INTERNATIONAL EDITION

Data and Computer Communications

TENTH EDITION

William Stallings

ALWAYS LEARNING

PEARSON

Data and Computer Communications

Tenth Edition

William Stallings

International Edition contributions by

Moumita Mitra Manna

Bangabasi College, Kolkata

PEARSON

Boston Columbus Indianapolis New York San Francisco Upper Saddle River
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montréal Toronto
Delhi Mexico City São Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

Editorial Director, ECS: Marcia Horton
Executive Editor: Tracy Johnson (Dunkelberger)
Editorial Assistant: Jenah Blitz-Stoehr
Director of Marketing: Christy Lesko
Marketing Manager: Yez Alayan
Marketing Assistant: Jon Bryant
Director of Program Management: Erin Gregg
Program Management-Team Lead: Scott Disanno
Program Manager: Carole Snyder
Project Management-Team Lead: Laura Burgess
Project Manager: Robert Engelhardt
Publishing Operations Director, International Edition: Angshuman Chakraborty
Manager, Publishing Operations, International Edition: Shokhi Shah Khandelwal
Associate Print and Media Editor, International Edition: Anuprova Dey Chowdhuri
Acquisitions Editor, International Edition: Sandhya Ghoshal
Publishing Administrator, International Edition: Hema Mehta
Project Editor, International Edition: Daniel Luiz
Editorial Assistant, International Edition: Sinjita Basu
Procurement Specialist: Linda Sager
Senior Manufacturing Controller, Production, International Edition: Trudy Kimber
Art Director: Jayne Conte
Cover Designer: Karen Noferi
Cover Photo Credit: Fotolia/Female photographer
Cover Printer: Courier Westford

Pearson Education Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England

and Associated Companies throughout the world

Visit us on the World Wide Web at: www.pearsoninternationaleditions.com

ISBN 10: 1-29-201438-5
ISBN 13: 978-1-29-201438-8

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

10 9 8 7 6 5 4 3 2 1
14 13 12 11 10

Typeset in Times LT Std-Roman by Integra Software Services Pvt. Ltd.

Printed and bound by Courier Westford in The United States of America

PEARSON

For Tricia

CONTENTS

[1]Online chapters and appendices are Premium Content, available via the access card at the front of this book.

PREFACE

WHAT'S NEW IN THE TENTH EDITION

Since the ninth edition of this book went to press, the pace of change in this field continues unabated. In this new edition, I try to capture these changes while maintaining a broad and comprehensive coverage of the entire field. To begin the process of revision, the ninth edition of this book was extensively reviewed by a number of professors who teach the subject and by professionals working in the field. The result is that, in many places, the narrative has been clarified and tightened, and illustrations have been improved.

Beyond these refinements to improve pedagogy and user friendliness, there have been major substantive changes throughout the book. The chapter organization has been changed somewhat so that now the material is organized into two Units, with Unit Two containing more advanced material and an expansion of the material related to the Internet. Beyond this organizational revision, the most noteworthy changes include the following:

- **Sockets programming:** A new section introduces sockets programming. Plus a number of sockets programming assignments, with sample solutions, are available for instructors.
- **Software-defined networks:** A new section covers this widely used technology.
- **Wireless transmission technology:** The book provides a unified treatment of important transmission technologies for wireless networks, including FDD, TDD, FDMA, TDMA, CDMA, OFDM, OFDMA, SC-FDMA, and MIMO.
- **4G cellular networks:** A new section covers 4G networks and the LTE-Advanced specification.
- **Gigabit Wi-Fi:** A new section covers the two new Wi-Fi standards, IEEE 802.11ac and 802.11ad, which provide Wi-Fi in the Gbps range.
- **Fixed broadband wireless access:** New sections cover fixed broadband wireless access to the Internet and the related WiMAX standard.
- **Forward error correction:** Forward error correction techniques are essential in wireless networks. This new edition contains substantially expanded coverage of this important topic.
- **Personal area networks:** New sections cover personal area networks and the Bluetooth standard.
- **Dynamic Host Configuration Protocol (DHCP):** DHCP is a widely used protocol that enables dynamic IP address assignment. A new section covers this protocol.
- **Datagram Congestion Control Protocol:** DCCP is a new protocol that meets the needs of multimedia applications for a congestion control transport protocol without the overhead of TCP. A new section covers DCCP.

- **Protocol Independent Multicast (PIM):** PIM, the most important Internet multicast routing algorithm, is covered in a new section.
- **Quality of service (QoS) architectural framework:** A new section covers ITU-T Recommendation Y.1291, which provides an overall framework for provision of Internet QoS facilities.
- **Electronic mail:** The section on e-mail in Chapter 24 has been expanded to include a discussion of the standard Internet mail architecture.
- **Animations:** As a powerful aid to understanding the material, over 150 online animations are provided covering a wide range of topics from the book. An icon at the beginning of many chapters indicates that supporting animations are available to enhance the student's understanding.
- **Learning objectives:** Each chapter now begins with a list of learning objectives.
- **Sample syllabus:** The text contains more material than can be conveniently covered in one semester. Accordingly, instructors are provided with several sample syllabi that guide the use of the text within limited time (e.g., 16 weeks or 12 weeks). These samples are based on real-world experience by professors with the ninth edition.

In addition, the material that carries over from the ninth edition has been revised, with new figures and revised and updated content.

OBJECTIVES

This book attempts to provide a unified overview of the broad field of data and computer communications. The organization of the book reflects an attempt to break this massive subject into comprehensible parts and to build, piece by piece, a survey of the state of the art. The book emphasizes basic principles and topics of fundamental importance concerning the technology and architecture of this field and provides a detailed discussion of leading-edge topics.

The following basic themes serve to unify the discussion:

- **Principles:** Although the scope of this book is broad, there are a number of basic principles that appear repeatedly as themes and that unify this field. Examples are multiplexing, flow control, and error control. The book highlights these principles and contrasts their application in specific areas of technology.
- **Design approaches:** The book examines alternative approaches to meeting specific communication requirements.
- **Standards:** Standards have come to assume an increasingly important, indeed dominant, role in this field. An understanding of the current status and future direction of technology requires a comprehensive discussion of the related standards.

SUPPORT OF ACM/IEEE COMPUTER SCIENCE CURRICULA 2013

The book is intended for both an academic and a professional audience. For the professional interested in this field, the book serves as a basic reference volume and is suitable for self-study. As a textbook, it can be used for a one-semester or two-semester course. This edition is designed to support the recommendations of the current (February 2013) draft version of the ACM/IEEE Computer Science Curricula 2013 (CS2013). The CS2013 curriculum recommendation includes Networking and Communication (NC) as one of the Knowledge Areas in the Computer Science Body of Knowledge. CS2013 divides all course work into three categories: Core-Tier 1 (all topics should be included in the curriculum), Core-Tier-2 (all or almost all topics should be included), and elective (desirable to provide breadth and depth). In the NC area, CS2013 includes two Tier 1 topics and five Tier 2 topics, each of which has a number of subtopics. This text covers all of the topics and subtopics listed by CS2013 in these two tiers.

Table P.1 shows the support for the NC Knowledge Area provided in this textbook.

Table P.1 Coverage of CS2013 Networking and Communication (NC) Knowledge Area

Topic	Chapter Coverage
Introduction (Tier 1) —Organization of the Internet (Internet Service Providers, Content Providers, etc.) —Switching techniques (Circuit, packet, etc.) —Physical pieces of a network (hosts, routers, switches, ISPs, wireless, LAN, access point, firewalls, etc.) —Layering principles (encapsulation, multiplexing) —Roles of the different layers (application, transport, network, datalink, physical)	1-Data Communications 2-Protocol Architecture 9-WAN Technology
Networked Applications (Tier 1) —Naming and address schemes (DNS, IP addresses, Uniform Resource Identifiers, etc.) —Distributed applications (client/server, peer-to-peer, cloud, etc.) —HTTP as an application layer protocol —Multiplexing with TCP and UDP —Socket APIs	24-Electronic mail, DNS, HTTP 2-Protocol Architecture
Reliable Data Delivery (Tier 2) —Error control (retransmission techniques, timers) —Flow control (acknowledgments, sliding window) —Performance issues (pipelining) —TCP	6-Error Detection and Correction 7-Data Link Control 15-Transport Protocols

Table P.1 Continued

Topic	Chapter Coverage
Routing And Forwarding (Tier 2) —Routing versus forwarding —Static routing —Internet Protocol (IP) —Scalability issues (hierarchical addressing)	19-Routing 14-The Internet Protocol
Local Area Networks (Tier 2) —Multiple Access Problem —Common approaches to multiple access (exponential-backoff, time division multiplexing, etc.) —Local Area Networks —Ethernet —Switching	11-Local Area Network Overview 12-Ethernet
Resource Allocation (Tier 2) —Need for resource allocation —Fixed allocation (TDM, FDM, WDM) versus dynamic allocation —End-to-end versus network-assisted approaches —Fairness —Principles of congestion control —Approaches to Congestion (Content Distribution Networks, etc.)	8-Multiplexing 20-Congestion Control 21-Internetwork QoS
Mobility (Tier 2) —Principles of cellular networks —802.11 networks —Issues in supporting mobile nodes (home agents)	10-Cellular Wireless Networks 13-Wireless LANs

PLAN OF THE TEXT

The book is divided into two units, comprising nine parts, which are described in Chapter 0:

- Unit One: Fundamentals of Data Communications and Networking
  - —Overview
  - —Data Communications
  - —Wide Area Networks
  - —Local Area Networks
  - —Internet and Transport Layers
- Unit Two: Advanced Topics in Data Communications and Networking
  - —Data Communications and Wireless Networks
  - —Internetworking
  - —Internet Applications
  - —Network Security

The book includes a number of pedagogic features, including the use of animations and numerous figures and tables to clarify the discussions. Each chapter includes a list of key words, review questions, homework problems, and suggestions for further reading. The book also includes an extensive online glossary, a list of frequently used acronyms, and a reference list. In addition, a test bank is available to instructors.

The chapters and parts of the book are sufficiently modular to provide a great deal of flexibility in the design of courses. See Chapter 0 for a number of detailed suggestions for both top-down and bottom-up course strategies.

INSTRUCTOR SUPPORT MATERIALS

The major goal of this text is to make it as effective a teaching tool for this exciting and fast-moving subject as possible. This goal is reflected both in the structure of the book and in the supporting material. The text is accompanied by the following supplementary material to aid the instructor:

- **Solutions manual:** Solutions to all end-of-chapter Review Questions and Problems.
- **Projects manual:** Suggested project assignments for all of the project categories in the next section.
- **PowerPoint slides:** A set of slides covering all chapters, suitable for use in lecturing.
- **PDF files:** Reproductions of all figures and tables from the book.
- **Test bank:** A chapter-by-chapter set of questions with a separate file of answers.
- **Sample syllabuses:** The text contains more material than can be conveniently covered in one semester. Accordingly, instructors are provided with several sample syllabuses that guide the use of the text within limited time. These samples are based on real-world experience by professors with the ninth edition.

All of these support materials are available at the **Instructor Resource Center (IRC)** for this textbook, which can be reached through the publisher's Web site www.pearsoninternationaleditions.com/stallings or by clicking on the link labeled *Pearson Resources for Instructors* at this book's Companion Web site at WilliamStallings.com/DataComm. To gain access to the IRC, please contact your local Pearson sales representative.

The **Companion Web site**, at WilliamStallings.com/DataComm (click on *Instructor Resources* link), includes the following:

- Links to Web sites for other courses being taught using this book.
- Sign-up information for an Internet mailing list for instructors using this book to exchange information, suggestions, and questions with each other and with the author.

PROJECTS AND OTHER STUDENT EXERCISES

For many instructors, an important component of a data communications or networking course is a project or set of projects by which the student gets hands-on experience to reinforce concepts from the text. This book provides an unparalleled degree of support for including a projects component in the course. The IRC not only provides guidance on how to assign and structure the projects but also includes a set of User's Manuals for various project types plus specific assignments, all written especially for this book. Instructors can assign work in the following areas:

- **Animation assignments:** Described in the following section.
- **Practical exercises:** Using network commands, the student gains experience in network connectivity.
- **Sockets programming projects:** Described subsequently in this Preface.
- **Wireshark projects:** Wireshark is a protocol analyzer that enables students to study the behavior of protocols. A video tutorial is provided to get students started, in addition to a set of Wireshark assignments.
- **Simulation projects:** The student can use the simulation package *cnet* to analyze network behavior. The IRC includes a number of student assignments.
- **Performance modeling projects:** Two performance modeling techniques are introduced: a *tools* package and OPNET. The IRC includes a number of student assignments.
- **Research projects:** The IRC includes a list of suggested research projects that would involve Web and literature searches.
- **Reading/report assignments:** The IRC includes a list of papers that can be assigned for reading and writing a report, plus suggested assignment wording.
- **Writing assignments:** The IRC includes a list of writing assignments to facilitate learning the material.
- **Discussion topics:** These topics can be used in a classroom, chat room, or message board environment to explore certain areas in greater depth and to foster student collaboration.

This diverse set of projects and other student exercises enables the instructor to use the book as one component in a rich and varied learning experience and to tailor a course plan to meet the specific needs of the instructor and students. See Appendix B for details.

ANIMATIONS

Animations provide a powerful tool for understanding the complex mechanisms discussed in this book, including forward error correction, signal encoding, and protocols. Over 150 Web-based animations are used to illustrate many of the data communications and protocol concepts in this book. These animations are available online at the Premium Web site. For those chapters for which animations are available, this icon appears at the beginning of the chapter:  .

Twelve of the animations have been designed to allow for two types of assignments. First, the student can be given a specific set of steps to invoke and watch the animation, and then be asked to analyze and comment on the results. Second, the student can be given a specific end point and is required to devise a sequence of steps that achieve the desired result. The IRC includes a set of assignments for each of these animations, plus suggested solutions so that instructors can assess the student's work.

SOCKETS PROGRAMMING

Sockets are the fundamental element behind any kind of network communication using the TCP/IP protocol suite. Sockets programming is a relatively straightforward topic that can result in very satisfying and effective hands-on projects for students. This book provides considerable support to enable students to learn and use Sockets programming to enhance their understanding of networking, including:

1. Chapter 2 provides a basic introduction to Sockets programming and includes a detailed analysis of a TCP server and a TCP client program.
2. Chapter 2 also includes some end-of-chapter programming assignments using Sockets. Sample solutions are available at the IRC for this book.
3. Additional Sockets programming assignments, plus sample solutions, are available for instructors at the IRC. These include a number of moderate-size assignments and a more substantial project that, step by step, implements a simplified instant messaging client and server.
4. A different, additional set of Sockets assignments, plus sample solutions, are included in the supplemental homework problems available to students at the Premium Web site.

Taken together, these resources provide students with a solid understanding of Sockets programming and experience in developing networking applications.

ONLINE DOCUMENTS FOR STUDENTS

For this new edition, a substantial amount of original supporting material for students has been made available online, at two Web locations. The **Companion Web site**, at WilliamStallings.com/DataComm (click on *Student Resources* link), includes a list of relevant links organized by chapter and an errata sheet for the book.

Purchasing this textbook new also grants the reader six months of access to the **Premium Content site**, which includes the following materials:

- **Online chapters:** To limit the size and cost of the book, two chapters of the book, covering security, are provided in PDF format. The chapters are listed in this book's table of contents.
- **Online appendices:** There are numerous interesting topics that support material found in the text but whose inclusion is not warranted in the printed text.

A total of 18 online appendices cover these topics for the interested student. The appendices are listed in this book's table of contents.

- **Homework problems and solutions:** To aid the student in understanding the material, a separate set of homework problems with solutions is available.

To access the Premium Content site, click on the *Premium Content* link at the Companion Web site or at www.pearsoninternationaleditions.com/stallings and enter the student access code found on the card in the front of the book.

ACKNOWLEDGMENTS

Through its multiple editions this book has benefited from review by hundreds of instructors and professionals, who gave generously of their time and expertise. Here I acknowledge those whose help contributed to this latest edition.

The following instructors reviewed all or a large part of the manuscript: Tibor Gyires (Illinois State University), Hossein Hosseini (University of Wisconsin-Milwaukee), Naeem Shareef (Ohio State University), Adrian Lauf (University of Louisville), and Michael Fang (University of Florida).

Thanks also to the many people who provided detailed technical reviews of a single chapter: Naji A. Albakay, Prof. (Dr). C. Annamalai, Rakesh Kumar Bachchan, Alan Cantrell, Colin Conrad, Vineet Chadha, George Chetcuti, Rajiv Dasmohapatra, Ajinkya Deshpande, Michel Garcia, Thomas Johnson, Adri Jovin, Joseph Kellegher, Robert Knox, Bo Lin, Yadi Ma, Luis Arturo Frigolet Mayo, Sushil Menon, Hien Nguyen, Kevin Sanchez-Cherry, Mahesh S. Sankpal, Gaurav Santhalia, Stephanie Sullivan, Doug Tiedt, Thriveni Venkatesh, and Pete Zeno.

Thanks also to the following contributors. Yadi Ma contributed homework problems on Sockets programming. Yunzhao Li developed some of the animation applets. Larry Tan of the University of Stirling in Scotland developed the animation assignments. Michael Harris of Indiana University initially developed the Wireshark exercises and user's guide. Dave Bremer, a principal lecturer at Otago Polytechnic in New Zealand, updated the material for the most recent Wireshark release; he also developed an online video tutorial for using Wireshark. Kim McLaughlin produced the PPT lecture slides.

Finally, I thank the many people responsible for the publication of this book, all of whom did their usual excellent job. This includes the staff at Pearson, particularly my editor Tracy Johnson, her assistant Jenah Blitz-Stoehr, program manager Carole Snyder, and permissions supervisor Bob Engelhardt. I also thank Shiny Rajesh and the production staff at Integra for another excellent and rapid job. Thanks also to the marketing and sales staffs at Pearson, without whose efforts this book would not be in front of you.

The publishers wish to thank Somitra Kumar Sanadhya, of the Indraprastha Institute of Information Technology, Delhi, for reviewing the content of the International Edition.

About the Author

Dr. William Stallings has authored 17 titles, and counting revised editions, over 40 books on computer security, computer networking, and computer architecture. His writings have appeared in numerous publications, including the *Proceedings of the IEEE, ACM Computing Reviews* and *Cryptologia*.

He has 12 times received the award for the best Computer Science textbook of the year from the Text and Academic Authors Association.

In over 30 years in the field, he has been a technical contributor, technical manager, and an executive with several high-technology firms. He has designed and implemented both TCP/IP-based and OSI-based protocol suites on a variety of computers and operating systems, ranging from microcomputers to mainframes. As a consultant, he has advised government agencies, computer and software vendors, and major users on the design, selection, and use of networking software and products.

He created and maintains the *Computer Science Student Resource Site* at ComputerScienceStudent.com. This site provides documents and links on a variety of subjects of general interest to computer science students (and professionals). He is a member of the editorial board of *Cryptologia,* a scholarly journal devoted to all aspects of cryptology.

Dr. Stallings holds a PhD from MIT in Computer Science and a BS from Notre Dame in electrical engineering.

CHAPTER 0

Guide for Readers and Instructors

This book, with its accompanying Web support, covers a lot of material. Here, we give the reader some basic background information.

0.1 OUTLINE OF THE BOOK

The book is organized into two units. Unit One provides a survey of the fundamentals of data communications, networks, and Internet protocols. Unit Two covers more advanced or difficult topics in data communications and networks, and provides a more comprehensive discussion of Internet protocols and operation.

Unit One is organized into five parts:

Part One. Overview: Provides an introduction to the range of topics covered in the book. This part includes a general overview of data communications and networking, and a discussion of protocols and the TCP/IP protocol suite.

Part Two. Data Communications: Presents material concerned primarily with the exchange of data between two directly connected devices. Within this restricted scope, the key aspects of transmission, transmission media, error detection, link control, and multiplexing are examined.

Part Three. Wide Area Networks: Examines the technologies and protocols that have been developed to support voice, data, and multimedia communications over long-distance networks. The traditional technologies of packet switching and circuit switching, as well as the more contemporary ATM and cellular networks, are examined.

Part Four. Local Area Networks: Explores the technologies and architectures that have been developed for networking over shorter distances. The transmission media, topologies, and medium access control protocols that are the key ingredients of a LAN design are explored. This is followed by a detailed discussion of Ethernet and Wi-Fi networks.

Part Five. Internet and Transport Protocols: Discusses protocols at the Internet and Transport layers.

Unit Two consists of three parts:

Part Six. Data Communications and Wireless Networks: Treats important topics in these areas not covered in Unit One.

Part Seven. Internetworking: Examines a range of protocols and standards related to the operation of the Internet, including routing, congestion control, and quality of service.

Part Eight. Internet Applications: Looks at a range of applications that operate over the Internet.

In addition, there is an online **Part Nine. Security:** It covers security threats and techniques for countering these threats. A number of online appendices cover additional topics relevant to the book.

0.2 A ROADMAP FOR READERS AND INSTRUCTORS

The text contains more material than can be conveniently covered in one semester. Accordingly, the Instructor Resource Center (IRC) for this book includes several sample syllabi that guide the use of the text within limited time (e.g., 16 weeks or 12 weeks). Each alternative syllabus suggests a selection of chapters and a weekly schedule. These samples are based on real-world experience by professors with the previous edition.

The organization of the book into two units is intended to divide the material, roughly, into introductory and fundamental topics (Unit One) and advanced topics (Unit Two). Thus, a one-semester course could be limited to all or most of the material in Unit One.

In this section, we provide some other suggestions for organizing the material for a course.

Course Emphasis

The material in this book is organized into four broad categories: data transmission and communication, communications networks, network protocols, and applications and security. The chapters and parts of the book are sufficiently modular to provide a great deal of flexibility in the design of courses. The following are suggestions for three different course designs:

- **Fundamentals of Data Communications:** Parts One (overview), Two (data communications), and Three (wired WANs and cellular networks).
- **Communications Networks:** If the student has a basic background in data communications, then this course could cover Parts One (overview), Three (WAN), and Four (LAN).
- **Computer Networks:** If the student has a basic background in data communications, then this course could cover Part One (overview), Chapters 6 and 7 (error detection and correction, and data link control), Part Five (internet and transport protocols), and part or all of Parts Seven (internetworking) and Eight (applications).

In addition, a more streamlined course that covers the entire book is possible by eliminating certain chapters that are not essential on a first reading. The sample syllabi document at the IRC provides guidance on chapter selection.

Bottom–Up versus Top–Down

The book is organized in a modular fashion. After reading Part One, the other parts can be read in a number of possible sequences. Table 0.1a shows the bottom–up approach provided by reading the book from front to back. With this approach, each part builds on the material in the previous part, so that it is always clear how

Table 0.1 Suggested Reading Orders

(a) A bottom–up approach	(b) A shorter bottom–up approach
Part One: Overview	Part One: Overview
Part Two: Data Communications	Part Two: Data Communications (Chapters 3, 6, 7, 8)
Part Three: Wide Area Networks	Part Three: Wide Area Networks
Part Four: Local Area Networks	Part Four: Local Area Networks
Part Five: Internet and Transport Layers	Part Five: Internet and Transport Layers
Part Seven: Internetworking	
Part Eight: Internet Applications	
(c) A top–down approach	**(d) A shorter top–down approach**
Part One: Overview	Part One: Overview
Chapter 14: The Internet Protocol	Chapter 14: The Internet Protocol
Part Eight: Internet Applications	Part Eight: Internet Applications
Chapter 15: Transport Protocols	Chapter 15: Transport Protocols
Part Seven: Internetworking	Part Seven: Internetworking (Chapters 19, 20, 21)
Part Three: Wide Area Networks	Part Three: Wide Area Networks
Part Four: Local Area Networks	Part Four: Local Area Networks (Chapter 11)
Part Two: Data Communications	

a given layer of functionality is supported from below. There is more material than can be comfortably covered in a single semester, but the book's organization makes it easy to eliminate some chapters and maintain the bottom–up sequence. Table 0.1b suggests one approach to a survey course.

Some readers, and some instructors, are more comfortable with a top–down approach. After the background material (Part One), the reader continues at the application level and works down through the protocol layers. This has the advantage of immediately focusing on the most visible part of the material, the applications, and then seeing, progressively, how each layer is supported by the next layer down. Table 0.1c is an example of a comprehensive treatment, and Table 0.1d is an example of a survey treatment.

0.3 INTERNET AND WEB RESOURCES

There are a number of resources available on the Internet and the Web that support this book and help readers keep up with developments in this field.

Web Sites for This Book

Three Web sites provide additional resources for students and instructors.

There is a **Companion Website** for this book at http://williamstallings.com/DataComm. For students, this Web site includes a list of relevant links, organized by chapter, and an errata list for the book. For instructors, this Web site provides

links to course pages by professors teaching from this book and provides a number of other useful documents and links.

There is also an access-controlled **Premium Content Website**, which provides a wealth of supporting material, including additional online chapters, additional online appendices, and a set of homework problems with solutions. See the card at the front of this book for access information.

Finally, additional material for instructors, including a solutions manual and a projects manual, is available at the **Instructor Resource Center (IRC)** for this book. See Preface for details and access information.

Computer Science Student Resource Site

I also maintain the **Computer Science Student Resource Site**, at ComputerScience Student.com. The purpose of this site is to provide documents, information, and links for computer science students and professionals. Links and documents are organized into seven categories:

- **Math:** Includes a basic math refresher, a queuing analysis primer, a number system primer, and links to numerous math sites.
- **How-to:** Advice and guidance for solving homework problems, writing technical reports, and preparing technical presentations.
- **Research resources:** Links to important collections of papers, technical reports, and bibliographies.
- **Other useful:** A variety of other useful documents and links.
- **Computer science careers:** Useful links and documents for those considering a career in computer science.
- **Writing help:** Help in becoming a clearer, more effective writer.
- **Miscellaneous topics and humor:** You have to take your mind off your work once in a while.

Other Web Sites

Numerous Web sites provide information related to the topics of this book. The Companion Website provides links to these sites, organized by chapter.

0.4 STANDARDS

Standards have come to play a dominant role in the information communications marketplace. Virtually all vendors of products and services are committed to supporting international standards. Throughout this book, we describe the most important standards in use or being developed for various aspects of data communications and networking. Various organizations have been involved in the development or promotion of these standards. The most important (in the current context) of these organizations are as follows:

- **Internet Society:** The Internet SOCiety (ISOC) is a professional membership society with worldwide organizational and individual membership. It provides

leadership in addressing issues that confront the future of the Internet and is the organization home for the groups responsible for Internet infrastructure standards, including the Internet Engineering Task Force (IETF) and the Internet Architecture Board (IAB). These organizations develop Internet standards and related specifications, all of which are published as Requests for Comments (RFCs).

- **IEEE 802:** The IEEE (Institute of Electrical and Electronics Engineers) 802 LAN/MAN Standards Committee develops local area network standards and metropolitan area network standards. The most widely used standards are for the Ethernet family, wireless LAN, bridging, and virtual bridged LANs. An individual working group provides the focus for each area.
- **ITU-T:** The International Telecommunication Union (ITU) is a United Nations agency in which governments and the private sector coordinate global telecom networks and services. The ITU Telecommunication Standardization Sector (ITU-T) is one of the three sectors of the ITU. ITU-T's mission is the production of standards covering all fields of telecommunications. ITU-T standards are referred to as Recommendations.
- **ISO:** The International Organization for Standardization (ISO)[1] is a worldwide federation of national standards bodies from more than 140 countries, one from each country. ISO is a nongovernmental organization that promotes the development of standardization and related activities with a view to facilitating the international exchange of goods and services, and to developing cooperation in the spheres of intellectual, scientific, technological, and economic activity. ISO's work results in international agreements that are published as International Standards.

A more detailed discussion of these organizations is contained in Appendix C.

[1]ISO is not an acronym (in which case it would be IOS), but a word, derived from the Greek, meaning *equal*.

UNIT ONE
FUNDAMENTALS

PART ONE: OVERVIEW

CHAPTER 1

DATA COMMUNICATIONS, DATA NETWORKS, AND THE INTERNET

LEARNING OBJECTIVES

After studying this chapter, you should be able to:

- Present an overview of data communications traffic volume trends.
- Understand the key elements of a data communications system.
- Summarize the types of data communications networks.
- Present an overview of the overall architecture of the Internet.

This book aims to provide a unified view of the broad field of data and computer communications. The organization of the book reflects an attempt to break this massive subject into comprehensible parts and to build, piece by piece, a survey of the state of the art. This introductory chapter begins with a general model of communications. Then a brief discussion introduces each of the Parts Two through Four and Six of this book. Chapter 2 provides an overview to Parts Five, Eight, and Nine.

1.1 DATA COMMUNICATIONS AND NETWORKING FOR TODAY'S ENTERPRISE

Effective and efficient data communication and networking facilities are vital to any enterprise. In this section, we first look at trends that are increasing the challenge for the business manager in planning and managing such facilities. Then we look specifically at the requirement for ever-greater transmission speeds and network capacity.

Trends

Three different forces have consistently driven the architecture and evolution of data communications and networking facilities: traffic growth, development of new services, and advances in technology.

Communication **traffic**, both local (within a building or business campus) and long distance, has been growing at a high and steady rate for decades. Network traffic is no longer limited to voice and data and increasingly includes image and video. Increasing business emphasis on web services, remote access, online transactions, and social networking means that this trend is likely to continue. Thus, business managers are constantly pressured to increase communication capacity in cost-effective ways.

As businesses rely more and more on information technology, the range of **services** that business users desire to consume is expanding. For example, mobile

broadband traffic growth is exploding as is the amount of data being pushed over mobile networks by business users' smart phones and tablets. In addition, over time, mobile users are increasingly demanding high-quality services to support their high-resolution camera phones, favorite video streams, and high-end audio. Similar demand growth is seen in landline access to the Internet and private networks. To keep up with mushrooming traffic generated by both consumers and business users, mobile service providers have to keep investing in high-capacity networking and transmission facilities. In turn, the growth in high-speed network offerings at competitive price points encourages the expansion of mobile applications and services. Thus, growth in services and in traffic capacity go hand in hand. As an example, Figure 1.1 [IEEE12] shows the mix of traffic and the growth trend for cable Internet subscribers.

Finally, trends in **technology** enable the provision of increasing traffic capacity and the support of a wide range of services. Four technology trends are particularly notable:

1. The trend toward faster and cheaper, in both computing and communications, continues. In terms of computing, this means more powerful computers and clusters of computers capable of supporting more demanding applications, such as multimedia applications. In terms of communications, the increasing use of optical fiber and high-speed wireless has brought transmission prices down and greatly increased capacity. For example, for long-distance telecommunication and data network links, dense wavelength division multiplexing (DWDM)

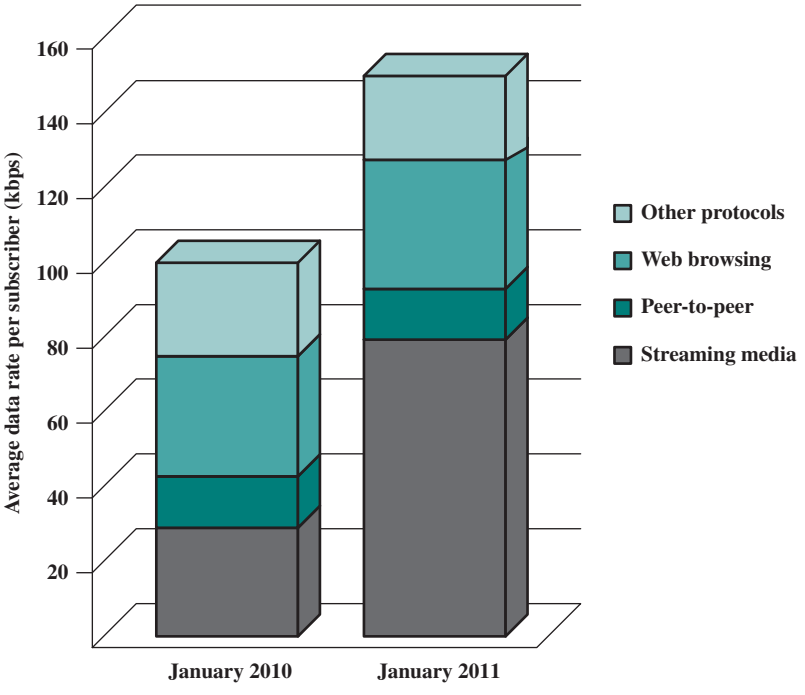


Figure 1.1 Average Downstream Traffic per Internet Subscriber

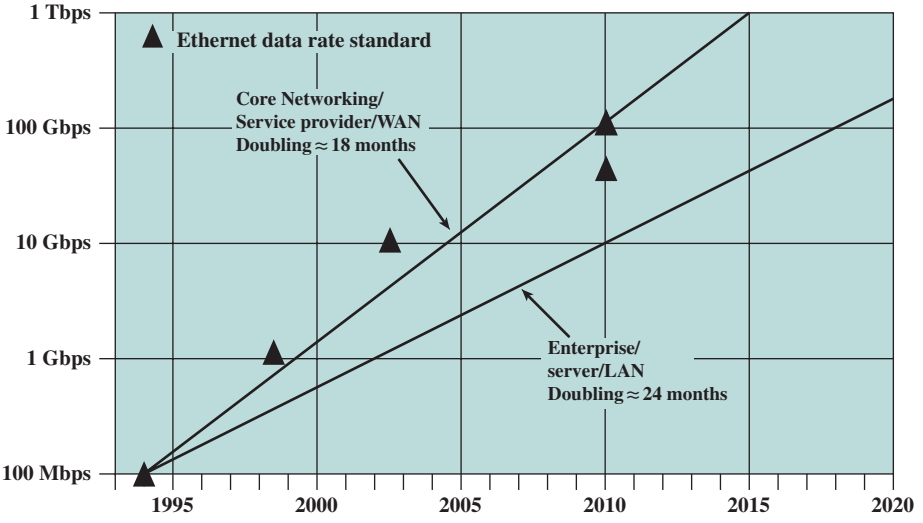


Figure 1.2 Past and Projected Growth in Ethernet Data Rate Demand Compared to Existing Ethernet Data Rates

enables communication traffic to be carried by fiber optic cables at rates of multiple terabits per second. For local area networks (LANs), many enterprises now have 40-Gbps Ethernet or 100-Gbps Ethernet backbone networks.[1] Figure 1.2 [IEEE12] indicates the Ethernet demand trend. As shown, usage statistics indicate that Internet backbone data rate demand in the network core doubles approximately every 18 months, while demand in enterprise/LAN applications doubles approximately every 24 months.

2. Today's networks are more "intelligent" than ever. Two areas of intelligence are noteworthy. First, today's networks can offer differing levels of quality of service (QoS), which include specifications for maximum delay, minimum throughput, and so on to ensure high-quality support for applications and services. Second, today's networks provide a variety of customizable services in the areas of network management and security.
3. The Internet, the Web, and associated applications have emerged as dominant features for both business and personal network landscapes. The migration to "everything over IP" continues and has created many opportunities and challenges for information and communications technology (ICT) managers. In addition to exploiting the Internet and the Web to reach customers, suppliers, and partners enterprises have formed intranets and extranets[2] to isolate proprietary information to keep it free from unwanted access.

[1]An explanation of numerical prefixes, such as *tera* and *giga*, is provided in the document *Prefix.pdf*, available at box.com/dcc10e.

[2]Briefly, an intranet uses Internet and Web technology in an isolated facility internal to an enterprise; an extranet extends a company's intranet out onto the Internet to allow selected customers, suppliers, and mobile workers to access the company's private data and applications.

4. Mobility is newest frontier for ICT managers, and popular consumer devices such as the iPhone, Droid, and iPad have become drivers of the evolution of business networks and their use. While there has been a trend toward mobility for decades, the mobility explosion has occurred and has liberated workers from the confines of the physical enterprise. Enterprise applications traditionally supported on terminals and office desktop computers are now routinely delivered on mobile devices. Cloud computing is being embraced by all major business software vendors including SAP, Oracle, and Microsoft, and this ensures that further mobility innovations will be forthcoming. Industry experts predict that mobile devices will become the dominant business computing platform by 2015 and that enhanced ability to use enterprise information resources and services anywhere-anytime will be a dominant trend for the remainder of the decade.

Data Transmission and Network Capacity Requirements

Momentous changes in the way organizations do business and process information have been driven by changes in networking technology and at the same time have driven those changes. It is hard to separate chicken and egg in this field. Similarly, the use of the Internet by both businesses and individuals reflects this cyclic dependency: The availability of new image-based services on the Internet (i.e., the Web) has resulted in an increase in the total number of users and the traffic volume generated by each user. This, in turn, has resulted in a need to increase the speed and efficiency of the Internet. On the other hand, it is only such increased speed that makes the use of Web-based applications palatable to the end user.

In this section, we survey some of the end-user factors that fit into this equation. We begin with the need for high-speed LANs in the business environment, because this need has appeared first and has forced the pace of networking development. Then we look at business WAN requirements. Finally, we offer a few words about the effect of changes in commercial electronics on network requirements.

The Emergence of High-Speed LANs Personal computers and microcomputer workstations began to achieve widespread acceptance in business computing in the early 1980s and have now achieved virtually the status of the telephone: an essential tool for office workers. Until relatively recently, office LANs provided basic connectivity services—connecting personal computers and terminals to mainframes and midrange systems that ran corporate applications, and providing workgroup connectivity at the departmental or divisional level. In both cases, traffic patterns were relatively light, with an emphasis on file transfer and electronic mail. The LANs that were available for this type of workload, primarily Ethernet and token ring, are well suited to this environment.

In the last 20 years, two significant trends altered the role of the personal computer and therefore the requirements on the LAN:

1. The speed and computing power of personal computers continued to enjoy explosive growth. These more powerful platforms support graphics-intensive applications and ever more elaborate graphical user interfaces to the operating system.

2. MIS (management information systems) organizations have recognized the LAN as a viable and essential computing platform, resulting in the focus on network computing. This trend began with client/server computing, which has become a dominant architecture in the business environment and the more recent Web-focused intranet trend. Both of these approaches involve the frequent transfer of potentially large volumes of data in a transaction-oriented environment.

The effect of these trends has been to increase the volume of data to be handled over LANs and, because applications are more interactive, to reduce the acceptable delay on data transfers. The earlier generation of 10-Mbps Ethernets and 16-Mbps token rings was simply not up to the job of supporting these requirements.

The following are examples of requirements that call for higher-speed LANs:

- **Centralized server farms:** In many applications, there is a need for user, or client, systems to be able to draw huge amounts of data from multiple centralized servers, called server farms. An example is a color publishing operation, in which servers typically contain tens of gigabytes of image data that must be downloaded to imaging workstations. As the performance of the servers themselves has increased, the bottleneck has shifted to the network.
- **Power workgroups:** These groups typically consist of a small number of cooperating users who need to draw massive data files across the network. Examples are a software development group that runs tests on a new software version, or a computer-aided design (CAD) company that regularly runs simulations of new designs. In such cases, large amounts of data are distributed to several workstations, processed, and updated at very high speed for multiple iterations.
- **High-speed local backbone:** As processing demand grows, LANs proliferate at a site, and high-speed interconnection is necessary.

Corporate Wide Area Networking Needs As recently as the early 1990s, there was an emphasis in many organizations on a centralized data processing model. In a typical environment, there might be significant computing facilities at a few regional offices, consisting of mainframes or well-equipped midrange systems. These centralized facilities could handle most corporate applications, including basic finance, accounting, and personnel programs, as well as many of the business-specific applications. Smaller, outlying offices (e.g., a bank branch) could be equipped with terminals or basic personal computers linked to one of the regional centers in a transaction-oriented environment.

This model began to change in the early 1990s, and the change accelerated since then. Many organizations have dispersed their employees into multiple smaller offices. There is a growing use of telecommuting. Most significant, the nature of the application structure has changed. First client/server computing and, more recently, intranet computing have fundamentally restructured the organizational data processing environment. There is now much more reliance on personal computers, workstations, and servers and much less use of centralized mainframe and midrange systems. Furthermore, the virtually universal deployment of graphical user interfaces to the desktop enables the end user to exploit graphic applications,

multimedia, and other data-intensive applications. In addition, most organizations require access to the Internet. When a few clicks of the mouse can trigger huge volumes of data, traffic patterns have become more unpredictable while the average load has risen.

All of these trends mean that more data must be transported off premises and into the wide area. It has long been accepted that in the typical business environment, about 80% of the traffic remains local and about 20% traverses wide area links. But this rule no longer applies to most companies, with a greater percentage of the traffic going into the WAN environment. This traffic flow shift places a greater burden on LAN backbones and, of course, on the WAN facilities used by a corporation. Thus, just as in the local area, changes in corporate data traffic patterns are driving the creation of high-speed WANs.

Digital Electronics The rapid conversion of consumer electronics to digital technology is having an impact on both the Internet and corporate intranets. As these new gadgets come into view and proliferate, they dramatically increase the amount of image and video traffic carried by networks.

Two noteworthy examples of this trend are digital versatile disks (DVDs) and digital still cameras. With the capacious DVD, the electronics industry at last found an acceptable replacement for the analog video home system (VHS) tapes. The DVD has replaced the videotape used in videocassette recorders (VCRs) and the CD-ROM in personal computers and servers. The DVD takes video into the digital age. It delivers movies with picture quality that outshines laser disks, and it can be randomly accessed like audio CDs, which DVD machines can also play. Vast volumes of data can be crammed onto the disk. With DVD's huge storage capacity and vivid quality, PC games have become more realistic and educational software incorporates more video. Following in the wake of these developments is a new crest of traffic over the Internet and corporate intranets, as this material is incorporated into Web sites.

A related product development is the digital camcorder. This product has made it easier for individuals and companies to make digital video files to be placed on corporate and Internet Web sites, again adding to the traffic burden.

Convergence

Convergence refers to the merger of previously distinct telephony and information technologies and markets. We can think of this convergence in terms of a three-layer model of enterprise communications:

- **Applications:** These are seen by the end users of a business. Convergence integrates communications applications, such as voice calling (telephone), voice mail, e-mail, and instant messaging, with business applications, such as workgroup collaboration, customer relationship management, and other back-office functions. With convergence, applications provide features that incorporate voice, data, and video in a seamless, organized, and value-added manner. One example is multimedia messaging, which enables a user to employ a single interface to access messages from a variety of sources (e.g., office voice mail, office e-mail, beeper, and fax).

- **Enterprise services:** At this level, the manager deals with the information network in terms of the services it provides to support applications. The network manager needs design, maintenance, and support services related to the deployment of convergence-based facilities. Also at this level, network managers deal with the enterprise network as a function-providing system. Such management services may include setting up authentication schemes; capacity management for various users, groups, and applications; and QoS provision.
- **Infrastructure:** The network and communications infrastructure consists of the communication links, LANs, WANs, and Internet connections available to the enterprise. Increasingly, enterprise network infrastructure also includes private and/or public cloud connections to data centers which host high-volume data storage and Web services. A key aspect of convergence at this level is the ability to carry voice, image, and video over networks that were originally designed to carry data traffic. Infrastructure convergence has also occurred for networks that were designed for voice traffic. For example, video, image, text, and data are routinely delivered to smart phone users over cell phone networks.

In simple terms, convergence involves moving voice into a data infrastructure, integrating all the voice and data networks inside a user organization into a single data network infrastructure, and then extending that into the wireless arena. The foundation of this convergence is packet-based transmission using the Internet Protocol (IP). Convergence increases the function and scope of both the infrastructure and the application base.

1.2 A COMMUNICATIONS MODEL

This section introduces a simple model of communications, illustrated by the block diagram in Figure 1.3a.

The fundamental purpose of a communications system is the exchange of data between two parties. Figure 1.3b presents one particular example, which is communication between a workstation and a server over a public telephone network. Another example is the exchange of voice signals between two telephones over the same network. The following are key elements of the model:

- **Source:** This device generates the data to be transmitted; examples are telephones and personal computers.
- **Transmitter:** Usually, the data generated by a source system are not transmitted directly in the form in which they were generated. Rather, a transmitter transforms and encodes the information in such a way as to produce electromagnetic signals that can be transmitted across some sort of transmission system. For example, a modem takes a digital bit stream from an attached device such as a personal computer and transforms that bit stream into an analog signal that can be handled by the telephone network.
- **Transmission system:** This can be a single transmission line or a complex network connecting source and destination.

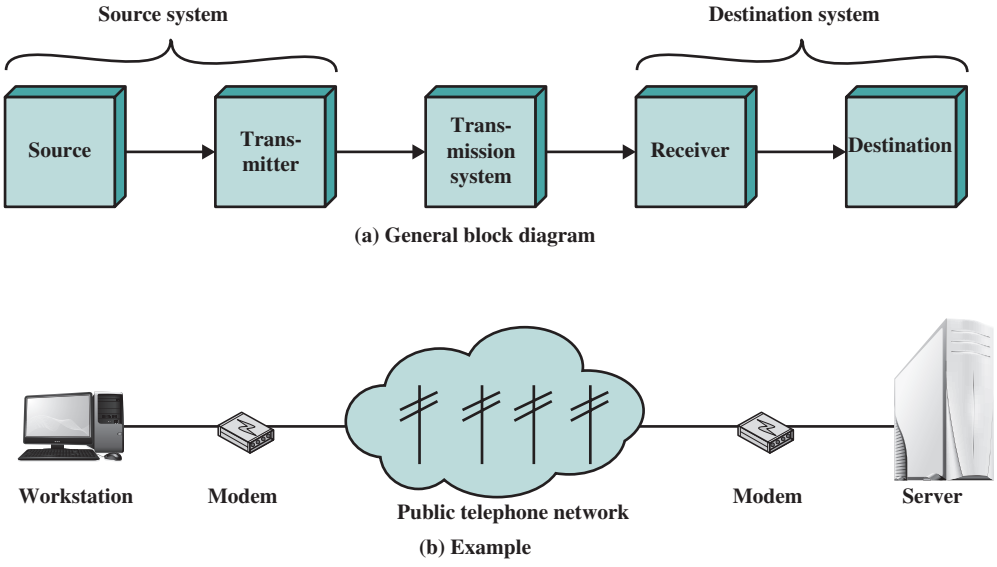


Figure 1.3 Simplified Communications Model

- **Receiver:** The receiver accepts the signal from the transmission system and converts it into a form that can be handled by the destination device. For example, a modem will accept an analog signal coming from a network or transmission line and convert it into a digital bit stream.
- **Destination:** Takes the incoming data from the receiver.

This simple narrative conceals a wealth of technical complexity. To get some idea of the scope of this complexity, Table 1.1 lists some of the key tasks that must be performed in a data communications system. The list is somewhat arbitrary: Elements could be added; items on the list could be merged; and some items represent several tasks that are performed at different "levels" of the system. However, the list as it stands is suggestive of the scope of this book.

The first item, **transmission system utilization**, refers to the need to make efficient use of transmission facilities that are typically shared among a number of communicating devices. Various techniques (referred to as multiplexing) are used to allocate the total capacity of a transmission medium among a number of users.

Table 1.1 Communications Tasks

Transmission system utilization	Addressing
Interfacing	Routing
Signal generation	Recovery
Synchronization	Message formatting
Exchange management	Security
Error detection and correction	Network management
Flow control	

Congestion control techniques may be required to assure that the system is not overwhelmed by excessive demand for transmission services.

To communicate, a device must **interface** with the transmission system. All forms of communication discussed in this book depend on the use of electromagnetic signals propagated over a transmission medium. Thus, once an interface is established, **signal generation** is required for communication. The properties of the signal, such as form and intensity, must be such that the signal is (1) capable of being propagated through the transmission system, and (2) interpretable as data at the receiver.

Not only must the signals be generated to conform to the requirements of the transmission system and receiver, but also there must be some form of **synchronization** between transmitter and receiver. The receiver must be able to determine when a signal begins to arrive and when it ends. It must also know the duration of each signal element.

Beyond the basic matter of deciding on the nature and timing of signals, there is a variety of requirements for communication between two parties that might be collected under the term **exchange management**. If data are to be exchanged in both directions over a period of time, the two parties must cooperate. For example, for two parties to engage in a telephone conversation, one party must dial the number of the other, causing signals to be generated that result in the ringing of the called phone. The called party completes a connection by lifting the receiver. For data processing devices, more will be needed than simply establishing a connection; certain conventions must be decided on. These conventions might include whether both devices may transmit simultaneously or must take turns, the amount of data to be sent at one time, the format of the data, and what to do if certain contingencies such as an error arise.

The next two items might have been included under exchange management, but they seem important enough to list separately. In all communications systems, there is a potential for error; transmitted signals are distorted to some extent before reaching their destination. **Error detection and correction** are required in circumstances where errors cannot be tolerated. This is usually the case with data processing systems. For example, in transferring a file from one computer to another, it is simply not acceptable for the contents of the file to be accidentally altered. **Flow control** is required to assure that the source does not overwhelm the destination by sending data faster than they can be processed and absorbed.

Next are the related but distinct concepts of **addressing** and **routing**. When more than two devices share a transmission facility, a source system must indicate the identity of the intended destination. The transmission system must assure that the destination system, and only that system, receives the data. Further, the transmission system may itself be a network through which various paths may be taken. A specific route through this network must be chosen.

Recovery is a concept distinct from that of error correction. Recovery techniques are needed in situations in which an information exchange, such as a database transaction or file transfer, is interrupted due to a fault somewhere in the system. The objective is either to be able to resume activity at the point of interruption or at least to restore the state of the systems involved to the condition prior to the beginning of the exchange.

Message formatting has to do with an agreement between two parties as to the form of the data to be exchanged or transmitted, such as the binary code for characters.

Frequently, it is important to provide some measure of **security** in a data communications system. The sender of data may wish to be assured that only the intended receiver actually receives the data. And the receiver of data may wish to be assured that the received data have not been altered in transit and that the data actually come from the purported sender.

Finally, a data communications facility is a complex system that cannot create or run itself. **Network management** capabilities are needed to configure the system, monitor its status, react to failures and overloads, and plan intelligently for future growth.

Thus, we have gone from the simple idea of data communication between source and destination to a rather formidable list of data communications tasks. In this book, we elaborate this list of tasks to describe and encompass the entire set of activities that can be classified under data and computer communications.

1.3 DATA COMMUNICATIONS

A Data Communications Model

To get some flavor for the focus of Part Two, Figure 1.4 provides a new perspective on the communications model of Figure 1.3a. We trace the details of this figure using electronic mail as an example.

Suppose that the input device and transmitter are components of a personal computer. The user of the PC wishes to send a message m to another user. The user activates the electronic mail package on the PC and enters the message via the keyboard (input device). The character string is briefly buffered in main memory. We can view it as a sequence of bits (g) in memory. The personal computer is connected to some transmission medium, such as a local area network, a digital subscriber line, or a wireless connection, by an I/O device (transmitter), such as a local network transceiver or a DSL modem. The input data are transferred to the transmitter as a

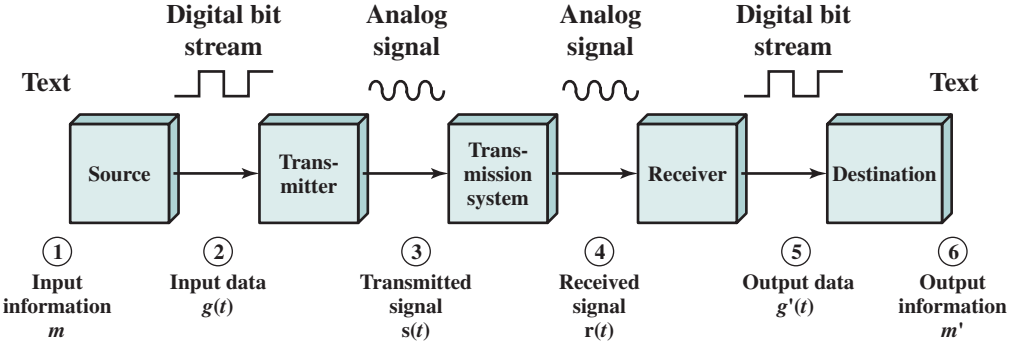


Figure 1.4 Simplified Data Communications Model

sequence of voltage shifts [$g(t)$] representing bits on some communications bus or cable. The transmitter is connected directly to the medium and converts the incoming stream [$g(t)$] into a signal [$s(t)$] suitable for transmission; specific alternatives will be described in Chapter 5.

The transmitted signal $s(t)$ presented to the medium is subject to a number of impairments, discussed in Chapter 3, before it reaches the receiver. Thus, the received signal $r(t)$ may differ from $s(t)$. The receiver will attempt to estimate the original $s(t)$, based on $r(t)$ and its knowledge of the medium, producing a sequence of bits $g'(t)$. These bits are sent to the output personal computer, where they are briefly buffered in memory as a block of bits (g'). In many cases, the destination system will attempt to determine if an error has occurred and, if so, cooperate with the source system to eventually obtain a complete, error-free block of data. These data are then presented to the user via an output device, such as a printer or screen. The message (m') as viewed by the user will usually be an exact copy of the original message (m).

Now consider a telephone conversation. In this case the input to the telephone is a message (m) in the form of sound waves. The sound waves are converted by the telephone into electrical signals of the same frequency. These signals are transmitted without modification over the telephone line. Hence the input signal $g(t)$ and the transmitted signal $s(t)$ are identical. The signal $s(t)$ will suffer some distortion over the medium, so that $r(t)$ will not be identical to $s(t)$. Nevertheless, the signal $r(t)$ is converted back into a sound wave with no attempt at correction or improvement of signal quality. Thus, m' is not an exact replica of m. However, the received sound message is generally comprehensible to the listener.

The discussion so far does not touch on other key aspects of data communications, including data link control techniques for controlling the flow of data and detecting and correcting errors, and multiplexing techniques for transmission efficiency.

The Transmission of Information

The basic building block of any enterprise network infrastructure is the transmission line. Much of the technical detail of how information is encoded and transmitted across a line is of no real interest to the business manager. The manager is concerned with whether the particular facility provides the required capacity, with acceptable reliability, at minimum cost. However, there are certain aspects of transmission technology that a manager must understand to ask the right questions and make informed decisions.

One of the basic choices facing a business user is the transmission medium. For use within the business premises, this choice is generally completely up to the business. For long-distance communications, the choice is generally but not always made by the long-distance carrier. In either case, changes in technology are rapidly changing the mix of media used. Of particular note are *fiber optic* transmission and *wireless* transmission (e.g., satellite and cellular communications). These two media are now driving the evolution of data communications transmission.

The ever-increasing availability of fiber optic communication circuits is making channel capacity a virtually free resource. Since the early 1980s, the growth of the market for optical fiber transmission systems is without precedent. During

the past 10 years, the cost of fiber optic transmission has dropped by more than an order of magnitude, and the capacity of such systems has grown at almost as rapid a rate. Almost all of the long-distance telephone communications trunks within the United States and the highest speed links on the Internet consist of fiber optic cable. Because of its high capacity and its security characteristics (fiber is difficult to tap), it is becoming increasingly used within office buildings and local area networks to carry the growing load of business information. The spreading use of fiber optic cable is also spurring advancements in communication switching technologies and network management architectures.

The increasing use of the second medium, wireless transmission, is a result of the trend toward universal personal telecommunications and universal access to communications. The first concept refers to the ability of a person to a single account to use any communication system anytime-anywhere, ideally globally. The second refers to the ability to use one's preferred computing device in a wide variety of environments to connect to information services (e.g., to have a laptop, smartphone, or tablet that will work equally well in the office, on the street, and on an airplane, bus, or train). Today, both concepts are subsumed under the business push to support mobility. Wireless LANs have become common components of enterprise networks as well as small office/home office networks, and smartphones and tablets with wireless capabilities are rapidly becoming mainstream business user communications devices. Mobility has the potential to unleash higher performance at all business levels: personal, workgroup, and enterprise-wide. This provides compelling rationale for further business investment in wireless technologies.

Despite the growth in the capacity and the drop in cost of transmission facilities, transmission services remain the most costly component of a communications budget for most businesses. Thus, the manager needs to be aware of techniques that increase the efficiency of the use of these facilities. The two major approaches to greater efficiency are multiplexing and compression. *Multiplexing* refers to the ability of a number of devices to share a transmission facility. If each device needs the facility only a fraction of the time, then a sharing arrangement allows the cost of the facility to be spread over many users. *Compression*, as the name indicates, involves squeezing the data down so that a lower-capacity, cheaper transmission facility can be used to meet a given demand. These two techniques show up separately and in combination in a number of types of communications equipment. The manager needs to understand these technologies to assess the appropriateness and cost-effectiveness of the various products on the market.

Transmission and Transmission Media Information can be communicated by converting it into an electromagnetic signal and transmitting that signal over some medium, such as a twisted-pair telephone line. The most commonly used transmission media are twisted-pair lines, coaxial cable, optical fiber cable, and terrestrial and satellite microwave. The data rates that can be achieved and the rate at which errors can occur depend on the nature of the signal and the type of medium. Chapters 3 and 4 examine the significant properties of electromagnetic signals and compare the various transmission media in terms of cost, performance, and applications.

Communication Techniques The transmission of information across a transmission medium involves more than simply inserting a signal on the medium. The

technique used to encode the information into an electromagnetic signal must be determined. There are various ways in which the encoding can be done, and the choice affects performance and reliability. Furthermore, the successful transmission of information involves a high degree of cooperation between the various components. The interface between a device and the transmission medium must be agreed on. Some means of controlling the flow of information and recovering from its loss or corruption must be used. These latter functions are performed by a data link control protocol. All these issues are examined in Chapters 5 through 7.

Transmission Efficiency A major cost in any computer/communications facility is transmission cost. Because of this, it is important to maximize the amount of information that can be carried over a given resource or, alternatively, to minimize the transmission capacity needed to satisfy a given information communications requirement. Two ways of achieving this objective are multiplexing and compression. The two techniques can be used separately or in combination. Chapter 8 examines the three most common multiplexing techniques: frequency division, synchronous time division, and statistical time division, as well as the important compression techniques.

1.4 NETWORKS

Globally, the number of Internet users is forecast to increase from approximately 2 billion in 2011 to 3 billion users in 2016. This figure is in fact misleading, as a single end user may have multiple types of devices. The estimate for 2016 is that there will be over 20 billion fixed and mobile networked devices and machine-to-machine connections, up from about 7 billion devices in 2011. The increase in the number of user devices, especially broadband devices, affects traffic volume in a number of ways. It enables a user to be continuously consuming network capacity, as well as to be consuming capacity on multiple devices simultaneously. Also, different broadband devices enable different applications, which may have greater traffic generation capability. The result is that the total annual traffic generated over the Internet and other IP-based networks is forecast to rise from 372 exabytes (372×2^{60} bytes) to 1.3 zettabytes (1.3×2^{70} bytes) in 2016 [CISC12a]. This traffic demand imposes stiff performance requirements on communications protocols, which is previewed in Chapter 2, and on communications and computer networks.

One type of network that has become commonplace is the local area network. Indeed, LANs are to be found in virtually all medium- and large-size office buildings. LANs, especially Wi-Fi LANs, are also increasingly used for small office and home networks. As the number and power of computing devices have grown, so have the number and capacity of LANs found in business networks. The development of internationally recognized standards for LANs has contributed to their proliferation in enterprises. Although Ethernet has emerged as the dominant LAN architecture, business managers still have choices to make about transmission rates (ranging from 100 Mbps to 100 Gbps) and the degree to which both wired and wireless LANs will be combined within an enterprise network. Interconnecting and managing a

diverse collection of local area networks and computing devices within today's business networks presents ongoing challenges for networking professionals.

A business need for a robust network to support voice, data, image, and video traffic is not confined to a single office building or LAN; today, it is an enterprise-wide communication requirement. Advances in LAN switches and other data communication technologies have led to greatly increased local area network transmission capacities and the concept of integration. *Integration* means that the communication equipment and networks can deal simultaneously with voice, data, image, and even video. Thus, a memo or report can be accompanied by voice commentary, presentation graphics, and perhaps even a short video introduction, demonstration, or summary. Image and video services that perform adequately within LANs often impose large demands on wide area network transmission and can be costly. Moreover, as LANs become ubiquitous and as their transmission rates increase, the need for enterprise networks to support interconnections among geographically dispersed areas has increased. This, in turn, has forced businesses increase wide area network transmission and switching capacity. Fortunately, the enormous and ever-increasing capacity of fiber optic and wireless transmission services provides ample resources to meet these business data communication needs. However, the development of switching systems that are capable of responding to the increasing capacities of transmission links and business communication traffic requirements is an ongoing challenge not yet conquered.

The opportunities for a business to use its enterprise network as an aggressive competitive tool and as a means of enhancing productivity and slashing costs are great. When business managers understand these technologies, they can deal effectively with data communication equipment vendors and service providers to enhance the company's competitive position.

In the remainder of this section, we provide a brief overview of various networks. Parts Three and Four cover these topics in depth.

Wide Area Networks

Wide area networks generally cover a large geographical area. They often require the crossing of public right-of-ways, and typically rely at least in part on circuits provided by one or more *common carriers*—companies that offer communication services to the general public. Typically, a WAN consists of a number of interconnected switching nodes. A trans-mission from any one device is routed through these internal nodes to the specified destination device. These nodes (including the boundary nodes) are not concerned with the content of the data; rather, their purpose is to provide a switching facility that will move the data from node to node until they reach their destination.

Traditionally, WANs have been implemented using one of two technologies: circuit switching and packet switching. Subsequently, frame relay and ATM networks assumed major roles. While ATM and, to some extent frame relay, are still widely used, their use is gradually being supplanted by services based on gigabit Ethernet and Internet Protocol technologies.

Circuit Switching In a circuit-switching network, a dedicated communications path is established between two stations through the nodes of the network. That

path is a connected sequence of physical links between nodes. On each link, a logical channel is dedicated to the connection. Data generated by the source station are transmitted along the dedicated path as rapidly as possible. At each node, incoming data are routed or switched to the appropriate outgoing channel without delay. The most common example of circuit switching is the telephone network.

Packet Switching In a packet-switching network, it is not necessary to dedicate transmission capacity along a path through the network. Rather, data are sent out in a sequence of small chunks, called packets. Each packet is passed through the network from node to node along some path leading from source to destination. At each node, the entire packet is received, stored briefly, and then transmitted to the next node. Packet-switching networks are commonly used for terminal-to-computer and computer-to-computer communications.

Frame Relay Packet switching was developed at a time when digital long-distance transmission facilities exhibited a relatively high error rate compared to today's facilities. As a result, there is a considerable amount of overhead built into packet-switching schemes to compensate for errors. The overhead includes additional bits added to each packet to introduce redundancy and additional processing at the end stations and the intermediate switching nodes to detect and recover from errors.

With modern high-speed telecommunications systems, this overhead is unnecessary and counterproductive. It is unnecessary because the rate of errors has been dramatically lowered and any remaining errors can easily be caught in the end systems by logic that operates above the level of the packet-switching logic. It is counterproductive because the overhead involved soaks up a significant fraction of the high capacity provided by the network.

Frame relay was developed to take advantage of these high data rates and low error rates. Whereas the original packet-switching networks were designed with a data rate to the end user of about 64 kbps, frame relay networks are designed to operate efficiently at user data rates of up to 2 Mbps. The key to achieving these high data rates is to strip out most of the overhead involved with error control.

ATM Asynchronous transfer mode, sometimes referred to as cell relay, is a culmination of developments in circuit switching and packet switching. ATM can be viewed as an evolution from frame relay. The most obvious difference between frame relay and ATM is that frame relay uses variable-length packets, called frames, and ATM uses fixed-length packets, called cells. As with frame relay, ATM provides little overhead for error control, depending on the inherent reliability of the transmission system and on higher layers of logic in the end systems to catch and correct errors. By using a fixed packet length, the processing overhead is reduced even further for ATM compared to frame relay. The result is that ATM is designed to work in the range of 10s and 100s of Mbps, and in the Gbps range.

ATM can also be viewed as an evolution from circuit switching. With circuit switching, only fixed-data-rate circuits are available to the end system. ATM allows the definition of multiple virtual channels with data rates that are dynamically defined at the time the virtual channel is created. By using small, fixed-size cells, ATM is so efficient that it can offer a constant-data-rate channel even though it is

using a packet-switching technique. Thus, ATM extends circuit switching to allow multiple channels with the data rate on each channel dynamically set on demand.

Local Area Networks

As with WANs, a LAN is a communications network that interconnects a variety of devices and provides a means for information exchange among those devices. There are several key distinctions between LANs and WANs:

1. The scope of the LAN is small, typically a single building or a cluster of buildings. This difference in geographic scope leads to different technical solutions, as we shall see.
2. It is usually the case that the LAN is owned by the same organization that owns the attached devices. For WANs, this is less often the case, or at least a significant fraction of the network assets is not owned. This has two implications. First, care must be taken in the choice of LAN, because there may be a substantial capital investment (compared to dial-up or leased charges for WANs) for both purchase and maintenance. Second, the network management responsibility for a LAN falls solely on the user.
3. The internal data rates of LANs are typically much greater than those of WANs.

LANs come in a number of different configurations. The most common are switched LANs and wireless LANs. The most common switched LAN is a switched Ethernet LAN, which may consist of a single switch with a number of attached devices, or a number of interconnected switches. The most common type of wireless LANs are Wi-Fi LANs.

Wireless Networks

As was just mentioned, wireless LANs are widely used in business environments. Wireless technology is also common for both wide area voice and data networks. Wireless networks provide advantages in the areas of mobility and ease of installation and configuration.

1.5 THE INTERNET

Origins of the Internet

The Internet evolved from the ARPANET, which was developed in 1969 by the Advanced Research Projects Agency (ARPA) of the U.S. Department of Defense. It was the first operational packet-switching network. ARPANET began operations in four locations. Today the number of hosts is in the hundreds of millions, the number of users in the billions, and the number of countries participating nearing 200. The number of connections to the Internet continues to grow exponentially.

The network was so successful that ARPA applied the same packet-switching technology to tactical radio communication (packet radio) and to satellite communication (SATNET). Because the three networks operated in very different

communication environments, the appropriate values for certain parameters, such as maximum packet size, were different in each case. Faced with the dilemma of integrating these networks, Vint Cerf and Bob Kahn of ARPA developed methods and protocols for *internetworking*—that is, communicating across arbitrary, multiple, packet-switched networks. They published a very influential paper in May 1974 [CERF74] outlining their approach to a Transmission Control Protocol. The proposal was refined and details filled in by the ARPANET community, with major contributions from participants from European networks eventually leading to the TCP (Transmission Control Protocol) and IP (Internet Protocol) protocols, which, in turn, formed the basis for what eventually became the TCP/IP protocol suite. This provided the foundation for the Internet.

Key Elements

Figure 1.5 illustrates the key elements that comprise the Internet. The purpose of the Internet, of course, is to interconnect end systems, called **hosts**; these include PCs, workstations, servers, mainframes, and so on. Most hosts that use the Internet are connected to a **network**, such as a local area network or a wide area network (WAN). These networks are in turn connected by **routers**. Each router attaches to two or more networks. Some hosts, such as mainframes or servers, connect directly to a router rather than through a network.

In essence, the Internet operates as follows. A host may send data to another host anywhere on the Internet. The source host breaks the data to be sent into a

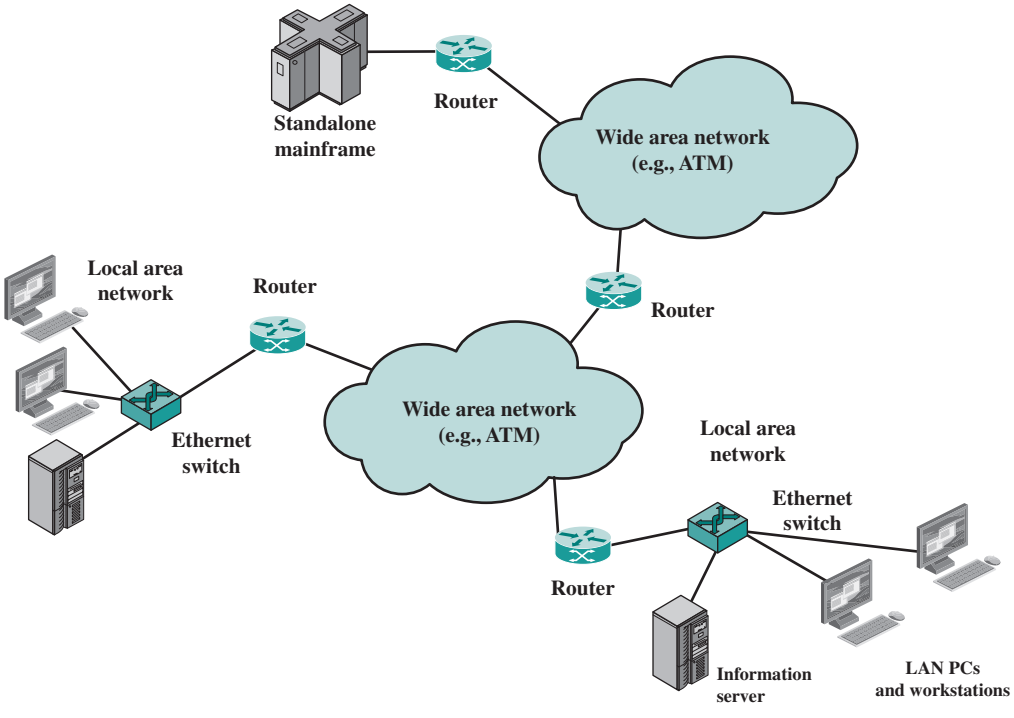


Figure 1.5 Key Elements of the Internet

sequence of packets, called **IP datagrams** or **IP packets**. Each packet includes a unique numeric address of the destination host. This address is referred to as an **IP address**, because the address is carried in an IP packet. Based on this destination address, each packet travels through a series of routers and networks from source to destination. Each router, as it receives a packet, makes a routing decision and forwards the packet along its way to the destination.

Internet Architecture

The Internet today is made up of thousands of overlapping hierarchical networks. Because of this, it is not practical to attempt a detailed description of the exact architecture or topology of the Internet. However, an overview of the common, general characteristics can be made. Figure 1.6 illustrates the discussion and Table 1.2 summarizes the terminology.

A key element of the Internet is the set of hosts attached to it. Simply put, a host is a computer. Today, computers come in many forms, including mobile phones and even cars. All of these forms can be hosts on the Internet. Hosts are sometimes grouped together in a LAN. This is the typical configuration in a corporate environment. Individual hosts and LANs are connected to an **Internet service provider (ISP)** through a **point of presence (POP)**. The connection is made in a series of steps starting with the **customer premises equipment (CPE)**. The CPE is the communications equipment located onsite with the host.

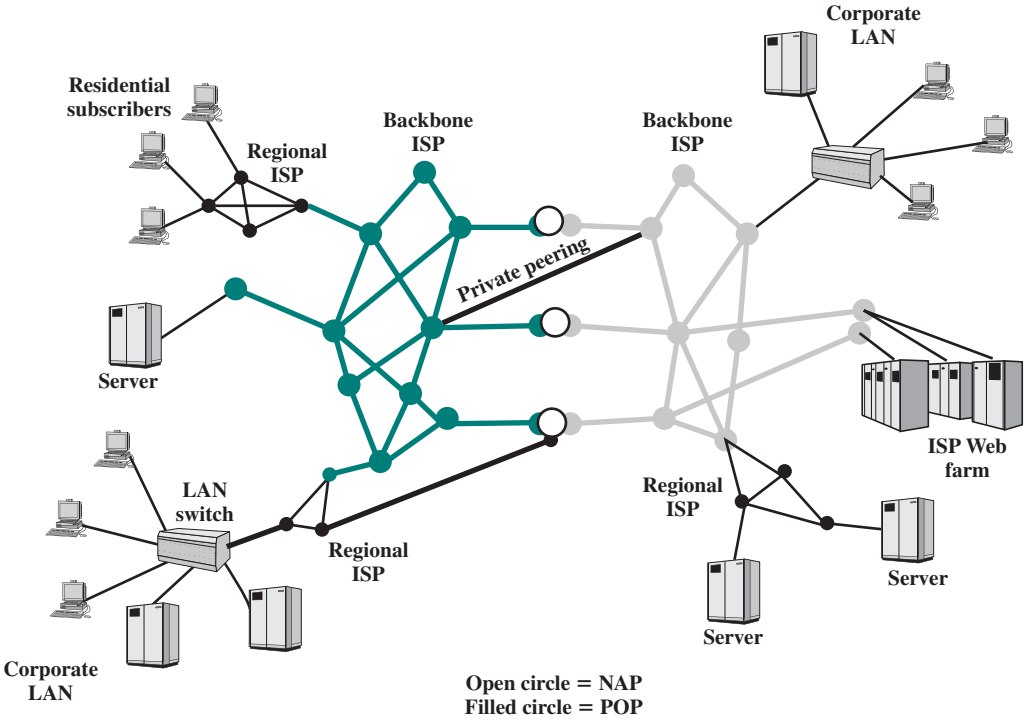


Figure 1.6 Simplified View of Portion of Internet

Table 1.2 Internet Terminology

Central Office (CO)

The place where telephone companies terminate customer lines and locate switching equipment to interconnect those lines with other networks.

Customer Premises Equipment (CPE)

Telecommunications equipment that is located on the customer's premises (physical location) rather than on the provider's premises or in between. Telephone handsets, modems, cable TV set-top boxes, and digital subscriber line routers are examples. Historically, this term referred to equipment placed at the customer's end of the telephone line and usually owned by the telephone company. Today, almost any end-user equipment can be called customer premises equipment and it can be owned by the customer or by the provider.

Internet Service Provider (ISP)

A company that provides other companies or individuals with access to, or presence on, the Internet. An ISP has the equipment and the telecommunication line access required to have a POP on the Internet for the geographic area served. The larger ISPs have their own high-speed leased lines so that they are less dependent on the telecommunication providers and can provide better service to their customers.

Network Access Point (NAP)

In the United States, a network access point (NAP) is one of several major Internet interconnection points that serve to tie all the ISPs together. Originally, four NAPs—in New York, Washington, D.C., Chicago, and San Francisco—were created and supported by the National Science Foundation as part of the transition from the original U.S. government–financed Internet to a commercially operated Internet. Since that time, several new NAPs have arrived, including WorldCom's "MAE West" site in San Jose, California, and ICS Network Systems' "Big East."

The NAPs provide major switching facilities that serve the public in general. Companies apply to use the NAP facilities. Much Internet traffic is handled without involving NAPs, using peering arrangements and interconnections within geographic regions.

Network Service Provider (NSP)

A company that provides backbone services to an Internet service provider. Typically, an ISP connects at a point called an Internet exchange (IX) to a regional ISP that in turn connects to an NSP backbone.

Point of Presence (POP)

A site that has a collection of telecommunications equipment, usually refers to ISP or telephone company sites. An ISP POP is the edge of the ISP's network; connections from users are accepted and authenticated here. An Internet access provider may operate several POPs distributed throughout its area of operation to increase the chance that their subscribers will be able to reach one with a local telephone call. The largest national ISPs have POPs all over the country.

For a number of home users, the CPE was traditionally a 56-kbps modem. This was adequate for e-mail and related services but marginal for graphics-intensive Web surfing. Today's CPE offerings provide greater capacity and guaranteed service in some cases. A sample of these access technologies includes DSL, cable modem, terrestrial wireless, and satellite. Users who connect to the Internet through their work often use workstations or PCs connected to their employer-owned LANs, which in turn connect through shared organizational trunks to an ISP. In these cases the shared circuit is often a T-1 connection (1.544 Mbps), while for very large organizations T-3 connections (44.736 Mbps) are sometimes found. Alternatively, an organization's LAN may be hooked to a WAN, such as a frame relay network, which in turn connects to an ISP.

The CPE is physically attached to the "local loop" or "last mile." This is the infrastructure between a provider's installation and the site where the host is located. For example, a residential user with a DSL modem attaches the modem to the telephone line. The telephone line is typically a pair of copper wires that runs from the house to a **central office (CO)** owned and operated by the telephone company, with perhaps the use of optical fiber for a large portion of that link. If the home user has a cable modem, the local loop is the coaxial cable that runs from the home to the cable company facilities. The preceding examples are a bit of an oversimplification, but they suffice for this discussion. In many cases the wires that leave a home are aggregated with wires from other homes and then converted to a different media such as fiber. In these cases the term *local loop* still refers to the path from the home to the CO or cable facility. The local loop provider is not necessarily the ISP. In many cases the local loop provider is the telephone company and the ISP is a large, national service organization. Often, however, the local loop provider is also the ISP.

Other forms of CPE-ISP connection do not go through a telephone company CO. For example, a cable link would connect the local user to the cable company site, which would include or link to an ISP. Mobile users can take advantage of a wireless link to a Wi-Fi access point that provides access to the Internet. And corporate access to an ISP may be by dedicated high-speed links or through a wide area network, such as an asynchronous transfer mode or frame relay network.

The ISP provides access to its larger network through a POP. A POP is simply a facility where customers can connect to the ISP network. The facility is sometimes owned by the ISP, but often the ISP leases space from the local loop carrier. A POP can be as simple as a bank of modems and an access server installed in a rack at the CO. The POPs are usually spread out over the geographic area where the provider offers service. The ISP acts as a gateway to the Internet, providing many important services. For most home users, the ISP provides the unique numeric IP address needed to communicate with other Internet hosts. Most ISPs also provide name resolution and other essential network services. The most important service an ISP provides, though, is access to other ISP networks. Access is facilitated by formal peering agreements between providers. Physical access can be implemented by connecting POPs from different ISPs. This can be done directly with a local connection if the POPs are collocated or with leased lines when the POPs are not collocated. A more commonly used mechanism is the **network access point (NAP)**.

A NAP is a physical facility that provides the infrastructure to move data between connected networks. In the United States, the National Science Foundation (NSF) privatization plan called for the creation of four NAPs. The NAPs were built and operated by the private sector. The number of NAPs has grown significantly over the years, and the technology employed has shifted from Fiber Distributed Data Interface (FDDI) and Ethernet to ATM and Gigabit Ethernet. Most NAPs today have an ATM core. The networks connected at a NAP are owned and operated by **network service providers (NSPs)**. A NSP can also be an ISP but this is not always the case. Peering agreements are between NSPs and do not include the NAP operator. The NSPs install routers at the NAP and connect them to the NAP infrastructure. The NSP equipment is responsible for routing, and the NAP infrastructure provides the physical access paths between routers.

1.6 AN EXAMPLE CONFIGURATION

To give some feel for the scope of concerns of this book, Figure 1.7 illustrates some of the typical communications and network elements in use today. At the center of the figure is an IP backbone network, which could represent a portion of the Internet or an enterprise IP network. Typically, the backbone consists of high-performance routers, called *core routers*, interconnected with high-volume optical links. The optical links make use of what is known as wavelength division multiplexing (WDM), such that each link has multiple logical channels occupying different portions of the optical bandwidth.

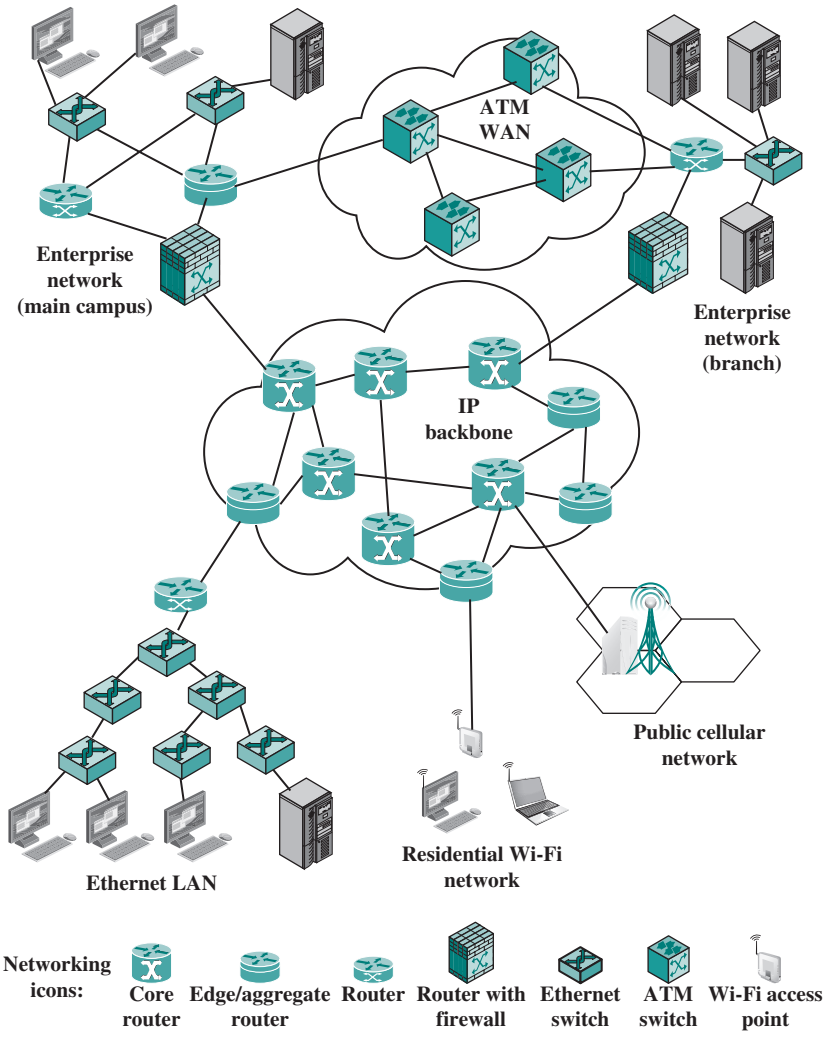


Figure 1.7 A Networking Configuration

At the periphery of an IP backbone are routers that provide connectivity to external networks and users. These routers are sometimes referred to as *edge routers,* or *aggregate routers.* Aggregate routers are also used within an enterprise network to connect a number of routers and switches to external resources, such as an IP backbone or a high-speed WAN. As an indication of the capacity requirements for core and aggregate routers, [XI11] reports on an analysis that projects these requirements for Internet backbone providers and large enterprise networks in China. The analysis concludes the aggregation router requirements to be in the range of 200 Gbps to 400 Gbps per optical link by 2020, and 400 Gbps to 1 Tbps per optical link for core routers by 2020.

The upper part of the figure depicts a portion of what might be a large enterprise network. The figure shows two sections of the network connected via a private high-speed asynchronous transfer mode (ATM) WAN, with switches interconnected with optical links. Enterprise assets are connected to, and protected from, an IP backbone or the Internet via routers with firewall capability, a not uncommon arrangement for implementing the firewall.

The lower left of the figure depicts what might be a configuration for a small or medium-size business, which relies on an Ethernet LAN configuration. Connection to the Internet through a router could be through a cable digital subscriber line connection or a dedicated high-speed link.

The lower portion of the figure also shows an individual residential user connected to an Internet service provider through some sort of subscriber connection. Common examples of such a connection are a digital subscriber line, which provides a high-speed link over telephone lines and requires a special DSL modem; and a cable TV facility, which requires a cable modem, or some type of wireless connection. In each case, there are separate issues concerning signal encoding, error control, and the internal structure of the subscriber network.

Finally, mobile devices, such as smartphones and tablets, can connect to the Internet through the public cellular network, which has a high-speed connection, typically optical, to the Internet.

The ISPs are not explicitly shown in the figure. Typically, an ISP network will consist of a number of interconnected routers connected to the Internet through a high-speed link. The Internet consists of a number of interconnected routers that span the globe. The routers forward packets of data from source to destination through the Internet.

A variety of design issues, such as signal encoding and error control, relate to the links between adjacent elements, such as between routers on the Internet or between switches in the ATM network, or between a subscriber and an ISP. The internal structure of the various networks (telephone, ATM, Ethernet) raises additional issues. We will be occupied throughout much of this book with the design features suggested by Figure 1.7.

CHAPTER 2

PROTOCOL ARCHITECTURE, TCP/IP, AND INTERNET-BASED APPLICATIONS

LEARNING OBJECTIVES

After reading this chapter, you should be able to:

- Define the term *protocol architecture* and explain the need for and benefits of a communications architecture.
- Describe the TCP/IP architecture and explain the functioning of each layer.
- Explain the motivation for the development of a standardized architecture and the reasons why a customer should use products based on a protocol architecture standard in preference to products based on a proprietary architecture.
- Explain the need for internetworking.
- Describe the operation of a router within the context of TCP/IP to provide internetworking.

This chapter provides a context for the detailed material that follows. It shows how the concepts of Parts Two through Five fit into the broader area of computer networks and computer communications. This chapter may be read in its proper sequence or it may be deferred until the beginning of Part Three, Four, or Five.[1]

We begin this chapter by introducing the concept of a layered **protocol architecture**. We then examine the most important such architecture, the TCP/IP protocol suite. TCP/IP is an Internet-based protocol suite and is the framework for developing a complete range of computer communications standards. Another well-known architecture is the **Open Systems Interconnection (OSI)** reference model. OSI is a standardized architecture that is often used to describe communications functions but that is now rarely implemented. OSI is examined in Appendix E.

2.1 THE NEED FOR A PROTOCOL ARCHITECTURE

When computers, terminals, and/or other data processing devices exchange data, the procedures involved can be quite complex. Consider, for example, the transfer of a file between two computers. There must be a data path between the two

[1]The reader may find it helpful just to skim this chapter on a first reading and then reread it more carefully just before embarking on Part Five.

computers, either directly or via a communication network. But more is needed. Typical tasks to be performed:

1. The source system must either activate the direct data communication path or inform the communication network of the identity of the desired destination system.
2. The source system must ascertain that the destination system is prepared to receive data.
3. The file transfer application on the source system must ascertain that the file management program on the destination system is prepared to accept and store the file for this particular user.
4. If the file formats used on the two systems are different, one or the other system must perform a format translation function.

It is clear that there must be a high degree of cooperation between the two computer systems. Instead of implementing the logic for this as a single module, the task is broken up into subtasks, each of which is implemented separately. In a protocol architecture, the modules are arranged in a vertical stack. Each layer in the stack performs a related subset of the functions required to communicate with another system. It relies on the next lower layer to perform more primitive functions and to conceal the details of those functions. It provides services to the next higher layer. Ideally, layers should be defined so that changes in one layer do not require changes in other layers.

Of course, it takes two to communicate, so the same set of layered functions must exist in two systems. Communication is achieved by having the corresponding, or **peer**, layers in two systems communicate. The **peer layers** communicate by means of formatted blocks of data that obey a set of rules or conventions known as a **protocol**. The key features of a protocol are as follows:

- **Syntax**: Concerns the format of the data blocks
- **Semantics**: Includes control information for coordination and error handling
- **Timing**: Includes speed matching and sequencing

Appendix 2A provides a specific example of a protocol, the **Internet** standard Trivial File Transfer Protocol (TFTP).

2.2 A SIMPLE PROTOCOL ARCHITECTURE

In very general terms, distributed data communications can be said to involve three agents: applications, computers, and networks. Examples of applications include file transfer and electronic mail. These applications execute on computers that typically support multiple simultaneous applications. Computers are connected to networks, and the data to be exchanged are transferred by the network from one computer to another. Thus, the transfer of data from one application to another involves first getting the data to the computer in which the application resides and then getting it to the intended application within the computer.

With these concepts in mind, it appears natural to organize the communication task into three relatively independent layers: network access layer, transport layer, and application layer.

The **network access layer** is concerned with the exchange of data between a computer and the network to which it is attached. The sending computer must provide the network with the address of the destination computer, so that the network may route the data to the appropriate destination. The sending computer may wish to invoke certain services, such as priority, that might be provided by the network. The specific software used at this layer depends on the type of network to be used; different standards have been developed for circuit switching, packet switching, local area networks (LANs), and others. For example, IEEE 802 is a standard that specifies the access to a LAN; this standard is described in Part Three. It makes sense to put those functions having to do with network access into a separate layer. By doing this, the remainder of the communications software, above the network access layer, need not be concerned about the specifics of the network to be used. The same higher-layer software should function properly regardless of the particular network to which the computer is attached.

Regardless of the nature of the applications that are exchanging data, there is usually a requirement that data be exchanged reliably. That is, we would like to be assured that all of the data arrive at the destination application and that the data arrive in the same order in which they were sent. As we shall see, the mechanisms for providing reliability are essentially independent of the nature of the applications. Thus, it makes sense to collect those mechanisms in a common layer shared by all applications; this is referred to as the **transport layer**.

Finally, the **application layer** contains the logic needed to support the various user applications. For each different type of application, such as file transfer, a separate module is needed that is peculiar to that application.

Figures 2.1 and 2.2 illustrate this simple architecture. Figure 2.1 shows three computers connected to a network. Each computer contains software at the network access and transport layers and at the application layer for one or more applications. For successful communication, every entity in the overall system must have a unique address. In the three-layer model, two levels of addressing are needed. Each computer on the network has a unique network address; this allows the network to deliver data to the proper computer. Each application on a computer has an address that is unique within that computer; this allows the transport layer to support multiple applications at each computer. These latter addresses are known as **service access points** (SAPs), or **ports**, connoting the fact that each application is individually accessing the services of the transport layer.

Figure 2.1 indicates that modules at the same level (peers) on different computers communicate with each other by means of a protocol. An application entity (e.g., a file transfer application) in one computer communicates with an application in another computer via an application-level protocol (e.g., the File Transfer Protocol). The interchange is not direct (indicated by the dashed line) but is mediated by a transport protocol that handles many of the details of transferring data between two computers. The transport protocol is also not direct, but relies on a network-level protocol to achieve network access and to route data through the network to the destination system. At each level, the cooperating peer entities focus on what they need to communicate to each other.

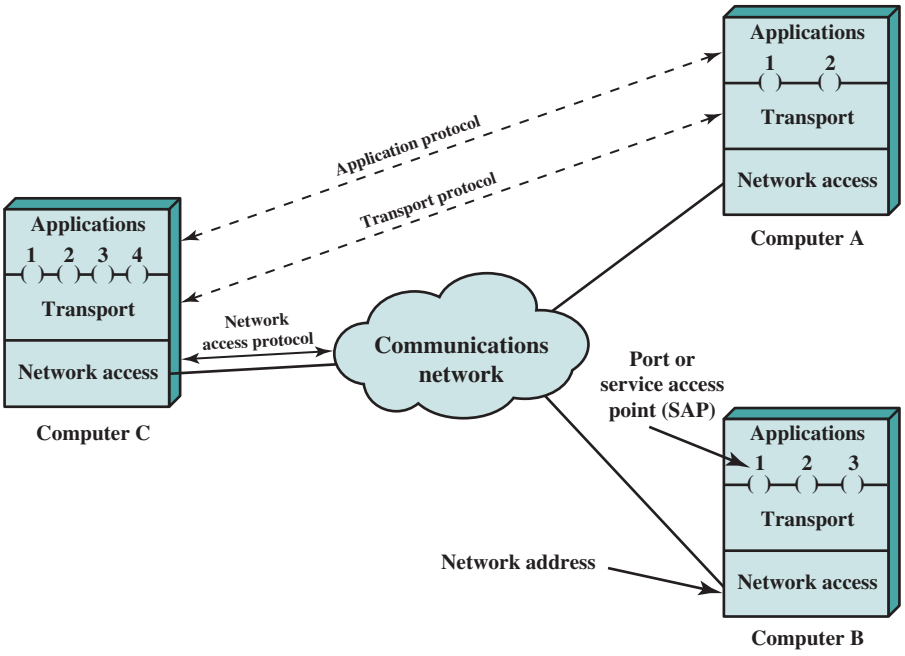


Figure 2.1 Protocol Architectures and Networks

Let us trace a simple operation. Suppose that an application, associated with port 1 at computer A, wishes to send a message to another application, associated with port 2 at computer B. The application at A hands the message over to its transport layer with instructions to send it to port 2 on computer B. The transport layer hands the message over to the network access layer, which instructs the network to send the message to computer B. Note that the network need not be told the identity of the destination port. All that it needs to know is that the data are intended for computer B.

To control this operation, control information, as well as user data, must be transmitted, as suggested in Figure 2.2. Let us say that the sending application generates a block of data and passes this to the transport layer. The transport layer may break this block into two smaller pieces for convenience, as discussed subsequently. To each of these pieces the transport layer appends a transport **header**, containing protocol control information. The addition of control information to data is referred to as **encapsulation**. The combination of data from the next higher layer and control information is known as a **protocol data unit (PDU)**; in this case, it is referred to as a transport PDU. Transport PDUs are typically called **segments**. The header in each segment contains control information to be used by the peer transport protocol at computer B. Examples of items that may be stored in this header include the following:

- **Source port:** This indicates the application that sent the data.
- **Destination port:** When the destination transport layer receives the segment, it must know to which application the data are to be delivered.

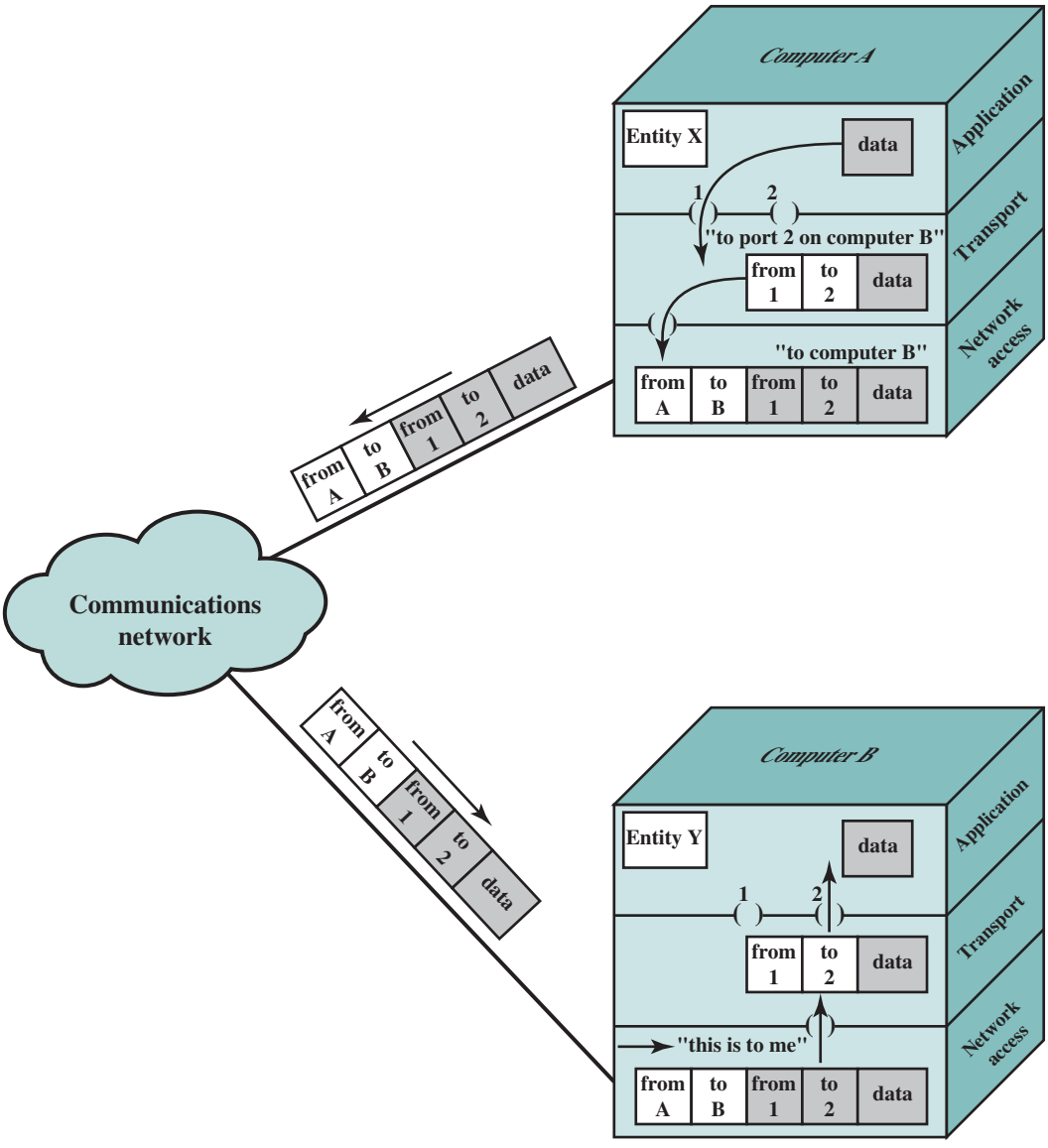


Figure 2.2 Protocols in a Simplified Architecture

- **Sequence number:** Because the transport protocol is sending a sequence of segments, it numbers them sequentially so that if they arrive out of order, the destination transport entity may reorder them.
- **Error-detection code:** The sending transport entity may include a code that is a function of the contents of the segment. The receiving transport protocol performs the same calculation and compares the result with the incoming code. A discrepancy results if there has been some error in transmission. In that case, the receiver can discard the segment and take corrective action. This code is also referred to as a **checksum** or **frame check sequence**.

The next step is for the transport layer to hand each segment over to the network layer, with instructions to transmit it to the destination computer. To satisfy this request, the network access protocol must present the data to the network with a request for transmission. As before, this operation requires the use of control information. In this case, the **network access protocol (NAP)** appends a network access header to the data it receives from the transport layer, creating a network access PDU, typically called a **packet**. Examples of the items that may be stored in the header include the following:

- **Source computer address:** Indicates the source of this packet.
- **Destination computer address:** The network must know to which computer on the network the data are to be delivered.
- **Facilities requests:** The network access protocol might want the network to make use of certain facilities, such as priority.

Note that the transport header is not "visible" at the network access layer; the network access layer is not concerned with the contents of the transport segment.

The network accepts the network packet from A and delivers it to B. The network access module in B receives the packet, strips off the packet header, and transfers the enclosed transport segment to B's transport layer module. The transport layer examines the segment header and, on the basis of the port field in the header, delivers the enclosed record to the appropriate application, in this case the file transfer module in B.

2.3 THE TCP/IP PROTOCOL ARCHITECTURE

TCP/IP is a result of protocol research and development conducted on the experimental packet-switched network, ARPANET, funded by the Defense Advanced Research Projects Agency (DARPA), and is generally referred to as the TCP/IP protocol suite. This protocol suite consists of a large collection of protocols that have been issued as Internet standards by the Internet Activities Board (IAB). Appendix C provides a discussion of Internet standards.

The TCP/IP Layers

In general terms, computer communications can be said to involve three agents: applications, computers, and networks. Examples of applications include file transfer and electronic mail. The applications that we are concerned with here are distributed applications that involve the exchange of data between two computer systems. These applications, and others, execute on computers that can often support multiple simultaneous applications. Computers are connected to networks, and the data to be exchanged are transferred by the network from one computer to another. Thus, the transfer of data from one application to another involves first getting the data to the computer in which the application resides and then getting the data to the intended application within the computer. With these concepts in

mind, we can organize the communication task into five relatively independent layers (Figure 2.3):

- Physical layer
- Network access/data link layer
- Internet layer
- Host-to-host, or transport layer
- Application layer

The **physical layer** covers the physical interface between a data transmission device (e.g., workstation, computer) and a transmission medium or network. This layer is concerned with specifying the characteristics of the transmission medium, the nature of the signals, the data rate, and related matters.

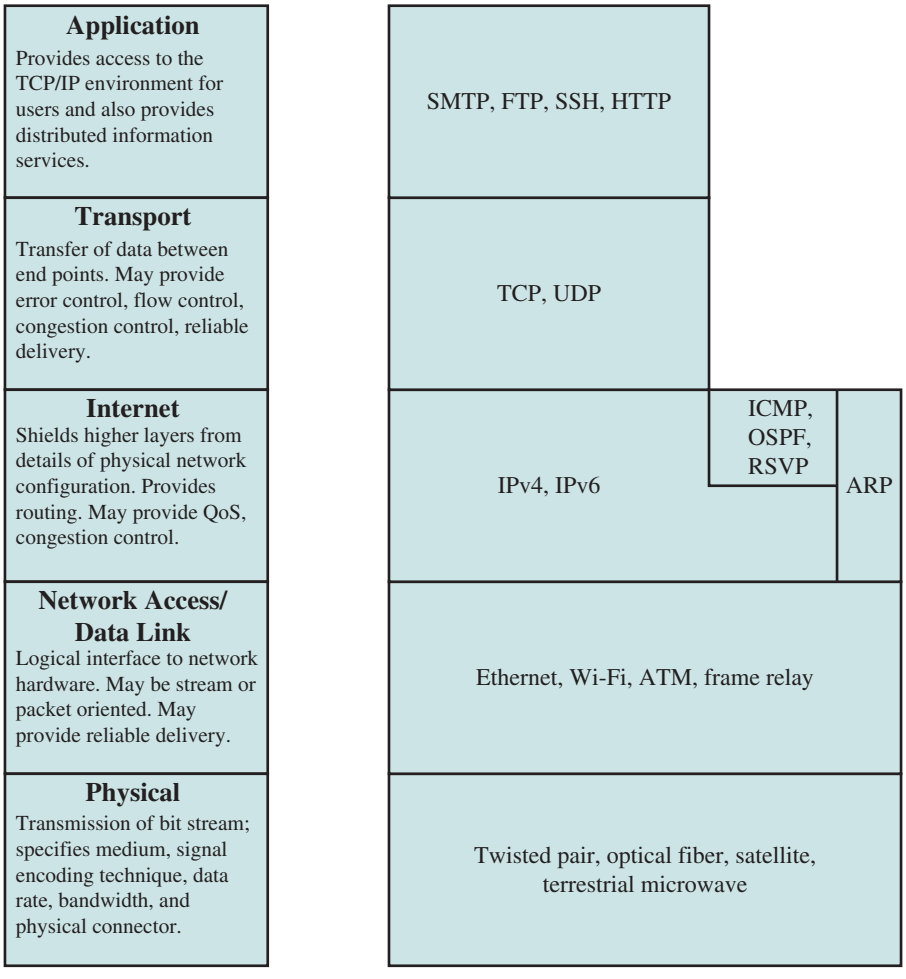


Figure 2.3 The TCP/IP Layers and Example Protocols

The **network access/data link layer** is discussed in Section 2.2. This layer is concerned with access to and routing data across a network for two end systems attached to the same network. In those cases where two devices are attached to different networks, procedures are needed to allow data to traverse multiple interconnected networks. This is the function of the **internet layer**. The **Internet Protocol (IP)** is used at this layer to provide the routing function across multiple networks. This protocol is implemented not only in the end systems but also in **routers**. A router is a processor that connects two networks and whose primary function is to relay data from one network to the other on its route from the source to the destination end system.

The **host-to-host layer**, or **transport layer**, may provide reliable end-to-end service, as discussed in Section 2.2, or merely an end-to-end delivery service without reliability mechanisms. The **Transmission Control Protocol (TCP)** is the most commonly used protocol to provide this functionality.

Finally, the *application layer* contains the logic needed to support the various user applications. For each different type of application, such as file transfer, a separate module is needed that is peculiar to that application.

Operation of TCP and IP

Figure 2.4 indicates how these protocols are configured for communications. To make clear that the total communications facility may consist of multiple networks, the constituent networks are usually referred to as **subnetworks**. Some sort of

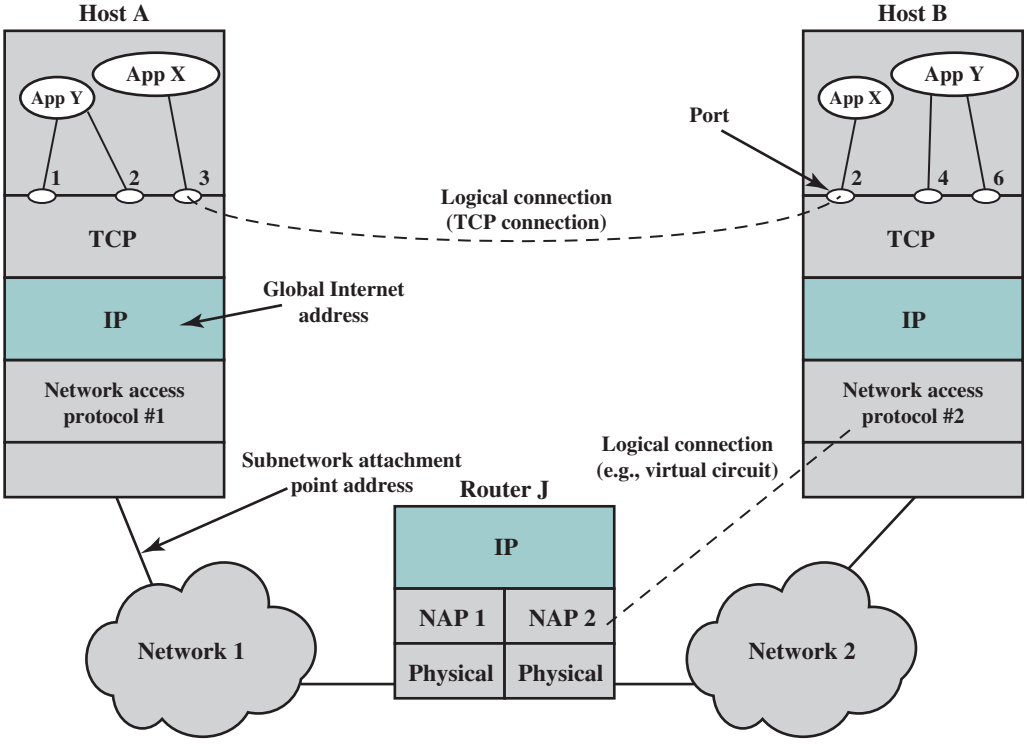


Figure 2.4 TCP/IP Concepts

network access protocol, such as the Ethernet or Wi-Fi logic, is used to connect a computer to a subnetwork. This protocol enables the host to send data across the subnetwork to another host or, if the target host is on another subnetwork, to a router that will forward the data. IP is implemented in all of the end systems and the routers. It acts as a relay to move a block of data from one host, through one or more routers, to another host. TCP is implemented only in the end systems; it keeps track of the blocks of data to assure that all are delivered reliably to the appropriate application.

As is mentioned in Section 2.2, every entity in the overall system must have a unique address. Each host on a subnetwork must have a unique global internet address; this allows the data to be delivered to the proper host. Each process with a host must have an address that is unique within the host; this allows the host-to-host protocol (TCP) to deliver data to the proper process. These latter addresses are known as *ports*.

Let us trace a simple operation. Suppose that a process, associated with port 3 at host A, wishes to send a message to another process, associated with port 2 at host B. The process at A hands the message down to TCP with instructions to send it to host B, port 2. TCP hands the message down to IP with instructions to send it to host B. Note that IP need not be told the identity of the destination port. All it needs to know is that the data are intended for host B. Next, IP hands the message down to the network access layer (e.g., Ethernet logic) with instructions to send it to router J (the first hop on the way to B).

To control this operation, control information, as well as user data, must be transmitted, as suggested in Figure 2.5. Let us say that the sending process generates a block of data and passes this to TCP. TCP may break this block into smaller pieces to make it more manageable. To each of these pieces, TCP appends control information known as the TCP header, forming a **TCP segment**. The control information

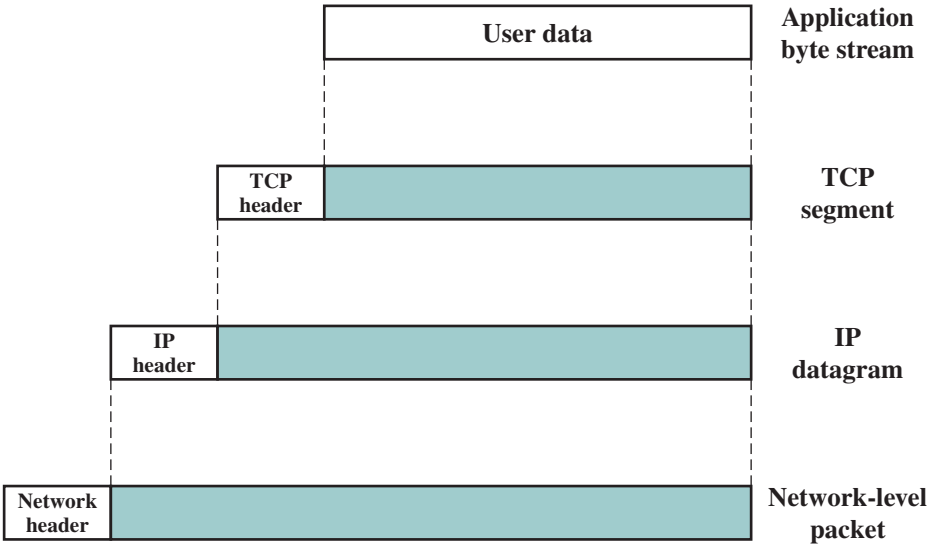


Figure 2.5 Protocol Data Units (PDUs) in the TCP/IP Architecture

is to be used by the peer TCP entity at host B. Examples of items in this header include:

- **Destination port:** When the TCP entity at B receives the segment, it must know to whom the data are to be delivered.
- **Sequence number:** TCP numbers the segments that it sends to a particular destination port sequentially, so that if they arrive out of order, the TCP entity at B can reorder them.
- **Checksum:** The sending TCP includes a code that is a function of the contents of the remainder of the segment. The receiving TCP performs the same calculation and compares the result with the incoming code. A discrepancy results if there has been some error in transmission.

Next, TCP hands each segment over to IP, with instructions to transmit it to B. These segments must be transmitted across one or more subnetworks and relayed through one or more intermediate routers. This operation, too, requires the use of control information. Thus IP appends a header of control information to each segment to form an **IP datagram**. An example of an item stored in the IP header is the destination host address (in this example, B).

Finally, each IP datagram is presented to the network access layer for transmission across the first subnetwork in its journey to the destination. The network access layer appends its own header, creating a packet, or frame. The packet is transmitted across the subnetwork to router J. The packet header contains the information that the subnetwork needs to transfer the data across the subnetwork.

At router J, the packet header is stripped off and the IP header is examined. On the basis of the destination address information in the IP header, the IP module in the router directs the datagram out across subnetwork 2 to B. To do this, the datagram is again augmented with a network access header.

When the data are received at B, the reverse process occurs. At each layer, the corresponding header is removed, and the remainder is passed on to the next higher layer, until the original user data are delivered to the destination process.

TCP and UDP

For most applications running as part of the TCP/IP architecture, the transport layer protocol is TCP. TCP provides a reliable connection for the transfer of data between applications. A connection is simply a temporary logical association between two entities in different systems. A logical connection refers to a given pair of port values. For the duration of the connection, each entity keeps track of TCP segments coming and going to the other entity, in order to regulate the flow of segments and to recover from lost or damaged segments.

Figure 2.6a shows the header format for TCP, which is a minimum of 20 octets, or 160 bits. The Source Port and Destination Port fields identify the applications at the source and destination systems that are using this connection. The Sequence Number, Acknowledgment Number, and Window fields provide flow control and error control. The checksum is a 16-bit frame check sequence used to detect errors in the TCP segment. Chapter 15 provides more details.

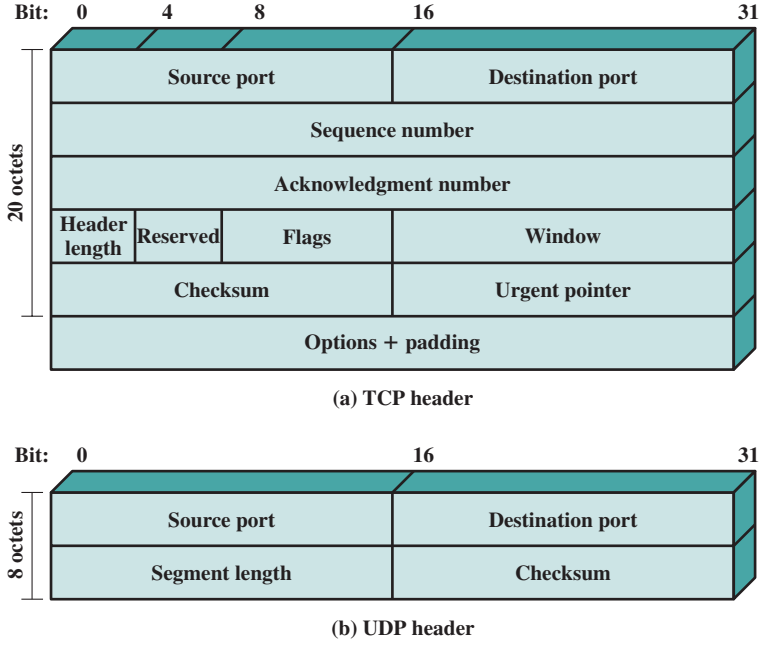
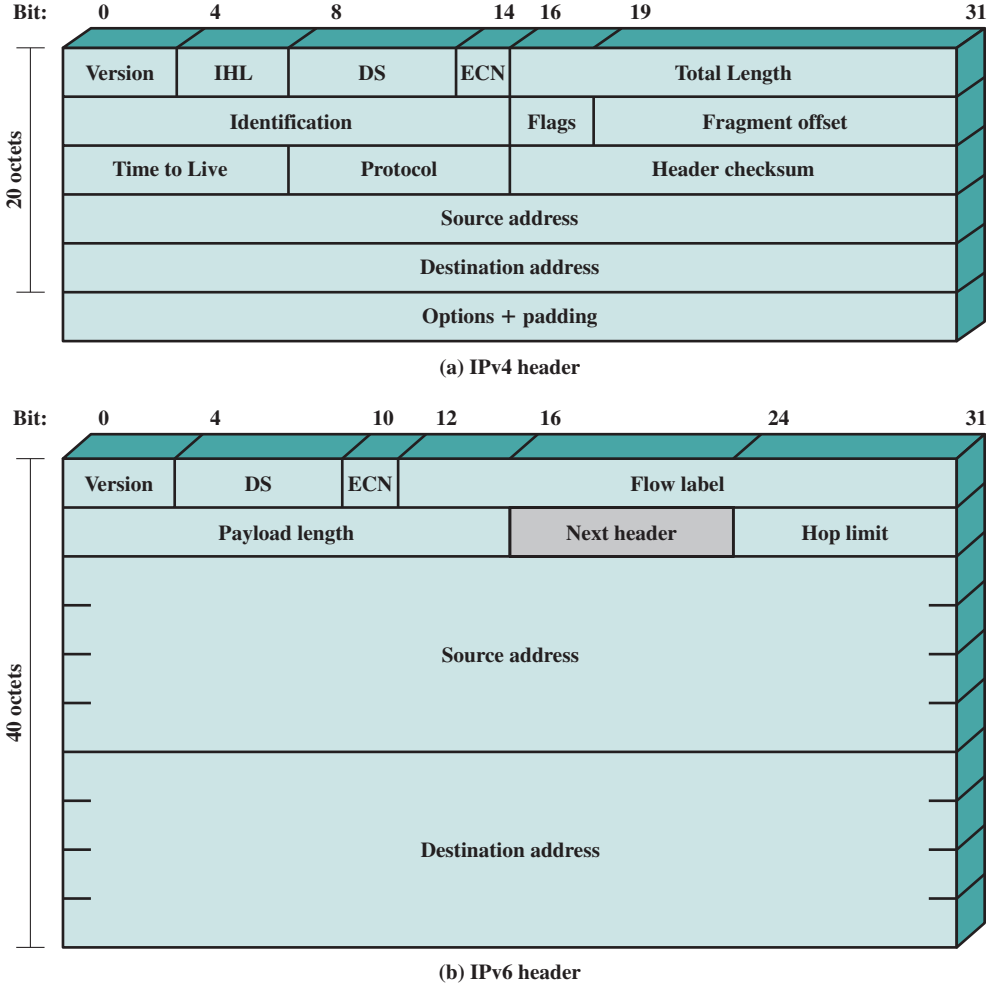


Figure 2.6 TCP and UDP Headers

In addition to TCP, there is one other transport-level protocol that is in common use as part of the TCP/IP protocol suite: the **User Datagram Protocol (UDP)**. UDP does not guarantee delivery, preservation of sequence, or protection against duplication. UDP enables a procedure to send messages to other procedures with a minimum of protocol mechanism. Some transaction-oriented applications make use of UDP; one example is SNMP (Simple Network Management Protocol), the standard network management protocol for TCP/IP networks. Because it is connectionless, UDP has very little to do. Essentially, it adds a port addressing capability to IP. This is best seen by examining the UDP header, shown in Figure 2.6b. UDP also includes a checksum to verify that no error occurs in the data; the use of the checksum is optional.

IP and IPv6

For decades, the keystone of the TCP/IP architecture has been IPv4, generally referred to as IP. Figure 2.7a shows the IP header format, which is a minimum of 20 octets, or 160 bits. The header, together with the segment from the transport layer, forms an IP-level PDU referred to as an IP datagram or an IP packet. The header includes 32-bit source and destination addresses. The Header Checksum field is used to detect errors in the header to avoid misdelivery. The Protocol field indicates which higher-layer protocol is using IP. The ID, Flags, and Fragment Offset fields are used in the fragmentation and reassembly process. Chapter 14 provides more details.



DS = Differentiated Services field
ECN = Explicit Congestion Notification field

Note: The 8-bit DS/ECN fields were formerly known as the Type of Service field in the IPv4 header and the Traffic Class field in the IPv6 header.

Figure 2.7 IP Headers

In 1995, the Internet Engineering Task Force (IETF), which develops protocol standards for the Internet, issued a specification for a next-generation IP, known then as IPng. This specification was turned into a standard in 1996 known as IPv6. IPv6 provides a number of functional enhancements over the existing IP, designed to accommodate the higher speeds of today's networks and the mix of data streams, including graphic and video, that are becoming more prevalent. But the driving force behind the development of the new protocol was the need for more addresses. IPv4 uses a 32-bit address to specify a source or destination. With the explosive growth

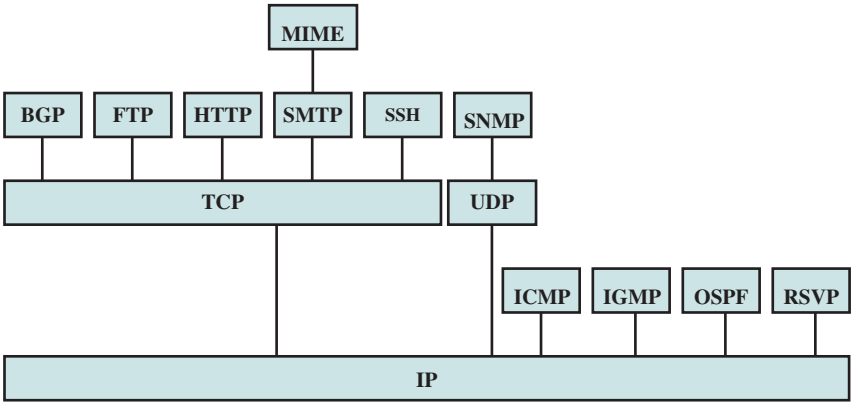
of the Internet and of private networks attached to the Internet, this address length became insufficient to accommodate all systems needing addresses. As Figure 2.7b shows, IPv6 includes 128-bit source and destination address fields.

Ultimately, all installations using TCP/IP are expected to migrate from the current IP to IPv6, but this process will take many years, if not decades.

Protocol Interfaces

Each layer in the TCP/IP suite interacts with its immediate adjacent layers. At the source, the application layer makes use of the services of the end-to-end layer and provides data down to that layer. A similar relationship exists at the interface between the transport and internet layers and at the interface of the internet and network access layers. At the destination, each layer delivers data up to the next higher layer.

This use of each individual layer is not required by the architecture. As Figure 2.8 suggests, it is possible to develop applications that directly invoke the services of any one of the layers. Most applications require a reliable end-to-end protocol and thus make use of TCP. Some special-purpose applications do not need the services of TCP. Some of these applications, such as the Simple Network Management Protocol (SNMP), use an alternative end-to-end protocol known as the User Datagram Protocol (UDP); others may make use of IP directly. Applications that do not involve **internetworking** and that do not need TCP have been developed to invoke the network access layer directly.



BGP	=	Border Gateway Protocol	OSPF	=	Open Shortest Path First
FTP	=	File Transfer Protocol	RSVP	=	Resource ReSerVation Protocol
HTTP	=	Hypertext Transfer Protocol	SMTP	=	Simple Mail Transfer Protocol
ICMP	=	Internet Control Message Protocol	SNMP	=	Simple Network Management Protocol
IGMP	=	Internet Group Management Protocol	SSH	=	Secure Shell
IP	=	Internet Protocol	TCP	=	Transmission Control Protocol
MIME	=	Multipurpose Internet Mail Extension	UDP	=	User Datagram Protocol

Figure 2.8 Some Protocols in the TCP/IP Protocol Suite

2.4 STANDARDIZATION WITHIN A PROTOCOL ARCHITECTURE

Standards and Protocol Layers

A protocol architecture, such as the TCP/IP architecture or OSI, provides a framework for standardization. Within the model, one or more protocol standards can be developed at each layer. The model defines in general terms the functions to be performed at that layer and facilitates the standards-making process in two ways:

- Because the functions of each layer are well defined, standards can be developed independently and simultaneously for each layer. This speeds up the standards-making process.
- Because the boundaries between layers are well defined, changes in standards in one layer need not affect already existing software in another layer. This makes it easier to introduce new standards.

Figure 2.9 illustrates the use of a protocol architecture as such a framework. The overall communications function is decomposed into a number of distinct layers. That is, the overall function is broken up into a number of modules, making the interfaces between modules as simple as possible. In addition, the design

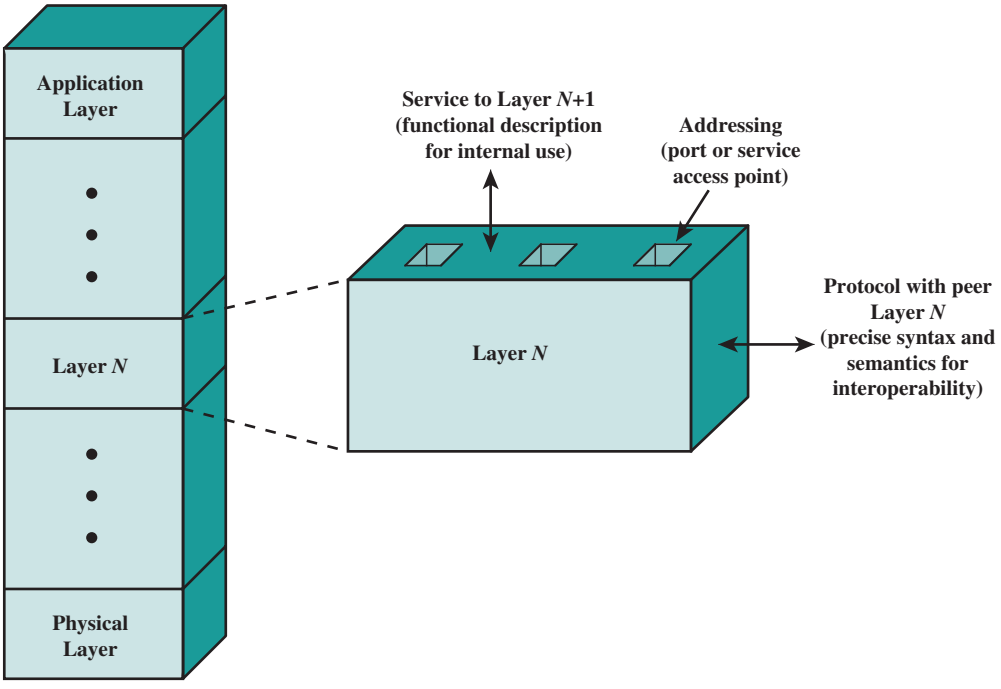


Figure 2.9 A Protocol Architecture as a Framework for Standardization

principle of information hiding is used: Lower layers are concerned with greater levels of detail; upper layers are independent of these details. Each layer provides services to the next higher layer and implements a protocol to the peer layer in other systems.

Figure 2.9 also shows more specifically the nature of the standardization required at each layer. Three elements are key:

- **Protocol specification:** Two entities at the same layer in different systems cooperate and interact by means of a protocol. Because two different open systems are involved, the protocol must be specified precisely. This includes the format of the protocol data units exchanged, the semantics of all fields, and the allowable sequence of PDUs.
- **Service definition:** In addition to the protocol or protocols that operate at a given layer, standards are needed for the services that each layer provides to the next higher layer. Typically, the definition of services is equivalent to a functional description that defines what services are provided, but not how the services are to be provided.
- **Addressing:** Each layer provides services to entities at the next higher layer. These entities are referenced by means of a port, or **service access point (SAP)**. Thus, a network service access point (NSAP) indicates a transport entity that is a user of the network service.

The need to provide a precise protocol specification for open systems is self-evident. The other two items listed warrant further comment. With respect to service definitions, the motivation for providing only a functional definition is as follows. First, the interaction between two adjacent layers takes place within the confines of a single open system and is not the concern of any other open system. Thus, as long as peer layers in different systems provide the same services to their next higher layers, the details of how the services are provided may differ from one system to another without loss of interoperability. Second, it will usually be the case that adjacent layers are implemented on the same processor. In that case, we would like to leave the system programmer free to exploit the hardware and operating system to provide an interface that is as efficient as possible.

With respect to addressing, the use of an address mechanism at each layer, implemented as a service access point, allows each layer to multiplex multiple users from the next higher layer. Multiplexing may not occur at each layer, but the model allows for that possibility.

Service Primitives and Parameters

The services between adjacent layers in a protocol architecture are expressed in terms of primitives and parameters. A primitive specifies the function to be performed, and the parameters are used to pass data and control information. The actual form of a primitive is implementation dependent. An example is a procedure call.

Four types of primitives are used in standards to define the interaction between adjacent layers in the architecture. These are defined in Table 2.1. The

Table 2.1 Service Primitive Types

Request	A primitive issued by a service user to invoke some service and to pass the parameters needed to specify fully the requested service.
Indication	A primitive issued by a service provider either to 1. indicate that a procedure has been invoked by thwe peer service user on the connection and to provide the associated parameters, or 2. notify the service user of a provider-initiated action.
Response	A primitive issued by a service user to acknowledge or complete some procedure previously invoked by an indication to that user.
Confirm	A primitive issued by a service provider to acknowledge or complete some procedure previously invoked by a request by the service user.

layout of Figure 2.10a suggests the time ordering of these events. For example, consider the transfer of data from an (*N*) entity to a peer (*N*) entity in another system. The following steps occur:

1. The source (*N*) entity invokes its $(N-1)$ entity with a *request* primitive. Associated with the primitive are the parameters needed, such as the data to be transmitted and the destination address.
2. The source $(N-1)$ entity prepares an $(N-1)$ PDU to be sent to its peer $(N-1)$ entity.
3. The destination $(N-1)$ entity delivers the data to the appropriate destination (*N*) entity via an *indication* primitive, which includes the data and source address as parameters.
4. If an acknowledgment is called for, the destination (*N*) entity issues a *response* primitive to its $(N-1)$ entity.
5. The $(N-1)$ entity conveys the acknowledgment in an $(N-1)$ PDU.
6. The acknowledgment is delivered to the (*N*) entity as a *confirm* primitive.

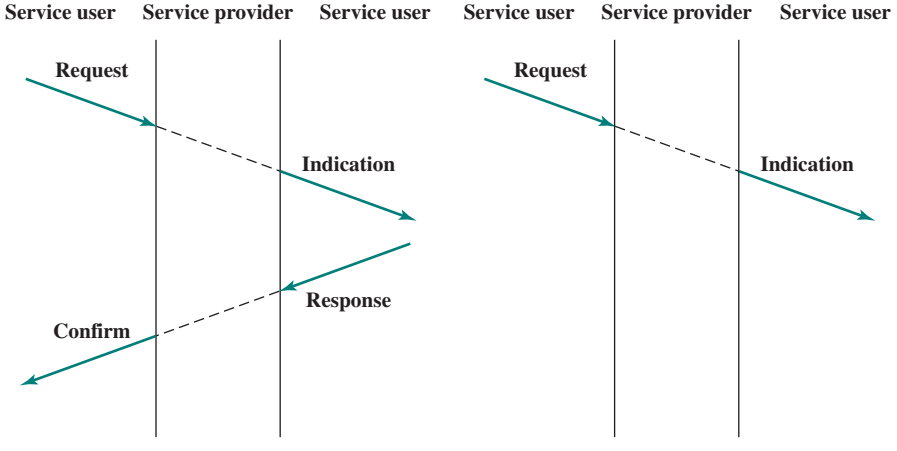


Figure 2.10 Time Sequence Diagrams for Service Primitives

This sequence of events is referred to as a *confirmed service*, as the initiator receives confirmation that the requested service has had the desired effect at the other end. If only request and indication primitives are involved (corresponding to steps 1 through 3), then the service dialog is a *nonconfirmed service*; the initiator receives no confirmation that the requested action has taken place (Figure 2.10b).

2.5 TRADITIONAL INTERNET-BASED APPLICATIONS

A number of applications have been standardized to operate on top of TCP. We mention three of the most common here.

The **Simple Mail Transfer Protocol (SMTP)** provides a basic electronic mail transport facility. It provides a mechanism for transferring messages among separate hosts. Features of SMTP include mailing lists, return receipts, and forwarding. SMTP does not specify the way in which messages are to be created; some local editing or native electronic mail facility is required. Once a message is created, SMTP accepts the message and makes use of TCP to send it to an SMTP module on another host. The target SMTP module will make use of a local electronic mail package to store the incoming message in a user's mailbox.

The **File Transfer Protocol (FTP)** is used to send files from one system to another under user command. Both text and binary files are accommodated, and the protocol provides features for controlling user access. When a user wishes to engage in file transfer, FTP sets up a TCP connection to the target system for the exchange of control messages. This connection allows user ID and password to be transmitted and allows the user to specify the file and file actions desired. Once a file transfer is approved, a second TCP connection is set up for the data transfer. The file is transferred over the data connection, without the overhead of any headers or control information at the application level. When the transfer is complete, the control connection is used to signal the completion and to accept new file transfer commands.

SSH (Secure Shell) provides a secure remote logon capability, which enables a user at a terminal or personal computer to log on to a remote computer and function as if directly connected to that computer. SSH also supports file transfer between the local host and a remote server. SSH enables the user and the remote server to authenticate each other; it also encrypts all traffic in both directions. SSH traffic is carried on a TCP connection.

2.6 MULTIMEDIA

With the increasing availability of broadband access to the Internet has come an increased interest in Web-based and Internet-based **multimedia** applications. The terms *multimedia* and *multimedia applications* are used rather loosely in the literature and in commercial publications, and no single definition of the term *multimedia* has been agreed. For our purposes, the definitions in Table 2.2 provide a starting point.

Table 2.2 Multimedia Terminology

Media
Refers to the form of information and includes text, still images, audio, and video.
Multimedia
Human–computer interaction involving text, graphics, voice, and video. Multimedia also refers to storage devices that are used to store multimedia content.
Streaming media
Refers to multimedia files, such as video clips and audio, that begin playing immediately or within seconds after it is received by a computer from the Internet or Web. Thus, the media content is consumed as it is delivered from the server rather than waiting until an entire file is downloaded.

One way to organize the concepts associated with multimedia is to look at a taxonomy that captures a number of dimensions of this field. Figure 2.11 looks at multimedia from the perspective of three different dimensions: type of media, applications, and the technology required to support the applications.

Media Types

Typically, the term *multimedia* refers to four distinct types of media: text, audio, graphics, and video.

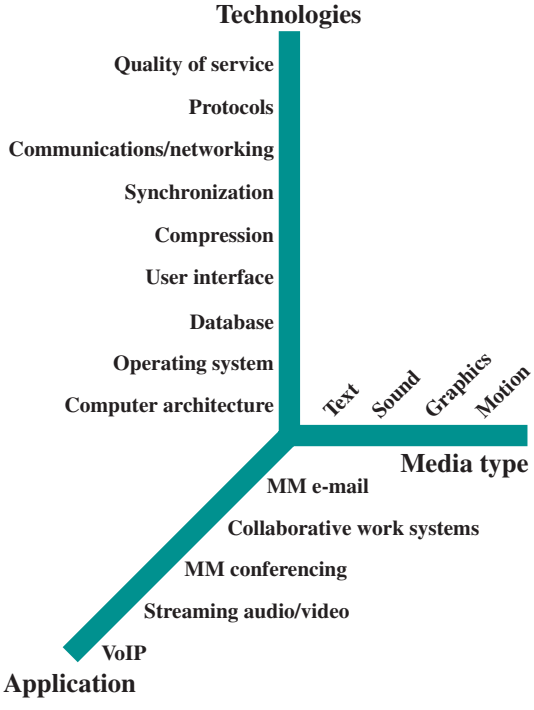


Figure 2.11 A Multimedia Taxonomy

From a communications perspective, the term **text** is self-explanatory, referring to information that can be entered via a keyboard and is directly readable and printable. Text messaging, instant messaging, and text (non-html) e-mail are common examples, as are chat rooms and message boards. However, the term often is used in the broader sense of data that can be stored in files and databases and that does not fit into the other three categories. For example, an organization's database may contain files of numerical data, in which the data are stored in a more compact form than printable characters.

The term **audio** generally encompasses two different ranges of sound. Voice, or speech, refers to sounds that are produced by the human speech mechanism. Generally, a modest bandwidth (under 4 kHz) is required to transmit voice. Telephony and related applications (e.g., voice mail, audio teleconferencing, and telemarketing) are the most common traditional applications of voice communications technology. A broader frequency spectrum is needed to support music applications, including the download of music files.

The **image** service supports the communication of individual pictures, charts, or drawings. Image-based applications include facsimile, computer-aided design (CAD), publishing, and medical imaging. Images can be represented in a vector graphics format, such as is used in drawing programs and PDF files. In a raster graphics format, an image is represented as a two-dimensional array of spots, called pixels.[2] The compressed JPG format is derived from a raster graphics format.

The **video** service carries sequences of pictures in time. In essence, video makes use of a sequence of raster-scan images.

Multimedia Applications

The Internet, until recently, has been dominated by information retrieval applications, e-mail, and file transfer, plus Web interfaces that emphasized text and images. Increasingly, the Internet is being used for multimedia applications that involve massive amounts of data for visualization and support of real-time interactivity. Streaming audio and video are perhaps the best known of such applications. An example of an interactive application is a virtual training environment involving distributed simulations and real-time user interaction [VIN98]. Some other examples are shown in Table 2.3.

Table 2.3 Domains of Multimedia Systems and Example Applications

Domain	Example Application
Information management	Hypermedia, multimedia-capable databases, content-based retrieval
Entertainment	Computer games, digital video, audio (MP3)
Telecommunication	Videoconferencing, shared workspaces, virtual communities
Information publishing/delivery	Online training, electronic books, streaming media

[2]A pixel, or picture element, is the smallest element of a digital image that can be assigned a gray level. Equivalently, a pixel is an individual dot in a dot-matrix representation of a picture.

[GONZ00] lists the following multimedia application domains:

- **Information systems:** These applications present information using multimedia. Examples include information kiosks, electronic books that include audio and video, and multimedia expert systems.
- **Communication systems:** These applications support collaborative work, such as videoconferencing.
- **Entertainment systems:** These applications include computer and network games and other forms of audiovisual entertainment.
- **Business systems:** These applications include business-oriented multimedia presentation, video brochures, and online shopping.
- **Educational systems:** These applications include electronic books with a multimedia component, simulation and modeling applets, and other teaching support systems.

One point worth noting is highlighted in Figure 2.11. Although traditionally the term *multimedia* has connoted the simultaneous use of multiple media types (e.g., video annotation of a text document), it has also come to refer to applications that require real-time processing or communication of video or audio alone. Thus, voice over IP (VoIP), streaming audio, and streaming video are considered multimedia applications even though each involves a single media type.

Multimedia Technologies

Figure 2.11 lists some of the technologies that are relevant to the support of multimedia applications. As can be seen, a wide range of technologies is involved. The lowest four items on the list are beyond the scope of this book. The other items represent only a partial list of communications and networking technologies for multimedia. These technologies and others are explored throughout the book. Here, we give a brief comment on each area.

- **Compression:** Digitized video, and to a much lesser extent audio, can generate an enormous amount of traffic on a network. A streaming application, which is delivered to many users, magnifies the traffic. Accordingly, standards have been developed for producing significant savings through compression. The most notable standards are JPG for still images and MPG for video.
- **Communications/networking:** This broad category refers to the transmission and networking technologies (e.g., SONET, ATM) that can support high-volume multimedia traffic.
- **Protocols:** A number of protocols are instrumental in supporting multimedia traffic. One example is the Real-time Transport Protocol (RTP), which is designed to support **inelastic traffic** (traffic that does not easily adapt, if at all, to changes in delay and throughput across an internet). RTP uses buffering and discarding strategies to assure that real-time traffic is received by the end user in a smooth continuous stream. Another example is the Session Initiation Protocol (SIP), an application-level control protocol for setting up, modifying, and terminating real-time sessions between participants over an IP data network.

- **Quality of service (QoS)**: The Internet and its underlying local area and wide area networks must include a QoS capability to provide differing levels of service to different types of application traffic. A QoS capability can deal with priority, delay constraints, delay variability constraints, and other similar requirements.

All of these matters are explored subsequently in this text.

2.7 SOCKETS PROGRAMMING

The concept of sockets and sockets programming was developed in the 1980s in the UNIX environment as the Berkeley Sockets Interface. In essence, a socket enables communication between a client and server process and may be either connection oriented or connectionless. A socket can be considered an end point in a communication. A client socket in one computer uses an address to call a server socket on another computer. Once the appropriate sockets are engaged, the two computers can exchange data.

Typically, computers with server sockets keep a TCP or UDP port open, ready for unscheduled incoming calls. The client typically determines the socket identification of the desired server by finding it in a Domain Name System (DNS) database. Once a connection is made, the server switches the dialogue to a different port number to free up the main port number for additional incoming calls.

Internet applications, such as TELNET and remote login (rlogin), make use of sockets, with the details hidden from the user. However, sockets can be constructed from within a program (in a language such as C, Java, or Python), enabling the programmer to easily support networking functions and applications. The sockets programming mechanism includes sufficient semantics to permit unrelated processes on different hosts to communicate.

The Berkeley Sockets Interface is the de facto standard **application programming interface (API)** for developing networking applications, spanning a wide range of operating systems. Windows Sockets (WinSock) is based on the Berkeley specification. The Sockets API provides generic access to interprocess communications services. Thus, the sockets capability is ideally suited for students to learn the principles of protocols and distributed applications by hands-on program development.

The Socket

Recall that each TCP and UDP header includes Source Port and Destination Port fields (Figure 2.6). These port values identify the respective users (applications) of the two TCP or UDP entities. Also, each IPv4 and IPv6 header includes Source Address and Destination Address fields (Figure 2.7); these **IP addresses** identify the respective host systems. The concatenation of a port value and an IP address forms a **socket**, which is unique throughout the Internet. Thus, in Figure 2.4, the combination of the IP address for host B and the port number for application X uniquely identifies the socket location of application X in host B. As the figure

indicates, an application may have multiple socket addresses, one for each port into the application.

The socket is used to define an API, which is a generic communication interface for writing programs that use TCP or UDP. In practice, when used as an API, a socket is identified by the triple (protocol, local address, local process). The local address is an IP address and the local process is a port number. Because port numbers are unique within a system, the port number implies the protocol (TCP or UDP). However, for clarity and ease of implementation, sockets used for an API include the protocol as well as the IP address and port number in defining a unique socket.

Corresponding to the two protocols, the Sockets API recognizes two types of sockets: stream sockets and datagram sockets. **Stream sockets** make use of TCP, which provides a connection-oriented reliable data transfer. Therefore, with stream sockets, all blocks of data sent between a pair of sockets are guaranteed for delivery and arrive in the order that they were sent. **Datagram sockets** make use of UDP, which does not provide the connection-oriented features of TCP. Therefore, with datagram sockets, delivery is not guaranteed, nor is order necessarily preserved.

There is a third type of socket provided by the Sockets API: raw sockets. **Raw sockets** allow direct access to lower-layer protocols, such as IP.

Sockets Interface Calls

This subsection summarizes the key system calls. Table 2.4 lists the core Socket functions.

SOCKETS SETUP The first step in using Sockets is to create a new socket using the `socket()` command. This command includes three parameters. The `domain` parameter refers to the area where the communicating processes exist. Commonly used domains include:

- AF_UNIX for communication between processes on one system;
- AF_INET for communication between processes using the IPv4 Internet Protocol;
- AF_INET6 for communication between processes using the IPv6 Internet Protocol.

`Type` specifies whether this is a stream or datagram socket, and `protocol` specifies either TCP or UDP. The reason that both `type` and `protocol` need to be specified is to allow additional transport-level protocols to be included in a future implementation. Thus, there might be more than one datagram-style transport protocol or more than one connection-oriented transport protocol. The `socket()` command returns an integer result that identifies this socket; it is similar to a UNIX file descriptor. The exact socket data structure depends on the implementation. It includes the source port and IP address and, if a connection is open or pending, the destination port and IP address and various options and parameters associated with the connection.

Table 2.4 Core Socket Functions

Format	Function	Parameters
`socket( )`	Initialize a socket	**domain** Protocol family of the socket to be created (AF_UNIX, AF_INET, AF_INET6) **type** Type of socket to be opened (stream, datagram, raw) **protocol** Protocol to be used on socket (UDP, TCP, ICMP)
`bind( )`	Bind a socket to a port address	**sockfd** Socket to be bound to the port address **localaddress** Socket address to which the socket is bound **addresslength** Length of the socket address structure
`listen( )`	Listen on a socket for inbound connections	**sockfd** Socket on which the application is to listen **queuesize** Number of inbound requests that can be queued at any time
`accept( )`	Accept an inbound connection	**sockfd** Socket on which the connection is to be accepted **remoteaddress** Remote socket address from which the connection was initiated **addresslength** Length of the socket address structure
`connect( )`	Connect outbound to a server	**sockfd** Socket on which the connection is to be opened **remoteaddress** Remote socket address to which the connection is to be opened **addresslength** Length of the socket address structure

`send( )`
`recv( )`
`read( )`
`write( )`
	Send and receive data on a stream socket (either send/recv or read/write can be used)	**sockfd** Socket across which the data will be sent or read
		data Data to be sent, or buffer into which the read data will be placed
		datalength Length of the data to be written, or amount of data to be read

`sendto( )` `recvfrom( )`	Send and receive data on a datagram socket	**sockfd** Socket across which the data will be sent or read **data** Data to be sent, or buffer into which the read data will be placed **datalength** Length of the data to be written, or amount of data to be read
`close( )`	Close a socket	**sockfd** Socket which is to be closed

After a socket is created, it must have an address to listen to. The `bind()` function binds a socket to a socket address. The address has the structure:

```
struct sockaddr_in {
  short int sin_family;         //Address family (TCP/IP)
  unsigned short int sin_port;  //Port number
  struct in_addr sin_addr;      //Internet address
  unsigned char sin_zero[8];    //Padding to make structure same
                                //size as struct sockaddr
};
```

SOCKETS CONNECTION For a stream socket, once the socket is created, a connection must be set up to a remote socket. One side functions as a client and requests a connection to the other side, which acts as a server.

The server side of a connection setup requires two steps. First, a server application issues a `listen()`, indicating that the given socket is ready to accept incoming connections. The parameter *backlog* is the number of connections allowed on the incoming queue. Each incoming connection is placed in this queue until a matching `accept()` is issued by the server side. Next, the `accept()` call is used to remove one request from the queue. If the queue is empty, the `accept()` blocks the process until a connection request arrives. If there is a waiting call, then `accept()` returns a new file descriptor for the connection. This creates a new socket, which has the IP address and port number of the remote party, the IP address of this system, and a new port number. The reason that a new socket with a new port number is assigned is that this enables the local application to continue to listen for more requests. As a result, an application may have multiple connections active at any time, each with a different local port number. This new port number is returned across the TCP connection to the requesting system.

A client application issues a `connect()` that specifies both a local socket and the address of a remote socket. If the connection attempt is unsuccessful `connect()` returns the value –1. If the attempt is successful, `connect()` returns a 0 and fills in the file descriptor parameter to include the IP address and port number of the local and foreign sockets. Recall that the remote port number may differ from that specified in the `foreignAddress` parameter because the port number is changed on the remote host.

Once a connection is set up, `getpeername()` can be used to find out who is on the other end of the connected stream socket. The function returns a value in the `sockfd` parameter.

SOCKETS COMMUNICATION For **stream communication**, the functions `send()` and `recv()` are used to send or receive data over the connection identified by the `sockfd` parameter. In the `send()` call, the `*msg` parameter points to the block of data to be sent and the `len` parameter specifies the number of bytes to be sent. The `flags` parameter contains control flags, typically set to 0. The `send()` call returns the number of bytes sent, which may be less than the number specified in the `len` parameter. In the `recv()` call, the `*buf` parameter points to the buffer for storing incoming data, with an upper limit on the number of bytes set by the `len` parameter.

At any time, either side can close the connection with the `close()` call, which prevents further sends and receives. The `shutdown()` call allows the caller to terminate sending or receiving or both.

Figure 2.12 shows the interaction of the clients and server sides in setting up, using, and terminating a connection.

For **datagram communication**, the functions `sendto()` and `recvfrom()` are used. The `sendto()` call includes all the parameters of the `send()` call plus a specification of the destination address (IP address and port). Similarly, the `recvfrom()` call includes an address parameter, which is filled in when data are received.

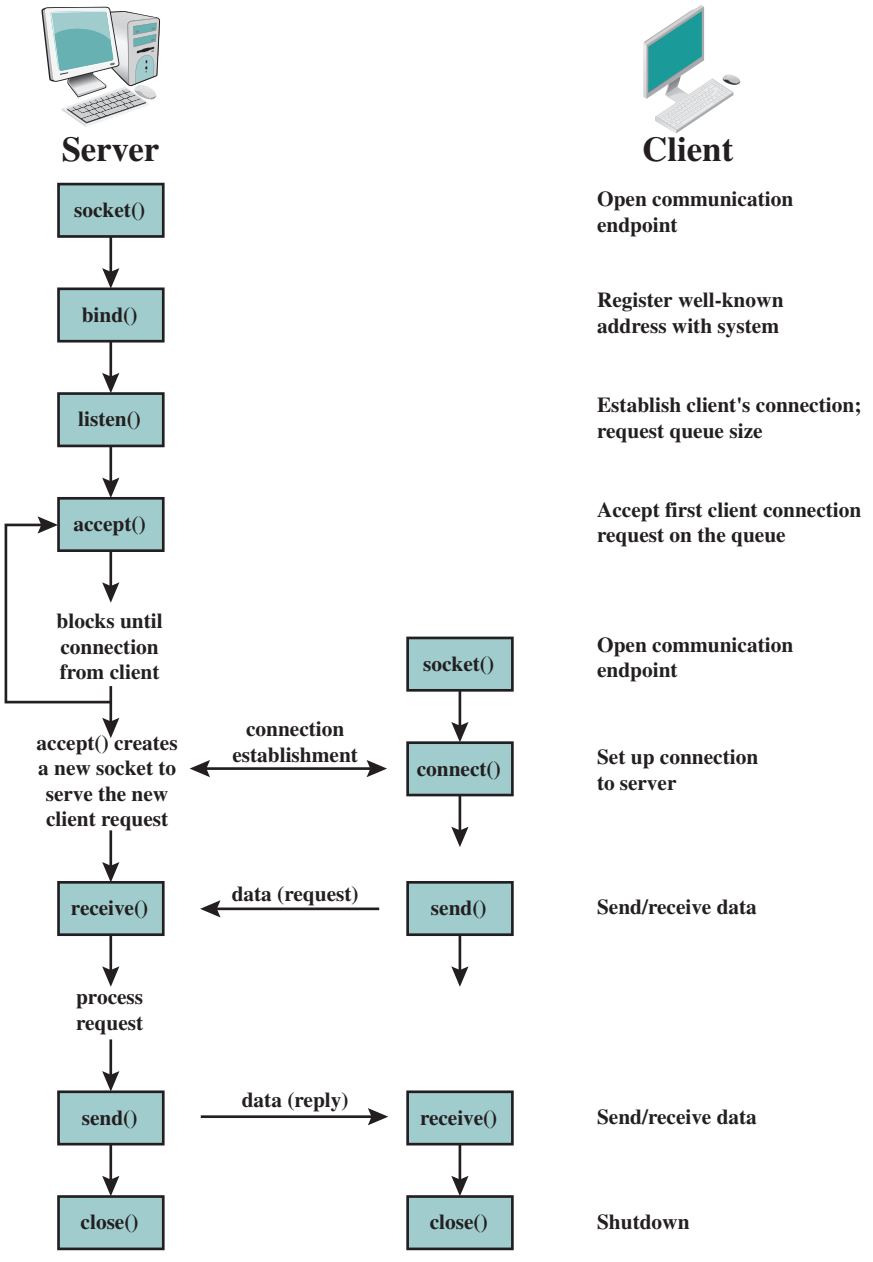


Figure 2.12 Socket System Calls for Connection-Oriented Protocol

Example

In this section, we give an example of a simple client and server implemented in C, and communicating using stream sockets over the Internet. The two programs are shown in Figures 2.13 and 2.14. Before reading the following discussion, the reader is advised to compile and execute the two programs to understand their operation.[3]

```
1  #include <stdio.h>
2  #include <sys/types.h>
3  #include <sys/socket.h>
4  #include <netinet/in.h>

5  void error(char *msg)
6  {
7      perror(msg);
8      exit(1);
9  }

10 int main(int argc, char *argv[])
11 {
12      int sockfd, newsockfd, portno, clilen;
13      char buffer[256];
14      struct sockaddr_in serv_addr, cli_addr;
15      int n;
16      if (argc < 2) {
17          fprintf(stderr,"ERROR, no port provided\n");
18          exit(1);
19      }
20      sockfd = socket(AF_INET, SOCK_STREAM, 0);
21      if (sockfd < 0)
22         error("ERROR opening socket");
23      bzero((char *) &serv_addr, sizeof(serv_addr));
24      portno = atoi(argv[1]);
25      serv_addr.sin_family = AF_INET;
26      serv_addr.sin_port = htons(portno);
27      serv_addr.sin_addr.s_addr = INADDR_ANY;
28      if (bind(sockfd, (struct sockaddr *) &serv_addr,
29               sizeof(serv_addr)) < 0)
30               error("ERROR on binding");
31      listen(sockfd,5);
32      clilen = sizeof(cli_addr);
33      newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
34      if (newsockfd < 0)
35           error("ERROR on accept");
36      bzero(buffer,256);
37      n = read(newsockfd,buffer,255);
38      if (n < 0) error("ERROR reading from socket");
39      printf("Here is the message: %s\n",buffer);
40      n = write(newsockfd,"I got your message",18);
41      if (n < 0) error("ERROR writing to socket");
42      return 0;
43 }
```

Figure 2.13 Sockets Server

[3]The two programs, without the line numbers, are available at box.com/dcc10e.

```
1  #include <stdio.h>
2  #include <sys/types.h>
3  #include <sys/socket.h>
4  #include <netinet/in.h>
5  #include <netdb.h>

6  void error(char *msg)
7  {
8      perror(msg);
9      exit(0);
10 }

11 int main(int argc, char *argv[])
12 {
13   int sockfd, portno, n;
14   struct sockaddr_in serv_addr;
15   struct hostent *server;
16   char buffer[256];
17   if (argc < 3) {
18      fprintf(stderr,"usage %s hostname port\n", argv[0]);
19      exit(0);
20   }
21   portno = atoi(argv[2]);
22   sockfd = socket(AF_INET, SOCK_STREAM, 0);
23   if (sockfd < 0)
24       error("ERROR opening socket");
25   server = gethostbyname(argv[1]);
26   if (server == NULL) {
27       fprintf(stderr,"ERROR, no such host\n");
28       exit(0);
29   }
30   bzero((char *) &serv_addr, sizeof(serv_addr));
31   serv_addr.sin_family = AF_INET;
32   bcopy((char *)server->h_addr,
33        (char *)&serv_addr.sin_addr.s_addr,
34        server->h_length);
35   serv_addr.sin_port = htons(portno);
36   if (connect(sockfd,(struct sockaddr *)&serv_addr,sizeof(serv_addr)) < 0)
37       error("ERROR connecting");
38   printf("Please enter the message: ");
39   bzero(buffer,256);
40   fgets(buffer,255,stdin);
41   n = write(sockfd,buffer,strlen(buffer));
42   if (n < 0)
43        error("ERROR writing to socket");
44   bzero(buffer,256);
45   n = read(sockfd,buffer,255);
46   if (n < 0)
47        error("ERROR reading from socket");
48   printf("%s\n",buffer);
49   return 0;
50 }
```

Figure 2.14 Sockets Client

1. Download the client and server programs into files called `server.c` and `client.c` and compile them into two executables called `server` and `client`. The commands would look something like this:

```
gcc server.c -o server
gcc client.c -o client
```

Ideally, you should run the client and server on separate hosts on the Internet. You can, if necessary, run the server in one window and the client in another on the same machine.

2. Start the server first by issuing a command to the server with a port number as the argument. This is the port on which the server will listen. Choose a number between 2000 and 65535. If the port is in use, the server will return a message. In that case, pick another number and try again. A typical command line is the following:

```
server 62828
```

3. Issue a command to start the client, with two arguments: the name of the host on which the server is running and the port number on which the server is listening. So, if the server is on host X, the command line would be:

```
client X 62828
```

If client and server are on the same machine, then the first argument is *localhost*.

4. The client will prompt you to enter a message. Subsequently, if there are no errors, the server will display the message on *stdout*, send an acknowledgment message to the client and terminate. The client then prints the acknowledgment message and terminates.

SERVER PROGRAM Now that you see what the two programs do, we can examine the code, starting with the server (Figure 2.13). Lines 1–4 define header files: `stdio.h` contains declarations used in most input and output operations; `types.h` defines a number of data types used in system calls; `socket.h` defines a number of structures needed for sockets; and `netinet/in.h` contains constants and structures needed for Internet domain addresses.

When a system call fails, the error program `void`, which contains the function `perror`, displays an error message on `stderr` and then aborts the program (lines 5–9).

Line 10 begins the definition of the main program. The first two integer variables, `sockfd` and `newsockfd`, are array subscripts into the file descriptor table. They store the values returned by the socket system call and the accept system call. `portno` stores the port number on which the server accepts connections. `clilen` stores the size of the address of the client, which is needed for the accept system call. The server reads characters from the socket connection into the buffer `char`.

Line 14 defines the client and server address structures, using the `sockaddr_in` Internet address structure. This structure is defined in `netinet/in.h`. The variable `n` designates the number of characters read or written by the `read()` and `write()` calls.

Lines 16–19 check that the user has provided a port number argument and displays an error message if the argument is not present.

Lines 20–22 deal with creating a new socket. The first parameter of the `socket()` call specifies the IPv4 domain. The second parameter specifies that a stream socket is requested. The third parameter, which defines the protocol to

be used, is set to zero. The zero value indicates that the default protocol should be used, which is TCP for a stream socket. The `socket()` call returns an entry into the file descriptor table, which defines the socket. If the call fails, it returns the value –1. The `if` statement in the code then displays the error message.

The function `bzero()` sets all values in a buffer to zero (line 23). It takes two arguments, the first is a pointer to the buffer and the second is the size of the buffer. Thus, this line initializes `serv_addr` to zeros.

Line 24 retrieves the port number that was supplied as an argument to the server program. This statement uses the `atoi()` function to convert the parameter from a string of digits to an integer stored in `portno`.

Lines 25–27 assign values to the variable `serv_addr`, which is a structure of type `struct sockaddr_in`. As shown earlier in this section, `sockaddr_in` has four fields, of which the first three must be assigned values. `serv_addr.sin_family` is set to `AF_INET`, for IPv4 communication. `serv_addr.sin_port` is derived from the port number argument. However, instead of simply copying the port number to this field, it is necessary to convert this to network byte order using the function `htons()`, which converts a port number in host byte order to a port number in network byte order. The field `serv_addr.sin_addr.s_addr` is assigned the Internet IPv4 address of the server, which is obtained from the symbolic constant `INADDR_ANY`.

Lines 28–30 include the `bind()` function, which, as mentioned earlier, binds a socket to a socket address. Its three arguments are the socket file descriptor, the address to which the socket is bound, and the size of the address. The second argument is a pointer to a structure of type `sockaddr`, but what is passed in is a structure of type `sockaddr_in`, and so this must be cast to the correct type. This can fail for a number of reasons, the most obvious being that this socket is already in use on this machine.

If the `bind()` operation succeeds, the server is now ready to listen on this socket. The `listen()` on line 31 takes as arguments the file descriptor for the socket and the size of the backlog queue, which is the number of connections that can be waiting while the process is handling a particular connection. Typically, the queue size is set to 5.

Lines 32–35 deal with the `accept()` system call, which causes the process to block until a client connects to the server. Thus, it wakes up the process when a connection from a client has been successfully established. It returns a new file descriptor, and all communication on this connection should be done using the new file descriptor. The second argument is a reference pointer to the address of the client on the other end of the connection, and the third argument is the size of this structure.

After a client has successfully connected to the server, lines 36–39 are executed. The variable buffer is set to all zeroes. Then, the `read()` function is invoked to read up to 255 bytes into buffer. The `read` call uses the new file descriptor, which was returned by `accept()`, not the original file descriptor returned by `socket()`. In addition to filling the buffer, `read()` returns the number of character read. Note also that the `read()` will block until there is something for it to read in the socket, that is, after the client has executed a `write()`. The read operation concludes when the server displays the message received over the connection.

After a successful read, the server writes a message that will be delivered over the socket connection to the client (lines 40–41). The parameters for the `write()` function are the socket file descriptor, the message, and a character count of the message.

Lines 42–43 terminate `main` and thus the program. Since `main` was declared to be of type `int`, many compilers complain if it does not return anything.

CLIENT PROGRAM The client program is shown in Figure 2.14. Lines 1–4 are the same header files as for the server. Line 5 adds the header file `netdb.h`, which defines the structure `hostent`, which is used in the client program.

In lines 6–14, the `error()` function is the same as for the server, as are the variables `sockfd`, `portno`, and `n`. The variable `serv_addr` will be assigned the address of the server to which the client wants to connect.

Line 15 defines the variable server as a pointer to a structure of type `hostent`, defined in `netdb.h`. The structure includes the following fields: `*h_name`, the name of the host; `**h_aliases`, a list of alternate names for the host; `h_addrtype`, currently always `AF_INET`; `h_length`, the length in bytes of the address; and `**h_addr_list`, a pointer to a list of network addresses for the named host.

Lines 16–24 are almost the same as that in the server.

In lines 25–29, the client attempts to get the `hostent` structure for the server. `argv[1]` is the first argument in the call to the client program and contains the name of the desired server host. The function `*gethostbyname(char, *name)` is defined in `netdb.h` as `struct hostent`. It takes a name as an argument and returns a pointer to a `hostent` structure for the named host. If the name is not known locally, the client machine uses the Domain Name System, described in a subsequent chapter.

Lines 30–35 set the fields in `serv_addr`, similar to what is done in the server program. However, because the field `server->h_addr` is a character string, we use the function `void bcopy(char *s1, char *s2, int length)`, which copies length bytes from `s1` to `s2`.

The client is now ready to make a request for a connection to the server, on lines 36–37. The function `connect()` takes three arguments: the client's socket file descriptor, the address of the requested host, and the size of this address. The function returns 0 on success and –1 on failure.

The remaining code, lines 38–50, should be fairly clear. It prompts the user to enter a message, uses `fgets` to read the message from `stdin`, writes the message to the socket, reads the reply from the socket, and displays this reply.

2.8 RECOMMENDED READING AND ANIMATION

For the reader interested in greater detail on TCP/IP, there are two three-volume works that are more than adequate. The works by Comer and Stevens have become classics and are considered definitive [COME14, COME99, COME01]. The works by Stevens, Wright, and Fall are equally worthwhile and more detailed with respect to protocol operation [FALL12, STEV96, WRIG95]. A more compact and very useful reference work is [PARZ06], which covers the spectrum of TCP/IP-related

protocols in a technically concise but thorough fashion, including coverage of some protocols not found in the other two works.

[GREE80] is a good tutorial overview of the concept of a layered protocol architecture. Two early papers that provide good discussions of the design philosophy of the TCP/IP protocol suite are [LEIN85] and [CLAR88].

Although somewhat dated, [FURH94] remains a good overview of multimedia topics. [VOGE95] is a good introduction to QoS considerations for multimedia. [HELL01] is a lengthy and worthwhile theoretical treatment of multimedia. An excellent concise introduction to using sockets is [DONA01]; another good overview is [HALL01].

CLAR88 Clark, D. "The Design Philosophy of the DARPA Internet Protocols." *ACM SIGCOMM Computer Communications Review*, August 1988.

COME99 Comer, D., and Stevens, D. *Internetworking with TCP/IP, Volume II: Design Implementation, and Internals.* Upper Saddle River, NJ: Prentice Hall, 1999.

COME01 Comer, D., and Stevens, D. *Internetworking with TCP/IP, Volume III: Client-Server Programming and Applications.* Upper Saddle River, NJ: Prentice Hall, 2001.

COME14 Comer, D. *Internetworking with TCP/IP, Volume I: Principles, Protocols, and Architecture.* Upper Saddle River, NJ: Prentice Hall, 2014.

DONA01 Donahoo, M., and Clavert, K. *The Pocket Guide to TCP/IP Sockets.* San Francisco, CA: Morgan Kaufmann, 2001.

FALL12 Fall, K., and Stevens, W. *TCP/IP Illustrated, Volume 1: The Protocols.* Reading, MA: Addison-Wesley, 2012.

FURH94 Furht, B. "Multimedia Systems: An Overview." *IEEE Multimedia*, Spring 1994.

GREE80 Green, P. "An Introduction to Network Architecture and Protocols." *IEEE Transactions on Communications*, April 1980.

HALL01 Hall, B. *Beej's Guide to Network Programming Using Internet Sockets.* 2001. http://beej.us/guide/bgnet.

HELL01 Heller, R., et al. "Using a Theoretical Multimedia Taxonomy Framework." *ACM Journal of Educational Resources in Computing*, Spring 2001.

LEIN85 Leiner, B.; Cole, R.; Postel, J.; and Mills, D. "The DARPA Internet Protocol Suite." *IEEE Communications Magazine*, March 1985.

PARZ06 Parziale, L., et al. *TCP/IP Tutorial and Technical Overview.* IBM Redbook GG24-3376-07, 2006, http://www.redbooks.ibm.com/abstracts/gg243376.html

STEV96 Stevens, W. *TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP, and the UNIX(R) Domain Protocol.* Reading, MA: Addison-Wesley, 1996.

VIN98 Vin, H. "Supporting Next-Generation Distributed Applications." *IEEE Multimedia*, July–September 1998,

VOGE95 Vogel, A., et al. "Distributed Multimedia and QoS: A Survey." *IEEE Multimedia*, Summer 1995.

WRIG95 Wright, G., and Stevens, W. *TCP/IP Illustrated, Volume 2: The Implementation.* Reading, MA: Addison-Wesley, 1995.

Animations

Animations that illustrate protocol encapsulation, data flow through a protocol stack, and TFTP are available at the Premium Web site. The reader is encouraged to view these animations to reinforce concepts from this chapter.

2.9 KEY TERMS, REVIEW QUESTIONS, AND PROBLEMS

Key Terms

application programming interface (API)	IP datagram	service access point (SAP)
application layer	multimedia	Simple Mail Transfer Protocol (SMTP)
audio	network access protocol (NAP)	socket
checksum	network access layer	socket programming
datagram communication	Open Systems Interconnection (OSI)	SSH (Secure Shell)
datagram sockets	packet	stream communication
encapsulation	peer layer	stream sockets
File Transfer Protocol (FTP)	physical layer	subnetwork
frame check sequence	port	syntax
header	protocol	TCP segment
image	protocol architecture	text
host-to-host layer	protocol data unit (PDU)	timing
inelastic traffic	quality of service (QoS)	Transmission Control Protocol (TCP)
Internet	raw sockets	transport layer
Internet layer	router	User Datagram Protocol (UDP)
Internet Protocol (IP)	segments	video
Internetworking	semantics	

Review Questions

2.1. What is the major function of the network access layer?
2.2. What tasks are performed by the transport layer?
2.3. What is a protocol?
2.4. What is a protocol data unit (PDU)?
2.5. What is a protocol architecture?
2.6. What is TCP/IP?
2.7. What are some advantages to layering as seen in the TCP/IP architecture?
2.8. What is a router?

2.9. Which version of IP is the most prevalent today?

2.10. Does all traffic running on the Internet use TCP?

2.11. Compare the address space between IPv4 and IPv6. How many bits are used in each?

Problems

2.1. Using the layer models in Figure 2.15, describe the ordering and delivery of a pizza, indicating the interactions at each level.

2.2. **a.** The French and Chinese prime ministers need to come to an agreement by telephone, but neither speaks the other's language. Further, neither has on hand a translator that can translate to the language of the other. However, both prime ministers have English translators on their staffs. Draw a diagram similar to Figure 2.15 to depict the situation, and describe the interaction at each level.

b. Now suppose that the Chinese prime minister's translator can translate only into Japanese and that the French prime minister has a German translator available. A translator between German and Japanese is available in Germany. Draw a new diagram that reflects this arrangement and describe the hypothetical phone conversation.

2.3. List the major disadvantages with the layered approach to protocols.

2.4. Two blue armies are each poised on opposite hills preparing to attack a single red army in the valley. The red army can defeat either of the blue armies separately but will fail to defeat both blue armies if they attack simultaneously. The blue armies communicate via an unreliable communications system (a foot soldier). The commander with one of the blue armies would like to attack at noon. His problem is this: If he

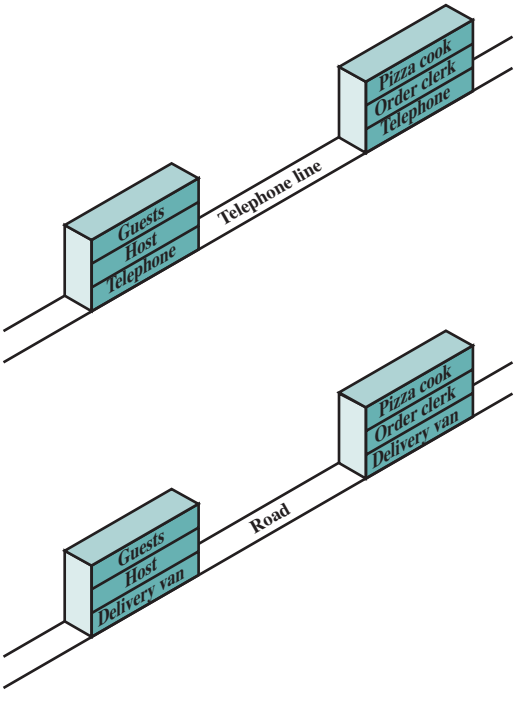


Figure 2.15 Architecture for Problem 2.1

sends a message to the other blue army, ordering the attack, he cannot be sure it will get through. He could ask for acknowledgment, but that might not get through. Is there a protocol that the two blue armies can use to avoid defeat?

2.5. A broadcast network is one in which a transmission from any one attached station is received by all other attached stations over a shared medium. Examples are a bus-topology local area network, such as Ethernet, and a wireless radio network. Discuss the need or lack of need for a network layer (OSI layer 3) in a broadcast network.

2.6. In Figure 2.5, exactly one protocol data unit (PDU) in layer N is encapsulated in a PDU at layer $(N - 1)$. It is also possible to break one N-level PDU into multiple $(N - 1)$-level PDUs (segmentation) or to group multiple N-level PDUs into one $(N - 1)$-level PDU (blocking).

a. In the case of segmentation, is it necessary that each $(N - 1)$-level segment contain a copy of the N-level header?

b. In the case of blocking, is it necessary that each N-level PDU retain its own header, or can the data be consolidated into a single N-level PDU with a single N-level header?

2.7. Suppose a protocol architecture has defined four layers for communication, say, base layer, net layer, transport layer, and max layer. The header length of the base layer and of the net layer is 2 bytes; the transport layer, 3 bytes; and max layer, 4 bytes. Besides this, each layer has a 4-bit end marker. If a 50-byte packet is received, what is the size of the actual message?

2.8. Can you identify some areas where UDP may be preferable to TCP?

2.9. IP, TCP, and UDP all discard a packet that arrives with a checksum error and do not attempt to notify the source. Why?

2.10. Why does the TCP header have a header length field while the UDP header does not?

2.11. The previous version of the TFTP specification, RFC 783, included the following statement:

> All packets other than those used for termination are acknowledged individually unless a timeout occurs.

The RFC 1350 specification revises this to say:

> All packets other than duplicate ACK's and those used for termination are acknowledged unless a timeout occurs.

The change was made to fix a problem referred to as the "Sorcerer's Apprentice." Deduce and explain the problem.

2.12. What is the limiting factor in the time required to transfer a file using TFTP?

2.13. Data comprising 5000 bytes needs to be transmitted. How many TFTP packets would be required? For transmission, each TFTP packet is handed to UDP, which adds an 8-byte header to form a UDP segment, which is in turn handed to IP, which adds a 20-byte header file to form an IP datagram. What is the size of this IP datagram? What is the total size of all the packets actually being transmitted? Find the overhead as a ratio of the size of headers to the size of the data file.

2.14. The TFTP specification (RFC 1350) states that the transfer identifiers (TIDs) for a connection should be randomly chosen, so that the probability that the same number is chosen twice in immediate succession is very low. What would be the problem of using the same TIDs twice in immediate succession?

2.15. In order to be able to retransmit lost packets, TFTP must keep a copy of the data it sends. How many packets of data must TFTP keep at a time to implement this retransmission mechanism?

2.16. TFTP, like most protocols, will never send an error packet in response to an error packet it receives. Why?

2.17. We have seen that in order to deal with lost packets, TFTP implements a timeout-and-retransmit scheme, by setting a retransmission timer when it transmits a packet to the remote host. Most TFTP implementations set this timer to a fixed value of about 5 seconds. Discuss the advantages and the disadvantages of using a fixed value for the retransmission timer.

2.18. TFTP's timeout-and-retransmission scheme implies that all data packets will eventually be received by the destination host. Will these data also be received uncorrupted? Why or why not?

2.19. This problem concerns material in Appendix E. Based on the principles enunciated in Table E.1.

a. Design an architecture with eight layers and make a case for it.

b. Design one with six layers and make a case for that.

2.10 SOCKETS PROGRAMMING ASSIGNMENTS

2.1. Determining a local machine's IP address is useful when you are handling network communication tasks. Write code to find the IP address of a local machine. Hint: You may use some public DNS, for example, Google public DNS is 8.8.8.8.

2.2. Write a sockets program to get a host name for a given IP address.

2.3. How to broadcast a message on the Internet? Two questions need to be answered: What address should be used as the broadcast address. How to send data to the broadcast address? A broadcast address is the subnet's network number with all one-bits set for the host portion of the address. For instance, if a network IP address is 192.168.1.0, and the netmask is 255.255.255.0, the last byte of the address is the host number (because the first three bytes, according to the netmask, correspond to the network number). So the broadcast address is 192.168.1.255. Under Unix, the `ifconfig` command will actually give you all this information.

a. Determine the broadcast address of your local machine;

b. Send a broadcast packet to your broadcast address. Write a code to implement this task.

2.4. Write a stream-based echo server and a client sending messages to it, and receiving back each message in turn. Hint: Modify the stream-based TCP client and server programs in this chapter or similar programs to transfer multiple messages back and forth (until the client terminates the connection).

2.5. Modify the server program from Exercise 2.4 to set the TCP window size for the server socket. Hint: Set the SO_RCVBUF size through a call to `setsockopt()`. Be aware that setting the TCP window size in this way would not guarantee the size for the entire life of the socket because of the inherent TCP window flow control. This exercise solely intends to demonstrate the use of `setsockopt()` and `getsockopt()` calls.

2.6. Use `poll()` function and socket programming to receive out-of-band data. Out-of-band data is also called urgent data in TCP, and is often received with first priority via a separate data stream. Hint: Check events and select POLLPRI.

APPENDIX 2A THE TRIVIAL FILE TRANSFER PROTOCOL

This appendix provides an overview of the Internet standard Trivial File Transfer Protocol (TFTP), defined in RFC 1350. Our purpose is to give the reader some flavor for the elements of a protocol. TFTP is simple enough to provide a concise example, but includes most of the significant elements found in other, more complex, protocols.

Introduction to TFTP

TFTP is far simpler than the Internet standard FTP (RFC 959). There are no provisions for access control or user identification, so TFTP is only suitable for public access file directories. Because of its simplicity, TFTP is easily and compactly implemented. For example, some diskless devices use TFTP to download their firmware at boot time.

TFTP runs on top of UDP. The TFTP entity that initiates the transfer does so by sending a read or write request in a UDP segment with a destination port of 69 to the target system. This port is recognized by the target UDP module as the identifier of the TFTP module. For the duration of the transfer, each side uses a transfer identifier (TID) as its port number.

TFTP Packets

TFTP entities exchange commands, responses, and file data in the form of packets, each of which is carried in the body of a UDP segment. TFTP supports five types of packets (Figure 2.16); the first two bytes contain an opcode that identifies the packet type:

- **RRQ:** The read request packet requests permission to transfer a file from the other system. The packet includes a file name, which is a sequence of ASCII[4] bytes terminated by a zero byte. The zero byte is the means by which the receiving TFTP entity knows when the file name is terminated. The packet also includes a mode field, which indicates whether the data file is to be interpreted as a string of ASCII bytes (netascii mode) or as raw 8-bit bytes (octet mode) of data. In netascii mode, the file is transferred as lines of characters, each terminated by a carriage return, line feed. Each system must translate between its own format for character files and the TFTP format.

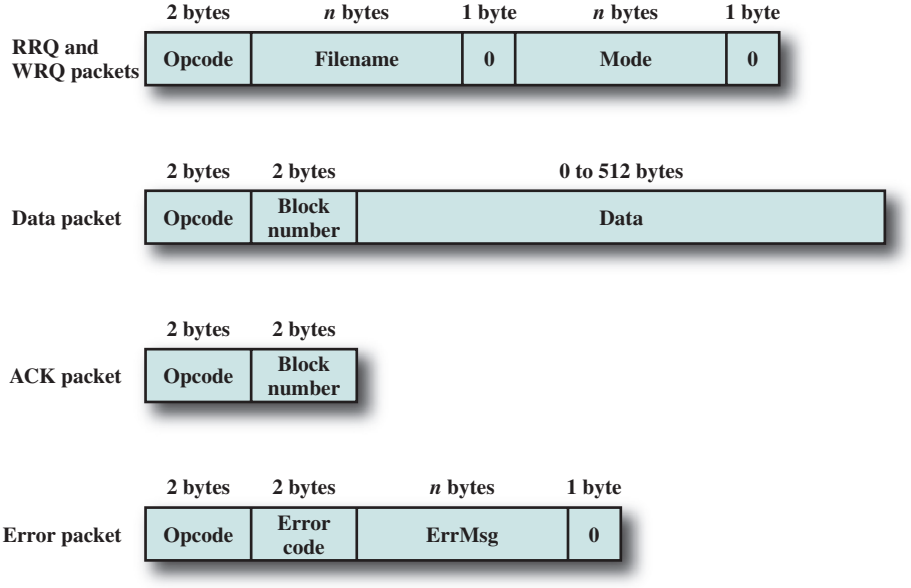


Figure 2.16 TFTP Packet Formats

[4]ASCII is the American Standard Code for Information Interchange, a standard of the American National Standards Institute. It designates a unique 7-bit pattern for each letter, with an eighth bit used for parity. ASCII is equivalent to the International Reference Alphabet (IRA), defined in ITU-T Recommendation T.50. See Appendix F for a discussion.