FIFTH EDITION

COMPUTER NETWORKS

TANENBAUM | WETHERALL

This page intentionally left blank

COMPUTER NETWORKS

FIFTH EDITION

This page intentionally left blank

COMPUTER NETWORKS

FIFTH EDITION

ANDREW S. TANENBAUM

Vrije Universiteit
Amsterdam, The Netherlands

DAVID J. WETHERALL

University of Washington
Seattle, WA

PRENTICE HALL

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

Editorial Director: Marcia Horton
Editor-in-Chief: Michael Hirsch
Executive Editor: Tracy Dunkelberger
Assistant Editor: Melinda Haggerty
Editorial Assistant: Allison Michael
Vice President, Marketing: Patrice Jones
Marketing Manager: Yezan Alayan
Marketing Coordinator: Kathryn Ferranti
Vice President, Production: Vince O'Brien
Managing Editor: Jeff Holcomb
Senior Operations Supervisor: Alan Fischer
Manufacturing Buyer: Lisa McDowell
Cover Direction: Andrew S. Tanenbaum, David J. Wetherall, Tracy Dunkelberger
Art Director: Linda Knowles
Cover Designer: Susan Paradise
Cover Illustration: Jason Consalvo
Interior Design: Andrew S. Tanenbaum
AV Production Project Manager: Gregory L. Dulles
Interior Illustrations: Laserwords, Inc.
Media Editor: Daniel Sandin
Composition: Andrew S. Tanenbaum
Copyeditor: Rachel Head
Proofreader: Joe Ruddick
Printer/Binder: Courier/Westford
Cover Printer: Lehigh-Phoenix Color/ Hagerstown

Credits and acknowledgments borrowed from other sources and reproduced, with permission, in this textbook appear on appropriate page within text.

Library of Congress Cataloging-in-Publication Data

Tanenbaum, Andrew S., 1944-
Computer networks / Andrew S. Tanenbaum, David J. Wetherall. -- 5th ed.
p. cm.
Includes bibliographical references and index.
ISBN-13: 978-0-13-212695-3 (alk. paper)
ISBN-10: 0-13-212695-8 (alk. paper)
1. Computer networks. I. Wetherall, D. (David) II. Title.
TK5105.5.T36 2011
004.6--dc22

2010034366

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

To Suzanne, Barbara, Daniel, Aron, Marvin, Matilde, and the memory of Bram, and Sweetie π (AST)

To Katrin, Lucy, and Pepper (DJW)

This page intentionally left blank

CONTENTS

This page intentionally left blank

PREFACE

This book is now in its fifth edition. Each edition has corresponded to a different phase in the way computer networks were used. When the first edition appeared in 1980, networks were an academic curiosity. When the second edition appeared in 1988, networks were used by universities and large businesses. When the third edition appeared in 1996, computer networks, especially the Internet, had become a daily reality for millions of people. By the fourth edition, in 2003, wireless networks and mobile computers had become commonplace for accessing the Web and the Internet. Now, in the fifth edition, networks are about content distribution (especially videos using CDNs and peer-to-peer networks) and mobile phones are small computers on the Internet.

New in the Fifth Edition

Among the many changes in this book, the most important one is the addition of Prof. David J. Wetherall as a co-author. David brings a rich background in networking, having cut his teeth designing metropolitan-area networks more than 20 years ago. He has worked with the Internet and wireless networks ever since and is a professor at the University of Washington, where he has been teaching and doing research on computer networks and related topics for the past decade.

Of course, the book also has many changes to keep up with the: ever-changing world of computer networks. Among these are revised and new material on

- Wireless networks (802.12 and 802.16)
- The 3G networks used by smart phones
- RFID and sensor networks
- Content distribution using CDNs
- Peer-to-peer networks
- Real-time media (from stored, streaming, and live sources)
- Internet telephony (voice over IP)
- Delay-tolerant networks

A more detailed chapter-by-chapter list follows.

Chapter 1 has the same introductory function as in the fourth edition, but the contents have been revised and brought up to date. The Internet, mobile phone networks, 802.11, and RFID and sensor networks are discussed as examples of computer networks. Material on the original Ethernet—with its vampire taps—has been removed, along with the material on ATM.

Chapter 2, which covers the physical layer, has expanded coverage of digital modulation (including OFDM as widely used in wireless networks) and 3G networks (based on CDMA). New technologies are discussed, including Fiber to the Home and power-line networking.

Chapter 3, on point-to-point links, has been improved in two ways. The material on codes for error detection and correction has been updated, and also includes a brief description of the modern codes that are important in practice (e.g., convolutional and LDPC codes). The examples of protocols now use Packet over SONET and ADSL. Sadly, the material on protocol verification has been removed as it is little used.

In Chapter 4, on the MAC sublayer, the principles are timeless but the technologies have changed. Sections on the example networks have been redone accordingly, including gigabit Ethernet, 802.11, 802.16, Bluetooth, and RFID. Also updated is the coverage of LAN switching, including VLANs.

Chapter 5, on the network layer, covers the same ground as in the fourth edition. The revisions have been to update material and add depth, particularly for quality of service (relevant for real-time media) and internetworking. The sections on BGP, OSPF and CIDR have been expanded, as has the treatment of multicast routing. Anycast routing is now included.

Chapter 6, on the transport layer, has had material added, revised, and removed. New material describes delay-tolerant networking and congestion control in general. The revised material updates and expands the coverage of TCP congestion control. The material removed described connection-oriented network layers, something rarely seen any more.

Chapter 7, on applications, has also been updated and enlarged. While material on DNS and email is similar to that in the fourth edition, in the past few years there have been many developments in the use of the Web, streaming media and content delivery. Accordingly, sections on the Web and streaming media have been brought up to date. A new section covers content distribution, including CDNs and peer-to-peer networks.

Chapter 8, on security, still covers both symmetric and public-key cryptography for confidentiality and authenticity. Material on the techniques used in practice, including firewalls and VPNs, has been updated, with new material on 802.11 security and Kerberos V5 added.

Chapter 9 contains a renewed list of suggested readings and a comprehensive bibliography of over 300 citations to the current literature. More than half of these are to papers and books written in 2000 or later, and the rest are citations to classic papers.

List of Acronyms

Computer books are full of acronyms. This one is no exception. By the time you are finished reading this one, the following should ring a bell: ADSL, AES, AJAX, AODV, AP, ARP, ARQ, AS, BGP, BOC, CDMA, CDN, CGI, CIDR, CRL, CSMA, CSS, DCT, DES, DHCP, DHT, DIFS, DMCA, DMT, DMZ, DNS, DOCSIS, DOM, DSLAM, DTN, FCFS, FDD, FDDI, FDM, FEC, FIFO, FSK, FTP, GPRS, GSM, HDTV, HFC, HMAC, HTTP, IAB, ICANN, ICMP, IDEA, IETF, IMAP, IMP, IP, IPTV, IRTF, ISO, ISP, ITU, JPEG, JSP, JVM, LAN, LATA, LEC, LEO, LLC, LSR, LTE, MAN, MFJ, MIME, MPEG, MPLS, MSC, MTSO, MTU, NAP, NAT, NRZ, NSAP, OFDM, OSI, OSPF, PAWS, PCM, PGP, PIM, PKI, POP, POTS, PPP, PSTN, QAM, QPSK, RED, RFC, RFID, RPC, RSA, RTSP, SHA, SIP, SMTP, SNR, SOAP, SONET, SPE, SSL, TCP, TDD, TDM, TSAP, UDP, UMTS, URL, VLAN, VSAT, WAN, WDM, and XML. But don't worry. Each will appear in **boldface type** and be carefully defined before it is used. As a fun test, see how many you can identify *before* reading the book, write the number in the margin, then try again *after* reading the book.

How to Use the Book

To help instructors use this book as a text for courses ranging in length from quarters to semesters, we have structured the chapters into core and optional material. The sections marked with a "*" in the table of contents are the optional ones. If a major section (e.g., 2.7) is so marked, all of its subsections are optional. They provide material on network technologies that is useful but can be omitted from a short course without loss of continuity. Of course, students should be encouraged to read those sections as well, to the extent they have time, as all the material is up to date and of value.

Instructors' Resource Materials

The following protected instructors' resource materials are available on the publisher's Web site at *www.pearsonhighered.com/tanenbaum*. For a username and password, please contact your local Pearson representative.

- Solutions manual
- PowerPoint lecture slides

Students' Resource Materials

Resources for students are available through the open-access Companion Web site link on *www.pearsonhighered.com/tanenbaum*, including

- Web resources, links to tutorials, organizations, FAQs, and more
- Figures, tables, and programs from the book
- Steganography demo
- Protocol simulators

Acknowledgements

Many people helped us during the course of the fifth edition. We would especially like to thank Emmanuel Agu (Worcester Polytechnic Institute), Yoris Au (University of Texas at Antonio), Nikhil Bhargava (Aircom International, Inc.), Michael Buettner (University of Washington), John Day (Boston University), Kevin Fall (Intel Labs), Ronald Fulle (Rochester Institute of Technology), Ben Greenstein (Intel Labs), Daniel Halperin (University of Washington), Bob Kinicki (Worcester Polytechnic Institute), Tadayoshi Kohno (University of Washington), Sarvish Kulkarni (Villanova University), Hank Levy (University of Washington), Ratul Mahajan (Microsoft Research), Craig Partridge (BBN), Michael Piatek (University of Washington), Joshua Smith (Intel Labs), Neil Spring (University of Maryland), David Teneyuca (University of Texas at Antonio), Tammy VanDegrift (University of Portland), and Bo Yuan (Rochester Institute of Technology), for providing ideas and feedback. Melody Kadenko and Julie Svendsen provided administrative support to David.

Shivakant Mishra (University of Colorado at Boulder) and Paul Nagin (Chimborazo Publishing, Inc.) thought of many new and challenging end-of-chapter problems. Our editor at Pearson, Tracy Dunkelberger, was her usual helpful self in many ways large and small. Melinda Haggerty and Jeff Holcomb did a good job of keeping things running smoothly. Steve Armstrong (LeTourneau University) prepared the PowerPoint slides. Stephen Turner (University of Michigan at Flint) artfully revised the Web resources and the simulators that accompany the text. Our copyeditor, Rachel Head, is an odd hybrid: she has the eye of an eagle and the memory of an elephant. After reading all her corrections, both of us wondered how we ever made it past third grade.

Finally, we come to the most important people. Suzanne has been through this 19 times now and still has endless patience and love. Barbara and Marvin now know the difference between good textbooks and bad ones and are always an inspiration to produce good ones. Daniel and Matilde are welcome additions to our family. Aron is unlikely to read this book soon, but he likes the nice pictures on page 866 (AST). Katrin and Lucy provided endless support and always managed to keep a smile on my face. Thank you (DJW).

ANDREW S. TANENBAUM

DAVID J. WETHERALL

1

INTRODUCTION

Each of the past three centuries was dominated by a single new technology. The 18th century was the era of the great mechanical systems accompanying the Industrial Revolution. The 19th century was the age of the steam engine. During the 20th century, the key technology was information gathering, processing, and distribution. Among other developments, we saw the installation of worldwide telephone networks, the invention of radio and television, the birth and unprecedented growth of the computer industry, the launching of communication satellites, and, of course, the Internet.

As a result of rapid technological progress, these areas are rapidly converging in the 21st century and the differences between collecting, transporting, storing, and processing information are quickly disappearing. Organizations with hundreds of offices spread over a wide geographical area routinely expect to be able to examine the current status of even their most remote outpost at the push of a button. As our ability to gather, process, and distribute information grows, the demand for ever more sophisticated information processing grows even faster.

Although the computer industry is still young compared to other industries (e.g., automobiles and air transportation), computers have made spectacular progress in a short time. During the first two decades of their existence, computer systems were highly centralized, usually within a single large room. Not infrequently, this room had glass walls, through which visitors could gawk at the great electronic wonder inside. A medium-sized company or university might have had

one or two computers, while very large institutions had at most a few dozen. The idea that within forty years vastly more powerful computers smaller than postage stamps would be mass produced by the billions was pure science fiction.

The merging of computers and communications has had a profound influence on the way computer systems are organized. The once-dominant concept of the ''computer center'' as a room with a large computer to which users bring their work for processing is now totally obsolete (although data centers holding thousands of Internet servers are becoming common). The old model of a single computer serving all of the organization's computational needs has been replaced by one in which a large number of separate but interconnected computers do the job. These systems are called **computer networks**. The design and organization of these networks are the subjects of this book.

Throughout the book we will use the term ''computer network'' to mean a collection of autonomous computers interconnected by a single technology. Two computers are said to be interconnected if they are able to exchange information. The connection need not be via a copper wire; fiber optics, microwaves, infrared, and communication satellites can also be used. Networks come in many sizes, shapes and forms, as we will see later. They are usually connected together to make larger networks, with the **Internet** being the most well-known example of a network of networks.

There is considerable confusion in the literature between a computer network and a **distributed system**. The key distinction is that in a distributed system, a collection of independent computers appears to its users as a single coherent system. Usually, it has a single model or paradigm that it presents to the users. Often a layer of software on top of the operating system, called **middleware**, is responsible for implementing this model. A well-known example of a distributed system is the **World Wide Web**. It runs on top of the Internet and presents a model in which everything looks like a document (Web page).

In a computer network, this coherence, model, and software are absent. Users are exposed to the actual machines, without any attempt by the system to make the machines look and act in a coherent way. If the machines have different hardware and different operating systems, that is fully visible to the users. If a user wants to run a program on a remote machine, he† has to log onto that machine and run it there.

In effect, a distributed system is a software system built on top of a network. The software gives it a high degree of cohesiveness and transparency. Thus, the distinction between a network and a distributed system lies with the software (especially the operating system), rather than with the hardware.

Nevertheless, there is considerable overlap between the two subjects. For example, both distributed systems and computer networks need to move files around. The difference lies in who invokes the movement, the system or the user.

† ''He'' should be read as ''he or she'' throughout this book.

Although this book primarily focuses on networks, many of the topics are also important in distributed systems. For more information about distributed systems, see Tanenbaum and Van Steen (2007).

1.1 USES OF COMPUTER NETWORKS

Before we start to examine the technical issues in detail, it is worth devoting some time to pointing out why people are interested in computer networks and what they can be used for. After all, if nobody were interested in computer networks, few of them would be built. We will start with traditional uses at companies, then move on to home networking and recent developments regarding mobile users, and finish with social issues.

1.1.1 Business Applications

Most companies have a substantial number of computers. For example, a company may have a computer for each worker and use them to design products, write brochures, and do the payroll. Initially, some of these computers may have worked in isolation from the others, but at some point, management may have decided to connect them to be able to distribute information throughout the company.

Put in slightly more general form, the issue here is **resource sharing**. The goal is to make all programs, equipment, and especially data available to anyone on the network without regard to the physical location of the resource or the user. An obvious and widespread example is having a group of office workers share a common printer. None of the individuals really needs a private printer, and a high-volume networked printer is often cheaper, faster, and easier to maintain than a large collection of individual printers.

However, probably even more important than sharing physical resources such as printers, and tape backup systems, is sharing information. Companies small and large are vitally dependent on computerized information. Most companies have customer records, product information, inventories, financial statements, tax information, and much more online. If all of its computers suddenly went down, a bank could not last more than five minutes. A modern manufacturing plant, with a computer-controlled assembly line, would not last even 5 seconds. Even a small travel agency or three-person law firm is now highly dependent on computer networks for allowing employees to access relevant information and documents instantly.

For smaller companies, all the computers are likely to be in a single office or perhaps a single building, but for larger ones, the computers and employees may be scattered over dozens of offices and plants in many countries. Nevertheless, a sales person in New York might sometimes need access to a product inventory

database in Singapore. Networks called **VPNs** (**Virtual Private Networks**) may be used to join the individual networks at different sites into one extended network. In other words, the mere fact that a user happens to be 15,000 km away from his data should not prevent him from using the data as though they were local. This goal may be summarized by saying that it is an attempt to end the "tyranny of geography."

In the simplest of terms, one can imagine a company's information system as consisting of one or more databases with company information and some number of employees who need to access them remotely. In this model, the data are stored on powerful computers called **servers**. Often these are centrally housed and maintained by a system administrator. In contrast, the employees have simpler machines, called **clients**, on their desks, with which they access remote data, for example, to include in spreadsheets they are constructing. (Sometimes we will refer to the human user of the client machine as the "client," but it should be clear from the context whether we mean the computer or its user.) The client and server machines are connected by a network, as illustrated in Fig. 1-1. Note that we have shown the network as a simple oval, without any detail. We will use this form when we mean a network in the most abstract sense. When more detail is required, it will be provided.

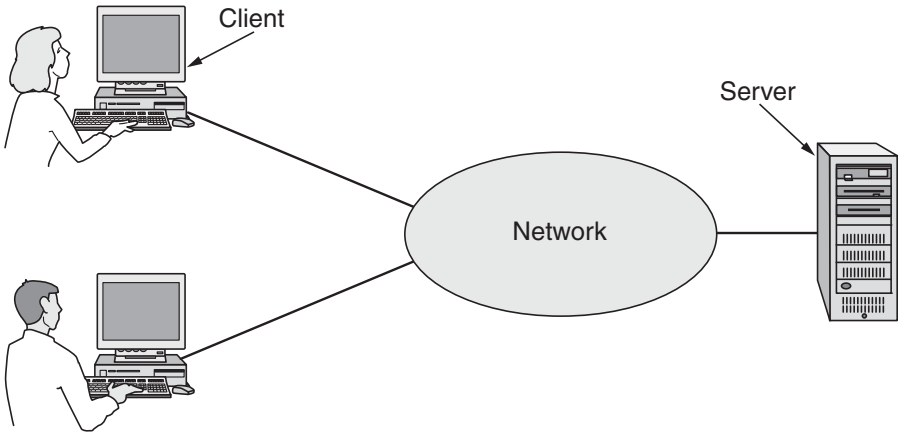


Figure 1-1. A network with two clients and one server.

This whole arrangement is called the **client-server model**. It is widely used and forms the basis of much network usage. The most popular realization is that of a **Web application**, in which the server generates Web pages based on its database in response to client requests that may update the database. The client-server model is applicable when the client and server are both in the same building (and belong to the same company), but also when they are far apart. For example, when a person at home accesses a page on the World Wide Web, the same model is employed, with the remote Web server being the server and the user's personal

computer being the client. Under most conditions, one server can handle a large number (hundreds or thousands) of clients simultaneously.

If we look at the client-server model in detail, we see that two processes (i.e., running programs) are involved, one on the client machine and one on the server machine. Communication takes the form of the client process sending a message over the network to the server process. The client process then waits for a reply message. When the server process gets the request, it performs the requested work or looks up the requested data and sends back a reply. These messages are shown in Fig. 1-2.

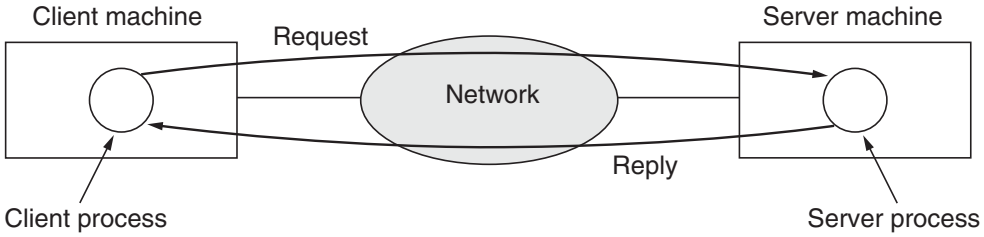


Figure 1-2. The client-server model involves requests and replies.

A second goal of setting up a computer network has to do with people rather than information or even computers. A computer network can provide a powerful **communication medium** among employees. Virtually every company that has two or more computers now has **email** (**electronic mail**), which employees generally use for a great deal of daily communication. In fact, a common gripe around the water cooler is how much email everyone has to deal with, much of it quite meaningless because bosses have discovered that they can send the same (often content-free) message to all their subordinates at the push of a button.

Telephone calls between employees may be carried by the computer network instead of by the phone company. This technology is called **IP telephony** or **Voice over IP** (**VoIP**) when Internet technology is used. The microphone and speaker at each end may belong to a VoIP-enabled phone or the employee's computer. Companies find this a wonderful way to save on their telephone bills.

Other, richer forms of communication are made possible by computer networks. Video can be added to audio so that employees at distant locations can see and hear each other as they hold a meeting. This technique is a powerful tool for eliminating the cost and time previously devoted to travel. **Desktop sharing** lets remote workers see and interact with a graphical computer screen. This makes it easy for two or more people who work far apart to read and write a shared blackboard or write a report together. When one worker makes a change to an online document, the others can see the change immediately, instead of waiting several days for a letter. Such a speedup makes cooperation among far-flung groups of people easy where it previously had been impossible. More ambitious forms of remote coordination such as telemedicine are only now starting to be used (e.g.,

remote patient monitoring) but may become much more important. It is sometimes said that communication and transportation are having a race, and whichever wins will make the other obsolete.

A third goal for many companies is doing business electronically, especially with customers and suppliers. This new model is called **e-commerce** (**electronic commerce**) and it has grown rapidly in recent years. Airlines, bookstores, and other retailers have discovered that many customers like the convenience of shopping from home. Consequently, many companies provide catalogs of their goods and services online and take orders online. Manufacturers of automobiles, aircraft, and computers, among others, buy subsystems from a variety of suppliers and then assemble the parts. Using computer networks, manufacturers can place orders electronically as needed. This reduces the need for large inventories and enhances efficiency.

1.1.2 Home Applications

In 1977, Ken Olsen was president of the Digital Equipment Corporation, then the number two computer vendor in the world (after IBM). When asked why Digital was not going after the personal computer market in a big way, he said: ''There is no reason for any individual to have a computer in his home.'' History showed otherwise and Digital no longer exists. People initially bought computers for word processing and games. Recently, the biggest reason to buy a home computer was probably for Internet access. Now, many consumer electronic devices, such as set-top boxes, game consoles, and clock radios, come with embedded computers and computer networks, especially wireless networks, and home networks are broadly used for entertainment, including listening to, looking at, and creating music, photos, and videos.

Internet access provides home users with **connectivity** to remote computers. As with companies, home users can access information, communicate with other people, and buy products and services with e-commerce. The main benefit now comes from connecting outside of the home. Bob Metcalfe, the inventor of Ethernet, hypothesized that the value of a network is proportional to the square of the number of users because this is roughly the number of different connections that may be made (Gilder, 1993). This hypothesis is known as ''Metcalfe's law.'' It helps to explain how the tremendous popularity of the Internet comes from its size.

Access to remote information comes in many forms. It can be surfing the World Wide Web for information or just for fun. Information available includes the arts, business, cooking, government, health, history, hobbies, recreation, science, sports, travel, and many others. Fun comes in too many ways to mention, plus some ways that are better left unmentioned.

Many newspapers have gone online and can be personalized. For example, it is sometimes possible to tell a newspaper that you want everything about corrupt

politicians, big fires, scandals involving celebrities, and epidemics, but no football, thank you. Sometimes it is possible to have the selected articles downloaded to your computer while you sleep. As this trend continues, it will cause massive unemployment among 12-year-old paperboys, but newspapers like it because distribution has always been the weakest link in the whole production chain. Of course, to make this model work, they will first have to figure out how to make money in this new world, something not entirely obvious since Internet users expect everything to be free.

The next step beyond newspapers (plus magazines and scientific journals) is the online digital library. Many professional organizations, such as the ACM (*www.acm.org*) and the IEEE Computer Society (*www.computer.org*), already have all their journals and conference proceedings online. Electronic book readers and online libraries may make printed books obsolete. Skeptics should take note of the effect the printing press had on the medieval illuminated manuscript.

Much of this information is accessed using the client-server model, but there is different, popular model for accessing information that goes by the name of **peer-to-peer** communication (Parameswaran et al., 2001). In this form, individuals who form a loose group can communicate with others in the group, as shown in Fig. 1-3. Every person can, in principle, communicate with one or more other people; there is no fixed division into clients and servers.

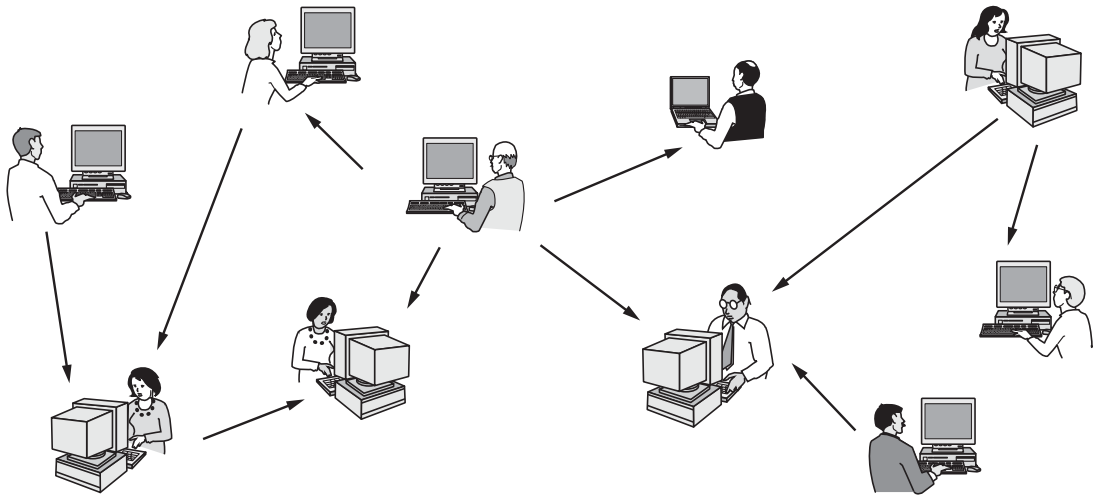


Figure 1-3. In a peer-to-peer system there are no fixed clients and servers.

Many peer-to-peer systems, such BitTorrent (Cohen, 2003), do not have any central database of content. Instead, each user maintains his own database locally and provides a list of other nearby people who are members of the system. A new user can then go to any existing member to see what he has and get the names of other members to inspect for more content and more names. This lookup process can be repeated indefinitely to build up a large local database of what is out there. It is an activity that would get tedious for people but computers excel at it.

Peer-to-peer communication is often used to share music and videos. It really hit the big time around 2000 with a music sharing service called Napster that was shut down after what was probably the biggest copyright infringement case in all of recorded history (Lam and Tan, 2001; and Macedonia, 2000). Legal applications for peer-to-peer communication also exist. These include fans sharing public domain music, families sharing photos and movies, and users downloading public software packages. In fact, one of the most popular Internet applications of all, email, is inherently peer-to-peer. This form of communication is likely to grow considerably in the future.

All of the above applications involve interactions between a person and a remote database full of information. The second broad category of network use is person-to-person communication, basically the 21st century's answer to the 19th century's telephone. E-mail is already used on a daily basis by millions of people all over the world and its use is growing rapidly. It already routinely contains audio and video as well as text and pictures. Smell may take a while.

Any teenager worth his or her salt is addicted to **instant messaging**. This facility, derived from the UNIX *talk* program in use since around 1970, allows two people to type messages at each other in real time. There are multi-person messaging services too, such as the **Twitter** service that lets people send short text messages called "tweets" to their circle of friends or other willing audiences.

The Internet can be used by applications to carry audio (e.g., Internet radio stations) and video (e.g., YouTube). Besides being a cheap way to call to distant friends, these applications can provide rich experiences such as telelearning, meaning attending 8 A.M. classes without the inconvenience of having to get out of bed first. In the long run, the use of networks to enhance human-to-human communication may prove more important than any of the others. It may become hugely important to people who are geographically challenged, giving them the same access to services as people living in the middle of a big city.

Between person-to-person communications and accessing information are **social network** applications. Here, the flow of information is driven by the relationships that people declare between each other. One of the most popular social networking sites is **Facebook**. It lets people update their personal profiles and shares the updates with other people who they have declared to be their friends. Other social networking applications can make introductions via friends of friends, send news messages to friends such as Twitter above, and much more.

Even more loosely, groups of people can work together to create content. A **wiki**, for example, is a collaborative Web site that the members of a community edit. The most famous wiki is the **Wikipedia**, an encyclopedia anyone can edit, but there are thousands of other wikis.

Our third category is electronic commerce in the broadest sense of the term. Home shopping is already popular and enables users to inspect the online catalogs of thousands of companies. Some of these catalogs are interactive, showing products from different viewpoints and in configurations that can be personalized.

After the customer buys a product electronically but cannot figure out how to use it, online technical support may be consulted.

Another area in which e-commerce is widely used is access to financial institutions. Many people already pay their bills, manage their bank accounts, and handle their investments electronically. This trend will surely continue as networks become more secure.

One area that virtually nobody foresaw is electronic flea markets (e-flea?). Online auctions of second-hand goods have become a massive industry. Unlike traditional e-commerce, which follows the client-server model, online auctions are peer-to-peer in the sense that consumers can act as both buyers and sellers.

Some of these forms of e-commerce have acquired cute little tags based on the fact that "to" and "2" are pronounced the same. The most popular ones are listed in Fig. 1-4.

Tag	Full name	Example
B2C	Business-to-consumer	Ordering books online
B2B	Business-to-business	Car manufacturer ordering tires from supplier
G2C	Government-to-consumer	Government distributing tax forms electronically
C2C	Consumer-to-consumer	Auctioning second-hand products online
P2P	Peer-to-peer	Music sharing

Figure 1-4. Some forms of e-commerce.

Our fourth category is entertainment. This has made huge strides in the home in recent years, with the distribution of music, radio and television programs, and movies over the Internet beginning to rival that of traditional mechanisms. Users can find, buy, and download MP3 songs and DVD-quality movies and add them to their personal collection. TV shows now reach many homes via **IPTV** (**IP TeleVision**) systems that are based on IP technology instead of cable TV or radio transmissions. Media streaming applications let users tune into Internet radio stations or watch recent episodes of their favorite TV shows. Naturally, all of this content can be moved around your house between different devices, displays and speakers, usually with a wireless network.

Soon, it may be possible to search for any movie or television program ever made, in any country, and have it displayed on your screen instantly. New films may become interactive, where the user is occasionally prompted for the story direction (should Macbeth murder Duncan or just bide his time?) with alternative scenarios provided for all cases. Live television may also become interactive, with the audience participating in quiz shows, choosing among contestants, and so on.

Another form of entertainment is game playing. Already we have multiperson real-time simulation games, like hide-and-seek in a virtual dungeon, and flight

simulators with the players on one team trying to shoot down the players on the opposing team. Virtual worlds provide a persistent setting in which thousands of users can experience a shared reality with three-dimensional graphics.

Our last category is **ubiquitous computing**, in which computing is embedded into everyday life, as in the vision of Mark Weiser (1991). Many homes are already wired with security systems that include door and window sensors, and there are many more sensors that can be folded in to a smart home monitor, such as energy consumption. Your electricity, gas and water meters could also report usage over the network. This would save money as there would be no need to send out meter readers. And your smoke detectors could call the fire department instead of making a big noise (which has little value if no one is home). As the cost of sensing and communication drops, more and more measurement and reporting will be done with networks.

Increasingly, consumer electronic devices are networked. For example, some high-end cameras already have a wireless network capability and use it to send photos to a nearby display for viewing. Professional sports photographers can also send their photos to their editors in real-time, first wirelessly to an access point then over the Internet. Devices such as televisions that plug into the wall can use **power-line networks** to send information throughout the house over the wires that carry electricity. It may not be very surprising to have these objects on the network, but objects that we do not think of as computers may sense and communicate information too. For example, your shower may record water usage, give you visual feedback while you lather up, and report to a home environmental monitoring application when you are done to help save on your water bill.

A technology called **RFID** (**Radio Frequency IDentification**) will push this idea even further in the future. RFID tags are passive (i.e., have no battery) chips the size of stamps and they can already be affixed to books, passports, pets, credit cards, and other items in the home and out. This lets RFID readers locate and communicate with the items over a distance of up to several meters, depending on the kind of RFID. Originally, RFID was commercialized to replace barcodes. It has not succeeded yet because barcodes are free and RFID tags cost a few cents. Of course, RFID tags offer much more and their price is rapidly declining. They may turn the real world into the Internet of things (ITU, 2005).

1.1.3 Mobile Users

Mobile computers, such as laptop and handheld computers, are one of the fastest-growing segments of the computer industry. Their sales have already overtaken those of desktop computers. Why would anyone want one? People on the go often want to use their mobile devices to read and send email, tweet, watch movies, download music, play games, or simply to surf the Web for information. They want to do all of the things they do at home and in the office. Naturally, they want to do them from anywhere on land, sea or in the air.

Connectivity to the Internet enables many of these mobile uses. Since having a wired connection is impossible in cars, boats, and airplanes, there is a lot of interest in wireless networks. Cellular networks operated by the telephone companies are one familiar kind of wireless network that blankets us with coverage for mobile phones. Wireless **hotspots** based on the 802.11 standard are another kind of wireless network for mobile computers. They have sprung up everywhere that people go, resulting in a patchwork of coverage at cafes, hotels, airports, schools, trains and planes. Anyone with a laptop computer and a wireless modem can just turn on their computer on and be connected to the Internet through the hotspot, as though the computer were plugged into a wired network.

Wireless networks are of great value to fleets of trucks, taxis, delivery vehicles, and repairpersons for keeping in contact with their home base. For example, in many cities, taxi drivers are independent businessmen, rather than being employees of a taxi company. In some of these cities, the taxis have a display the driver can see. When a customer calls up, a central dispatcher types in the pickup and destination points. This information is displayed on the drivers' displays and a beep sounds. The first driver to hit a button on the display gets the call.

Wireless networks are also important to the military. If you have to be able to fight a war anywhere on Earth at short notice, counting on using the local networking infrastructure is probably not a good idea. It is better to bring your own.

Although wireless networking and mobile computing are often related, they are not identical, as Fig. 1-5 shows. Here we see a distinction between **fixed wireless** and **mobile wireless** networks. Even notebook computers are sometimes wired. For example, if a traveler plugs a notebook computer into the wired network jack in a hotel room, he has mobility without a wireless network.

Wireless	Mobile	Typical applications
No	No	Desktop computers in offices
No	Yes	A notebook computer used in a hotel room
Yes	No	Networks in unwired buildings
Yes	Yes	Store inventory with a handheld computer

Figure 1-5. Combinations of wireless networks and mobile computing.

Conversely, some wireless computers are not mobile. In the home, and in offices or hotels that lack suitable cabling, it can be more convenient to connect desktop computers or media players wirelessly than to install wires. Installing a wireless network may require little more than buying a small box with some electronics in it, unpacking it, and plugging it in. This solution may be far cheaper than having workmen put in cable ducts to wire the building.

Finally, there are also true mobile, wireless applications, such as people walking around stores with a handheld computers recording inventory. At many busy

airports, car rental return clerks work in the parking lot with wireless mobile computers. They scan the barcodes or RFID chips of returning cars, and their mobile device, which has a built-in printer, calls the main computer, gets the rental information, and prints out the bill on the spot.

Perhaps the key driver of mobile, wireless applications is the mobile phone. **Text messaging** or **texting** is tremendously popular. It lets a mobile phone user type a short message that is then delivered by the cellular network to another mobile subscriber. Few people would have predicted ten years ago that having teenagers tediously typing short text messages on mobile phones would be an immense money maker for telephone companies. But texting (or **Short Message Service** as it is known outside the U.S.) is very profitable since it costs the carrier but a tiny fraction of one cent to relay a text message, a service for which they charge far more.

The long-awaited convergence of telephones and the Internet has finally arrived, and it will accelerate the growth of mobile applications. **Smart phones**, such as the popular iPhone, combine aspects of mobile phones and mobile computers. The (3G and 4G) cellular networks to which they connect can provide fast data services for using the Internet as well as handling phone calls. Many advanced phones connect to wireless hotspots too, and automatically switch between networks to choose the best option for the user.

Other consumer electronics devices can also use cellular and hotspot networks to stay connected to remote computers. Electronic book readers can download a newly purchased book or the next edition of a magazine or today's newspaper wherever they roam. Electronic picture frames can update their displays on cue with fresh images.

Since mobile phones know their locations, often because they are equipped with **GPS** (**Global Positioning System**) receivers, some services are intentionally location dependent. Mobile maps and directions are an obvious candidate as your GPS-enabled phone and car probably have a better idea of where you are than you do. So, too, are searches for a nearby bookstore or Chinese restaurant, or a local weather forecast. Other services may record location, such as annotating photos and videos with the place at which they were made. This annotation is known as "geo-tagging."

An area in which mobile phones are now starting to be used is **m-commerce** (**mobile-commerce**) (Senn, 2000). Short text messages from the mobile are used to authorize payments for food in vending machines, movie tickets, and other small items instead of cash and credit cards. The charge then appears on the mobile phone bill. When equipped with **NFC** (**Near Field Communication**) technology the mobile can act as an RFID smartcard and interact with a nearby reader for payment. The driving forces behind this phenomenon are the mobile device makers and network operators, who are trying hard to figure out how to get a piece of the e-commerce pie. From the store's point of view, this scheme may save them most of the credit card company's fee, which can be several percent.

Of course, this plan may backfire, since customers in a store might use the RFID or barcode readers on their mobile devices to check out competitors' prices before buying and use them to get a detailed report on where else an item can be purchased nearby and at what price.

One huge thing that m-commerce has going for it is that mobile phone users are accustomed to paying for everything (in contrast to Internet users, who expect everything to be free). If an Internet Web site charged a fee to allow its customers to pay by credit card, there would be an immense howling noise from the users. If, however, a mobile phone operator its customers to pay for items in a store by waving the phone at the cash register and then tacked on a fee for this convenience, it would probably be accepted as normal. Time will tell.

No doubt the uses of mobile and wireless computers will grow rapidly in the future as the size of computers shrinks, probably in ways no one can now foresee. Let us take a quick look at some possibilities. **Sensor networks** are made up of nodes that gather and wirelessly relay information they sense about the state of the physical world. The nodes may be part of familiar items such as cars or phones, or they may be small separate devices. For example, your car might gather data on its location, speed, vibration, and fuel efficiency from its on-board diagnostic system and upload this information to a database (Hull et al., 2006). Those data can help find potholes, plan trips around congested roads, and tell you if you are a ''gas guzzler'' compared to other drivers on the same stretch of road.

Sensor networks are revolutionizing science by providing a wealth of data on behavior that could not previously be observed. One example is tracking the migration of individual zebras by placing a small sensor on each animal (Juang et al., 2002). Researchers have packed a wireless computer into a cube 1 mm on edge (Warneke et al., 2001). With mobile computers this small, even small birds, rodents, and insects can be tracked.

Even mundane uses, such as in parking meters, can be significant because they make use of data that were not previously available. Wireless parking meters can accept credit or debit card payments with instant verification over the wireless link. They can also report when they are in use over the wireless network. This would let drivers download a recent parking map to their car so they can find an available spot more easily. Of course, when a meter expires, it might also check for the presence of a car (by bouncing a signal off it) and report the expiration to parking enforcement. It has been estimated that city governments in the U.S. alone could collect an additional $10 billion this way (Harte et al., 2000).

Wearable computers are another promising application. Smart watches with radios have been part of our mental space since their appearance in the Dick Tracy comic strip in 1946; now you can buy them. Other such devices may be implanted, such as pacemakers and insulin pumps. Some of these can be controlled over a wireless network. This lets doctors test and reconfigure them more easily. It could also lead to some nasty problems if the devices are as insecure as the average PC and can be hacked easily (Halperin et al., 2008).

1.1.4 Social Issues

Computer networks, like the printing press 500 years ago, allow ordinary citizens to distribute and view content in ways that were not previously possible. But along with the good comes the bad, as this new-found freedom brings with it many unsolved social, political, and ethical issues. Let us just briefly mention a few of them; a thorough study would require a full book, at least.

Social networks, message boards, content sharing sites, and a host of other applications allow people to share their views with like-minded individuals. As long as the subjects are restricted to technical topics or hobbies like gardening, not too many problems will arise.

The trouble comes with topics that people actually care about, like politics, religion, or sex. Views that are publicly posted may be deeply offensive to some people. Worse yet, they may not be politically correct. Furthermore, opinions need not be limited to text; high-resolution color photographs and video clips are easily shared over computer networks. Some people take a live-and-let-live view, but others feel that posting certain material (e.g., verbal attacks on particular countries or religions, pornography, etc.) is simply unacceptable and that such content must be censored. Different countries have different and conflicting laws in this area. Thus, the debate rages.

In the past, people have sued network operators, claiming that they are responsible for the contents of what they carry, just as newspapers and magazines are. The inevitable response is that a network is like a telephone company or the post office and cannot be expected to police what its users say.

It should now come only as a slight surprise to learn that some network operators block content for their own reasons. Some users of peer-to-peer applications had their network service cut off because the network operators did not find it profitable to carry the large amounts of traffic sent by those applications. Those same operators would probably like to treat different companies differently. If you are a big company and pay well then you get good service, but if you are a small-time player, you get poor service. Opponents of this practice argue that peer-to-peer and other content should be treated in the same way because they are all just bits to the network. This argument for communications that are not differentiated by their content or source or who is providing the content is known as **network neutrality** (Wu, 2003). It is probably safe to say that this debate will go on for a while.

Many other parties are involved in the tussle over content. For instance, pirated music and movies fueled the massive growth of peer-to-peer networks, which did not please the copyright holders, who have threatened (and sometimes taken) legal action. There are now automated systems that search peer-to-peer networks and fire off warnings to network operators and users who are suspected of infringing copyright. In the United States, these warnings are known as **DMCA takedown notices** after the **Digital Millennium Copyright Act**. This

search is an arms' race because it is hard to reliably catch copyright infringement. Even your printer might be mistaken for a culprit (Piatek et al., 2008).

Computer networks make it very easy to communicate. They also make it easy for the people who run the network to snoop on the traffic. This sets up conflicts over issues such as employee rights versus employer rights. Many people read and write email at work. Many employers have claimed the right to read and possibly censor employee messages, including messages sent from a home computer outside working hours. Not all employees agree with this, especially the latter part.

Another conflict is centered around government versus citizen's rights. The FBI has installed systems at many Internet service providers to snoop on all incoming and outgoing email for nuggets of interest. One early system was originally called Carnivore, but bad publicity caused it to be renamed to the more innocent-sounding DCS1000 (Blaze and Bellovin, 2000; Sobel, 2001; and Zacks, 2001). The goal of such systems is to spy on millions of people in the hope of perhaps finding information about illegal activities. Unfortunately for the spies, the Fourth Amendment to the U.S. Constitution prohibits government searches without a search warrant, but the government often ignores it.

Of course, the government does not have a monopoly on threatening people's privacy. The private sector does its bit too by **profiling** users. For example, small files called **cookies** that Web browsers store on users' computers allow companies to track users' activities in cyberspace and may also allow credit card numbers, social security numbers, and other confidential information to leak all over the Internet (Berghel, 2001). Companies that provide Web-based services may maintain large amounts of personal information about their users that allows them to study user activities directly. For example, Google can read your email and show you advertisements based on your interests if you use its email service, **Gmail**.

A new twist with mobile devices is location privacy (Beresford and Stajano, 2003). As part of the process of providing service to your mobile device the network operators learn where you are at different times of day. This allows them to track your movements. They may know which nightclub you frequent and which medical center you visit.

Computer networks also offer the potential to increase privacy by sending anonymous messages. In some situations, this capability may be desirable. Beyond preventing companies from learning your habits, it provides, for example, a way for students, soldiers, employees, and citizens to blow the whistle on illegal behavior on the part of professors, officers, superiors, and politicians without fear of reprisals. On the other hand, in the United States and most other democracies, the law specifically permits an accused person the right to confront and challenge his accuser in court so anonymous accusations cannot be used as evidence.

The Internet makes it possible to find information quickly, but a great deal of it is ill considered, misleading, or downright wrong. That medical advice you

plucked from the Internet about the pain in your chest may have come from a Nobel Prize winner or from a high-school dropout.

Other information is frequently unwanted. Electronic junk mail (spam) has become a part of life because spammers have collected millions of email addresses and would-be marketers can cheaply send computer-generated messages to them. The resulting flood of spam rivals the flow messages from real people. Fortunately, filtering software is able to read and discard the spam generated by other computers, with lesser or greater degrees of success.

Still other content is intended for criminal behavior. Web pages and email messages containing active content (basically, programs or macros that execute on the receiver's machine) can contain viruses that take over your computer. They might be used to steal your bank account passwords, or to have your computer send spam as part of a **botnet** or pool of compromised machines.

Phishing messages masquerade as originating from a trustworthy party, for example, your bank, to try to trick you into revealing sensitive information, for example, credit card numbers. Identity theft is becoming a serious problem as thieves collect enough information about a victim to obtain credit cards and other documents in the victim's name.

It can be difficult to prevent computers from impersonating people on the Internet. This problem has led to the development of **CAPTCHAs**, in which a computer asks a person to solve a short recognition task, for example, typing in the letters shown in a distorted image, to show that they are human (von Ahn, 2001). This process is a variation on the famous Turing test in which a person asks questions over a network to judge whether the entity responding is human.

A lot of these problems could be solved if the computer industry took computer security seriously. If all messages were encrypted and authenticated, it would be harder to commit mischief. Such technology is well established and we will study it in detail in Chap. 8. The problem is that hardware and software vendors know that putting in security features costs money and their customers are not demanding such features. In addition, a substantial number of the problems are caused by buggy software, which occurs because vendors keep adding more and more features to their programs, which inevitably means more code and thus more bugs. A tax on new features might help, but that might be a tough sell in some quarters. A refund for defective software might be nice, except it would bankrupt the entire software industry in the first year.

Computer networks raise new legal problems when they interact with old laws. Electronic gambling provides an example. Computers have been simulating things for decades, so why not simulate slot machines, roulette wheels, blackjack dealers, and more gambling equipment? Well, because it is illegal in a lot of places. The trouble is, gambling is legal in a lot of other places (England, for example) and casino owners there have grasped the potential for Internet gambling. What happens if the gambler, the casino, and the server are all in different countries, with conflicting laws? Good question.

1.2 NETWORK HARDWARE

It is now time to turn our attention from the applications and social aspects of networking (the dessert) to the technical issues involved in network design (the spinach). There is no generally accepted taxonomy into which all computer networks fit, but two dimensions stand out as important: transmission technology and scale. We will now examine each of these in turn.

Broadly speaking, there are two types of transmission technology that are in widespread use: **broadcast** links and **point-to-point** links.

Point-to-point links connect individual pairs of machines. To go from the source to the destination on a network made up of point-to-point links, short messages, called **packets** in certain contexts, may have to first visit one or more intermediate machines. Often multiple routes, of different lengths, are possible, so finding good ones is important in point-to-point networks. Point-to-point transmission with exactly one sender and exactly one receiver is sometimes called **unicasting**.

In contrast, on a broadcast network, the communication channel is shared by all the machines on the network; packets sent by any machine are received by all the others. An address field within each packet specifies the intended recipient. Upon receiving a packet, a machine checks the address field. If the packet is intended for the receiving machine, that machine processes the packet; if the packet is intended for some other machine, it is just ignored.

A wireless network is a common example of a broadcast link, with communication shared over a coverage region that depends on the wireless channel and the transmitting machine. As an analogy, consider someone standing in a meeting room and shouting "Watson, come here. I want you." Although the packet may actually be received (heard) by many people, only Watson will respond; the others just ignore it.

Broadcast systems usually also allow the possibility of addressing a packet to *all* destinations by using a special code in the address field. When a packet with this code is transmitted, it is received and processed by every machine on the network. This mode of operation is called **broadcasting**. Some broadcast systems also support transmission to a subset of the machines, which known as **multicasting**.

An alternative criterion for classifying networks is by scale. Distance is important as a classification metric because different technologies are used at different scales.

In Fig. 1-6 we classify multiple processor systems by their rough physical size. At the top are the personal area networks, networks that are meant for one person. Beyond these come longer-range networks. These can be divided into local, metropolitan, and wide area networks, each with increasing scale. Finally, the connection of two or more networks is called an internetwork. The worldwide Internet is certainly the best-known (but not the only) example of an internetwork.

Soon we will have even larger internetworks with the **Interplanetary Internet** that connects networks across space (Burleigh et al., 2003).

Interprocessor distance	Processors located in same	Example
1 m	Square meter	Personal area network
10 m	Room	Local area network
100 m	Building	
1 km	Campus	
10 km	City	Metropolitan area network
100 km	Country	Wide area network
1000 km	Continent	
10,000 km	Planet	The Internet

Figure 1-6. Classification of interconnected processors by scale.

In this book we will be concerned with networks at all these scales. In the following sections, we give a brief introduction to network hardware by scale.

1.2.1 Personal Area Networks

PANs (**Personal Area Networks**) let devices communicate over the range of a person. A common example is a wireless network that connects a computer with its peripherals. Almost every computer has an attached monitor, keyboard, mouse, and printer. Without using wireless, this connection must be done with cables. So many new users have a hard time finding the right cables and plugging them into the right little holes (even though they are usually color coded) that most computer vendors offer the option of sending a technician to the user's home to do it. To help these users, some companies got together to design a short-range wireless network called **Bluetooth** to connect these components without wires. The idea is that if your devices have Bluetooth, then you need no cables. You just put them down, turn them on, and they work together. For many people, this ease of operation is a big plus.

In the simplest form, Bluetooth networks use the master-slave paradigm of Fig. 1-7. The system unit (the PC) is normally the master, talking to the mouse, keyboard, etc., as slaves. The master tells the slaves what addresses to use, when they can broadcast, how long they can transmit, what frequencies they can use, and so on.

Bluetooth can be used in other settings, too. It is often used to connect a headset to a mobile phone without cords and it can allow your digital music player

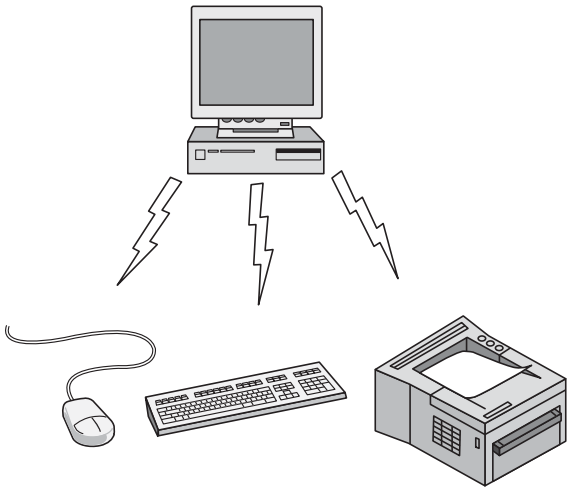


Figure 1-7. Bluetooth PAN configuration.

to connect to your car merely being brought within range. A completely different kind of PAN is formed when an embedded medical device such as a pacemaker, insulin pump, or hearing aid talks to a user-operated remote control. We will discuss Bluetooth in more detail in Chap. 4.

PANs can also be built with other technologies that communicate over short ranges, such as RFID on smartcards and library books. We will study RFID in Chap. 4.

1.2.2 Local Area Networks

The next step up is the **LAN** (**Local Area Network**). A LAN is a privately owned network that operates within and nearby a single building like a home, office or factory. LANs are widely used to connect personal computers and consumer electronics to let them share resources (e.g., printers) and exchange information. When LANs are used by companies, they are called **enterprise networks**.

Wireless LANs are very popular these days, especially in homes, older office buildings, cafeterias, and other places where it is too much trouble to install cables. In these systems, every computer has a radio modem and an antenna that it uses to communicate with other computers. In most cases, each computer talks to a device in the ceiling as shown in Fig. 1-8(a). This device, called an **AP** (**Access Point**), **wireless router**, or **base station**, relays packets between the wireless computers and also between them and the Internet. Being the AP is like being the popular kid as school because everyone wants to talk to you. However, if other computers are close enough, they can communicate directly with one another in a peer-to-peer configuration.

There is a standard for wireless LANs called **IEEE 802.11**, popularly known as **WiFi**, which has become very widespread. It runs at speeds anywhere from 11

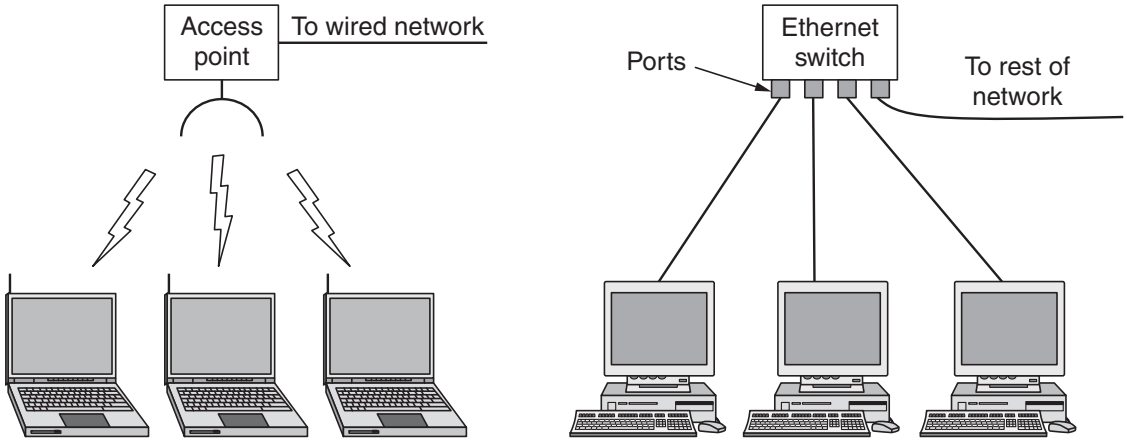


Figure 1-8. Wireless and wired LANs. (a) 802.11. (b) Switched Ethernet.

to hundreds of Mbps. (In this book we will adhere to tradition and measure line speeds in megabits/sec, where 1 Mbps is 1,000,000 bits/sec, and gigabits/sec, where 1 Gbps is 1,000,000,000 bits/sec.) We will discuss 802.11 in Chap. 4.

Wired LANs use a range of different transmission technologies. Most of them use copper wires, but some use optical fiber. LANs are restricted in size, which means that the worst-case transmission time is bounded and known in advance. Knowing these bounds helps with the task of designing network protocols. Typically, wired LANs run at speeds of 100 Mbps to 1 Gbps, have low delay (microseconds or nanoseconds), and make very few errors. Newer LANs can operate at up to 10 Gbps. Compared to wireless networks, wired LANs exceed them in all dimensions of performance. It is just easier to send signals over a wire or through a fiber than through the air.

The topology of many wired LANs is built from point-to-point links. IEEE 802.3, popularly called **Ethernet**, is, by far, the most common type of wired LAN. Fig. 1-8(b) shows a sample topology of **switched Ethernet**. Each computer speaks the Ethernet protocol and connects to a box called a **switch** with a point-to-point link. Hence the name. A switch has multiple **ports**, each of which can connect to one computer. The job of the switch is to relay packets between computers that are attached to it, using the address in each packet to determine which computer to send it to.

To build larger LANs, switches can be plugged into each other using their ports. What happens if you plug them together in a loop? Will the network still work? Luckily, the designers thought of this case. It is the job of the protocol to sort out what paths packets should travel to safely reach the intended computer. We will see how this works in Chap. 4.

It is also possible to divide one large physical LAN into two smaller logical LANs. You might wonder why this would be useful. Sometimes, the layout of the network equipment does not match the organization's structure. For example, the

engineering and finance departments of a company might have computers on the same physical LAN because they are in the same wing of the building but it might be easier to manage the system if engineering and finance logically each had its own network **Virtual LAN** or **VLAN**. In this design each port is tagged with a ''color,'' say green for engineering and red for finance. The switch then forwards packets so that computers attached to the green ports are separated from the computers attached to the red ports. Broadcast packets sent on a red port, for example, will not be received on a green port, just as though there were two different LANs. We will cover VLANs at the end of Chap. 4.

There are other wired LAN topologies too. In fact, switched Ethernet is a modern version of the original Ethernet design that broadcast all the packets over a single linear cable. At most one machine could successfully transmit at a time, and a distributed arbitration mechanism was used to resolve conflicts. It used a simple algorithm: computers could transmit whenever the cable was idle. If two or more packets collided, each computer just waited a random time and tried later. We will call that version **classic Ethernet** for clarity, and as you suspected, you will learn about it in Chap. 4.

Both wireless and wired broadcast networks can be divided into static and dynamic designs, depending on how the channel is allocated. A typical static allocation would be to divide time into discrete intervals and use a round-robin algorithm, allowing each machine to broadcast only when its time slot comes up. Static allocation wastes channel capacity when a machine has nothing to say during its allocated slot, so most systems attempt to allocate the channel dynamically (i.e., on demand).

Dynamic allocation methods for a common channel are either centralized or decentralized. In the centralized channel allocation method, there is a single entity, for example, the base station in cellular networks, which determines who goes next. It might do this by accepting multiple packets and prioritizing them according to some internal algorithm. In the decentralized channel allocation method, there is no central entity; each machine must decide for itself whether to transmit. You might think that this approach would lead to chaos, but it does not. Later we will study many algorithms designed to bring order out of the potential chaos.

It is worth spending a little more time discussing LANs in the home. In the future, it is likely that every appliance in the home will be capable of communicating with every other appliance, and all of them will be accessible over the Internet. This development is likely to be one of those visionary concepts that nobody asked for (like TV remote controls or mobile phones), but once they arrived nobody can imagine how they lived without them.

Many devices are already capable of being networked. These include computers, entertainment devices such as TVs and DVDs, phones and other consumer electronics such as cameras, appliances like clock radios, and infrastructure like utility meters and thermostats. This trend will only continue. For instance, the average home probably has a dozen clocks (e.g., in appliances), all of which could

adjust to daylight savings time automatically if the clocks were on the Internet. Remote monitoring of the home is a likely winner, as many grown children would be willing to spend some money to help their aging parents live safely in their own homes.

While we could think of the home network as just another LAN, it is more likely to have different properties than other networks. First, the networked devices have to be very easy to install. Wireless routers are the most returned consumer electronic item. People buy one because they want a wireless network at home, find that it does not work "out of the box," and then return it rather than listen to elevator music while on hold on the technical helpline.

Second, the network and devices have to be foolproof in operation. Air conditioners used to have one knob with four settings: OFF, LOW, MEDIUM, and HIGH. Now they have 30-page manuals. Once they are networked, expect the chapter on security alone to be 30 pages. This is a problem because only computer users are accustomed to putting up with products that do not work; the car-, television-, and refrigerator-buying public is far less tolerant. They expect products to work 100% without the need to hire a geek.

Third, low price is essential for success. People will not pay a $50 premium for an Internet thermostat because few people regard monitoring their home temperature from work that important. For $5 extra, though, it might sell.

Fourth, it must be possible to start out with one or two devices and expand the reach of the network gradually. This means no format wars. Telling consumers to buy peripherals with IEEE 1394 (FireWire) interfaces and a few years later retracting that and saying USB 2.0 is the interface-of-the-month and then switching that to 802.11g—oops, no, make that 802.11n—I mean 802.16 (different wireless networks)—is going to make consumers very skittish. The network interface will have to remain stable for decades, like the television broadcasting standards.

Fifth, security and reliability will be very important. Losing a few files to an email virus is one thing; having a burglar disarm your security system from his mobile computer and then plunder your house is something quite different.

An interesting question is whether home networks will be wired or wireless. Convenience and cost favors wireless networking because there are no wires to fit, or worse, retrofit. Security favors wired networking because the radio waves that wireless networks use are quite good at going through walls. Not everyone is overjoyed at the thought of having the neighbors piggybacking on their Internet connection and reading their email. In Chap. 8 we will study how encryption can be used to provide security, but it is easier said than done with inexperienced users.

A third option that may be appealing is to reuse the networks that are already in the home. The obvious candidate is the electric wires that are installed throughout the house. **Power-line networks** let devices that plug into outlets broadcast information throughout the house. You have to plug in the TV anyway, and this way it can get Internet connectivity at the same time. The difficulty is

how to carry both power and data signals at the same time. Part of the answer is that they use different frequency bands.

In short, home LANs offer many opportunities and challenges. Most of the latter relate to the need for the networks to be easy to manage, dependable, and secure, especially in the hands of nontechnical users, as well as low cost.

1.2.3 Metropolitan Area Networks

A **MAN** (**Metropolitan Area Network**) covers a city. The best-known examples of MANs are the cable television networks available in many cities. These systems grew from earlier community antenna systems used in areas with poor over-the-air television reception. In those early systems, a large antenna was placed on top of a nearby hill and a signal was then piped to the subscribers' houses.

At first, these were locally designed, ad hoc systems. Then companies began jumping into the business, getting contracts from local governments to wire up entire cities. The next step was television programming and even entire channels designed for cable only. Often these channels were highly specialized, such as all news, all sports, all cooking, all gardening, and so on. But from their inception until the late 1990s, they were intended for television reception only.

When the Internet began attracting a mass audience, the cable TV network operators began to realize that with some changes to the system, they could provide two-way Internet service in unused parts of the spectrum. At that point, the cable TV system began to morph from simply a way to distribute television to a metropolitan area network. To a first approximation, a MAN might look something like the system shown in Fig. 1-9. In this figure we see both television signals and Internet being fed into the centralized **cable headend** for subsequent distribution to people's homes. We will come back to this subject in detail in Chap. 2.

Cable television is not the only MAN, though. Recent developments in high-speed wireless Internet access have resulted in another MAN, which has been standardized as IEEE 802.16 and is popularly known as **WiMAX**. We will look at it in Chap. 4.

1.2.4 Wide Area Networks

A **WAN** (**Wide Area Network**) spans a large geographical area, often a country or continent. We will begin our discussion with wired WANs, using the example of a company with branch offices in different cities.

The WAN in Fig. 1-10 is a network that connects offices in Perth, Melbourne, and Brisbane. Each of these offices contains computers intended for running user (i.e., application) programs. We will follow traditional usage and call these machines **hosts**. The rest of the network that connects these hosts is then called the

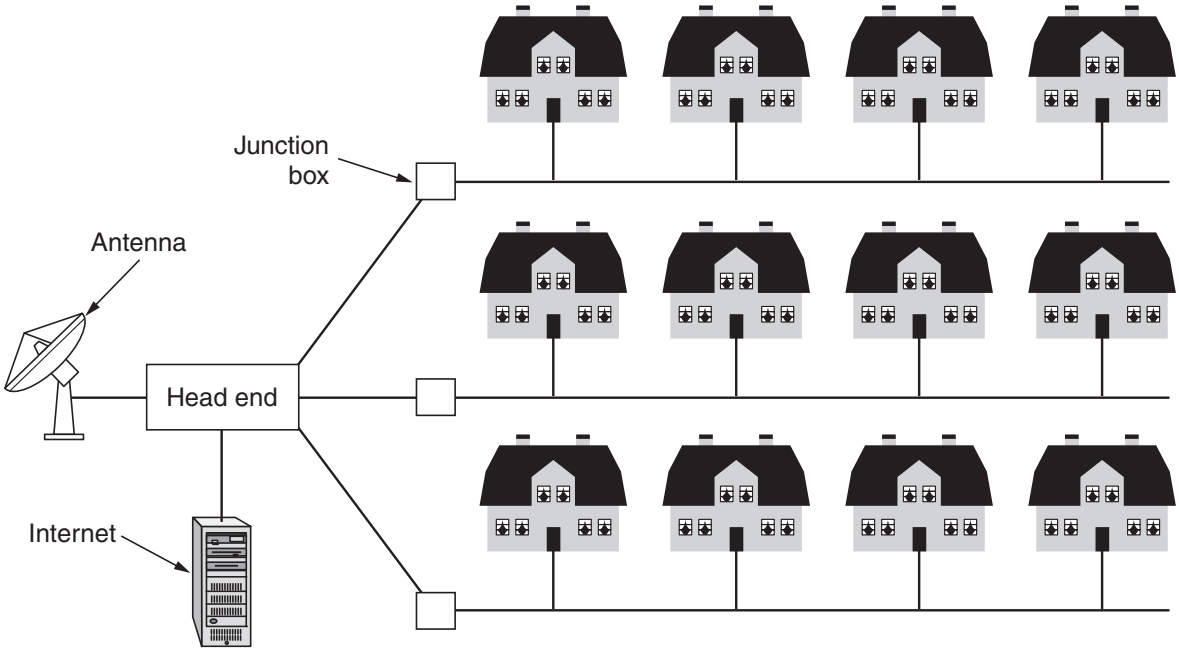


Figure 1-9. A metropolitan area network based on cable TV.

communication subnet, or just **subnet** for short. The job of the subnet is to carry messages from host to host, just as the telephone system carries words (really just sounds) from speaker to listener.

In most WANs, the subnet consists of two distinct components: transmission lines and switching elements. **Transmission lines** move bits between machines. They can be made of copper wire, optical fiber, or even radio links. Most companies do not have transmission lines lying about, so instead they lease the lines from a telecommunications company. **Switching elements**, or just **switches**, are specialized computers that connect two or more transmission lines. When data arrive on an incoming line, the switching element must choose an outgoing line on which to forward them. These switching computers have been called by various names in the past; the name **router** is now most commonly used. Unfortunately, some people pronounce it "rooter" while others have it rhyme with "doubter." Determining the correct pronunciation will be left as an exercise for the reader. (Note: the perceived correct answer may depend on where you live.)

A short comment about the term "subnet" is in order here. Originally, its **only** meaning was the collection of routers and communication lines that moved packets from the source host to the destination host. Readers should be aware that it has acquired a second, more recent meaning in conjunction with network addressing. We will discuss that meaning in Chap. 5 and stick with the original meaning (a collection of lines and routers) until then.

The WAN as we have described it looks similar to a large wired LAN, but there are some important differences that go beyond long wires. Usually in a WAN, the hosts and subnet are owned and operated by different people. In our

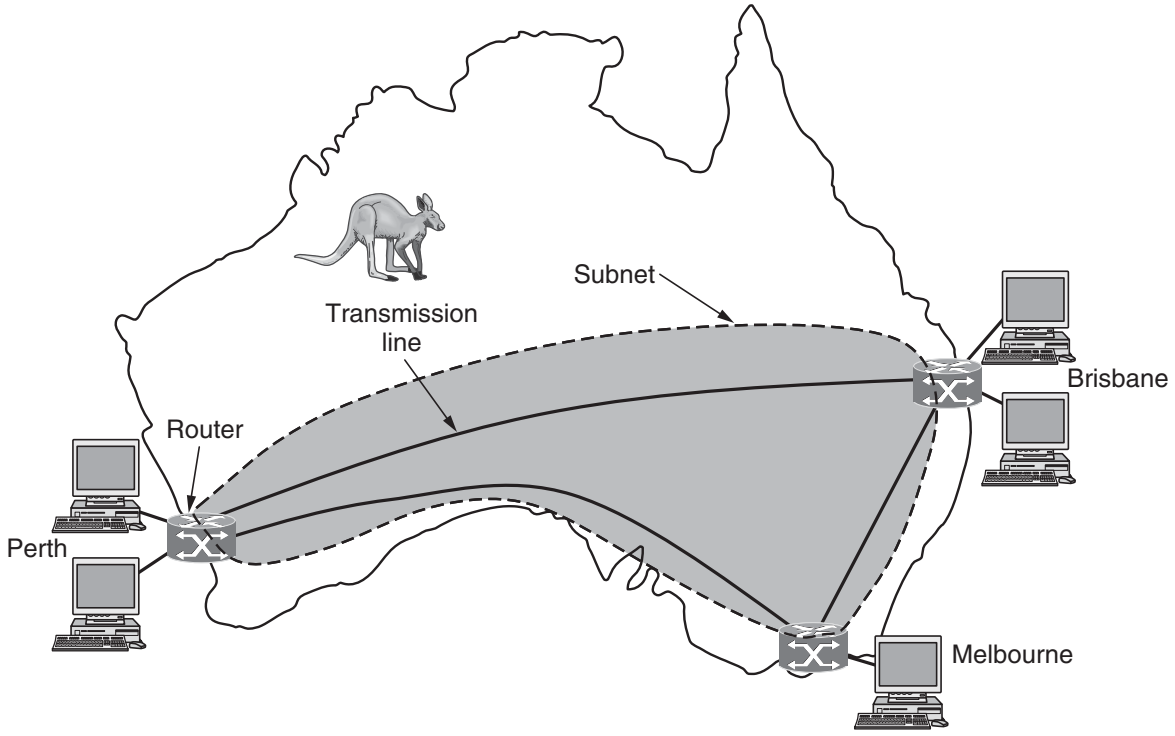


Figure 1-10. WAN that connects three branch offices in Australia.

example, the employees might be responsible for their own computers, while the company's IT department is in charge of the rest of the network. We will see clearer boundaries in the coming examples, in which the network provider or telephone company operates the subnet. Separation of the pure communication aspects of the network (the subnet) from the application aspects (the hosts) greatly simplifies the overall network design.

A second difference is that the routers will usually connect different kinds of networking technology. The networks inside the offices may be switched Ethernet, for example, while the long-distance transmission lines may be SONET links (which we will cover in Chap. 2). Some device needs to join them. The astute reader will notice that this goes beyond our definition of a network. This means that many WANs will in fact be **internetworks**, or composite networks that are made up of more than one network. We will have more to say about internetworks in the next section.

A final difference is in what is connected to the subnet. This could be individual computers, as was the case for connecting to LANs, or it could be entire LANs. This is how larger networks are built from smaller ones. As far as the subnet is concerned, it does the same job.

We are now in a position to look at two other varieties of WANs. First, rather than lease dedicated transmission lines, a company might connect its offices to the Internet This allows connections to be made between the offices as virtual links

that use the underlying capacity of the Internet. This arrangement, shown in Fig. 1-11, is called a **VPN** (**Virtual Private Network**). Compared to the dedicated arrangement, a VPN has the usual advantage of virtualization, which is that it provides flexible reuse of a resource (Internet connectivity). Consider how easy it is to add a fourth office to see this. A VPN also has the usual disadvantage of virtualization, which is a lack of control over the underlying resources. With a dedicated line, the capacity is clear. With a VPN your mileage may vary with your Internet service.

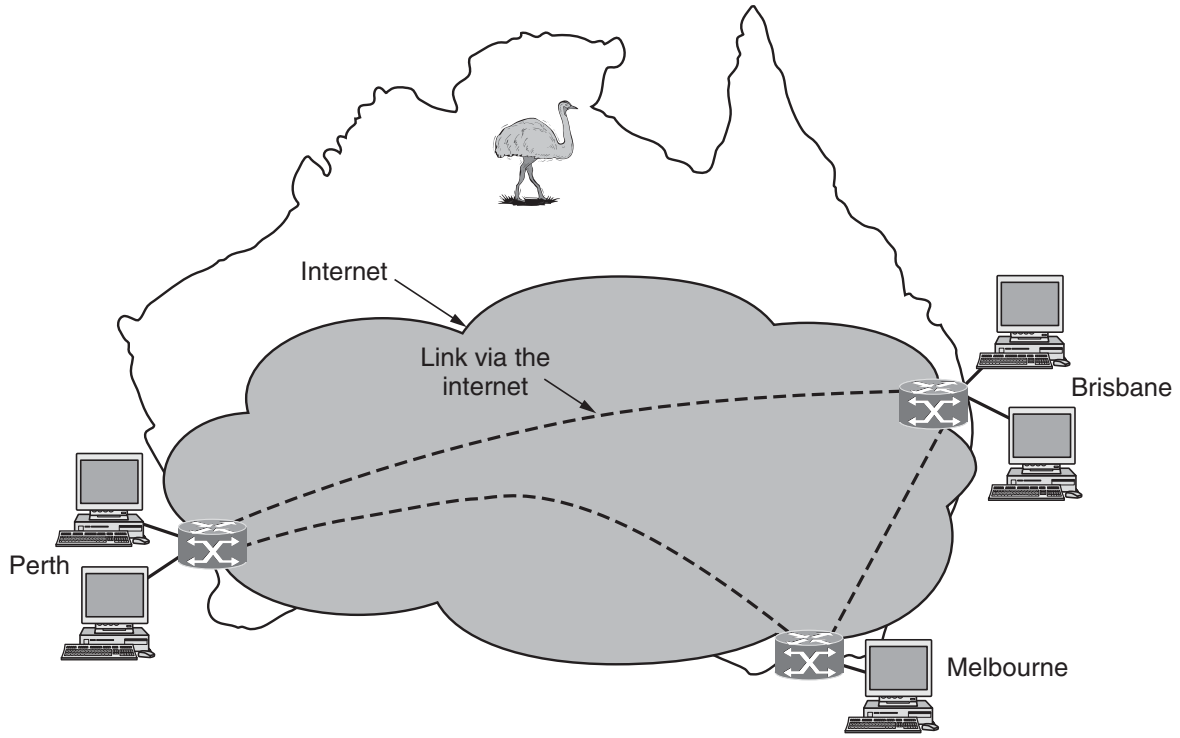


Figure 1-11. WAN using a virtual private network.

The second variation is that the subnet may be run by a different company. The subnet operator is known as a **network service provider** and the offices are its customers. This structure is shown in Fig. 1-12. The subnet operator will connect to other customers too, as long as they can pay and it can provide service. Since it would be a disappointing network service if the customers could only send packets to each other, the subnet operator will also connect to other networks that are part of the Internet. Such a subnet operator is called an **ISP** (**Internet Service Provider**) and the subnet is an **ISP network**. Its customers who connect to the ISP receive Internet service.

We can use the ISP network to preview some key issues that we will study in later chapters. In most WANs, the network contains many transmission lines, each connecting a pair of routers. If two routers that do not share a transmission line wish to communicate, they must do this indirectly, via other routers. There

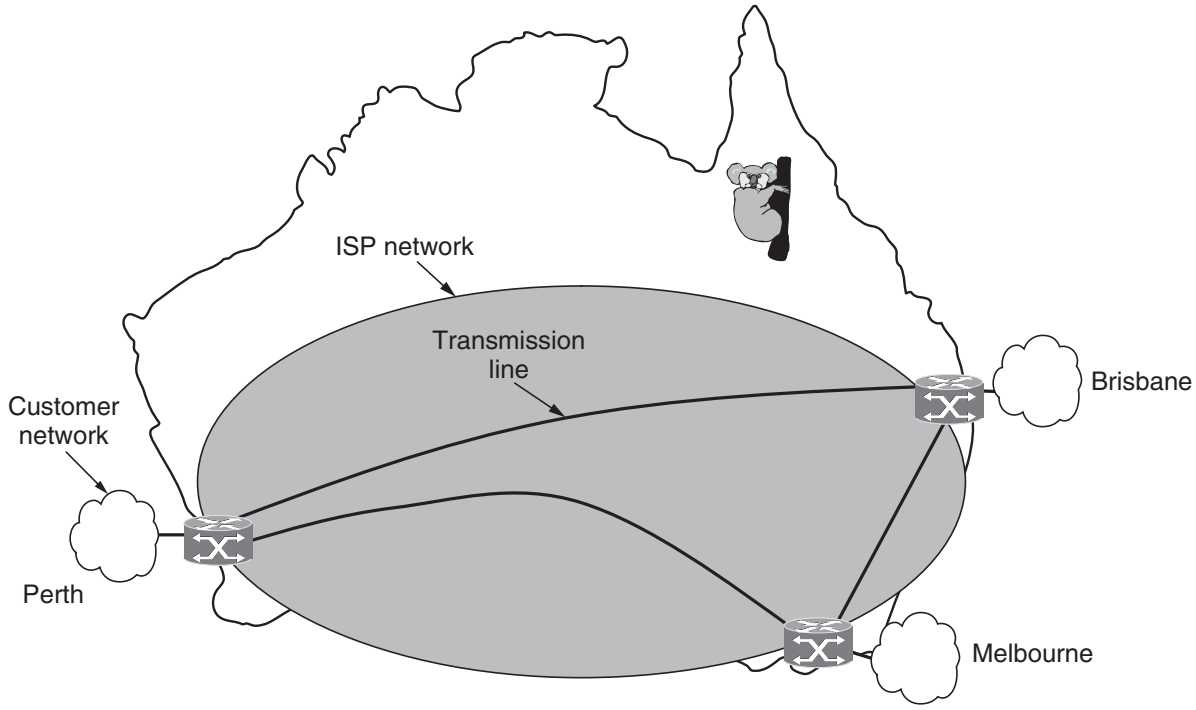


Figure 1-12. WAN using an ISP network.

may be many paths in the network that connect these two routers. How the network makes the decision as to which path to use is called the **routing algorithm**. Many such algorithms exist. How each router makes the decision as to where to send a packet next is called the **forwarding algorithm**. Many of them exist too. We will study some of both types in detail in Chap. 5.

Other kinds of WANs make heavy use of wireless technologies. In satellite systems, each computer on the ground has an antenna through which it can send data to and receive data from to a satellite in orbit. All computers can hear the output *from* the satellite, and in some cases they can also hear the upward transmissions of their fellow computers *to* the satellite as well. Satellite networks are inherently broadcast and are most useful when the broadcast property is important.

The cellular telephone network is another example of a WAN that uses wireless technology. This system has already gone through three generations and a fourth one is on the horizon. The first generation was analog and for voice only. The second generation was digital and for voice only. The third generation is digital and is for both voice and data. Each cellular base station covers a distance much larger than a wireless LAN, with a range measured in kilometers rather than tens of meters. The base stations are connected to each other by a backbone network that is usually wired. The data rates of cellular networks are often on the order of 1 Mbps, much smaller than a wireless LAN that can range up to on the order of 100 Mbps. We will have a lot to say about these networks in Chap. 2.

1.2.5 Internetworks

Many networks exist in the world, often with different hardware and software. People connected to one network often want to communicate with people attached to a different one. The fulfillment of this desire requires that different, and frequently incompatible, networks be connected. A collection of interconnected networks is called an **internetwork** or **internet**. These terms will be used in a generic sense, in contrast to the worldwide Internet (which is one specific internet), which we will always capitalize. The Internet uses ISP networks to connect enterprise networks, home networks, and many other networks. We will look at the Internet in great detail later in this book.

Subnets, networks, and internetworks are often confused. The term "subnet" makes the most sense in the context of a wide area network, where it refers to the collection of routers and communication lines owned by the network operator. As an analogy, the telephone system consists of telephone switching offices connected to one another by high-speed lines, and to houses and businesses by low-speed lines. These lines and equipment, owned and managed by the telephone company, form the subnet of the telephone system. The telephones themselves (the hosts in this analogy) are not part of the subnet.

A network is formed by the combination of a subnet and its hosts. However, the word "network" is often used in a loose sense as well. A subnet might be described as a network, as in the case of the "ISP network" of Fig. 1-12. An internetwork might also be described as a network, as in the case of the WAN in Fig. 1-10. We will follow similar practice, and if we are distinguishing a network from other arrangements, we will stick with our original definition of a collection of computers interconnected by a single technology.

Let us say more about what constitutes an internetwork. We know that an internet is formed when distinct networks are interconnected. In our view, connecting a LAN and a WAN or connecting two LANs is the usual way to form an internetwork, but there is little agreement in the industry over terminology in this area. There are two rules of thumb that are useful. First, if different organizations have paid to construct different parts of the network and each maintains its part, we have an internetwork rather than a single network. Second, if the underlying technology is different in different parts (e.g., broadcast versus point-to-point and wired versus wireless), we probably have an internetwork.

To go deeper, we need to talk about how two different networks can be connected. The general name for a machine that makes a connection between two or more networks and provides the necessary translation, both in terms of hardware and software, is a **gateway**. Gateways are distinguished by the layer at which they operate in the protocol hierarchy. We will have much more to say about layers and protocol hierarchies starting in the next section, but for now imagine that higher layers are more tied to applications, such as the Web, and lower layers are more tied to transmission links, such as Ethernet.

Since the benefit of forming an internet is to connect computers across networks, we do not want to use too low-level a gateway or we will be unable to make connections between different kinds of networks. We do not want to use too high-level a gateway either, or the connection will only work for particular applications. The level in the middle that is "just right" is often called the network layer, and a router is a gateway that switches packets at the network layer. We can now spot an internet by finding a network that has routers.

1.3 NETWORK SOFTWARE

The first computer networks were designed with the hardware as the main concern and the software as an afterthought. This strategy no longer works. Network software is now highly structured. In the following sections we examine the software structuring technique in some detail. The approach described here forms the keystone of the entire book and will occur repeatedly later on.

1.3.1 Protocol Hierarchies

To reduce their design complexity, most networks are organized as a stack of **layers** or **levels**, each one built upon the one below it. The number of layers, the name of each layer, the contents of each layer, and the function of each layer differ from network to network. The purpose of each layer is to offer certain services to the higher layers while shielding those layers from the details of how the offered services are actually implemented. In a sense, each layer is a kind of virtual machine, offering certain services to the layer above it.

This concept is actually a familiar one and is used throughout computer science, where it is variously known as information hiding, abstract data types, data encapsulation, and object-oriented programming. The fundamental idea is that a particular piece of software (or hardware) provides a service to its users but keeps the details of its internal state and algorithms hidden from them.

When layer *n* on one machine carries on a conversation with layer *n* on another machine, the rules and conventions used in this conversation are collectively known as the layer *n* protocol. Basically, a **protocol** is an agreement between the communicating parties on how communication is to proceed. As an analogy, when a woman is introduced to a man, she may choose to stick out her hand. He, in turn, may decide to either shake it or kiss it, depending, for example, on whether she is an American lawyer at a business meeting or a European princess at a formal ball. Violating the protocol will make communication more difficult, if not completely impossible.

A five-layer network is illustrated in Fig. 1-13. The entities comprising the corresponding layers on different machines are called **peers**. The peers may be

software processes, hardware devices, or even human beings. In other words, it is the peers that communicate by using the protocol to talk to each other.

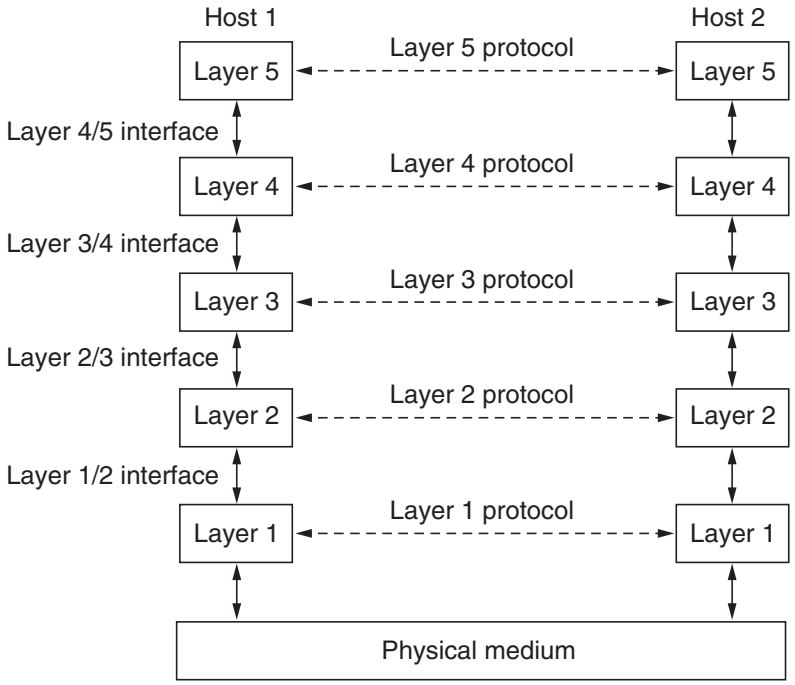


Figure 1-13. Layers, protocols, and interfaces.

In reality, no data are directly transferred from layer *n* on one machine to layer *n* on another machine. Instead, each layer passes data and control information to the layer immediately below it, until the lowest layer is reached. Below layer 1 is the **physical medium** through which actual communication occurs. In Fig. 1-13, virtual communication is shown by dotted lines and physical communication by solid lines.

Between each pair of adjacent layers is an **interface**. The interface defines which primitive operations and services the lower layer makes available to the upper one. When network designers decide how many layers to include in a network and what each one should do, one of the most important considerations is defining clean interfaces between the layers. Doing so, in turn, requires that each layer perform a specific collection of well-understood functions. In addition to minimizing the amount of information that must be passed between layers, clear-cut interfaces also make it simpler to replace one layer with a completely different protocol or implementation (e.g., replacing all the telephone lines by satellite channels) because all that is required of the new protocol or implementation is that it offer exactly the same set of services to its upstairs neighbor as the old one did. It is common that different hosts use different implementations of the same protocol (often written by different companies). In fact, the protocol itself can change in some layer without the layers above and below it even noticing.

A set of layers and protocols is called a **network architecture**. The specification of an architecture must contain enough information to allow an implementer to write the program or build the hardware for each layer so that it will correctly obey the appropriate protocol. Neither the details of the implementation nor the specification of the interfaces is part of the architecture because these are hidden away inside the machines and not visible from the outside. It is not even necessary that the interfaces on all machines in a network be the same, provided that each machine can correctly use all the protocols. A list of the protocols used by a certain system, one protocol per layer, is called a **protocol stack**. Network architectures, protocol stacks, and the protocols themselves are the principal subjects of this book.

An analogy may help explain the idea of multilayer communication. Imagine two philosophers (peer processes in layer 3), one of whom speaks Urdu and English and one of whom speaks Chinese and French. Since they have no common language, they each engage a translator (peer processes at layer 2), each of whom in turn contacts a secretary (peer processes in layer 1). Philosopher 1 wishes to convey his affection for *oryctolagus cuniculus* to his peer. To do so, he passes a message (in English) across the 2/3 interface to his translator, saying "I like rabbits," as illustrated in Fig. 1-14. The translators have agreed on a neutral language known to both of them, Dutch, so the message is converted to "Ik vind konijnen leuk." The choice of the language is the layer 2 protocol and is up to the layer 2 peer processes.

The translator then gives the message to a secretary for transmission, for example, by email (the layer 1 protocol). When the message arrives at the other secretary, it is passed to the local translator, who translates it into French and passes it across the 2/3 interface to the second philosopher. Note that each protocol is completely independent of the other ones as long as the interfaces are not changed. The translators can switch from Dutch to, say, Finnish, at will, provided that they both agree and neither changes his interface with either layer 1 or layer 3. Similarly, the secretaries can switch from email to telephone without disturbing (or even informing) the other layers. Each process may add some information intended only for its peer. This information is not passed up to the layer above.

Now consider a more technical example: how to provide communication to the top layer of the five-layer network in Fig. 1-15. A message, *M*, is produced by an application process running in layer 5 and given to layer 4 for transmission. Layer 4 puts a **header** in front of the message to identify the message and passes the result to layer 3. The header includes control information, such as addresses, to allow layer 4 on the destination machine to deliver the message. Other examples of control information used in some layers are sequence numbers (in case the lower layer does not preserve message order), sizes, and times.

In many networks, no limit is placed on the size of messages transmitted in the layer 4 protocol but there is nearly always a limit imposed by the layer 3 protocol. Consequently, layer 3 must break up the incoming messages into smaller

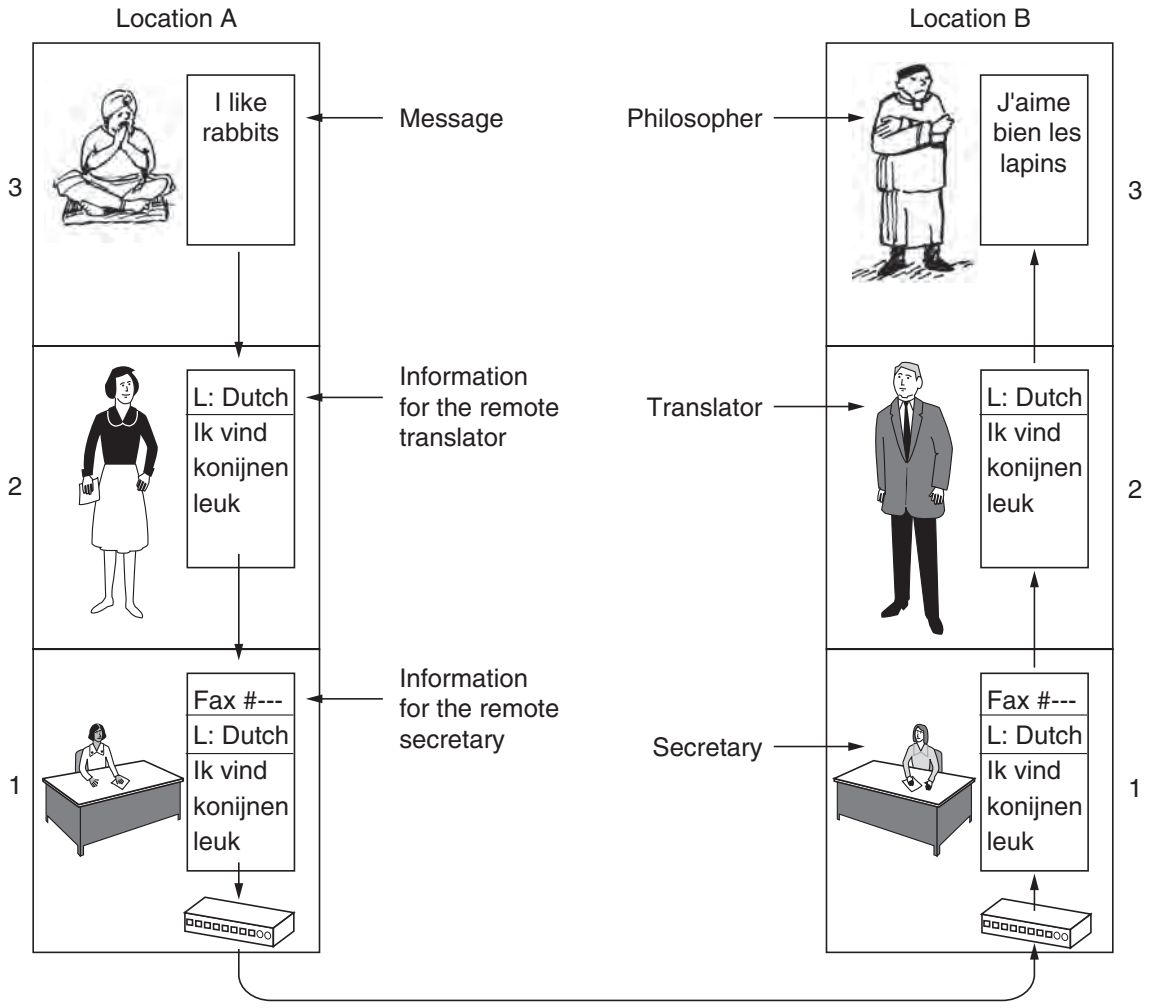


Figure 1-14. The philosopher-translator-secretary architecture.

units, packets, prepending a layer 3 header to each packet. In this example, M is split into two parts, M_1 and M_2, that will be transmitted separately.

Layer 3 decides which of the outgoing lines to use and passes the packets to layer 2. Layer 2 adds to each piece not only a header but also a trailer, and gives the resulting unit to layer 1 for physical transmission. At the receiving machine the message moves upward, from layer to layer, with headers being stripped off as it progresses. None of the headers for layers below n are passed up to layer n.

The important thing to understand about Fig. 1-15 is the relation between the virtual and actual communication and the difference between protocols and interfaces. The peer processes in layer 4, for example, conceptually think of their communication as being "horizontal," using the layer 4 protocol. Each one is likely to have procedures called something like *SendToOtherSide* and *GetFromOtherSide*, even though these procedures actually communicate with lower layers across the 3/4 interface, and not with the other side.

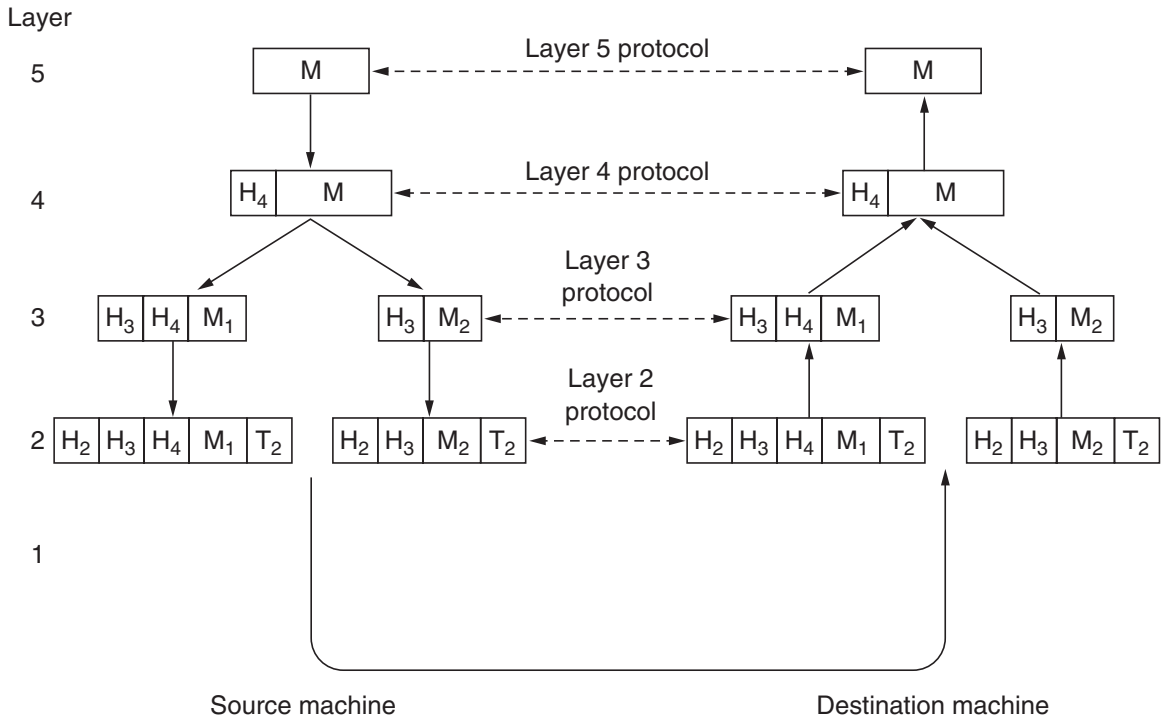


Figure 1-15. Example information flow supporting virtual communication in layer 5.

The peer process abstraction is crucial to all network design. Using it, the unmanageable task of designing the complete network can be broken into several smaller, manageable design problems, namely, the design of the individual layers.

Although Sec. 1.3 is called "Network Software," it is worth pointing out that the lower layers of a protocol hierarchy are frequently implemented in hardware or firmware. Nevertheless, complex protocol algorithms are involved, even if they are embedded (in whole or in part) in hardware.

1.3.2 Design Issues for the Layers

Some of the key design issues that occur in computer networks will come up in layer after layer. Below, we will briefly mention the more important ones.

Reliability is the design issue of making a network that operates correctly even though it is made up of a collection of components that are themselves unreliable. Think about the bits of a packet traveling through the network. There is a chance that some of these bits will be received damaged (inverted) due to fluke electrical noise, random wireless signals, hardware flaws, software bugs and so on. How is it possible that we find and fix these errors?

One mechanism for finding errors in received information uses codes for **error detection**. Information that is incorrectly received can then be retransmitted

until it is received correctly. More powerful codes allow for **error correction**, where the correct message is recovered from the possibly incorrect bits that were originally received. Both of these mechanisms work by adding redundant information. They are used at low layers, to protect packets sent over individual links, and high layers, to check that the right contents were received.

Another reliability issue is finding a working path through a network. Often there are multiple paths between a source and destination, and in a large network, there may be some links or routers that are broken. Suppose that the network is down in Germany. Packets sent from London to Rome via Germany will not get through, but we could instead send packets from London to Rome via Paris. The network should automatically make this decision. This topic is called **routing**.

A second design issue concerns the evolution of the network. Over time, networks grow larger and new designs emerge that need to be connected to the existing network. We have recently seen the key structuring mechanism used to support change by dividing the overall problem and hiding implementation details: **protocol layering**. There are many other strategies as well.

Since there are many computers on the network, every layer needs a mechanism for identifying the senders and receivers that are involved in a particular message. This mechanism is called **addressing** or **naming**, in the low and high layers, respectively.

An aspect of growth is that different network technologies often have different limitations. For example, not all communication channels preserve the order of messages sent on them, leading to solutions that number messages. Another example is differences in the maximum size of a message that the networks can transmit. This leads to mechanisms for disassembling, transmitting, and then reassembling messages. This overall topic is called **internetworking**.

When networks get large, new problems arise. Cities can have traffic jams, a shortage of telephone numbers, and it is easy to get lost. Not many people have these problems in their own neighborhood, but citywide they may be a big issue. Designs that continue to work well when the network gets large are said to be **scalable**.

A third design issue is resource allocation. Networks provide a service to hosts from their underlying resources, such as the capacity of transmission lines. To do this well, they need mechanisms that divide their resources so that one host does not interfere with another too much.

Many designs share network bandwidth dynamically, according to the short-term needs of hosts, rather than by giving each host a fixed fraction of the bandwidth that it may or may not use. This design is called **statistical multiplexing**, meaning sharing based on the statistics of demand. It can be applied at low layers for a single link, or at high layers for a network or even applications that use the network.

An allocation problem that occurs at every level is how to keep a fast sender from swamping a slow receiver with data. Feedback from the receiver to the

sender is often used. This subject is called **flow control**. Sometimes the problem is that the network is oversubscribed because too many computers want to send too much traffic, and the network cannot deliver it all. This overloading of the network is called **congestion**. One strategy is for each computer to reduce its demand when it experiences congestion. It, too, can be used in all layers.

It is interesting to observe that the network has more resources to offer than simply bandwidth. For uses such as carrying live video, the timeliness of delivery matters a great deal. Most networks must provide service to applications that want this **real-time** delivery at the same time that they provide service to applications that want high throughput. **Quality of service** is the name given to mechanisms that reconcile these competing demands.

The last major design issue is to secure the network by defending it against different kinds of threats. One of the threats we have mentioned previously is that of eavesdropping on communications. Mechanisms that provide **confidentiality** defend against this threat, and they are used in multiple layers. Mechanisms for **authentication** prevent someone from impersonating someone else. They might be used to tell fake banking Web sites from the real one, or to let the cellular network check that a call is really coming from your phone so that you will pay the bill. Other mechanisms for **integrity** prevent surreptitious changes to messages, such as altering "debit my account $10" to "debit my account $1000." All of these designs are based on cryptography, which we shall study in Chap. 8.

1.3.3 Connection-Oriented Versus Connectionless Service

Layers can offer two different types of service to the layers above them: connection-oriented and connectionless. In this section we will look at these two types and examine the differences between them.

Connection-oriented service is modeled after the telephone system. To talk to someone, you pick up the phone, dial the number, talk, and then hang up. Similarly, to use a connection-oriented network service, the service user first establishes a connection, uses the connection, and then releases the connection. The essential aspect of a connection is that it acts like a tube: the sender pushes objects (bits) in at one end, and the receiver takes them out at the other end. In most cases the order is preserved so that the bits arrive in the order they were sent.

In some cases when a connection is established, the sender, receiver, and subnet conduct a **negotiation** about the parameters to be used, such as maximum message size, quality of service required, and other issues. Typically, one side makes a proposal and the other side can accept it, reject it, or make a counterproposal. A **circuit** is another name for a connection with associated resources, such as a fixed bandwidth. This dates from the telephone network in which a circuit was a path over copper wire that carried a phone conversation.

In contrast to connection-oriented service, **connectionless** service is modeled after the postal system. Each message (letter) carries the full destination address,

and each one is routed through the intermediate nodes inside the system independent of all the subsequent messages. There are different names for messages in different contexts; a **packet** is a message at the network layer. When the intermediate nodes receive a message in full before sending it on to the next node, this is called **store-and-forward switching**. The alternative, in which the onward transmission of a message at a node starts before it is completely received by the node, is called **cut-through switching**. Normally, when two messages are sent to the same destination, the first one sent will be the first one to arrive. However, it is possible that the first one sent can be delayed so that the second one arrives first.

Each kind of service can further be characterized by its reliability. Some services are reliable in the sense that they never lose data. Usually, a reliable service is implemented by having the receiver acknowledge the receipt of each message so the sender is sure that it arrived. The acknowledgement process introduces overhead and delays, which are often worth it but are sometimes undesirable.

A typical situation in which a reliable connection-oriented service is appropriate is file transfer. The owner of the file wants to be sure that all the bits arrive correctly and in the same order they were sent. Very few file transfer customers would prefer a service that occasionally scrambles or loses a few bits, even if it is much faster.

Reliable connection-oriented service has two minor variations: message sequences and byte streams. In the former variant, the message boundaries are preserved. When two 1024-byte messages are sent, they arrive as two distinct 1024-byte messages, never as one 2048-byte message. In the latter, the connection is simply a stream of bytes, with no message boundaries. When 2048 bytes arrive at the receiver, there is no way to tell if they were sent as one 2048-byte message, two 1024-byte messages, or 2048 1-byte messages. If the pages of a book are sent over a network to a phototypesetter as separate messages, it might be important to preserve the message boundaries. On the other hand, to download a DVD movie, a byte stream from the server to the user's computer is all that is needed. Message boundaries within the movie are not relevant.

For some applications, the transit delays introduced by acknowledgements are unacceptable. One such application is digitized voice traffic for **voice over IP**. It is less disruptive for telephone users to hear a bit of noise on the line from time to time than to experience a delay waiting for acknowledgements. Similarly, when transmitting a video conference, having a few pixels wrong is no problem, but having the image jerk along as the flow stops and starts to correct errors is irritating.

Not all applications require connections. For example, spammers send electronic junk-mail to many recipients. The spammer probably does not want to go to the trouble of setting up and later tearing down a connection to a recipient just to send them one item. Nor is 100 percent reliable delivery essential, especially if it costs more. All that is needed is a way to send a single message that has a high

probability of arrival, but no guarantee. Unreliable (meaning not acknowledged) connectionless service is often called **datagram** service, in analogy with telegram service, which also does not return an acknowledgement to the sender. Despite it being unreliable, it is the dominant form in most networks for reasons that will become clear later

In other situations, the convenience of not having to establish a connection to send one message is desired, but reliability is essential. The **acknowledged datagram** service can be provided for these applications. It is like sending a registered letter and requesting a return receipt. When the receipt comes back, the sender is absolutely sure that the letter was delivered to the intended party and not lost along the way. Text messaging on mobile phones is an example.

Still another service is the **request-reply** service. In this service the sender transmits a single datagram containing a request; the reply contains the answer. Request-reply is commonly used to implement communication in the client-server model: the client issues a request and the server responds to it. For example, a mobile phone client might send a query to a map server to retrieve the map data for the current location. Figure 1-16 summarizes the types of services discussed above.

	Service	Example
Connection-oriented	Reliable message stream	Sequence of pages
	Reliable byte stream	Movie download
	Unreliable connection	Voice over IP
Connectionless	Unreliable datagram	Electronic junk mail
	Acknowledged datagram	Text messaging
	Request-reply	Database query

Figure 1-16. Six different types of service.

The concept of using unreliable communication may be confusing at first. After all, why would anyone actually prefer unreliable communication to reliable communication? First of all, reliable communication (in our sense, that is, acknowledged) may not be available in a given layer. For example, Ethernet does not provide reliable communication. Packets can occasionally be damaged in transit. It is up to higher protocol levels to recover from this problem. In particular, many reliable services are built on top of an unreliable datagram service. Second, the delays inherent in providing a reliable service may be unacceptable, especially in real-time applications such as multimedia. For these reasons, both reliable and unreliable communication coexist.

1.3.4 Service Primitives

A service is formally specified by a set of **primitives** (operations) available to user processes to access the service. These primitives tell the service to perform some action or report on an action taken by a peer entity. If the protocol stack is located in the operating system, as it often is, the primitives are normally system calls. These calls cause a trap to kernel mode, which then turns control of the machine over to the operating system to send the necessary packets.

The set of primitives available depends on the nature of the service being provided. The primitives for connection-oriented service are different from those of connectionless service. As a minimal example of the service primitives that might provide a reliable byte stream, consider the primitives listed in Fig. 1-17. They will be familiar to fans of the Berkeley socket interface, as the primitives are a simplified version of that interface.

Primitive	Meaning
LISTEN	Block waiting for an incoming connection
CONNECT	Establish a connection with a waiting peer
ACCEPT	Accept an incoming connection from a peer
RECEIVE	Block waiting for an incoming message
SEND	Send a message to the peer
DISCONNECT	Terminate a connection

Figure 1-17. Six service primitives that provide a simple connection-oriented service.

These primitives might be used for a request-reply interaction in a client-server environment. To illustrate how, We sketch a simple protocol that implements the service using acknowledged datagrams.

First, the server executes LISTEN to indicate that it is prepared to accept incoming connections. A common way to implement LISTEN is to make it a blocking system call. After executing the primitive, the server process is blocked until a request for connection appears.

Next, the client process executes CONNECT to establish a connection with the server. The CONNECT call needs to specify who to connect to, so it might have a parameter giving the server's address. The operating system then typically sends a packet to the peer asking it to connect, as shown by (1) in Fig. 1-18. The client process is suspended until there is a response.

When the packet arrives at the server, the operating system sees that the packet is requesting a connection. It checks to see if there is a listener, and if so it unblocks the listener. The server process can then establish the connection with the ACCEPT call. This sends a response (2) back to the client process to accept the

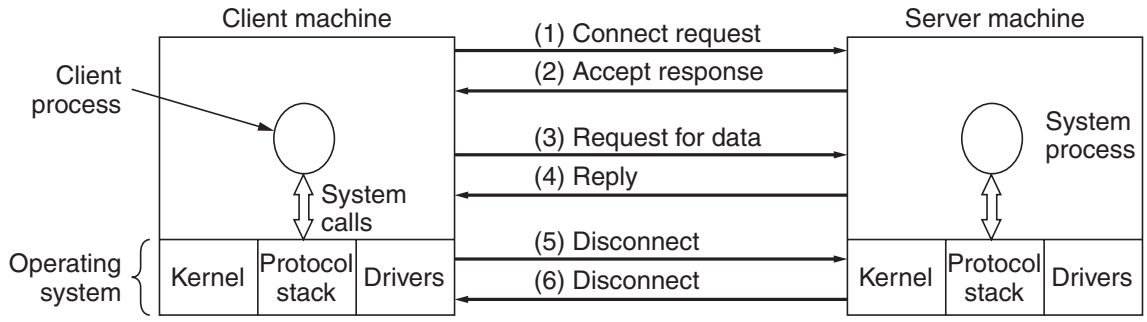


Figure 1-18. A simple client-server interaction using acknowledged datagrams.

connection. The arrival of this response then releases the client. At this point the client and server are both running and they have a connection established.

The obvious analogy between this protocol and real life is a customer (client) calling a company's customer service manager. At the start of the day, the service manager sits next to his telephone in case it rings. Later, a client places a call. When the manager picks up the phone, the connection is established.

The next step is for the server to execute RECEIVE to prepare to accept the first request. Normally, the server does this immediately upon being released from the LISTEN, before the acknowledgement can get back to the client. The RECEIVE call blocks the server.

Then the client executes SEND to transmit its request (3) followed by the execution of RECEIVE to get the reply. The arrival of the request packet at the server machine unblocks the server so it can handle the request. After it has done the work, the server uses SEND to return the answer to the client (4). The arrival of this packet unblocks the client, which can now inspect the answer. If the client has additional requests, it can make them now.

When the client is done, it executes DISCONNECT to terminate the connection (5). Usually, an initial DISCONNECT is a blocking call, suspending the client and sending a packet to the server saying that the connection is no longer needed. When the server gets the packet, it also issues a DISCONNECT of its own, acknowledging the client and releasing the connection (6). When the server's packet gets back to the client machine, the client process is released and the connection is broken. In a nutshell, this is how connection-oriented communication works.

Of course, life is not so simple. Many things can go wrong here. The timing can be wrong (e.g., the CONNECT is done before the LISTEN), packets can get lost, and much more. We will look at these issues in great detail later, but for the moment, Fig. 1-18 briefly summarizes how client-server communication might work with acknowledged datagrams so that we can ignore lost packets.

Given that six packets are required to complete this protocol, one might wonder why a connectionless protocol is not used instead. The answer is that in a perfect world it could be, in which case only two packets would be needed: one

for the request and one for the reply. However, in the face of large messages in either direction (e.g., a megabyte file), transmission errors, and lost packets, the situation changes. If the reply consisted of hundreds of packets, some of which could be lost during transmission, how would the client know if some pieces were missing? How would the client know whether the last packet actually received was really the last packet sent? Suppose the client wanted a second file. How could it tell packet 1 from the second file from a lost packet 1 from the first file that suddenly found its way to the client? In short, in the real world, a simple request-reply protocol over an unreliable network is often inadequate. In Chap. 3 we will study a variety of protocols in detail that overcome these and other problems. For the moment, suffice it to say that having a reliable, ordered byte stream between processes is sometimes very convenient.

1.3.5 The Relationship of Services to Protocols

Services and protocols are distinct concepts. This distinction is so important that we emphasize it again here. A *service* is a set of primitives (operations) that a layer provides to the layer above it. The service defines what operations the layer is prepared to perform on behalf of its users, but it says nothing at all about how these operations are implemented. A service relates to an interface between two layers, with the lower layer being the service provider and the upper layer being the service user.

A *protocol*, in contrast, is a set of rules governing the format and meaning of the packets, or messages that are exchanged by the peer entities within a layer. Entities use protocols to implement their service definitions. They are free to change their protocols at will, provided they do not change the service visible to their users. In this way, the service and the protocol are completely decoupled. This is a key concept that any network designer should understand well.

To repeat this crucial point, services relate to the interfaces between layers, as illustrated in Fig. 1-19. In contrast, protocols relate to the packets sent between peer entities on different machines. It is very important not to confuse the two concepts.

An analogy with programming languages is worth making. A service is like an abstract data type or an object in an object-oriented language. It defines operations that can be performed on an object but does not specify how these operations are implemented. In contrast, a protocol relates to the *implementation* of the service and as such is not visible to the user of the service.

Many older protocols did not distinguish the service from the protocol. In effect, a typical layer might have had a service primitive SEND PACKET with the user providing a pointer to a fully assembled packet. This arrangement meant that all changes to the protocol were immediately visible to the users. Most network designers now regard such a design as a serious blunder.

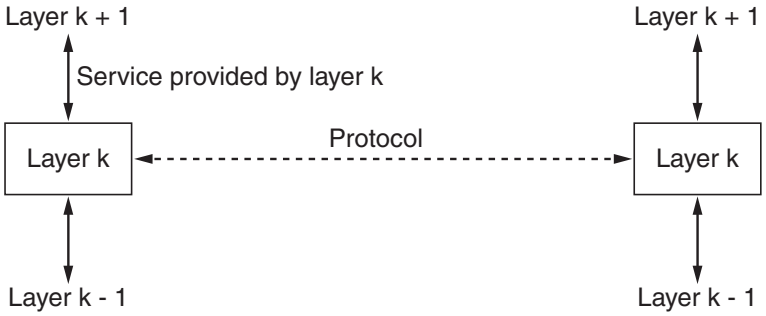


Figure 1-19. The relationship between a service and a protocol.

1.4 REFERENCE MODELS

Now that we have discussed layered networks in the abstract, it is time to look at some examples. We will discuss two important network architectures: the OSI reference model and the TCP/IP reference model. Although the *protocols* associated with the OSI model are not used any more, the *model* itself is actually quite general and still valid, and the features discussed at each layer are still very important. The TCP/IP model has the opposite properties: the model itself is not of much use but the protocols are widely used. For this reason we will look at both of them in detail. Also, sometimes you can learn more from failures than from successes.

1.4.1 The OSI Reference Model

The OSI model (minus the physical medium) is shown in Fig. 1-20. This model is based on a proposal developed by the International Standards Organization (ISO) as a first step toward international standardization of the protocols used in the various layers (Day and Zimmermann, 1983). It was revised in 1995 (Day, 1995). The model is called the ISO **OSI** (**Open Systems Interconnection**) Reference Model because it deals with connecting open systems—that is, systems that are open for communication with other systems. We will just call it the **OSI model** for short.

The OSI model has seven layers. The principles that were applied to arrive at the seven layers can be briefly summarized as follows:

1. A layer should be created where a different abstraction is needed.
2. Each layer should perform a well-defined function.
3. The function of each layer should be chosen with an eye toward defining internationally standardized protocols.

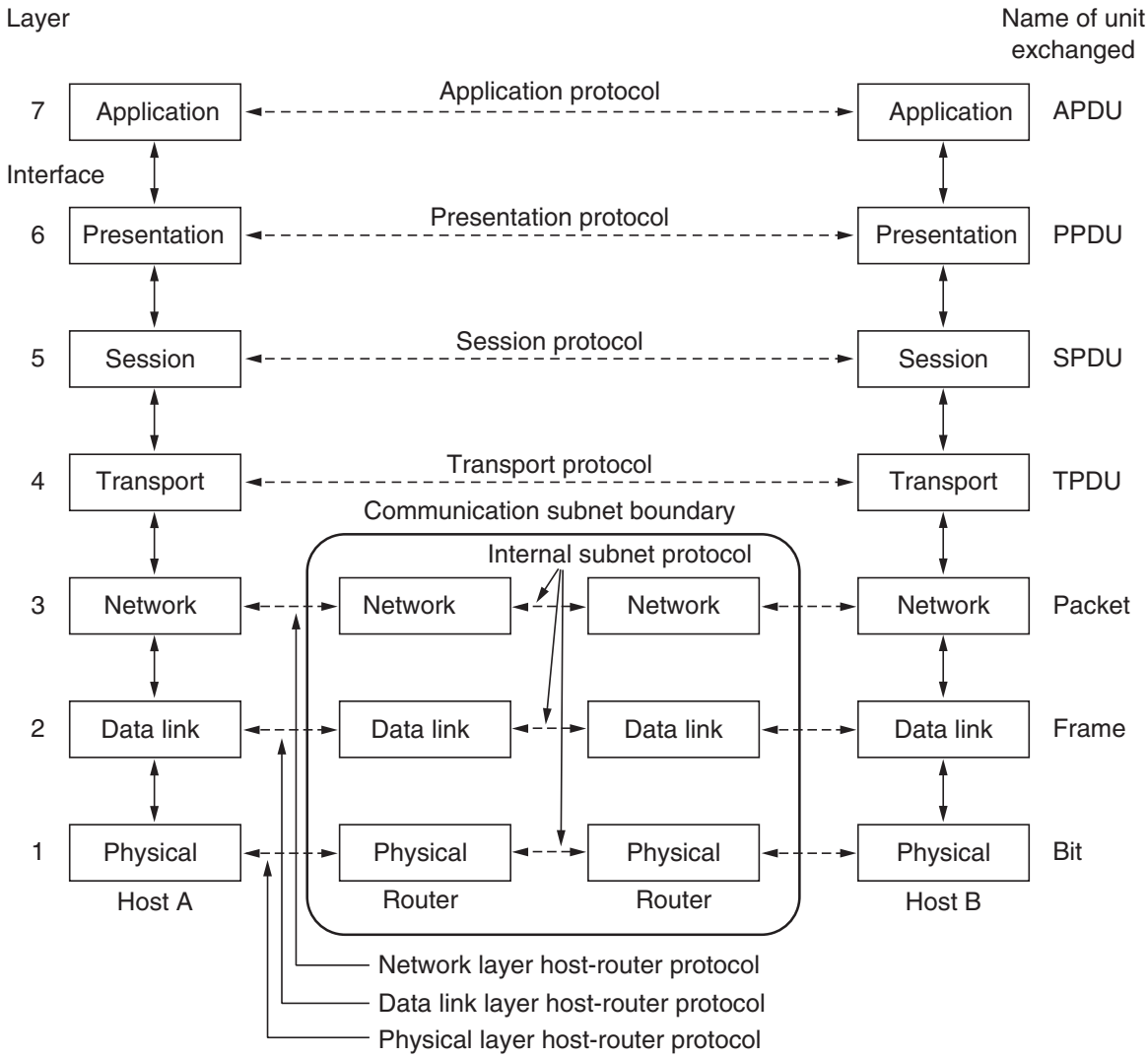


Figure 1-20. The OSI reference model.

4. The layer boundaries should be chosen to minimize the information flow across the interfaces.

5. The number of layers should be large enough that distinct functions need not be thrown together in the same layer out of necessity and small enough that the architecture does not become unwieldy.

Below we will discuss each layer of the model in turn, starting at the bottom layer. Note that the OSI model itself is not a network architecture because it does not specify the exact services and protocols to be used in each layer. It just tells what each layer should do. However, ISO has also produced standards for all the layers, although these are not part of the reference model itself. Each one has been published as a separate international standard. The *model* (in part) is widely used although the associated protocols have been long forgotten.

The Physical Layer

The **physical layer** is concerned with transmitting raw bits over a communication channel. The design issues have to do with making sure that when one side sends a 1 bit it is received by the other side as a 1 bit, not as a 0 bit. Typical questions here are what electrical signals should be used to represent a 1 and a 0, how many nanoseconds a bit lasts, whether transmission may proceed simultaneously in both directions, how the initial connection is established, how it is torn down when both sides are finished, how many pins the network connector has, and what each pin is used for. These design issues largely deal with mechanical, electrical, and timing interfaces, as well as the physical transmission medium, which lies below the physical layer.

The Data Link Layer

The main task of the **data link layer** is to transform a raw transmission facility into a line that appears free of undetected transmission errors. It does so by masking the real errors so the network layer does not see them. It accomplishes this task by having the sender break up the input data into **data frames** (typically a few hundred or a few thousand bytes) and transmit the frames sequentially. If the service is reliable, the receiver confirms correct receipt of each frame by sending back an **acknowledgement frame**.

Another issue that arises in the data link layer (and most of the higher layers as well) is how to keep a fast transmitter from drowning a slow receiver in data. Some traffic regulation mechanism may be needed to let the transmitter know when the receiver can accept more data.

Broadcast networks have an additional issue in the data link layer: how to control access to the shared channel. A special sublayer of the data link layer, the **medium access control** sublayer, deals with this problem.

The Network Layer

The **network layer** controls the operation of the subnet. A key design issue is determining how packets are routed from source to destination. Routes can be based on static tables that are ''wired into'' the network and rarely changed, or more often they can be updated automatically to avoid failed components. They can also be determined at the start of each conversation, for example, a terminal session, such as a login to a remote machine. Finally, they can be highly dynamic, being determined anew for each packet to reflect the current network load.

If too many packets are present in the subnet at the same time, they will get in one another's way, forming bottlenecks. Handling congestion is also a responsibility of the network layer, in conjunction with higher layers that adapt the load

they place on the network. More generally, the quality of service provided (delay, transit time, jitter, etc.) is also a network layer issue.

When a packet has to travel from one network to another to get to its destination, many problems can arise. The addressing used by the second network may be different from that used by the first one. The second one may not accept the packet at all because it is too large. The protocols may differ, and so on. It is up to the network layer to overcome all these problems to allow heterogeneous networks to be interconnected.

In broadcast networks, the routing problem is simple, so the network layer is often thin or even nonexistent.

The Transport Layer

The basic function of the **transport layer** is to accept data from above it, split it up into smaller units if need be, pass these to the network layer, and ensure that the pieces all arrive correctly at the other end. Furthermore, all this must be done efficiently and in a way that isolates the upper layers from the inevitable changes in the hardware technology over the course of time.

The transport layer also determines what type of service to provide to the session layer, and, ultimately, to the users of the network. The most popular type of transport connection is an error-free point-to-point channel that delivers messages or bytes in the order in which they were sent. However, other possible kinds of transport service exist, such as the transporting of isolated messages with no guarantee about the order of delivery, and the broadcasting of messages to multiple destinations. The type of service is determined when the connection is established. (As an aside, an error-free channel is completely impossible to achieve; what people really mean by this term is that the error rate is low enough to ignore in practice.)

The transport layer is a true end-to-end layer; it carries data all the way from the source to the destination. In other words, a program on the source machine carries on a conversation with a similar program on the destination machine, using the message headers and control messages. In the lower layers, each protocols is between a machine and its immediate neighbors, and not between the ultimate source and destination machines, which may be separated by many routers. The difference between layers 1 through 3, which are chained, and layers 4 through 7, which are end-to-end, is illustrated in Fig. 1-20.

The Session Layer

The session layer allows users on different machines to establish **sessions** between them. Sessions offer various services, including **dialog control** (keeping track of whose turn it is to transmit), **token management** (preventing two parties from attempting the same critical operation simultaneously), and **synchronization**

(checkpointing long transmissions to allow them to pick up from where they left off in the event of a crash and subsequent recovery).

The Presentation Layer

Unlike the lower layers, which are mostly concerned with moving bits around, the **presentation layer** is concerned with the syntax and semantics of the information transmitted. In order to make it possible for computers with different internal data representations to communicate, the data structures to be exchanged can be defined in an abstract way, along with a standard encoding to be used ''on the wire.'' The presentation layer manages these abstract data structures and allows higher-level data structures (e.g., banking records) to be defined and exchanged.

The Application Layer

The **application layer** contains a variety of protocols that are commonly needed by users. One widely used application protocol is **HTTP** (**HyperText Transfer Protocol**), which is the basis for the World Wide Web. When a browser wants a Web page, it sends the name of the page it wants to the server hosting the page using HTTP. The server then sends the page back. Other application protocols are used for file transfer, electronic mail, and network news.

1.4.2 The TCP/IP Reference Model

Let us now turn from the OSI reference model to the reference model used in the grandparent of all wide area computer networks, the ARPANET, and its successor, the worldwide Internet. Although we will give a brief history of the ARPANET later, it is useful to mention a few key aspects of it now. The ARPANET was a research network sponsored by the DoD (U.S. Department of Defense). It eventually connected hundreds of universities and government installations, using leased telephone lines. When satellite and radio networks were added later, the existing protocols had trouble interworking with them, so a new reference architecture was needed. Thus, from nearly the beginning, the ability to connect multiple networks in a seamless way was one of the major design goals. This architecture later became known as the **TCP/IP Reference Model**, after its two primary protocols. It was first described by Cerf and Kahn (1974), and later refined and defined as a standard in the Internet community (Braden, 1989). The design philosophy behind the model is discussed by Clark (1988).

Given the DoD's worry that some of its precious hosts, routers, and internetwork gateways might get blown to pieces at a moment's notice by an attack from the Soviet Union, another major goal was that the network be able to survive loss of subnet hardware, without existing conversations being broken off. In other

words, the DoD wanted connections to remain intact as long as the source and destination machines were functioning, even if some of the machines or transmission lines in between were suddenly put out of operation. Furthermore, since applications with divergent requirements were envisioned, ranging from transferring files to real-time speech transmission, a flexible architecture was needed.

The Link Layer

All these requirements led to the choice of a packet-switching network based on a connectionless layer that runs across different networks. The lowest layer in the model, the **link layer** describes what links such as serial lines and classic Ethernet must do to meet the needs of this connectionless internet layer. It is not really a layer at all, in the normal sense of the term, but rather an interface between hosts and transmission links. Early material on the TCP/IP model has little to say about it.

The Internet Layer

The **internet layer** is the linchpin that holds the whole architecture together. It is shown in Fig. 1-21 as corresponding roughly to the OSI network layer. Its job is to permit hosts to inject packets into any network and have them travel independently to the destination (potentially on a different network). They may even arrive in a completely different order than they were sent, in which case it is the job of higher layers to rearrange them, if in-order delivery is desired. Note that ''internet'' is used here in a generic sense, even though this layer is present in the Internet.

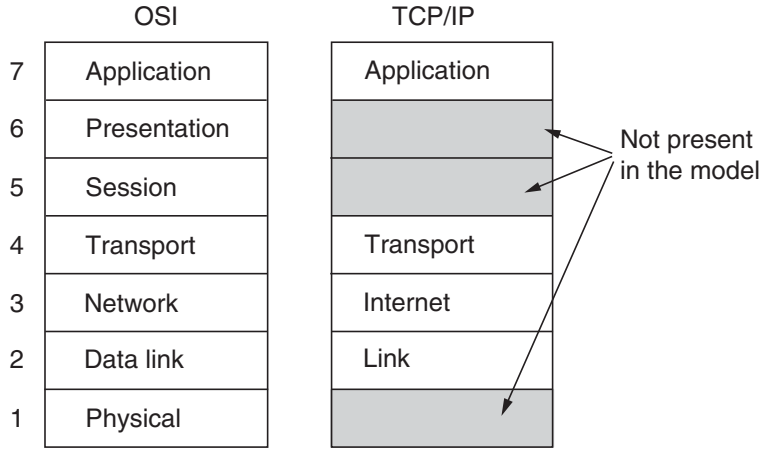


Figure 1-21. The TCP/IP reference model.

The analogy here is with the (snail) mail system. A person can drop a sequence of international letters into a mailbox in one country, and with a little luck,

most of them will be delivered to the correct address in the destination country. The letters will probably travel through one or more international mail gateways along the way, but this is transparent to the users. Furthermore, that each country (i.e., each network) has its own stamps, preferred envelope sizes, and delivery rules is hidden from the users.

The internet layer defines an official packet format and protocol called **IP** (**Internet Protocol**), plus a companion protocol called **ICMP** (**Internet Control Message Protocol**) that helps it function. The job of the internet layer is to deliver IP packets where they are supposed to go. Packet routing is clearly a major issue here, as is congestion (though IP has not proven effective at avoiding congestion).

The Transport Layer

The layer above the internet layer in the TCP/IP model is now usually called the **transport layer**. It is designed to allow peer entities on the source and destination hosts to carry on a conversation, just as in the OSI transport layer. Two end-to-end transport protocols have been defined here. The first one, **TCP** (**Transmission Control Protocol**), is a reliable connection-oriented protocol that allows a byte stream originating on one machine to be delivered without error on any other machine in the internet. It segments the incoming byte stream into discrete messages and passes each one on to the internet layer. At the destination, the receiving TCP process reassembles the received messages into the output stream. TCP also handles flow control to make sure a fast sender cannot swamp a slow receiver with more messages than it can handle.

The second protocol in this layer, **UDP** (**User Datagram Protocol**), is an unreliable, connectionless protocol for applications that do not want TCP's sequencing or flow control and wish to provide their own. It is also widely used for one-shot, client-server-type request-reply queries and applications in which prompt delivery is more important than accurate delivery, such as transmitting speech or video. The relation of IP, TCP, and UDP is shown in Fig. 1-22. Since the model was developed, IP has been implemented on many other networks.

The Application Layer

The TCP/IP model does not have session or presentation layers. No need for them was perceived. Instead, applications simply include any session and presentation functions that they require. Experience with the OSI model has proven this view correct: these layers are of little use to most applications.

On top of the transport layer is the **application layer**. It contains all the higher-level protocols. The early ones included virtual terminal (TELNET), file transfer (FTP), and electronic mail (SMTP). Many other protocols have been added to these over the years. Some important ones that we will study, shown in Fig. 1-22,

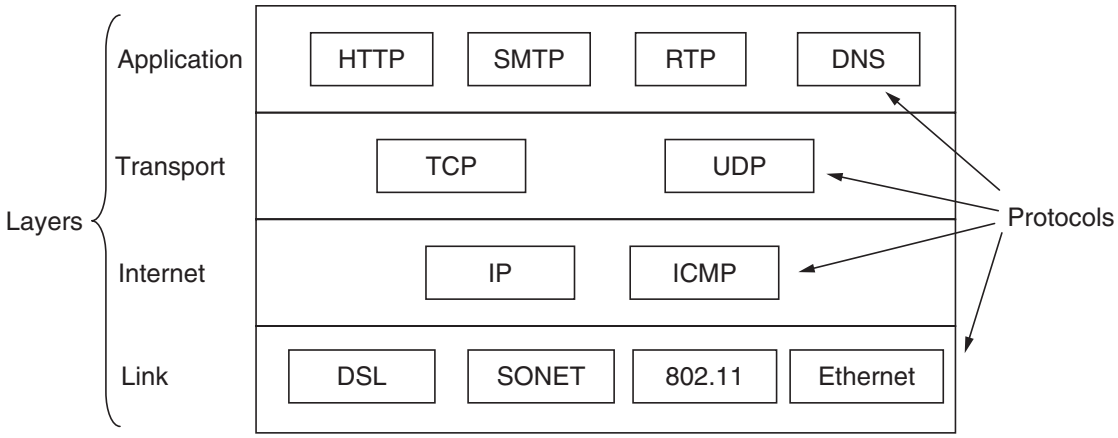


Figure 1-22. The TCP/IP model with some protocols we will study.

include the Domain Name System (DNS), for mapping host names onto their network addresses, HTTP, the protocol for fetching pages on the World Wide Web, and RTP, the protocol for delivering real-time media such as voice or movies.

1.4.3 The Model Used in This Book

As mentioned earlier, the strength of the OSI reference model is the *model* itself (minus the presentation and session layers), which has proven to be exceptionally useful for discussing computer networks. In contrast, the strength of the TCP/IP reference model is the *protocols*, which have been widely used for many years. Since computer scientists like to have their cake and eat it, too, we will use the hybrid model of Fig. 1-23 as the framework for this book.

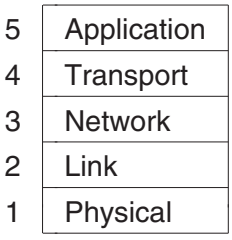


Figure 1-23. The reference model used in this book.

This model has five layers, running from the physical layer up through the link, network and transport layers to the application layer. The physical layer specifies how to transmit bits across different kinds of media as electrical (or other analog) signals. The link layer is concerned with how to send finite-length messages between directly connected computers with specified levels of reliability. Ethernet and 802.11 are examples of link layer protocols.

The network layer deals with how to combine multiple links into networks, and networks of networks, into internetworks so that we can send packets between distant computers. This includes the task of finding the path along which to send the packets. IP is the main example protocol we will study for this layer. The transport layer strengthens the delivery guarantees of the Network layer, usually with increased reliability, and provide delivery abstractions, such as a reliable byte stream, that match the needs of different applications. TCP is an important example of a transport layer protocol.

Finally, the application layer contains programs that make use of the network. Many, but not all, networked applications have user interfaces, such as a Web browser. Our concern, however, is with the portion of the program that uses the network. This is the HTTP protocol in the case of the Web browser. There are also important support programs in the application layer, such as the DNS, that are used by many applications.

Our chapter sequence is based on this model. In this way, we retain the value of the OSI model for understanding network architectures, but concentrate primarily on protocols that are important in practice, from TCP/IP and related protocols to newer ones such as 802.11, SONET, and Bluetooth.

1.4.4 A Comparison of the OSI and TCP/IP Reference Models

The OSI and TCP/IP reference models have much in common. Both are based on the concept of a stack of independent protocols. Also, the functionality of the layers is roughly similar. For example, in both models the layers up through and including the transport layer are there to provide an end-to-end, network-independent transport service to processes wishing to communicate. These layers form the transport provider. Again in both models, the layers above transport are application-oriented users of the transport service.

Despite these fundamental similarities, the two models also have many differences. In this section we will focus on the key differences between the two reference models. It is important to note that we are comparing the *reference models* here, not the corresponding *protocol stacks*. The protocols themselves will be discussed later. For an entire book comparing and contrasting TCP/IP and OSI, see Piscitello and Chapin (1993).

Three concepts are central to the OSI model:

1. Services.
2. Interfaces.
3. Protocols.

Probably the biggest contribution of the OSI model is that it makes the distinction between these three concepts explicit. Each layer performs some *services* for the

layer above it. The service definition tells what the layer does, not how entities above it access it or how the layer works. It defines the layer's semantics.

A layer's *interface* tells the processes above it how to access it. It specifies what the parameters are and what results to expect. It, too, says nothing about how the layer works inside.

Finally, the peer *protocols* used in a layer are the layer's own business. It can use any protocols it wants to, as long as it gets the job done (i.e., provides the offered services). It can also change them at will without affecting software in higher layers.

These ideas fit very nicely with modern ideas about object-oriented programming. An object, like a layer, has a set of methods (operations) that processes outside the object can invoke. The semantics of these methods define the set of services that the object offers. The methods' parameters and results form the object's interface. The code internal to the object is its protocol and is not visible or of any concern outside the object.

The TCP/IP model did not originally clearly distinguish between services, interfaces, and protocols, although people have tried to retrofit it after the fact to make it more OSI-like. For example, the only real services offered by the internet layer are SEND IP PACKET and RECEIVE IP PACKET. As a consequence, the protocols in the OSI model are better hidden than in the TCP/IP model and can be replaced relatively easily as the technology changes. Being able to make such changes transparently is one of the main purposes of having layered protocols in the first place.

The OSI reference model was devised *before* the corresponding protocols were invented. This ordering meant that the model was not biased toward one particular set of protocols, a fact that made it quite general. The downside of this ordering was that the designers did not have much experience with the subject and did not have a good idea of which functionality to put in which layer.

For example, the data link layer originally dealt only with point-to-point networks. When broadcast networks came around, a new sublayer had to be hacked into the model. Furthermore, when people started to build real networks using the OSI model and existing protocols, it was discovered that these networks did not match the required service specifications (wonder of wonders), so convergence sublayers had to be grafted onto the model to provide a place for papering over the differences. Finally, the committee originally expected that each country would have one network, run by the government and using the OSI protocols, so no thought was given to internetworking. To make a long story short, things did not turn out that way.

With TCP/IP the reverse was true: the protocols came first, and the model was really just a description of the existing protocols. There was no problem with the protocols fitting the model. They fit perfectly. The only trouble was that the *model* did not fit any other protocol stacks. Consequently, it was not especially useful for describing other, non-TCP/IP networks.

Turning from philosophical matters to more specific ones, an obvious difference between the two models is the number of layers: the OSI model has seven layers and the TCP/IP model has four. Both have (inter)network, transport, and application layers, but the other layers are different.

Another difference is in the area of connectionless versus connection-oriented communication. The OSI model supports both connectionless and connection-oriented communication in the network layer, but only connection-oriented communication in the transport layer, where it counts (because the transport service is visible to the users). The TCP/IP model supports only one mode in the network layer (connectionless) but both in the transport layer, giving the users a choice. This choice is especially important for simple request-response protocols.

1.4.5 A Critique of the OSI Model and Protocols

Neither the OSI model and its protocols nor the TCP/IP model and its protocols are perfect. Quite a bit of criticism can be, and has been, directed at both of them. In this section and the next one, we will look at some of these criticisms. We will begin with OSI and examine TCP/IP afterward.

At the time the second edition of this book was published (1989), it appeared to many experts in the field that the OSI model and its protocols were going to take over the world and push everything else out of their way. This did not happen. Why? A look back at some of the reasons may be useful. They can be summarized as:

1. Bad timing.
2. Bad technology.
3. Bad implementations.
4. Bad politics.

Bad Timing

First let us look at reason one: bad timing. The time at which a standard is established is absolutely critical to its success. David Clark of M.I.T. has a theory of standards that he calls the *apocalypse of the two elephants*, which is illustrated in Fig. 1-24.

This figure shows the amount of activity surrounding a new subject. When the subject is first discovered, there is a burst of research activity in the form of discussions, papers, and meetings. After a while this activity subsides, corporations discover the subject, and the billion-dollar wave of investment hits.

It is essential that the standards be written in the trough in between the two ''elephants.'' If they are written too early (before the research results are well

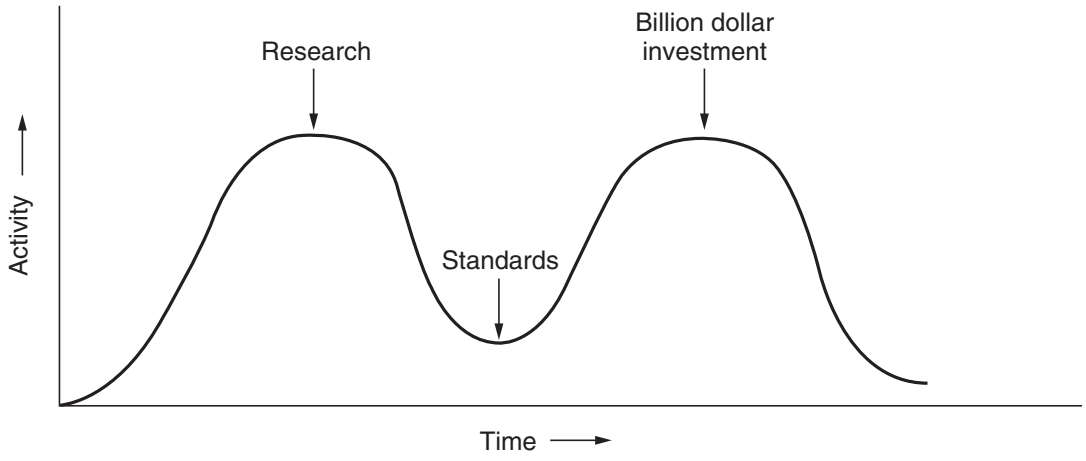


Figure 1-24. The apocalypse of the two elephants.

established), the subject may still be poorly understood; the result is a bad standard. If they are written too late, so many companies may have already made major investments in different ways of doing things that the standards are effectively ignored. If the interval between the two elephants is very short (because everyone is in a hurry to get started), the people developing the standards may get crushed.

It now appears that the standard OSI protocols got crushed. The competing TCP/IP protocols were already in widespread use by research universities by the time the OSI protocols appeared. While the billion-dollar wave of investment had not yet hit, the academic market was large enough that many vendors had begun cautiously offering TCP/IP products. When OSI came around, they did not want to support a second protocol stack until they were forced to, so there were no initial offerings. With every company waiting for every other company to go first, no company went first and OSI never happened.

Bad Technology

The second reason that OSI never caught on is that both the model and the protocols are flawed. The choice of seven layers was more political than technical, and two of the layers (session and presentation) are nearly empty, whereas two other ones (data link and network) are overfull.

The OSI model, along with its associated service definitions and protocols, is extraordinarily complex. When piled up, the printed standards occupy a significant fraction of a meter of paper. They are also difficult to implement and inefficient in operation. In this context, a riddle posed by Paul Mockapetris and cited by Rose (1993) comes to mind:

Q: What do you get when you cross a mobster with an international standard?
A: Someone who makes you an offer you can't understand.

In addition to being incomprehensible, another problem with OSI is that some functions, such as addressing, flow control, and error control, reappear again and again in each layer. Saltzer et al. (1984), for example, have pointed out that to be effective, error control must be done in the highest layer, so that repeating it over and over in each of the lower layers is often unnecessary and inefficient.

Bad Implementations

Given the enormous complexity of the model and the protocols, it will come as no surprise that the initial implementations were huge, unwieldy, and slow. Everyone who tried them got burned. It did not take long for people to associate "OSI" with "poor quality." Although the products improved in the course of time, the image stuck.

In contrast, one of the first implementations of TCP/IP was part of Berkeley UNIX and was quite good (not to mention, free). People began using it quickly, which led to a large user community, which led to improvements, which led to an even larger community. Here the spiral was upward instead of downward.

Bad Politics

On account of the initial implementation, many people, especially in academia, thought of TCP/IP as part of UNIX, and UNIX in the 1980s in academia was not unlike parenthood (then incorrectly called motherhood) and apple pie.

OSI, on the other hand, was widely thought to be the creature of the European telecommunication ministries, the European Community, and later the U.S. Government. This belief was only partly true, but the very idea of a bunch of government bureaucrats trying to shove a technically inferior standard down the throats of the poor researchers and programmers down in the trenches actually developing computer networks did not aid OSI's cause. Some people viewed this development in the same light as IBM announcing in the 1960s that PL/I was the language of the future, or the DoD correcting this later by announcing that it was actually Ada.

1.4.6 A Critique of the TCP/IP Reference Model

The TCP/IP model and protocols have their problems too. First, the model does not clearly distinguish the concepts of services, interfaces, and protocols. Good software engineering practice requires differentiating between the specification and the implementation, something that OSI does very carefully, but TCP/IP does not. Consequently, the TCP/IP model is not much of a guide for designing new networks using new technologies.

Second, the TCP/IP model is not at all general and is poorly suited to describing any protocol stack other than TCP/IP. Trying to use the TCP/IP model to describe Bluetooth, for example, is completely impossible.

Third, the link layer is not really a layer at all in the normal sense of the term as used in the context of layered protocols. It is an interface (between the network and data link layers). The distinction between an interface and a layer is crucial, and one should not be sloppy about it.

Fourth, the TCP/IP model does not distinguish between the physical and data link layers. These are completely different. The physical layer has to do with the transmission characteristics of copper wire, fiber optics, and wireless communication. The data link layer's job is to delimit the start and end of frames and get them from one side to the other with the desired degree of reliability. A proper model should include both as separate layers. The TCP/IP model does not do this.

Finally, although the IP and TCP protocols were carefully thought out and well implemented, many of the other protocols were ad hoc, generally produced by a couple of graduate students hacking away until they got tired. The protocol implementations were then distributed free, which resulted in their becoming widely used, deeply entrenched, and thus hard to replace. Some of them are a bit of an embarrassment now. The virtual terminal protocol, TELNET, for example, was designed for a ten-character-per-second mechanical Teletype terminal. It knows nothing of graphical user interfaces and mice. Nevertheless, it is still in use some 30 years later.

1.5 EXAMPLE NETWORKS

The subject of computer networking covers many different kinds of networks, large and small, well known and less well known. They have different goals, scales, and technologies. In the following sections, we will look at some examples, to get an idea of the variety one finds in the area of computer networking.

We will start with the Internet, probably the best known network, and look at its history, evolution, and technology. Then we will consider the mobile phone network. Technically, it is quite different from the Internet, contrasting nicely with it. Next we will introduce IEEE 802.11, the dominant standard for wireless LANs. Finally, we will look at RFID and sensor networks, technologies that extend the reach of the network to include the physical world and everyday objects.

1.5.1 The Internet

The Internet is not really a network at all, but a vast collection of different networks that use certain common protocols and provide certain common services. It is an unusual system in that it was not planned by anyone and is not controlled by anyone. To better understand it, let us start from the beginning and see how it has developed and why. For a wonderful history of the Internet, John Naughton's (2000) book is highly recommended. It is one of those rare books that is not only fun to read, but also has 20 pages of *ibid.*'s and *op. cit.*'s for the serious historian. Some of the material in this section is based on this book.

Of course, countless technical books have been written about the Internet and its protocols as well. For more information, see, for example, Maufer (1999).

The ARPANET

The story begins in the late 1950s. At the height of the Cold War, the U.S. DoD wanted a command-and-control network that could survive a nuclear war. At that time, all military communications used the public telephone network, which was considered vulnerable. The reason for this belief can be gleaned from Fig. 1-25(a). Here the black dots represent telephone switching offices, each of which was connected to thousands of telephones. These switching offices were, in turn, connected to higher-level switching offices (toll offices), to form a national hierarchy with only a small amount of redundancy. The vulnerability of the system was that the destruction of a few key toll offices could fragment it into many isolated islands.

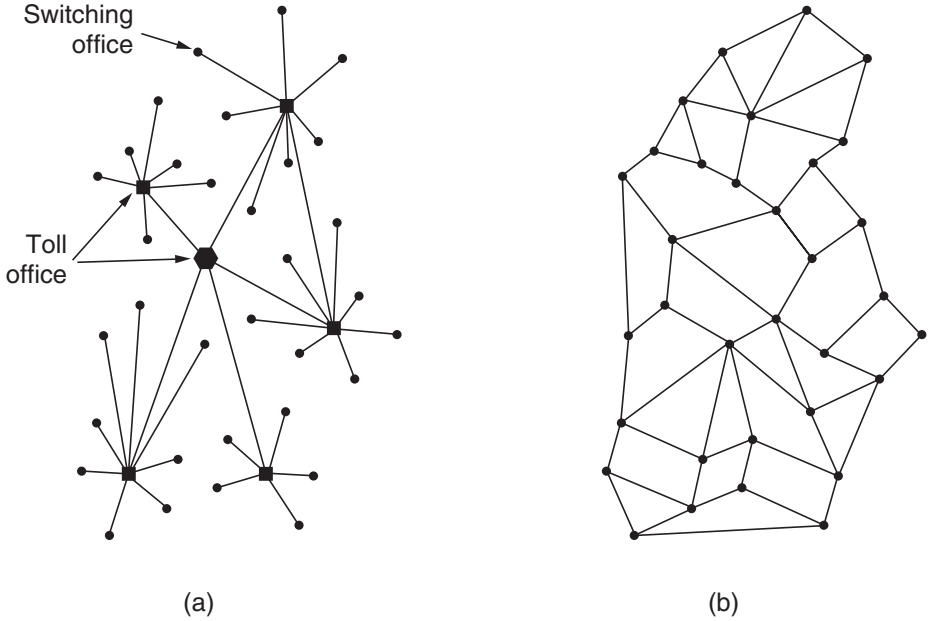


Figure 1-25. (a) Structure of the telephone system. (b) Baran's proposed distributed switching system.

Around 1960, the DoD awarded a contract to the RAND Corporation to find a solution. One of its employees, Paul Baran, came up with the highly distributed and fault-tolerant design of Fig. 1-25(b). Since the paths between any two switching offices were now much longer than analog signals could travel without distortion, Baran proposed using digital packet-switching technology. Baran wrote several reports for the DoD describing his ideas in detail (Baran, 1964). Officials at the Pentagon liked the concept and asked AT&T, then the U.S.' national telephone monopoly, to build a prototype. AT&T dismissed Baran's ideas out of hand. The biggest and richest corporation in the world was not about to allow

some young whippersnapper tell it how to build a telephone system. They said Baran's network could not be built and the idea was killed.

Several years went by and still the DoD did not have a better command-and-control system. To understand what happened next, we have to go back all the way to October 1957, when the Soviet Union beat the U.S. into space with the launch of the first artificial satellite, Sputnik. When President Eisenhower tried to find out who was asleep at the switch, he was appalled to find the Army, Navy, and Air Force squabbling over the Pentagon's research budget. His immediate response was to create a single defense research organization, **ARPA**, the **Advanced Research Projects Agency**. ARPA had no scientists or laboratories; in fact, it had nothing more than an office and a small (by Pentagon standards) budget. It did its work by issuing grants and contracts to universities and companies whose ideas looked promising to it.

For the first few years, ARPA tried to figure out what its mission should be. In 1967, the attention of Larry Roberts, a program manager at ARPA who was trying to figure out how to provide remote access to computers, turned to networking. He contacted various experts to decide what to do. One of them, Wesley Clark, suggested building a packet-switched subnet, connecting each host to its own router.

After some initial skepticism, Roberts bought the idea and presented a somewhat vague paper about it at the ACM SIGOPS Symposium on Operating System Principles held in Gatlinburg, Tennessee in late 1967 (Roberts, 1967). Much to Roberts' surprise, another paper at the conference described a similar system that had not only been designed but actually fully implemented under the direction of Donald Davies at the National Physical Laboratory in England. The NPL system was not a national system (it just connected several computers on the NPL campus), but it demonstrated that packet switching could be made to work. Furthermore, it cited Baran's now discarded earlier work. Roberts came away from Gatlinburg determined to build what later became known as the **ARPANET**.

The subnet would consist of minicomputers called **IMPs** (**Interface Message Processors**) connected by 56-kbps transmission lines. For high reliability, each IMP would be connected to at least two other IMPs. The subnet was to be a datagram subnet, so if some lines and IMPs were destroyed, messages could be automatically rerouted along alternative paths.

Each node of the network was to consist of an IMP and a host, in the same room, connected by a short wire. A host could send messages of up to 8063 bits to its IMP, which would then break these up into packets of at most 1008 bits and forward them independently toward the destination. Each packet was received in its entirety before being forwarded, so the subnet was the first electronic store-and-forward packet-switching network.

ARPA then put out a tender for building the subnet. Twelve companies bid for it. After evaluating all the proposals, ARPA selected BBN, a consulting firm based in Cambridge, Massachusetts, and in December 1968 awarded it a contract

to build the subnet and write the subnet software. BBN chose to use specially modified Honeywell DDP-316 minicomputers with 12K 16-bit words of core memory as the IMPs. The IMPs did not have disks, since moving parts were considered unreliable. The IMPs were interconnected by 56-kbps lines leased from telephone companies. Although 56 kbps is now the choice of teenagers who cannot afford DSL or cable, it was then the best money could buy.

The software was split into two parts: subnet and host. The subnet software consisted of the IMP end of the host-IMP connection, the IMP-IMP protocol, and a source IMP to destination IMP protocol designed to improve reliability. The original ARPANET design is shown in Fig. 1-26.

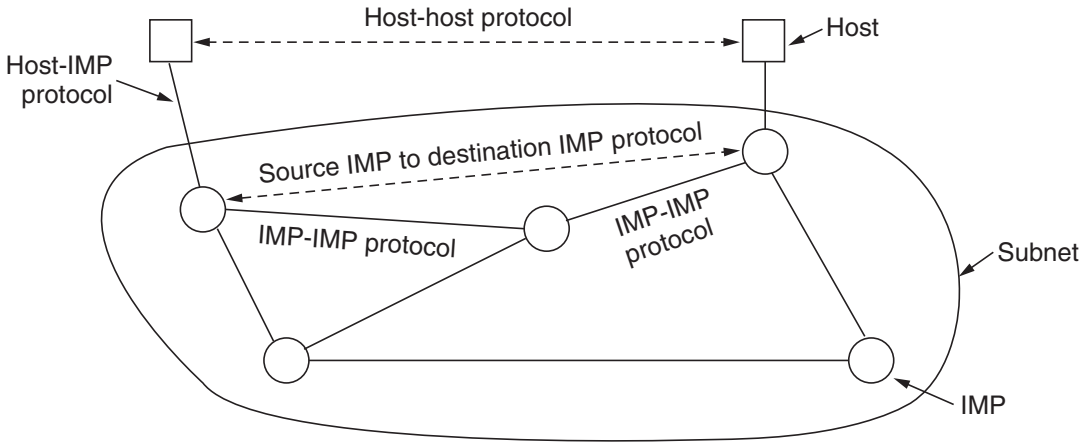


Figure 1-26. The original ARPANET design.

Outside the subnet, software was also needed, namely, the host end of the host-IMP connection, the host-host protocol, and the application software. It soon became clear that BBN was of the opinion that when it had accepted a message on a host-IMP wire and placed it on the host-IMP wire at the destination, its job was done.

Roberts had a problem, though: the hosts needed software too. To deal with it, he convened a meeting of network researchers, mostly graduate students, at Snowbird, Utah, in the summer of 1969. The graduate students expected some network expert to explain the grand design of the network and its software to them and then assign each of them the job of writing part of it. They were astounded when there was no network expert and no grand design. They had to figure out what to do on their own.

Nevertheless, somehow an experimental network went online in December 1969 with four nodes: at UCLA, UCSB, SRI, and the University of Utah. These four were chosen because all had a large number of ARPA contracts, and all had different and completely incompatible host computers (just to make it more fun). The first host-to-host message had been sent two months earlier from the UCLA

node by a team led by Len Kleinrock (a pioneer of the theory of packet switching) to the SRI node. The network grew quickly as more IMPs were delivered and installed; it soon spanned the United States. Figure 1-27 shows how rapidly the ARPANET grew in the first 3 years.

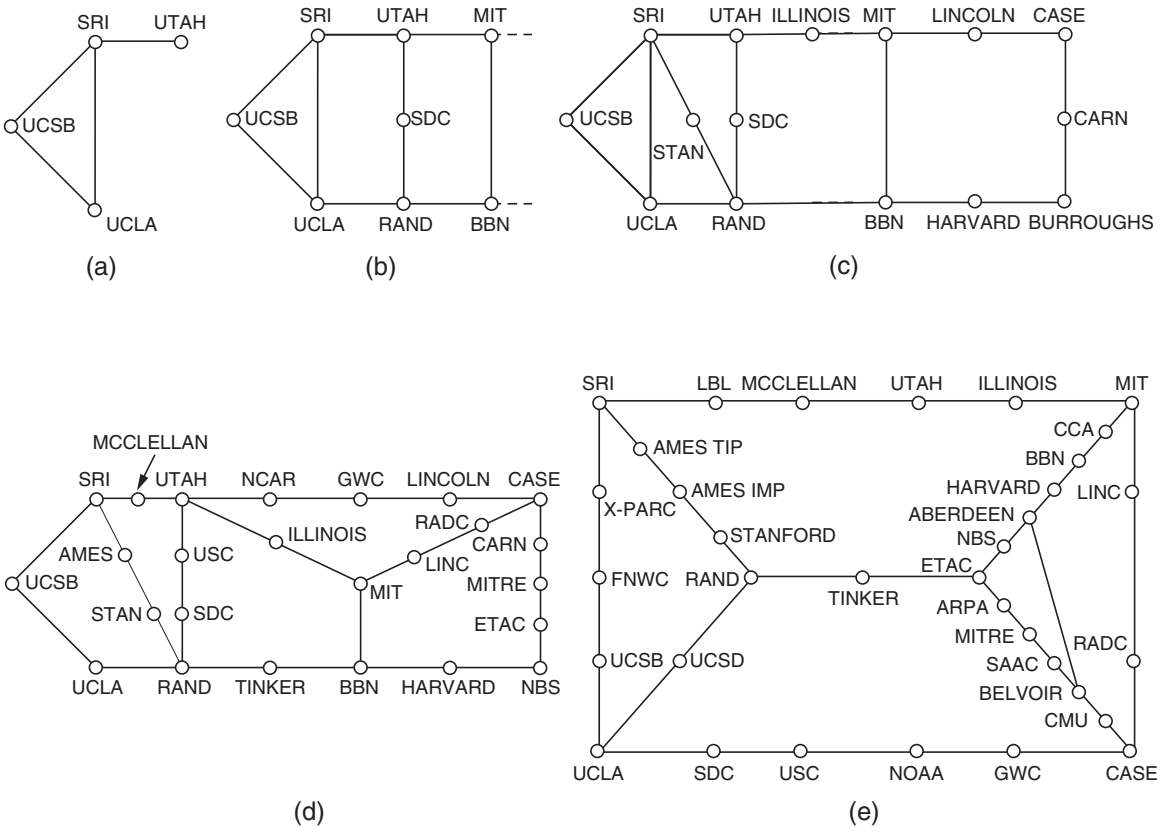


Figure 1-27. Growth of the ARPANET. (a) December 1969. (b) July 1970. (c) March 1971. (d) April 1972. (e) September 1972.

In addition to helping the fledgling ARPANET grow, ARPA also funded research on the use of satellite networks and mobile packet radio networks. In one now famous demonstration, a truck driving around in California used the packet radio network to send messages to SRI, which were then forwarded over the ARPANET to the East Coast, where they were shipped to University College in London over the satellite network. This allowed a researcher in the truck to use a computer in London while driving around in California.

This experiment also demonstrated that the existing ARPANET protocols were not suitable for running over different networks. This observation led to more research on protocols, culminating with the invention of the TCP/IP model and protocols (Cerf and Kahn, 1974). TCP/IP was specifically designed to handle communication over internetworks, something becoming increasingly important as more and more networks were hooked up to the ARPANET.

To encourage adoption of these new protocols, ARPA awarded several contracts to implement TCP/IP on different computer platforms, including IBM, DEC, and HP systems, as well as for Berkeley UNIX. Researchers at the University of California at Berkeley rewrote TCP/IP with a new programming interface called **sockets** for the upcoming 4.2BSD release of Berkeley UNIX. They also wrote many application, utility, and management programs to show how convenient it was to use the network with sockets.

The timing was perfect. Many universities had just acquired a second or third VAX computer and a LAN to connect them, but they had no networking software. When 4.2BSD came along, with TCP/IP, sockets, and many network utilities, the complete package was adopted immediately. Furthermore, with TCP/IP, it was easy for the LANs to connect to the ARPANET, and many did.

During the 1980s, additional networks, especially LANs, were connected to the ARPANET. As the scale increased, finding hosts became increasingly expensive, so **DNS** (**Domain Name System**) was created to organize machines into domains and map host names onto IP addresses. Since then, DNS has become a generalized, distributed database system for storing a variety of information related to naming. We will study it in detail in Chap. 7.

NSFNET

By the late 1970s, NSF (the U.S. National Science Foundation) saw the enormous impact the ARPANET was having on university research, allowing scientists across the country to share data and collaborate on research projects. However, to get on the ARPANET a university had to have a research contract with the DoD. Many did not have a contract. NSF's initial response was to fund the Computer Science Network (**CSNET**) in 1981. It connected computer science departments and industrial research labs to the ARPANET via dial-up and leased lines. In the late 1980s, the NSF went further and decided to design a successor to the ARPANET that would be open to all university research groups.

To have something concrete to start with, NSF decided to build a backbone network to connect its six supercomputer centers, in San Diego, Boulder, Champaign, Pittsburgh, Ithaca, and Princeton. Each supercomputer was given a little brother, consisting of an LSI-11 microcomputer called a **fuzzball**. The fuzzballs were connected with 56-kbps leased lines and formed the subnet, the same hardware technology the ARPANET used. The software technology was different however: the fuzzballs spoke TCP/IP right from the start, making it the first TCP/IP WAN.

NSF also funded some (eventually about 20) regional networks that connected to the backbone to allow users at thousands of universities, research labs, libraries, and museums to access any of the supercomputers and to communicate with one another. The complete network, including backbone and the regional networks, was called **NSFNET**. It connected to the ARPANET through a link between an

IMP and a fuzzball in the Carnegie-Mellon machine room. The first NSFNET backbone is illustrated in Fig. 1-28 superimposed on a map of the U.S.

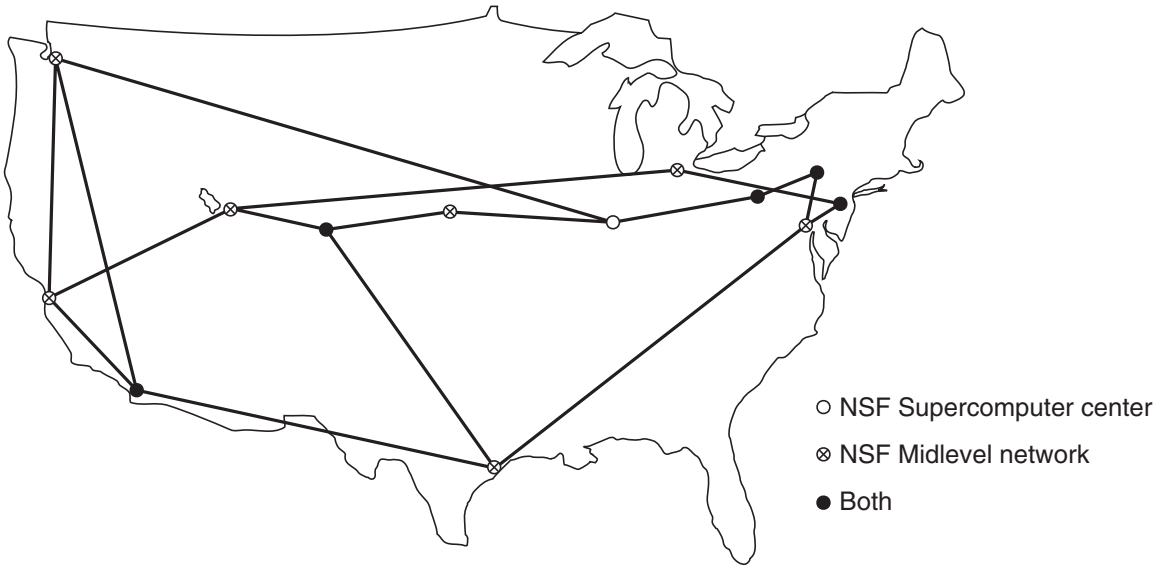


Figure 1-28. The NSFNET backbone in 1988.

NSFNET was an instantaneous success and was overloaded from the word go. NSF immediately began planning its successor and awarded a contract to the Michigan-based MERIT consortium to run it. Fiber optic channels at 448 kbps were leased from MCI (since merged with WorldCom) to provide the version 2 backbone. IBM PC-RTs were used as routers. This, too, was soon overwhelmed, and by 1990, the second backbone was upgraded to 1.5 Mbps.

As growth continued, NSF realized that the government could not continue financing networking forever. Furthermore, commercial organizations wanted to join but were forbidden by NSF's charter from using networks NSF paid for. Consequently, NSF encouraged MERIT, MCI, and IBM to form a nonprofit corporation, **ANS** (**Advanced Networks and Services**), as the first step along the road to commercialization. In 1990, ANS took over NSFNET and upgraded the 1.5-Mbps links to 45 Mbps to form **ANSNET**. This network operated for 5 years and was then sold to America Online. But by then, various companies were offering commercial IP service and it was clear the government should now get out of the networking business.

To ease the transition and make sure every regional network could communicate with every other regional network, NSF awarded contracts to four different network operators to establish a **NAP** (**Network Access Point**). These operators were PacBell (San Francisco), Ameritech (Chicago), MFS (Washington, D.C.), and Sprint (New York City, where for NAP purposes, Pennsauken, New Jersey counts as New York City). Every network operator that wanted to provide backbone service to the NSF regional networks had to connect to all the NAPs.

This arrangement meant that a packet originating on any regional network had a choice of backbone carriers to get from its NAP to the destination's NAP. Consequently, the backbone carriers were forced to compete for the regional networks' business on the basis of service and price, which was the idea, of course. As a result, the concept of a single default backbone was replaced by a commercially driven competitive infrastructure. Many people like to criticize the Federal Government for not being innovative, but in the area of networking, it was DoD and NSF that created the infrastructure that formed the basis for the Internet and then handed it over to industry to operate.

During the 1990s, many other countries and regions also built national research networks, often patterned on the ARPANET and NSFNET. These included EuropaNET and EBONE in Europe, which started out with 2-Mbps lines and then upgraded to 34-Mbps lines. Eventually, the network infrastructure in Europe was handed over to industry as well.

The Internet has changed a great deal since those early days. It exploded in size with the emergence of the World Wide Web (WWW) in the early 1990s. Recent data from the Internet Systems Consortium puts the number of visible Internet hosts at over 600 million. This guess is only a low-ball estimate, but it far exceeds the few million hosts that were around when the first conference on the WWW was held at CERN in 1994.

The way we use the Internet has also changed radically. Initially, applications such as email-for-academics, newsgroups, remote login, and file transfer dominated. Later it switched to email-for-everyman, then the Web and peer-to-peer content distribution, such as the now-shuttered Napster. Now real-time media distribution, social networks (e.g., Facebook), and microblogging (e.g., Twitter) are taking off. These switches brought richer kinds of media to the Internet and hence much more traffic. In fact, the dominant traffic on the Internet seems to change with some regularity as, for example, new and better ways to work with music or movies can become very popular very quickly.

Architecture of the Internet

The architecture of the Internet has also changed a great deal as it has grown explosively. In this section, we will attempt to give a brief overview of what it looks like today. The picture is complicated by continuous upheavals in the businesses of telephone companies (telcos), cable companies and ISPs that often make it hard to tell who is doing what. One driver of these upheavals is telecommunications convergence, in which one network is used for previously different uses. For example, in a ''triple play'' one company sells you telephony, TV, and Internet service over the same network connection on the assumption that this will save you money. Consequently, the description given here will be of necessity somewhat simpler than reality. And what is true today may not be true tomorrow.

The big picture is shown in Fig. 1-29. Let us examine this figure piece by piece, starting with a computer at home (at the edges of the figure). To join the Internet, the computer is connected to an **Internet Service Provider**, or simply **ISP**, from who the user purchases **Internet access** or **connectivity**. This lets the computer exchange packets with all of the other accessible hosts on the Internet. The user might send packets to surf the Web or for any of a thousand other uses, it does not matter. There are many kinds of Internet access, and they are usually distinguished by how much bandwidth they provide and how much they cost, but the most important attribute is connectivity.

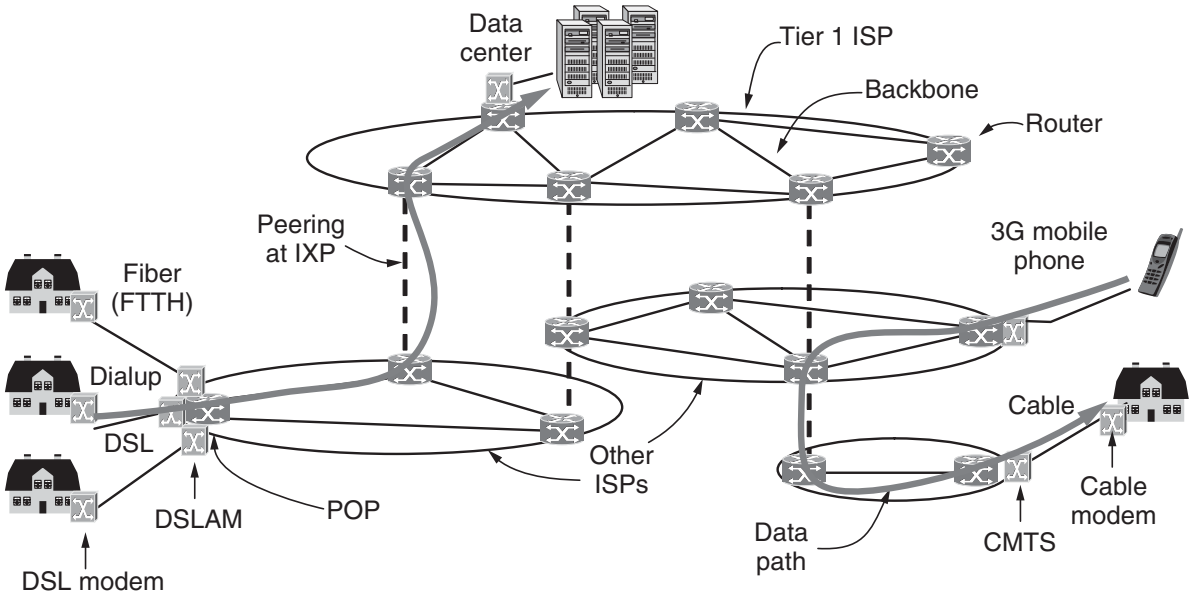


Figure 1-29. Overview of the Internet architecture.

A common way to connect to an ISP is to use the phone line to your house, in which case your phone company is your ISP. **DSL**, short for **Digital Subscriber Line**, reuses the telephone line that connects to your house for digital data transmission. The computer is connected to a device called a **DSL modem** that converts between digital packets and analog signals that can pass unhindered over the telephone line. At the other end, a device called a **DSLAM** (**Digital Subscriber Line Access Multiplexer**) converts between signals and packets.

Several other popular ways to connect to an ISP are shown in Fig. 1-29. DSL is a higher-bandwidth way to use the local telephone line than to send bits over a traditional telephone call instead of a voice conversation. That is called **dial-up** and done with a different kind of modem at both ends. The word **modem** is short for "*mo*dulator *dem*odulator" and refers to any device that converts between digital bits and analog signals.

Another method is to send signals over the cable TV system. Like DSL, this is a way to reuse existing infrastructure, in this case otherwise unused cable TV

channels. The device at the home end is called a **cable modem** and the device at the **cable headend** is called the **CMTS** (**Cable Modem Termination System**).

DSL and cable provide Internet access at rates from a small fraction of a megabit/sec to multiple megabit/sec, depending on the system. These rates are much greater than dial-up rates, which are limited to 56 kbps because of the narrow bandwidth used for voice calls. Internet access at much greater than dial-up speeds is called **broadband**. The name refers to the broader bandwidth that is used for faster networks, rather than any particular speed.

The access methods mentioned so far are limited by the bandwidth of the ''last mile'' or last leg of transmission. By running optical fiber to residences, faster Internet access can be provided at rates on the order of 10 to 100 Mbps. This design is called **FTTH** (**Fiber to the Home**). For businesses in commercial areas, it may make sense to lease a high-speed transmission line from the offices to the nearest ISP. For example, in North America, a T3 line runs at roughly 45 Mbps.

Wireless is used for Internet access too. An example we will explore shortly is that of 3G mobile phone networks. They can provide data delivery at rates of 1 Mbps or higher to mobile phones and fixed subscribers in the coverage area.

We can now move packets between the home and the ISP. We call the location at which customer packets enter the ISP network for service the ISP's **POP** (**Point of Presence**). We will next explain how packets are moved between the POPs of different ISPs. From this point on, the system is fully digital and packet switched.

ISP networks may be regional, national, or international in scope. We have already seen that their architecture is made up of long-distance transmission lines that interconnect routers at POPs in the different cities that the ISPs serve. This equipment is called the **backbone** of the ISP. If a packet is destined for a host served directly by the ISP, that packet is routed over the backbone and delivered to the host. Otherwise, it must be handed over to another ISP.

ISPs connect their networks to exchange traffic at **IXPs** (**Internet eXchange Points**). The connected ISPs are said to **peer** with each other. There are many IXPs in cities around the world. They are drawn vertically in Fig. 1-29 because ISP networks overlap geographically. Basically, an IXP is a room full of routers, at least one per ISP. A LAN in the room connects all the routers, so packets can be forwarded from any ISP backbone to any other ISP backbone. IXPs can be large and independently owned facilities. One of the largest is the Amsterdam Internet Exchange, to which hundreds of ISPs connect and through which they exchange hundreds of gigabits/sec of traffic.

The peering that happens at IXPs depends on the business relationships between ISPs. There are many possible relationships. For example, a small ISP might pay a larger ISP for Internet connectivity to reach distant hosts, much as a customer purchases service from an Internet provider. In this case, the small ISP is said to pay for **transit**. Alternatively, two large ISPs might decide to exchange

traffic so that each ISP can deliver some traffic to the other ISP without having to pay for transit. One of the many paradoxes of the Internet is that ISPs who publicly compete with one another for customers often privately cooperate to do peering (Metz, 2001).

The path a packet takes through the Internet depends on the peering choices of the ISPs. If the ISP delivering a packet peers with the destination ISP, it might deliver the packet directly to its peer. Otherwise, it might route the packet to the nearest place at which it connects to a paid transit provider so that provider can deliver the packet. Two example paths across ISPs are drawn in Fig. 1-29. Often, the path a packet takes will not be the shortest path through the Internet.

At the top of the food chain are a small handful of companies, like AT&T and Sprint, that operate large international backbone networks with thousands of routers connected by high-bandwidth fiber optic links. These ISPs do not pay for transit. They are usually called **tier 1** ISPs and are said to form the backbone of the Internet, since everyone else must connect to them to be able to reach the entire Internet.

Companies that provide lots of content, such as Google and Yahoo!, locate their computers in **data centers** that are well connected to the rest of the Internet. These data centers are designed for computers, not humans, and may be filled with rack upon rack of machines called a **server farm**. **Colocation** or **hosting** data centers let customers put equipment such as servers at ISP POPs so that short, fast connections can be made between the servers and the ISP backbones. The Internet hosting industry has become increasingly virtualized so that it is now common to rent a virtual machine that is run on a server farm instead of installing a physical computer. These data centers are so large (tens or hundreds of thousands of machines) that electricity is a major cost, so data centers are sometimes built in areas where electricity is cheap.

This ends our quick tour of the Internet. We will have a great deal to say about the individual components and their design, algorithms, and protocols in subsequent chapters. One further point worth mentioning here is that what it means to be on the Internet is changing. It used to be that a machine was on the Internet if it: (1) ran the TCP/IP protocol stack; (2) had an IP address; and (3) could send IP packets to all the other machines on the Internet. However, ISPs often reuse IP addresses depending on which computers are in use at the moment, and home networks often share one IP address between multiple computers. This practice undermines the second condition. Security measures such as firewalls can also partly block computers from receiving packets, undermining the third condition. Despite these difficulties, it makes sense to regard such machines as being on the Internet while they are connected to their ISPs.

Also worth mentioning in passing is that some companies have interconnected all their existing internal networks, often using the same technology as the Internet. These **intranets** are typically accessible only on company premises or from company notebooks but otherwise work the same way as the Internet.

1.5.2 Third-Generation Mobile Phone Networks

People love to talk on the phone even more than they like to surf the Internet, and this has made the mobile phone network the most successful network in the world. It has more than four billion subscribers worldwide. To put this number in perspective, it is roughly 60% of the world's population and more than the number of Internet hosts and fixed telephone lines combined (ITU, 2009).

The architecture of the mobile phone network has changed greatly over the past 40 years along with its tremendous growth. First-generation mobile phone systems transmitted voice calls as continuously varying (analog) signals rather than sequences of (digital) bits. **AMPS** (**Advanced Mobile Phone System**), which was deployed in the United States in 1982, was a widely used first-generation system. Second-generation mobile phone systems switched to transmitting voice calls in digital form to increase capacity, improve security, and offer text messaging. **GSM** (**Global System for Mobile communications**), which was deployed starting in 1991 and has become the most widely used mobile phone system in the world, is a 2G system.

The third generation, or 3G, systems were initially deployed in 2001 and offer both digital voice and broadband digital data services. They also come with a lot of jargon and many different standards to choose from. 3G is loosely defined by the ITU (an international standards body we will discuss in the next section) as providing rates of at least 2 Mbps for stationary or walking users and 384 kbps in a moving vehicle. **UMTS** (**Universal Mobile Telecommunications System**), also called **WCDMA** (**Wideband Code Division Multiple Access**), is the main 3G system that is being rapidly deployed worldwide. It can provide up to 14 Mbps on the downlink and almost 6 Mbps on the uplink. Future releases will use multiple antennas and radios to provide even greater speeds for users.

The scarce resource in 3G systems, as in 2G and 1G systems before them, is radio spectrum. Governments license the right to use parts of the spectrum to the mobile phone network operators, often using a spectrum auction in which network operators submit bids. Having a piece of licensed spectrum makes it easier to design and operate systems, since no one else is allowed transmit on that spectrum, but it often costs a serious amount of money. In the UK in 2000, for example, five 3G licenses were auctioned for a total of about $40 billion.

It is the scarcity of spectrum that led to the **cellular network** design shown in Fig. 1-30 that is now used for mobile phone networks. To manage the radio interference between users, the coverage area is divided into cells. Within a cell, users are assigned channels that do not interfere with each other and do not cause too much interference for adjacent cells. This allows for good reuse of the spectrum, or **frequency reuse**, in the neighboring cells, which increases the capacity of the network. In 1G systems, which carried each voice call on a specific frequency band, the frequencies were carefully chosen so that they did not conflict with neighboring cells. In this way, a given frequency might only be reused once

in several cells. Modern 3G systems allow each cell to use all frequencies, but in a way that results in a tolerable level of interference to the neighboring cells. There are variations on the cellular design, including the use of directional or sectored antennas on cell towers to further reduce interference, but the basic idea is the same.

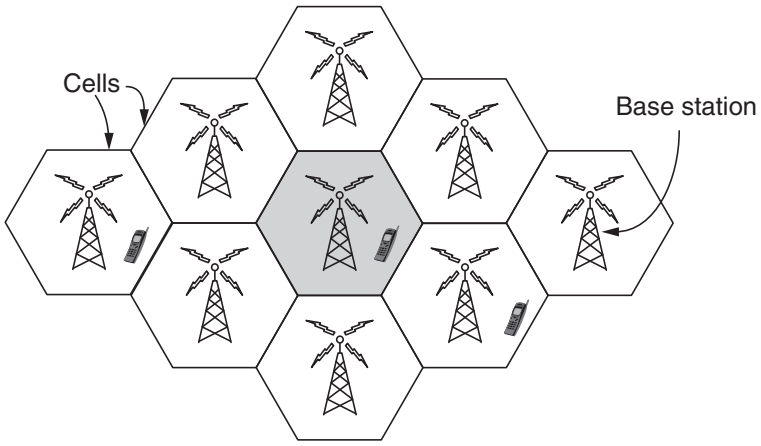


Figure 1-30. Cellular design of mobile phone networks.

The architecture of the mobile phone network is very different than that of the Internet. It has several parts, as shown in the simplified version of the UMTS architecture in Fig. 1-31. First, there is the **air interface**. This term is a fancy name for the radio communication protocol that is used over the air between the mobile device (e.g., the cell phone) and the **cellular base station**. Advances in the air interface over the past decades have greatly increased wireless data rates. The UMTS air interface is based on **Code Division Multiple Access** (**CDMA**), a technique that we will study in Chap. 2.

The cellular base station together with its controller forms the **radio access network**. This part is the wireless side of the mobile phone network. The controller node or **RNC** (**Radio Network Controller**) controls how the spectrum is used. The base station implements the air interface. It is called **Node B**, a temporary label that stuck.

The rest of the mobile phone network carries the traffic for the radio access network. It is called the **core network**. The UMTS core network evolved from the core network used for the 2G GSM system that came before it. However, something surprising is happening in the UMTS core network.

Since the beginning of networking, a war has been going on between the people who support packet networks (i.e., connectionless subnets) and the people who support circuit networks (i.e., connection-oriented subnets). The main proponents of packets come from the Internet community. In a connectionless design, every packet is routed independently of every other packet. As a consequence, if some routers go down during a session, no harm will be done as long as the system can

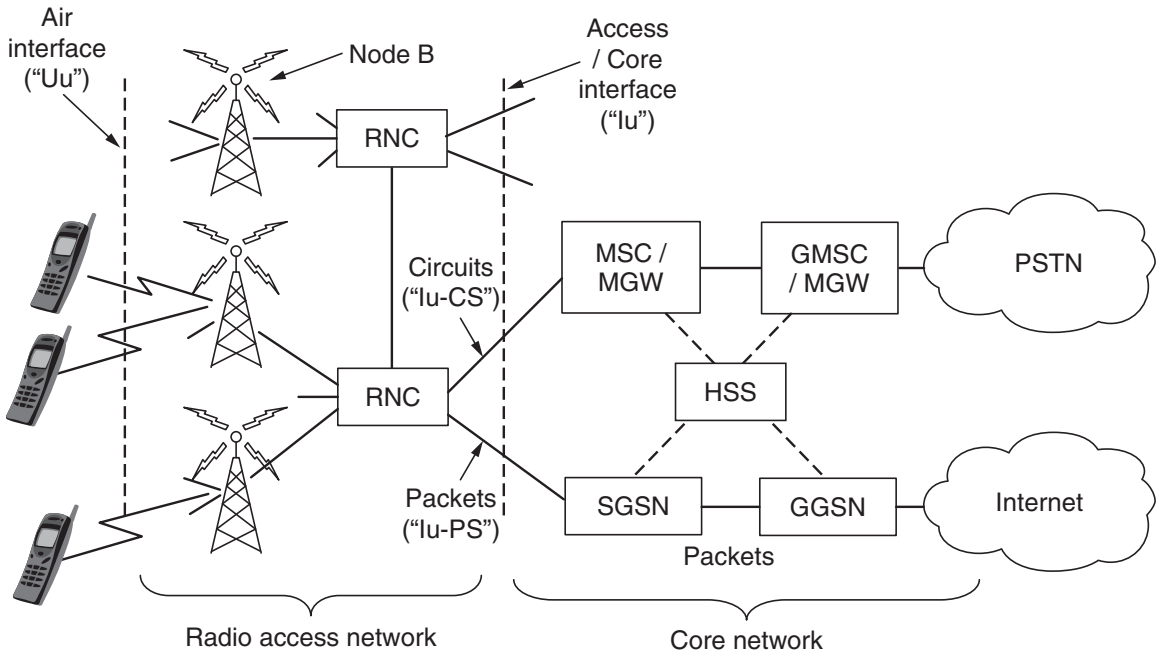


Figure 1-31. Architecture of the UMTS 3G mobile phone network.

dynamically reconfigure itself so that subsequent packets can find some route to the destination, even if it is different from that which previous packets used.

The circuit camp comes from the world of telephone companies. In the telephone system, a caller must dial the called party's number and wait for a connection before talking or sending data. This connection setup establishes a route through the telephone system that is maintained until the call is terminated. All words or packets follow the same route. If a line or switch on the path goes down, the call is aborted, making it less fault tolerant than a connectionless design.

The advantage of circuits is that they can support quality of service more easily. By setting up a connection in advance, the subnet can reserve resources such as link bandwidth, switch buffer space, and CPU. If an attempt is made to set up a call and insufficient resources are available, the call is rejected and the caller gets a kind of busy signal. In this way, once a connection has been set up, the connection will get good service.

With a connectionless network, if too many packets arrive at the same router at the same moment, the router will choke and probably lose packets. The sender will eventually notice this and resend them, but the quality of service will be jerky and unsuitable for audio or video unless the network is lightly loaded. Needless to say, providing adequate audio quality is something telephone companies care about very much, hence their preference for connections.

The surprise in Fig. 1-31 is that there is both packet and circuit switched equipment in the core network. This shows the mobile phone network in transition, with mobile phone companies able to implement one or sometimes both of

the alternatives. Older mobile phone networks used a circuit-switched core in the style of the traditional phone network to carry voice calls. This legacy is seen in the UMTS network with the **MSC** (**Mobile Switching Center**), **GMSC** (**Gateway Mobile Switching Center**), and **MGW** (**Media Gateway**) elements that set up connections over a circuit-switched core network such as the **PSTN** (**Public Switched Telephone Network**).

Data services have become a much more important part of the mobile phone network than they used to be, starting with text messaging and early packet data services such as **GPRS** (**General Packet Radio Service**) in the GSM system. These older data services ran at tens of kbps, but users wanted more. Newer mobile phone networks carry packet data at rates of multiple Mbps. For comparison, a voice call is carried at a rate of 64 kbps, typically 3–4x less with compression.

To carry all this data, the UMTS core network nodes connect directly to a packet-switched network. The **SGSN** (**Serving GPRS Support Node**) and the **GGSN** (**Gateway GPRS Support Node**) deliver data packets to and from mobiles and interface to external packet networks such as the Internet.

This transition is set to continue in the mobile phone networks that are now being planned and deployed. Internet protocols are even used on mobiles to set up connections for voice calls over a packet data network, in the manner of voice-over-IP. IP and packets are used all the way from the radio access through to the core network. Of course, the way that IP networks are designed is also changing to support better quality of service. If it did not, then problems with chopped-up audio and jerky video would not impress paying customers. We will return to this subject in Chap. 5.

Another difference between mobile phone networks and the traditional Internet is mobility. When a user moves out of the range of one cellular base station and into the range of another one, the flow of data must be re-routed from the old to the new cell base station. This technique is known as **handover** or **handoff**, and it is illustrated in Fig. 1-32.

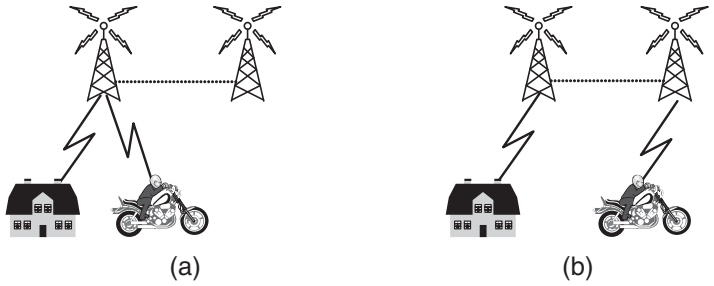


Figure 1-32. Mobile phone handover (a) before, (b) after.

Either the mobile device or the base station may request a handover when the quality of the signal drops. In some cell networks, usually those based on CDMA

technology, it is possible to connect to the new base station before disconnecting from the old base station. This improves the connection quality for the mobile because there is no break in service; the mobile is actually connected to two base stations for a short while. This way of doing a handover is called a **soft handover** to distinguish it from a **hard handover**, in which the mobile disconnects from the old base station before connecting to the new one.

A related issue is how to find a mobile in the first place when there is an incoming call. Each mobile phone network has a **HSS** (**Home Subscriber Server**) in the core network that knows the location of each subscriber, as well as other profile information that is used for authentication and authorization. In this way, each mobile can be found by contacting the HSS.

A final area to discuss is security. Historically, phone companies have taken security much more seriously than Internet companies for a long time because of the need to bill for service and avoid (payment) fraud. Unfortunately that is not saying much. Nevertheless, in the evolution from 1G through 3G technologies, mobile phone companies have been able to roll out some basic security mechanisms for mobiles.

Starting with the 2G GSM system, the mobile phone was divided into a handset and a removable chip containing the subscriber's identity and account information. The chip is informally called a **SIM card**, short for **Subscriber Identity Module**. SIM cards can be switched to different handsets to activate them, and they provide a basis for security. When GSM customers travel to other countries on vacation or business, they often bring their handsets but buy a new SIM card for few dollars upon arrival in order to make local calls with no roaming charges.

To reduce fraud, information on SIM cards is also used by the mobile phone network to authenticate subscribers and check that they are allowed to use the network. With UMTS, the mobile also uses the information on the SIM card to check that it is talking to a legitimate network.

Another aspect of security is privacy. Wireless signals are broadcast to all nearby receivers, so to make it difficult to eavesdrop on conversations, cryptographic keys on the SIM card are used to encrypt transmissions. This approach provides much better privacy than in 1G systems, which were easily tapped, but is not a panacea due to weaknesses in the encryption schemes.

Mobile phone networks are destined to play a central role in future networks. They are now more about mobile broadband applications than voice calls, and this has major implications for the air interfaces, core network architecture, and security of future networks. 4G technologies that are faster and better are on the drawing board under the name of **LTE** (**Long Term Evolution**), even as 3G design and deployment continues. Other wireless technologies also offer broadband Internet access to fixed and mobile clients, notably 802.16 networks under the common name of **WiMAX**. It is entirely possible that LTE and WiMAX are on a collision course with each other and it is hard to predict what will happen to them.

1.5.3 Wireless LANs: 802.11

Almost as soon as laptop computers appeared, many people had a dream of walking into an office and magically having their laptop computer be connected to the Internet. Consequently, various groups began working on ways to accomplish this goal. The most practical approach is to equip both the office and the laptop computers with short-range radio transmitters and receivers to allow them to talk.

Work in this field rapidly led to wireless LANs being marketed by a variety of companies. The trouble was that no two of them were compatible. The proliferation of standards meant that a computer equipped with a brand *X* radio would not work in a room equipped with a brand *Y* base station. In the mid 1990s, the industry decided that a wireless LAN standard might be a good idea, so the IEEE committee that had standardized wired LANs was given the task of drawing up a wireless LAN standard.

The first decision was the easiest: what to call it. All the other LAN standards had numbers like 802.1, 802.2, and 802.3, up to 802.10, so the wireless LAN standard was dubbed 802.11. A common slang name for it is **WiFi** but it is an important standard and deserves respect, so we will call it by its proper name, 802.11.

The rest was harder. The first problem was to find a suitable frequency band that was available, preferably worldwide. The approach taken was the opposite of that used in mobile phone networks. Instead of expensive, licensed spectrum, 802.11 systems operate in unlicensed bands such as the **ISM** (**Industrial, Scientific, and Medical**) bands defined by ITU-R (e.g., 902-928 MHz, 2.4-2.5 GHz, 5.725-5.825 GHz). All devices are allowed to use this spectrum provided that they limit their transmit power to let different devices coexist. Of course, this means that 802.11 radios may find themselves competing with cordless phones, garage door openers, and microwave ovens.

802.11 networks are made up of clients, such as laptops and mobile phones, and infrastructure called **APs** (**access points**) that is installed in buildings. Access points are sometimes called **base stations**. The access points connect to the wired network, and all communication between clients goes through an access point. It is also possible for clients that are in radio range to talk directly, such as two computers in an office without an access point. This arrangement is called an **ad hoc network**. It is used much less often than the access point mode. Both modes are shown in Fig. 1-33.

802.11 transmission is complicated by wireless conditions that vary with even small changes in the environment. At the frequencies used for 802.11, radio signals can be reflected off solid objects so that multiple echoes of a transmission may reach a receiver along different paths. The echoes can cancel or reinforce each other, causing the received signal to fluctuate greatly. This phenomenon is called **multipath fading**, and it is shown in Fig. 1-34.

The key idea for overcoming variable wireless conditions is **path diversity**, or the sending of information along multiple, independent paths. In this way, the

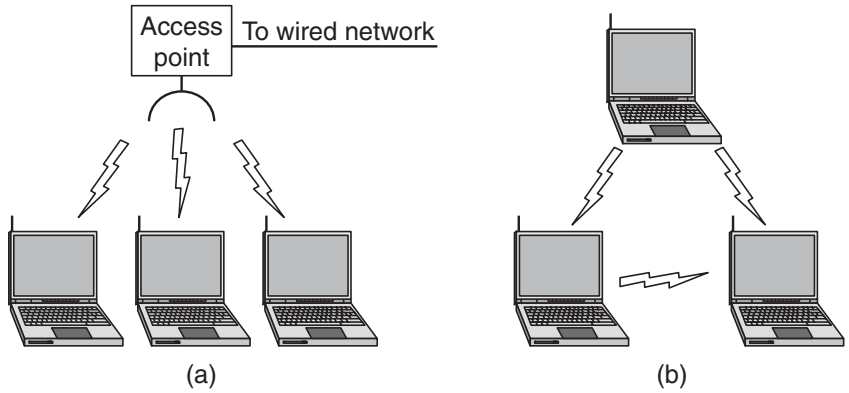


Figure 1-33. (a) Wireless network with an access point. (b) Ad hoc network.

information is likely to be received even if one of the paths happens to be poor due to a fade. These independent paths are typically built into the digital modulation scheme at the physical layer. Options include using different frequencies across the allowed band, following different spatial paths between different pairs of antennas, or repeating bits over different periods of time.

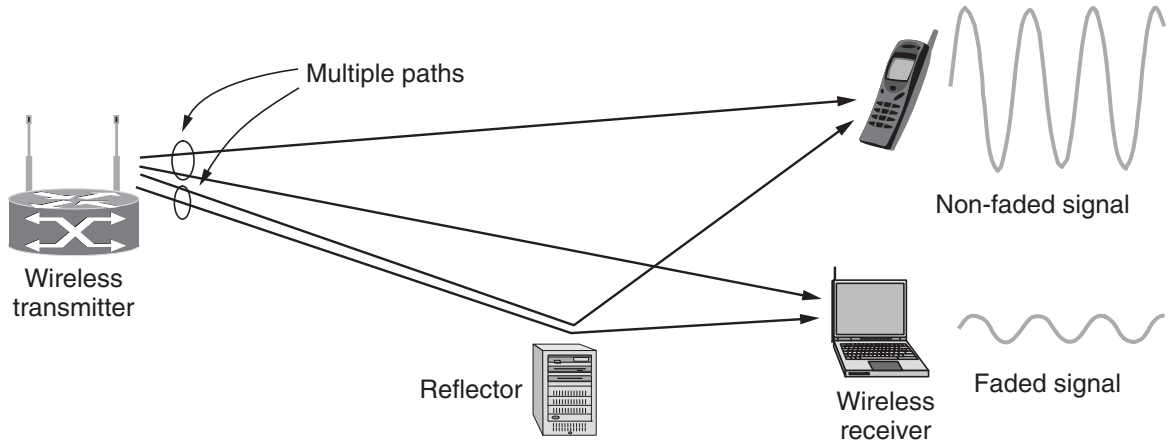


Figure 1-34. Multipath fading.

Different versions of 802.11 have used all of these techniques. The initial (1997) standard defined a wireless LAN that ran at either 1 Mbps or 2 Mbps by hopping between frequencies or spreading the signal across the allowed spectrum. Almost immediately, people complained that it was too slow, so work began on faster standards. The spread spectrum design was extended and became the (1999) 802.11b standard running at rates up to 11 Mbps. The 802.11a (1999) and 802.11g (2003) standards switched to a different modulation scheme called **OFDM** (**Orthogonal Frequency Division Multiplexing**). It divides a wide band of spectrum into many narrow slices over which different bits are sent in parallel. This improved scheme, which we will study in Chap. 2, boosted the 802.11a/g bit

rates up to 54 Mbps. That is a significant increase, but people still wanted more throughput to support more demanding uses. The latest version is 802.11n (2009). It uses wider frequency bands and up to four antennas per computer to achieve rates up to 450 Mbps.

Since wireless is inherently a broadcast medium, 802.11 radios also have to deal with the problem that multiple transmissions that are sent at the same time will collide, which may interfere with reception. To handle this problem, 802.11 uses a **CSMA** (**Carrier Sense Multiple Access**) scheme that draws on ideas from classic wired Ethernet, which, ironically, drew from an early wireless network developed in Hawaii and called **ALOHA**. Computers wait for a short random interval before transmitting, and defer their transmissions if they hear that someone else is already transmitting. This scheme makes it less likely that two computers will send at the same time. It does not work as well as in the case of wired networks, though. To see why, examine Fig. 1-35. Suppose that computer *A* is transmitting to computer *B*, but the radio range of *A*'s transmitter is too short to reach computer *C*. If *C* wants to transmit to *B* it can listen before starting, but the fact that it does not hear anything does not mean that its transmission will succeed. The inability of *C* to hear *A* before starting causes some collisions to occur. After any collision, the sender then waits another, longer, random delay and retransmits the packet. Despite this and some other issues, the scheme works well enough in practice.

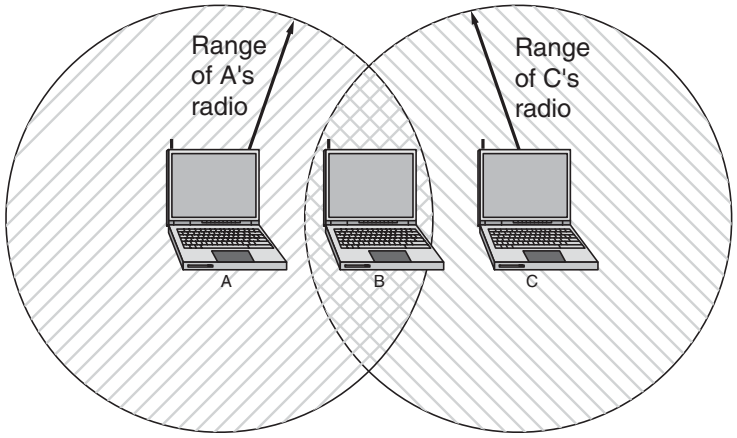


Figure 1-35. The range of a single radio may not cover the entire system.

Another problem is that of mobility. If a mobile client is moved away from the access point it is using and into the range of a different access point, some way of handing it off is needed. The solution is that an 802.11 network can consist of multiple cells, each with its own access point, and a distribution system that connects the cells. The distribution system is often switched Ethernet, but it can use any technology. As the clients move, they may find another access point with a better signal than the one they are currently using and change their association. From the outside, the entire system looks like a single wired LAN.

That said, mobility in 802.11 has been of limited value so far compared to mobility in the mobile phone network. Typically, 802.11 is used by nomadic clients that go from one fixed location to another, rather than being used on-the-go. Mobility is not really needed for nomadic usage. Even when 802.11 mobility is used, it extends over a single 802.11 network, which might cover at most a large building. Future schemes will need to provide mobility across different networks and across different technologies (e.g., 802.21).

Finally, there is the problem of security. Since wireless transmissions are broadcast, it is easy for nearby computers to receive packets of information that were not intended for them. To prevent this, the 802.11 standard included an encryption scheme known as **WEP** (**Wired Equivalent Privacy**). The idea was to make wireless security like that of wired security. It is a good idea, but unfortunately the scheme was flawed and soon broken (Borisov et al., 2001). It has since been replaced with newer schemes that have different cryptographic details in the 802.11i standard, also called **WiFi Protected Access**, initially called **WPA** but now replaced by **WPA2**.

802.11 has caused a revolution in wireless networking that is set to continue. Beyond buildings, it is starting to be installed in trains, planes, boats, and automobiles so that people can surf the Internet wherever they go. Mobile phones and all manner of consumer electronics, from game consoles to digital cameras, can communicate with it. We will come back to it in detail in Chap. 4.

1.5.4 RFID and Sensor Networks

The networks we have studied so far are made up of computing devices that are easy to recognize, from computers to mobile phones. With **Radio Frequency IDentification** (**RFID**), everyday objects can also be part of a computer network.

An RFID tag looks like a postage stamp-sized sticker that can be affixed to (or embedded in) an object so that it can be tracked. The object might be a cow, a passport, a book or a shipping pallet. The tag consists of a small microchip with a unique identifier and an antenna that receives radio transmissions. RFID readers installed at tracking points find tags when they come into range and interrogate them for their information as shown in Fig. 1-36. Applications include checking identities, managing the supply chain, timing races, and replacing barcodes.

There are many kinds of RFID, each with different properties, but perhaps the most fascinating aspect of RFID technology is that most RFID tags have neither an electric plug nor a battery. Instead, all of the energy needed to operate them is supplied in the form of radio waves by RFID readers. This technology is called **passive RFID** to distinguish it from the (less common) **active RFID** in which there is a power source on the tag.

One common form of RFID is **UHF RFID** (**Ultra-High Frequency RFID**). It is used on shipping pallets and some drivers licenses. Readers send signals in

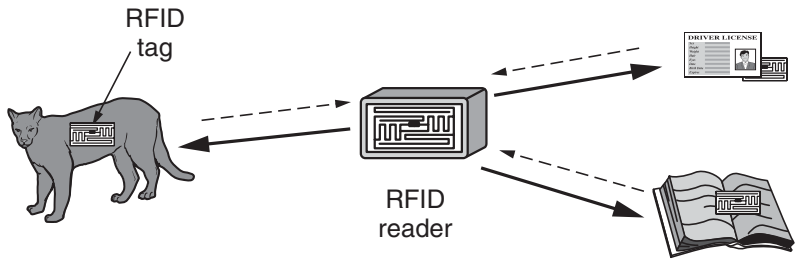


Figure 1-36. RFID used to network everyday objects.

the 902-928 MHz band in the United States. Tags communicate at distances of several meters by changing the way they reflect the reader signals; the reader is able to pick up these reflections. This way of operating is called **backscatter**.

Another popular kind of RFID is **HF RFID** (**High Frequency RFID**). It operates at 13.56 MHz and is likely to be in your passport, credit cards, books, and noncontact payment systems. HF RFID has a short range, typically a meter or less, because the physical mechanism is based on induction rather than backscatter. There are also other forms of RFID using other frequencies, such as **LF RFID** (**Low Frequency RFID**), which was developed before HF RFID and used for animal tracking. It is the kind of RFID likely to be in your cat.

RFID readers must somehow solve the problem of dealing with multiple tags within reading range. This means that a tag cannot simply respond when it hears a reader, or the signals from multiple tags may collide. The solution is similar to the approach taken in 802.11: tags wait for a short random interval before responding with their identification, which allows the reader to narrow down individual tags and interrogate them further.

Security is another problem. The ability of RFID readers to easily track an object, and hence the person who uses it, can be an invasion of privacy. Unfortunately, it is difficult to secure RFID tags because they lack the computation and communication power to run strong cryptographic algorithms. Instead, weak measures like passwords (which can easily be cracked) are used. If an identity card can be remotely read by an official at a border, what is to stop the same card from being tracked by other people without your knowledge? Not much.

RFID tags started as identification chips, but are rapidly turning into full-fledged computers. For example, many tags have memory that can be updated and later queried, so that information about what has happened to the tagged object can be stored with it. Rieback et al. (2006) demonstrated that this means that all of the usual problems of computer malware apply, only now your cat or your passport might be used to spread an RFID virus.

A step up in capability from RFID is the **sensor network**. Sensor networks are deployed to monitor aspects of the physical world. So far, they have mostly been used for scientific experimentation, such as monitoring bird habitats, volcanic activity, and zebra migration, but business applications including healthcare,

monitoring equipment for vibration, and tracking of frozen, refrigerated, or otherwise perishable goods cannot be too far behind.

Sensor nodes are small computers, often the size of a key fob, that have temperature, vibration, and other sensors. Many nodes are placed in the environment that is to be monitored. Typically, they have batteries, though they may scavenge energy from vibrations or the sun. As with RFID, having enough energy is a key challenge, and the nodes must communicate carefully to be able to deliver their sensor information to an external collection point. A common strategy is for the nodes to self-organize to relay messages for each other, as shown in Fig. 1-37. This design is called a **multihop network**.

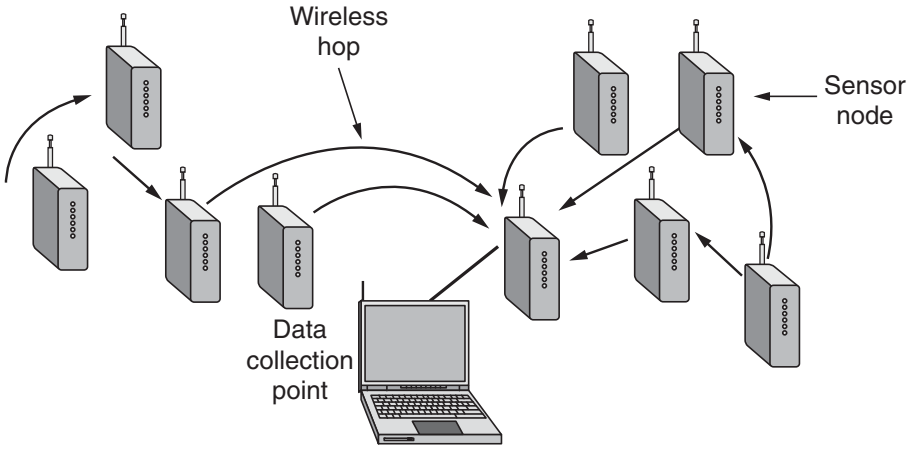


Figure 1-37. Multihop topology of a sensor network.

RFID and sensor networks are likely to become much more capable and pervasive in the future. Researchers have already combined the best of both technologies by prototyping programmable RFID tags with light, movement, and other sensors (Sample et al., 2008).

1.6 NETWORK STANDARDIZATION

Many network vendors and suppliers exist, each with its own ideas of how things should be done. Without coordination, there would be complete chaos, and users would get nothing done. The only way out is to agree on some network standards. Not only do good standards allow different computers to communicate, but they also increase the market for products adhering to the standards. A larger market leads to mass production, economies of scale in manufacturing, better implementations, and other benefits that decrease price and further increase acceptance.

In this section we will take a quick look at the important but little-known, world of international standardization. But let us first discuss what belongs in a

standard. A reasonable person might assume that a standard tells you how a protocol should work so that you can do a good job of implementing it. That person would be wrong.

Standards define what is needed for interoperability: no more, no less. That lets the larger market emerge and also lets companies compete on the basis of how good their products are. For example, the 802.11 standard defines many transmission rates but does not say when a sender should use which rate, which is a key factor in good performance. That is up to whoever makes the product. Often getting to interoperability this way is difficult, since there are many implementation choices and standards usually define many options. For 802.11, there were so many problems that, in a strategy that has become common practice, a trade group called the **WiFi Alliance** was started to work on interoperability within the 802.11 standard.

Similarly, a protocol standard defines the protocol over the wire but not the service interface inside the box, except to help explain the protocol. Real service interfaces are often proprietary. For example, the way TCP interfaces to IP within a computer does not matter for talking to a remote host. It only matters that the remote host speaks TCP/IP. In fact, TCP and IP are commonly implemented together without any distinct interface. That said, good service interfaces, like good APIs, are valuable for getting protocols used, and the best ones (such as Berkeley sockets) can become very popular.

Standards fall into two categories: de facto and de jure. **De facto** (Latin for ''from the fact'') standards are those that have just happened, without any formal plan. HTTP, the protocol on which the Web runs, started life as a de facto standard. It was part of early WWW browsers developed by Tim Berners-Lee at CERN, and its use took off with the growth of the Web. Bluetooth is another example. It was originally developed by Ericsson but now everyone is using it.

De jure (Latin for ''by law'') standards, in contrast, are adopted through the rules of some formal standardization body. International standardization authorities are generally divided into two classes: those established by treaty among national governments, and those comprising voluntary, nontreaty organizations. In the area of computer network standards, there are several organizations of each type, notably ITU, ISO, IETF and IEEE, all of which we will discuss below.

In practice, the relationships between standards, companies, and standardization bodies are complicated. De facto standards often evolve into de jure standards, especially if they are successful. This happened in the case of HTTP, which was quickly picked up by IETF. Standards bodies often ratify each others' standards, in what looks like patting one another on the back, to increase the market for a technology. These days, many ad hoc business alliances that are formed around particular technologies also play a significant role in developing and refining network standards. For example, **3GPP** (**Third Generation Partnership Project**) is a collaboration between telecommunications associations that drives the UMTS 3G mobile phone standards.

1.6.1 Who's Who in the Telecommunications World

The legal status of the world's telephone companies varies considerably from country to country. At one extreme is the United States, which has over 2000 separate, (mostly very small) privately owned telephone companies. A few more were added with the breakup of AT&T in 1984 (which was then the world's largest corporation, providing telephone service to about 80 percent of America's telephones), and the Telecommunications Act of 1996 that overhauled regulation to foster competition.

At the other extreme are countries in which the national government has a complete monopoly on all communication, including the mail, telegraph, telephone, and often radio and television. Much of the world falls into this category. In some cases the telecommunication authority is a nationalized company, and in others it is simply a branch of the government, usually known as the **PTT** (**Post, Telegraph & Telephone** administration). Worldwide, the trend is toward liberalization and competition and away from government monopoly. Most European countries have now (partially) privatized their PTTs, but elsewhere the process is still only slowly gaining steam.

With all these different suppliers of services, there is clearly a need to provide compatibility on a worldwide scale to ensure that people (and computers) in one country can call their counterparts in another one. Actually, this need has existed for a long time. In 1865, representatives from many European governments met to form the predecessor to today's **ITU** (**International Telecommunication Union**). Its job was to standardize international telecommunications, which in those days meant telegraphy. Even then it was clear that if half the countries used Morse code and the other half used some other code, there was going to be a problem. When the telephone was put into international service, ITU took over the job of standardizing telephony (pronounced te-LEF-ony) as well. In 1947, ITU became an agency of the United Nations.

ITU has about 200 governmental members, including almost every member of the United Nations. Since the United States does not have a PTT, somebody else had to represent it in ITU. This task fell to the State Department, probably on the grounds that ITU had to do with foreign countries, the State Department's specialty. ITU also has more than 700 sector and associate members. They include telephone companies (e.g., AT&T, Vodafone, Sprint), telecom equipment manufacturers (e.g., Cisco, Nokia, Nortel), computer vendors (e.g., Microsoft, Agilent, Toshiba), chip manufacturers (e.g., Intel, Motorola, TI), and other interested companies (e.g., Boeing, CBS, VeriSign).

ITU has three main sectors. We will focus primarily on **ITU-T**, the Telecommunications Standardization Sector, which is concerned with telephone and data communication systems. Before 1993, this sector was called **CCITT**, which is an acronym for its French name, Comité Consultatif International Télégraphique et Téléphonique. **ITU-R**, the Radiocommunications Sector, is concerned with

coordinating the use by competing interest groups of radio frequencies worldwide. The other sector is ITU-D, the Development Sector. It promotes the development of information and communication technologies to narrow the "digital divide" between countries with effective access to the information technologies and countries with limited access.

ITU-T's task is to make technical recommendations about telephone, telegraph, and data communication interfaces. These often become internationally recognized standards, though technically the recommendations are only suggestions that governments can adopt or ignore, as they wish (because governments are like 13-year-old boys—they do not take kindly to being given orders). In practice, a country that wishes to adopt a telephone standard different from that used by the rest of the world is free to do so, but at the price of cutting itself off from everyone else. This might work for North Korea, but elsewhere it would be a real problem.

The real work of ITU-T is done in its Study Groups. There are currently 10 Study Groups, often as large as 400 people, that cover topics ranging from telephone billing to multimedia services to security. SG 15, for example, standardizes the DSL technologies popularly used to connect to the Internet. In order to make it possible to get anything at all done, the Study Groups are divided into Working Parties, which are in turn divided into Expert Teams, which are in turn divided into ad hoc groups. Once a bureaucracy, always a bureaucracy.

Despite all this, ITU-T actually does get things done. Since its inception, it has produced more than 3000 recommendations, many of which are widely used in practice. For example, Recommendation H.264 (also an ISO standard known as MPEG-4 AVC) is widely used for video compression, and X.509 public key certificates are used for secure Web browsing and digitally signed email.

As the field of telecommunications completes the transition started in the 1980s from being entirely national to being entirely global, standards will become increasingly important, and more and more organizations will want to become involved in setting them. For more information about ITU, see Irmer (1994).

1.6.2 Who's Who in the International Standards World

International standards are produced and published by **ISO** (**International Standards Organization**†), a voluntary nontreaty organization founded in 1946. Its members are the national standards organizations of the 157 member countries. These members include ANSI (U.S.), BSI (Great Britain), AFNOR (France), DIN (Germany), and 153 others.

ISO issues standards on a truly vast number of subjects, ranging from nuts and bolts (literally) to telephone pole coatings [not to mention cocoa beans (ISO 2451), fishing nets (ISO 1530), women's underwear (ISO 4416) and quite a few

† For the purist, ISO's true name is the International Organization for Standardization.

other subjects one might not think were subject to standardization]. On issues of telecommunication standards, ISO and ITU-T often cooperate (ISO is a member of ITU-T) to avoid the irony of two official and mutually incompatible international standards.

Over 17,000 standards have been issued, including the OSI standards. ISO has over 200 Technical Committees (TCs), numbered in the order of their creation, each dealing with a specific subject. TC1 deals with the nuts and bolts (standardizing screw thread pitches). JTC1 deals with information technology, including networks, computers, and software. It is the first (and so far only) Joint Technical Committee, created in 1987 by merging TC97 with activities in IEC, yet another standardization body. Each TC has subcommittees (SCs) divided into working groups (WGs).

The real work is done largely in the WGs by over 100,000 volunteers worldwide. Many of these ''volunteers'' are assigned to work on ISO matters by their employers, whose products are being standardized. Others are government officials keen on having their country's way of doing things become the international standard. Academic experts also are active in many of the WGs.

The procedure used by ISO for adopting standards has been designed to achieve as broad a consensus as possible. The process begins when one of the national standards organizations feels the need for an international standard in some area. A working group is then formed to come up with a **CD** (**Committee Draft**). The CD is then circulated to all the member bodies, which get 6 months to criticize it. If a substantial majority approves, a revised document, called a **DIS** (**Draft International Standard**) is produced and circulated for comments and voting. Based on the results of this round, the final text of the **IS** (**International Standard**) is prepared, approved, and published. In areas of great controversy, a CD or DIS may have to go through several versions before acquiring enough votes, and the whole process can take years.

NIST (**National Institute of Standards and Technology**) is part of the U.S. Department of Commerce. It used to be called the National Bureau of Standards. It issues standards that are mandatory for purchases made by the U.S. Government, except for those of the Department of Defense, which defines its own standards.

Another major player in the standards world is **IEEE** (**Institute of Electrical and Electronics Engineers**), the largest professional organization in the world. In addition to publishing scores of journals and running hundreds of conferences each year, IEEE has a standardization group that develops standards in the area of electrical engineering and computing. IEEE's 802 committee has standardized many kinds of LANs. We will study some of its output later in this book. The actual work is done by a collection of working groups, which are listed in Fig. 1-38. The success rate of the various 802 working groups has been low; having an 802.x number is no guarantee of success. Still, the impact of the success stories (especially 802.3 and 802.11) on the industry and the world has been enormous.

Number	Topic
802.1	Overview and architecture of LANs
802.2 ↓	Logical link control
802.3 *	Ethernet
802.4 ↓	Token bus (was briefly used in manufacturing plants)
802.5	Token ring (IBM's entry into the LAN world)
802.6 ↓	Dual queue dual bus (early metropolitan area network)
802.7 ↓	Technical advisory group on broadband technologies
802.8 †	Technical advisory group on fiber optic technologies
802.9 ↓	Isochronous LANs (for real-time applications)
802.10 ↓	Virtual LANs and security
802.11 *	Wireless LANs (WiFi)
802.12 ↓	Demand priority (Hewlett-Packard's AnyLAN)
802.13	Unlucky number; nobody wanted it
802.14 ↓	Cable modems (defunct: an industry consortium got there first)
802.15 *	Personal area networks (Bluetooth, Zigbee)
802.16 *	Broadband wireless (WiMAX)
802.17	Resilient packet ring
802.18	Technical advisory group on radio regulatory issues
802.19	Technical advisory group on coexistence of all these standards
802.20	Mobile broadband wireless (similar to 802.16e)
802.21	Media independent handoff (for roaming over technologies)
802.22	Wireless regional area network

Figure 1-38. The 802 working groups. The important ones are marked with *. The ones marked with ↓ are hibernating. The one marked with † gave up and disbanded itself.

1.6.3 Who's Who in the Internet Standards World

The worldwide Internet has its own standardization mechanisms, very different from those of ITU-T and ISO. The difference can be crudely summed up by saying that the people who come to ITU or ISO standardization meetings wear suits, while the people who come to Internet standardization meetings wear jeans (except when they meet in San Diego, when they wear shorts and T-shirts).

ITU-T and ISO meetings are populated by corporate officials and government civil servants for whom standardization is their job. They regard standardization as a Good Thing and devote their lives to it. Internet people, on the other hand, prefer anarchy as a matter of principle. However, with hundreds of millions of

people all doing their own thing, little communication can occur. Thus, standards, however regrettable, are sometimes needed. In this context, David Clark of M.I.T. once made a now-famous remark about Internet standardization consisting of "rough consensus and running code."

When the ARPANET was set up, DoD created an informal committee to oversee it. In 1983, the committee was renamed the **IAB** (**Internet Activities Board**) and was given a slighter broader mission, namely, to keep the researchers involved with the ARPANET and the Internet pointed more or less in the same direction, an activity not unlike herding cats. The meaning of the acronym "IAB" was later changed to **Internet Architecture Board**.

Each of the approximately ten members of the IAB headed a task force on some issue of importance. The IAB met several times a year to discuss results and to give feedback to the DoD and NSF, which were providing most of the funding at this time. When a standard was needed (e.g., a new routing algorithm), the IAB members would thrash it out and then announce the change so the graduate students who were the heart of the software effort could implement it. Communication was done by a series of technical reports called **RFCs** (**Request For Comments**). RFCs are stored online and can be fetched by anyone interested in them from *www.ietf.org/rfc*. They are numbered in chronological order of creation. Over 5000 now exist. We will refer to many RFCs in this book.

By 1989, the Internet had grown so large that this highly informal style no longer worked. Many vendors by then offered TCP/IP products and did not want to change them just because ten researchers had thought of a better idea. In the summer of 1989, the IAB was reorganized again. The researchers were moved to the **IRTF** (**Internet Research Task Force**), which was made subsidiary to IAB, along with the **IETF** (**Internet Engineering Task Force**). The IAB was repopulated with people representing a broader range of organizations than just the research community. It was initially a self-perpetuating group, with members serving for a 2-year term and new members being appointed by the old ones. Later, the **Internet Society** was created, populated by people interested in the Internet. The Internet Society is thus in a sense comparable to ACM or IEEE. It is governed by elected trustees who appoint the IAB's members.

The idea of this split was to have the IRTF concentrate on long-term research while the IETF dealt with short-term engineering issues. The IETF was divided up into working groups, each with a specific problem to solve. The chairmen of these working groups initially met as a steering committee to direct the engineering effort. The working group topics include new applications, user information, OSI integration, routing and addressing, security, network management, and standards. Eventually, so many working groups were formed (more than 70) that they were grouped into areas and the area chairmen met as the steering committee.

In addition, a more formal standardization process was adopted, patterned after ISOs. To become a **Proposed Standard**, the basic idea must be explained in an RFC and have sufficient interest in the community to warrant consideration.

To advance to the **Draft Standard** stage, a working implementation must have been rigorously tested by at least two independent sites for at least 4 months. If the IAB is convinced that the idea is sound and the software works, it can declare the RFC to be an **Internet Standard**. Some Internet Standards have become DoD standards (MIL-STD), making them mandatory for DoD suppliers.

For Web standards, the **World Wide Web Consortium** (**W3C**) develops protocols and guidelines to facilitate the long-term growth of the Web. It is an industry consortium led by Tim Berners-Lee and set up in 1994 as the Web really begun to take off. W3C now has more than 300 members from around the world and has produced more than 100 W3C Recommendations, as its standards are called, covering topics such as HTML and Web privacy.

1.7 METRIC UNITS

To avoid any confusion, it is worth stating explicitly that in this book, as in computer science in general, metric units are used instead of traditional English units (the furlong-stone-fortnight system). The principal metric prefixes are listed in Fig. 1-39. The prefixes are typically abbreviated by their first letters, with the units greater than 1 capitalized (KB, MB, etc.). One exception (for historical reasons) is kbps for kilobits/sec. Thus, a 1-Mbps communication line transmits 10^6 bits/sec and a 100-psec (or 100-ps) clock ticks every 10^{-10} seconds. Since milli and micro both begin with the letter "m," a choice had to be made. Normally, "m" is used for milli and "μ" (the Greek letter mu) is used for micro.

Exp.	Explicit	Prefix	Exp.	Explicit	Prefix
10^{-3}	0.001	milli	10^{3}	1,000	Kilo
10^{-6}	0.000001	micro	10^{6}	1,000,000	Mega
10^{-9}	0.000000001	nano	10^{9}	1,000,000,000	Giga
10^{-12}	0.000000000001	pico	10^{12}	1,000,000,000,000	Tera
10^{-15}	0.000000000000001	femto	10^{15}	1,000,000,000,000,000	Peta
10^{-18}	0.000000000000000001	atto	10^{18}	1,000,000,000,000,000,000	Exa
10^{-21}	0.000000000000000000001	zepto	10^{21}	1,000,000,000,000,000,000,000	Zetta
10^{-24}	0.000000000000000000000001	yocto	10^{24}	1,000,000,000,000,000,000,000,000	Yotta

Figure 1-39. The principal metric prefixes.

It is also worth pointing out that for measuring memory, disk, file, and database sizes, in common industry practice, the units have slightly different meanings. There, kilo means 2^{10} (1024) rather than 10^3 (1000) because memories are always a power of two. Thus, a 1-KB memory contains 1024 bytes, not 1000 bytes. Note also the capital "B" in that usage to mean "bytes" (units of eight

bits), instead of a lowercase "b" that means "bits." Similarly, a 1-MB memory contains 2^{20} (1,048,576) bytes, a 1-GB memory contains 2^{30} (1,073,741,824) bytes, and a 1-TB database contains 2^{40} (1,099,511,627,776) bytes. However, a 1-kbps communication line transmits 1000 bits per second and a 10-Mbps LAN runs at 10,000,000 bits/sec because these speeds are not powers of two. Unfortunately, many people tend to mix up these two systems, especially for disk sizes. To avoid ambiguity, in this book, we will use the symbols KB, MB, GB, and TB for 2^{10}, 2^{20}, 2^{30}, and 2^{40} bytes, respectively, and the symbols kbps, Mbps, Gbps, and Tbps for 10^3, 10^6, 10^9, and 10^{12} bits/sec, respectively.

1.8 OUTLINE OF THE REST OF THE BOOK

This book discusses both the principles and practice of computer networking. Most chapters start with a discussion of the relevant principles, followed by a number of examples that illustrate these principles. These examples are usually taken from the Internet and wireless networks such as the mobile phone network since these are both important and very different. Other examples will be given where relevant.

The book is structured according to the hybrid model of Fig. 1-23. Starting with Chap. 2, we begin working our way up the protocol hierarchy beginning at the bottom. We provide some background in the field of data communication that covers both wired and wireless transmission systems. This material is concerned with how to deliver information over physical channels, although we cover only the architectural rather than the hardware aspects. Several examples of the physical layer, such as the public switched telephone network, the mobile telephone network, and the cable television network are also discussed.

Chapters 3 and 4 discuss the data link layer in two parts. Chap. 3 looks at the problem of how to send packets across a link, including error detection and correction. We look at DSL (used for broadband Internet access over phone lines) as a real-world example of a data link protocol.

In Chap. 4, we examine the medium access sublayer. This is the part of the data link layer that deals with how to share a channel between multiple computers. The examples we look at include wireless, such as 802.11 and RFID, and wired LANs such as classic Ethernet. Link layer switches that connect LANs, such as switched Ethernet, are also discussed here.

Chapter 5 deals with the network layer, especially routing. Many routing algorithms, both static and dynamic, are covered. Even with good routing algorithms, though, if more traffic is offered than the network can handle, some packets will be delayed or discarded. We discuss this issue from how to prevent congestion to how to guarantee a certain quality of service. Connecting heterogeneous networks to form internetworks also leads to numerous problems that are discussed here. The network layer in the Internet is given extensive coverage.

Chapter 6 deals with the transport layer. Much of the emphasis is on connection-oriented protocols and reliability, since many applications need these. Both Internet transport protocols, UDP and TCP, are covered in detail, as are their performance issues.

Chapter 7 deals with the application layer, its protocols, and its applications. The first topic is DNS, which is the Internet's telephone book. Next comes email, including a discussion of its protocols. Then we move on to the Web, with detailed discussions of static and dynamic content, and what happens on the client and server sides. We follow this with a look at networked multimedia, including streaming audio and video. Finally, we discuss content-delivery networks, including peer-to-peer technology.

Chapter 8 is about network security. This topic has aspects that relate to all layers, so it is easiest to treat it after all the layers have been thoroughly explained. The chapter starts with an introduction to cryptography. Later, it shows how cryptography can be used to secure communication, email, and the Web. The chapter ends with a discussion of some areas in which security collides with privacy, freedom of speech, censorship, and other social issues.

Chapter 9 contains an annotated list of suggested readings arranged by chapter. It is intended to help those readers who would like to pursue their study of networking further. The chapter also has an alphabetical bibliography of all the references cited in this book.

The authors' Web site at Pearson:

http://www.pearsonhighered.com/tanenbaum

has a page with links to many tutorials, FAQs, companies, industry consortia, professional organizations, standards organizations, technologies, papers, and more.

1.9 SUMMARY

Computer networks have many uses, both for companies and for individuals, in the home and while on the move. Companies use networks of computers to share corporate information, typically using the client-server model with employee desktops acting as clients accessing powerful servers in the machine room. For individuals, networks offer access to a variety of information and entertainment resources, as well as a way to buy and sell products and services. Individuals often access the Internet via their phone or cable providers at home, though increasingly wireless access is used for laptops and phones. Technology advances are enabling new kinds of mobile applications and networks with computers embedded in appliances and other consumer devices. The same advances raise social issues such as privacy concerns.

Roughly speaking, networks can be divided into LANs, MANs, WANs, and internetworks. LANs typical cover a building and operate at high speeds. MANs

usually cover a city. An example is the cable television system, which is now used by many people to access the Internet. WANs may cover a country or a continent. Some of the technologies used to build these networks are point-to-point (e.g., a cable) while others are broadcast (e.g.,wireless). Networks can be interconnected with routers to form internetworks, of which the Internet is the largest and best known example. Wireless networks, for example 802.11 LANs and 3G mobile telephony, are also becoming extremely popular.

Network software is built around protocols, which are rules by which processes communicate. Most networks support protocol hierarchies, with each layer providing services to the layer above it and insulating them from the details of the protocols used in the lower layers. Protocol stacks are typically based either on the OSI model or on the TCP/IP model. Both have link, network, transport, and application layers, but they differ on the other layers. Design issues include reliability, resource allocation, growth, security, and more. Much of this book deals with protocols and their design.

Networks provide various services to their users. These services can range from connectionless best-efforts packet delivery to connection-oriented guaranteed delivery. In some networks, connectionless service is provided in one layer and connection-oriented service is provided in the layer above it.

Well-known networks include the Internet, the 3G mobile telephone network, and 802.11 LANs. The Internet evolved from the ARPANET, to which other networks were added to form an internetwork. The present-day Internet is actually a collection of many thousands of networks that use the TCP/IP protocol stack. The 3G mobile telephone network provides wireless and mobile access to the Internet at speeds of multiple Mbps, and, of course, carries voice calls as well. Wireless LANs based on the IEEE 802.11 standard are deployed in many homes and cafes and can provide connectivity at rates in excess of 100 Mbps. New kinds of networks are emerging too, such as embedded sensor networks and networks based on RFID technology.

Enabling multiple computers to talk to each other requires a large amount of standardization, both in the hardware and software. Organizations such as ITU-T, ISO, IEEE, and IAB manage different parts of the standardization process.

PROBLEMS

1. Imagine that you have trained your St. Bernard, Bernie, to carry a box of three 8-mm tapes instead of a flask of brandy. (When your disk fills up, you consider that an emergency.) These tapes each contain 7 gigabytes. The dog can travel to your side, wherever you may be, at 18 km/hour. For what range of distances does Bernie have a higher data rate than a transmission line whose data rate (excluding overhead) is 150 Mbps? How does your answer change if (i) Bernie's speed is doubled; (ii) each tape capacity is doubled; (iii) the data rate of the transmission line is doubled.

Option	Description
Security	Specifies how secret the datagram is
Strict source routing	Gives the complete path to be followed
Loose source routing	Gives a list of routers not to be missed
Record route	Makes each router append its IP address
Timestamp	Makes each router append its address and timestamp

Figure 5-47. Some of the IP options.

exact route. It is most useful for system managers who need to send emergency packets when the routing tables have been corrupted, or for making timing measurements.

The *Loose source routing* option requires the packet to traverse the list of routers specified, in the order specified, but it is allowed to pass through other routers on the way. Normally, this option will provide only a few routers, to force a particular path. For example, to force a packet from London to Sydney to go west instead of east, this option might specify routers in New York, Los Angeles, and Honolulu. This option is most useful when political or economic considerations dictate passing through or avoiding certain countries.

The *Record route* option tells each router along the path to append its IP address to the *Options* field. This allows system managers to track down bugs in the routing algorithms (“Why are packets from Houston to Dallas visiting Tokyo first?”). When the ARPANET was first set up, no packet ever passed through more than nine routers, so 40 bytes of options was plenty. As mentioned above, now it is too small.

Finally, the *Timestamp* option is like the *Record route* option, except that in addition to recording its 32-bit IP address, each router also records a 32-bit timestamp. This option, too, is mostly useful for network measurement.

Today, IP options have fallen out of favor. Many routers ignore them or do not process them efficiently, shunting them to the side as an uncommon case. That is, they are only partly supported and they are rarely used.

5.6.2 IP Addresses

A defining feature of IPv4 is its 32-bit addresses. Every host and router on the Internet has an IP address that can be used in the *Source address* and *Destination address* fields of IP packets. It is important to note that an IP address does not actually refer to a host. It really refers to a network interface, so if a host is on two networks, it must have two IP addresses. However, in practice, most hosts are on one network and thus have one IP address. In contrast, routers have multiple interfaces and thus multiple IP addresses.

Prefixes

IP addresses are hierarchical, unlike Ethernet addresses. Each 32-bit address is comprised of a variable-length network portion in the top bits and a host portion in the bottom bits. The network portion has the same value for all hosts on a single network, such as an Ethernet LAN. This means that a network corresponds to a contiguous block of IP address space. This block is called a **prefix**.

IP addresses are written in **dotted decimal notation**. In this format, each of the 4 bytes is written in decimal, from 0 to 255. For example, the 32-bit hexadecimal address 80D00297 is written as 128.208.2.151. Prefixes are written by giving the lowest IP address in the block and the size of the block. The size is determined by the number of bits in the network portion; the remaining bits in the host portion can vary. This means that the size must be a power of two. By convention, it is written after the prefix IP address as a slash followed by the length in bits of the network portion. In our example, if the prefix contains 2^8 addresses and so leaves 24 bits for the network portion, it is written as 128.208.0.0/24.

Since the prefix length cannot be inferred from the IP address alone, routing protocols must carry the prefixes to routers. Sometimes prefixes are simply described by their length, as in a "/16" which is pronounced "slash 16." The length of the prefix corresponds to a binary mask of 1s in the network portion. When written out this way, it is called a **subnet mask**. It can be ANDed with the IP address to extract only the network portion. For our example, the subnet mask is 255.255.255.0. Fig. 5-48 shows a prefix and a subnet mask.

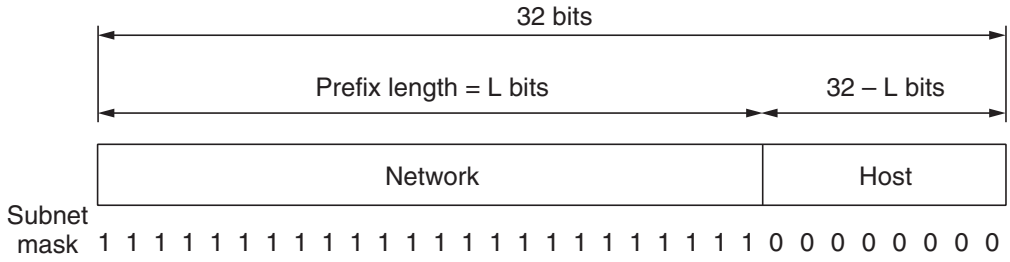


Figure 5-48. An IP prefix and a subnet mask.

Hierarchical addresses have significant advantages and disadvantages. The key advantage of prefixes is that routers can forward packets based on only the network portion of the address, as long as each of the networks has a unique address block. The host portion does not matter to the routers because all hosts on the same network will be sent in the same direction. It is only when the packets reach the network for which they are destined that they are forwarded to the correct host. This makes the routing tables much smaller than they would otherwise be. Consider that the number of hosts on the Internet is approaching one billion. That would be a very large table for every router to keep. However, by using a hierarchy, routers need to keep routes for only around 300,000 prefixes.

While using a hierarchy lets Internet routing scale, it has two disadvantages. First, the IP address of a host depends on where it is located in the network. An Ethernet address can be used anywhere in the world, but every IP address belongs to a specific network, and routers will only be able to deliver packets destined to that address to the network. Designs such as mobile IP are needed to support hosts that move between networks but want to keep the same IP addresses.

The second disadvantage is that the hierarchy is wasteful of addresses unless it is carefully managed. If addresses are assigned to networks in (too) large blocks, there will be (many) addresses that are allocated but not in use. This allocation would not matter much if there were plenty of addresses to go around. However, it was realized more than two decades ago that the tremendous growth of the Internet was rapidly depleting the free address space. IPv6 is the solution to this shortage, but until it is widely deployed there will be great pressure to allocate IP addresses so that they are used very efficiently.

Subnets

Network numbers are managed by a nonprofit corporation called **ICANN** (**Internet Corporation for Assigned Names and Numbers**), to avoid conflicts. In turn, ICANN has delegated parts of the address space to various regional authorities, which dole out IP addresses to ISPs and other companies. This is the process by which a company is allocated a block of IP addresses.

However, this process is only the start of the story, as IP address assignment is ongoing as companies grow. We have said that routing by prefix requires all the hosts in a network to have the same network number. This property can cause problems as networks grow. For example, consider a university that started out with our example /16 prefix for use by the Computer Science Dept. for the computers on its Ethernet. A year later, the Electrical Engineering Dept. wants to get on the Internet. The Art Dept. soon follows suit. What IP addresses should these departments use? Getting further blocks requires going outside the university and may be expensive or inconvenient. Moreover, the /16 already allocated has enough addresses for over 60,000 hosts. It might be intended to allow for significant growth, but until that happens, it is wasteful to allocate further blocks of IP addresses to the same university. A different organization is required.

The solution is to allow the block of addresses to be split into several parts for internal use as multiple networks, while still acting like a single network to the outside world. This is called **subnetting** and the networks (such as Ethernet LANs) that result from dividing up a larger network are called **subnets**. As we mentioned in Chap. 1, you should be aware that this new usage of the term conflicts with older usage of "subnet" to mean the set of all routers and communication lines in a network.

Fig. 5-49 shows how subnets can help with our example. The single /16 has been split into pieces. This split does not need to be even, but each piece must be

aligned so that any bits can be used in the lower host portion. In this case, half of the block (a /17) is allocated to the Computer Science Dept, a quarter is allocated to the Electrical Engineering Dept. (a /18), and one eighth (a /19) to the Art Dept. The remaining eighth is unallocated. A different way to see how the block was divided is to look at the resulting prefixes when written in binary notation:

Computer Science:	10000000	11010000	1|xxxxxxx	xxxxxxxx
Electrical Eng.:	10000000	11010000	00|xxxxxx	xxxxxxxx
Art:	10000000	11010000	011|xxxxx	xxxxxxxx

Here, the vertical bar (|) shows the boundary between the subnet number and the host portion.

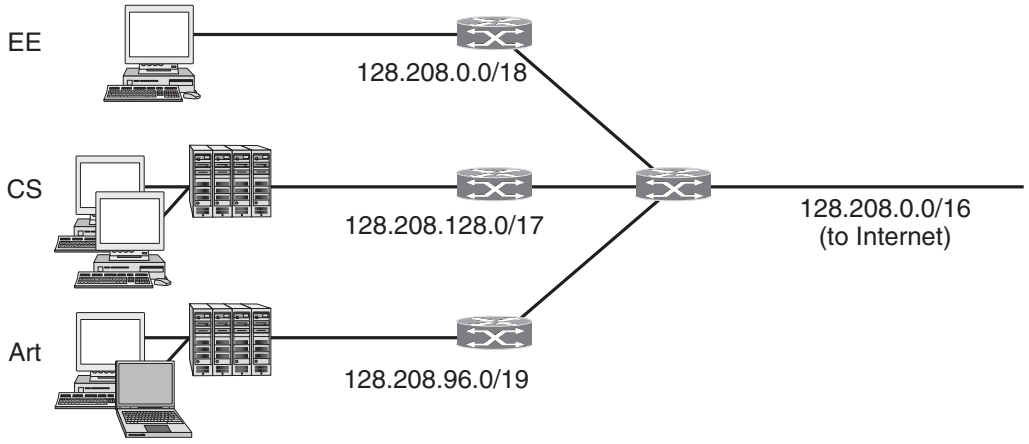


Figure 5-49. Splitting an IP prefix into separate networks with subnetting.

When a packet comes into the main router, how does the router know which subnet to give it to? This is where the details of our prefixes come in. One way would be for each router to have a table with 65,536 entries telling it which outgoing line to use for each host on campus. But this would undermine the main scaling benefit we get from using a hierarchy. Instead, the routers simply need to know the subnet masks for the networks on campus.

When a packet arrives, the router looks at the destination address of the packet and checks which subnet it belongs to. The router can do this by ANDing the destination address with the mask for each subnet and checking to see if the result is the corresponding prefix. For example, consider a packet destined for IP address 128.208.2.151. To see if it is for the Computer Science Dept., we AND with 255.255.128.0 to take the first 17 bits (which is 128.208.0.0) and see if they match the prefix address (which is 128.208.128.0). They do not match. Checking the first 18 bits for the Electrical Engineering Dept., we get 128.208.0.0 when ANDing with the subnet mask. This does match the prefix address, so the packet is forwarded onto the interface which leads to the Electrical Engineering network.

The subnet divisions can be changed later if necessary, by updating all subnet masks at routers inside the university. Outside the network, the subnetting is not visible, so allocating a new subnet does not require contacting ICANN or changing any external databases.

CIDR—Classless InterDomain Routing

Even if blocks of IP addresses are allocated so that the addresses are used efficiently, there is still a problem that remains: routing table explosion.

Routers in organizations at the edge of a network, such as a university, need to have an entry for each of their subnets, telling the router which line to use to get to that network. For routes to destinations outside of the organization, they can use the simple default rule of sending the packets on the line toward the ISP that connects the organization to the rest of the Internet. The other destination addresses must all be out there somewhere.

Routers in ISPs and backbones in the middle of the Internet have no such luxury. They must know which way to go to get to every network and no simple default will work. These core routers are said to be in the **default-free zone** of the Internet. No one really knows how many networks are connected to the Internet any more, but it is a large number, probably at least a million. This can make for a very large table. It may not sound large by computer standards, but realize that routers must perform a lookup in this table to forward every packet, and routers at large ISPs may forward up to millions of packets per second. Specialized hardware and fast memory are needed to process packets at these rates, not a general-purpose computer.

In addition, routing algorithms require each router to exchange information about the addresses it can reach with other routers. The larger the tables, the more information needs to be communicated and processed. The processing grows at least linearly with the table size. Greater communication increases the likelihood that some parts will get lost, at least temporarily, possibly leading to routing instabilities.

The routing table problem could have been solved by going to a deeper hierarchy, like the telephone network. For example, having each IP address contain a country, state/province, city, network, and host field might work. Then, each router would only need to know how to get to each country, the states or provinces in its own country, the cities in its state or province, and the networks in its city. Unfortunately, this solution would require considerably more than 32 bits for IP addresses and would use addresses inefficiently (and Liechtenstein would have as many bits in its addresses as the United States).

Fortunately, there is something we can do to reduce routing table sizes. We can apply the same insight as subnetting: routers at different locations can know about a given IP address as belonging to prefixes of different sizes. However, instead of splitting an address block into subnets, here we combine multiple small

prefixes into a single larger prefix. This process is called **route aggregation**. The resulting larger prefix is sometimes called a **supernet**, to contrast with subnets as the division of blocks of addresses.

With aggregation, IP addresses are contained in prefixes of varying sizes. The same IP address that one router treats as part of a /22 (a block containing 2^{10} addresses) may be treated by another router as part of a larger /20 (which contains 2^{12} addresses). It is up to each router to have the corresponding prefix information. This design works with subnetting and is called **CIDR** (**Classless Inter-Domain Routing**), which is pronounced ''cider,'' as in the drink. The most recent version of it is specified in RFC 4632 (Fuller and Li, 2006). The name highlights the contrast with addresses that encode hierarchy with classes, which we will describe shortly.

To make CIDR easier to understand, let us consider an example in which a block of 8192 IP addresses is available starting at 194.24.0.0. Suppose that Cambridge University needs 2048 addresses and is assigned the addresses 194.24.0.0 through 194.24.7.255, along with mask 255.255.248.0. This is a /21 prefix. Next, Oxford University asks for 4096 addresses. Since a block of 4096 addresses must lie on a 4096-byte boundary, Oxford cannot be given addresses starting at 194.24.8.0. Instead, it gets 194.24.16.0 through 194.24.31.255, along with subnet mask 255.255.240.0. Finally, the University of Edinburgh asks for 1024 addresses and is assigned addresses 194.24.8.0 through 194.24.11.255 and mask 255.255.252.0. These assignments are summarized in Fig. 5-50.

University	First address	Last address	How many	Prefix
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12.0/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

Figure 5-50. A set of IP address assignments.

All of the routers in the default-free zone are now told about the IP addresses in the three networks. Routers close to the universities may need to send on a different outgoing line for each of the prefixes, so they need an entry for each of the prefixes in their routing tables. An example is the router in London in Fig. 5-51.

Now let us look at these three universities from the point of view of a distant router in New York. All of the IP addresses in the three prefixes should be sent from New York (or the U.S. in general) to London. The routing process in London notices this and combines the three prefixes into a single aggregate entry for the prefix 194.24.0.0/19 that it passes to the New York router. This prefix contains 8K addresses and covers the three universities and the otherwise unallocated 1024 addresses. By using aggregation, three prefixes have been reduced to one, reducing

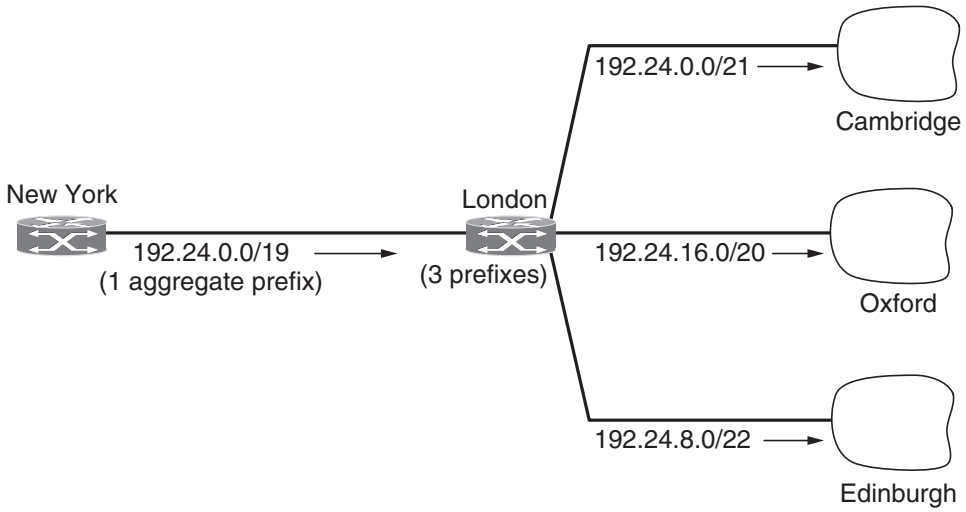


Figure 5-51. Aggregation of IP prefixes.

the prefixes that the New York router must be told about and the routing table entries in the New York router.

When aggregation is turned on, it is an automatic process. It depends on which prefixes are located where in the Internet not on the actions of an administrator assigning addresses to networks. Aggregation is heavily used throughout the Internet and can reduce the size of router tables to around 200,000 prefixes.

As a further twist, prefixes are allowed to overlap. The rule is that packets are sent in the direction of the most specific route, or the **longest matching prefix** that has the fewest IP addresses. Longest matching prefix routing provides a useful degree of flexibility, as seen in the behavior of the router at New York in Fig. 5-52. This router still uses a single aggregate prefix to send traffic for the three universities to London. However, the previously available block of addresses within this prefix has now been allocated to a network in San Francisco. One possibility is for the New York router to keep four prefixes, sending packets for three of them to London and packets for the fourth to San Francisco. Instead, longest matching prefix routing can handle this forwarding with the two prefixes that are shown. One overall prefix is used to direct traffic for the entire block to London. One more specific prefix is also used to direct a portion of the larger prefix to San Francisco. With the longest matching prefix rule, IP addresses within the San Francisco network will be sent on the outgoing line to San Francisco, and all other IP addresses in the larger prefix will be sent to London.

Conceptually, CIDR works as follows. When a packet comes in, the routing table is scanned to determine if the destination lies within the prefix. It is possible that multiple entries with different prefix lengths will match, in which case the entry with the longest prefix is used. Thus, if there is a match for a /20 mask and a /24 mask, the /24 entry is used to look up the outgoing line for the packet. However, this process would be tedious if the table were really scanned entry by entry.

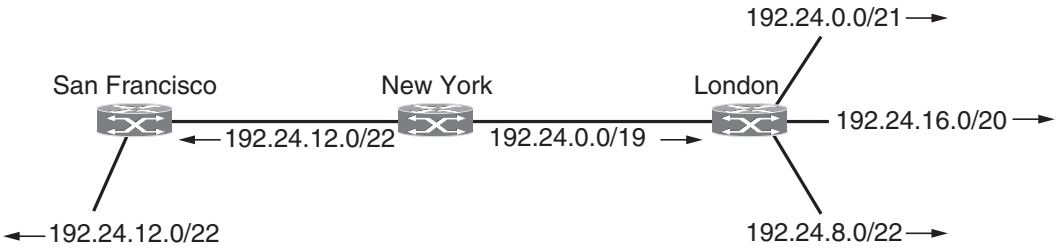


Figure 5-52. Longest matching prefix routing at the New York router.

Instead, complex algorithms have been devised to speed up the address matching process (Ruiz-Sanchez et al., 2001). Commercial routers use custom VLSI chips with these algorithms embedded in hardware.

Classful and Special Addressing

To help you better appreciate why CIDR is so useful, we will briefly relate the design that predated it. Before 1993, IP addresses were divided into the five categories listed in Fig. 5-53. This allocation has come to be called **classful addressing**.

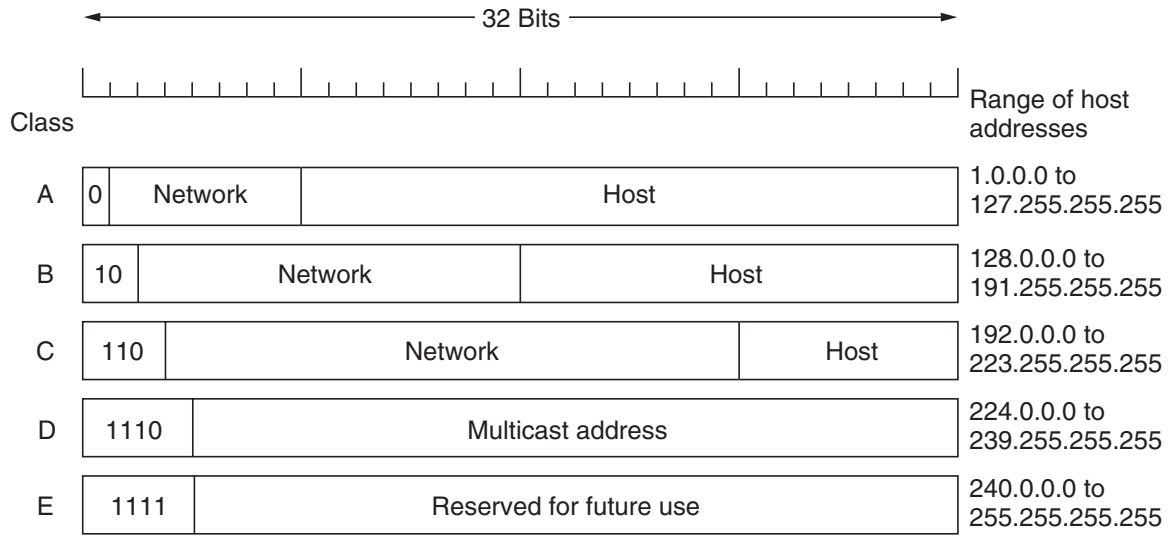


Figure 5-53. IP address formats.

The class A, B, and C formats allow for up to 128 networks with 16 million hosts each, 16,384 networks with up to 65,536 hosts each, and 2 million networks (e.g., LANs) with up to 256 hosts each (although a few of these are special). Also supported is multicast (the class D format), in which a datagram is directed to multiple hosts. Addresses beginning with 1111 are reserved for use in the future. They would be valuable to use now given the depletion of the IPv4 address space.

Unfortunately, many hosts will not accept these addresses as valid because they have been off-limits for so long and it is hard to teach old hosts new tricks.

This is a hierarchical design, but unlike CIDR the sizes of the address blocks are fixed. Over 2 billion addresses exist, but organizing the address space by classes wastes millions of them. In particular, the real villain is the class B network. For most organizations, a class A network, with 16 million addresses, is too big, and a class C network, with 256 addresses is too small. A class B network, with 65,536, is just right. In Internet folklore, this situation is known as the **three bears problem** [as in *Goldilocks and the Three Bears* (Southey, 1848)].

In reality, though, a class B address is far too large for most organizations. Studies have shown that more than half of all class B networks have fewer than 50 hosts. A class C network would have done the job, but no doubt every organization that asked for a class B address thought that one day it would outgrow the 8-bit host field. In retrospect, it might have been better to have had class C networks use 10 bits instead of 8 for the host number, allowing 1022 hosts per network. Had this been the case, most organizations would probably have settled for a class C network, and there would have been half a million of them (versus only 16,384 class B networks).

It is hard to fault the Internet's designers for not having provided more (and smaller) class B addresses. At the time the decision was made to create the three classes, the Internet was a research network connecting the major research universities in the U.S. (plus a very small number of companies and military sites doing networking research). No one then perceived the Internet becoming a mass-market communication system rivaling the telephone network. At the time, someone no doubt said: "The U.S. has about 2000 colleges and universities. Even if all of them connect to the Internet and many universities in other countries join, too, we are never going to hit 16,000, since there are not that many universities in the whole world. Furthermore, having the host number be an integral number of bytes speeds up packet processing" (which was then done entirely in software). Perhaps some day people will look back and fault the folks who designed the telephone number scheme and say: "What idiots. Why didn't they include the planet number in the phone number?" But at the time, it did not seem necessary.

To handle these problems, subnets were introduced to flexibly assign blocks of addresses within an organization. Later, CIDR was added to reduce the size of the global routing table. Today, the bits that indicate whether an IP address belongs to class A, B, or C network are no longer used, though references to these classes in the literature are still common.

To see how dropping the classes made forwarding more complicated, consider how simple it was in the old classful system. When a packet arrived at a router, a copy of the IP address was shifted right 28 bits to yield a 4-bit class number. A 16-way branch then sorted packets into A, B, C (and D and E) classes, with eight of the cases for class A, four of the cases for class B, and two of the cases for class C. The code for each class then masked off the 8-, 16-, or 24-bit network

number and right aligned it in a 32-bit word. The network number was then looked up in the A, B, or C table, usually by indexing for A and B networks and hashing for C networks. Once the entry was found, the outgoing line could be looked up and the packet forwarded. This is much simpler than the longest matching prefix operation, which can no longer use a simple table lookup because an IP address may have any length prefix.

Class D addresses continue to be used in the Internet for multicast. Actually, it might be more accurate to say that they are starting to be used for multicast, since Internet multicast has not been widely deployed in the past.

There are also several other addresses that have special meanings, as shown in Fig. 5-54. The IP address 0.0.0.0, the lowest address, is used by hosts when they are being booted. It means "this network" or "this host." IP addresses with 0 as the network number refer to the current network. These addresses allow machines to refer to their own network without knowing its number (but they have to know the network mask to know how many 0s to include). The address consisting of all 1s, or 255.255.255.255—the highest address—is used to mean all hosts on the indicated network. It allows broadcasting on the local network, typically a LAN. The addresses with a proper network number and all 1s in the host field allow machines to send broadcast packets to distant LANs anywhere in the Internet. However, many network administrators disable this feature as it is mostly a security hazard. Finally, all addresses of the form 127.*xx.yy.zz* are reserved for loopback testing. Packets sent to that address are not put out onto the wire; they are processed locally and treated as incoming packets. This allows packets to be sent to the host without the sender knowing its number, which is useful for testing.

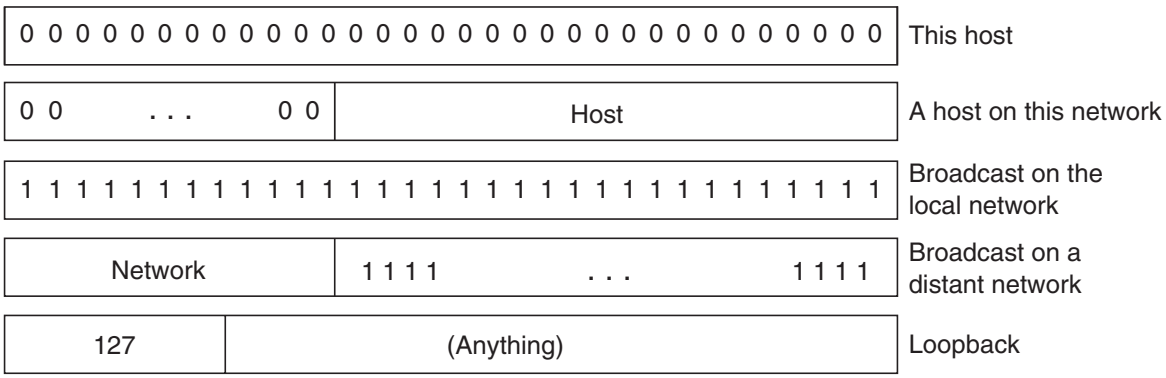


Figure 5-54. Special IP addresses.

NAT—Network Address Translation

IP addresses are scarce. An ISP might have a /16 address, giving it 65,534 usable host numbers. If it has more customers than that, it has a problem.

This scarcity has led to techniques to use IP addresses sparingly. One approach is to dynamically assign an IP address to a computer when it is on and using the network, and to take the IP address back when the host becomes inactive. The IP address can then be assigned to another computer that becomes active. In this way, a single /16 address can handle up to 65,534 active users.

This strategy works well in some cases, for example, for dialup networking and mobile and other computers that may be temporarily absent or powered off. However, it does not work very well for business customers. Many PCs in businesses are expected to be on continuously. Some are employee machines, backed up at night, and some are servers that may have to serve a remote request at a moment's notice. These businesses have an access line that always provides connectivity to the rest of the Internet.

Increasingly, this situation also applies to home users subscribing to ADSL or Internet over cable, since there is no connection charge (just a monthly flat rate charge). Many of these users have two or more computers at home, often one for each family member, and they all want to be online all the time. The solution is to connect all the computers into a home network via a LAN and put a (wireless) router on it. The router then connects to the ISP. From the ISP's point of view, the family is now the same as a small business with a handful of computers. Welcome to Jones, Inc. With the techniques we have seen so far, each computer must have its own IP address all day long. For an ISP with many thousands of customers, particularly business customers and families that are just like small businesses, the demand for IP addresses can quickly exceed the block that is available.

The problem of running out of IP addresses is not a theoretical one that might occur at some point in the distant future. It is happening right here and right now. The long-term solution is for the whole Internet to migrate to IPv6, which has 128-bit addresses. This transition is slowly occurring, but it will be years before the process is complete. To get by in the meantime, a quick fix was needed. The quick fix that is widely used today came in the form of **NAT** (**Network Address Translation**), which is described in RFC 3022 and which we will summarize below. For additional information, see Dutcher (2001).

The basic idea behind NAT is for the ISP to assign each home or business a single IP address (or at most, a small number of them) for Internet traffic. *Within* the customer network, every computer gets a unique IP address, which is used for routing intramural traffic. However, just before a packet exits the customer network and goes to the ISP, an address translation from the unique internal IP address to the shared public IP address takes place. This translation makes use of three ranges of IP addresses that have been declared as private. Networks may use them internally as they wish. The only rule is that no packets containing these addresses may appear on the Internet itself. The three reserved ranges are:

10.0.0.0	– 10.255.255.255/8	(16,777,216 hosts)
172.16.0.0	– 172.31.255.255/12	(1,048,576 hosts)
192.168.0.0	– 192.168.255.255/16	(65,536 hosts)

The first range provides for 16,777,216 addresses (except for all 0s and all 1s, as usual) and is the usual choice, even if the network is not large.

The operation of NAT is shown in Fig. 5-55. Within the customer premises, every machine has a unique address of the form 10.*x.y.z*. However, before a packet leaves the customer premises, it passes through a **NAT box** that converts the internal IP source address, 10.0.0.1 in the figure, to the customer's true IP address, 198.60.42.12 in this example. The NAT box is often combined in a single device with a firewall, which provides security by carefully controlling what goes into the customer network and what comes out of it. We will study firewalls in Chap. 8. It is also possible to integrate the NAT box into a router or ADSL modem.

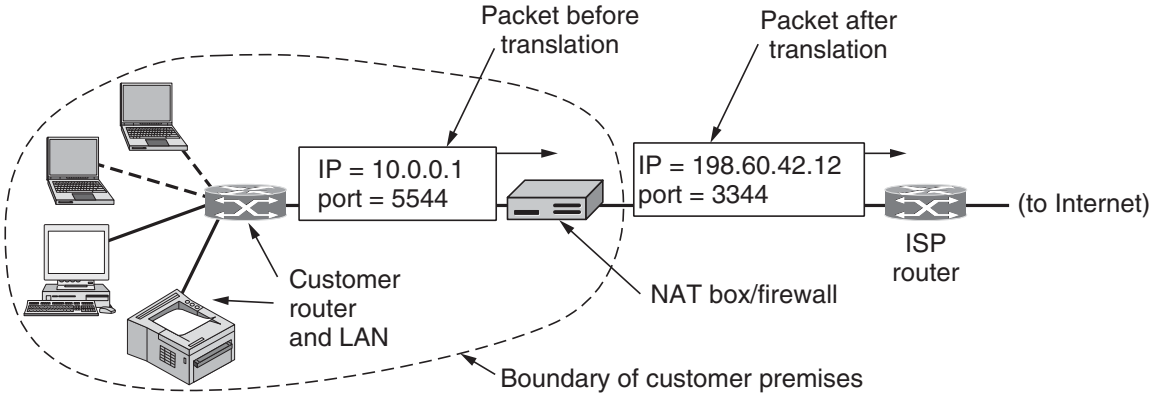


Figure 5-55. Placement and operation of a NAT box.

So far, we have glossed over one tiny but crucial detail: when the reply comes back (e.g., from a Web server), it is naturally addressed to 198.60.42.12, so how does the NAT box know which internal address to replace it with? Herein lies the problem with NAT. If there were a spare field in the IP header, that field could be used to keep track of who the real sender was, but only 1 bit is still unused. In principle, a new option could be created to hold the true source address, but doing so would require changing the IP code on all the machines on the entire Internet to handle the new option. This is not a promising alternative for a quick fix.

What actually happens is as follows. The NAT designers observed that most IP packets carry either TCP or UDP payloads. When we study TCP and UDP in Chap. 6, we will see that both of these have headers containing a source port and a destination port. Below we will just discuss TCP ports, but exactly the same story holds for UDP ports. The ports are 16-bit integers that indicate where the TCP connection begins and ends. These ports provide the field needed to make NAT work.

When a process wants to establish a TCP connection with a remote process, it attaches itself to an unused TCP port on its own machine. This is called the **source port** and tells the TCP code where to send incoming packets belonging to this connection. The process also supplies a **destination port** to tell who to give

the packets to on the remote side. Ports 0–1023 are reserved for well-known services. For example, port 80 is the port used by Web servers, so remote clients can locate them. Each outgoing TCP message contains both a source port and a destination port. Together, these ports serve to identify the processes using the connection on both ends.

An analogy may make the use of ports clearer. Imagine a company with a single main telephone number. When people call the main number, they reach an operator who asks which extension they want and then puts them through to that extension. The main number is analogous to the customer's IP address and the extensions on both ends are analogous to the ports. Ports are effectively an extra 16 bits of addressing that identify which process gets which incoming packet.

Using the *Source port* field, we can solve our mapping problem. Whenever an outgoing packet enters the NAT box, the 10.*x.y.z* source address is replaced by the customer's true IP address. In addition, the TCP *Source port* field is replaced by an index into the NAT box's 65,536-entry translation table. This table entry contains the original IP address and the original source port. Finally, both the IP and TCP header checksums are recomputed and inserted into the packet. It is necessary to replace the *Source port* because connections from machines 10.0.0.1 and 10.0.0.2 may both happen to use port 5000, for example, so the *Source port* alone is not enough to identify the sending process.

When a packet arrives at the NAT box from the ISP, the *Source port* in the TCP header is extracted and used as an index into the NAT box's mapping table. From the entry located, the internal IP address and original TCP *Source port* are extracted and inserted into the packet. Then, both the IP and TCP checksums are recomputed and inserted into the packet. The packet is then passed to the customer router for normal delivery using the 10.*x.y.z* address.

Although this scheme sort of solves the problem, networking purists in the IP community have a tendency to regard it as an abomination-on-the-face-of-the-earth. Briefly summarized, here are some of the objections. First, NAT violates the architectural model of IP, which states that every IP address uniquely identifies a single machine worldwide. The whole software structure of the Internet is built on this fact. With NAT, thousands of machines may (and do) use address 10.0.0.1.

Second, NAT breaks the end-to-end connectivity model of the Internet, which says that any host can send a packet to any other host at any time. Since the mapping in the NAT box is set up by outgoing packets, incoming packets cannot be accepted until after outgoing ones. In practice, this means that a home user with NAT can make TCP/IP connections to a remote Web server, but a remote user cannot make connections to a game server on the home network. Special configuration or **NAT traversal** techniques are needed to support this kind of situation.

Third, NAT changes the Internet from a connectionless network to a peculiar kind of connection-oriented network. The problem is that the NAT box must maintain information (i.e., the mapping) for each connection passing through it.

Having the network maintain connection state is a property of connection-oriented networks, not connectionless ones. If the NAT box crashes and its mapping table is lost, all its TCP connections are destroyed. In the absence of NAT, a router can crash and restart with no long-term effect on TCP connections. The sending process just times out within a few seconds and retransmits all unacknowledged packets. With NAT, the Internet becomes as vulnerable as a circuit-switched network.

Fourth, NAT violates the most fundamental rule of protocol layering: layer k may not make any assumptions about what layer $k + 1$ has put into the payload field. This basic principle is there to keep the layers independent. If TCP is later upgraded to TCP-2, with a different header layout (e.g., 32-bit ports), NAT will fail. The whole idea of layered protocols is to ensure that changes in one layer do not require changes in other layers. NAT destroys this independence.

Fifth, processes on the Internet are not required to use TCP or UDP. If a user on machine *A* decides to use some new transport protocol to talk to a user on machine *B* (for example, for a multimedia application), introduction of a NAT box will cause the application to fail because the NAT box will not be able to locate the TCP *Source port* correctly.

A sixth and related problem is that some applications use multiple TCP/IP connections or UDP ports in prescribed ways. For example, **FTP**, the standard **File Transfer Protocol**, inserts IP addresses in the body of packet for the receiver to extract and use. Since NAT knows nothing about these arrangements, it cannot rewrite the IP addresses or otherwise account for them. This lack of understanding means that FTP and other applications such as the H.323 Internet telephony protocol (which we will study in Chap. 7) will fail in the presence of NAT unless special precautions are taken. It is often possible to patch NAT for these cases, but having to patch the code in the NAT box every time a new application comes along is not a good idea.

Finally, since the TCP *Source port* field is 16 bits, at most 65,536 machines can be mapped onto an IP address. Actually, the number is slightly less because the first 4096 ports are reserved for special uses. However, if multiple IP addresses are available, each one can handle up to 61,440 machines.

A view of these and other problems with NAT is given in RFC 2993. Despite the issues, NAT is widely used in practice, especially for home and small business networks, as the only expedient technique to deal with the IP address shortage. It has become wrapped up with firewalls and privacy because it blocks unsolicited incoming packets by default. For this reason, it is unlikely to go away even when IPv6 is widely deployed.

5.6.3 IP Version 6

IP has been in heavy use for decades. It has worked extremely well, as demonstrated by the exponential growth of the Internet. Unfortunately, IP has become a victim of its own popularity: it is close to running out of addresses. Even

with CIDR and NAT using addresses more sparingly, the last IPv4 addresses are expected to be assigned by ICANN before the end of 2012. This looming disaster was recognized almost two decades ago, and it sparked a great deal of discussion and controversy within the Internet community about what to do about it.

In this section, we will describe both the problem and several proposed solutions. The only long-term solution is to move to larger addresses. **IPv6** (**IP version 6**) is a replacement design that does just that. It uses 128-bit addresses; a shortage of these addresses is not likely any time in the foreseeable future. However, IPv6 has proved very difficult to deploy. It is a different network layer protocol that does not really interwork with IPv4, despite many similarities. Also, companies and users are not really sure why they should want IPv6 in any case. The result is that IPv6 is deployed and used on only a tiny fraction of the Internet (estimates are 1%) despite having been an Internet Standard since 1998. The next several years will be an interesting time, as the few remaining IPv4 addresses are allocated. Will people start to auction off their IPv4 addresses on eBay? Will a black market in them spring up? Who knows.

In addition to the address problems, other issues loom in the background. In its early years, the Internet was largely used by universities, high-tech industries, and the U.S. Government (especially the Dept. of Defense). With the explosion of interest in the Internet starting in the mid-1990s, it began to be used by a different group of people, often with different requirements. For one thing, numerous people with smart phones use it to keep in contact with their home bases. For another, with the impending convergence of the computer, communication, and entertainment industries, it may not be that long before every telephone and television set in the world is an Internet node, resulting in a billion machines being used for audio and video on demand. Under these circumstances, it became apparent that IP had to evolve and become more flexible.

Seeing these problems on the horizon, in 1990 IETF started work on a new version of IP, one that would never run out of addresses, would solve a variety of other problems, and be more flexible and efficient as well. Its major goals were:

1. Support billions of hosts, even with inefficient address allocation.
2. Reduce the size of the routing tables.
3. Simplify the protocol, to allow routers to process packets faster.
4. Provide better security (authentication and privacy).
5. Pay more attention to the type of service, particularly for real-time data.
6. Aid multicasting by allowing scopes to be specified.
7. Make it possible for a host to roam without changing its address.
8. Allow the protocol to evolve in the future.
9. Permit the old and new protocols to coexist for years.

The design of IPv6 presented a major opportunity to improve all of the features in IPv4 that fall short of what is now wanted. To develop a protocol that met all these requirements, IETF issued a call for proposals and discussion in RFC 1550. Twenty-one responses were initially received. By December 1992, seven serious proposals were on the table. They ranged from making minor patches to IP, to throwing it out altogether and replacing it with a completely different protocol.

One proposal was to run TCP over CLNP, the network layer protocol designed for OSI. With its 160-bit addresses, CLNP would have provided enough address space forever as it could give every molecule of water in the oceans enough addresses (roughly 2^5) to set up a small network. This choice would also have unified two major network layer protocols. However, many people felt that this would have been an admission that something in the OSI world was actually done right, a statement considered Politically Incorrect in Internet circles. CLNP was patterned closely on IP, so the two are not really that different. In fact, the protocol ultimately chosen differs from IP far more than CLNP does. Another strike against CLNP was its poor support for service types, something required to transmit multimedia efficiently.

Three of the better proposals were published in *IEEE Network* (Deering, 1993; Francis, 1993; and Katz and Ford, 1993). After much discussion, revision, and jockeying for position, a modified combined version of the Deering and Francis proposals, by now called **SIPP** (**Simple Internet Protocol Plus**) was selected and given the designation **IPv6.**

IPv6 meets IETF's goals fairly well. It maintains the good features of IP, discards or deemphasizes the bad ones, and adds new ones where needed. In general, IPv6 is not compatible with IPv4, but it is compatible with the other auxiliary Internet protocols, including TCP, UDP, ICMP, IGMP, OSPF, BGP, and DNS, with small modifications being required to deal with longer addresses. The main features of IPv6 are discussed below. More information about it can be found in RFCs 2460 through 2466.

First and foremost, IPv6 has longer addresses than IPv4. They are 128 bits long, which solves the problem that IPv6 set out to solve: providing an effectively unlimited supply of Internet addresses. We will have more to say about addresses shortly.

The second major improvement of IPv6 is the simplification of the header. It contains only seven fields (versus 13 in IPv4). This change allows routers to process packets faster and thus improves throughput and delay. We will discuss the header shortly, too.

The third major improvement is better support for options. This change was essential with the new header because fields that previously were required are now optional (because they are not used so often). In addition, the way options are represented is different, making it simple for routers to skip over options not intended for them. This feature speeds up packet processing time.

A fourth area in which IPv6 represents a big advance is in security. IETF had its fill of newspaper stories about precocious 12-year-olds using their personal computers to break into banks and military bases all over the Internet. There was a strong feeling that something had to be done to improve security. Authentication and privacy are key features of the new IP. These were later retrofitted to IPv4, however, so in the area of security the differences are not so great any more.

Finally, more attention has been paid to quality of service. Various half-hearted efforts to improve QoS have been made in the past, but now, with the growth of multimedia on the Internet, the sense of urgency is greater.

The Main IPv6 Header

The IPv6 header is shown in Fig. 5-56. The *Version* field is always 6 for IPv6 (and 4 for IPv4). During the transition period from IPv4, which has already taken more than a decade, routers will be able to examine this field to tell what kind of packet they have. As an aside, making this test wastes a few instructions in the critical path, given that the data link header usually indicates the network protocol for demultiplexing, so some routers may skip the check. For example, the Ethernet *Type* field has different values to indicate an IPv4 or an IPv6 payload. The discussions between the "Do it right" and "Make it fast" camps will no doubt be lengthy and vigorous.

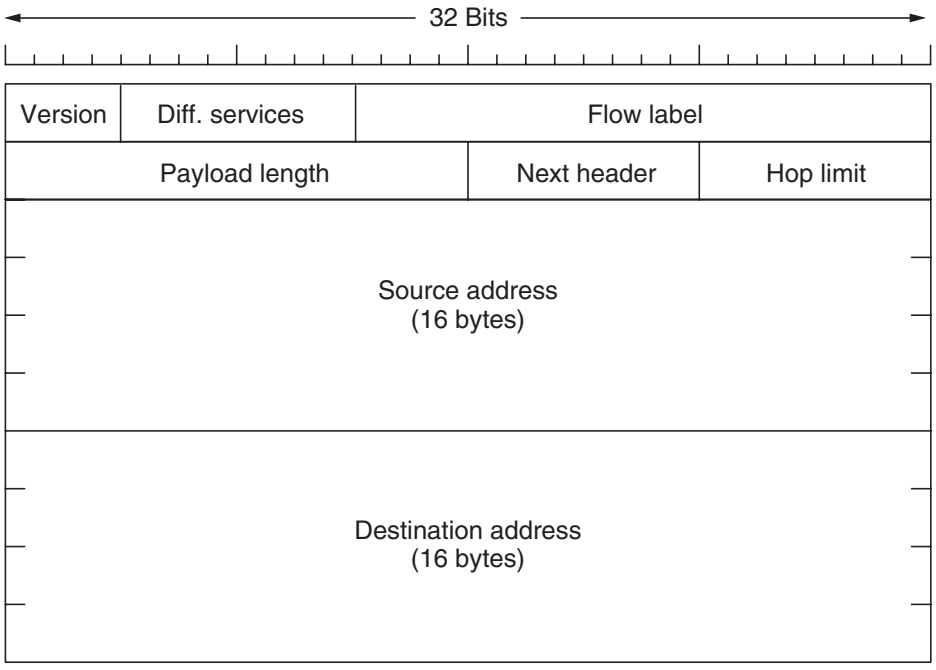


Figure 5-56. The IPv6 fixed header (required).

The *Differentiated services* field (originally called *Traffic class*) is used to distinguish the class of service for packets with different real-time delivery

requirements. It is used with the differentiated service architecture for quality of service in the same manner as the field of the same name in the IPv4 packet. Also, the low-order 2 bits are used to signal explicit congestion indications, again in the same way as with IPv4.

The *Flow label* field provides a way for a source and destination to mark groups of packets that have the same requirements and should be treated in the same way by the network, forming a pseudoconnection. For example, a stream of packets from one process on a certain source host to a process on a specific destination host might have stringent delay requirements and thus need reserved bandwidth. The flow can be set up in advance and given an identifier. When a packet with a nonzero *Flow label* shows up, all the routers can look it up in internal tables to see what kind of special treatment it requires. In effect, flows are an attempt to have it both ways: the flexibility of a datagram network and the guarantees of a virtual-circuit network.

Each flow for quality of service purposes is designated by the source address, destination address, and flow number. This design means that up to 2^{20} flows may be active at the same time between a given pair of IP addresses. It also means that even if two flows coming from different hosts but with the same flow label pass through the same router, the router will be able to tell them apart using the source and destination addresses. It is expected that flow labels will be chosen randomly, rather than assigned sequentially starting at 1, so routers are expected to hash them.

The *Payload length* field tells how many bytes follow the 40-byte header of Fig. 5-56. The name was changed from the IPv4 *Total length* field because the meaning was changed slightly: the 40 header bytes are no longer counted as part of the length (as they used to be). This change means the payload can now be 65,535 bytes instead of a mere 65,515 bytes.

The *Next header* field lets the cat out of the bag. The reason the header could be simplified is that there can be additional (optional) extension headers. This field tells which of the (currently) six extension headers, if any, follow this one. If this header is the last IP header, the *Next header* field tells which transport protocol handler (e.g., TCP, UDP) to pass the packet to.

The *Hop limit* field is used to keep packets from living forever. It is, in practice, the same as the *Time to live* field in IPv4, namely, a field that is decremented on each hop. In theory, in IPv4 it was a time in seconds, but no router used it that way, so the name was changed to reflect the way it is actually used.

Next come the *Source address* and *Destination address* fields. Deering's original proposal, SIP, used 8-byte addresses, but during the review process many people felt that with 8-byte addresses IPv6 would run out of addresses within a few decades, whereas with 16-byte addresses it would never run out. Other people argued that 16 bytes was overkill, whereas still others favored using 20-byte addresses to be compatible with the OSI datagram protocol. Still another faction wanted variable-sized addresses. After much debate and more than a few words

unprintable in an academic textbook, it was decided that fixed-length 16-byte addresses were the best compromise.

A new notation has been devised for writing 16-byte addresses. They are written as eight groups of four hexadecimal digits with colons between the groups, like this:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Since many addresses will have many zeros inside them, three optimizations have been authorized. First, leading zeros within a group can be omitted, so 0123 can be written as 123. Second, one or more groups of 16 zero bits can be replaced by a pair of colons. Thus, the above address now becomes

8000::123:4567:89AB:CDEF

Finally, IPv4 addresses can be written as a pair of colons and an old dotted decimal number, for example:

::192.31.20.46

Perhaps it is unnecessary to be so explicit about it, but there are a lot of 16-byte addresses. Specifically, there are 2^{128} of them, which is approximately 3×10^{38}. If the entire earth, land and water, were covered with computers, IPv6 would allow 7×10^{23} IP addresses per square meter. Students of chemistry will notice that this number is larger than Avogadro's number. While it was not the intention to give every molecule on the surface of the earth its own IP address, we are not that far off.

In practice, the address space will not be used efficiently, just as the telephone number address space is not (the area code for Manhattan, 212, is nearly full, but that for Wyoming, 307, is nearly empty). In RFC 3194, Durand and Huitema calculated that, using the allocation of telephone numbers as a guide, even in the most pessimistic scenario there will still be well over 1000 IP addresses per square meter of the entire earth's surface (land and water). In any likely scenario, there will be trillions of them per square meter. In short, it seems unlikely that we will run out in the foreseeable future.

It is instructive to compare the IPv4 header (Fig. 5-46) with the IPv6 header (Fig. 5-56) to see what has been left out in IPv6. The *IHL* field is gone because the IPv6 header has a fixed length. The *Protocol* field was taken out because the *Next header* field tells what follows the last IP header (e.g., a UDP or TCP segment).

All the fields relating to fragmentation were removed because IPv6 takes a different approach to fragmentation. To start with, all IPv6-conformant hosts are expected to dynamically determine the packet size to use. They do this using the path MTU discovery procedure we described in Sec. 5.5.5. In brief, when a host sends an IPv6 packet that is too large, instead of fragmenting it, the router that is unable to forward it drops the packet and sends an error message back to the

sending host. This message tells the host to break up all future packets to that destination. Having the host send packets that are the right size in the first place is ultimately much more efficient than having the routers fragment them on the fly. Also, the minimum-size packet that routers must be able to forward has been raised from 576 to 1280 bytes to allow 1024 bytes of data and many headers.

Finally, the *Checksum* field is gone because calculating it greatly reduces performance. With the reliable networks now used, combined with the fact that the data link layer and transport layers normally have their own checksums, the value of yet another checksum was deemed not worth the performance price it extracted. Removing all these features has resulted in a lean and mean network layer protocol. Thus, the goal of IPv6—a fast, yet flexible, protocol with plenty of address space—is met by this design.

Extension Headers

Some of the missing IPv4 fields are occasionally still needed, so IPv6 introduces the concept of (optional) **extension headers**. These headers can be supplied to provide extra information, but encoded in an efficient way. Six kinds of extension headers are defined at present, as listed in Fig. 5-57. Each one is optional, but if more than one is present they must appear directly after the fixed header, and preferably in the order listed.

Extension header	Description
Hop-by-hop options	Miscellaneous information for routers
Destination options	Additional information for the destination
Routing	Loose list of routers to visit
Fragmentation	Management of datagram fragments
Authentication	Verification of the sender's identity
Encrypted security payload	Information about the encrypted contents

Figure 5-57. IPv6 extension headers.

Some of the headers have a fixed format; others contain a variable number of variable-length options. For these, each item is encoded as a (*Type, Length, Value*) tuple. The *Type* is a 1-byte field telling which option this is. The *Type* values have been chosen so that the first 2 bits tell routers that do not know how to process the option what to do. The choices are: skip the option; discard the packet; discard the packet and send back an ICMP packet; and discard the packet but do not send ICMP packets for multicast addresses (to prevent one bad multicast packet from generating millions of ICMP reports).

The *Length* is also a 1-byte field. It tells how long the value is (0 to 255 bytes). The *Value* is any information required, up to 255 bytes.

The hop-by-hop header is used for information that all routers along the path must examine. So far, one option has been defined: support of datagrams exceeding 64 KB. The format of this header is shown in Fig. 5-58. When it is used, the *Payload length* field in the fixed header is set to 0.

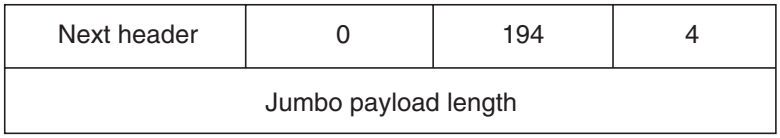


Figure 5-58. The hop-by-hop extension header for large datagrams (jumbograms).

As with all extension headers, this one starts with a byte telling what kind of header comes next. This byte is followed by one telling how long the hop-by-hop header is in bytes, excluding the first 8 bytes, which are mandatory. All extensions begin this way.

The next 2 bytes indicate that this option defines the datagram size (code 194) and that the size is a 4-byte number. The last 4 bytes give the size of the datagram. Sizes less than 65,536 bytes are not permitted and will result in the first router discarding the packet and sending back an ICMP error message. Datagrams using this header extension are called **jumbograms**. The use of jumbograms is important for supercomputer applications that must transfer gigabytes of data efficiently across the Internet.

The destination options header is intended for fields that need only be interpreted at the destination host. In the initial version of IPv6, the only options defined are null options for padding this header out to a multiple of 8 bytes, so initially it will not be used. It was included to make sure that new routing and host software can handle it, in case someone thinks of a destination option some day.

The routing header lists one or more routers that must be visited on the way to the destination. It is very similar to the IPv4 loose source routing in that all addresses listed must be visited in order, but other routers not listed may be visited in between. The format of the routing header is shown in Fig. 5-59.

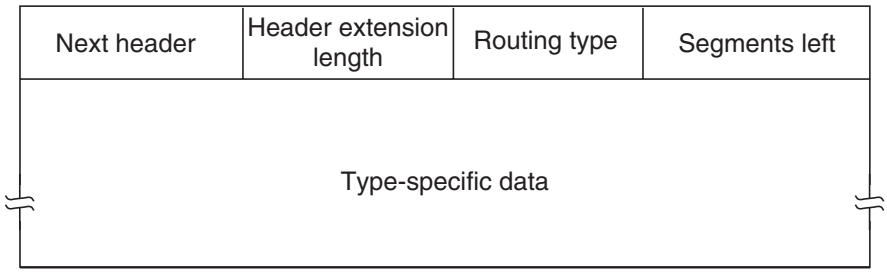


Figure 5-59. The extension header for routing.

The first 4 bytes of the routing extension header contain four 1-byte integers. The *Next header* and *Header extension length* fields were described above. The *Routing type* field gives the format of the rest of the header. Type 0 says that a reserved 32-bit word follows the first word, followed by some number of IPv6 addresses. Other types may be invented in the future, as needed. Finally, the *Segments left* field keeps track of how many of the addresses in the list have not yet been visited. It is decremented every time one is visited. When it hits 0, the packet is on its own with no more guidance about what route to follow. Usually, at this point it is so close to the destination that the best route is obvious.

The fragment header deals with fragmentation similarly to the way IPv4 does. The header holds the datagram identifier, fragment number, and a bit telling whether more fragments will follow. In IPv6, unlike in IPv4, only the source host can fragment a packet. Routers along the way may not do this. This change is a major philosophical break with the original IP, but in keeping with current practice for IPv4. Plus, it simplifies the routers' work and makes routing go faster. As mentioned above, if a router is confronted with a packet that is too big, it discards the packet and sends an ICMP error packet back to the source. This information allows the source host to fragment the packet into smaller pieces using this header and try again.

The authentication header provides a mechanism by which the receiver of a packet can be sure of who sent it. The encrypted security payload makes it possible to encrypt the contents of a packet so that only the intended recipient can read it. These headers use the cryptographic techniques that we will describe in Chap. 8 to accomplish their missions.

Controversies

Given the open design process and the strongly held opinions of many of the people involved, it should come as no surprise that many choices made for IPv6 were highly controversial, to say the least. We will summarize a few of these briefly below. For all the gory details, see the RFCs.

We have already mentioned the argument about the address length. The result was a compromise: 16-byte fixed-length addresses.

Another fight developed over the length of the *Hop limit* field. One camp felt strongly that limiting the maximum number of hops to 255 (implicit in using an 8-bit field) was a gross mistake. After all, paths of 32 hops are common now, and 10 years from now much longer paths may be common. These people argued that using a huge address size was farsighted but using a tiny hop count was shortsighted. In their view, the greatest sin a computer scientist can commit is to provide too few bits somewhere.

The response was that arguments could be made to increase every field, leading to a bloated header. Also, the function of the *Hop limit* field is to keep packets from wandering around for too long a time and 65,535 hops is far, far too long.

Finally, as the Internet grows, more and more long-distance links will be built, making it possible to get from any country to any other country in half a dozen hops at most. If it takes more than 125 hops to get from the source and the destination to their respective international gateways, something is wrong with the national backbones. The 8-bitters won this one.

Another hot potato was the maximum packet size. The supercomputer community wanted packets in excess of 64 KB. When a supercomputer gets started transferring, it really means business and does not want to be interrupted every 64 KB. The argument against large packets is that if a 1-MB packet hits a 1.5-Mbps T1 line, that packet will tie the line up for over 5 seconds, producing a very noticeable delay for interactive users sharing the line. A compromise was reached here: normal packets are limited to 64 KB, but the hop-by-hop extension header can be used to permit jumbograms.

A third hot topic was removing the IPv4 checksum. Some people likened this move to removing the brakes from a car. Doing so makes the car lighter so it can go faster, but if an unexpected event happens, you have a problem.

The argument against checksums was that any application that really cares about data integrity has to have a transport layer checksum anyway, so having another one in IP (in addition to the data link layer checksum) is overkill. Furthermore, experience showed that computing the IP checksum was a major expense in IPv4. The anticheckSum camp won this one, and IPv6 does not have a checksum.

Mobile hosts were also a point of contention. If a portable computer flies halfway around the world, can it continue operating there with the same IPv6 address, or does it have to use a scheme with home agents? Some people wanted to build explicit support for mobile hosts into IPv6. That effort failed when no consensus could be found for any specific proposal.

Probably the biggest battle was about security. Everyone agreed it was essential. The war was about where to put it and how. First where. The argument for putting it in the network layer is that it then becomes a standard service that all applications can use without any advance planning. The argument against it is that really secure applications generally want nothing less than end-to-end encryption, where the source application does the encryption and the destination application undoes it. With anything less, the user is at the mercy of potentially buggy network layer implementations over which he has no control. The response to this argument is that these applications can just refrain from using the IP security features and do the job themselves. The rejoinder to that is that the people who do not trust the network to do it right do not want to pay the price of slow, bulky IP implementations that have this capability, even if it is disabled.

Another aspect of where to put security relates to the fact that many (but not all) countries have very stringent export laws concerning cryptography. Some, notably France and Iraq, also restrict its use domestically, so that people cannot have secrets from the government. As a result, any IP implementation that used a

cryptographic system strong enough to be of much value could not be exported from the United States (and many other countries) to customers worldwide. Having to maintain two sets of software, one for domestic use and one for export, is something most computer vendors vigorously oppose.

One point on which there was no controversy is that no one expects the IPv4 Internet to be turned off on a Sunday evening and come back up as an IPv6 Internet Monday morning. Instead, isolated ''islands'' of IPv6 will be converted, initially communicating via tunnels, as we showed in Sec. 5.5.3. As the IPv6 islands grow, they will merge into bigger islands. Eventually, all the islands will merge, and the Internet will be fully converted.

At least, that was the plan. Deployment has proved the Achilles heel of IPv6. It remains little used, even though all major operating systems fully support it. Most deployments are new situations in which a network operator—for example, a mobile phone operator— needs a large number of IP addresses. Many strategies have been defined to help ease the transition. Among them are ways to automatically configure the tunnels that carry IPv6 over the IPv4 Internet, and ways for hosts to automatically find the tunnel endpoints. Dual-stack hosts have an IPv4 and an IPv6 implementation so that they can select which protocol to use depending on the destination of the packet. These strategies will streamline the substantial deployment that seems inevitable when IPv4 addresses are exhausted. For more information about IPv6, see Davies (2008).

5.6.4 Internet Control Protocols

In addition to IP, which is used for data transfer, the Internet has several companion control protocols that are used in the network layer. They include ICMP, ARP, and DHCP. In this section, we will look at each of these in turn, describing the versions that correspond to IPv4 because they are the protocols that are in common use. ICMP and DHCP have similar versions for IPv6; the equivalent of ARP is called NDP (Neighbor Discovery Protocol) for IPv6.

IMCP—The Internet Control Message Protocol

The operation of the Internet is monitored closely by the routers. When something unexpected occurs during packet processing at a router, the event is reported to the sender by the **ICMP** (**Internet Control Message Protocol**). ICMP is also used to test the Internet. About a dozen types of ICMP messages are defined. Each ICMP message type is carried encapsulated in an IP packet. The most important ones are listed in Fig. 5-60.

The DESTINATION UNREACHABLE message is used when the router cannot locate the destination or when a packet with the *DF* bit cannot be delivered because a ''small-packet'' network stands in the way.

A first step in this direction is to have each CA periodically issue a **CRL** (**Certificate Revocation List**) giving the serial numbers of all certificates that it has revoked. Since certificates contain expiry times, the CRL need only contain the serial numbers of certificates that have not yet expired. Once its expiry time has passed, a certificate is automatically invalid, so no distinction is needed between those that just timed out and those that were actually revoked. In both cases, they cannot be used any more.

Unfortunately, introducing CRLs means that a user who is about to use a certificate must now acquire the CRL to see if the certificate has been revoked. If it has been, it should not be used. However, even if the certificate is not on the list, it might have been revoked just after the list was published. Thus, the only way to really be sure is to ask the CA. And on the next use of the same certificate, the CA has to be asked again, since the certificate might have been revoked a few seconds ago.

Another complication is that a revoked certificate could conceivably be reinstated, for example, if it was revoked for nonpayment of some fee that has since been paid. Having to deal with revocation (and possibly reinstatement) eliminates one of the best properties of certificates, namely, that they can be used without having to contact a CA.

Where should CRLs be stored? A good place would be the same place the certificates themselves are stored. One strategy is for the CA to actively push out CRLs periodically and have the directories process them by simply removing the revoked certificates. If directories are not used for storing certificates, the CRLs can be cached at various places around the network. Since a CRL is itself a signed document, if it is tampered with, that tampering can be easily detected.

If certificates have long lifetimes, the CRLs will be long, too. For example, if credit cards are valid for 5 years, the number of revocations outstanding will be much longer than if new cards are issued every 3 months. A standard way to deal with long CRLs is to issue a master list infrequently, but issue updates to it more often. Doing this reduces the bandwidth needed for distributing the CRLs.

8.6 COMMUNICATION SECURITY

We have now finished our study of the tools of the trade. Most of the important techniques and protocols have been covered. The rest of the chapter is about how these techniques are applied in practice to provide network security, plus some thoughts about the social aspects of security at the end of the chapter.

In the following four sections, we will look at communication security, that is, how to get the bits secretly and without modification from source to destination and how to keep unwanted bits outside the door. These are by no means the only security issues in networking, but they are certainly among the most important ones, making this a good place to start our study.

8.6.1 IPsec

IETF has known for years that security was lacking in the Internet. Adding it was not easy because a war broke out about where to put it. Most security experts believe that to be really secure, encryption and integrity checks have to be end to end (i.e., in the application layer). That is, the source process encrypts and/or integrity protects the data and sends them to the destination process where they are decrypted and/or verified. Any tampering done in between these two processes, including within either operating system, can then be detected. The trouble with this approach is that it requires changing all the applications to make them security aware. In this view, the next best approach is putting encryption in the transport layer or in a new layer between the application layer and the transport layer, making it still end to end but not requiring applications to be changed.

The opposite view is that users do not understand security and will not be capable of using it correctly and nobody wants to modify existing programs in any way, so the network layer should authenticate and/or encrypt packets without the users being involved. After years of pitched battles, this view won enough support that a network layer security standard was defined. In part, the argument was that having network layer encryption does not prevent security-aware users from doing it right and it does help security-unaware users to some extent.

The result of this war was a design called **IPsec** (**IP security**), which is described in RFCs 2401, 2402, and 2406, among others. Not all users want encryption (because it is computationally expensive). Rather than make it optional, it was decided to require encryption all the time but permit the use of a null algorithm. The null algorithm is described and praised for its simplicity, ease of implementation, and great speed in RFC 2410.

The complete IPsec design is a framework for multiple services, algorithms, and granularities. The reason for multiple services is that not everyone wants to pay the price for having all the services all the time, so the services are available a la carte. The major services are secrecy, data integrity, and protection from replay attacks (where the intruder replays a conversation). All of these are based on symmetric-key cryptography because high performance is crucial.

The reason for having multiple algorithms is that an algorithm that is now thought to be secure may be broken in the future. By making IPsec algorithm-independent, the framework can survive even if some particular algorithm is later broken.

The reason for having multiple granularities is to make it possible to protect a single TCP connection, all traffic between a pair of hosts, or all traffic between a pair of secure routers, among other possibilities.

One slightly surprising aspect of IPsec is that even though it is in the IP layer, it is connection oriented. Actually, that is not so surprising because to have any security, a key must be established and used for some period of time—in essence, a kind of connection by a different name. Also, connections amortize the setup

costs over many packets. A "connection" in the context of IPsec is called an **SA** (**Security Association**). An SA is a simplex connection between two endpoints and has a security identifier associated with it. If secure traffic is needed in both directions, two security associations are required. Security identifiers are carried in packets traveling on these secure connections and are used to look up keys and other relevant information when a secure packet arrives.

Technically, IPsec has two principal parts. The first part describes two new headers that can be added to packets to carry the security identifier, integrity control data, and other information. The other part, **ISAKMP** (**Internet Security Association and Key Management Protocol**), deals with establishing keys. ISAKMP is a framework. The main protocol for carrying out the work is **IKE** (**Internet Key Exchange**). Version 2 of IKE as described in RFC 4306 should be used, as the earlier version was deeply flawed, as pointed out by Perlman and Kaufman (2000).

IPsec can be used in either of two modes. In **transport mode**, the IPsec header is inserted just after the IP header. The *Protocol* field in the IP header is changed to indicate that an IPsec header follows the normal IP header (before the TCP header). The IPsec header contains security information, primarily the SA identifier, a new sequence number, and possibly an integrity check of the payload.

In **tunnel mode**, the entire IP packet, header and all, is encapsulated in the body of a new IP packet with a completely new IP header. Tunnel mode is useful when the tunnel ends at a location other than the final destination. In some cases, the end of the tunnel is a security gateway machine, for example, a company firewall. This is commonly the case for a VPN (Virtual Private Network). In this mode, the security gateway encapsulates and decapsulates packets as they pass through it. By terminating the tunnel at this secure machine, the machines on the company LAN do not have to be aware of IPsec. Only the security gateway has to know about it.

Tunnel mode is also useful when a bundle of TCP connections is aggregated and handled as one encrypted stream because it prevents an intruder from seeing who is sending how many packets to whom. Sometimes just knowing how much traffic is going where is valuable information. For example, if during a military crisis, the amount of traffic flowing between the Pentagon and the White House were to drop sharply, but the amount of traffic between the Pentagon and some military installation deep in the Colorado Rocky Mountains were to increase by the same amount, an intruder might be able to deduce some useful information from these data. Studying the flow patterns of packets, even if they are encrypted, is called **traffic analysis**. Tunnel mode provides a way to foil it to some extent. The disadvantage of tunnel mode is that it adds an extra IP header, thus increasing packet size substantially. In contrast, transport mode does not affect packet size as much.

The first new header is **AH** (**Authentication Header**). It provides integrity checking and antireplay security, but not secrecy (i.e., no data encryption). The

use of AH in transport mode is illustrated in Fig. 8-27. In IPv4, it is interposed between the IP header (including any options) and the TCP header. In IPv6, it is just another extension header and is treated as such. In fact, the format is close to that of a standard IPv6 extension header. The payload may have to be padded out to some particular length for the authentication algorithm, as shown.

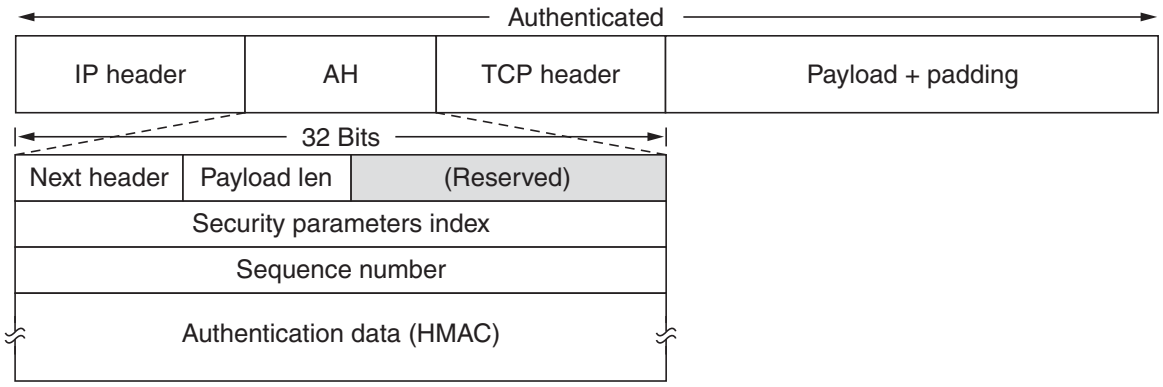


Figure 8-27. The IPsec authentication header in transport mode for IPv4.

Let us now examine the AH header. The *Next header* field is used to store the value that the IP *Protocol* field had before it was replaced with 51 to indicate that an AH header follows. In most cases, the code for TCP (6) will go here. The *Payload length* is the number of 32-bit words in the AH header minus 2.

The *Security parameters index* is the connection identifier. It is inserted by the sender to indicate a particular record in the receiver's database. This record contains the shared key used on this connection and other information about the connection. If this protocol had been invented by ITU rather than IETF, this field would have been called *Virtual circuit number*.

The *Sequence number* field is used to number all the packets sent on an SA. Every packet gets a unique number, even retransmissions. In other words, the retransmission of a packet gets a different number here than the original (even though its TCP sequence number is the same). The purpose of this field is to detect replay attacks. These sequence numbers may not wrap around. If all 2^{32} are exhausted, a new SA must be established to continue communication.

Finally, we come to *Authentication data*, which is a variable-length field that contains the payload's digital signature. When the SA is established, the two sides negotiate which signature algorithm they are going to use. Normally, public-key cryptography is not used here because packets must be processed extremely rapidly and all known public-key algorithms are too slow. Since IPsec is based on symmetric-key cryptography and the sender and receiver negotiate a shared key before setting up an SA, the shared key is used in the signature computation. One simple way is to compute the hash over the packet plus the shared key. The shared key is not transmitted, of course. A scheme like this is called an **HMAC**

(**Hashed Message Authentication Code**). It is much faster to compute than first running SHA-1 and then running RSA on the result.

The AH header does not allow encryption of the data, so it is mostly useful when integrity checking is needed but secrecy is not needed. One noteworthy feature of AH is that the integrity check covers some of the fields in the IP header, namely, those that do not change as the packet moves from router to router. The *Time to live* field changes on each hop, for example, so it cannot be included in the integrity check. However, the IP source address is included in the check, making it impossible for an intruder to falsify the origin of a packet.

The alternative IPsec header is **ESP** (**Encapsulating Security Payload**). Its use for both transport mode and tunnel mode is shown in Fig. 8-28.

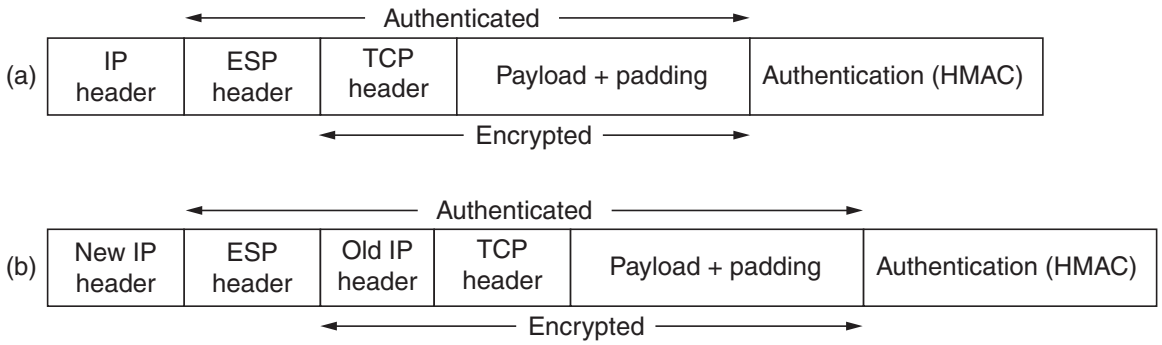


Figure 8-28. (a) ESP in transport mode. (b) ESP in tunnel mode.

The ESP header consists of two 32-bit words. They are the *Security parameters index* and *Sequence number* fields that we saw in AH. A third word that generally follows them (but is technically not part of the header) is the *Initialization vector* used for the data encryption, unless null encryption is used, in which case it is omitted.

ESP also provides for HMAC integrity checks, as does AH, but rather than being included in the header, they come after the payload, as shown in Fig. 8-28. Putting the HMAC at the end has an advantage in a hardware implementation: the HMAC can be calculated as the bits are going out over the network interface and appended to the end. This is why Ethernet and other LANs have their CRCs in a trailer, rather than in a header. With AH, the packet has to be buffered and the signature computed before the packet can be sent, potentially reducing the number of packets/sec that can be sent.

Given that ESP can do everything AH can do and more and is more efficient to boot, the question arises: why bother having AH at all? The answer is mostly historical. Originally, AH handled only integrity and ESP handled only secrecy. Later, integrity was added to ESP, but the people who designed AH did not want to let it die after all that work. Their only real argument is that AH checks part of the IP header, which ESP does not, but other than that it is really a weak argument. Another weak argument is that a product supporting AH but not ESP might

have less trouble getting an export license because it cannot do encryption. AH is likely to be phased out in the future.

8.6.2 Firewalls

The ability to connect any computer, anywhere, to any other computer, anywhere, is a mixed blessing. For individuals at home, wandering around the Internet is lots of fun. For corporate security managers, it is a nightmare. Most companies have large amounts of confidential information online—trade secrets, product development plans, marketing strategies, financial analyses, etc. Disclosure of this information to a competitor could have dire consequences.

In addition to the danger of information leaking out, there is also a danger of information leaking in. In particular, viruses, worms, and other digital pests can breach security, destroy valuable data, and waste large amounts of administrators' time trying to clean up the mess they leave. Often they are imported by careless employees who want to play some nifty new game.

Consequently, mechanisms are needed to keep "good" bits in and "bad" bits out. One method is to use IPsec. This approach protects data in transit between secure sites. However, IPsec does nothing to keep digital pests and intruders from getting onto the company LAN. To see how to accomplish this goal, we need to look at firewalls.

Firewalls are just a modern adaptation of that old medieval security standby: digging a deep moat around your castle. This design forced everyone entering or leaving the castle to pass over a single drawbridge, where they could be inspected by the I/O police. With networks, the same trick is possible: a company can have many LANs connected in arbitrary ways, but all traffic to or from the company is forced through an electronic drawbridge (firewall), as shown in Fig. 8-29. No other route exists.

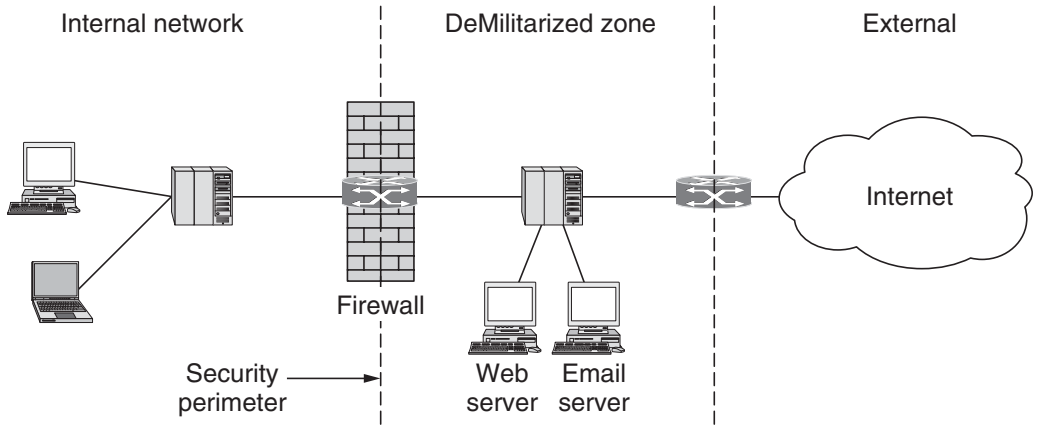


Figure 8-29. A firewall protecting an internal network.

The firewall acts as a **packet filter**. It inspects each and every incoming and outgoing packet. Packets meeting some criterion described in rules formulated by the network administrator are forwarded normally. Those that fail the test are uncermoniously dropped.

The filtering criterion is typically given as rules or tables that list sources and destinations that are acceptable, sources and destinations that are blocked, and default rules about what to do with packets coming from or going to other machines. In the common case of a TCP/IP setting, a source or destination might consist of an IP address and a port. Ports indicate which service is desired. For example, TCP port 25 is for mail, and TCP port 80 is for HTTP. Some ports can simply be blocked. For example, a company could block incoming packets for all IP addresses combined with TCP port 79. It was once popular for the Finger service to look up people's email addresses but is little used today.

Other ports are not so easily blocked. The difficulty is that network administrators want security but cannot cut off communication with the outside world. That arrangement would be much simpler and better for security, but there would be no end to user complaints about it. This is where arrangements such as the **DMZ** (**DeMilitarized Zone**) shown in Fig. 8-29 come in handy. The DMZ is the part of the company network that lies outside of the security perimeter. Anything goes here. By placing a machine such as a Web server in the DMZ, computers on the Internet can contact it to browse the company Web site. Now the firewall can be configured to block incoming TCP traffic to port 80 so that computers on the Internet cannot use this port to attack computers on the internal network. To allow the Web server to be managed, the firewall can have a rule to permit connections between internal machines and the Web server.

Firewalls have become much more sophisticated over time in an arms race with attackers. Originally, firewalls applied a rule set independently for each packet, but it proved difficult to write rules that allowed useful functionality but blocked all unwanted traffic. **Stateful firewalls** map packets to connections and use TCP/IP header fields to keep track of connections. This allows for rules that, for example, allow an external Web server to send packets to an internal host, but only if the internal host first establishes a connection with the external Web server. Such a rule is not possible with stateless designs that must either pass or drop all packets from the external Web server.

Another level of sophistication up from stateful processing is for the firewall to implement **application-level gateways**. This processing involves the firewall looking inside packets, beyond even the TCP header, to see what the application is doing. With this capability, it is possible to distinguish HTTP traffic used for Web browsing from HTTP traffic used for peer-to-peer file sharing. Administrators can write rules to spare the company from peer-to-peer file sharing but allow Web browsing that is vital for business. For all of these methods, outgoing traffic can be inspected as well as incoming traffic, for example, to prevent sensitive documents from being emailed outside of the company.

As the above discussion should make clear, firewalls violate the standard layering of protocols. They are network layer devices, but they peek at the transport and applications layers to do their filtering. This makes them fragile. For instance, firewalls tend to rely on standard port numbering conventions to determine what kind of traffic is carried in a packet. Standard ports are often used, but not by all computers, and not by all applications either. Some peer-to-peer applications select ports dynamically to avoid being easily spotted (and blocked). Encryption with IPSEC or other schemes hides higher-layer information from the firewall. Finally, a firewall cannot readily talk to the computers that communicate through it to tell them what policies are being applied and why their connection is being dropped. It must simply pretend to be a broken wire. For all these reasons, networking purists consider firewalls to be a blemish on the architecture of the Internet. However, the Internet can be a dangerous place if you are a computer. Firewalls help with that problem, so they are likely to stay.

Even if the firewall is perfectly configured, plenty of security problems still exist. For example, if a firewall is configured to allow in packets from only specific networks (e.g., the company's other plants), an intruder outside the firewall can put in false source addresses to bypass this check. If an insider wants to ship out secret documents, he can encrypt them or even photograph them and ship the photos as JPEG files, which bypasses any email filters. And we have not even discussed the fact that, although three-quarters of all attacks come from outside the firewall, the attacks that come from inside the firewall, for example, from disgruntled employees, are typically the most damaging (Verizon, 2009).

A different problem with firewalls is that they provide a single perimeter of defense. If that defense is breached, all bets are off. For this reason, firewalls are often used in a layered defense. For example, a firewall may guard the entrance to the internal network and each computer may also run its own firewall. Readers who think that one security checkpoint is enough clearly have not made an international flight on a scheduled airline recently.

In addition, there is a whole other class of attacks that firewalls cannot deal with. The basic idea of a firewall is to prevent intruders from getting in and secret data from getting out. Unfortunately, there are people who have nothing better to do than try to bring certain sites down. They do this by sending legitimate packets at the target in great numbers until it collapses under the load. For example, to cripple a Web site, an intruder can send a TCP *SYN* packet to establish a connection. The site will then allocate a table slot for the connection and send a *SYN + ACK* packet in reply. If the intruder does not respond, the table slot will be tied up for a few seconds until it times out. If the intruder sends thousands of connection requests, all the table slots will fill up and no legitimate connections will be able to get through. Attacks in which the intruder's goal is to shut down the target rather than steal data are called **DoS** (**Denial of Service**) attacks. Usually, the request packets have false source addresses so the intruder cannot be traced easily. DoS attacks against major Web sites are common on the Internet.

An even worse variant is one in which the intruder has already broken into hundreds of computers elsewhere in the world, and then commands all of them to attack the same target at the same time. Not only does this approach increase the intruder's firepower, but it also reduces his chances of detection since the packets are coming from a large number of machines belonging to unsuspecting users. Such an attack is called a **DDoS** (**Distributed Denial of Service**) attack. This attack is difficult to defend against. Even if the attacked machine can quickly recognize a bogus request, it does take some time to process and discard the request, and if enough requests per second arrive, the CPU will spend all its time dealing with them.

8.6.3 Virtual Private Networks

Many companies have offices and plants scattered over many cities, sometimes over multiple countries. In the olden days, before public data networks, it was common for such companies to lease lines from the telephone company between some or all pairs of locations. Some companies still do this. A network built up from company computers and leased telephone lines is called a **private network**.

Private networks work fine and are very secure. If the only lines available are the leased lines, no traffic can leak out of company locations and intruders have to physically wiretap the lines to break in, which is not easy to do. The problem with private networks is that leasing a dedicated T1 line between two points costs thousands of dollars a month, and T3 lines are many times more expensive. When public data networks and later the Internet appeared, many companies wanted to move their data (and possibly voice) traffic to the public network, but without giving up the security of the private network.

This demand soon led to the invention of **VPN**s (**Virtual Private Networks**), which are overlay networks on top of public networks but with most of the properties of private networks. They are called ''virtual'' because they are merely an illusion, just as virtual circuits are not real circuits and virtual memory is not real memory.

One popular approach is to build VPNs directly over the Internet. A common design is to equip each office with a firewall and create tunnels through the Internet between all pairs of offices, as illustrated in Fig. 8-30(a). A further advantage of using the Internet for connectivity is that the tunnels can be set up on demand to include, for example, the computer of an employee who is at home or traveling as long as the person has an Internet connection. This flexibility is much greater then is provided with leased lines, yet from the perspective of the computers on the VPN, the topology looks just like the private network case, as shown in Fig. 8-30(b). When the system is brought up, each pair of firewalls has to negotiate the parameters of its SA, including the services, modes, algorithms, and keys. If IPsec is used for the tunneling, it is possible to aggregate all traffic between any

INDEX

Numbers

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

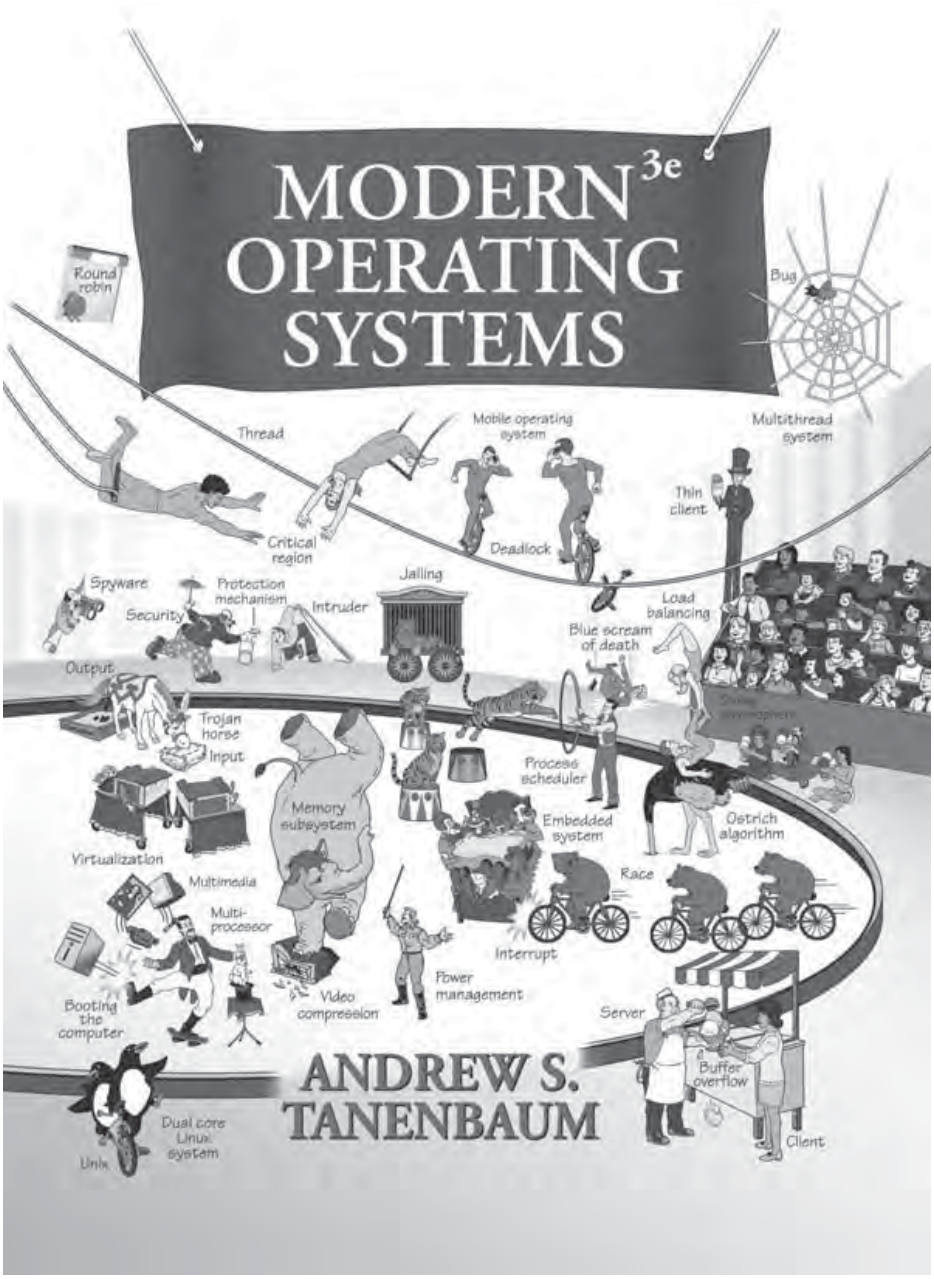
X

Z

Also by Andrew S. Tanenbaum

Modern Operating Systems, 3rd ed.

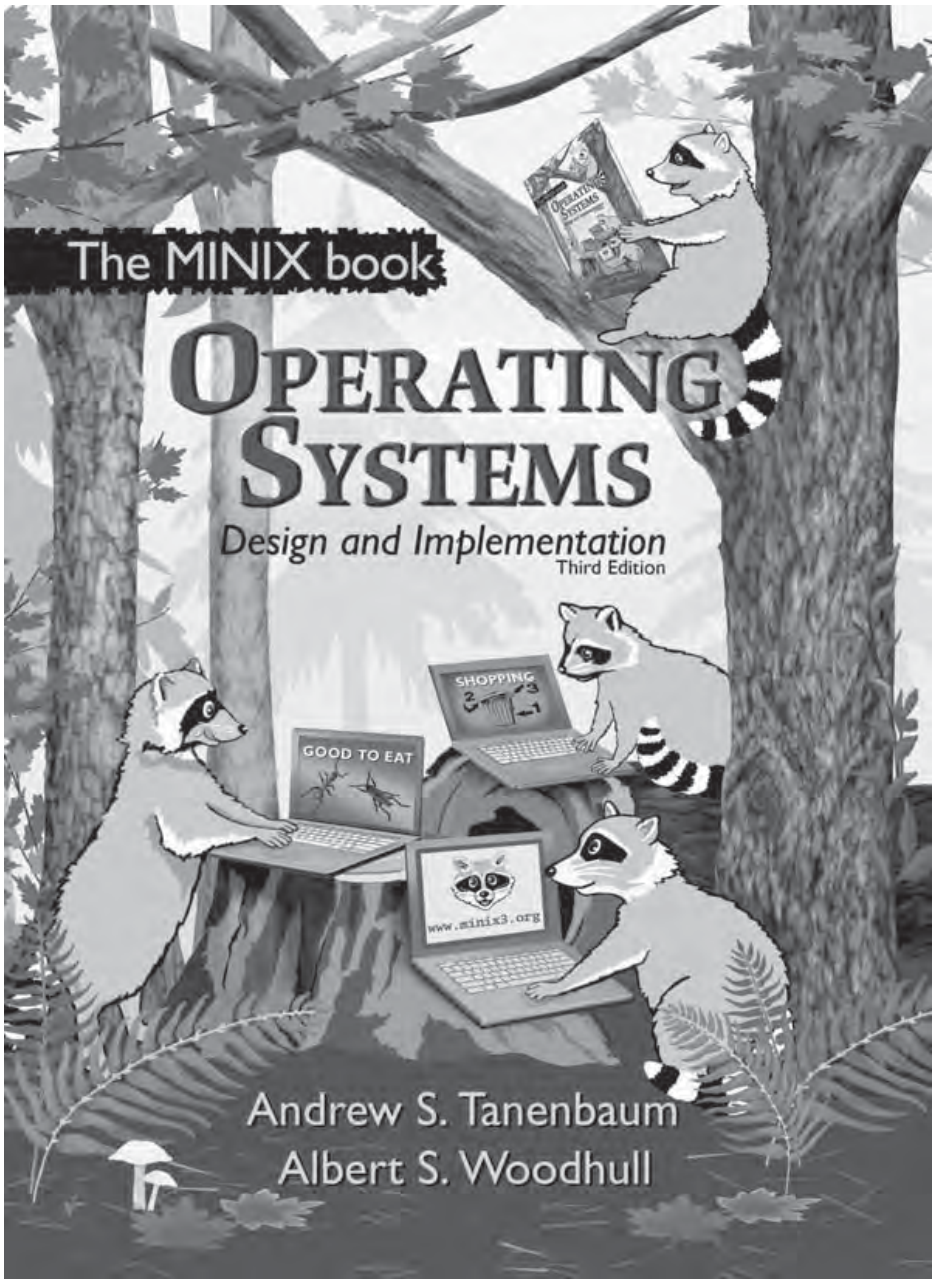
This worldwide best-seller incorporates the latest developments in operating systems. The book starts with chapters on the principles, including processes, memory management, file systems, I/O, and so on. Then it goes into three chapter-long case studies: Linux, Windows, and Symbian. Tanenbaum's experience as the designer of three operating systems (Amoeba, Globe, and MINIX) gives him a background few other authors can match, so the final chapter distills his long experience into advice for operating system designers.



Also by Andrew S. Tanenbaum and Albert S. Woodhull

Operating Systems: Design and Implementation, 3rd ed.

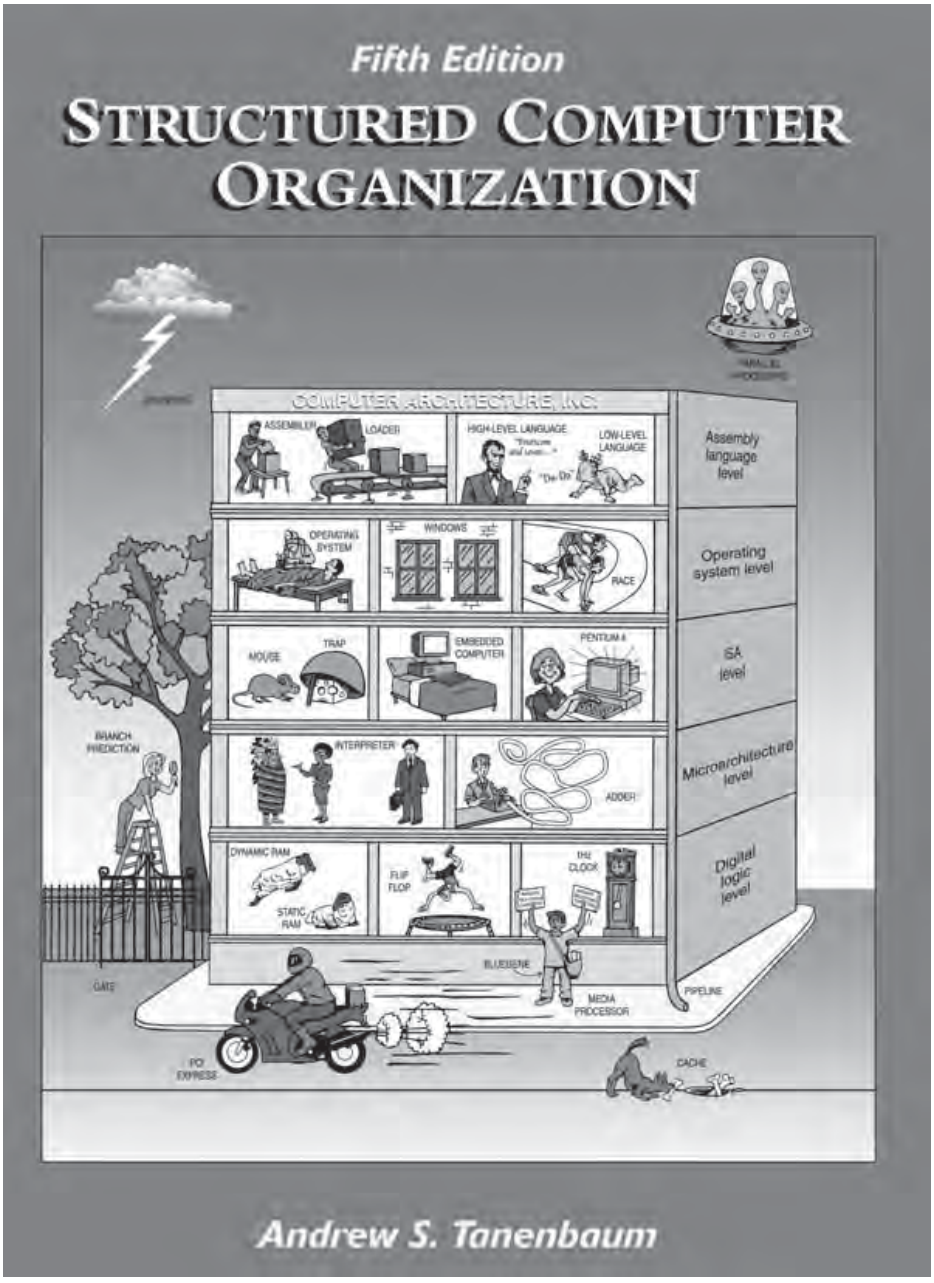
All other textbooks on operating systems are long on theory and short on practice. This one is different. In addition to the usual material on processes, memory management, file systems, I/O, and so on, it contains a CD-ROM with the source code (in C) of a small, but complete, POSIX-conformant operating system called MINIX 3 (see *www.minix3.org*). All the principles are illustrated by showing how they apply to MINIX 3. The reader can also compile, test, and experiment with MINIX 3, leading to in-depth knowledge of how an operating system really works.



Also by Andrew S. Tanenbaum

Structured Computer Organization, 5th ed.

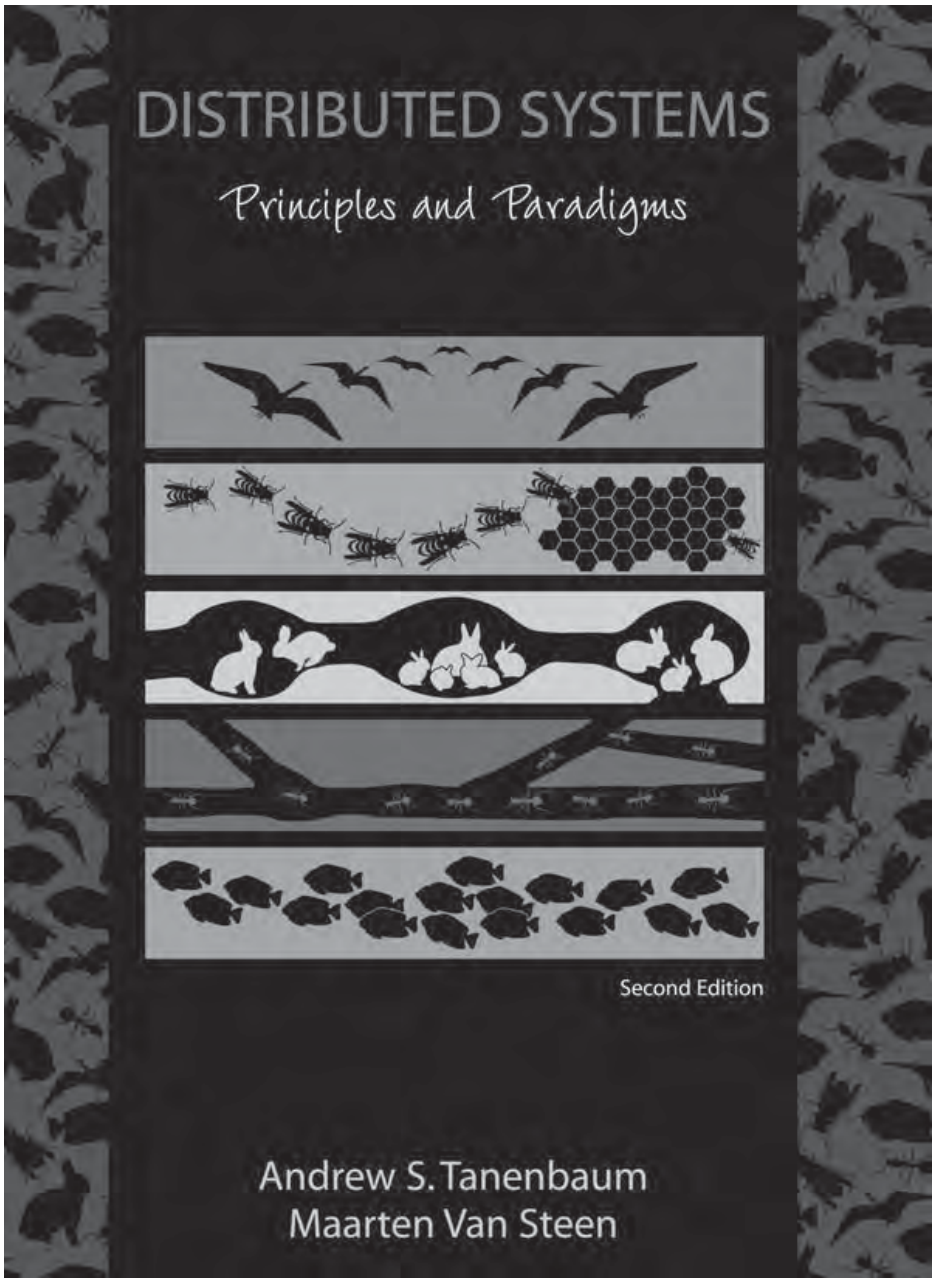
A computer can be structured as a hierarchy of levels, from the hardware up through the operating system. This book treats all of them, starting with how a transistor works and ending with operating system design. No previous experience with either hardware or software is needed to follow this book, however, as all the topics are self contained and explained in simple terms starting right at the beginning. The running examples used throughout the book range from the high-end UltraSPARC III, through the ever-popular x86 (Pentium) to the tiny Intel 8051 used in small embedded systems.



Also by Andrew S. Tanenbaum and Maarten van Steen

Distributed Systems: Principles and Paradigms, 2nd ed.

Distributed systems are becoming ever-more important in the world and this book explains their principles and illustrates them with numerous examples. Among the topics covered are architectures, processes, communication, naming, synchronization, consistency, fault tolerance, and security. Examples are taken from distributed object-based, file, Web-based, and coordination-based systems.



ABOUT THE AUTHORS

Andrew S. Tanenbaum has an S.B. degree from M.I.T. and a Ph.D. from the University of California at Berkeley. He is currently a Professor of Computer Science at the Vrije Universiteit where he has taught operating systems, networks, and related topics for over 30 years. His current research is on highly reliable operating systems although he has worked on compilers, distributed systems, security, and other topics over the years. These research projects have led to over 150 refereed papers in journals and conferences.

Prof. Tanenbaum has also (co)authored five books which have now appeared in 19 editions. The books have been translated into 21 languages, ranging from Basque to Thai and are used at universities all over the world. In all, there are 159 versions (language/edition combinations), which are listed at *www.cs.vu.nl/~ast/publications.*

Prof. Tanenbaum has also produced a considerable volume of software, including the Amsterdam Compiler Kit (a retargetable portable compiler), Amoeba (an early distributed system used on LANs), and Globe (a wide-area distributed system).

He is also the author of MINIX, a small UNIX clone initially intended for use in student programming labs. It was the direct inspiration for Linux and the platform on which Linux was initially developed. The current version of MINIX, called MINIX 3, is now focused on being an extremely reliable and secure operating system. Prof. Tanenbaum will consider his work done when no computer is equipped with a reset button and no living person has ever experienced a system crash. MINIX 3 is an on-going open-source project to which you are invited to contribute. Go to *www.minix3.org* to download a free copy and find out what is happening.

Tanenbaum is a Fellow of the ACM, a Fellow of the the IEEE, and a member of the Royal Netherlands Academy of Arts and Sciences. He has also won numerous scientific prizes, including:

- 2010 TAA McGuffey Award for Computer Science and Engineering books
- 2007 IEEE James H. Mulligan, Jr. Education Medal
- 2002 TAA Texty Award for Computer Science and Engineering books
- 1997 ACM/SIGCSE Award for Outstanding Contributions to Computer Science Education
- 1994 ACM Karl V. Karlstrom Outstanding Educator Award

His home page on the World Wide Web can be found at *http://www.cs.vu.nl/~ast/.*

David J. Wetherall is an Associate Professor of Computer Science and Engineering at the University of Washington in Seattle, and advisor to Intel Labs in Seattle. He hails from Australia, where he received his B.E. in electrical enginering from the University of Western Australia and his Ph.D. in computer science from M.I.T.

Prof. Wetherall has worked in the area of networking for the past two decades. His research is focused on network systems, especially wireless networks and mobile computing, the design of Internet protocols, and network measurement.

He received the ACM SIGCOMM Test-of-Time award for research that pioneered active networks, an architecture for rapidly introducing new network services. He received the IEEE William Bennett Prize for breakthroughs in Internet mapping. His research was recognized with an NSF CAREER award in 2002, and he became a Sloan Fellow in 2004.

As well as teaching networking, Prof. Wetherall participates in the networking research community. He has co-chaired the program committees of SIGCOMM, NSDI and MobiSys, and co-founded the ACM HotNets workshops. He has served on numerous program committees for networking conferences, and is an editor for ACM Computer Communication Review.

His home page on the World Wide Web can be found at *http://djw.cs.washington.edu*.