news, share prices, sports, directory services, or indeed a combination of one or more of these services. For example, CNN's Web site is www.CNN.com. Sometime, you might also type the **HTTP** prefix, which makes the name of the same Web site <http://www.CNN.com>. This only indicates that we use the HTTP protocol to communicate with the Web site. Mentioning this is optional. If nothing is mentioned, the system assumes HTTP. We shall see what this HTTP means, later, but will stop mentioning this explicitly as this is assumed.

The term WWW refers to a set of Internet protocols and software, which together present information to a user in a format called **hypertext**, as we shall study. The WWW became quite popular in the mid 1990s. Tim Berners-Lee did the primary work in the development of the WWW at the European Laboratory for Particle Physics (CERN).

20.1 THE BASICS OF WWW AND BROWSING

20.1.1 How Does a Web Server Work?

A Web server is a program running on a server computer. Additionally, it consists of the Web site containing a number of Web pages. A Web page constitutes simply a special type of computer file written in a specially designed language called **Hyper Text Markup Language (HTML)**. Each Web page can contain text, graphics, sound, video and animation that people want to see or hear.

The Web server constantly and passively waits for a request for a Web page from a browser program running on the client and when any such request is received, it locates that corresponding page and sends it to the requesting client computer. To do this, every Web site has a server **process** (a running instance of a program) that listens to TCP connection requests coming from different clients all the time. After a TCP connection is established, the client sends one request and the server sends one response. Then the server releases the connection. This request-response model is governed by a protocol called **Hyper Text Transfer Protocol (HTTP)**. For instance, HTTP software on the client prepares the request for a Web page, whereas the HTTP software on the server interprets such a request and prepares a response to be sent back to the client. Thus, both client and server computers need to have an HTTP software running on them. Do not confuse HTTP with HTML. HTML is a special language in which the Web pages are written and stored on the server. HTTP is a protocol, which governs the dialog between the client and the server.

You would realize that this is a typical client-server interaction, as we have discussed so far – the Web server obviously acting as a server, in this case. Rather than requesting for a file as in case of FTP, a client requests for a specific Web page here. As we know, each Web page is stored in HTML format on the server. The server receives a request for a specific Web page, locates it with the help of the operating system, and sends it back to the client using TCP/IP as the basic message transport mechanism. After receiving the Web page in the HTML format in the memory of the client computer, the browser *interprets* it, i.e., displays it on the screen of the client computer.

20.1.2 How Does a Web Browser Work?

A Web browser acts as the client in the WWW interaction. Using this program, a user requests for a Web page (which is a disk file, as we have noted) stored on a Web server. The Web server locates this Web page and sends it back to the client computer. The Web browser then interprets the Web page written in the HTML language/format and then displays it on the client computer's screen.

The typical interaction between a Web browser (the client) and a Web server (the server) is as shown in Fig. 20.1 and takes place as explained below.

1. The user on the client computer types the full file name including the domain name of the Web server that hosts the Web page that s/he is interested in. This name is typed on a screen provided by the Web browser program running on her/his computer. As we know, this full file name is called Uniform Resource Locator (URL). A URL signifies the full, unique path of any file on the Internet.

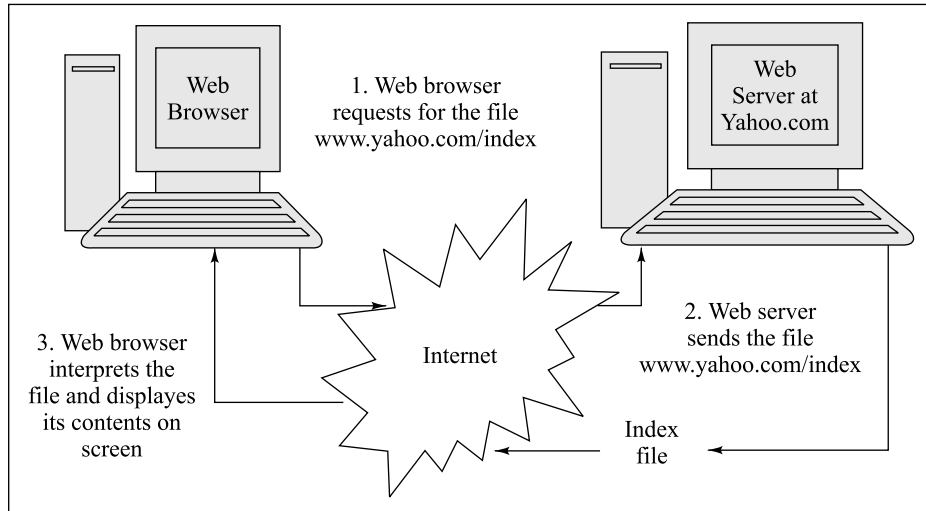


Fig. 20.1 Interaction between a Web browser and a Web server

For instance, a URL could be `http://www.yahoo.com/index` or only `www.yahoo.com/index`, because specifying `http` is optional, as we have mentioned. First let us understand its anatomy using Fig. 20.2.

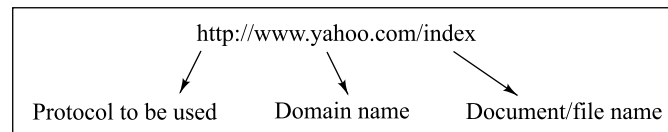


Fig. 20.2 Anatomy of a URL

Here, *http* indicates the protocol (discussed later). *Index* is the name of the file. It is stored on the Web server whose domain name is *yahoo.com*. Because it is a WWW application, it also has a *www* prefix. The forward slash (/) character indicates that the file is one of the many files stored in the domain *yahoo.com*. Suppose, the user wants another file called *newsoftheday* from this site, s/he would type `http://www.yahoo.com/newsoftheday`. Alternatively, the user can just type the name of the domain (e.g. `www.yahoo.com`). Every Web server has one default file, which can be used in these cases. When the user types just the domain name without mentioning the file name, this default file is used. This default file is returned to the Web browser in such cases. If *index* file is such a default file, this will be another way of reaching the *index* file at *yahoo.com*. The page displayed from the default file (in this case the *index* file) may then provide links to other files (or *Web pages*) stored at that domain. If a user clicks on one of them (e.g., *finance*), the *finance* file stored at the *yahoo* domain is then displayed.

2. The browser requests DNS for the IP address corresponding to `www.yahoo.com`.
3. DNS replies with the IP address for `www.yahoo.com` (let us say it is `120.10.23.21`).
4. The browser makes a TCP connection with the computer `120.10.23.21`.

5. The client makes an explicit request for the Web page (in this case, the file corresponding to the page *index* at *yahoo.com*) to the Web server using an **HTTP request**. The HTTP request is a series of lines, which, among other things, contains two important statements *GET* and *HOST*, as shown in our current example (all HTTP and HTML keywords are case-insensitive):
GET /index.htm and Host: yahoo.com

The *GET* statement indicates that the *index.htm* file needs to be retrieved (the *.htm* extension indicates that the file is written in HTML). The *Host* parameter indicates that the *index* file needs to be retrieved from the domain *yahoo.com*.

6. The request is handed over to the HTTP software running on the client machine to be transmitted to the server.
7. The HTTP software on the client now hands over the HTTP request to the TCP/IP software running on the client.
8. The TCP/IP software running on the client breaks the HTTP request into packets and sends them over TCP to the Web server (in this case, *yahoo.com*).
9. The TCP/IP software running on the Web server reassembles the HTTP request using the packets thus received and gives it to the HTTP software running on the Web server, which is *yahoo.com* in this case.
10. The HTTP software running on the Web server interprets the HTTP request. It realizes that the browser has asked for the file *index.htm* on the server. Therefore, it requests the operating system running on the server for that file.
11. The operating system on the Web server locates *index.htm* file and gives it to the HTTP software running on the Web server.
12. The HTTP software running on the Web server adds some headers to the file to form an **HTTP response**. The HTTP response is a series of lines that contains this header information (such as date and time when the response is being sent, etc.) and the HTML text corresponding to the requested file (in this case, *index.htm*).
13. The HTTP software on the Web server now hands over this HTTP response to the TCP/IP software running on the Web server.
14. The TCP/IP software running on the Web server breaks the HTTP response into packets and sends it over the TCP connection to the client. Once all packets have been transmitted correctly to the client, the TCP/IP software on the Web server informs the HTTP software on the Web server.
15. The TCP/IP software on the client computer checks the packets for correctness and reassembles them to form the original Web page in the HTML format. It informs the HTTP software on the server that the Web page was received correctly.
16. Realizing this, the HTTP software on the Web server terminates the TCP connection between itself and the client. Therefore, HTTP is called **stateless protocol**. The TCP connection between the client and the server is established for every page, even if all the pages requested by the client reside on the same server. Moreover, if the Web page contains images (photographs, icons, images etc.), for each such image, sound etc., a separate file is required, which stores it in specific formats (GIF, JPEG, PNG, etc). Therefore, if a Web page contains text, sound and image, it will take three HTTP requests to locate and bring three files residing on the same or different servers to the client, after which the browser on the client can compose these together to display that Web page. Needless to say that this involves a DNS search as many times as well. This is the reason why retrieving a Web page that contains many graphical elements is very slow.

Thus, HTTP does not remember anything about the previous request. It does not maintain any information about the state, and hence the term *stateless*. Keeping HTTP stateless is an attempt to keep the Web simple – usually more clients would be requesting Web pages from a lesser number of servers. If each such request from a client is to be *remembered*, the Web server would run out of processing power and memory in a short time.

17. The TCP/IP software on the client now hands over the Web page to the Web browser for interpretation. It is only the browser, which understands the “HTML code language” to decipher which elements (text, photo, video, etc.) should be displayed where and how. This is the meaning of ‘interpretation’. How does the browser do this?

To understand this, we shall study the Web pages more in depth.

20.1.3 HTTP Commands

Let us discuss a few commands in the HTTP protocol when a client requests a server for a Web page, as summarized in Fig. 20.3.

HTTP command	Description
GET	Request for obtaining a Web page
HEAD	Request to read the header of a Web page
PUT	Requests the server to store a Web page
POST	Similar to PUT, but is used for updating a Web page
DELETE	Remove a Web page
LINK	Connects two resources
UNLINK	Disconnects two resources

Fig. 20.3  HTTP Request commands

A browser uses the commands shown in Fig. 20.3 when it sends an HTTP request to a Web server. Let us discuss each of them. Note that these commands are case-sensitive.

- **GET** – A browser uses this command to request a Web server to send a particular Web page.
- **HEAD** – This command does not request for a Web page, but only requests for its header. For instance, if a browser wants to know the last modified date of a Web page, it would use the HEAD command, rather than the GET command.
- **PUT** – This command is exactly opposite of the GET command. Rather than requesting for a file, it sends a file to the server for storing it there.
- **POST** – This command is very similar to the PUT command. The PUT command is used to send a new file, whereas the POST command is used to update an existing file with additional data.
- **DELETE** – This command allows a browser to send an HTTP request for deleting a particular Web page.
- **LINK** – This command is used to establish hyperlinks between two pages.
- **UNLINK** – This command is used to remove existing hyperlinks between two pages.

Note that GET is the most common command sent by a client browser as a part of the HTTP request to a Web server. This is because, not many Web servers would allow a client to delete/add/

link/unlink files. This can be fatal. However, for the sake of completeness, we have discussed them briefly.

When a browser sends such an HTTP request command to a Web server, the server sends back a status line (indicating the success or failure, as a result of executing that command) and additional information (which can be the Web page itself). The status line contains error codes. For example, a status code of 200 means success (OK), 403 means authorization failure, etc.

For instance, the following line is an example command sent by a browser to a server for obtaining a Web page named *information.html* from the site *www.mysite.com*, as shown in Fig. 20.4.

```
GET http://WWW.mysite.com/information.html HTTP/1.0
```

Fig. 20.4 Example of GET command sent by a browser

This GET command requests the Web server at *www.mysite.com* for a file called *information.html*. The HTTP/1.0 portion of the command indicates that the browser uses the 1.0 version of the HTTP protocol.

In response, the server might send the following HTTP response back to the browser, as shown in Fig. 20.5.

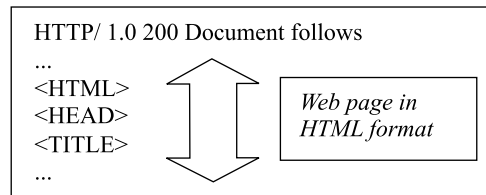


Fig. 20.5 HTTP response from the server

The first line indicates to the browser that the server is also using HTTP 1.0 as its protocol version. Also, the return code of 200 meant that the server processed the browser's HTTP request successfully. After that, there would be a few other parameters, which are not shown. After these parameters, the following line starts:

```
<HTML>
<HEAD>
<TITLE>
```

This is a Web page codified in HTML format.

We shall see what these statements mean while discussing HTML. However, for now, just keep in mind that the actual contents of the Web page are sent by the Web server to the browser with the help of these **tags**. A tag is a HTML keyword usually enclosed between less than and greater than symbols. For instance, the `<HTML>` statement (i.e., tag) indicates that the HTML contents of the Web page start now.

20.1.4 Example of an HTTP Interaction

Let us study an example of an HTTP request and response model. In this example, the browser (i.e., the client) retrieves a HTML document from the Web server. We shall assume that the TCP connection between the client and the server is already established, and we shall not discuss it further.

As shown in Fig. 20.6, the client sends a GET command to retrieve an image with the path /files/new/image1. That is, the name of the file is image1, and it is stored in the files/new directory of the Web server. Instead, the Web browser could have, of course, requested for a HTML page (i.e., a file with *html* extension).

In response, the Web server sends an appropriate return code of 200, which means that the request was successfully processed, and also the image data, as requested. We shall discuss the details shown in Fig. 20.6 after taking a look at it.

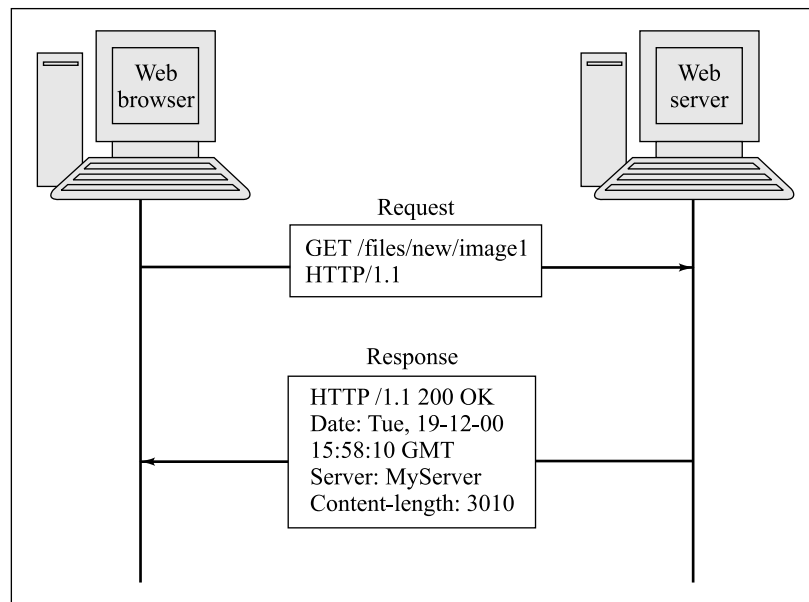


Fig. 20.6 Sample HTTP request and response interaction between a Web browser and a Web server

The browser sends a request with the GET command, as discussed. It also sends two more parameters by using two *Accept* commands. These parameters specify that the browser is capable of handling images in the GIF and JPEG format. Therefore, the server should send the image file only if it is in one of these formats.

In response, the server sends a return code of 200 (OK). It also sends the information about the date and time when this response was sent back to the browser. The server's name is the same as the domain name. Finally, the server indicates that it is sending 3010 bytes of data (i.e., the image file is made up of bits equivalent to 3010 bytes). This is followed by the actual data of the image file (not shown in the figure).

20.1.5 Proxy Server

The trouble with Web servers is that not all understand the HTTP protocol. Some of the old Web servers *speak* protocols such as FTP, Gopher (an almost obsolete Web protocol), etc. However, there is a tremendous amount of useful information on these Web servers that do not speak HTTP. How can a Web browser then be used to access this information without using HTTP? After all, a Web browser surely understands HTTP and FTP but not other protocols such as Gopher, etc. One solution, of course, is to add all these protocols to the Web browser, so that it can speak not only HTTP and FTP, but also all these protocols. However, in practice, this is not recommended, since it would mean that the Web browser would need to understand many protocols, and would make it a very bulky piece of software. Still, many browsers are programmed to have capabilities beyond HTTP and FTP. A more common approach, however, is to make use of a **proxy server**.

A proxy server is similar to a gateway that speaks HTTP to the browser, but FTP, Gopher or an older protocol to the Web server. Thus, it acts as an interpreter between the Web browser and the Web server for transforming a non-HTTP protocol to HTTP and vice versa. This means that the browser does not have to understand and/or interpret any additional protocols. This is shown in Fig. 20.7.

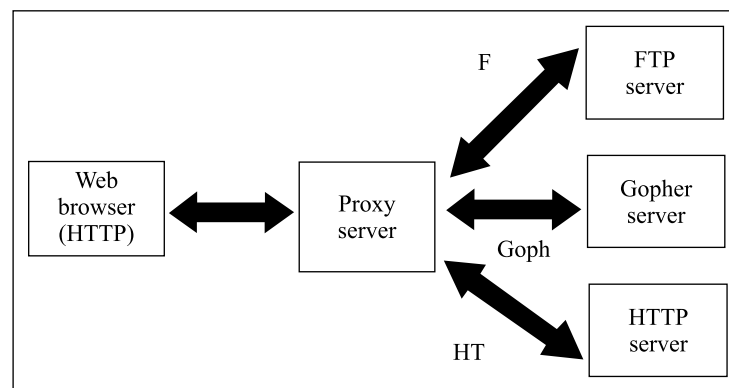


Fig. 20.7 Proxy server

Usually, the proxy server is installed on a dedicated computer in an organization, and the organization's connection to the Internet is directed via the proxy. This means that every user's Internet connection passes via the proxy.

Apart from protocol transformation, proxy servers also perform the functions of caching (keeping in main memory all the Web pages that have been requested by all the users, so that the same *cached* Web page can be sent for repeated requests for the same page from different users). That is, when a user requests for a Web page, the proxy checks if it is already available in its main memory cache, and if so, it serves the user's request rather than re-requesting the same page.

20.2 LOCATING INFORMATION ON THE INTERNET

The amount of information available on the Internet is mind boggling. However, finding the right kind of information is usually very cumbersome. For enabling people to search the required information relatively easily, the concept of a **search engine** came into being. A search engine is also referred to as **crawler**, **worm** or **knowbot** (*knowledge robot*). Google, Altavista, Yahoo, Infoseek are some of the most popular search engines on the Web.

The search engines continuously crawl through the Web pages on the Internet (hence the name *crawler*) and gather information about them to facilitate search for users in those Web pages. Most search engines use the principles of data structures such as trees and graphs to organize information. For instance, many search engines consider the Web as a huge graph, and all the Web pages as its nodes. Furthermore, all the hyperlinks are considered as arcs of the graph. The important keywords in each document are also stored along with this information by the search engine in its databases.

From the user's perspective, using a search engine is fairly easy. All that the user has to do is to enter the keywords to be searched for in the screen provided by the search engine and press a button usually called *Search* or *Submit*. In response, the search engine has to perform the search in its databases, find out the keywords matching the user's request and show them to the user.

The fact that the data on the Web is enormous and is always changing makes the challenge of constantly updating this information in the form of graphs non-trivial.

A detailed discussion of the algorithms employed by the search engines is beyond the scope of this text.

20.3 HYPER TEXT MARKUP LANGUAGE (HTML)

20.3.1 Introduction

To create Web pages, a special language called *Hyper Text Markup Language (HTML)* is used. Files created using HTML originally contained only text. However, these days, HTML allows graphics, sound and video. The new specifications make HTML much more interesting and interactive. This is mainly due to the multimedia additions to the Web pages. We would not go into their details at this juncture. We shall see how these multimedia features can be added in Web pages later. Firstly, let us study how HTML is used for codifying text pages. Basically, HTML is used to specify where and how to display headings, where a new paragraph starts, which text to display in what font and color, etc.

HTML uses *tags* to define the contents of Web pages. Tags are enclosed in angled brackets. For example, a tag `<P>` indicates the beginning of a new paragraph. Thus, when a Web page containing a `<P>` tag is sent by a Web server to a Web browser, the Web browser interprets this tag and starts displaying the contents after this point in a new paragraph. Most tags end with a corresponding `</>` tag. For example, the `<P>` tag would end with a `</P>` tag. There are many such tags to create document titles, headings, changing fonts and styles, etc.

As an example, let us consider how the tag pair `` and `` can be used to change the text font to boldface. This is shown in Fig. 20.8.

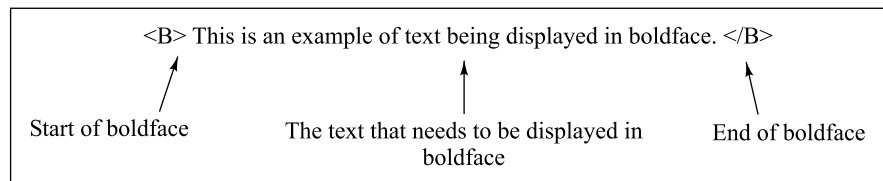


Fig. 20.8 Example of the `` and `` HTML tags to display the specified text in boldface

When a browser comes across this portion of a HTML document, it realizes that the portion of the text embedded within the `` and `` tags needs to be displayed in boldface. Therefore, it displays this text in boldface, as shown in Fig. 20.9.

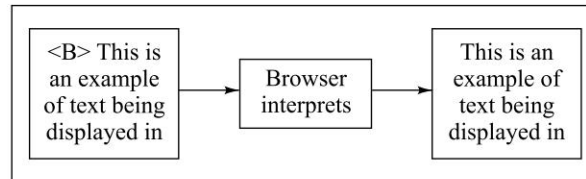


Fig. 20.9 Output resulting from the use of the `` and `` HTML tags to display the specified text in boldface

20.3.2 HTML – An Example

Let us take a simple example using some of these tags. Refer to Fig. 20.10. The figure shows on the left side, the HTML page, and on the right, it shows how the screen would look. We shall then study the HTML code in the page.

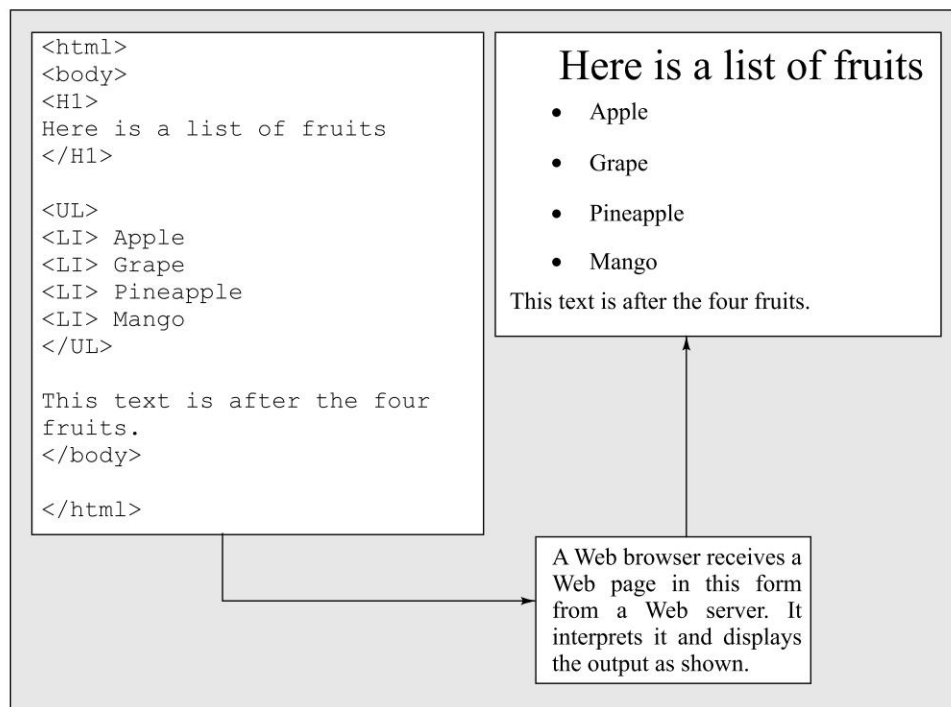


Fig. 20.10 HTML page and its corresponding output on the screen

- `<HTML>` indicates the beginning of a HTML page and `</HTML>` indicates the end of a HTML page.


- `<BODY>` indicates the beginning of the contents of the page and `</BODY>` indicates the end of the contents of the page.
- `<H1>` indicates that the text starting here is a heading and as such, should be displayed in bigger font until it hits `</H1>`.
- `` means an unordered list of items, where a `` tag indicates each item. Usually, these items are shown bulleted. The end of the list is indicated by ``.

Clearly, a Web browser must understand the various tags defined in HTML and make sense out of them. Therefore, we can call a Web browser as an HTML interpreter. However, modern Web browsers are not so simple. Besides interpreting HTML for displaying a Web page on a client computer's screen, they need to be aware of various other things, which we shall study later. For now, we assume that the only job of a Web browser is to interpret HTML and display the contents of Web pages on the screen of the client's computer. As we have noted, files containing Web pages often have *htm* or *html* extension. This tells the Web browser that it is an HTML file.

20.3.3 HTML Tags – Summary

Figure 20.11 summarizes some of the more common HTML tags.

Tag	Description
<code><html> ... </html></code>	Declare the start and end of a HTML page
<code><head> ... </head></code>	Delimit the main heading of the page
<code><title> ... </title></code>	Define the page title displayed at the left-hand top of the page
<code><body> ... </body></code>	Encapsulate the main body of the page's contents
<code><h<i>n</i>> ... </h<i>n</i>></code>	Delimit a heading at level <i>n</i>
<code> ... </code>	Display the specified text in boldface
<code><i> ... </i></code>	Display the specified text in italics
<code>
</code>	Force a line to break here when displaying
<code><p></code>	Indicates the start of a new paragraph
<code> ... </code>	Delimits an unordered list of elements that have bullets
<code></code>	Start an item in the list (There is no <code></code>)
<code></code>	Load an image at this location on the page
<code> ... </code>	Define a hyperlink (explained next)

Fig. 20.11  Common HTML tags

As you can see, most HTML tags have a corresponding ending tag. However, some have none.

20.3.4 Hyperlinks

HTML has a very interesting property. It supports the concept of **links**. In fact, the original reason behind developing HTML and WWW was links. WWW initially was just a set of linked HTML documents. Web pages usually offer links to other Web pages. For example, suppose you are currently viewing a Web page that shows a report on today's stock market. There might be links in that Web page to some of the companies mentioned in the report. These could be shown underlined or in different colors to indicate that these are actually links to other Web pages. Such links are called **hyperlinks**. You can point your mouse on any such hyperlink and click. In response, the browser would actually take you to that Web page. A conceptual view of how hyperlinks work is shown in Fig. 20.12.

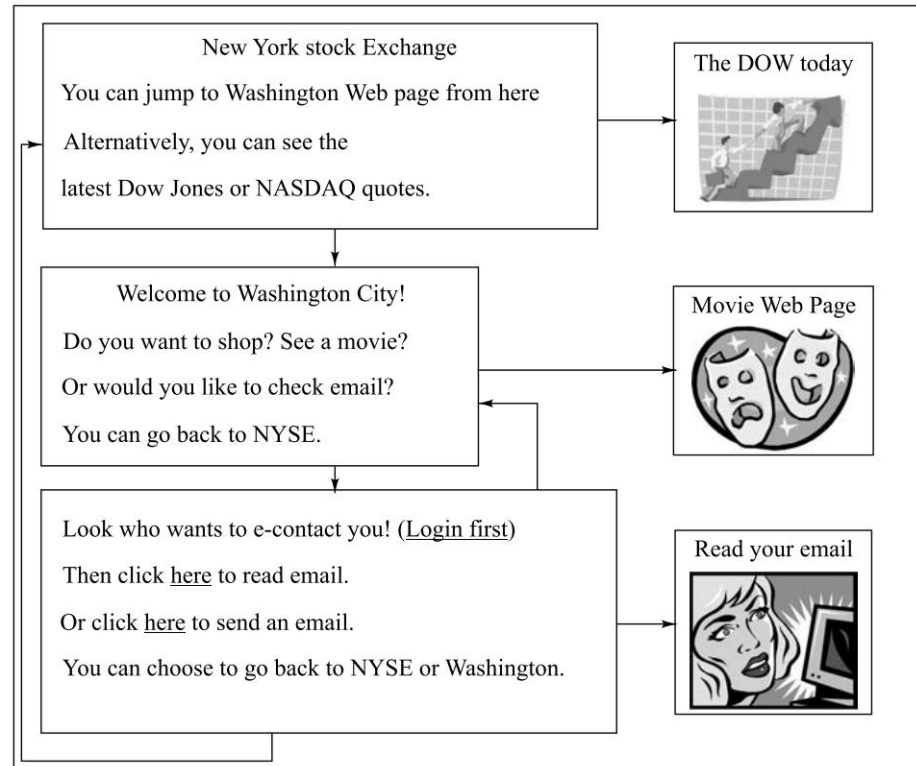


Fig. 20.12 Concept of hyperlinks

Suppose the user is presently viewing the *New York Stock Exchange* page. Three hyperlinks are shown to the user on this page (see the underlined text). By clicking on any of these hyperlinks, the user can go to the corresponding Web page. For instance, when the user clicks on the hyperlink Washington, the browser would take her/him to the *Washington City* home page, as shown. In turn, the *Washington City* home page has many hyper links that allow the user to either visit a shopping Web page or a watch a movie online. How can a Web page give the reference of other Web pages, that is, how can hyperlinks be created?

HTML uses the **anchor tag** for creating hyperlinks. For example, suppose Linda has a Web page from which she wants to allow the user an option of going to the *amazon.com* Web page. What she needs to do is simple. Add an anchor tag, specifying the Uniform Resource Locator (URL) of *amazon.com*, as shown in Fig. 20.13, at the appropriate place in her/his Web page.

```
<html>
<title> Linda's Home page!!! </title>
Hi! This is Linda Johnson's home page.
.....
<a href="http://www.amazon.com">Click here to go to amazon.com </a>
....
```

Fig. 20.13 How to add hyper links

As a result, the Web page, when displayed to an end user, would look like the one shown in Fig. 20.14. Note that the URL does not appear on the Web page. Instead, the message associated with it is displayed. When the user clicks on this hyperlink, the X and Y coordinates of the cursor are captured and fed to a program that is event-driven and running all the time. This program waits for some input and based on that input, and takes an appropriate action. This is why it is called an *event-driven* program. The action in this case is to map the captured X and Y coordinates to a URL using a table stored in the memory. In this case, it yields the URL of *amazon.com*. The program picks it up based on the coordinates and makes a connection with the server at *amazon.com* (after translating this to the IP address using the DNS, etc).


Linda's Home page!!!

Hi! This is Linda Johnson's home page.

.....

Click here to go to amazon.com

....

Fig. 20.14  *Redirection using hyperlinks*

Let us understand this step-by-step.

1. The anchor tag is specified by the `<a>` and `` boundaries. `<a>` indicates its beginning and `` indicates its end.
2. The HREF (hyperlink reference) attribute specifies which URL you want to specify in this anchor tag. In this case, we have specified `www.amazon.com` as the site where this anchor tag should take us. A table of X, Y coordinates of the cursor on the screen and the URL is maintained internally in some format. Note that this table contains the range of cursor values for which the URL is the same. This is because you must get transferred to that URL even if you click anywhere on the underlined text.
3. Next, we can optionally specify a message to be displayed on the screen corresponding to this anchor tag when the user views the Web page. Note that when we specify the anchor tag, HTML automatically adds an underline to indicate that it is a hyperlink, and not pure text.
4. Now, if the person viewing this Web page clicks on the underlined message shown in Fig. 20.14, using the HTTP protocol, the Web browser sends a request to the `amazon.com` Web site to send its *home page* as we have discussed earlier. A home page is the default Web page on a Web site. It is sent to the browser when the browser does not request for a specific Web page stored on the Web site, but instead, specifies just the Web site's name. Of course, internally the actual transmission of the contents of the Web page would happen using the TCP/IP protocol.

This is how Web pages can specify logical connections to other Web pages. When a user is browsing the Internet, most of the times, s/he actually follows the hyperlinks by clicking the mouse over them. As we know, internally the anchor tag and the associated URL result in the navigation from the displayed Web page to the hyperlinked Web page.

Now we know how a Web browser knows which file to request for, when a hyperlink is clicked. The Web server at the newly visited Web site (in this case, `amazon.com`) as usual sends its home page to the Web browser, next. A Web page typically has many such hyperlinks. These enable the user to do a variety of things. As we have seen, these hyperlinks need not be for the same Web site.

For example, suppose there is a reference of a technical book on a research site. Then, you could have a hyperlink to a bookseller who sells that book—as shown in our example, where amazon.com was the bookseller!

This simple feature of hyperlinks allows users to jump from one Web page to another on the same or any other Web site on the WWW just at the click of the mouse button. This feature also allows logical grouping of information, although the information is located on different computers that are physically scattered. For instance, I can create a Web page that lists all bookshops, which are all hyperlinked. When I click on a specific bookshop, it would take me to the home page of that bookshop by using its associated URL. When I click on the second bookshop, it will take me to the Web site (i.e., the home page) of the other bookshop by using its associated URL. While in the Web page of the second bookshop, if I click on some other hyperlink, I can go there. This is how a browser can navigate. While navigating, it keeps track of previous pages and corresponding URLs also. This comes handy if we click on the *Back* or *Cancel* buttons.

20.4 WEB BROWSER ARCHITECTURE

20.4.1 Introduction

Web browsers have a more complex structure than Web servers. This is because a Web server's task is relatively simple. It has to endlessly wait for a browser to open a new TCP connection and request for a specific Web page. When a Web server receives such a request, it locates the requested Web page, sends it back to the requesting browser, closes the TCP connection with that browser and waits for another request. That is why we say that a Web server waits for TCP connections *passively*. It does not initiate HTTP requests but waits for HTTP requests from one or more clients, and serves them. Therefore, a Web server is said to execute a *passive open* call upon start, as we have discussed before.

It is the responsibility of the browser to display the document on the user's screen when it receives it from the server. As a result, a browser consists of several large software components that work together to provide an abstracted view of a seamless service. Let us take a look at the architecture of a typical Web browser. This will give us more insight into its working; see Fig. 20.15 given below.

A browser contains some pieces of software that are mandatory and some that are optional depending upon the usage. HTTP client program shown in the above figure as (2) and HTML interpreter program as (3) are mandatory. Some other interpreter programs as in (4), Java interpreter program as (5) and other optional interpreter programs as (6) are optional. The browser also has a controller, shown as (1), which manages all of them. The controller is like the control unit in a computer's CPU. It interprets both mouse clicks/selections and keyboard inputs. Based on these inputs, it calls the rest of the browser's components to perform the specific tasks. For instance, when a user types a URL, the controller calls the HTTP client program to fetch the requested Web page from a remote Web server whose address is given by the URL. When the Web page is received, the controller calls the HTML interpreter to interpret the tags and display the Web page on the screen.

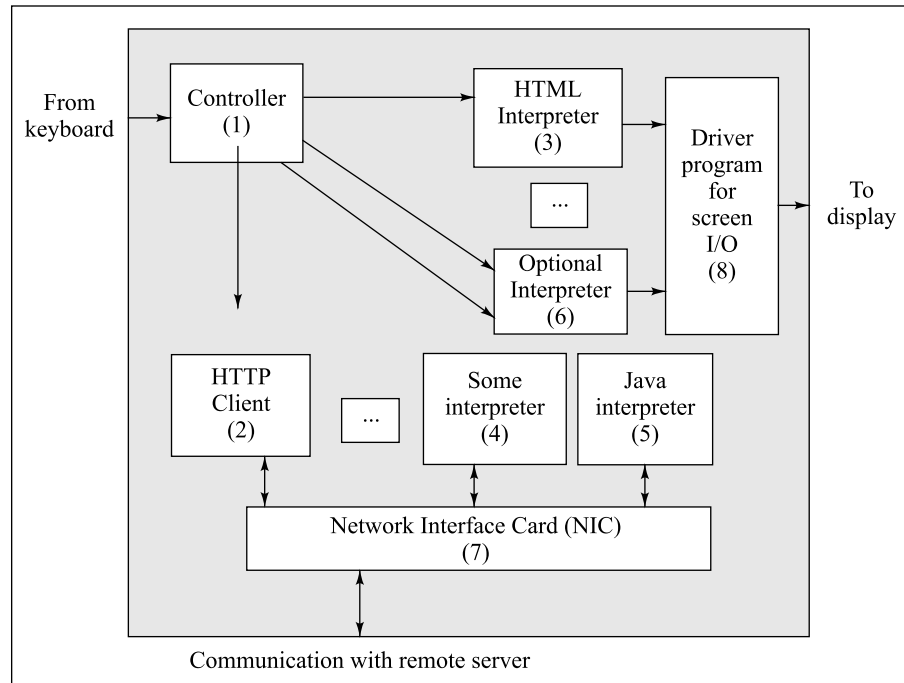


Fig. 20.15 Internal architecture of a Web browser

The HTML interpreter takes an HTML document as input and produces a formatted version of it for displaying it on the screen. For this, it interprets the various HTML tags and translates them into display commands based on the display hardware in the user's computer. For instance, when the interpreter sees a tag to make the text bold, it instructs the display hardware to display the text in the bold format. Similarly, when the interpreter encounters a tag to change paragraphs, it performs the necessary display functions in conjunction with the display hardware.

20.4.2 Optional Clients

Apart from the HTTP client and an HTML interpreter, a browser can contain additional clients. We have seen applications such as FTP and email. For supporting these applications, a browser contains FTP and email client programs. These enable the browsers to perform FTP and email services. The interesting point is that a user need not explicitly invoke these special services. Instead, the browser invokes them automatically on behalf of the user. It hides these details from the user. For example, for sending an email, there would be a link on a HTML page. Usually, there is such a link on every Web site so that the user can send an email to the owner or technical support staff of the Web site to resolve any query or obtain more information, report problems, etc.

If the user clicks that link with a mouse, the controller of the browser would interpret this and then it would invoke the email client program automatically. Similarly, the user could just select an option on the screen to invoke the FTP service. That mouse click would be interpreted by the controller of the browser and then it would invoke the FTP program through the FTP client program. The user need not be aware of this. S/he gets a feeling that transferring a file or sending

an email can be achieved through the browser. From a user's point of view, s/he is just using the Web browser as usual.

20.5 WEB PAGES AND MULTIMEDIA

Let us take a brief look at how a Web page can contain graphics. Non-textual information such as a graphics image or a digitized picture is not stored as a part of the HTML document. Instead, they reside as separate files at separate locations on the Web server. There are references to them from the original HTML document, thus establishing a link between them.

When a browser retrieves the Web page and encounters such a reference to an image or a picture, it follows the specified link mentioned in the HTML document and goes to the location on the Web server where the image is actually stored. Of course, it has to use the HTTP protocol for each such file (e.g., audio, image, video, etc.) to establish and break the link between the client and the Web server each time. This is true in case of the HTTP protocol version 1.0. However, realizing that this results into a lot of connection creations and terminations between a client and a server even when the same two hosts (a client and a server) are communicating with each other for a group of activities, the HTTP protocol version 1.1 allows a browser to request for all items in a Web page in the lifetime of the same TCP connection between the client and the server. Only when the browser receives all the items for that Web page, the server breaks the connection. This model results into a significantly less number of connections being used.

The browser then obtains a copy of the image and inserts it in the place of the link in the displayed document on the user's computer screen. The user does not realize that the image is not stored along with the HTML page, but that only a link to it is actually stored, and the image file is downloaded separately and displayed.

For linking an image with an HTML document, the **img** tag is used. For example, consider the following HTML statement:

```
<img src = "ana.gif">
```

This statement specifies that a file `ana.gif` contains an image that the browser should insert in the document at the place where this tag is written. In many cases, the file name also contains the full path name where the image file is stored. For simplicity, it is omitted here.

The file specified in an IMG tag is not stored as a normal text file. Instead, such a file contains binary data that corresponds to the pixels in an image. Normally, a file containing image, audio or video is stored and transmitted in a compressed fashion. Therefore, when that file is sent from the server to the client machine, the browser has to decompress and interpret it, i.e., display it at the appropriate location. Therefore, the browser must understand the format in which the image/audio/video file is stored. For each such type, there can be multiple compression algorithms and corresponding formats. How can any one browser be intelligent enough to understand all these formats?

To solve this problem, the idea of **plug-ins** was developed. A plug-in is a program that resides on the server along with the compressed file. This program understands the format of the file and therefore, has a method of decompressing it. This plug-in program is also sent from the server along with the compressed file to the client. The browser running on the client then uses this program to *interpret* the files, i.e., decompress and display it at the right location. Once a plug-in program is sent to the client, it is stored there. Therefore, it need not be sent again and again.

We can now make an interesting observation. If a Web page contains image, audio and video, there will be three hyperlinks corresponding to the three digitized files on the same or a different server. Therefore, to display the Web page, the client will have to make three connections to the same or different servers using the HTTP protocol, get all the three files in the memory of the client machine and then using different plug-in programs downloaded from the server, interpret them (i.e., display/play them). This is why Web pages with many heavy multimedia elements tend to be slow, as we must have noticed. It is also for this reason that to improve the performance, a good Web page design recommends that very heavy multimedia contents should be avoided.

Let us see a sample HTML code and the corresponding results of embedding an image file in a Web page, as shown in Fig. 20.16.

```
<html>
<head>
<title>IMAGE EXAMPLE</title>
</head>
<body>
<p>An image is shown below.</p>
<p>For this, the <b>IMG SRC </b>tag is used.</p>
<p>Here is the output:</p>
<p><a href="http://images/gift"></a></p>
</body>
</html>
```

Fig. 20.16 HTML code for displaying an image

The resultant Web page is shown in Fig. 20.17.



Fig. 20.17 Resultant page displayed in the browser as a result of the HTML page created in Figure 20.16

20.6 REMOTE LOGIN (TELNET)

20.6.1 Introduction

The **TELNET** protocol allows remote login services, so that a user on a client computer can connect to a server on a remote system. TELNET has two parts, viz., a client and a server. The client portion of TELNET software resides on an end user's machine, and the server portion resides on a remote server machine. That is, the remote server is the TELNET server, which provides an interactive terminal session to execute commands on the remote host. Once a user using the services of a TELNET client connects to the remote TELNET server computer, the keystrokes typed by the user on the client are sent to the remote server to be interpreted/acted upon to give an impression as if the user is using the server computer directly.

The TELNET protocol emerged in the days of timesharing operating systems such as Unix. In a timesharing environment, a common server computer serves the requests of multiple users in turns. Although many users use the server at the same time, the speed is normally so fast that every user gets an illusion that s/he is the only user, using that server computer. The interaction between a user and the server computer happens through a *dumb* terminal. Such a dumb terminal also has to have a microprocessor inside. Thus it can be considered to be a very primitive computer that simply has a keyboard, mouse and a screen and almost no processing power. In such an environment, all the processing is essentially done by the central server computer. When a user enters a command using the keyboard, for example, the command travels all the way to the server computer, which executes it and sends the results back to the user's terminal. At the same time, another user might have entered another command. This command also travels to the server, which processes it and sends the results back to that user's terminal. Neither user is concerned with the fact that the server is processing the requests from another user as well. Both users feel that they have exclusive access to the server's resources.

Thus, timesharing creates an environment in which every user has an illusion of using a dedicated computer. The user can execute a program, access the system's resources, switch between two or more programs, and so on. How this is possible, we elucidate ahead.

20.6.2 Local Login

In timesharing systems, all users log into the central server computer and use its resources. This is called **local login**. A user's terminal sends the commands entered by the user to a program called **terminal driver**, which is running on the central server computer. It is a part of the server computer's operating system. The terminal driver program passes the commands entered by the user to the appropriate module of the server computer's operating system. The operating system then processes these commands and invokes the appropriate application program, which executes on the server computer and its results are sent back to the user's terminal. This is shown in Fig. 20.18.

This forms the basis for further discussions about the TELNET protocol, as we shall study in the next section.

20.6.3 Remote Login and TELNET

In contrast to local login, sometimes a user wants to access an application program located on a remote computer. For this, the user logs on to the remote computer in a process called **remote login**. A user specifies the domain name or IP address to select a remote server with which it wants

to establish a TELNET session. This is where TELNET comes into picture. TELNET stands for **TERminal NETwork**. This is shown in Fig. 20.19. The step numbers shown in the figure followed by their descriptions depict how TELNET works, in detail.

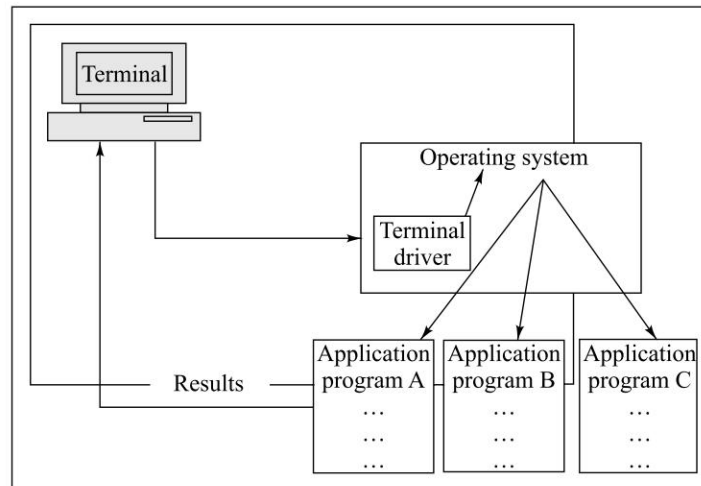


Fig. 20.18 Local login

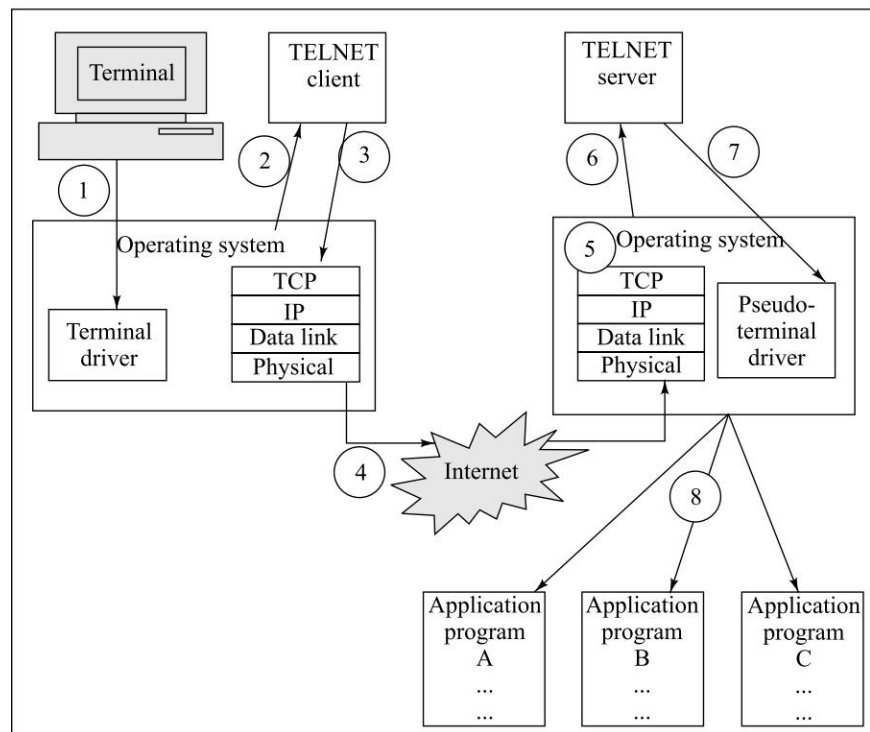


Fig. 20.19 Remote login using TELNET

1. As usual, the commands and characters typed by the user are sent to the operating system on the common server computer. However, unlike a local login set-up, the operating system now does not interpret the commands and characters entered by the user.
2. Instead, the local operating system sends these commands and characters to a **TELNET client** program, which is located on the same server computer.
3. The TELNET client transforms the characters entered by the user to a universally agreed format called **Network Virtual Terminal (NVT)** characters and sends them to the TCP/IP protocol stack of the local server computer. TELNET was designed to work between any host (i.e., any operating system) and any terminal. NVT is an imaginary device, which is common between the client and the server. Thus, the client operating system maps whatever terminal type the user is using to NVT. At the other end, the server operating system maps NVT onto whatever actual terminal type the server is using. This concept is illustrated in Fig. 20.20.

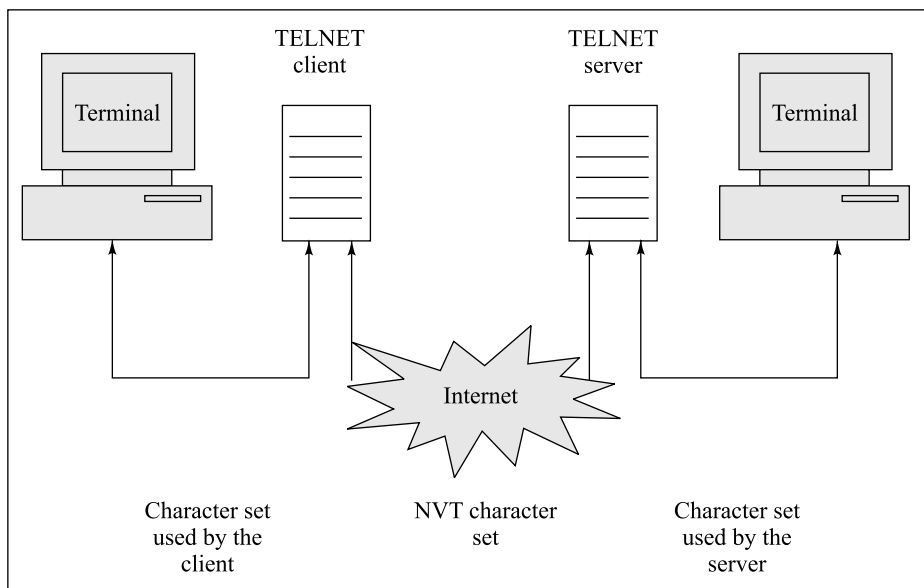


Fig. 20.20  *Concept of NVT*

4. The commands or text in the NVT format then travel from the local server computer to the TCP/IP stack of the remote computer via the Internet infrastructure. That is, the commands or text are first broken into TCP and then IP packets, and are sent across the physical medium from the local server computer to the remote computer. This works exactly similar to the way IP packets (and then physical hardware frames) travel over the Internet, as described earlier.
5. At the remote computer's end, the TCP/IP software collects all the IP packets, verifies their correctness/completeness, and reconstructs the original command so that it can hand over these commands or text to that computer's operating system.
6. The operating system of the remote computer hands over these commands or text to the **TELNET server** program, which is executing on that remote computer, passively waiting for requests from TELNET clients.

7. The TELNET server program on the remote computer then transforms the commands or text from the NVT format to the format understood by the remote computer. However, the TELNET server cannot directly hand over the commands or text to the operating system, because the operating system is designed so that it can accept characters only from a terminal driver and not from a TELNET server. To solve this problem, a software program called **pseudo-terminal driver** is added, which pretends that the characters are coming from a terminal and not from a TELNET server. The TELNET server hands over the commands or text to this pseudo-terminal driver.
8. The pseudo-terminal driver program then hands over the commands or text to the operating system of the remote computer, which then invokes the appropriate application program on the remote server.

The client using the terminal on the other side can thus access this remote computer as if it is a local server computer.

20.6.4 TELNET – A Technical Perspective

Technically, the TELNET server is actually quite complicated. It has to handle requests from many clients at the same time. These concurrent requests must be responded to in real time, as the users perceive TELNET as a real-time application. To handle this issue effectively, the TELNET server uses the principle of delegation. Whenever there is a new client request for a TELNET connection, the TELNET server creates a new child process and lets that child process handle that client's TELNET connection. When the client wants to close down the TELNET connection, the child process terminates itself. Thus, if there are 10 clients utilizing TELNET services at the same time, there would be 10 child processes running, each servicing one client. There would, of course, be the main TELNET server process executing to coordinate the creation and handling of child processes.

TELNET uses only one TCP connection (unlike FTP, which uses two). The server waits for TELNET client connection requests (made using TCP) at a well-known port 23. The client opens a TELNET connection (made using TCP) from its side whenever the user requests for one. The same TCP connection is used to transfer data and control characters. The control characters are embedded inside data characters. How does TELNET then distinguish a control character from a data character? For this, it mandates that each sequence of control characters must be preceded by a special control character called *Interpret As Control (IAC)*.

20.7 STATIC, DYNAMIC, AND ACTIVE WEB PAGES

20.7.1 Static Web Pages

All the Web pages that we have discussed so far are called **static Web pages**. The reason for this is simple. An HTML page is created by an application developer/designer and is stored on a Web server. Whenever any user requests for that page, the Web server sends back the page without performing any additional processing. All it does is to locate that page on its hard disk, add HTTP headers, and send back an HTTP response. Thus, the contents of the Web page do not change depending on the request—they are always the same (unless, of course, they are physically changed on the server's hard disk). Hence, the name *static*. Static Web pages follow a simple HTTP request-response flow as shown in Fig. 20.21.

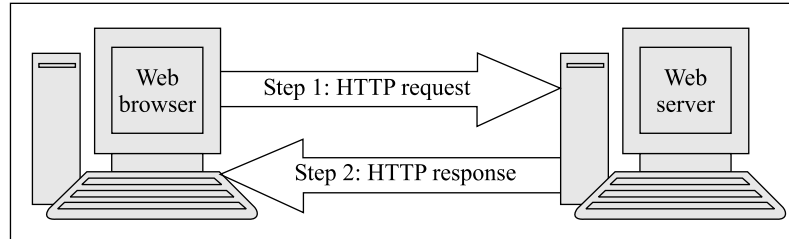


Fig. 20.21 *Static Web page*

20.7.2 Dynamic Web Pages

Static Web pages are not always useful. They are suitable for contents that do not change often. For instance, a static Web page would be a good candidate for showing the corporate information of an organization—the organization’s home page or the history of a country on a country’s home page. However, for information that changes quite often, for example, stock prices, weather information, news and sports updates, static Web pages would not serve the purpose. Imagine the situation wherein somebody (usually the person who maintains a Web site) has to physically change the Web page every 10 seconds to show the latest update of the stock prices! Clearly, it is simply infeasible to physically alter the HTML page so frequently.

Dynamic Web pages offer a solution to such problems. A dynamic Web page is in the true sense, dynamic! The contents of a dynamic Web page can vary all day depending on a number of parameters. For instance, they could change to reflect the latest stock prices or depending on whether a user A or B has asked for certain information, the contents could be filtered out and only those allowed for A or B could be shown. It is obvious that dynamic Web pages, therefore, are more complex than static Web pages. In fact, dynamic Web pages involve much more than only HTML. Creating dynamic Web pages involves server-side programming.

Conceptually, when a user requests for a dynamic Web page, the Web server cannot simply send back an HTML page unlike what happens in the case of a static Web page. Here, the Web server actually invokes a program that resides on its hard disk. The program might, in turn, access databases, perform transaction processing, etc. However, in any case, the program outputs HTML, which is used to construct an HTTP response by the Web server. The Web server sends the HTTP response thus formed, back to the Web browser. This is shown in Fig. 20.22.

For instance, a dynamic Web page could be written for reading the latest stock prices from a database and using them to respond to HTTP requests for that page. The actual process of updating the database with the latest stock prices need not anyway depend on the dynamic Web page. That could be handled totally separately.

As we can see, the major difference between a static Web page and a dynamic Web page is the involvement of an application program on the server’s side in the latter case. However, to reiterate, both static and dynamic Web pages have to return back HTML contents to the Web browser using the HTTP protocol, so that the browser can interpret them and display them.

Many tools allow the creation of dynamic Web pages. **Common Gateway Interface (CGI)** was quite popular dynamic Web page creation technology for many years. These days, however,

Microsoft's **Active Server Pages (ASP)**, Sun Microsystems's **Java Servlets** and **Java Server Pages (JSP)** are the most popular technologies for creating dynamic Web pages.

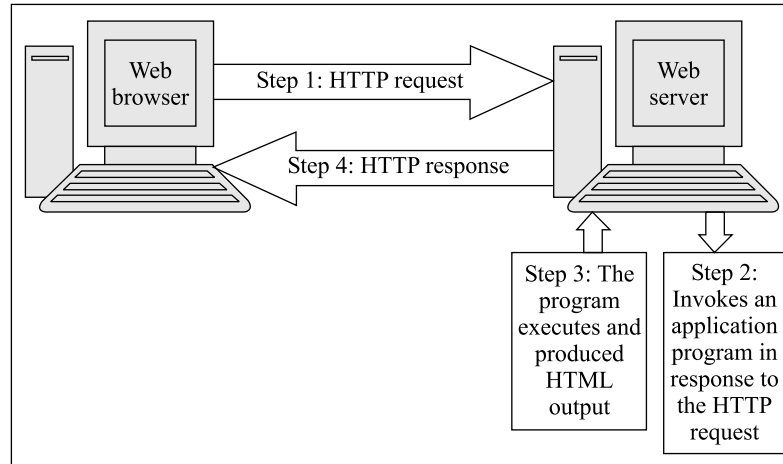


Fig. 20.22 Dynamic Web page

20.7.3 Active Web Pages

With the arrival of the popular programming language Java, **active Web pages** have become quite popular. The idea behind active Web pages is actually quite simple. When a client sends an HTTP request for an active Web page, the Web server sends back an HTTP response that contains an HTML page as usual. In addition, the HTML page also contains a small program that executes on the client's computer inside the Web browser. This is shown in Fig. 20.23.

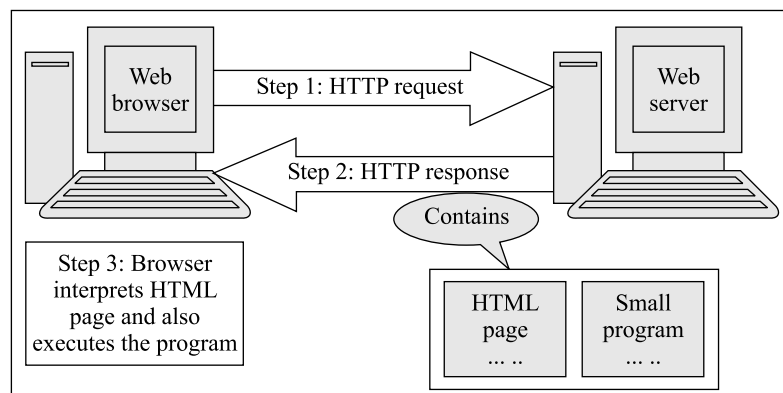


Fig. 20.23 Active Web Pages

Usually, the small program sent to the browser along with the HTML page is called **Java applet**. An applet is a client-side program written in the Java programming language that can be executed by a Web browser. Thus, a Web browser needs to have a Java interpreter to interpret the code of an applet and execute it on the client's side.

Applets can be used to perform a variety of tasks such as painting images, graphs, charts and other drawing objects on the client's browser screen. In addition, it can be used for other purposes such as requesting the Web page to automatically refresh after a fixed interval (say, every 30 seconds). Thus, we can construct active Web page containing stock prices in the HTML format and an applet that refreshes the HTML contents after every 30 seconds. Note that in order to do this, the applet would need to open a TCP connection with the server every 30 seconds. Since the client keeps requesting information automatically from the server (i.e., the client *pulls* information) after a specified interval, this technology is called **client pull**.

Microsoft has also implemented active Web pages technology with its **ActiveX controls**. ActiveX controls are conceptually quite similar to Java applets. The main difference between applets and ActiveX controls is that applets have a lot of restrictions on them, whereas ActiveX controls are reasonably free to do what they want. For instance, an applet cannot write to the client's hard disk. On the other hand, an ActiveX control has no such restrictions. Therefore, the general tendency is to trust applets more than ActiveX controls. However, these days, **signed applets** are allowed far more access to the client's computer, blurring the differences between applets and ActiveX controls.

Another significant difference between applets and ActiveX controls is that an applet is downloaded with an active Web page, executed inside the browser, and destroyed when the user exits that Web page. On the other hand, once downloaded, an ActiveX control remains on the client's computer until it is explicitly deleted. This every time downloading makes applets quite slow as compared to ActiveX controls.

SUMMARY

The World Wide Web (WWW) is the second most popular application on the Internet, after email. It also works on the basis of client-server architecture, and uses a request-response paradigm.

An organization hosts a Web site, consisting of Web pages. Anybody armed with a Web browser and wanting to access these Web pages can do so. Each Web site has a unique identifier called Uniform Resource Locator (URL), which is essentially an address of the home page of the Web site. The WWW application uses the Hyper Text Transfer Protocol (HTTP) to request for and serve Web pages. A client browser sends an HTTP request for a Web page to a Web server. The Web server responds with an HTTP response, that contains the HTTP response headers and the Web page.

A Web server is a program running on a server computer. Additionally, it consists of the Web site containing a number of Web pages. The contents of a Web page are written using a special tag language, called Hyper Text Markup Language (HTML). The browser first establishes a TCP connection with the server. To request a specific HTML Web page, a browser then sends an HTTP request that also contains the URL for the file, which the browser wants to retrieve from the Web server. The Web server locates that file on its disk, and sends it back to the browser as a part of the HTTP response.

There are various HTTP commands to request, upload, and delete Web pages that a browser can use. In response, the Web server sends a status code that indicates whether the HTTP request sent by the browser was processed successfully. This status code is also a part of the HTTP response sent by the server to the browser.

HTTP is a stateless protocol. This means that the TCP connection between a client and a server is established and broken for every Web page request. Even if the same client sends a request for

another Web page to the same server, a new TCP connection is established between the client and the server, as the previous connection would be broken by then.

A Web browser can understand and interpret HTTP and many times FTP. However, many old Web servers used other protocols such as Gopher, which are almost obsolete now. To access information stored on these servers with the help of a Web browser, proxy servers are used. A proxy server translates HTTP into FTP/Gopher/other older protocols and vice versa, so that the browser does not need to know all these different protocols. Proxies are also helpful in caching Web pages, so that repeated requests can be served locally.

Search engines (also called by many other names) facilitate searching of information on the Web. These are programs that constantly look for all kinds of documents on the Web, index and store them along with keywords in the documents in their databases, and allow users to search for keywords. On user requests, they search their databases and produce a list of matching entries.

The Hyper Text Markup Language (HTML) is used for creating Web pages. HTML is a presentation language that uses tags to demark different text formats such as boldface, italics, underline, paragraphs, headings, colors, etc. Initially, HTML contained plain text as it was primarily created for document exchange over the Web. However, these days, it is almost impossible to find a Web page that does not contain multimedia extensions to HTML such as pictures, graphics, sounds, and animation.

Most HTML tags come in pairs. The `<` symbol signifies the start of a tag, and the `</>` symbol signifies its end. For instance, the `<HTML>` tag indicates the start of an HTML page, and the `</HTML>` tag indicates the end of the page. HTML also supports the concept of hyperlinks. Using hyperlinks, one HTML page can point to other HTML pages. This allows a user to easily move across HTML pages with the help of a simple click of a mouse button.

In its simplest form, a Web browser is a program that sends HTTP requests to Web servers, receives HTTP responses from them and interprets (i.e., displays) the HTML pages received from the servers. However, these days, a Web browser is an extremely complex piece of software that performs many tasks apart from just communicating with the Web server and interpreting HTML pages. A Web browser also includes interpreters for modern Web technologies such as Java.

Modern browser architectures also allow users to invoke email and FTP clients from within browsers. This means that users have a single interface (the browser) to perform all activities such as Web browsing, email and file upload/download.

There are a number of multimedia file formats available. These formats are incompatible with each other. Therefore, simply adding multimedia data to HTML pages is not enough. The browser would not know how to display that image, as it cannot interpret it. To solve this problem, the concept of plug-ins was developed. A plug-in is a small program that can be downloaded from a Web server along with an image, onto the browser. The plug-in then interprets the image and shows it on the user's screen, on behalf of the browser. Thus, a variety of multimedia formats can be developed, and their plug-ins made available, so that anybody can access the multimedia files.

The TELNET protocol allows remote login services, so that a user on a client computer can connect to a server on a remote system. The remote server is the TELNET server, which provides an interactive terminal session to execute commands on the remote host. The TELNET protocol emerged in the days of timesharing operating systems such as Unix. In a timesharing environment, a common server computer serves the requests of multiple users in turns.

In timesharing systems, all users log into the central server computer and use its resources. This is called local login. In contrast to local login, sometimes a user wants to access an application program located on a remote computer. For this, the user logs on to the remote computer through a process called remote login.

In TELNET, the user's commands are not processed by the local operating system. Instead, they are directed to a remote server to which the user is connected. From the user's point of view, this does not make any significant difference as compared to what happens in local processing.

KEY TERMS AND CONCEPTS

Active Server Pages (ASP)	Plug-in
Active Web pages	Portal
ActiveX control	Process
Anchor tag	Proxy server
Common Gateway Interface (CGI)	Pseudo-terminal driver
Client pull	Remote login
Crawler	Search engine
Dynamic Web pages	Signed applet
Home page	Stateless protocol
Hyper link	Tag
Hyper Text Markup Language (HTML)	TELNET
Hyper Text Transfer Protocol (HTTP)	TELNET client
HTTP request	TELNET server
HTTP response	Terminal Client
Java	Terminal driver
Java applet	Uniform Resource Locator (URL)
Java Server Pages (JSP)	Web browser
Java Servlets	Web page
Knowbot	Web server
Link	Web site
Local login	World Wide Web (WWW)
Network Virtual Terminal (NVT)	Worm

QUESTIONS

True/False

1. URL may or may not identify a unique Web site/file on the Internet.
2. A Web page should be written in HTML.
3. A browser may or may not establish a TCP connection with a Web server before it sends an HTTP request.
4. The browser closes the TCP connection with the Web server.
5. The PUT command is used to send a file from a Web browser to a Web server for storing it there.
6. A proxy server is the same as an HTTP server.

7. Now HTML allows graphics, sound, and video.
8. A browser must be able to interpret HTML tags.
9. Usually every HTML tag has a pair, viz., starting tag and ending tag.
10. HTML uses the body tag for creating hyperlinks.
11. A Web browser is a very simple program.
12. Multimedia is supported on the Web.
13. The TELNET protocol allows remote login services.
14. The TELNET protocol emerged in the days of single-user operating systems such as DOS.
15. A pseudo-terminal driver pretends that the characters are coming from a terminal and not from a TELNET server.
16. JSP is an active Web page technology.
17. Static Web pages do not involve any programs on the server's side.

Multiple-Choice Questions

1. The main page of a Web site is generally called the _____.
 (a) chief page (b) main page
 (c) home page (d) house page
2. The world's first real Web browser was _____.
 (a) Mosaic (b) Internet Explorer
 (c) Netscape Navigator (d) None of the above
3. Web pages are created in the _____ language.
 (a) HTTP (b) WWW
 (c) Java (d) HTML
4. The portion after the words WWW identify a _____.
 (a) client (b) Web server
 (c) database server (d) application server
5. GET and PUT commands are used to _____ an HTML document.
 (a) download (b) upload
 (c) delete (d) modify
6. HTTP is called a _____ protocol.
 (a) stateful (b) stateless
 (c) state-aware (d) connection-oriented
7. The _____ command allows a client to remove a file from a Web server using HTTP.
 (a) GET (b) POST
 (c) UNLINK (d) DELETE
8. Proxy server is used to transform _____ protocol to _____ format.
 (a) TCP/IP, OSI (b) OSI, TCP/IP
 (c) Non-HTTP, HTTP (d) TCP/IP, HTTP
9. Generally, the closing HTML tag is indicated by the _____ character.
 (a) * (b) /
 (c) \ (d) @
10. The _____ tag can be used to create hyper links.
 (a) anchor (b) arrow
 (c) link (d) pointer

11. The main function of a browser is to _____.
 (a) compile HTML (b) interpret HTML
 (c) decompile HTML (d) interpret CGI programs
12. _____ is used to understand different multimedia formats.
 (a) Java (b) HTML
 (c) Plug-in (d) Compiler
13. TELNET uses the _____ concept for dealing with character differences.
 (a) NVT (b) floating point format
 (c) ASCII (d) EBCDIC
14. _____ Web pages involve server-side programs.
 (a) Static (b) Dynamic
 (c) Active (d) All of the above
15. Use of applets is a _____ technology.
 (a) client pull (b) client push
 (c) server pull (d) server push
16. ASP is promoted by _____.
 (a) Sun Microsystems (b) Netscape Corporation
 (c) Microsoft (d) IBM
17. For active Web pages to work, a browser must have _____.
 (a) Java compiler (b) CGI interpreter
 (c) CGI compiler (d) Java interpreter

Detailed Questions

1. Define the terms Web site, Web page, Web server, URL and home page.
2. What is the purpose of HTTP?
3. How does a Web browser work?
4. Describe the steps involved when a Web browser requests for and obtains a Web page from a Web server.
5. Why is HTTP called a stateless protocol? Why is it so?
6. Discuss any three HTTP commands.
7. What is the purpose of a proxy server?
8. Why is the job of search engines not easy?
9. Discuss the idea of HTML tags with an example.
10. Describe any three HTML tags.
11. What is meant by a hyperlink? How is it useful?
12. Describe the architecture of a Web browser.
13. How are optional clients useful?
14. How can email/FTP be invoked from a browser?
15. How can multimedia work on the Internet?
16. What are plug-ins? Why are they important?
17. What is the difference between local and remote login?
18. Why does TELNET protocol require a timesharing operating system?
19. Describe the terminal driver program.
20. What do TELNET client and server programs do?
21. Distinguish between static, dynamic, and active Web pages. In which situations would each of them be useful?

21 Multimedia Communications

21.0 INTRODUCTION

The term multimedia has evoked a lot of interest in the last few years. Earlier, computers were restricted to processing only text. However, with the capabilities of new hardware, they can now store, retrieve and display other forms of data such as images, sound, video, etc. Of course, computers play a key role in creating multimedia content in the first place.

All this keen interest in multimedia technology has led to a need for multimedia networking. Sending text data over the network is pretty simple. However, multimedia data tends to be large and complex. Moreover, newer multimedia applications such as computer telephony, video conferencing and video on demand mandate special protocols to be used for multimedia networking. We look at these issues and protocols in this chapter. Multimedia also requires the use of data compression techniques to save disk space and bandwidth demands. We have already examined these aspects earlier, and would not reiterate them here.

21.1 BASICS OF MULTIMEDIA

As we have discussed before, the concept of multimedia is relatively new to the world of information technology. The early history of computing was merely concerned with data processing. This meant that computer-based information was purely of textual form. However, with the progress of technology and the desire to provide better user interfaces, the Graphical User Interface (GUI) was born. Computers could now display not only text, but also images, pictures and video and play sound. As we have studied, computers work with only the binary number system. Character codes such as ASCII or EBCDIC are used to map alphabets, numbers, and special characters to the binary strings of numbers, and help humans and computers to interact with each other. Similarly, when we need to store or display images, computers need additional hardware and software to store these images in the binary form.

This convergence of media of information such as text, pictures, images, video and sound is called multimedia (abbreviation of *multiple media*). A multiple computer is the one that has capabilities of *handling* all these formats. By *handling*, we mean the ability to accept, store and retrieve these different forms of information. Internally, of course, a computer does not know any of this. Internally, the computer perceives all multimedia formats as binary. We need additional hardware and software capabilities to deal with multimedia. For instance, we need the

Pulse Code Modulation (PCM) technique to transform analog sound signals into their corresponding binary sound files, or we need animation capabilities for creating the illusion of continuous motion. We have studied all these concepts in Chapter 2, and will not revisit them.

Before we discuss multimedia, it is vital to deal with a few confusions. A book can be considered as a multimedia work, as it contains both text as well as images. However, in general, when people talk of multimedia, they mean two or more continuous media (e.g., audio and video together). Therefore, a book is practically not considered as a multimedia work, although technically it can be considered as such. This, however, raises a new complication. When we talk of video, does it mean video alone, or video *and audio*? Technically, the signals for video and audio are distinct. However, as a general perception, when someone talks of video, audio is implicitly considered. Therefore, when someone speaks about videoconferencing, audio is an inherent part of it.

Moreover, multimedia files can be big in size. This is because it takes a lot of bits, for instance, to represent the minute details and colours of a picture file or a video. The more the clarity desired, the more is the number of bits required for this purpose. For this purpose, the concept of data compression is used, wherein repeated bit patterns are replaced with an identifier. There are various algorithms to do this task, which lead to the various multimedia file formats (such as WAVE, JPEG, MPEG, GIF, etc.). Each format is nothing but a compression algorithm that specifies how a computer should interpret (i.e., compress and uncompress) a multimedia file. We have studied all these concepts in Chapter 6.

With this background, we shall focus our attention to multimedia communications in the rest of this chapter.

21.2 MULTIMEDIA APPLICATIONS

At a broad level, multimedia applications can be classified into three types, as shown in Fig. 21.1.

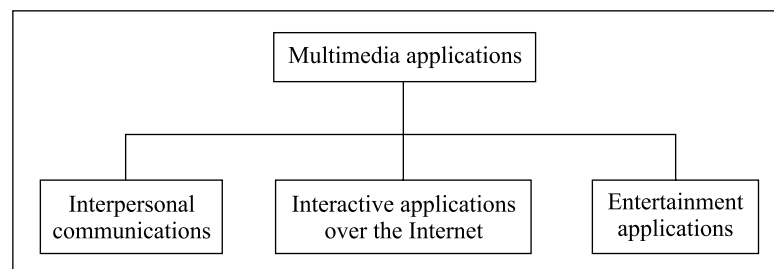


Fig. 21.1 *Multimedia application types*

Let us have a brief overview of these three types. The reason we need to study these is interesting. In each of these categories, there are certain applications that were initially designed for and which supported only the basic textual form. However, as technology progressed, these applications were upgraded to support additional multimedia formats as well. Interestingly, the same basic networking technologies are used to support these multimedia applications in addition to the basic textual applications.

21.2.1 Interpersonal Communications

Voice-based Communications

Interpersonal communications generally consists of speech, images, text or video. Traditionally, interpersonal communications consisted mainly of telephonic conversation between two parties. The technology to support that was in the form of PSTN or ISDN networks, as we have studied. These days, cellular telephony is catching up at a fast pace. Interpersonal multimedia applications consist of a multimedia PC that has a microphone and a speaker built in. The user can make or receive telephone calls using her/his PC, using this approach. This requires a telephone interface and associated software, called **Computer Telephony Integration (CTI)**. This provides many additional features to the user, such as creating her/his private directory of numbers, and initiating a call just by selecting the desired number on the computer screen. Going a step forward, telephony can be integrated with all other applications such as Customer Relationship Management (CRM), to provide better service to the customers and save processing time and costs.

Internet telephony can work in one of the two possible ways:

- **Computer-to-computer telephony** – Initially, the Internet supported only computer-to-computer telephony. However, now people can also use their telephone machines instead (as we shall study subsequently). In the case of a computer-to-computer telephony, the standard IP addresses of the computers connected to the Internet are used in a fashion that is very similar to the way they are used for data transfer. However, because the Internet is based on packet switching, both the computers need to have the required hardware and software to transform the analog speech signals from the microphone into digital bits first and then transform them into the packets at the speaker's end, and transform these received digital data packets back into analog voice signals at the listener's end. The protocol that facilitates these telephone calls over the Internet is called **Voice Over Internet Protocol (VOIP)**.

The way this works is as follows.

1. The caller looks up the name of the person to be called in a special directory. This directory maps the names of the people with their IP addresses (e.g., Name = Atul, IP address = 127.41.3.26).
2. The caller selects the name of the person to be called (in this case, Atul), and selects the appropriate option on the screen to make a call.
3. The call is directed to the computer of the called person using the called person's IP address (in this case, 127.41.3.26).
4. During the conversation, the software converts the voice into binary data files, which computers can understand. The computer data representing voice is also compressed, so as to save on the transmission size.

This concept is illustrated in Fig. 21.2.

- **Telephone-to-telephone telephony** – When a telephone user needs to make a call to a user using another telephone that is connected to the PSTN/ISDN, the circuit-switching mechanism is used underneath. When the Internet is used for this purpose, an internetworking unit called **telephony gateway** is used at both ends. The caller dials a special number dedicated to this gateway, and then the telephone number of the called party. The gateway establishes a connection with the gateway that is closest to the called party over the Internet. Thus, the conversation goes like this between the caller and the caller's gateway; the PSTN/ISDN lines

are used. Between the caller's gateway and the called party's gateway, the voice is sent as packets over the Internet. The line between the called party's gateway and the called party's telephone again consists of the local ISDN/PSTN network. Thus, the caller pays local telephone charges (plus some service charges) for making a telephone call anywhere in the world.

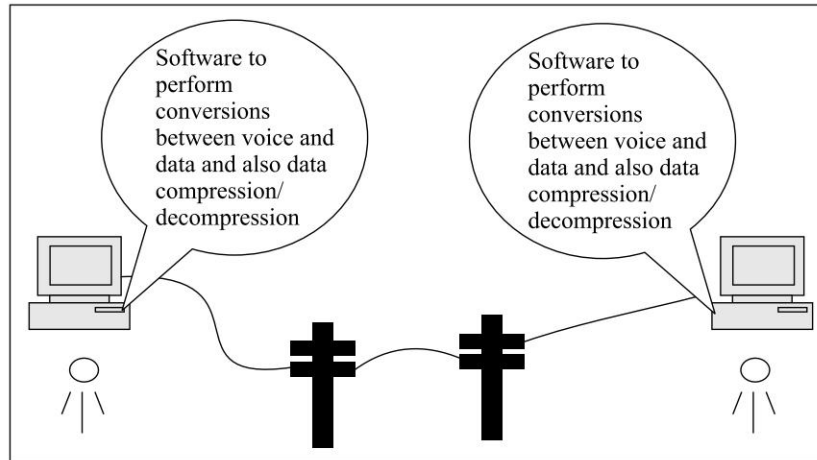


Fig. 21.2 *Computer-to-computer telephony*

The logical view of this is as shown in Fig. 21.3.

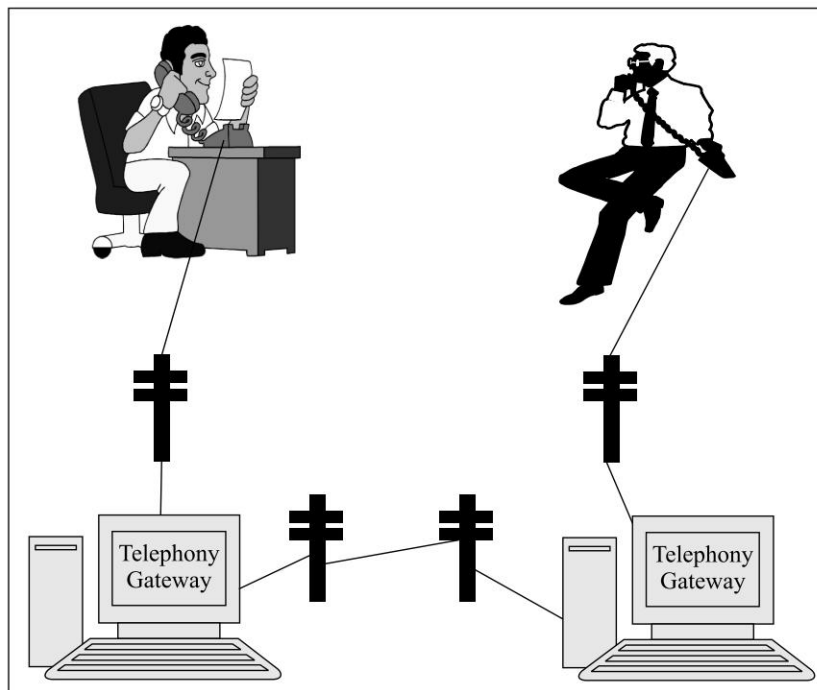


Fig. 21.3 *Telephone-to-telephone telephony*

Additional services such as voicemail and teleconferencing are also facilitated. In a voicemail application, if the called party is not available, the caller can leave a message in the voice mailbox of the user, which is controlled by a voicemail server. In teleconferencing, multiple parties can speak with each other at the same time. A central unit called audio bridge coordinates such a call between more than two parties.

Image-based Communications

The most common form of interpersonal communication that involves images is the facsimile or fax. This is an exchange of the electronic image of a document. Normally, both parties have a fax machine for this purpose. To send a fax, the sender keys in the fax number of the recipient just as the caller in a telephonic conversation dials the telephone number of the called person. At this juncture, a circuit is established between the two fax machines similar to the way a circuit is established between two telephones during a telephone call. The two fax machines then exchange operational parameters after which the sending fax machine starts to scan and digitize each page of the document to be sent in turn. Both the fax machines have a built-in modem. As the sending modem receives the binary values corresponding to the scanned image, it transforms them into analog signals and sends them to the receiving modem. The receiving modem receives these signals, transforms them back into digitized values, which cause the printing of a page at the receiver's end. Nowadays, computers can be used to receive/send faxes, instead of fax machines. Computers essentially perform the same task as fax machines, i.e., a computer stores the image in the form of digital bits which a user then can see on the screen or print if s/he wishes.

Text-based Communications

The most common form of interpersonal text-based communication is electronic mail (email). We have discussed the email mechanism and technology in great detail. Therefore, we shall not discuss it again here.

21.2.2 Interactive Applications Over the Internet

Apart from a number of interpersonal communication applications, the Internet also supports a range of interactive applications. As we have discussed, this is in the form of a number of Web pages, which have cross-references to each other in the form of hyperlinks. Web pages can now contain images, sound and video apart from simple text.

Interactive applications over the Internet are usually in the form of some multimedia additions to the basic contents of a Web page. For instance, an organization can set up its shopping cart, and provide videos of its products or multimedia demos to the buyers so as to give them a real-life look and feel. A user has to usually click on a hyperlink to view such additional video contents. The other forms of multimedia (e.g., images) get downloaded automatically along with the Web page, and the user does not have to take any action. Regardless, for many multimedia data types (such as sound and video), the user's computer must have an additional piece of software called plug-in. If the right plug-in is not available, usually it can be downloaded over the Internet and used.

21.2.3 Entertainment Applications

At a broad level, all entertainment applications can be classified into one of the two major types: (a) Movie or **Video-on-demand**, and (b) **Interactive television**.

Video-on-demand

The first type (movie or video-on-demand) is actually quite similar to the interactive applications discussed above. However, the major difference is that when a video is stored on a Web server with the intention of allowing users to download and view it, usually its resolution and sound quality are superb. High resolution and stereo quality sound is the norm. A digitized movie/video with sound needs a minimum data rate of 1.5 Mbps. Therefore, only high-speed connections (such as cable modems or ADSL networks) can support video-on-demand. The ordinary telephone line would be too slow for such an application.

In simple terms, video-on-demand means the ability of a home user to select a movie at any time by using the television set's remote control or the computer's mouse, and view it. To cater to the demands of video-on-demand, specialized software and hardware mechanisms are needed. The following aspects are crucial:

1. The Web site that facilitates video-on-demand must have a **video server** set-up. It is a very powerful server with a lot of disk space and memory. It stores a large number of video files and allows users to play them back as and when they like.
2. A **distribution network** consists of high-speed data links between the video server and the home users. This network must be capable of transmitting data at very high speed in real time. Usually, optical fiber is used for this purpose. Of course, the optical fiber does not generally extend to each of the home users, and terminates at the local junction box (common point for many home users in a locality). The connection between the junction box and an individual user's home is called the last mile, as we have seen, and this generally consists of twisted pair telephone lines, ADSL or cables.
3. Every home user needs an additional **set-top-box**, where the ADSL or cable connection terminates. A set-top-box is actually a special-purpose computer, which has certain chips for video decoding and decompression. It also has a CPU, RAM, ROM and an interface to ADSL or the cable. Of course, an alternative is to use an existing computer itself. Of course, a dedicated set-top-box is always better, since it is designed with the sole purpose of viewing multimedia content.

In this scheme, a user has controls similar to a VCR. The user can pause the picture, fast forward or rewind as s/he wishes. Pausing is easy, as the browser simply informs the server at which frame the client has pressed the pause button. The server has to note it down, and resume when the client wishes to resume.

However, implementing fast forward and reverse is not quite easy. For instance, suppose that the user is currently watching frame 45,000 and fast-forwards it by 1000 units. The server must locate frame number 46,000 and start playing from that point onwards. Recall that multimedia files are in compressed formats (such as MPEG) on the server and are sent as they are to the client. The client decompresses the files and plays them. This means that the server has to decompress the file, locate frame 46,000 and start sending the compressed version from that point onwards. This puts a lot of demand on the server operating system and the underlying network. To add to the problems, the audio compression is done separately from the video compression. Thus, for each video frame, the server must locate the corresponding audio frame.

Researchers are still working on schemes that would solve all these issues in a neat fashion. However, until then, video-on-demand is very attractive, but not yet completely mastered.

Near Video-on-demand

The problem with video-on-demand is that users can request for a movie at any given moment, and fast forward or rewind it. So, if there are 1000 users, all requesting for the same movie at about 9 pm, it is quite likely that no two users send their request at precisely the same point of time. Therefore, they cannot share a video stream (more so because one may want to fast forward, whereas another may not.). Therefore, for supporting 1000 users, the server operating system has to play 1000 copies of the same movie file, using techniques such as multitasking or multithreading, as appropriate. This puts a heavy burden on the server operating system, as well as on the network traffic.

A better technique, called **near video-on-demand**, solves this problem elegantly. The word *near* indicates that although this is very similar to the technique video-on-demand, it is not the same as video-on-demand. There are subtle, and yet crucial differences. What the server does in near video-on-demand is to start playing a stream (i.e., copy) of the same movie file after a predetermined time interval (say, 5 minutes). Thus, the first stream of the movie will start at 9 p.m. The second stream will start at 9.05 p.m. The third stream will start at 9.10 p.m., and so on. This also means that at 9.10 p.m., the first stream would have progressed 10 minutes, and the second stream would have progressed 5 minutes. Since there are 12 five-minute slots in an hour, at the end of the hour, we will have 12 streams of the movie running (the first one started at 9 p.m., the second started at 9.05 p.m., and so on, and the last started at 9.55 p.m.).

Note that no matter how many users are connected, the server has to worry about only 12 streams. This is far better for the server. From a user's perspective, what does this mean? A simple analogy is to compare owning a car versus traveling by a public bus. If you own a car, whenever you want to go out, you simply drive out. However, if you have to travel by a bus, you must wait at the bus stop until the first bus arrives. Video-on-demand is similar to having a car—you request for a movie as and when you please. In contrast, near video-on-demand is similar to traveling in a bus—you must wait until the next bus arrives. Clearly, the first option is better for a user, but not desirable for the server, whereas the second option is far better for the server, although a user might have to wait for the next stream to start. For instance, in the near video-on-demand technique if a user wishes to start watching a movie at 9.02 p.m., there is no choice for the user but to wait for three minutes when the next stream begins at 9.05 p.m. Of course, the user can opt to view the older 9 p.m. stream and lose the first two minutes of action (and revisit that later, when s/he completes watching the movie).

Quite clearly, in this scheme, a user does not have VCR-like controls. Thus, a user cannot pause, fast forward or rewind a movie. Thus, if a user misses some portion of the movie (perhaps because the door bell rang and s/he had to attend the visitor), s/he has to note which portion s/he has missed, and see an appropriate stream that starts at a later point in time.

Interactive Television

For interactive television, the user's computer must have an additional unit called set-top-box. The main feature of interactive television is that unlike the normal television (which is simplex), this is duplex. This means that not only does the user get to see what is going on a particular television channel, but can actively participate in the programs. For instance, the user can vote, participate in a game, do home shopping, and so on.

Having discussed the user-side of multimedia applications, let us now take a look at the technical side of these applications.

21.3 MULTIMEDIA PROTOCOLS

It is needless to say that TCP/IP is the underlying transmission protocol governing any communication over the Internet, and multimedia data is no exception. With that assumption firmly in place, we shall concentrate only on the multimedia-specific aspects of this transmission technology.

When an application involves the transfer of a real-time audio/video stream, the timing information required by the receiver to output the packet stream at the desired rate is provided by specific protocols. The **Real-time Transport Protocol (RTP)** is used for the basic transmission. Another protocol called **Real-time Transport Control Protocol (RTCP)** is used for more system-level processing, as we shall study. Conceptually, we can think of RTP as similar to the data connection in the File Transfer Protocol (FTP). Similarly, RTCP can be roughly equated to the control connection in FTP.

Let us now discuss these two protocols.

21.3.1 Real Time Protocol (RTP)

The process involving the use of RTP is as follows:

1. On the sender's end, the audio is digitized using a special device called coder-decoder (codec). A codec accepts analog audio/video signals, and transforms them into the corresponding bit patterns. We have studied this in the previous chapters.
2. After digitization, the bit streams are handed over to the RTP layer.
3. The RTP layer adds its own headers, and hands over the modified packets to the UDP layer.
4. The UDP layer adds its own headers as usual, and hands the UDP packets to the IP layer.
5. The IP layer adds its own headers as usual, and sends the IP packets to the receiver over the transmission medium in the form of the appropriate signals (e.g., voltage pluses).

At the receiver's end, the opposite process takes place. This is shown in Fig. 21.4.

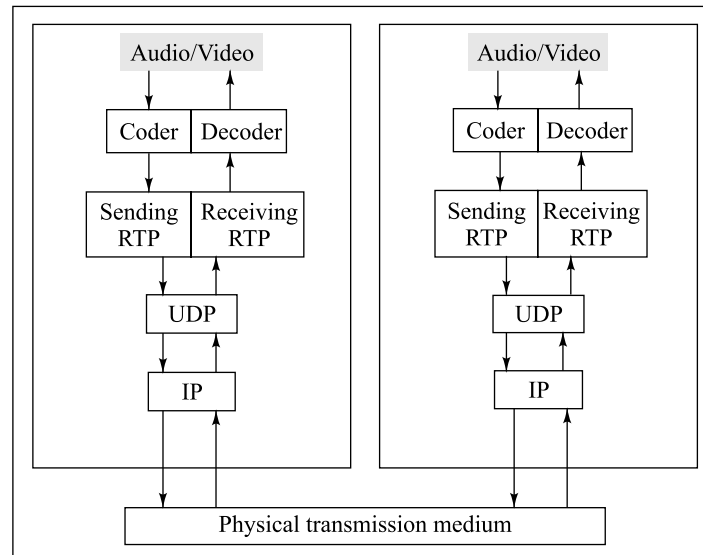


Fig. 21.4 *Real Time Protocol (RTP)*

The resulting bit streams at each stage are shown in Fig. 21.5.

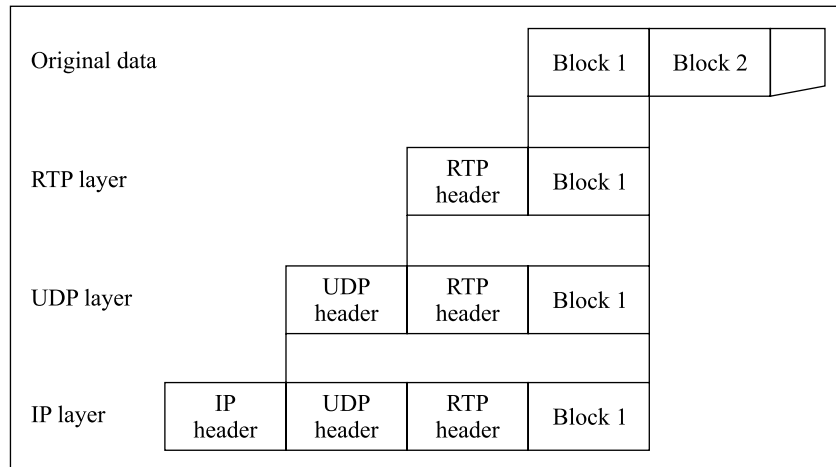


Fig. 21.5 Transformation of audio data into IP datagrams

We would now see why RTP is really required in this scheme. As we know, UDP is an unreliable transport layer protocol. This means that UDP does not guarantee the correct delivery of a packet to the destination. The packet may or may not reach the destination. This also means that there is no mechanism for (a) re-sequencing of packets that arrive out of order, (b) detecting duplicates and rejecting them, and (c) identifying missing packets and retransmitting them.

Missing packets may not be so much of concern in audio, as long as the proportion of missing packets to the packets arrived correctly is not high. Similar is the case of duplicate packets. However, the major concern is packets arriving out of order. More specifically, a slightly distorted audio is ok, but hearing musical bits or words that were supposed to be heard after some other bits or words is quite clumsy, and clearly not acceptable. Ensuring that these situations do not occur is the responsibility of RTP. Thus, RTP supplements the unreliable UDP protocol by adding certain checks and mechanisms to ensure the consistent delivery. In order to understand how RTP achieves this, we need to discuss the RTP packet layout, which is shown in Fig. 21.6.

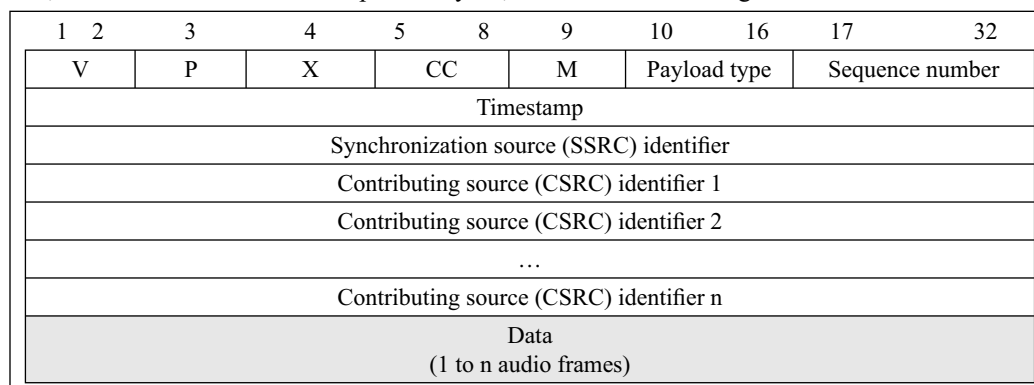


Fig. 21.6 RTP packet format

Let us now discuss the various fields in a RTP packet, as shown in Table 21.1.

Table 21.1 *RTP packet fields*

Field	Description
V (Version)	This field indicates the version of RTP in use.
P (Pad)	This is a pad bit.
X (Extension)	This flag allows extensions to the basic header to be defined in the future.
CC (CSRC count)	This 4-bit field identifies the number of CSRC (discussed subsequently). Since this is a 4-bit field, there can be a maximum of 15 CSRCs in a RTP packet.
M (Marker)	The bit stream produced by different codec is made up of a sequence of blocks or frames. Each such frame starts and ends with a unique delimiter. This field identifies the boundaries of a frame.
Payload type	This field identifies the type of coder used for coding the packet. There are multiple types of coders available in the market.
Sequence number	This field is used to detect packet duplications, out-of-sequence packets or packet losses. When the receiver detects a packet loss, it uses the contents of the last correctly received packet in its place. To overcome the issue of out-of-order packets, the receiver buffers a few packets before playing their contents. Duplicate packets are simply dropped.
Timestamp	This field identifies the time when the packet was created. It is used to determine the current mean transmission time and the level of jitter that is being experienced. This information is crucial in establishing the Quality Of Service (QoS) of the transmission system. It also helps in the sender changing the speed/compression algorithm used, if required, or for the receiver to adjust the buffer size.
Synchronization source (SSRC) identifier	This field identifies the source device that has produced a given packet. For instance, in a videoconferencing call, a number of devices (such as microphones, cameras, computers, etc) could be generating packets. This field identifies which of these has produced the contents of the packet. The receiver uses this information to output it to the correct destination device interface.
Contributing source (CSRC) identifier	In a multicast call or session, each participant that actively contributes to the session, instead of simply listening passively, is called a CSRC. It is identified by using a 32-bit number (which is typically the IP address of the computer). Since there can be many packets from multiple sources, a receiver should be in a position to relate a packet to a particular sender. This field helps in establishing that link.

21.3.2 Real-time Transport Control Protocol (RTCP)

As we have mentioned before, the RTP protocol deals with the transmission of individual streams of digitized data related to a multimedia call/session. The Real-time Transport Control Protocol (RTCP) adds the system-level functionality to the related RTP. Examples of these are the transmission of the Quality of Service (QoS) information and the provision of the necessary help to RTP to integrate and synchronize the individual packet streams together and send them. RTCP can also provide user level details. For instance, RTCP is used to inform all the users who are the other participants

before a call is initiated by identifying their names and other details such as email ids. Thus, even end users access RTCP information.

RTCP works alongside RTP and shares information with it. However, in order to make it independent of RTP, RTCP runs on its own dedicated UDP port. The RTCP software pieces of all the participants exchange information with each other periodically. Each message is sent inside an RTCP packet to the same network address, but with a different port number (which is the RTCP port number).

The overall schematic of RTCP along with RTP is as shown in Fig. 21.7.

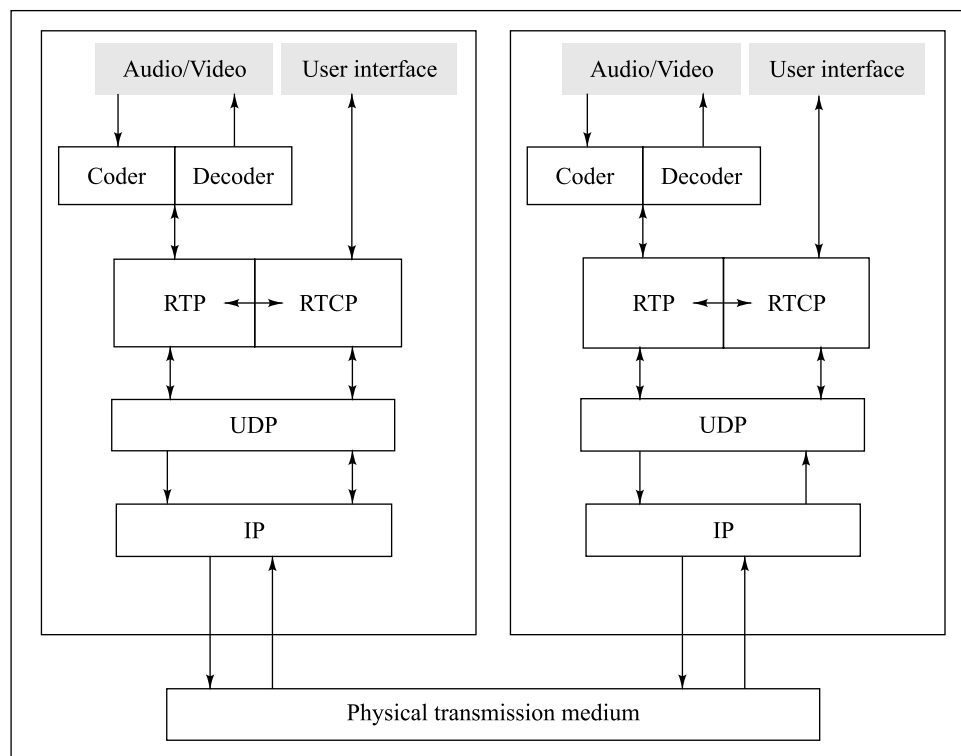


Fig. 21.7 Real-time Transport Control Protocol (RTCP)

As the figure shows, the end users can specifically access RTCP. In addition, RTCP interacts with RTP and has its own connection with UDP. The typical situations where RTCP is used are as follows.

1. **QOS data** – The number of lost packets, the level of jitter and the mean transmission delay are constantly computed by RTP for each received packet. The corresponding RTCP sends a message containing the related information to all other participating RTCPs periodically. Each RTCP then performs a system-level function, such as modifying the resolution or the compression algorithm used or the size of the buffer.
2. **Integrated media synchronization** – Applications that contain audio, as well as video streams need proper mixing, integration and synchronization. A common system time clock is used

for this purpose, which is coordinated by RTCP. Usually, the initiator of the call/conference provides this functionality. However, in some cases, an independent time server is also used.

3. **Participation reports** – These features are useful during a call. For example, a participant wishing to leave a call can indicate this fact to the other participants. This is done by the RTCP of the person who is leaving the call. The participant enters an appropriate message, which is sent via that person's RTCP interface to the RTCP interface of all the other users. Each RTCP then usually displays this information on the screen for the user's action.
4. **Participation details** – Details such as user names, email addresses, phone numbers, etc., regarding each participant are sent to all the other users in the form of a RTCP message. This way, each participant comes to know about all the other co-participants in the conference/call.

21.4 SESSION INITIATION PROTOCOL (SIP)

RTCP was designed to provide basic control functionality. RTCP does not provide features such as explicit membership control and session set-up. The designers thought that a separate session control protocol should take care of these issues. We shall discuss this protocol now.

The **Session Initiation Protocol (SIP)** is a session layer control protocol, which can be used to establish, change and close multimedia sessions/calls with one or more participants. A multimedia session in this context could be, for instance, an Internet telephony call or a multimedia videoconference. The participants in a session can be people or media devices. SIP provides support for user mobility by using the proxy approach and by redirecting the requests to the user's current location. SIP is independent of the underlying transmission protocol (i.e., it can be TCP or UDP).

SIP is a text-based protocol that is designed on the client/server paradigm. A typical SIP transaction involves a request from a client and responses from one or more servers. This is similar to the way SIP operates. A client sends one of the following requests: INVITE, ACK, OPTIONS, BYE, CANCEL and REGISTER. The most commonly used request is INVITE, which is used to initiate a call.

An SIP implementation consists of two portions: *user agents* and *network servers*.

- The **user agent** provides a user interface to an end user. The user agent further contains two sub-portions: (a) a protocol client, called User Agent Client (UAC), which is used to initiate a call, and (b) a protocol server, called User Agent Server (UAS), which is used to answer a call. The UAC and UAS together constitute a client/server mechanism.
- A **network server** performs call routing. That is, it identifies a desired user within a network. A network server can be of two types: (a) a proxy server accepts a request, determines which server to contact, and forwards the request to that server, and (b) a redirect server does not forward a request, and instead provides the client with the actual address of the server, so that the client can contact it directly.

To establish a call, an INVITE request needs to be sent to the UAS of the appropriate user. Generally, the IP address or the host name of that user is not known. Consequently, this information has to be derived from other pieces of information, such as the email address or telephone number. The REGISTER call sends a user's location information to a SIP server. This is useful in routing packets to that user. BYE terminates the session between two users. OPTIONS requests for information about the capabilities of a party to be called, without needing to set up a call for this purpose. ACK helps in a reliable message exchange mechanism, and CANCEL terminates a pending request.

SUMMARY

Multimedia means multiple media. Computers processed only textual data until a few years back. Now, they also deal with other forms of data such as images, sound and video. These additional forms have led to the idea of multimedia computing. Multimedia networking takes the idea one step further, and deals with the transmission of multimedia data over computer networks.

Multimedia applications can be classified into three broad categories, viz., interpersonal communications, interactive applications over the Internet, and entertainment applications. Some of the most popular multimedia applications are Internet telephony, video-on-demand, and videoconferencing.

The Real-time Transport Protocol (RTP) is used for basic transmission of multimedia content. The Real-time Transport Control Protocol (RTCP) is used for more system-level processing. The Session Initiation Protocol (SIP) can be used to establish, change and close multimedia sessions/calls with one or more participants.

KEY TERMS AND CONCEPTS

Computer Telephony Integration (CTI)	Session Initiation Protocol (SIP)
Distribution network	Set-top-box
Interactive television	Telephony gateway
Near Video on demand	Video-on-demand
Real-Time Protocol (RP)	Video server
Real-time Transport Control Protocol (RTCP)	Voice over Internet Protocol (VoIP)

QUESTIONS

True/False

1. Multimedia means only text and images.
2. Computer telephony integration is useful in CRM applications.
3. Video-on-demand needs a video server.
4. Telephone-to-telephone telephony requires Voice over IP.
5. Real-time Transport Control Protocol (RTCP) is used for the basic multimedia content transmissions.
6. Real-time Transport Control Protocol (RTCP) is related to the Quality of Service (Qos) parameters.
7. The Session Initiation Protocol (SIP) works only with UDP.

Multiple-Choice Questions

1. Multimedia is _____.

(a) only text	(b) only text and audio
(c) only text, and video	(d) more than one medium

2. _____ is very important in multimedia.
 - (a) Data compression
 - (b) Data encryption
 - (c) Error checking
 - (d) None of the above
3. Near video on demand puts _____ demands on a multimedia application as compared to a video on demand application.
 - (a) more
 - (b) same
 - (c) less
 - (d) undefined
4. Real Time Protocol (RTP) provides _____.
 - (a) unreliable delivery
 - (b) reliable delivery
 - (c) no error checking
 - (d) physical bit transmission layer
5. _____ is used to inform all the users who are the other participants before a call is initiated.
 - (a) RTP
 - (b) SIP
 - (c) RTCP
 - (d) None of the above
6. _____ is used to establish, change and close multimedia sessions/calls.
 - (a) RTP
 - (b) SIP
 - (c) RTCP
 - (d) None of the above

Detailed Questions

1. Define the term multimedia.
2. Why is data compression important in multimedia applications?
3. What is the difference between *video-on-demand* and *near video-on-demand*?
4. What are the ways to make telephone calls using the Internet?
5. Explain the purpose of RTP, RTCP and SIP.

Appendix A

Internet Protocol Version 6 (IPv6)

IPv4

Designed in the 1970s, the Internet Protocol (IP) provides host-to-host communication between computers on the Internet. IP has been extremely successful. It has made possible, handling of heterogeneous networks in the Internet. It defines a uniform packet format (called IP datagram), and a packet transmission mechanism. The current version of IP is 4, which is why IP is called IP Version 4 (IPv4). Although IP has been widely accepted as an extremely efficient network layer protocol, several concerns regarding IP have also been raised. These concerns are as follows.

1. A computer on the Internet is identified using its IP address. The IP address consists of two parts, viz., network id and host id. There are five classes of the network id, A through E. This scheme is not the most efficient one. For example, one class A network can address 16 million hosts. Thus, when an organization obtains a class A network id, it has an exclusive control over all the 16 million IP addresses under that network id. On the contrary, if an organization chooses class C, only 256 IP addresses are available to it, which is too small a number!
2. Since the IP address is 32 bits long, the IP address space is likely to get exhausted soon. The number of hosts connected to the Internet roughly doubles every year, and the IP address space cannot cope up with this increase. When it was designed, very few computers were connected to the Internet, and such an exponential growth of the Internet was not envisaged.
3. IPv4 does not address the issues of Internet security—no encryption/decryption mechanisms are built into the protocol itself.
4. Modern Internet application demand real-time audio and video support, which is not provided in IPv4.

Considering these drawbacks, the **Internet Protocol Version 6 (IPv6)**, also called **Internet Protocol Next Generation (Ipng)** was developed. Version 5 of IP was an academic interest, and was called ST, which never entered the commercial arena.

FEATURES OF IPv6

IPv6 retains the basic features of IPv4, which have made the latter such a successful protocol. IPv6 is connectionless (each datagram is independent, and contains the source and destination IP addresses) like IPv4, and also retains the other IPv4 options. In addition, IPv6 offers the following features.

1. **Address size** – IPv4 addresses are 32 bits long. In contrast, the IPv6 address space is huge, because it contains 128 bit addresses. It is expected to serve the growing Internet for several decades.
2. **Support for audio/video** – IPv6 facilitates a provision by which the sender and receiver can establish a high-quality transmission path via the underlying network over which datagrams can be sent. This will be very useful for audio/video applications that need real-time transfer of bits.
3. **Header format** – The datagram header format of IPv6 is completely different from the datagram header format of IPv4 with many fields changed in terms of structure or replaced completely.
4. **Extension headers** – IPv4 has a single header format for all datagrams. In contrast, IPv6 has many types of headers. A datagram contains the base header, followed by zero or more extension headers, followed by packet data.

IPv6 DATAGRAM FORMAT

The IPv6 datagram format is as shown in Fig. A.1 below.

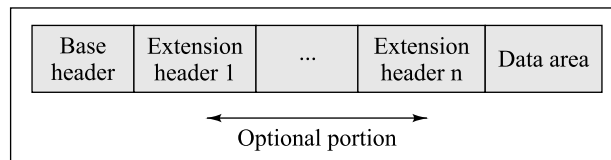


Fig. A.1 *IPv6 datagram format*

The base header itself is shown in Fig. A.2.

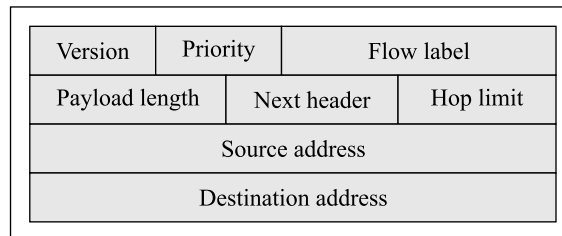



Fig. A.2 *IPv6 base header format*

The base header fields are described in Fig. A.3.

THE NEED FOR MULTIPLE HEADERS

Why does IPv6 provide for multiple headers? IPv4 does not have any such provision. There is only one header in an IPv4 datagram. Why is this changed in IPv6? Well, there are two main reasons behind this. The first reason is to allow IPv6 to reduce wastage, and the second is to make it flexible.

Field	Size	Description
Version	4 bits	Always contains 6 to indicate that this is IP Version 6.
Priority	4 bits	Defines the routing priority of the datagram with relation to the current network conditions.
Flow label	3 bytes	Used with new applications that demand performance guarantees to be met. This field associates a datagram with a particular network path. The flow label consists of two parts, viz., (a) The <i>traffic class</i> specifies the general needs of the datagram (e.g., to send interactive traffic such as keyboard strokes or mouse movements, we expect low delays) and (b) The <i>specific path</i> identifies a specific network route to be used for real-time data such as audio and video (because it guarantees very fast response times).
Payload length	2 bytes	Depicts the total length of the IP datagram, excluding the base header.
Next header	8 bits	Defines the type of information that follows the current header. If the next portion is an extension header, this field specifies the type of the extension header (e.g., fragmentation or authentication). If the next portion is data, it specifies the type of data (e.g., TCP data).
Hop limit	8 bits	Specifies the hop limit (how long can a datagram survive), same as the TTL field in the IPv4 header.
Source address	16 bytes	Identifies the source host's IP address (note that is 16 bytes, or 128 bits, unlike the 32-bit IPv4 address).
Destination address	16 bytes	Identifies the destination host's IP address (note that is 16 bytes, or 128 bits, unlike the 32-bit IPv4 address).


Fig. A.3 IPv6 base header fields

As we discussed, IPv6 provides many additional features such as support for real-time audio/video services. Thus, we must denote somewhere that the current IPv6 datagram contains bits corresponding to real-time audio/video, which need special treatment (e.g., fast delivery). However, the main purpose of a IPv6 datagram still continues to be carrying user data. Datagrams carrying user data generally do not require any special treatment. Therefore, they do not need any additional header information, unlike the special datagrams that carry real time audio/video. Thus, we have IPv6 serving the needs of varying data types. If we store information about all the possible data types and their associated overhead information in a single header, that header, and therefore, the IPv6 datagram, would become very bulky. Instead, the designers of IPv6 have selected an approach of using one base header that is common for all, and then additional headers as and when they are required by special applications/data types. This reduces wastage of space as well as helps reduce transmission bottlenecks.

The second advantage provides flexibility to the designers to extend the IPv6 datagram to support new features/applications/data types. For example, suppose there is a new application tomorrow that needs to use IPv6 as its network layer protocol, and it also has its own overhead data (feature details) that must be carried by the IP datagram header. If we use a single header, the IPv6 datagram format, and therefore, its size would have to undergo a change. Instead, by requesting each application to have its own header, which can be inserted into the basic datagram, any new features/applications/data types can be easily supported by IPv6.

IPv6 ADDRESSES


Like an IPv4 address, an IPv6 address uniquely identifies a computer on the Internet. Thus, when a computer connected to the Internet wants to send a data packet to any other computer on the Internet, the IP software of the sending computer must encapsulate the data packet inside an IP datagram, insert its address in the *source address* field of the datagram, the receiving computer's address in the *destination address* field of the IP datagram, and send the IP datagram along the Internet transmission path. However, the IPv6 addressing scheme differs substantially from the IPv4 addressing scheme in many ways, as described below.

The fundamental difference between IPv6 addressing and IPv4 addressing is the address length. Whereas IPv4 works with 32-bit IP addresses, IPv6 uses 128-bit IP addresses, making the address space extremely large.

Like IPv4, IPv6 also defines an addressing format that consists of a network id and a host id. However, unlike IPv4, IPv6 does not define any classes. Thus, it is simply not possible to determine how many bits are reserved for the network id portion, and how many bits are reserved for the host id portion simply by looking at the IP address anymore. Recall that an IPv4 address starts with a class id. Each class id has a unique bit pattern. By looking at the bit pattern, we know what is the class id. From there, we also know what is the network id and what is the host id, because the class id provides information as to how many bits of the remaining IPv4 address are reserved for the network id and how many bits are reserved for the host id. Since IPv6 does not have the concept of classes, just by looking at an IPv6 address, we do not know how many bits of the address are reserved for the network id and host id. Thus, an additional *prefix length* is required to know this fact.

Furthermore, an IPv6 address can be one of three types, viz., unicast, multicast and anycast. Unlike IPv4, there is no concept of broadcast addresses in IPv6. The IPv6 address types are described in Fig. A.4.

Address type	Description
Unicast	This address identifies a single computer. A datagram heading for an unicast address is routed to the destination computer with the shortest possible path.
Multicast	This address identifies a set of computers, each one of which can be at a different physical location. When a datagram refers to a multicast address, a copy of that datagram is delivered to each of the computers that mapped to a multicast address.
Anycast	This address identifies a set of computers that share a common address prefix (single location). However, the datagram is delivered only to one of the set. This concept is similar to Web server clustering, wherein a set of Web servers is set up to handle HTTP requests from clients. All the Web servers in the set up have the same address prefix. When user requests arrive, each of the available servers can serve the request in turn.

Fig. A.4  IPv6 address types


Since IPv6 addresses can be very long to read/write (remember that they consist of 128 bits), the dotted decimal notation used in case of IPv4 is not used here. Instead, a new syntax called **colon hexadecimal notation** is used to write IPv6 addresses.

Example of such a notation is an address 68DC:7764:FFFF:FFFF:0:1267:8C0A:FFFF.

Further optimizations of the IPv6 addressing are possible. One of them is to drop leading zeroes. Suppose that a portion of an IPv6 address is 00F6:0089. Then, it can be written as F6:89. Another optimization is to replace an address portion containing zero with another colon. Thus, if an address portion is 68CD:0:1267, we can write it as 68CD::1267 (note that the zero portion is replaced by a colon).

Having discussed the various aspects of IPv6, let us summarize the differences between IPv4 and IPv6 headers as shown in Fig. A.5.

IPv4 field	IPv6 field
Header length	Eliminated
Service type	Eliminated (Combination of Priority and Flow label fields serves the same purpose)
Total length	Replaced by payload length
Identification, Flag, Offset	Eliminated
TTL	Replaced by hop limit
Protocol	Replaced by next header
Checksum	Assumed to be provided by higher layer protocols
Options	Replaced by extension headers

Fig. A.5  *Comparison of IPv4 and IPv6 header fields*

Appendix B

Hardware for Error Detection

We shall discuss the hardware devices required for error detection in data transmission. In order to understand how these devices work, we need to know how these devices are constituted. For this, let us discuss the basic electronic devices that make up these devices. After that, we shall study the devices that perform error checking.

ELECTRONIC DEVICES

Logical gates are the basis for all the processing inside a computer. Two gates are very important from the perspective of error detection hardware: the XOR gate and the NOT gate. Another electronic device called the shift register is also crucial for building error detection hardware.

XOR Gate

An eXclusive OR (XOR) gate has two inputs and one output. Each of the inputs and the resulting output can be a 0 or a 1. The simple philosophy of an XOR gate is as follows:

Only if the two inputs are different from each other, the output is 1. Otherwise, it is 0.

Thus, only if one of the inputs is 0 and the other is 1, the output is 1. If both inputs are 0, or both are 1, the output is 0. This is depicted in Fig. B.1.

An XOR gate is indicated by the symbol shown in Fig. B.2.

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

Fig. B.1 XOR gate

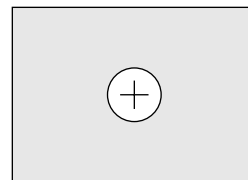


Fig. B.2 Symbol for XOR gate

NOT gate

A NOT gate has only one input, which it reverses to produce the corresponding output. The output of a NOT gate is NOT the same as its input. Thus, if the input to the NOT gate is a 0, the output is a 1, and vice versa. This is shown in Fig. B.3.

A NOT gate is indicated by the symbol shown in Fig. B.4.

Input	Output
0	1
1	0

Fig. B.3 NOT gate

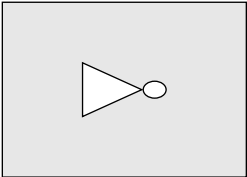


Fig. B.4 Symbol for XOR gate

Shift Register

A shift register is a storage area that holds some bits in a series, one after the other. We can shift the contents of that area to the right or to the left (hence the name *shift register*). This shifting is with respect to time. Thus, at time t_0 , the shift register might contain some bits, which can be left/right shifted at time t_1 by one position, then again at time t_2 by one position, and so on. Example of a right shift register is shown in Fig. B.5.

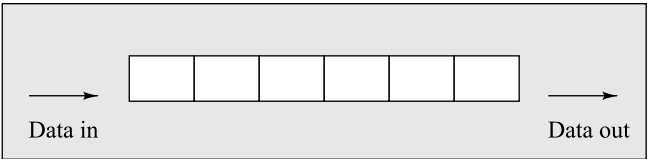


Fig. B.5 Right shift register

Let us see what happens when three bits 111 are arriving as input to a right shift register that contains 101010, as shown in Fig. B.6.

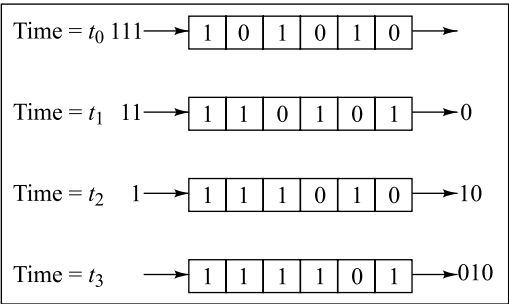


Fig. B.6 Shift register example with three right shifts

VERTICAL REDUNDANCY CHECK (VRC) GENERATOR

We have learnt that the Vertical Redundancy Check (VRC) or parity check is used to detect single-bit transmission errors. VRC checking can be performed by using a device called VRC generator.

It is composed of many XOR gates arranged in a sequence. To calculate the parity of a data unit consisting of n bits, we require $n-1$ XOR gates. Figure B.7 shows the example of an even parity check for an input of 7 bits. The output of each XOR operation is passed to the next XOR gate, and the next input bit is added to it.

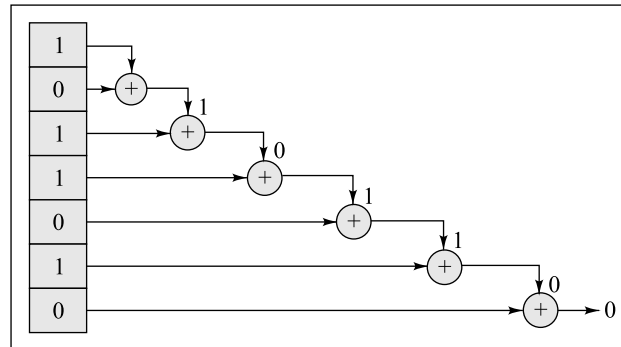


Fig. B.7 Even parity VRC generator

As the figure shows, the final parity bit generated by the VRC generator is 0. This is calculated at the sender's end, and sent along with the data. The receiver calculates the same independently, and compares it with what was received from the sender. If the two match, the receiver assumes that the data has been received without error, and accepts it. If the two do not match, it performs the appropriate error handling procedure.

If we want an odd parity VRC generator, the output of the even parity VRC generator is simply passed through a NOT gate to reverse it.

Longitudinal Redundancy Check (LRC) Generator

The Longitudinal Redundancy Check (LRC) method of error checking is a more stringent error checking mechanism. Figure B.8 shows an example of the LRC generator. Two input streams (shown with dark color) are present here, each containing eight bits. The process of calculating the LRC is to first take the last bits of the two input streams, XOR them, and write the result into the last bit position of the LRC generator. Then we repeat this for the second-last bit, and so on, for all the bits of the two input streams. Additional steps include XORing all the LRC bits in the end with the last bit of the LRC itself (to have a double-level checking, which is not shown).

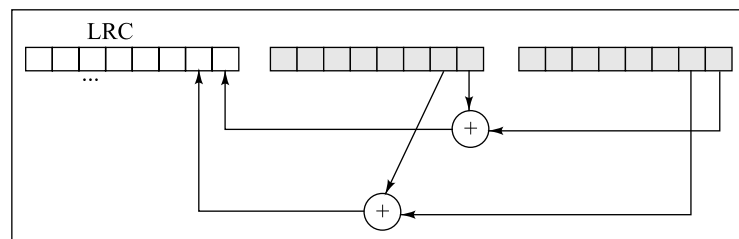


Fig. B.8 LRC check

Appendix C

Network Management and Monitoring

Due to the tremendous increase in the use of networks and the Internet, managing these networks (i.e. ensuring that they are up and running, taking corrective measures, performing maintenance activities) has become extremely crucial. In this regard, the TCP/IP protocol suite has come up with the concept of the **Simple Network Management Protocol (SNMP)**. It resides at the application layer, and therefore, is independent of the underlying protocols and hardware-specific details.

Actually, SNMP is a framework that works with two other components, called **Structure of Management Information (SMI)** and **Management Information Base (MIB)**. This is shown in Fig. C.1.

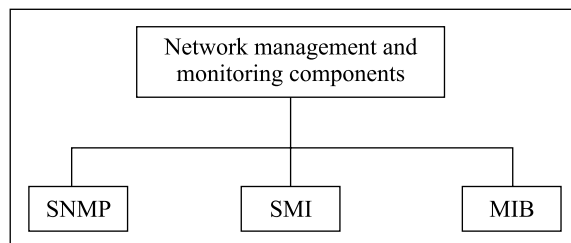


Fig. C.1 *Network management and monitoring components*

SNMP is based on two concepts: managers and agents. A manager is the host that performs the job of network management and monitoring, and manages other hosts or routers, which are called agents. A manager runs a SNMP client, an agent runs a SNMP server. The agent keeps information about itself in a database. The manager can access this information and take an appropriate action. For instance, a router (agent) can store information regarding the number of packets received and the number of packets delivered. Based on their comparison, the management can take an appropriate action to solve the network congestion problems, if any.

Let us discuss the three components, namely, SNMP, SMI and MIB now.

SNMP

As we mentioned, SNMP uses SMI and MIB to perform networking monitoring and management. It is simply an application program that has a client on the manager and a server on the agent. The SNMP allows a manager to perform three important tasks:

1. Retrieve any information from an agent database (i.e. retrieve variables defined by the agent).
2. Store/update the value in an agent database.
3. Receive alerts from agents in case of problems.

To achieve these objectives, SNMP defines five message types, as shown in Fig. C.2. Note that all these messages are exchanged as simple UDP packets between the manager and the agent. Since these messages are simple request-response pairs, they do not require any connection/session to be established. Therefore, TCP is not required in this case.

Message	Description
GetRequest	The manager (client) sends this message to the agent (server) to request for the value of an agent variable.
GetNextRequest	If the agent variables are in the form of a table, the manager cannot use the GetRequest message, as it does not know where the current table pointer is located. This message is used in such cases.
GetResponse	The agent replies to a manager's GetRequest or GetNextRequest messages using this message. This message contains the values of the variables requested for by the manager.
SetRequest	The manager can ask the agent to store a particular value in an agent variable using this message.
Trap	The agent can report an error or any such event to the manager using this message.

Fig. C.2  *SNMP messages*

SMI

As we know, Structure of Management Information (SMI) is one of the two components used by SNMP in network management and monitoring. SMI provides a guideline for SNMP for handling objects (variables, router names, etc.) that are used in SNMP messages. These guidelines are related to three aspects of the objects: naming conventions, data types and encoding methods for facilitating network transmission of these objects.

Naming Conventions

SMI defines each object in SNMP with a unique name. For this, it uses the concept of hierarchical naming (similar to the Domain Name System or DNS), that ensures that the global name of an object is unique, although the local parts of its hierarchy may be duplicated. This unique global name is called *object identifier*. For example, an object identifier can be 1.5.2.4, which can map to iso.com.pki.mgmt.

Data Types

SMI follows and extends the Abstract Syntax Notation 1 (ASN.1) services. ASN.1 is a notation used for representing data that can be sent across a network. It is like a standard data format, which does not depend on a particular vendor. For example, ASN.1 specifies that all integers should be defined as containing 32 bytes, etc. SMI uses the ASN.1 notation, and also uses some more data types that are not a part of the ASN.1 set.


Using ASN.1, SMI defines two categories of data types: simple and structured. Some important simple data types are defined in Fig. C.3.

Data type	Size	Description
Integer	4 bytes	An integer value between 0 and $2^{32} - 1$
String	Variable	Zero or more characters
IP Address	4 bytes	32-bit IP address of a computer

Fig. C.3  *SMI simple data types*

Structured data types are listed in Fig. C.4.

Data type	Description
Sequence	Combination of similar/dissimilar simple data types – similar to structures in the C language, or records in any programming language
Sequence	Combination of same data types of simple or sequence types – similar to the concept of arrays in programming languages

Fig. C.4  *SMI structured data types*

Encoding Methods

SMI uses the Basic Encoding Rules (BER) to encode data that needs to be transmitted over the network. For example, when sending a variable, the BER standard specifies what is its data type (in binary format, corresponding to the data types discussed earlier), length and value. That is, the BER standard specifies that each data item that is to be sent should be encoded to contain three pieces or fields of information: tag, length and value, which are described in Fig. C.5.

BER field	Description
Tag	This one-byte field defines the type of data, which contains three sub-fields, namely, class, format and number. This is simply the numeric representation (encoding) of the data types discussed earlier. For instance, an integer is represented as 00000010, string is represented as 00000100, etc.
Length	This field specifies the length of the item in binary form. So, if this field contains 11, the length of the data item defined by the tag field is 3.
Value	This field contains the actual value of the data.

Fig. C.5  *BER specifications*

MIB

The Management Information Base (MIB) is used by SNMP for network monitoring and management, along with SNI. MIB is the collection of objects for an agent. Each agent has its MIB. The MIB objects are classified into 8 categories: system, interface, address translation, IP, ICMP, TCP, UDP and EGP.

How to Access MIB Objects

The simple concept of hierarchical addressing is used to access MIB variables. This is also very similar to the Domain Name System (DNS) mechanism. For instance, suppose we have *Main* as the root, which has two branches, *Sub1* and *Sub2*. Furthermore, *Sub1* has two more branches, *Sub1Sub1* and *Sub1Sub2*. This is shown in Fig. C.6.

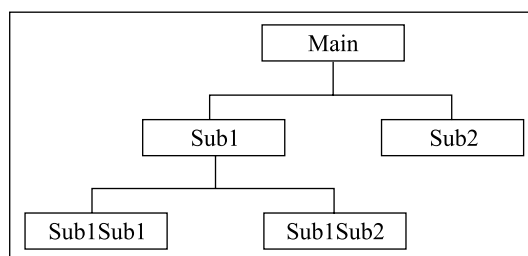


Fig. C.6 Concept of MIB object hierarchy

Using the structure shown in the figure, we can easily determine that the complete hierarchical address of *Sub1Sub2* is *Main.Sub1.Sub1Sub2*, which, in numeric form is 1.1.2, assuming we start numbering addresses from 1 left-to-right. Any MIB object is also accessed in a similar hierarchical fashion, using its full address, starting with the root object.

SNMP PORTS

As we mentioned, SNMP uses UDP packets for communication. An SNMP server (agent) uses the well-known (i.e. pre-determined) port number 161. The SNMP client (manager) uses the well-known port number 162. The agent issues a passive open on port number 161, and waits for connection requests from the SNMP clients. The manager issues an active open on port number 162.

Interestingly, SNMP uses well-known port numbers for both the client and the server. In almost all other client-server protocols, usually only the server's port number is well-known. Why is SNMP different? Recall that an SNMP server (agent) can send a *Trap* message to the client (manager). In order for the server to be able to send this message, it must know the client's port number in advance. This is the reason why SNMP uses well-known client and server port numbers.

HDLC

The **High Level Data Link Control (HDLC)** protocol specifies a standard for sending packets over serial transmission lines. It is a data link layer protocol. Several modes of transmission are supported in HDLC. This can include guaranteed delivery mode using the sliding window technique. But since these days, reliability is established in the transport layer by the TCP protocol, HDLC is run in the unreliable mode. Guarantees are hence taken care of in the higher layers.

Figure C.7 shows the frame structure in HDLC.

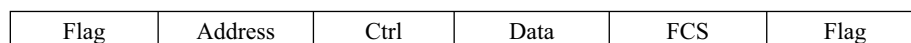


Fig. C.7 HDLC frame structure

The start and end of every HDLC frame is designated with the help of an 8-bit flag, which contains a fixed value, which is 01111110 in binary. This flag is used to detect the start and end of a frame. Naturally, this bit pattern cannot appear as a part of data or any other field in the HDLC frame. If it does appear there, certain transparently done modifications are used to handle these situations.

HDLC supports many types of frames. For example, it can be used to send a command or to receive a response. If it is a command frame, the address field contains the address of the host to which the command is being sent. If it is a response frame, the address is the address of the host that has sent this frame.

The control field contains 8 or more bits. It identifies the current frame either as a command or as a response. The variable-length data field contains the actual data to be sent, usually sent as a multiple of 8 bits.

The Frame Check Sequence (FCS) field is a 16-bit or 32-bit field, which contains a CRC for ensuring data integrity. The CRC is computed over the address, control, and information fields. The receiver can make use of the FCS field for detecting errors in transmission.

HDLC was created by ISO. It provides both connection-oriented and connectionless services. HDLC can work in the half-duplex or full-duplex transmission modes. It can provide point-to-point or multi-point services. Transmission can be based on switched or non-switched channels.

In HDLC, a station can perform the role of a primary station or a secondary station. A primary station acts as a master, and transmits command frames. It receives response frames from secondary stations. For this to be possible, a master station maintains a separate dedicated link with every secondary station by using a multipoint line. A secondary station acts as a slave to a primary station. It receives command frames and acts on those commands. It then sends response frames to the primary station. Two secondary stations cannot communicate with each other.

Hence, the addressing in HDLC is simpler. Command frames contain the address of the selected secondary station, which is supposed to receive and act on the command. Response frames contain the address of the responding secondary station.

SDLC

Synchronous Data Link Control (SDLC) is a data link layer protocol, which used to be a part of IBM's proprietary SNA network. SDLC can work with single or multipoint links. It has capabilities for error detection. SDLC was mainly used by IBM on its mainframe range of computers. Subsequently, it got adopted for various other systems as well. However, SNA itself is becoming obsolete, with the increasing use of TCP/IP. As such, SDLC is also rarely used now. The original development of SDLC can be dated to the year 1975. This was subsequently enhanced and adopted as a standard by ISO in the form of HDLC.

SDLC provides both connection-oriented and connectionless services. HDLC can work in the half-duplex or full-duplex transmission modes. It can provide point-to-point or multipoint services. Transmission can be based on switched or non-switched channels.

SDLC works with two types of network stations, viz., primary and secondary. Primary stations control the working of the other stations, which are called secondary stations. A primary station polls the secondary stations in a predetermined sequence. The secondary stations are then allowed to

transmit if they have any data to be sent. The primary station also creates and terminates connections. It is also responsible for managing the operation of the connection/link while it is operational. Secondary stations are controlled by a primary station. Thus, secondary stations can send data to a primary station only if the primary station allows for it.

Four types of configurations are possible in SDLC with reference to primary and secondary stations, as mentioned below.

1. Point to point – In this mode, there is only one primary station and one secondary station.
2. Multipoint – In this mode, there is one primary station and there are multiple secondary stations.
3. Loop – In this mode, we have a loop topology. Here, the primary station is connected to the first and the last secondary stations. Intermediate secondary stations pass messages through one another while responding to the requests of the primary station.
4. Hub go-ahead – In this mode, we have an inbound and an outbound channel. The primary station uses the outbound channel to communicate with the secondary stations. The secondary stations use the inbound channel to communicate with the primary station. The inbound channel is 'daisy-chained' back to the primary station through each secondary station.

The SDLC frame format is shown in Fig. C.8.

Flag	Address	Ctrl	Data	FCS	Flag
------	---------	------	------	-----	------

Fig. C.8  *SDLC frame structure*

The 8-bit flag field is used to indicate the start and end of a frame. It helps in initiating and closing error checking of data. The 8-bit or 16-bit address field specifies the address of the secondary station. This field indicates whether the frame originated from a primary station or from a secondary station. The 8-bit or 16-bit control field specifies whether the frame is of information type, supervisory type, or is an unnumbered frame. The variable-length data field contains the actual data being sent. The 16-bit Frame Check Sequence (FCS) field is used for error detection.

BISYNC

Binary Synchronous Communication (BSC or Bisync) is a very old protocol. This protocol was also devised in the days of IBM mainframe way back in 1967. In contemporary times, it has very little relevance. Bisync is an IBM link protocol. It was originally designed to replace the Synchronous-Transmit-Receive (STR) protocol used by the earlier second generation computers. The idea behind Bisync was to be able to use a common set of link management rules for three different alphabets for encoding messages.

The reason why Bisync protocol was created was to provide support for batch communication between a System/360 mainframe and another mainframe or a Remote Job Entry (RJE) terminal such as the IBM 2780 and IBM 3780. However, later some non-IBM hardware vendors also used Bisync for other purposes such as tape-to-tape transmission.

Appendix D

Answers to True or False



CHAPTER 1

- | | | | | |
|----------|----------|---------|----------|-----------|
| 1. False | 2. True | 3. True | 4. True | 5. False |
| 6. True | 7. False | 8. True | 9. False | 10. False |

CHAPTER 2

- | | | | | |
|----------|----------|----------|----------|----------|
| 1. False | 2. True | 3. False | 4. False | 5. False |
| 6. False | 7. False | 8. True | | |

CHAPTER 3

- | | | | | |
|---------|----------|----------|----------|---------|
| 1. True | 2. False | 3. False | 4. False | 5. True |
| 6. True | 7. True | 8. False | | |

CHAPTER 4

- | | | | | |
|-----------|----------|----------|----------|----------|
| 1. True | 2. False | 3. False | 4. False | 5. False |
| 6. False | 7. True | 8. True | 9. True | 10. True |
| 11. False | | | | |

CHAPTER 5

- | | | | | |
|-----------|-----------|----------|----------|-----------|
| 1. True | 2. True | 3. True | 4. False | 5. False |
| 6. False | 7. False | 8. True | 9. False | 10. False |
| 11. False | 12. False | 13. True | 14. True | |

CHAPTER 6

- | | | | | |
|----------|----------|----------|----------|---------|
| 1. False | 2. True | 3. False | 4. False | 5. True |
| 6. False | 7. False | 8. False | | |

CHAPTER 7

- | | | | | |
|-----------|----------|-----------|----------|----------|
| 1. True | 2. True | 3. True | 4. False | 5. False |
| 6. True | 7. False | 8. False | 9. False | 10. True |
| 11. False | 12. True | 13. True | 14. True | 15. True |
| 16. True | 17. True | 18. False | | |

CHAPTER 8

- | | | | | |
|----------|----------|----------|----------|----------|
| 1. False | 2. False | 3. True | 4. True | 5. False |
| 6. True | 7. False | 8. False | 9. False | 10. True |
| 11. True | 12. True | | | |

CHAPTER 9

- | | | | | |
|-----------|-----------|-----------|----------|-----------|
| 1. True | 2. False | 3. True | 4. False | 5. False |
| 6. False | 7. True | 8. True | 9. True | 10. True |
| 11. False | 12. True | 13. False | 14. True | 15. True |
| 16. True | 17. False | 18. True | 19. True | 20. False |
| 21. True | 22. True | 23. False | | |

CHAPTER 10

- | | | | | |
|----------|-----------|-----------|----------|----------|
| 1. False | 2. True | 3. False | 4. True | 5. False |
| 6. False | 7. True | 8. True | 9. True | 10. True |
| 11. True | 12. True | 13. False | 14. True | 15. True |
| 16. True | 17. False | | | |

CHAPTER 11

- | | | | | |
|---------|----------|---------|----------|----------|
| 1. True | 2. False | 3. True | 4. True | 5. False |
| 6. True | 7. False | 8. True | 9. False | 10. True |

CHAPTER 12

- | | | | | |
|----------|----------|----------|---------|----------|
| 1. False | 2. True | 3. True | 4. True | 5. False |
| 6. False | 7. False | 8. True | 9. True | 10. True |
| 11. True | 12. True | 13. True | | |

CHAPTER 13

- | | | | | |
|----------|---------|----------|----------|----------|
| 1. False | 2. True | 3. True | 4. False | 5. True |
| 6. False | 7. True | 8. False | 9. True | 10. True |

- | | | | | |
|-----------|-----------|-----------|----------|----------|
| 11. True | 12. False | 13. False | 14. True | 15. True |
| 16. False | 17. True | 18. True | | |

CHAPTER 14

- | | | | | |
|----------|-----------|----------|----------|-----------|
| 1. False | 2. True | 3. False | 4. False | 5. True |
| 6. False | 7. False | 8. True | 9. True | 10. False |
| 11. True | 12. False | | | |

CHAPTER 15

- | | | | | |
|-----------|-----------|----------|-----------|-----------|
| 1. False | 2. False | 3. False | 4. True | 5. False |
| 6. True | 7. True | 8. True | 9. False | 10. True |
| 11. True | 12. False | 13. True | 14. False | 15. True |
| 16. False | 17. False | 18. True | 19. True | 20. False |
| 21. True | 22. True | 23. True | 24. False | 25. True |

CHAPTER 16

- | | | | | |
|----------|---------|----------|---------|----------|
| 1. False | 2. True | 3. True | 4. True | 5. False |
| 6. True | 7. True | 8. False | 9. True | 10. True |

CHAPTER 17

- | | | | | |
|-----------|-----------|-----------|-----------|-----------|
| 1. False | 2. False | 3. True | 4. True | 5. False |
| 6. True | 7. False | 8. True | 9. True | 10. True |
| 11. True | 12. True | 13. False | 14. False | 15. True |
| 16. True | 17. False | 18. False | 19. True | 20. True |
| 21. True | 22. True | 23. False | 24. True | 25. False |
| 26. False | 27. True | 28. True | 29. False | 30. False |
| 31. True | 32. True | 33. True | 34. False | 35. True |

CHAPTER 18

- | | | | | |
|-----------|-----------|----------|-----------|----------|
| 1. True | 2. False | 3. True | 4. True | 5. False |
| 6. True | 7. True | 8. False | 9. True | 10. True |
| 11. True | 12. True | 13. True | 14. False | 15. True |
| 16. False | 17. False | | | |

CHAPTER 19

- | | | | | |
|---------|----------|----------|----------|-----------|
| 1. True | 2. False | 3. False | 4. True | 5. False |
| 6. True | 7. False | 8. False | 9. False | 10. False |

- | | | | | |
|-----------|-----------|-----------|----------|-----------|
| 11. False | 12. False | 13. False | 14. True | 15. False |
|-----------|-----------|-----------|----------|-----------|

CHAPTER 20

- | | | | | |
|-----------|----------|----------|-----------|-----------|
| 1. False | 2. True | 3. False | 4. False | 5. True |
| 6. False | 7. True | 8. True | 9. True | 10. False |
| 11. False | 12. True | 13. True | 14. False | 15. True |
| 16. False | 17. True | | | |

CHAPTER 21

- | | | | | |
|----------|----------|---------|----------|----------|
| 1. False | 2. True | 3. True | 4. False | 5. False |
| 6. True | 7. False | | | |

Appendix E

Answers to Multiple-Choice Questions

CHAPTER 1

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. c | 2. b | 3. c | 4. b | 5. d |
| 6. a | 7. a | 8. b | 9. d | 10. a |
| 11. a | 12. c | 13. b | 14. a | 15. d |
| 16. d | 17. c | 18. a | 19. d | 20. c |

CHAPTER 2

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. a | 2. a | 3. b | 4. c | 5. b |
| 6. a | 7. c | 8. a | 9. c | 10. c |
| 11. a | 12. d | 13. a | 14. d | 15. b |

CHAPTER 3

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. b | 2. d | 3. c | 4. b | 5. b |
| 6. d | 7. b | 8. a | 9. a | 10. b |
| 11. d | 12. d | 13. c | 14. b | 15. b |

CHAPTER 4

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. b | 2. a | 3. c | 4. c | 5. b |
| 6. b | 7. b | 8. a | 9. b | 10. c |
| 11. a | 12. c | 13. a | 14. b | 15. a |
| 16. c | | | | |

CHAPTER 5

- | | | | | |
|------|------|------|------|-------|
| 1. a | 2. d | 3. a | 4. c | 5. b |
| 6. c | 7. d | 8. a | 9. b | 10. d |

- | | | | | |
|-------|-------|-------|-------|-------|
| 11. a | 12. a | 13. d | 14. b | 15. c |
| 16. a | 17. b | 18. d | | |

CHAPTER 6

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. c | 2. b | 3. a | 4. b | 5. c |
| 6. a | 7. a | 8. a | 9. d | 10. a |
| 11. b | 12. a | 13. d | 14. b | |

CHAPTER 7

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. c | 2. b | 3. a | 4. a | 5. d |
| 6. a | 7. a | 8. b | 9. b | 10. d |
| 11. c | 12. d | 13. b | 14. d | 15. d |
| 16. a | 17. b | | | |

CHAPTER 8

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. b | 2. c | 3. d | 4. a | 5. d |
| 6. c | 7. c | 8. b | 9. b | 10. d |
| 11. b | 12. d | 13. d | 14. a | |

CHAPTER 9

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. d | 2. a | 3. c | 4. d | 5. a |
| 6. d | 7. c | 8. c | 9. b | 10. a |
| 11. a | 12. c | 13. c | 14. b | 15. d |

CHAPTER 10

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. c | 2. a | 3. b | 4. d | 5. a |
| 6. c | 7. a | 8. a | 9. a | 10. a |
| 11. c | 12. d | 13. b | 14. c | 15. c |
| 16. a | | | | |

CHAPTER 11

- | | | | | |
|------|------|------|------|-------|
| 1. c | 2. a | 3. d | 4. a | 5. c |
| 6. a | 7. c | 8. b | 9. b | 10. a |

CHAPTER 12

- | | | | | |
|------|------|------|------|------|
| 1. b | 2. d | 3. a | 4. c | 5. b |
|------|------|------|------|------|

- | | | | | |
|-------|-------|-------|------|-------|
| 6. b | 7. d | 8. c | 9. a | 10. b |
| 11. b | 12. d | 13. a | | |

CHAPTER 13

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. c | 2. b | 3. a | 4. a | 5. c |
| 6. c | 7. b | 8. a | 9. c | 10. c |
| 11. a | 12. b | 13. d | 14. a | 15. c |
| 16. c | 17. c | 18. | | |

CHAPTER 14

- | | | | | |
|------|------|------|------|-------|
| 1. d | 2. a | 3. b | 4. d | 5. c |
| 6. a | 7. c | 8. c | 9. d | 10. c |

CHAPTER 15

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. b | 2. c | 3. a | 4. a | 5. c |
| 6. a | 7. b | 8. c | 9. d | 10. d |
| 11. a | 12. c | 13. a | 14. d | |

CHAPTER 16

- | | | | | |
|------|------|------|------|-------|
| 1. c | 2. a | 3. c | 4. a | 5. a |
| 6. b | 7. c | 8. d | 9. c | 10. a |

CHAPTER 17

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. c | 2. d | 3. a | 4. b | 5. c |
| 6. a | 7. c | 8. b | 9. d | 10. c |
| 11. d | 12. c | 13. b | 14. d | 15. a |
| 16. c | 17. d | 18. a | | |

CHAPTER 18

- | | | | | |
|-------|-------|-------|------|-------|
| 1. d | 2. b | 3. a | 4. c | 5. d |
| 6. a | 7. b | 8. c | 9. d | 10. a |
| 11. b | 12. c | 13. a | | |

CHAPTER 19

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. a | 2. c | 3. b | 4. c | 5. d |
| 6. b | 7. b | 8. a | 9. a | 10. d |
| 11. b | 12. d | 13. b | 14. a | 15. b |

CHAPTER 20

- | | | | | |
|-------|-------|-------|-------|-------|
| 1. c | 2. a | 3. d | 4. b | 5. b |
| 6. b | 7. d | 8. c | 9. b | 10. a |
| 11. b | 12. c | 13. a | 14. b | 15. a |
| 16. c | 17. d | | | |

CHAPTER 21

- | | | | | |
|------|------|------|------|------|
| 1. d | 2. a | 3. c | 4. b | 5. c |
| 6. b | | | | |

Appendix F

Fundamental Mathematics

for Problem Solving

NUMBER SYSTEMS

A number system contains a set of numbers that have common characteristics. Let us discuss the common number systems that are used by humans (decimal) as well as computers (binary, octal, hexadecimal). In any number system, the following three aspects determine the value of each digit within a number:

1. The digit itself.
2. The position of the digit in that number.
3. The base of the number system.

The base of a number system is the number of different symbols used in that system. For example, in decimal number system the base is 10, because it uses 10 different symbols from 0 to 9.

Decimal Number System

As we know, the decimal number system contains 10 different symbols, 0 to 9. Therefore, its base is 10. Thus, a number in the decimal number system is actually represented by multiplying each digit by its weight, and then by adding all the products. For example, the value of a number 4510 in decimal number system is calculated as shown in Fig. F.1.

Thousands position		Hundreds position		Tens position		Ones position		Total
4×10^3	+	5×10^2	+	1×10^1	+	0×1^0		—
4×1000	+	5×100	+	1×10	+	0×1		—
4000	+	500	+	10	+	0	=	4510

Fig. F.1 Representation of a decimal number

Binary Number System

The binary number system contains only two distinct symbols (called binary digits, or bits), 0 and 1. Therefore, its base is 2. We had used the increasing powers of 10 from right to left to represent decimal numbers. Here, we use increasing powers of 2. Thus, the value of a binary number 1001 in decimal is calculated as equal to 9 as shown in Fig. F.2.

4th position		3rd position		2nd position		1st position	Total
1×2^3	+	0×2^2	+	0×2^1	+	1×2^0	—
1×8	+	0×4	+	0×2	+	1×1	—
8	+	0	+	0	+	1	= 9

Fig. F.2 Binary number representation

How can we convert a decimal number to its equivalent binary number? For that, we divide the number continuously by 2 till we get a quotient of 0. In every stage, we get 0 or 1 as the remainder. When we write these remainders in the reverse order, we get the equivalent binary number. This is as shown in Fig. F.3, for a decimal number 500.

Divisor	Quotient	Remainder
2	500	
2	250	0
2	125	0
2	62	1
2	31	0
2	15	1
2	7	1
2	3	1
2	1	1
	0	1

Fig. F.3 Decimal to binary conversion

The number 500 is divided continuously by 2, each time noting down the remainder (0 or 1). Finally, when the quotient is 0, we stop the process, and write down the remainders in the reverse order. As we can see, the binary equivalent of a decimal number 500 is 111110100.

Octal Number System


We know that the decimal number system has a base of 10, and that the binary number system has a base of 2. Similarly, the octal number system has a base of 8 as it represents 8 distinct symbols (0 to 7). The same principles, as discussed earlier are used for converting an octal number to decimal or vice versa. The only change is, now 8 is used for multiplying weights or dividing quotients successively. So, a number 432 in octal can be converted into its decimal equivalent as shown in Fig. F.4.

3rd position		2nd position		1st position	Total
4×8^2	+	3×8^1	+	2×8^0	—
4×64	+	3×8	+	2×1	—
256	+	24	+	2	= 282

Fig. F.4 Octal to decimal conversion

Let us work backwards and convert the decimal number 282 thus obtained to see if we get back the original octal number 432, as shown in Fig. F.5. Note that we go on dividing the decimal number continuously by 8 now, until we obtain a quotient of 0. In each case, we note the remainder, and in the end, write all the remainders in the reverse order to get the equivalent octal number.

Divisor	Quotient	Remainder
8	282	
8	35	2
8	4	3
	0	4

Fig. F.5  *Decimal to binary conversion*


As the figure shows, the decimal number 282 is the same as octal number 432.

Hexadecimal Number System

Hexadecimal number system is actually a superset of the decimal number system. We know that the decimal number system has 10 distinct symbols, from 0 to 9. Hexadecimal number system has all these 10 symbols, and has 6 more (A through F). Thus, the hexadecimal number system contains 16 different symbols from 0 to 9 and then A through F. The symbol A in hexadecimal is equivalent to the decimal number 10, the symbol B is equivalent to the decimal number 11, and so on, which means that the last symbol F in hexadecimal number system is equivalent of the decimal number 15.


We follow the same conversion logic as we used for binary and octal number systems. The base is now 16. Let us convert a hexadecimal number 683C into its decimal equivalent, as shown in Fig. F.6.

4th position		3rd position		2nd position		1st position	Total
6×16^3	+	8×16^2	+	3×16^1	+	$C \times 16^0$	—
6×4096	+	8×256	+	3×16	+	12×1	—
24576	+	2048	+	48	+	12	= 26684

Fig. D.6  *Hexadecimal to decimal conversion*

As before, let us convert this decimal number 26684 to see if we get back the hexadecimal number 683C using a decimal-to-hexadecimal conversion method, as shown in Fig. F.7.

Divisor	Quotient	Remainder
16	26684	
16	1667	C
16	104	3
16	6	8
	0	6

Fig. F.7  *Decimal to hexadecimal conversion*

We can see that we indeed obtain 683C in hexadecimal as equivalent of the decimal number 26684.

Binary Number Representation

From the context of computers and data communications, the binary number system is the most important, because it is used inside the computers to represent any alphabet, number or symbol. Therefore, let us discuss some aspects related to binary numbers.

Unsigned Binary Number

All binary numbers are unsigned by default, which means that they are positive. This is true in case of decimal numbers as well. When we write a decimal number 457, we mean that it is implicitly (+)457. Let us now try to figure out the maximum number representation capabilities of the binary number system. As before, let us take the simple example of decimal number system.


Suppose we have a single slot, where we can store one decimal digit. How many different values it can store? Of course, it can store a digit between 0 and 9, thus 10 different values. Instead, if we decide to store a binary digit there, how many different values can it take? Of course, it can take a value of 0 or 1, thus 2 different values, where 1 is the maximum.

Now let us assume that we have two slots. If we decide to write decimal digits in each of them, then we can store a minimum number of 00 and a maximum number of 99 in decimal in these slots. Similarly, in binary, we can write 00 as the minimum number and 11 as the maximum number. Since binary 11 is 3 in decimal, the maximum number that we can in binary is actually decimal 3.

If we note, a pattern is emerging. Each new slot can accommodate the maximum digit in a given number system. Thus, if we have three slots, we can have the maximum decimal number as 999, and the maximum binary number as 111 (which is 7 in decimal).

Let us tabulate these results only for binary numbers, as shown in Fig. F.8. We shall continue the list, as we are observing some commonalities as shown in the last column.

Available slots (bit positions)	Maximum number in binary	Maximum number in decimal	Another way to represent the same number
1	1	1	$2^1 - 1$
2	11	3	$2^2 - 1$
3	111	7	$2^3 - 1$
4	1111	15	$2^4 - 1$
5	11111	31	$2^5 - 1$
6	111111	63	$2^6 - 1$
7	1111111	127	$2^7 - 1$
8	11111111	255	$2^8 - 1$

Fig. F.8  Maximum binary numbers per bit positions allowed

The last column indicates how we can find out the maximum number possible in any storage allocated for binary numbers. It is simply 2 raised to the number of bit positions allocated minus 1. Thus, when we have 8 slots, or 8 bit positions, we can represent decimal numbers from 0 (minimum number) to 255 (maximum number), a total of 256 different numbers. Since a computer

byte generally contains 8 bits, a byte can represent one of the 256 distinct values (i.e. a number between 0 and 255).

This is how unsigned binary numbers are represented.

Signed Binary Numbers

When we want to represent signed binary numbers, we need an extra slot. This is similar to decimal number system. Suppose we want to store a decimal number -89 . Then we need three slots – two would not be enough. The only difference is that in case of the binary number system, even the sign is represented by a 0 (indicates the $+$ sign) and 1 (indicates the $-$ sign). Also, the leftmost bit represents the sign.

Thus, if we have a number 110, and we are told that it is a signed number, we must interpret the leftmost bit (i.e. 1) as its sign, which is negative. Thus, only the two remaining bits (i.e. 10) should be considered as its value, which is 2 in decimal. When we append the negative sign, 110 in binary becomes -2 in decimal. This may sound very confusing. How do we know if we should treat 110 a signed or unsigned? If unsigned, the leftmost 1 is data, otherwise it is sign. But how do we know this fact? Well, it depends on the context. Unless we know whether we are dealing with unsigned binary numbers or signed binary numbers, we would simply not know this!

Binary Arithmetic

For simplicity, we shall ignore signed numbers. Let us consider the four basic binary arithmetic operations: addition, subtraction, multiplication and division. Of these, multiplication and division are straightforward, and we shall not discuss them. Instead, we shall just discuss binary addition and subtraction, as there are slight complications in these.

Binary Addition

The binary addition rules are quite simple as shown in Fig. F.9. The last case is notable, when we add a 1 to a 1, we get 0 as the sum and 1 as the carry in the next bit position.

Let us add two binary numbers 111 and 110 as shown in Fig. F.10.

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 0$ with carry
1

Fig. F.9 Binary addition rules

Carry	1	1		
		1	1	1
+		1	1	0
Sum	1	1	0	1

Fig. F.10 Example of binary addition

Binary Subtraction

The rules for binary subtraction are shown in Fig. F.11. Here also, the last case is significant. When 1 is to be subtracted from 0, the result is 1 with 1 borrowed from the previous position.

Let us subtract a binary number 1011 from another binary number 10111 as shown in Fig. F.12.

0 – 0 = 0
0 – 1 = 1
1 – 0 = 1
1 – 1 = 0 with borrow
1

Fig. F.11 Binary subtraction rules

Borrow	1				
	1	0	1	1	1
–		1	0	1	1
Result	0	1	1	0	0

Fig. F.12 Example of binary subtraction

Fourier Series

We know that an analog signal has three important properties, using which it can be described. These properties are: amplitude, frequency and phase.

A famous mathematician, Jean Baptiste Fourier developed a theory, which says that any periodic signal can be expressed as a sum of infinite sine functions that have different amplitudes, frequencies and phase shifts. The sum is called as **Fourier series**. Mathematically, if $s(t)$ is a periodic signal with period P , it can be represented as:

$$s(t) = a_0/2 + \sum [a_i \times \cos(2\pi it/P) + b_i \times \sin(2\pi it/P)]$$

where $i = 1$ to infinity

The coefficients a_i ($i = 0, 1, 2, \dots$) and b_i ($i = 1, 2, 3, \dots$) can be found as follows:

$$a_i = \frac{2}{P} \int_{-P/2}^{P/2} s(t) \times \cos(2\pi it/P) dt \quad (i = 0, 1, 2, \dots) \text{ and}$$

$$b_i = \frac{2}{P} \int_{-P/2}^{P/2} s(t) \times \sin(2\pi it/P) dt \quad (i = 0, 1, 2, \dots) \text{ and}$$

The point is, any analog signal can be considered to be made up of a series of analog waves, and their sum constitutes the analog signal.

Let us have an example. Suppose that $s(t)$ is defined as follow:

$$s(t) = \begin{cases} 1 & \text{for } 0 \leq t < \pi, 2\pi \leq t < 3\pi, 4\pi \leq t < 5\pi, \text{ etc.} \\ -1 & \text{for } \pi \leq t < 2\pi, 3\pi \leq t < 4\pi, 5\pi \leq t < 6\pi, \text{ etc.} \end{cases}$$

We can write this as a Fourier series. In this case, all the constants a_i for $0 \leq i$ are 0. Constants b_i can be defined as:

$$b_i = \begin{cases} 0 & \text{if } i \text{ is even} \\ 4/\pi i & \text{if } i \text{ is odd} \end{cases}$$

Finally, the periodic function can be written as:

$$s(t) = \sum 4/\pi i \sin(it) \text{ for all odd values of } i \text{ from } 1 \text{ to infinity}$$

What is the use of this analysis in real life? This is useful in sending complex analog signals over a medium that has limited bandwidth. Because, it is the same concept as approximating the function using some of the Fourier series terms. The more the terms we are able to transmit, the higher will be the accuracy of the signal being transmitted, and therefore, the better is the approximation of the original signal.

Fourier series can also be used in constructing transmission hardware. For instance, a device called as filter is used to intentionally discard, or filter out, certain frequencies. The Fourier analysis can tell which frequencies should ideally be filtered out, without impacting the representation of the original signal.

Glossary

- Acknowledgement - Response sent by a receiver to indicate the successful receipt of data.
- Active hub - A hub that repeats (regenerates) data, like a repeater.
- Active open - Making an open request for a connection.
- Active Server Pages (ASP) - A technology promoted by Microsoft for creating dynamic Web pages.
- Active Web pages - A Web page that involves some client-side processing code using Java applets or ActiveX controls.
- ActiveX control - Microsoft's technology for Active Web pages.
- Adaptive Bridge - A bridge that keeps learning and updating itself with changes to network conditions or paths.
- Adaptive Differential Pulse Code Modulation (ADPCM) - A modified version of Pulse Code Modulation.
- Address binding - Process of mapping logical address to physical address.
- Address Resolution Protocol (ARP) - Protocol in TCP/IP that accepts IP address and returns physical address of a computer or router.
- Addressing - The mechanism of assigning address to a computer.
- Advanced Mobile Phone System (AMPS) - A connection mechanism in cellular telephony.
- Advanced Research Project Agency (ARPA) - The US government agency that funded ARPANET.
- American National Standards Institute (ANSI) - A body that governs many standards related to computing and networking.
- American Standard Code for Information Interchange (ASCII) - Character encoding scheme devised by ANSI and used extensively in computers and networking.
- Amplifier - A device that accepts an analog signal and boosts it to produce the output.
- Amplitude - The strength of a signal, measured in volts or amperes.
- Amplitude Modulation (AM) - An analog-to-analog encoding mechanism of modifying the carrier wave with the amplitude of the incoming signal.
- Amplitude Shift Keying (ASK) - A digital-to-analog encoding mechanism of modifying the amplitude of the carrier wave to represent the digital bit values.
- Analog - Something that varies continuously, and is another way of representing the same thing.

Analog-to-Digital Converter (A-D) - A device that accepts analog pulses, and gives out digital bit information.

Animation - Technology that creates a feeling of motion by rapidly showing one picture after the other.

Application Adaptation Layer (AAL) - The first layer in the ATM protocol, which facilitates ATM to work with other protocols.

Application layer - The topmost (seventh) layer in the OSI model that provides access to applications over a network.

ARPAnet - The packet switching network funded by ARPA, which paved way for the Internet.

Asymmetric DSL (ADSL) - A faster data access technology that works over the existing copper wires but provides much higher data rates.

Asynchronous Transfer Mode (ATM) - A modern fast packet-switching protocol that works with text, voice, audio, video and can be used in LAN, MAN, WAN.

Asynchronous transmission - A mechanism of data transmission where the sender and the receiver do not require to synchronize before transmission begins, the sender can transmit at any time.

ATM Layer - The second layer in the ATM protocol.

Attenuation - The loss of a signal's strength as it travels across a transmission medium.

Authentication - Establishing a proof of identity before communication can begin.

Bandwidth - The information carrying capacity of a signal or a medium, calculated using the difference between the highest and the lowest frequencies.

Bandwidth-on-demand - An agreement that allows a user to make use of additional bandwidth as and when required.

Basic Rate Interface (BRI) - The ISDN interface for home users that provides a data rate of 192 Kbps using 2B+D channels.

Baud rate - The number of signal elements transmitted in one second.

Best effort delivery - A delivery mechanism that attempts to, but gives no guarantee about, the successful delivery of packets.

Binary Coded Decimal (BCD) - An encoding mechanism that uses four bits to encode each digit.

Binary exponential backoff policy - Used in Ethernet, this algorithm minimizes the chances of two stations transmitting the data at the same time after a collision.

Bit - Binary digit, can be a 0 or a 1 value.

Bit rate - The transmission rate in case of digital signals, usually represented in Bits Per Second (BPS).

Bits Per Second (BPS) - See Bit rate.

Bridge - A network device that operates at the two lowest OSI layers, and has the intelligence of forwarding and filtering packets between two network segments.

Broadband ISDN (B-ISDN) - ISDN that offers high data rates, suitable for video transmissions.

Browser-based emails - A technology that allows end users to access emails using the Web browser interface.

Browsing - Accessing Web pages with the help of a Web browser.

Bursty data - Data transmission whose rate is unpredictable.

Bus topology - A network arrangement wherein all the computers attach to a shared medium.

Byte - Usually, but not always, combination of eight bits.

Cable modem - A device that allows the transmission of Internet data over cable television networks.

Carrier Sense Multiple Access with Collision Detect (CSMA/CD) - The protocol used by Ethernet networks to sense if two or more stations have transmitted at the same time, and if so, how to recover from the resulting collision.

Carrierless Amplitude Phase (CAP) - One of the two ways of sending data in DSL technology.

Cell - A small unit of data (48 bytes in ATM) that is transmitted across the network.

Cell Relay - Another name for ATM.

Central Office (CO) - The regional telephone office from where people receive their telephone signals.

Circuit switching - The switching technology of establishing a dedicated connection between the sender and the receiver before transmission begins.

Client computer - The computer which initiates an interaction with a server.

Client pull - The technology employed in active Web pages, where the client automatically keeps requesting for information to the server.

Coaxial cable - A transmission medium that contains conducting core, insulating material and a second conducting sheath.

Code Division Multiple Access (CDMA) - A technology used in wireless transmission, based on the frequencies of the signals.

CODEC - Coder + Decoder - It is used in Pulse Code Modulation.

Collaborative authoring - Mechanism of allowing multiple users sitting at different locations to simultaneously access and change the contents of a document in real time.

Collision - The result of two or more computers transmitting data at the same time.

Common Gateway Interface (CGI) - The earliest technology of dynamic Web pages, which is not so popular these days.

Compression - Reduction of message size without impacting its contents to reduce transmission demands.

Confidentiality - Allowing only the sender and the receiver an access to the contents of transmission.

Congestion - Slow state of the network caused by too many senders, receivers or transmissions.

Congestion control - Mechanism of reducing congestion to improve network throughput.

Consultative Committee for International telegraphy and Telephony (CCITT) - An international standards group, now known as ITU-T.

Convergence Sub-layer (CS) - An ATM sub-layer, which is the upper portion of the AAL.

Crawler - Software on the Internet that automatically updates the content base of a search engine.

Crosstalk - Problem of noise caused by transmissions on another line.

Cryptography - The art of encoding and decoding messages so as to protect the contents of the messages from the eyes of the attackers.

Cyclic Redundancy Check (CRC) - Highly reliable error-detection mechanism that works on multiple bits.

Data (D) channel - An ISDN channel used primarily for carrying control signals, which can also be used for low-speed data transfers.

Data Communication Equipment (DCE) - The equipment used between a DTE and a network.

Data Encryption Standard (DES) - A symmetric key encryption algorithm, originally developed by IBM.

Data link layer - The second layer in the OSI model, it is responsible for node-to-node delivery.

Data Transmission Equipment (DTE) - A device that is the source or destination of information (usually a computer or router). It is connected to a network via a DCE.

Datagram - An independent data unit in packet switching networks.

de facto standard - A standard that is not approved by a body, but is popular because of its widespread usage.

de jure standard - A standard that is approved by an official body.

Decimal number system - The number scheme that contains ten symbols, 0 to 9.

Decompression - Opposite of compression, this mechanism recovers original data from its compressed version.

Decryption - Opposite of encryption, decryption obtains plain text message from cipher text.

Delta Modulation - A modified version of Pulse Code Modulation, which measures only changes to adjacent signal values.

Demodulator - Opposite of modulator, this equipment transforms analog signals to digital values.

Demultiplexer - Opposite of multiplexer, this equipment separates multiple signals on a single medium to obtain individual signals.

Digital - An encoding mechanism that consists of discrete values.

Digital bit pipe - A high-speed connection (logical pipe).

Digital certificates - Equivalent to real-life passports, these help people and organizations to establish proof of their identity on the Internet.

Digital line - A transmission system that carries discrete signal values.

Digital signature - The equivalent of a paper signature, this concept allows preserve message integrity and provide for non-repudiation.

Digital Subscriber Line (DSL) - The modern technology that allows higher data rates on existing copper wires.

Digital-to-Analog Converter (D-A) - Equipment that transforms digital pulses into analog signals.

Dijkstra algorithm - An algorithm that finds the shortest path to other routers.

Discrete MultiTone (DMT) - An implementation of the DSL technology.

- Distance vector routing - A routing protocol wherein each router sends information to its neighbors about the networks it can reach.
- Distributed database - A database whose portions can physically reside in different locations.
- Distributed Queue Dual Bus (DQDB) - A LAN/MAN protocol.
- DNS server - See Domain Name System
- Domain name - Human-readable name assigned to a computer on the Internet.
- Domain Name System (DNS) - A TCP/IP application that maps human-readable computer names to IP addresses.
- Dotted decimal notation - The mechanism of identifying computers on the Internet using four numbers separated by dots, used in IPv4.
- DSL Access Multiplexer (DSLAM) - The equivalent of modem in DSL.
- Dual bus - Two buses.
- Duplication control - Mechanism that prevents processing of duplicate packets.
- Dynamic Web pages - Web pages that are produced by a server side program.
- Electronic Industries Association (EIA) - An organization that promotes electronics manufacturing organizations.
- Electronic mail (Email) - Mechanism of sending messages electronically using the TCP/IP protocol.
- Encryption - The mechanism of encoding messages to achieve security objectives.
- End to end delivery - Complete delivery of all portions of a message from the sender to the receiver.
- Error control - Detection of errors in transmission.
- Ethernet - A LAN protocol based on CSMA/CD.
- Even parity - A error detection mechanism based on the total number of 1s in a message being 0.
- Extended Binary Coded Decimal Interchange Code (EBCDIC) - An encoding scheme used primarily in IBM mainframes.
- Federal Communications Commission (FCC) - A government agency that regulates radio, television and cable communications.
- Fiber Distributed Data Interface (FDDI) - A high-speed LAN that uses two rings and token passing mechanism.
- File Transfer Protocol (FTP) - A TCP/IP application protocol for transferring files over the Internet.
- Flag - Additional bits in a packet format containing miscellaneous information.
- Flooding - Saturating a network with too many transmissions.
- Flow control - Mechanism for regulating traffic rate on a network.
- Forum - An organization that tests, evaluates or establishes a standard.
- Fourier Analysis - The mathematical reasoning behind the composition of an analog signal.

Fragmentation - Dividing a logical packet into smaller packets as per the MTU of the underlying physical networks.

Frame - The hardware level unit of data transmission.

Frame Relay - A fast packet switching protocol that has the two lowermost OSI layers.

Freeware - Software that can be used freely without paying any licensing fees.

Frequency - The number of times a signal changes its value in one second.

Frequency Division Multiplexing (FDM) - A multiplexing technique wherein multiple signals are combined into a single channel using different frequency groups.

Frequency Modulation (FM) - An analog-to-analog encoding method in which the frequency of the carrier wave is a function of the modulating wave

Frequency Shift Keying (FSK) - A digital-to-analog encoding method in which the frequency of the carrier wave represents the value of the digital bits.

Full duplex transmission - A transmission facility wherein both parties can transmit at the same time.

Gateway - A device that connects two networks which can have the same or different communication protocols.

Geosynchronous satellites - A fixed orbit for satellites moving over the earth surface.

Go-back-n - An error control mechanism in which all the frames including and after the one in error must be retransmitted.

Group - Combination of smaller transmission lines.

Guard band - The unused portion between two FDM channels.

Guided transmission medium - A transmission medium that has physical wiring.

Half duplex transmission - A transmission system wherein both parties can transmit, but not at the same time.

Handoff - The mechanism in cellular telephony wherein a cell office hands a weak call over to an adjacent cell office.

Hardware address - The physical address of a computer, usually hard coded into its Network Interface Card.

Hash - The *fingerprint* of a message.

Hertz (Hz) - Unit of measuring frequency in number of cycles per second.

High-level Data Link Control (HDLC) - A bit-oriented data link protocol used in X.25.

Home page - The first Web page on a Web site.

Host id - The portion of IP address that identifies a host on a network.

Host number - *Same as host id.*

Hub - A central device in a star network.

Huffman code - A data compression algorithm that uses variable length codes based on the frequency of characters.

Hybrid (H) channel – The least used channel type in ISDN.

Hybrid topology – A topology made up of multiple basic topologies.

Hyper link – Mechanism to establish link between two Web pages.

Hyper Text Markup Language (HTML) – Language used for creating Web pages.

Hyper Text Transfer Protocol (HTTP) – A TCP/IP application layer protocol used for requesting for and sending Web pages.

Improved Mobile Telephone System (IMTS) – A cellular telephone system.

In-band signaling – Method of sending data and control signals over the same transmission line.

Institute of Electrical and Electronics Engineers (IEEE) – An organization promoting standards related to electrical and electronics systems.

Integrated Services Digital Network (ISDN) – A digital transmission system that uses the existing copper wires to provide higher data rates for voice, data and simple multimedia applications.

Interface – The boundary between two equipments.

International Standards Organization (ISO) – The worldwide standards body that covers computing and non-computing standards.

International Telecommunications Union-Telecommunications Standards Sector (ITU-T) – Earlier knows as CCITT, it is a standards body.

internet – A collection of networks interconnected by routers or gateways.

Internet – The worldwide network of computers and computer networks.

Internet Assigned Number Authority (IANA) – The body who decides on IP address allocation strategies.

Internet Control Message Protocol (ICMP) – A TCP/IP protocol that deals with errors.

Internet Mail Access Protocol (IMAP) – A modified version of *POP*.

Internet Protocol (IP) – The network layer protocol in TCP/IP.

Internet Service Provider (ISP) – An organization that provides Internet services to its customers.

Internetwork – *See internet*.

Internetworking – Mechanism of connecting networks to each other.

IP address – A 32-bit number In Ipv4, and a 128-bit number in Ipv6, this identifies a computer on the Internet.

IP Next Generation (IPng) – The next version of the IP protocol.

IP version 4 (IPv4) – The current version of the IP protocol.

IP version 6 (IPv6) – *Same as Ipng*.

Java – A programming language developed by Sun Microsystems, and used to create active Web pages.

Java applet – A small client-side Java program.

Java Server Pages (JSP) – A dynamic Web pages technology.

Java Servlets – The precursor to JSP.

Jumbogroup – The highest level in the AT&T analog hierarchy.

Key pair – A pair of private and public key for an individual or an organization.

Leased line – A dedicated transmission system that guarantees predetermined data rates.

Link Access Procedure Balanced (LAPB) – A LAP protocol used in balanced mode.

Link Access Procedure for the D channel (LAPD) – A LAP protocol used in D channel of ISDN.

Link state database – A database used in link state routing that is common to all routers.

Link State Packet (LSP) – A small packet containing routing information, sent by one router to all other routers in link state routing.

Link state routing – A routing method in which each router shares its knowledge about changes in its neighborhood with all other routers.

Local Area Network (LAN) – A network of computers in the same building or area.

Local login – Mechanism of a user using the local terminal.

Local loop – The link between a subscriber and the local telephone exchange.

Logical address – The address used by the network layer protocols.

Longitudinal Redundancy Check (LRC) – An error detection mechanism that performs parity checks on rows and columns of the data unit.

Loss-less compression – A compression technique that ensures that there is no change to the original data.

Lossy compression – A compression technique that can result into small losses to the original data.

Mail Transfer Agent (MTA) – A mail system that accepts a message, examines it and sends it forward.

Mailbox – The logical representation of a user's email storage on a computer's disk.

Mastergroup – One of the levels in the AT&T analog hierarchy.

Maximum Transmission Unit (MTU) – The largest size of a data unit that a particular network can handle.

Media Access and Control (MAC) – The lower sub layer in the data link layer that defines the access method and access control in different LAN protocols.

Mesh topology – A topology in which each node is connected to every other node.

Message digest – The hash, or fingerprint, of a message, which is used in digital signatures.

Message switching – The switching system wherein the whole message is stored in a switch, and then forwarded to the next host/switch.

Metropolitan Area Network (MAN) – A network that can span across the geographical area of a city.

Microwave – Electromagnetic waves ranging from 2 GHz to 40 GHz.

Mobile Telephone Switching Office (MTSO) – The office responsible for coordinating calls between a cell and the central office.

Modem – Abbreviation of modulation and demodulation, this device connects a computer to an analog transmission line.

Morse code – An encoding method to represent alphabets using dots and dashes.

Mosaic – The first Web browser.

Multimedia – The technology of representing sound, pictures and video as bits.

Multiplexer – The device that is used in multiplexing.

Multiplexing – The process of combining incoming signals from multiple channels and sending them across a single transmission medium.

Multipurpose Internet Mail Extensions (MIME) – Extension added to the basic email system to allow for non-textual data support.

Narrowband ISDN (B-ISDN) – The *basic* ISDN system that allows for BRI and PRI interfaces.

Negative Acknowledgement (NAK) – A message that indicates the rejection of data.

Network id – The network identifier portion of IP address.

Network Interface Card (NIC) – The interface between a host and network.

Network layer – The third layer in the OSI model that takes care of end-node delivery.

Network Termination 1 (NT1) – The ISDN device between a user site and central office.

Network Termination 2 (NT2) – The ISDN device that performs functions in the first three layers of the OSI model.

Network topology – The way computers are connected to each other to form a network.

Network Virtual Terminal (NVT) – A TCP/IP protocol that allows remote login.

Network-Network Interface (NNI) – The interface between two networks in ATM.

Next-hop table – The table maintained by a router for delivering packets to the next destination.

Node – An addressable communication device, such as a computer or a router.

Node to node delivery – Delivery of packets from one node to the next one.

Noise – Random electrical signals that can be picked up during transmission, causing degradation or distortion of data.

Non repudiation – Mechanism that does not allow a sender to refute having sent a message.

Null suppression – A compression technique that replaces nulls in a message with their count.

Nyquist theorem – A theorem put forth by Nyquist, which says that to recover the original analog signal from a sampled signal, the sampling rate must be at least twice the highest frequency component of the original signal.

Octet – Eight bits

Odd parity – An error detection method in which an extra bit is added to the data unit so that the sum of all 1-bits becomes odd.

Open source code – The movement towards free software.

Open Systems Interconnection (OSI) – A seven-layer model for data communications defined by ISO.

Optical fiber – A high-bandwidth transmission system that sends data bits as light pulses.

Out-band signaling – A transmission system that has two separate channels, one for data portion and one for control signals.

Packet – The network layer term for a data unit.

Packet Layer Protocol (PLP) – The network layer in X.25 protocol.

Packet lifetime – The number of hops a packet can make before it is discarded.

Packet switching – Data transmission using a packet switched network.

Pair of sockets – The two end points of a transmission.

Parallel transmission – Transmission mechanism where multiple bits are sent at the same time, each on a separate wire.

Parity bit – The additional bit added to a packet for error detection.

Parity checking – The mechanism of error detection using parity bits.

Passive hub – A hub used for connection, but not regeneration of a signal.

Passive open – The mechanism whereby a server waits for active connections from clients.

Period – The time required for a signal to complete one cycle.

Permanent Virtual Circuit (PVC) – A virtual circuit that is usually set up in hardware, and is not created on demand.

Phase – The relative signal of a position with respect to time.

Phase Shift Keying (PSK) – A digital-to-analog encoding mechanism in which the phase of the carrier signal is changed with respect to the incoming digital pulses.

Physical address – The hardware address of a computer.

Physical layer – The lowest layer in the OSI protocol model, which deals with the physical characteristics of a transmission system.

Pixel – Abbreviation for picture element, which is the smallest addressable unit on a computer screen.

Plain Old Telephone Service (POTS) – The traditional analog telephone network.

Plain text – The normal text before it is encrypted to form cipher text.

Plug-in – A piece of software that is downloaded from a Web server to a browser on demand.

Point-to-point communication – A dedicated connection between two devices.

Point-to-point protocol (PPP) – The protocol used in a dial-up connection between a user's modem and an ISP's modem.

Port – A number that identifies a particular application on a client or a server.

Portal – A Web site that is a collection of multiple utility Web pages.

Positive Acknowledgement (ACK) – An acknowledgment mechanism that involves acknowledging the correct receipt of a packet.

Post Office Protocol (POP) – An email protocol that allows retrieval of email messages from an email server using a remote connection.

Preamble – Bits used for synchronization.

Presentation layer – The sixth layer of the OSI model used for translation, encryption, authentication and compression.

Pretty Good Privacy (PGP) – A security protocol for emails.

Primary Rate Interface (PRI) – 23B + D ISDN connection for business users.

Privacy Enhanced Mail (PEM) - A security protocol for emails.

Private Branch eXchange (PBX) – A private telephone system set up.

Private key – One of the two keys used in asymmetric key encryption, which must be held securely by a user.

Private key symmetric encryption – An encryption mechanism where the same key is used for encryption and decryption.

Protocol – A set of rules and conventions used in data communication.

Proxy server – A server set up by an organization to cache and provide frequently accessed Web pages, and also to set up filters.

Public key – One of the keys in asymmetric key encryption.

Public key encryption - Asymmetric key encryption mechanism where the decryption key is different from the encryption key.

Pulse Code Modulation (PCM) – Digitizing analog signals using sampling and quantizing techniques.

Quadrature Amplitude Modulation (QAM) – A digital-to-analog encoding technique where the amplitude and the phase values of the carrier wave can be changed to represent digital information.

Quantizing – Measuring sampled analog signals to get their numeric equivalents.

Reference point – The interface between two ISDN devices.

Regenerator – A device that accepts input signals and regenerates them.

Regulatory agencies – Agencies that protect public interests.

Remote login – Process of connecting to a remote computer as if it is a local computer.

Repeater – A device that regenerates a digital signal so that it can be carried to longer distances.

Resolver – A piece of software that is used in Domain Name System.

Reverse Address Resolution Protocol (RARP) – The TCP/IP protocol which accepts a physical address and returns back the corresponding IP address.

Ring topology – A topology wherein all the devices are connected to each other to form a ring.

Router – An interconnecting device that operates in the first three layers of the OSI model, and forwards packets from one network to another.

Routing table - A table maintained by a router to decide how to route the incoming packets to its ultimate destination.

RSA algorithm – A public key encryption algorithm invented by Rivest, Shamir and Adleman.

Run Length Encoding (RLE) – A data compression mechanism wherein a series of repeating characters are replaced by a single character the number of times it was present in the original message.

Sampling – The first process in Pulse Code Modulation where an analog signal is *measured* periodically.

Search engine – Software programs that accept search strings and return names of Web pages that contain those strings.

Secret key encryption – An encryption scheme where the encryption and decryption key is the same.

Segmentation – Splitting a message into smaller blocks, usually done in the transport layer.

Segmentation And Reassembly (SAR) – The lower sub layer in AAL of ATM networks.

Serial Line Internet Protocol (SLIP) – Protocol between dial up modem and ISP modem.

Serial transmission - Transmission mechanism wherein one bit is sent at a time, using a single wire.

Server - A program that provides services to other programs (which are called as clients).

Session layer – The fifth layer of the OSI model, responsible for establishing, managing and terminating logical connections between the sender and the receiver.

Shannon capacity – Defines the maximum data rate of a channel.

Shielded Twisted Pair (STP) – Twisted-pair cable enclosed in mesh shield that prevents electromagnetic interference.

Signal – Electromagnetic wave propagating across a transmission medium.

Simple Mail Transfer Protocol (SMTP) – TCP/IP protocol for electronic mail transfer across the Internet.

Simplex transmission – Transmission system where only one side can transmit, the other side can only receive.

Sliding window - A protocol that allows several data units to transit before an acknowledgement is received.

Socket – Identifies the end point of a connection, and is a combination of IP address and port number.

Source address – The address of the sender of a message.

Standards creation committees – A group that produces a model for protocols and standards.

Star topology - A topology in which all stations are attached to a central device (hub).

Start bit – A bit indicating the start of transmission in asynchronous transmission.

Stateless protocol – A protocol that has a single request and a single response mechanism.

Static routing – Routing scheme where the routing tables do not change.

Statistical TDM – A type of multiplexing depending on how much data is to be transmitted by a station.

Stop bit – One or more bits that indicate the end of transmission in asynchronous transmission.

- Stop-and-wait - Flow control mechanism where each data unit must be acknowledged before the next one can be sent.
- Store and forward – Same as message switching.
- Supergroup – A level in the AT&T hierarchy.
- Switch – A device connecting multiple communication lines.
- Switched Multimegabit Data Services (SMDS) – A fast packet-switching MAN protocol.
- Switched Virtual Circuit (SVC) – A virtual circuit mechanism where the virtual circuits are created on demand.
- Synchronous TDM – Time division multiplexing mechanism wherein the time slots for multiplexing are of pre-fixed duration.
- Synchronous transmission – A transmission system that needs the sender and the receiver to coordinate before transmission can take place.
- T lines – A hierarchy of digital lines used to carry speech and other signals in digital form, and consists of T-1 to T-4 lines.
- Tag – Indicates the beginning and end of a specific command in the HTML language.
- TELNET – A remote login protocol in TCP/IP.
- Terminal Adapter (TA) – A device that connects non-ISDN devices to ISDN network.
- Terminal Equipment 1 (TE1) – A standard ISDN terminal.
- Terminal Equipment 2 (TE2) – A non-ISDN terminal.
- Three way handshake – A protocol for establishing or terminating a logical connection between two end points.
- Time Division Multiplexing (TDM) – The mechanism of combining signals from multiple incoming lines into a single output line to share time.
- Token bus – A LAN technology that uses token passing access method and bus topology.
- Token frame – The packet used in token passing access method.
- Token Ring - A LAN technology that uses token passing access method and bus topology.
- Transceiver – A device that both transmits and receives.
- Translation – Change from one protocol to another.
- Transmission Control Protocol (TCP) – The transport layer reliable delivery protocol in TCP/IP.
- Transmission Control Protocol/Internet Protocol (TCP/IP) – The data communications protocol used in the Internet.
- Transmission medium – The wire that carries signals between the communicating parties.
- Transport layer – The fourth layer in the OSI model, responsible for reliable end-to-end delivery.
- Tree topology – A topology in which the nodes are attached to each other to create a hierarchy of hubs.
- Trivial File Transfer protocol (TFTP) – An unreliable file transport protocol in TCP/IP.
- Twisted-pair wire – A transmission medium that consists of two insulated copper wires in a twisted format.

Unguided transmission media – A transmission medium that has no physical boundaries.

Uniform Resource Locator (URL) – A string identifier that identifies a page on the World Wide Web.

Universal address – Address of a computer or router that is independent of the underlying physical address.

Unshielded Twisted Pair (UTP) – A cable containing wires that are twisted together to reduce noise and crosstalk.

Uplink – Transmission from an earth station to a satellite.

User agent – A component in some protocols like X.400 that interacts with the user.

User Datagram Protocol (UDP) – A connectionless, unreliable transport layer protocol in TCP/IP.

User-Network Interface (UNI) – The interface between a user and ATM network.

Vertical Redundancy Check (VRC) – An error detection mechanism based on the parity checking of each character.

Virtual circuit – A logical circuit between the sender and the receiving computer. All the packets in the transmission then follow the same route and always arrive in order.

Virtual Circuit Identifier (VCI) – A field in the ATM cell header that identifies a virtual circuit.

Virtual Path Identifier (VPI) – A field in the ATM cell header that identifies a virtual path.

Voice Over Frame Relay (VOFR) – The method of sending voice traffic over a Frame Relay network.

Web browser – A software program that interprets and displays the contents of HTML Web pages.

Web page – HTML contents that are displayed in a Web browser.

Web server – A server that hosts HTML pages and dispatches them on request.

Web site – A collection of Web pages.

Well-known ports – Standard port numbers used by applications.

Wide Area Network (WAN) – A network that spans across large physical regions.

World Wide Web (WWW) – An Internet application that allows users to view Web pages and move from one Web page to another.

X.21 – Interface between DTE and DCE.

X.25 – Interface between a DTE and a packet switching network.

References

Author	Title	Publication	Year
Forouzan, Behrouz	Introduction to data communications and networking	Tata McGraw Hill	1999
Stallings, William	Data and computer communications	Prentice Hall of India	1996
Tanenbaum, Andrew	Computer networks	Prentice Hall of India	1997
Buts, Regis and Gregory, Donald	Voice and data communications handbook	Tata McGraw Hill	2000
Comer, Douglas	Computer networks and Internets	Prentice Hall	2000
Comer, Douglas	TCP/IP (Volume I) - Principles, Protocols and Architecture	Prentice Hall of India	1999
Forouzan, Behrouz	TCP/IP protocol suite	Tata McGraw Hill	2000
Comer, Douglas	The Internet book	Prentice Hall of India	
Wood, David	Programming Internet email	O'Reilly and Associates	2000
Held, Gilbert	Understanding data communications	Prentice Hall of India	1996
Gralla, Preston	How the Internet works	QUE Corporation	2000
Moulton, Pete	The telecommunications survival guide	Pearson Education Asia	2001
Walrand, Jean and Varaiya, Pravin	High performance communication networks	Hardcourt Asia Private Limited	2000
Stallings, William	ISDN,		
Kurose, James and Ross, Keith	Computer networking	Pearson Education Asia	2001
Norton, Peter and Kearns, Dave	Peter Norton's guide to networking	SAMS Publishing	1999
Hall, Eric	Internet core protocols	O'Reilly and Associates	2000
Martin, James	Telecommunications and the computer	Prentice Hall of India	1992

Index



100BASE-TX 186

A

Active open 408
Active Web page 479
Adaptive Differential Pulse Code Modulation (ADPCM) 34
Address Resolution Protocol (ARP) 369, 380
Advanced Mobile Phone System (AMPS) 117
ALOHA 206
Amplifier 24
Amplitude 10
Amplitude modulation 27
Amplitude Shift Keying (ASK) 27
Analog signal 8
Analog to Digital (A/D) convertor 33
Application gateway 99
Application layer 171
ARPA 325
ARPANet 325
Asymmetric Key Encryption 90
Asynchronous communication 43
Asynchronous Transfer Mode (ATM) 3, 265
Attenuation 24, 62
Authentication 86

B

Bandwidth 9, 15
Baud 30
Best-effort delivery 367

Bit rate 30
Bluetooth 3
Bluetooth 291
Bridge 316
Broadband ISDN (B-ISDN) 231
Burst error 63
Bus topology 128

C

Cable modem 343, 348
Caesar Cipher 88
Carrier Sense Multiple Access with Collision Detect (CSMA/CD) 181
Cell Relay 266
Cellular communication 117
Certification Authority (CA) 93
Channel allocation 212
Checksum 64
Cipher text 87
Circuit switching 131
Client computer 326
Coaxial cable 106, 108
Code Division Multiple Access (CDMA) 116
Codec 33
Collision 155, 181
Computer network 2
Computer Telephony Integration (CTI) 487
Confidentiality 85
Configurable address 361
Congestion 257

Crosstalk 107
 Cryptography 87
 Cyclic Redundancy Check (CRC) 66

D

Data Communications Equipment (DCE) 238, 350
 Data Encryption Standard (DES) 89
 Data link layer 164
 Data Transmission Equipment (DTE) 238, 350
 Datagram approach 133
 Decryption 87
 Delay distortion 62
 Delta modulation 34
 Demultiplexer 2
 Demultiplexing 50
 Dial-up program 332
 Digital cellular telephone 120
 Digital certificate 93
 Digital envelope 101
 Digital signal 8
 Digital signature 94
 Digital Subscriber Line (DSL) 343
 Digital to Analog (D/A) convertor 33
 Distance vector routing 139
 Distributed Queue Dual Bus (DQDB) 195
 Domain name 421
 Domain Name System (DNS) 421, 425
 Dotted decimal notation 375
 Duplication control 401
 Dynamic address 362
 Dynamic channel allocation 213
 Dynamic routing 147
 Dynamic Web page 478

E

Electronic mail 429
 Email 429
 Encryption 87
 Error control 155, 400
 Ethernet 177
 Exposed station problem 303

F

Fast Ethernet 186
 Fiber Distributed Data Interface (FDDI) 192
 File Transfer Protocol (FTP) 325, 445
 Firewall 97
 Fixed channel allocation 213
 Flow control 155
 Fourier analysis 14
 Fragmentation 390
 Frame Relay 3, 248
 Frequency 11
 Frequency Division Multiple Access (FDMA) 114
 Frequency Division Multiplexing (FDM) 50
 Frequency Hopping Spread Spectrum (FHSS) 295
 Frequency modulation 28
 Frequency Shift Keying (FSK) 27
 Full-duplex communication 46

G

Gateway 324
 Geosynchronous satellites 113
 Gigabit Ethernet 186
 Go-back-n 71
 Guard bands 36, 52
 Guided media 106

H

Half-duplex communication 46
 Hamming Code 67
 Handoff 118
 Hardware address 179
 Hidden station problem 302
 High-level Data Link Control (HDLC) 235
 Huffman code 81
 Hybrid channel allocation 213
 Hybrid topology 128
 Hyper Text Markup Language (HTML) 458, 465
 Hyper Text Transfer Protocol (HTTP) 458
 Hyperlink 467
 Hypertext 458

I

IEEE 802.1 288
 IEEE 802.11 3
 IEEE 802.11 290, 298
 IEEE 802.15 291
 IEEE 802.2 289
 IEEE 802.3 289
 IEEE 802.4 289
 IEEE 802.5 290
 Improved Mobile Telephone System (IMTS) 117
 Indirect delivery 374
 Infrared communication (IR) 290
 Integrated Services Digital Network (ISDN) 3, 217
 Integrity 86
 Interactive television 491
 Internet 3, 308
 Internet Control Message Protocol (ICMP) 384
 Internet Mail Access Protocol (IMAP) 443
 Internet Service Provider (ISP) 329
 Internetworking 308
 IP address 331, 341, 360, 366, 371
 IP Control Protocol (IPCP) 343
 IP datagram 365
 IP version 4 (IPv4) 365
 IP version 6 (IPv6) 365, 379

K

Key 87
 Key pair 90

L

Leaky bucket algorithm 258
 Learning bridge 320
 Leased line 343
 Lempel-Ziv code 85
 Line Control Protocol (LCP) 342
 Link state routing 143
 Local Area Network (LAN) 3, 176
 Logical address 324, 363
 Logical carrier sensing 304
 Longitudinal Redundancy Check (LRC) 65

Loss control 401
 Lossless compression 85
 Lossy compression 85

M

Mailbox 431
 Maximum Transmission Unit (MTU) 389
 Medium Access Control (MAC) 124, 177, 189, 214
 Mesh topology 124
 Message switching 136
 Metropolitan Area Network (MAN) 3, 176, 195
 Microwave communication 112
 Mobile phone 117
 Mobile Telephone Switching Office (MTSO) 117
 Modem 2, 26, 332
 Multicasting 374
 Multiplexer 2, 50
 Multiplexing 50
 Multiport bridge 321
 Multipurpose Internet Mail Extensions (MIME) 444

N

Narrowband ISDN (N-ISDN) 231
 Near video on demand 491
 Negative Acknowledgement (NAK) 70
 Network Access Point (NAP) 330
 Network Access Provider (NAP) 329
 Network Control Protocol (NCP) 343
 Network Interface Card (NIC) 178
 Network layer 166
 Network topology 124
 Noise 63
 Non-repudiation 87
 Null suppression 79
 Nyquist theorem 35

O

Open Standards Interconnection (OSI) 5
 Optical fiber 106, 108
 OSI model 159

P

Packet filter 98
 Packet switching 133
 Parallel communication 40
 Parity bit 43
 Parity check 64
 Passive open 408
 Password Authentication Protocol (PAP) 343
 Period 10
 Persistent connection 413
 Phase 13
 Phase Shift Keying (PSK) 27
 Physical address 179
 Physical carrier sensing 303
 Physical layer 156, 163, 215
 Plain Old Telephone Service (POTS) 344
 Plain text 87
 Plug-in 472
 Point-to-point protocol (PPP) 341
 Port 404
 Port-to-port communication 402
 Positive Acknowledgement (ACK) 70
 Post Office Protocol (POP) 438
 Presentation layer 170
 Pretty Good Privacy (PGP) 100, 445
 Privacy Enhanced Mail (PEM) 99, 445
 Private key 90
 Protocol 2, 3, 4
 Proxy server 464
 Public key 90
 Public key cryptography 90
 Pulse Code Modulation (PCM) 32
 Pure ALOHA 206
 Push-to-talk system 117

Q

Quadrature Amplitude Modulation (QAM) 29
 Quadrature Phase Shift Keying (QPSK) 30
 Quantization error 33
 Quantizing 33
 Queuing theory 172

R

Radio layer 294
 Rail-fence technique 88
 Real-time Transport Control Protocol (RTCP) 492
 Real-time Transport Protocol (RTP) 492
 Remote Access Server (RAS) 332
 Repeater 25, 315
 Reverse Address Resolution Protocol (RARP) 383
 Ring topology 127
 Router 136, 321
 Routing 2, 136
 RS-232 351
 RS-449 351
 RSA algorithm 91
 Run Length Encoding (RLE) 79

S

Sampling 33
 Satellite communication 113
 Search engine 464
 Secure Socket Layer (SSL) 95
 Sequence control 401
 Serial communication 41
 Serial Line Internet Protocol (SLIP) 341
 Server computer 326
 Session Initiation Protocol (SIP) 496
 Session layer 169
 Shannon capacity 120
 Shielded Twisted Pair (STP) 107
 Signal 2
 Signal propagation 2, 6
 Simple bridge 320
 Simple coding scheme 78
 Simple Mail Transfer Protocol (SMTP) 438
 Simplex communication 46
 Single-bit error 63
 Skew 41
 Sliding window protocol 71
 Slotted ALOHA 206
 SMTP server 334

Socket 406
 Spooling 432
 Star topology 125
 Start bit 43
 Static address 361
 Static channel allocation 213
 Static routing 147
 Static Web page 477
 Statistical compression 80
 Statistical TDM 54
 Stop bit 43
 Stop-and-wait 70
 Subnetting 377
 Substitution cipher 88
 Switched Multimegabit Data Services (SMDS) 195
 Switching 130
 Symmetric Key Encryption 89
 Synchronous communication 44
 Synchronous Digital Hierarchy (SDH) 352
 Synchronous Optical Networking (SONET) 352
 Synchronous TDM 53

T

Telephone Central Office (TCO) 117
 TELNET 325, 474
 Three-way handshake 409
 Time Division Multiplexing (TDM) 50
 Time Division Multiplexing Access (TDMA) 115
 Time out 156
 Token bucket algorithm 259
 Token Ring 188
 Transceiver 178
 Transmission Control Protocol/Internet Protocol (TCP/IP) 3, 309, 399
 Transmission medium 1, 3, 106
 Transport layer 167
 Transport Layer Security (TLS) 95
 Transposition cipher 88
 Tree topology 125
 Trivial File Transfer Protocol (TFTP) 452
 Twisted pair wire 106

U

Unguided media 111
 Uniform Resource Locator (URL) 457
 Universal Asynchronous Receiver Transmitter (UART) 42
 Universal Serial Bus (USB) 351
 Universal Synchronous Asynchronous Receiver Transmitter (USART) 42
 Unshielded Twisted Pair (UTP) 107
 User Datagram Protocol (UDP) 414

V

Vertical Redundancy Check (VRC) 64
 Video on demand 489
 Virtual circuit 134
 Virtual LAN 184
 Virtual network 313
 Voice Over Internet Protocol (VOIP) 487

W

Wavelength Division Multiplexing (WSM) 56
 Web browser 326
 Web server 326, 423
 Web site 423, 457
 Well-known port 407
 Wide Area Network (WAN) 3, 176, 199
 WiFi 3
 WiMAX 3
 Wireless communication 111, 287
 Wireless Local Area Networks (Wireless LAN) 288
 Wireless Metropolitan Area Networks (Wireless MAN) 288
 Wireless Personal Area Networks (Wireless PAN) 287
 Wireless Wide Area Networks (Wireless WAN) 288
 World Wide Web (WWW) 326

X

X.21 245
 X.25 3, 205, 235