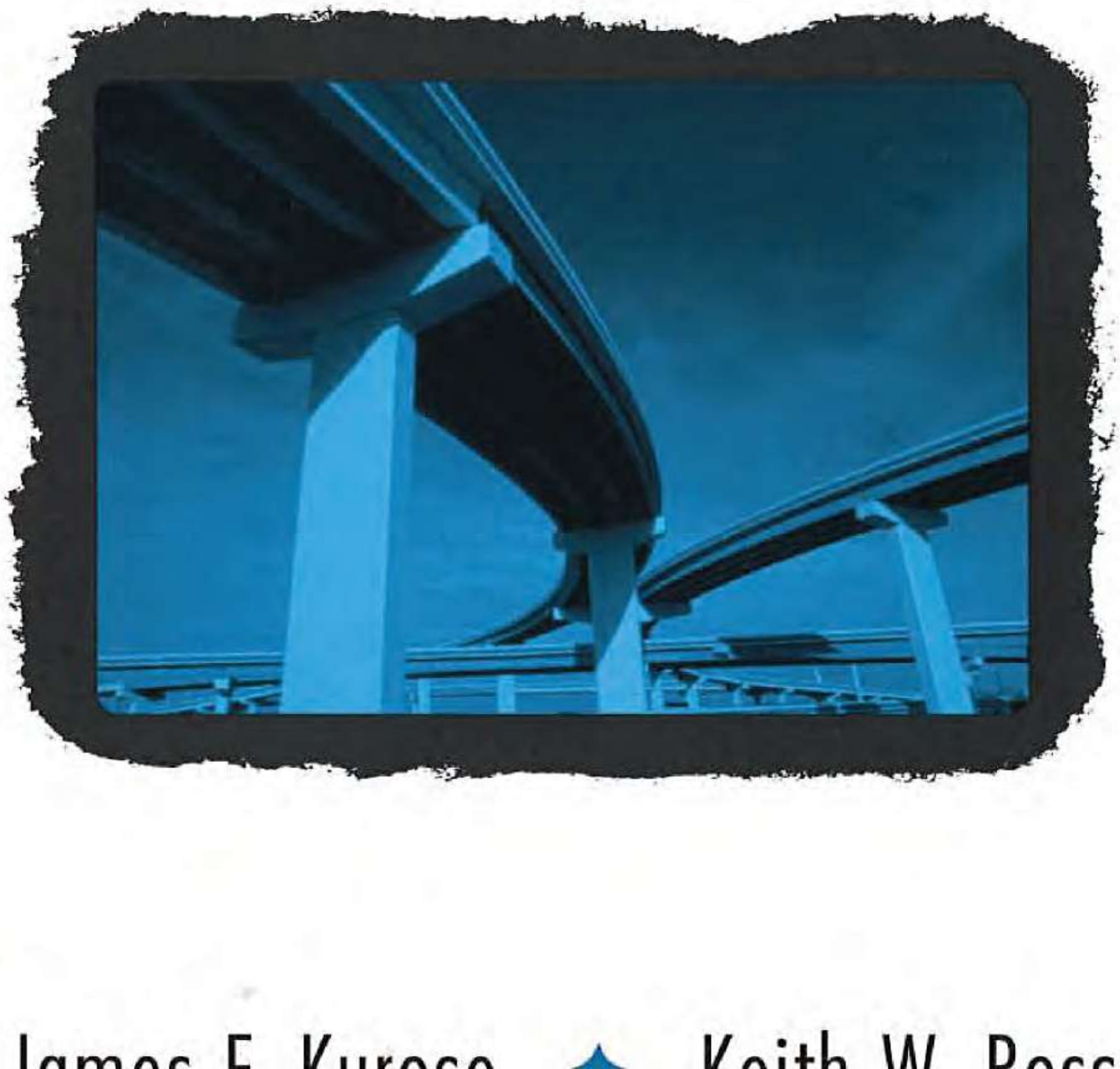
Computer Networking

A Top-Down Approach Featuring the Internet



James F. Kurose ✦ Keith W. Ross

Senior Acquisitions Editor	Susan Hartman
Assistant Editor	Lisa Kalner
Production Supervisor	Patty Mahtani
Art Editor	Helen Reebenacker
Executive Marketing Manager	Michael Hirsch
Composition	Pre-Press Company, Inc.
Technical Art	PD & PS
Copyeditor	Roberta Lewis
Proofreader	Holly McLean Aldis
Cover Design	Joyce Cosentino
Interior Design	Delgado Design
Design Manager	Regina Hagen
Cover Image	© 1999 PhotoDisc, Inc.

Access the latest information about Addison-Wesley titles from our World Wide Web site:
http://www.awl.com/cs

Library of Congress Cataloging-in-Publication Data

Ross, Keith W., 1956-

Computer networking: a top-down approach featuring the Internet / Keith W. Ross, James F. Kurose.

p. cm.

Includes bibliographic references and index

ISBN 0-201-47711-4

1. Internet (Computer network) I. Kurose, James F.

TK5105.875.I57 R689 2001

004.6—dc21 00-025295

2 3 4 5 6 7 8 9 10-MA-0201

Chapter 1

Computer Networks and the Internet

1.1 ✦ What Is the Internet?

In this book we use the public Internet, a specific computer network (and one which probably most readers have used), as our principle vehicle for discussing computer networking protocols. But what is the Internet? We would like to be able to give you a one-sentence definition of the Internet, a definition that you can take home and share with your family and friends. Alas, the Internet is very complex, both in terms of its hardware and software components, as well as in the services it provides.

1.1.1 A Nuts and Bolts Description

Instead of giving a one-sentence definition, let's try a more descriptive approach. There are a couple of ways to do this. One way is to describe the nuts and bolts of the Internet, that is, the basic hardware and software components that make up the Internet. Another way is to describe the Internet in terms of a networking infrastructure that provides services to distributed applications. Let's begin with the nuts-and-bolts description, using Figure 1.1 to illustrate our discussion.

The public Internet is a world-wide **computer network,** that is, a network that interconnects millions of computing devices throughout the world. Most of these computing devices are traditional desktop PCs, Unix-based workstations, and so

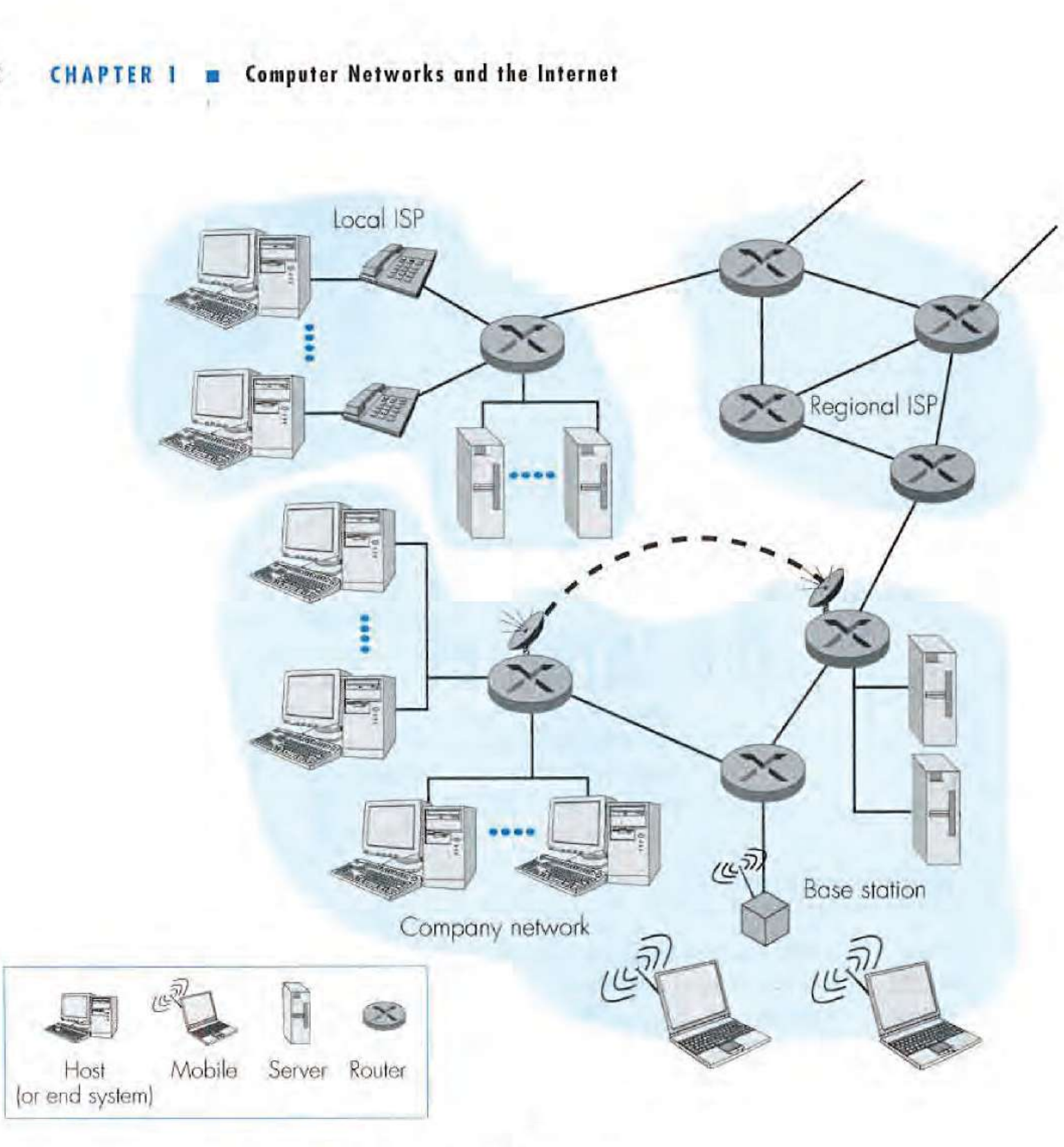


Figure 1.1 ✦ Some pieces of the Internet

called servers that store and transmit information such as Web (WWW) pages and e-mail messages. Increasingly, nontraditional computing devices such as Web TVs, mobile computers, pagers, and toasters are being connected to the Internet. (Toasters are not the only rather unusual devices to have been hooked up to the Internet; see the The Future of the Living Room [Greenberg 1997].) In the Internet jargon, all of these devices are called **hosts** or **end systems.** The Internet applications with which many of us are familiar, such as the Web and e-mail, are **network application programs** that run on such end systems. We will look into Internet end systems in more detail in Section 1.3 and then delve deeply into the study of network applications in Chapter 2.

End systems, as well as most other "pieces" of the Internet, run **protocols** that control the sending and receiving of information within the Internet. **TCP** (the Transmission Control Protocol) and **IP** (the Internet Protocol) are two of the most important protocols in the Internet. The Internet's principal protocols are collectively known as **TCP/IP.** We begin looking into protocols in Section 1.2. But that's just a start—much of this book is concerned with computer network protocols!

End systems are connected together by **communication links.** We'll see in Section 1.5 that there are many types of communication links. Links are made up of different types of **physical media,** including coaxial cable, copper wire, fiber optics, and radio spectrum. Different links can transmit data at different rates. The link transmission rate is often called the **link bandwidth** and is typically measured in bits/second.

Usually, end systems are not directly attached to each other via a single communication link. Instead, they are indirectly connected to each other through intermediate switching devices known as **routers.** A router takes information arriving on one of its incoming communication links and then forwards that information on one of its outgoing communication links. The **IP protocol** specifies the format of the information that is sent and received among routers and end systems. The path that transmitted information takes from the sending end system, through a series of communications links and routers, to the receiving end system is known as a **route** or **path** through the network. We introduce routing in more detail in Section 1.4 and study the algorithms used to determine routes, as well as the internal structure of a router itself, in Chapter 4.

Rather than provide a *dedicated* path between communicating end systems, the Internet uses a technique known as **packet switching** that allows multiple communicating end systems to share a path, or parts of a path, at the same time. The earliest ancestors of the Internet were the first packet-switched networks.

The Internet is really a network of networks. That is, the Internet is an interconnected set of privately and publicly owned and managed networks. Any network connected to the Internet must run the IP protocol and conform to certain naming and addressing conventions. Other than these few constraints, however, a network operator can configure and run its network (that is, its little piece of the Internet) however it chooses. Because of the universal use of the IP protocol in the Internet, the IP protocol is sometimes referred to as the **Internet dial tone.**

The topology of the Internet, that is, the structure of the interconnection among the various pieces of the Internet, is loosely hierarchical. Roughly speaking, from bottom-to-top, the hierarchy consists of end systems connected to local **Internet service providers (ISPs)** though **access networks.** An access network may be a so-called local area network within a company or university, a dial telephone line with a modem, or a high-speed cable-based or phone-based access network. Local ISPs are in turn connected to regional ISPs, which are in turn connected to national and international ISPs. The national and international ISPs are connected together at the highest tier in the hierarchy. New tiers and branches (that is, new networks, and new networks of networks) can be added just as a new piece of Lego can be attached to an existing Lego construction. In the first half of 1996, approximately 40,000 *new* networks were added to the Internet [Network 1996]—an astounding growth rate.

At the technical and developmental level, the Internet is made possible through creation, testing, and implementation of **Internet standards.** These standards are developed by the Internet Engineering Task Force (IETF). The IETF standards documents are called **RFCs** (request for comments). RFCs started out as general request for comments (hence the name) to resolve architecture problems that faced the precursor to the Internet. RFCs, though not formally standards, have evolved to the point where they are cited as such. RFCs tend to be quite technical and detailed. They define protocols such as TCP, IP, HTTP (for the Web), and SMTP (for open-standards e-mail). There are more than 2,000 different RFCs.

The public Internet (that is, the global network of networks discussed above) is the network that one typically refers to as *the* Internet. There are also many private networks, such as certain corporate and government networks, whose hosts are not accessible from (that is, they cannot exchange messages with) hosts outside of that private network. These private networks are often referred to as **intranets,** as they often use the same Internet technology (for example, the same types of host, routers, links, protocols, and standards) as the public Internet.

1.1.2 A Service Description

The preceding discussion has identified many of the pieces that make up the Internet. Let's now leave the nuts-and-bolts description and take a more abstract, service-oriented view:

- The Internet allows **distributed applications** running on its end systems to exchange data with each other. These applications include remote login, file transfer, electronic mail, audio and video streaming, real-time audio and video conferencing, distributed games, the World Wide Web, and much, much more [AT&T Apps 1998]. It is worth emphasizing that the Web is not a separate network but rather just one of many distributed applications that use the communication services provided by the Internet. The Web *could* also run over a network besides the Internet. One reason that the Internet is the communication medium of choice for the Web, however, is that no other existing packet-switched network connects more than 43 million [Network 1999] computers together and has over 100 million users [Almanac 1998]. (By the way, determining the number of computers hooked up to the Internet is a very difficult task, as no one is responsible for maintaining a list of who's connected. When a new network is added to the Internet, its administrators do not need to report which end systems are connected to that network. Similarly, an exiting network does not report its changes in connected end systems to any central authority.)

- The Internet provides two services to its distributed applications: a **connection-oriented service** and a **connectionless service.** Loosely speaking, connection-oriented service guarantees that data transmitted from a sender to a receiver will eventually be delivered to the receiver in order and in its entirety. Connectionless service does not make any guarantees about eventual delivery. Typically, a distributed application makes use of one or the other of these two services and not both. We examine these two different services in Section 1.3 and in great detail in Chapter 3.
- Currently, the Internet does not provide a service that makes promises about *how long* it will take to deliver the data from sender to receiver. And except for increasing your access bit rate to your Internet service provider, you currently cannot obtain better service (for example, shorter delays) by paying more—a state of affairs that some (particularly Americans!) find odd. We'll take a look at state-of-the art Internet research that is aimed at changing this situation in Chapter 6.

Our second description of the Internet—in terms of the services it provides to distributed applications—is a nontraditional, but important, one. Increasingly, advances in the nuts-and-bolts components of the Internet are being driven by the needs of new applications. So it's important to keep in mind that the Internet is an *infrastructure* in which new applications are being constantly invented and deployed.

We have given two descriptions of the Internet, one in terms of its hardware and software components, the other in terms of the services it provides to distributed applications. But perhaps you are still confused as to what the Internet is. What is packet switching, TCP/IP, and connection-oriented service? What are routers? What kinds of communication links are present in the Internet? What is a distributed application? What does the Internet have to do with children's toys? If you feel a bit overwhelmed by all of this now, don't worry—the purpose of this book is to introduce you to both the nuts and bolts of the Internet, as well as the principles that govern how and why it works. We will explain these important terms and questions in the subsequent sections and chapters.

1.1.3 Some Good Hyperlinks

As every Internet researcher knows, some of the best and most accurate information about the Internet and its protocols is not in hard-copy books, journals, or magazines. The best stuff about the Internet is in the Internet itself! Of course, there's really too much material to sift through, and sometimes the gems are few and far between. Below, we list a few generally excellent Web sites for network- and Internet-related material. Throughout the book, we will also present links to relevant, high quality URLs that provide background, original, or advanced material related to the particular topic under study. Here is a set of key links that you may want to consult while you proceed through this book:

- Internet Engineering Task Force (IETF), http://www.ietf.org: The IETF is an open international community concerned with the development and operation of the Internet and its architecture. The IETF was formally established by the Internet Architecture Board (IAB), http://www.isi.edu/iab, in 1986. The IETF meets three times a year; much of its ongoing work is conducted via mailing lists by working groups. Typically, based upon previous IETF proceedings, working groups will convene at meetings to discuss the work of the IETF working groups. The IETF is administered by the Internet Society, http://www.isoc.org, whose Web site contains lots of high-quality, Internet-related material.
- The World Wide Web Consortium (W3C), http://www.w3.org/Consortium: The W3C was founded in 1994 to develop common protocols for the evolution of the World Wide Web. This is an outstanding site with fascinating information on emerging Web technologies, protocols, and standards.
- The Association for Computing Machinery (ACM), http://www.acm.org, and the Institute of Electrical and Electronics Engineers (IEEE), http://www.ieee.org: These are the two main international professional societies that have technical conferences, magazines, and journals in the networking area. The ACM Special Interest Group in Data Communications (SIGCOMM), http://www.acm.org/sigcomm, the IEEE Communications Society, http://www.comsoc.org, and the IEEE Computer Society, http://www.computer.org, are the groups within these bodies whose efforts are most closely related to networking.
- Data communications tutorials from the online magazine Data Communications, http://www.data.com: One of the better magazines for data communications technology. The site includes many excellent tutorials.
- Media History Project, http://www.mediahistory.com: You may be wondering how the Internet got started. Or you may wonder how electrical communications got started in the first place. And you may even wonder about what preceded electrical communications! Fortunately, the Web contains an abundance of excellent resources available on these subjects. This site promotes the study of media history from petroglyphs to pixels. It covers the history of digital media, mass media, electrical media, print media, and even oral and scribal culture.

1.2 ✦ What Is a Protocol?

Now that we've got a bit of a feel for what the Internet is, let's consider another important buzzword in computer networking: "protocol." What *is* a protocol? What does a protocol *do?* How would you recognize a protocol if you met one?

1.2.1 A Human Analogy

It is probably easiest to understand the notion of a computer network protocol by first considering some human analogies, since we humans execute protocols all of the time. Consider what you do when you want to ask someone for the time of day. A typical exchange is shown in Figure 1.2. Human protocol (or good manners, at least) dictates that one first offers a greeting (the first "Hi" in Figure 1.2) to initiate communication with someone else. The typical response to a "Hi" message (at least outside of New York City) is a returned "Hi" message. Implicitly, one then takes a cordial "Hi" response as an indication that one can proceed ahead and ask for the time of day. A different response to the initial "Hi" (such as "Don't bother me!" or "I don't speak English," or an unprintable reply that one might receive in New York City) might indicate an unwillingness or inability to communicate. In this case, the human protocol would be to not ask for the time of day. Sometimes one gets no response at all to a question, in which case one typically gives up asking that person for the time. Note that in our human protocol, *there are specific messages we send, and specific actions we take in response to the received reply messages or other events (such as no reply within some given amount of time).* Clearly transmitted and received messages, and actions taken when these messages are sent or received or other events occur, play a central role in a human protocol. If people run different protocols (for example, if one person has manners but the other does not, or if one

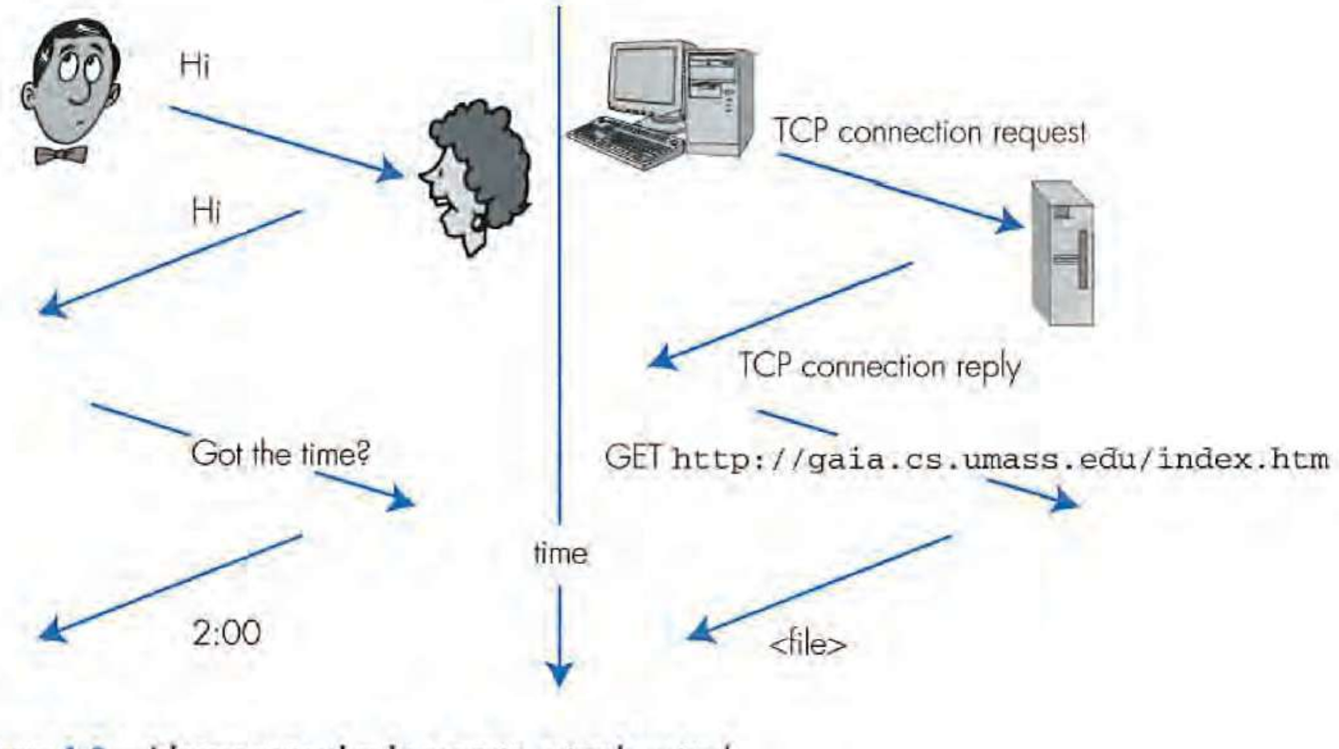


Figure 1.2 ♦ A human protocol and a computer network protocol

understands the concept of time and the other does not) the protocols do not interoperate and no useful work can be accomplished. The same is true in networking—it takes two (or more) communicating entities running the same protocol in order to accomplish a task.

Let's consider a second human analogy. Suppose you're in a college class (a computer networking class, for example!). The teacher is droning on about protocols and you're confused. The teacher stops to ask, "Are there any questions?" (a message that is transmitted to, and received by, all students who are not sleeping). You raise your hand (transmitting an implicit message to the teacher). Your teacher acknowledges you with a smile, saying "Yes . . ." (a transmitted message encouraging you to ask your question—teachers *love* to be asked questions) and you then ask your question (that is, transmit your message to your teacher). Your teacher hears your question (receives your question message) and answers (transmits a reply to you). Once again, we see that the transmission and receipt of messages, and a set of conventional actions taken when these messages are sent and received, are at the heart of this question-and-answer protocol.

1.2.2 Network Protocols

A network protocol is similar to a human protocol, except that the entities exchanging messages and taking actions are hardware or software components of a computer network, components that we will study shortly in the following sections. All activity in the Internet that involves two or more communicating remote entities is governed by a protocol. Protocols in routers determine a packet's path from source to destination; hardware-implemented protocols in the network interface cards of two physically connected computers control the flow of bits on the "wire" between the two computers; a congestion-control protocol controls the rate at which packets are transmitted between sender and receiver. Protocols are running everywhere in the Internet, and consequently much of this book is about computer network protocols.

As an example of a computer network protocol with which you are probably familiar, consider what happens when you make a request to a Web server, that is, when you type in the URL of a Web page into your Web browser. The scenario is illustrated in the right half of Figure 1.2. First, your computer will send a "connection request" message to the Web server and wait for a reply. The Web server will eventually receive your connection request message and return a "connection reply" message. Knowing that it is now OK to request the Web document, your computer then sends the name of the Web page it wants to fetch from that Web server in a "get" message. Finally, the Web server returns the contents of the Web document to your computer.

Given the human and networking examples above, the exchange of messages and the actions taken when these messages are sent and received are the key defining elements of a protocol:

A ***protocol*** *defines the format and the order of messages exchanged between two or more communicating entities, as well as the actions taken on the transmission and/or receipt of a message or other event.*

The Internet, and computer networks in general, make extensive use of protocols. Different protocols are used to accomplish different communication tasks. As you read through this book, you will learn that some protocols are simple and straightforward, while others are complex and intellectually deep. Mastering the field of computer networking is equivalent to understanding the what, why, and how of networking protocols.

1.3 ✦ The Network Edge

In the previous sections we presented a high-level description of the Internet and networking protocols. We are now going to delve a bit more deeply into the components of the Internet. We begin in this section at the edge of network and look at the components with which we are most familiar—the computers (for example, PCs and workstations) that we use on a daily basis. In the next section we will move from the network edge to the network core and examine switching and routing in computer networks. Then in Section 1.5 we will discuss the actual physical links that carry the signals sent between the computers and the switches.

1.3.1 End Systems, Clients, and Servers

In computer networking jargon, the computers that we use on a daily basis are often referred to as **hosts** or **end systems.** They are referred to as hosts because they host (run) application-level programs such as a Web browser or server program, or an e-mail program. They are also referred to as end systems because they sit at the edge of the Internet, as shown in Figure 1.3. Throughout this book we will use the terms hosts and end systems interchangeably; that is, *host = end system.*

Hosts are sometimes further divided into two categories: **clients** and **servers.** Informally, clients often tend to be desktop PCs or workstations, whereas servers are more powerful machines. But there is a more precise meaning of a client and a server in computer networking. In the so-called **client/server model,** a client program running on one end system requests and receives information from a server running on another end system. Studied in detail in Chapter 2, this client/server model is undoubtedly the most prevalent structure for Internet applications. The Web, e-mail, file transfer, remote login (for example, Telnet), newgroups, and many other popular applications adopt the client/server model. Since a client typically runs on one computer and the server runs on another computer, client/server Internet applications are,

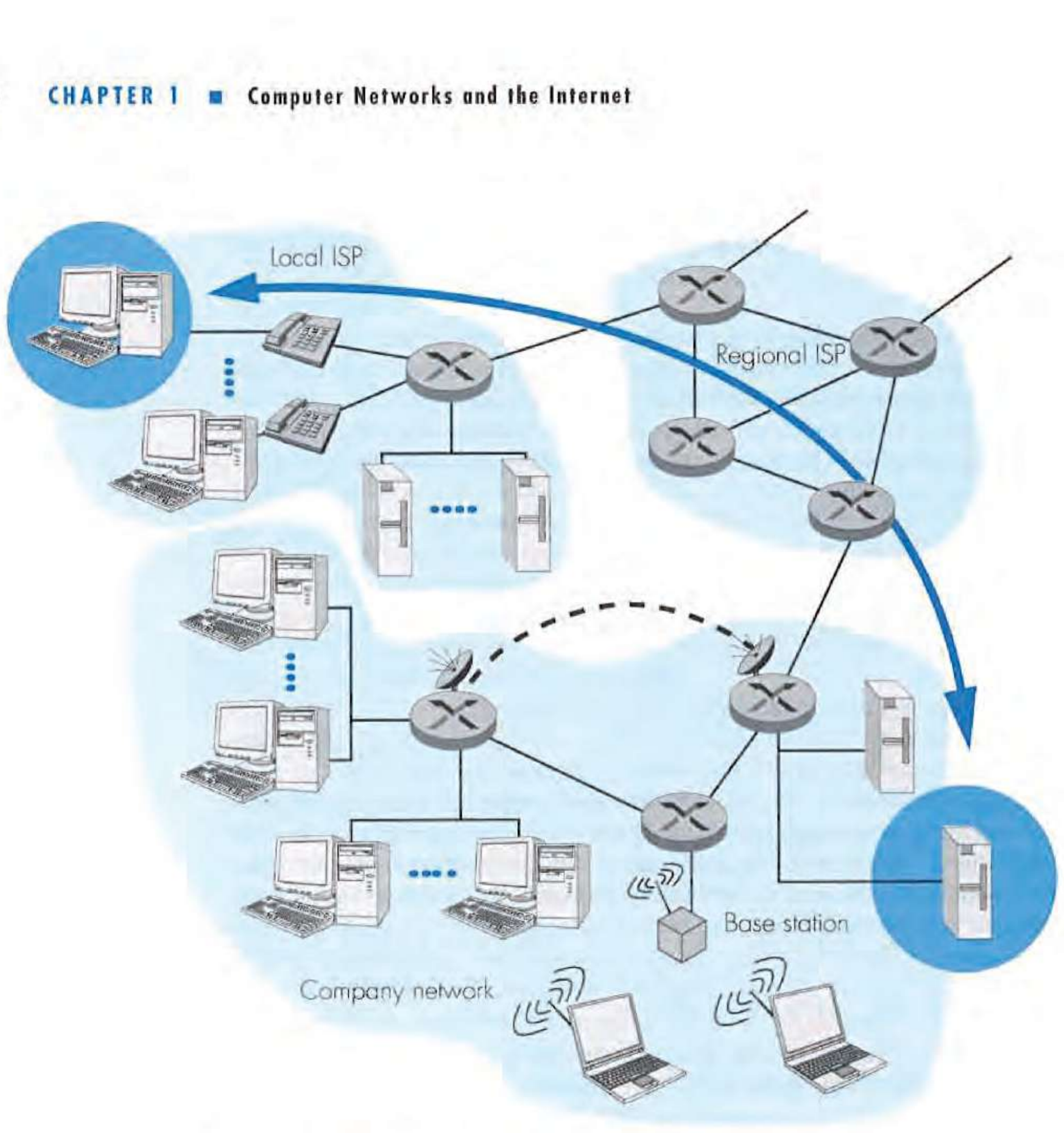


Figure 1.3 ◆ End-system interaction

by definition, **distributed applications.** The client and the server interact with each other by communicating (that is, sending each other messages) over the Internet. At this level of abstraction, the routers, links and other "pieces" of the Internet serve as a "black box" that transfers messages between the distributed, communicating components of an Internet application. This is the level of abstraction depicted in Figure 1.3.

Computers (for example, a PC or a workstation), operating as clients and servers, are the most prevalent type of end system. However, an increasing number of alternative devices, such as so-called network computers and thin clients [Thinplanet 2000], Web TVs and set top boxes [Mills 1998], digital cameras, and other devices are being attached to the Internet as end systems. For an interesting discussion of the continuing evolution of Internet applications, see [AT&T Apps 1998; Dertouzos 1999; Lucky 1997].

1.3.2 Connectionless and Connection-Oriented Services

We have seen that end systems exchange messages with each other according to an application-level protocol in order to accomplish some task. The links, routers, and other pieces of the Internet provide the means to transport these messages between the end-system applications. But what are the characteristics of the communication services that are provided? The Internet, and more generally TCP/IP networks, provide two types of services to its applications: **connectionless service** and **connection-oriented service.** A developer creating an Internet application (for example, an e-mail application, a file transfer application, a Web application, or an Internet phone application) must program the application to use one of these two services. Here, we only briefly describe these two services; we discuss them in much more detail in Chapter 3, which covers transport layer protocols.

Connection-Oriented Service

When an application uses the connection-oriented service, the client and the server (residing in different end systems) send control packets to each other before sending packets with real data (such as e-mail messages). This so-called handshaking procedure alerts the client and server, allowing them to prepare for an onslaught of packets. It is interesting to note that this initial handshaking procedure is similar to the protocol used in human interaction. The exchange of "Hi's" we saw in Figure 1.2 is an example of a human "handshaking protocol" (even though handshaking is not literally taking place between the two people). The two TCP messages that are exchanged as part of the WWW interaction shown in Figure 1.2 are two of the three messages exchanged when TCP sets up a connection between a sender and receiver. The third TCP message (not shown) that forms the final part of the TCP three-way handshake (see Section 3.5) is contained in the get message shown in Figure 1.2.

Once the handshaking procedure is finished, a connection is said to be established between the two end systems. But the two end systems are connected in a very loose manner, hence the terminology connection-oriented. In particular, only the end systems themselves are aware of this connection; the packet switches (that is, routers) within the Internet are completely oblivious to the connection. This is because a TCP connection consists of nothing more than allocated resources (buffers) and state variables in the end systems. The packet switches do not maintain any connection-state information.

The Internet's connection-oriented service comes bundled with several other services, including reliable data transfer, flow control, and congestion control. By **reliable data transfer,** we mean that an application can rely on the connection to deliver all of its data without error and in the proper order. Reliability in the Internet is achieved through the use of acknowledgments and retransmissions. To get a preliminary idea about how the Internet implements the reliable transport service, consider an application that has established a connection between end systems A and B. When end system B receives a packet from A, it sends an acknowledgment; when end system A receives the acknowledgment, it knows that the corresponding packet has definitely been received. When end system A doesn't receive an acknowledgment, it assumes that the packet it sent was not received by B; it therefore retransmits the packet. **Flow control** makes sure that neither side of a connection overwhelms the other side by sending too many packets too fast. Indeed, the application at one side of the connection may not be able to process information as quickly as it receives the information. Therefore, there is a risk of overwhelming either side of an application. The flow-control service forces the sending end system to reduce its rate whenever there is such a risk. We shall see in Chapter 3 that the Internet implements the flow control service by using sender and receiver buffers in the communicating end systems. The Internet's **congestion-control** service helps prevent the Internet from entering a state of gridlock. When a router becomes congested, its buffers can overflow and packet loss can occur. In such circumstances, if every pair of communicating end systems continues to pump packets into the network as fast as they can, gridlock sets in and few packets are delivered to their destinations. The Internet avoids this problem by forcing end systems to decrease the rate at which they send packets into the network during periods of congestion. End systems are alerted to the existence of severe congestion when they stop receiving acknowledgments for the packets they have sent.

We emphasize here that although the Internet's connection-oriented service comes bundled with reliable data transfer, flow control, and congestion control, these three features are by no means essential components of a connection-oriented service. A different type of computer network may provide a connection-oriented service to its applications without bundling in one or more of these features. Indeed, any protocol that performs handshaking between the communicating entities before transferring data is a connection-oriented service [Iren 1999].

The Internet's connection-oriented service has a name—**TCP** (Transmission Control Protocol); the initial version of the TCP protocol is defined in the Internet Request for Comments RFC 793 [RFC 793]. The services that TCP provides to an application include reliable transport, flow control, and congestion control. It is important to note that an application need only care about the services that are provided; it need not worry about *how* TCP actually implements reliability, flow control, or congestion control. *We*, of course, are very interested in how TCP implements these services, and we shall cover these topics in detail in Chapter 3.

Connectionless Service

There is no handshaking with the Internet's connectionless service. When one side of an application wants to send packets to another side of an application, the sending application simply sends the packets. Since there is no handshaking procedure prior to the transmission of the packets, data can be delivered faster. But there are no acknowledgments either, so a source never knows for sure which packets arrive at the destination. Moreover, the service makes no provision for flow control or congestion control. The Internet's connectionless service is provided by **UDP** (User Datagram Protocol); UDP is defined in the Internet Request for Comments RFC 768.

Most of the more familiar Internet applications use TCP, the Internet's connection-oriented service. These applications include Telnet (remote login), SMTP (for electronic mail), FTP (for file transfer), and HTTP (for the Web). Nevertheless, UDP, the Internet's connectionless service, is used by many applications, including many of the emerging multimedia applications, such as Internet phone, audio-on-demand, and video conferencing.

1.4 ✦ The Network Core

Having examined the end systems and end–end transport service model of the Internet in Section 1.3, let us now delve more deeply into the "inside" of the network. In this section we study the network core—the mesh of routers that interconnect the Internet's end systems. Figure 1.4 highlights the network core in the thick, shaded lines.

1.4.1 Circuit Switching, Packet Switching, and Message Switching

There are two fundamental approaches towards building a network core: **circuit switching** and **packet switching.** In circuit-switched networks, the resources needed along a path (buffers, link bandwidth) to provide for communication between the end systems are *reserved* for the duration of the session. In packet-switched networks, these resources are *not* reserved; a session's messages use the resource on demand, and as a consequence, may have to wait (that is, queue) for access to a communication link. As a simple analogy, consider two restaurants—one that requires reservations and another that neither requires reservations nor accepts them. For the restaurant that requires reservations, we have to go through the hassle of first calling before we leave home. But when we arrive at the restaurant we can, in principle, immediately communicate with the waiter and order our meal. For the restaurant that does not require reservations, we don't need to bother to reserve a table. But when we arrive at the restaurant, we may have to wait for a table before we can communicate with the waiter.

The ubiquitous telephone networks are examples of circuit-switched networks. Consider what happens when one person wants to send information (voice or facsimile) to another over a telephone network. Before the sender can send the information, the network must first establish a connection between the sender and the receiver. In contrast with the TCP connection that we discussed in the previous section, this is a

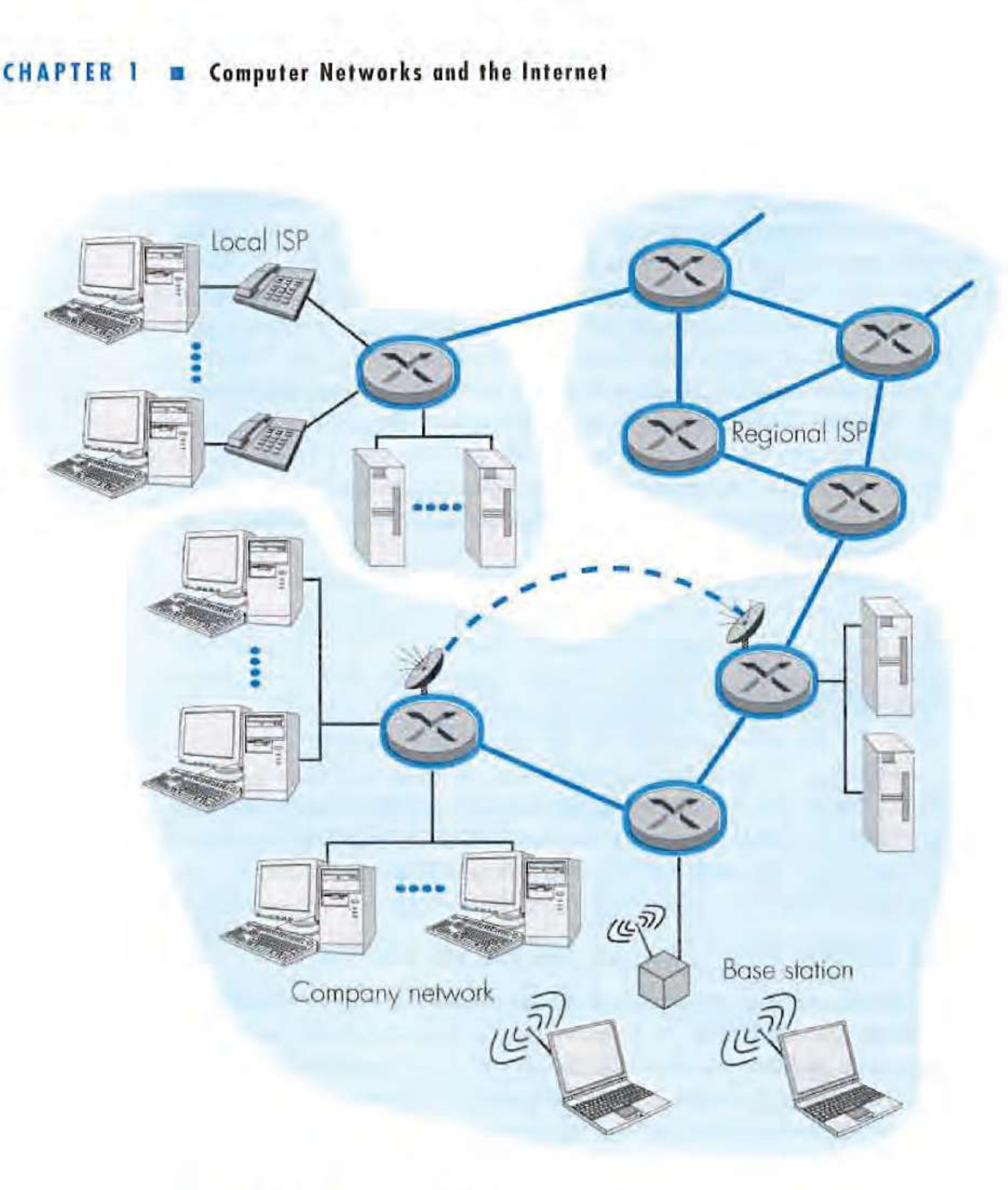


Figure 1.4 ✦ The network core

bona fide connection for which the switches on the path between the sender and receiver maintain connection state for that connection. In the jargon of telephony, this connection is called a **circuit.** When the network establishes the circuit, it also reserves a constant transmission rate in the network's links for the duration of the connection. This reservation allows the sender to transfer the data to the receiver at the *guaranteed* constant rate.

Today's Internet is a quintessential packet-switched network. Consider what happens when one host wants to send a packet to another host over a packet-switched network. As with circuit switching, the packet is transmitted over a series of communication links. But with packet switching, the packet is sent into the network without reserving any bandwidth whatsoever. If one of the links is congested because other packets need to be transmitted over the link at the same time, then our packet will have to wait in a buffer at the sending side of the transmission line, and suffer a delay. The Internet makes its *best effort* to deliver the data in a timely manner, but it does not make any guarantees.

Not all telecommunication networks can be neatly classified as pure circuit-switched networks or pure packet-switched networks. For example, for networks based on the ATM technology, a connection can make a reservation and yet its messages may still wait for congested resources! Nevertheless, this fundamental classification into packet- and circuit-switched networks is an excellent starting point in understanding telecommunication network technology.

Circuit Switching

This book is about computer networks, the Internet, and packet switching, not about telephone networks and circuit switching. Nevertheless, it is important to understand why the Internet and other computer networks use packet switching rather than the more traditional circuit-switching technology used in the telephone networks. For this reason, we now give a brief overview of circuit switching.

Figure 1.5 illustrates a circuit-switched network. In this network, the three circuit switches are interconnected by two links; each of these links has n circuits, so that each link can support n simultaneous connections. The end systems (for example, PCs and workstations) are each directly connected to one of the switches. (Ordinary telephones are also connected to the switches, but they are not shown in the diagram.) Notice that some of the hosts have analog access to the switches, whereas others have direct digital access. For analog access, a modem is required. When two hosts desire to communicate, the network establishes a dedicated *end-to-end circuit* between two hosts. (Conference calls between more than two devices are, of course, also possible. But to keep things simple, let's suppose for now that there are only two hosts for each connection.) Thus, in order for host A to send messages to host B, the network must first reserve one circuit on each of two links. Each link has n circuits; each end-to-end circuit over a link gets the fraction $1/n$ of the link's bandwidth for the duration of the circuit.

Multiplexing

A circuit in a link is implemented with either **frequency-division multiplexing (FDM)** or **time-division multiplexing (TDM)**. With FDM, the frequency spectrum of a link is shared among the connections established across the link. Specifically, the link dedicates a frequency band to each connection for the duration of the connection. In telephone networks, this frequency band typically has a width of 4

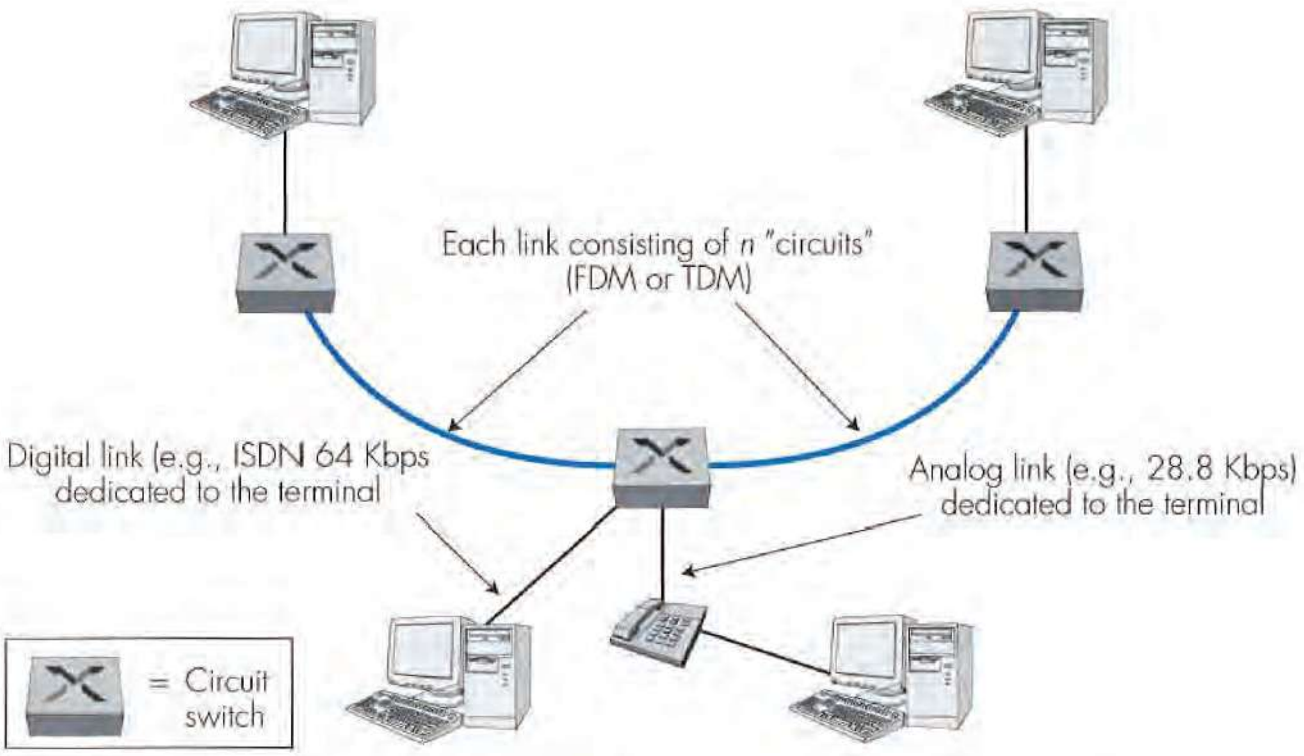

Each link consisting of n "circuits" (FDM or TDM)
Digital link (e.g., ISDN 64 Kbps dedicated to the terminal
Analog link (e.g., 28.8 Kbps) dedicated to the terminal
= Circuit switch

Figure 1.5 ✦ A simple circuit-switched network consisting of three circuit switches interconnected with two links

kHz (that is, 4,000 Hertz or 4,000 cycles per second). The width of the band is called, not surprisingly, the **bandwidth.** FM radio stations also use FDM to share the microwave frequency spectrum.

The trend in modern telephony is to replace FDM with TDM. Most links in most telephone systems in the United States and in other developed countries currently employ TDM. For a TDM link, time is divided into frames of fixed duration, and each frame is divided into a fixed number of time slots. When the network establishes a connection across a link, the network dedicates one time slot in every frame to the connection. These slots are dedicated for the sole use of that connection, with a time slot available for use (in every frame) to transmit the connection's data.

Figure 1.6 illustrates FDM and TDM for a specific network link. For FDM, the frequency domain is segmented into a number of circuits, each of bandwidth 4 kHz. For TDM, the time domain is segmented into four circuits; each circuit is assigned the same dedicated slot in the revolving TDM frames. The transmission rate of the frame is equal to the frame rate multiplied by the number of bits in a slot. For example, if the link transmits 8,000 frames per second and each slot consists of 8 bits, then the transmission rate is 64 Kbps.

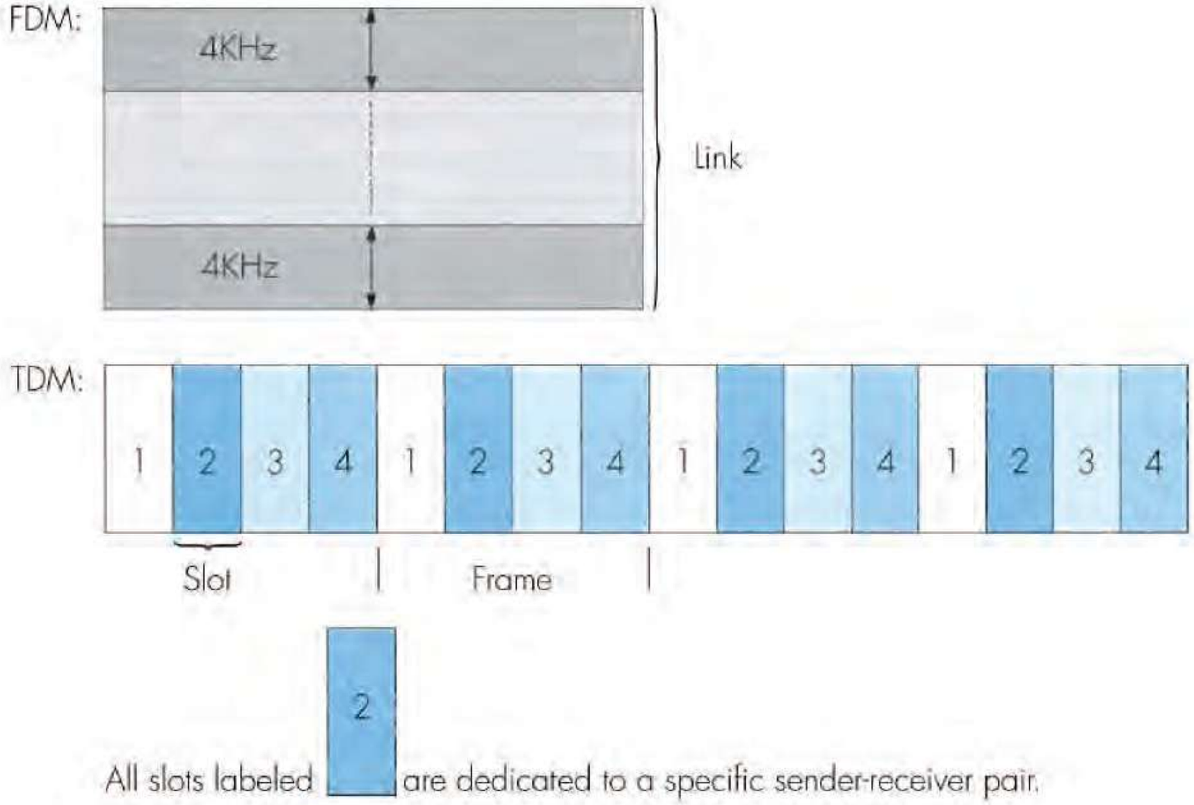

FDM:
4KHz
4KHz
Link
TDM:
1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
Slot
Frame
2
All slots labeled 2 are dedicated to a specific sender-receiver pair.

Figure 1.6 ♦ With FDM, each circuit continuously gets a fraction of the bandwidth. With TDM, each circuit gets all of the bandwidth periodically during brief intervals of time (that is, during slots).

Proponents of packet switching have always argued that circuit switching is wasteful because the dedicated circuits are idle during **silent periods.** For example, when one of the conversants in a telephone call stops talking, the idle network resources (frequency bands or slots in the links along the connection's route) cannot be used by other ongoing connections. As another example of how these resources can be underutilized, consider a radiologist who uses a circuit-switched network to remotely access a series of x-rays. The radiologist sets up a connection, requests an image, contemplates the image, and then requests a new image. Network resources are wasted during the radiologist's contemplation periods. Proponents of packet switching also enjoy pointing out that establishing end-to-end circuits and reserving end-to-end bandwidth is complicated and requires complex signaling software to coordinate the operation of the switches along the end-to-end path.

Before we finish our discussion of circuit switching, let's work through a numerical example that should shed further insight on the matter. Let us consider how long it takes to send a file of 640 Kbits from host A to host B over a circuit- switched network. Suppose that all links in the network use TDM with 24 slots and have a bit rate of 1.536 Mbps. Also suppose that it takes 500 msec to establish an end-to-end circuit before A can begin to transmit the file. How long does it take to send the file? Each circuit has a transmission rate of (1.536 Mbps)/24 = 64 Kbps, so it takes (640 Kbits)/(64 Kbps) = 10 seconds to transmit the file. To this 10 seconds we add the circuit establishment time, giving 10.5 seconds to send the file. Note that the transmission time is independent of the number of links. The transmission time would be 10 seconds if the end-to-end circuit passes through one link or one hundred links. (The actual end-to-end delay also includes a propagation delay; see Section 1.6). AT&T Labs provides an interactive site [AT&T Bandwidth 1998] to explore transmission delay for various file types and transmission technologies.

Packet Switching

We saw in Sections 1.2 and 1.3 that application-level protocols exchange **messages** in accomplishing their task. Messages can contain anything the protocol designer desires. Messages may perform a control function (for example, the "Hi" messages in our handshaking example) or can contain data, such as an ASCII file, a Postscript file, a Web page, or a digital audio file. In modern packet-switched networks, the source breaks long messages into smaller **packets.** Between source and destination, each of these packets traverse communication links and **packet switches** (also known as **routers**). Packets are transmitted over each communication link at a rate equal to the *full* transmission rate of the link. Most packet switches use **store-and-forward transmission** at the inputs to the links. Store-and-forward transmission means that the switch must receive the entire packet before it can begin to transmit the first bit of the packet onto the outbound link. Thus store-and-forward packet switches introduce a **store-and-forward delay** at the input to each link along the packet's route. This delay is proportional to the packet's length in bits. In particular, if a packet consists of L bits, and the packet is to be forwarded onto an outbound link of R bps, then the store-and-forward delay at the switch is L/R seconds.

Within each router there are multiple buffers (also called queues), with each link having an **input buffer** (to store packets that have just arrived to that link) and an **output buffer.** The output buffers play a key role in packet switching. If an arriving packet needs to be transmitted across a link but finds the link busy with the transmission of another packet, the arriving packet must wait in the output buffer. Thus, in addition to the store-and-forward delays, packets suffer output buffer **queuing delays.** These delays are variable and depend on the level of congestion in the network. Since the amount of buffer space is finite, an arriving packet may find that the buffer is completely filled with other packets waiting for transmission. In this case, **packet loss** will occur—either the arriving packet or one of the already-queued packets will be dropped. Returning to our restaurant analogy from earlier in this section, the queuing delay is analogous to the amount of time one spends waiting for a table. Packet loss is analogous to being told by the waiter that you must leave the premises because

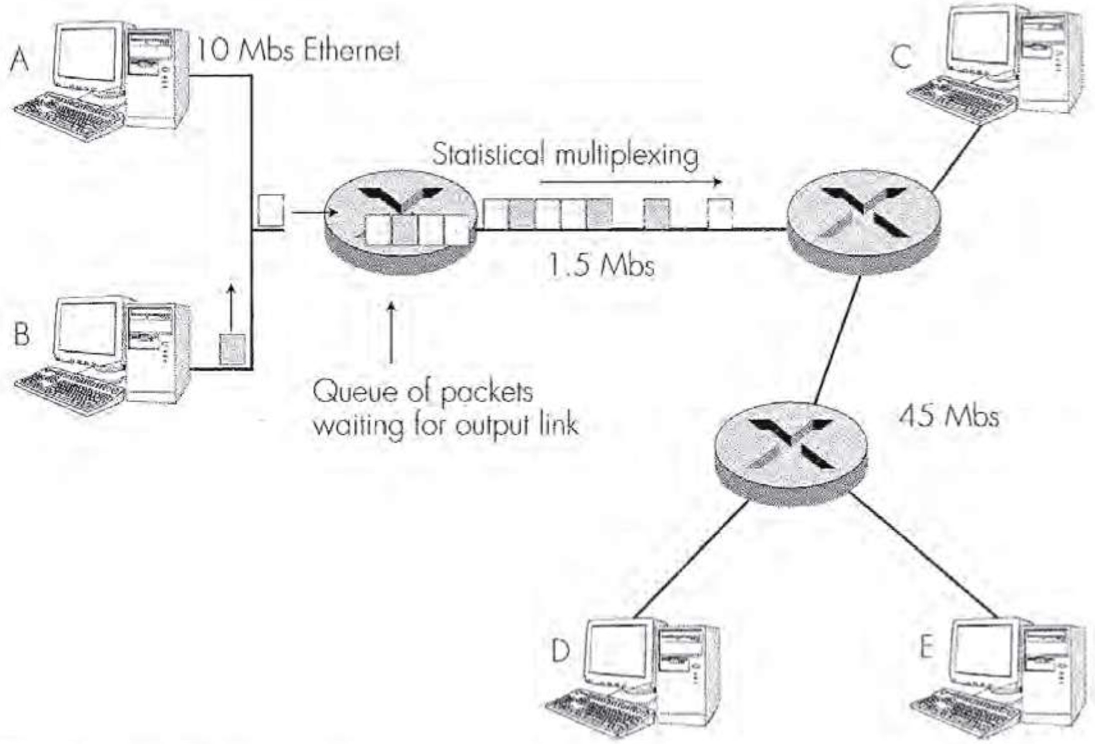


Figure 1.7 ◆ Packet switching

there are already too many other people waiting at the bar for a table.

Figure 1.7 illustrates a simple packet-switched network. Suppose Hosts A and B are sending packets to Host E. Hosts A and B first send their packets along the 10 Mbps link to the first packet switch. The packet switch directs these packets to the 1.544 Mbps link. If there is congestion at this link, the packets queue in the link's output buffer before they can be transmitted onto the link. Consider now how Host A and Host B packets are transmitted onto this link. As shown in Figure 1.7, the sequence of A and B packets does not follow any periodic ordering; the ordering is random or statistical because packets are sent whenever they happen to be present at the link. For this reason, we often say that packet switching employs **statistical multiplexing.** Statistical multiplexing sharply contrasts with time-division multiplexing (TDM), for which each host gets the same slot in a revolving TDM frame.

Let us now consider how long it takes to send a packet of L bits from one host to another host across a packet-switched network. Let us suppose that there are Q links between the two hosts, each of rate R bps. Assume that queuing delays and end-to-end propagation delays are negligible and that there is no connection establishment. The packet must first be transmitted onto the first link emanating from host A; this takes L/R seconds. It must then be transmitted on each of the $Q-1$ remaining links, that is, it must be stored and forwarded $Q-1$ times. Thus the total delay is QL/R.

Packet Switching versus Circuit Switching

Having described circuit switching and packet switching, let us compare the two. Opponents of packet switching have often argued that packet switching is not suitable for real-time services (for example, telephone calls and video conference calls) because of its variable and unpredictable delays. Proponents of packet switching argue that (1) it offers better sharing of bandwidth than circuit switching and (2) it is simpler, more efficient, and less costly to implement than circuit switching. Generally speaking, people who do not like to hassle with restaurant reservations prefer packet switching to circuit switching.

Why is packet switching more efficient? Let us look at a simple example. Suppose users share a 1 Mbps link. Also suppose that each user alternates between periods of activity (when it generates data at a constant rate of 100 Kbps) and periods of inactivity (when it generates no data). Suppose further that a user is active only 10 percent of the time (and is idle drinking coffee during the remaining 90 percent of the time). With circuit switching, 100 Kbps must be *reserved* for *each* user at all times. Thus, the link can support only 10 simultaneous users. With packet switching, if there are 35 users, the probability that there are 10 or more simultaneously active users is less than 0.0017. If there are 10 or fewer simultaneously active users (which happens with probability 0.9983), the aggregate arrival rate of data is less than 1 Mbps (the output rate of the link). Thus, users' packets flow through the link essentially without delay, as is the case with circuit switching. When there are more than 10 simultaneously active users, then the aggregate arrival rate of packets will exceed the output capacity of the link, and the output queue will begin to grow (until the aggregate input rate falls back below 1 Mbps, at which point the queue will begin to diminish in length). Because the probability of having 10 or more simultaneously active users is very very small, packet-switching almost always has the same delay performance as circuit switching, *but does so while allowing for more than three times the number of users.*

Although packet switching and circuit switching are both very prevalent in today's telecommunication networks, the trend is certainly in the direction of packet switching. Even many of today's circuit-switched telephone networks are slowly migrating towards packet switching. In particular, telephone networks often convert to packet switching for the expensive overseas portion of a telephone call.

Message Switching

In a modern packet-switched network, the source host segments long messages into smaller packets and sends the smaller packets into the network; the receiver reassembles the packets back into the original message. But why bother to segment the messages into packets in the first place, only to have to reassemble packets into messages? Doesn't this place an additional and unnecessary burden on the source and destination? Although the segmentation and reassembly do complicate the design of

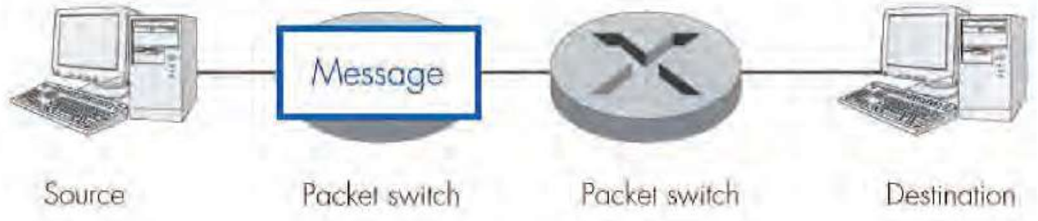


Figure 1.8 ✦ A simple message-switched network

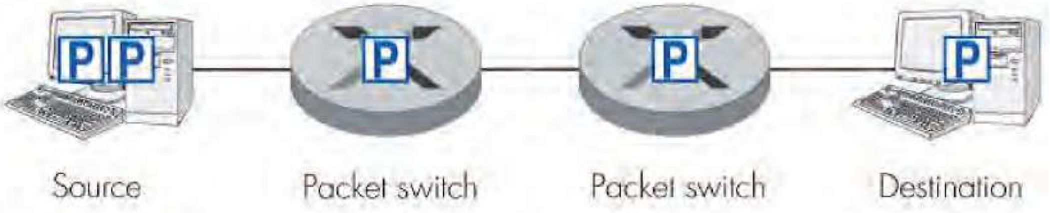


Figure 1.9 ✦ A simple packet-switched network

the source and receiver, researchers and network designers concluded in the early days of packet switching that the advantages of segmentation greatly compensate for its complexity. Before discussing some of these advantages, we need to introduce some terminology. We say that a packet-switched network performs **message switching** if the sources do not segment messages (that is, they send a message into the network as a whole). Thus message switching is a specific kind of packet switching, whereby the packets traversing the network are themselves entire messages.

Figure 1.8 illustrates message switching in a route consisting of two packet switches (PSs) and three links. With message switching, the message stays intact as it traverses the network. Because the switches are store-and-forward packet switches, a packet switch must receive the entire message before it can begin to forward the message on an outbound link.

Figure 1.9 illustrates packet switching for the same network. In this example, the original message has been divided into five distinct packets. In Figure 1.9, the first packet has arrived at the destination, the second and third packets are in transit in the network, and the last two packets are still in the source. Again, because the switches are store-and-forward packet switches, a packet switch must receive an entire packet before it can begin to forward the packet on an outbound link.

One major advantage of packet switching (with segmented messages) is that it achieves end-to-end delays that are typically much smaller than the delays associated with message switching. We illustrate this point with the following simple example. Consider a message that is 7.5 Mbits long. Suppose that between source and destination there are two packet switches and three links, and that each link has a

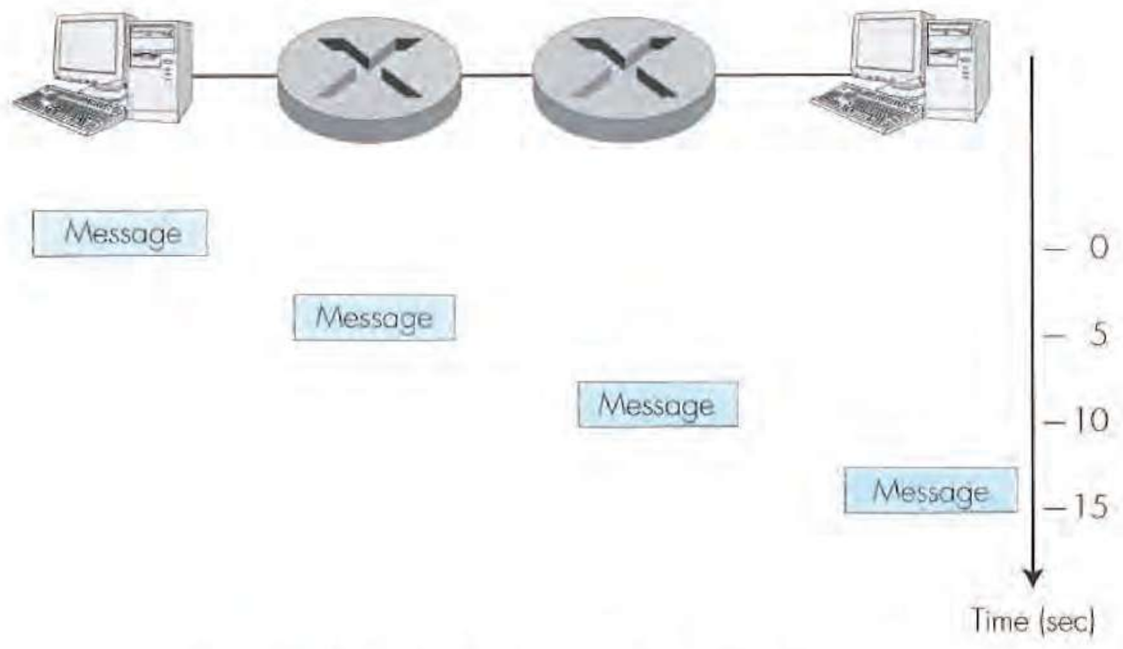


Figure 1.10 ◆ Timing of message transfer of a 7.5 Mbit message in a message-switched network

transmission rate of 1.5 Mbps. Assuming there is no congestion in the network, how much time is required to move the message from source to destination with message switching? It takes the source 5 seconds to move the message from the source to the first switch. Because the switches use store-and-forward transmission, the first switch cannot begin to transmit any bits in the message onto the link until this first switch has received the entire message. Once the first switch has received the entire message, it takes 5 seconds to move the message from the first switch to the second switch. Thus it takes 10 seconds to move the message from the source to the second switch. Following this logic we see that a total of 15 seconds is needed to move the message from source to destination. These delays are illustrated in Figure 1.10.

Continuing with the same example, now suppose that the source breaks the message into 5,000 packets, with each packet being 1.5 Kbits long. Again assuming that there is no congestion in the network, how long does it take to move the 5,000 packets from source to destination? It takes the source 1 msec to move the first packet from the source to the first switch. And it takes the first switch 1 msec to move this first packet from the first to the second switch. But while the first packet is being moved from the first switch to the second switch, the second packet is *simultaneously* moved from the source to the first switch. Thus the second packet reaches the first switch at time = 2 msec. Following this logic we see that the last packet is completely received at the first switch at time = 5,000 msec = 5 seconds. Since this last packet has to be transmitted on two more links, the last packet is received by the destination at 5.002 seconds (see Figure 1.11).

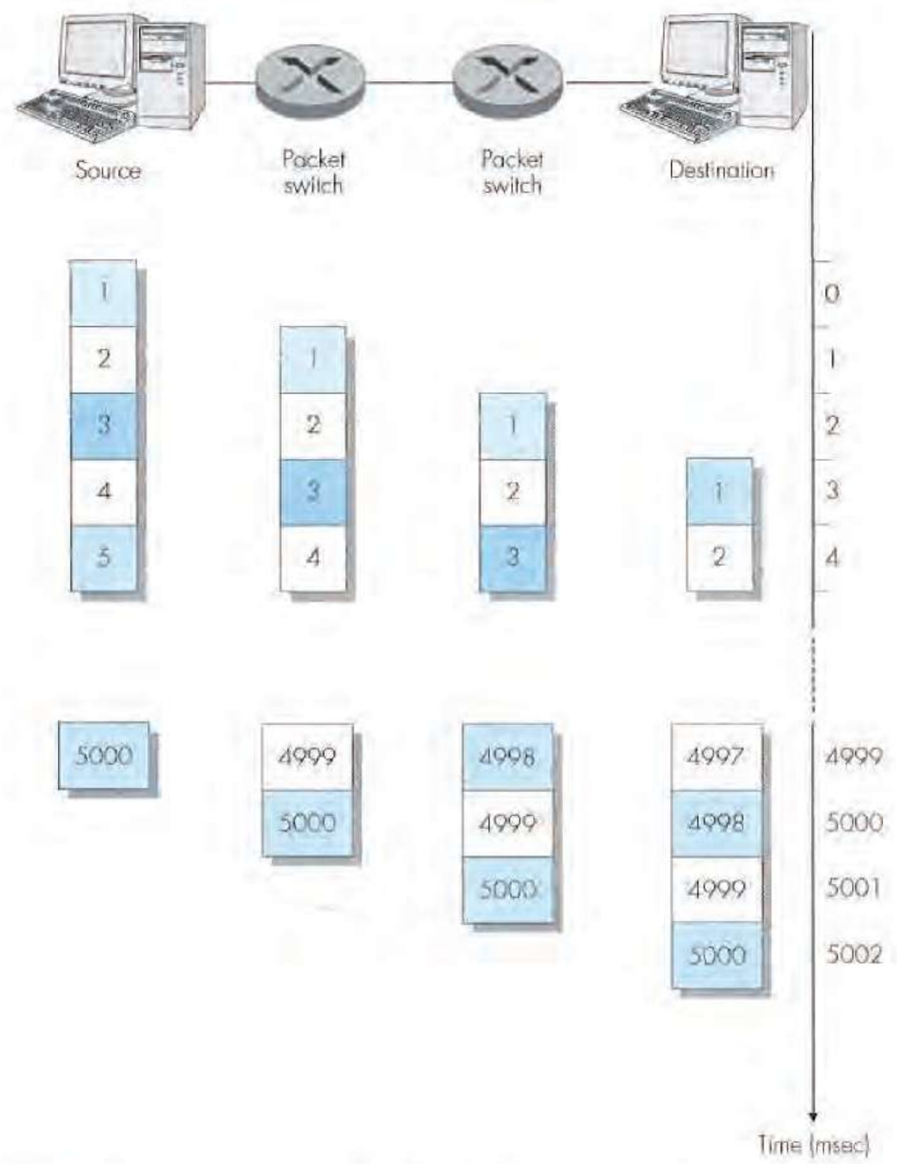


Figure 1.11 ♦ Timing of packet transfer of a 7.5 Mbit message, divided into 5,000 packets, in a packet-switched network

Amazingly enough, packet switching has reduced the message-switching delay by a factor of three! But why is this so? What is packet switching doing that is different from message switching? The key difference is that message switching is performing sequential transmission whereas packet switching is performing parallel transmission. Observe that with message switching, while one node (the source or one of the switches) is transmitting, the remaining nodes are idle. With packet switching, once the first packet reaches the last switch, three nodes transmit at the same time.

Packet switching has yet another important advantage over message switching. As we will discuss later in this book, bit errors can be introduced into packets as they transit the network. When a switch detects an error in a packet, it typically discards the entire packet. So, if the entire message is a packet and one bit in the message gets corrupted, the entire message is discarded. If, on the other hand, the message is segmented into many packets and one bit in one of the packets is corrupted, then only that one packet is discarded.

Packet switching is not without its disadvantages, however. We will see that each packet or message must carry, in addition to the data being sent from the sending application to the receiving application, an amount of control information. This information, which is carried in the packet or message **header**, might include the identity of the sender and receiver and a packet or message identifier (for example, number). Since the amount of header information would be approximately the same for a message or a packet, the amount of header overhead per byte of data is higher for packet switching than for message switching.

Before moving on to the next subsection, you are highly encouraged to explore the Message-Switching Java Applet (http://www.awl.com/kurose-ross) that is available on the WWW site for this book. This applet will allow you to experiment with different message and packet sizes, and will allow you to examine the effect of additional propagation delays.

1.4.2 Routing in Data Networks

There are two broad classes of packet-switched networks: datagram networks and virtual circuit networks. They differ according to whether they route packets according to host destination addresses or according to virtual circuit numbers. We shall call any network that routes packets according to host destination addresses a **datagram network.** The IP protocol of the Internet routes packets according to the destination addresses; hence the Internet is a datagram network. We shall call any network that routes packets according to virtual circuit numbers a **virtual circuit network.** Examples of packet-switching technologies that use virtual circuits include X.25, frame relay, and ATM (asynchronous transfer mode).

Virtual Circuit Networks

A virtual circuit (VC) consists of (1) a path (that is, a series of links and packet switches) between the source and destination hosts, (2) virtual circuit numbers, one number for each link along the path, and (3) entries in VC-number translation tables in each packet switch along the path. Once a VC is established between source and destination, packets can be sent with the appropriate VC numbers. Because a VC has a different VC number on each link, an intermediate packet switch must replace the

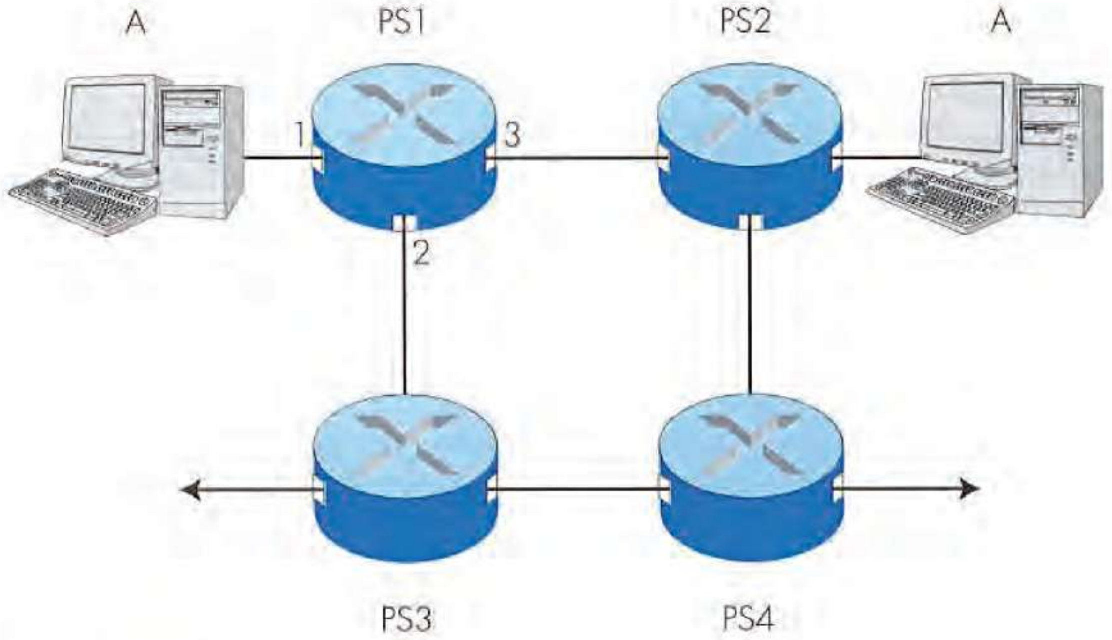


Figure 1.12 ✦ A simple virtual circuit network

VC number of each traversing packet with a new one. The new VC number is obtained from the VC-number translation table.

To illustrate the concept, consider the network shown in Figure 1.12. Suppose host A requests that the network establish a VC between itself and host B. Suppose that the network chooses the path A–PS1–PS2–B and assigns VC numbers 12, 22, 32 to the three links in this path. Then, when a packet as part of this VC leaves host A, the value in the VC-number field is 12; when it leaves PS1, the value is 22; and when it leaves PS2, the value is 32. The numbers next to the links of PS1 are the **interface numbers.**

How does the switch determine the replacement VC number for a packet traversing the switch? Each switch has a VC-number translation table; for example, the VC-number translation table in PS1 might look something like this:

Incoming Interface	Incoming VC #	Outgoing Interface	Outgoing VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
...	...	...	...

Whenever a new VC is established across a switch, an entry is added to the VC-number table. Similarly, whenever a VC terminates, the entries in each table along its path are removed.

You might be wondering why a packet doesn't just keep the same VC number on each of the links along its route. The answer is twofold. First, by replacing the number from link to link, the length of the VC field is reduced. Second, and more importantly, by permitting a different VC number for each link along the path of the VC, a network management function is simplified. Specifically, with multiple VC numbers, each link in the path can choose a VC number independently of what the other links in the path chose. If a common number were required for all links along the path, the switches would have to exchange and process a substantial number of messages to agree on the VC number to be used for a connection.

If a network employs virtual circuits, then the network's switches must maintain **state information** for the ongoing connections. Specifically, each time a new connection is established across a switch, a new connection entry must be added to the switch's VC-number translation table; and each time a connection is released, an entry must be removed from the table. Note that even if there is no VC-number translation, it is still necessary to maintain state information that associates VC numbers to interface numbers. The issue of whether or not a switch or router maintains state information for each ongoing connection is a crucial one—one that we return to shortly below.

Datagram Networks

Datagram networks are analogous in many respects to the postal services. When a sender sends a letter to a destination, the sender wraps the letter in an envelope and writes the destination address on the envelope. This destination address has a hierarchical structure. For example, letters sent to a location in the United States include the country (USA), the state (for example, Pennsylvania), the city (for example, Philadelphia), the street (for example, Walnut Street) and the number of the house on the street (for example, 421). The postal services use the address on the envelope to route the letter to its destination. For example, if the letter is sent from France, then a postal office in France will first direct the letter to a postal center in the United States. This postal center in the United States will then send the letter to a postal center in Philadelphia. Finally, a mail person working in Philadelphia will deliver the letter to its ultimate destination.

In a datagram network, each packet that traverses the network contains in its header the address of the destination. As with postal addresses, this address has a hierarchical structure. When a packet arrives at a packet switch in the network, the packet switch examines a portion of the packet's destination address and forwards the packet to an adjacent switch. More specifically, each packet switch has a routing table that maps destination addresses (or portions of the destination addresses) to an outbound link. When a packet arrives at a switch, the switch examines the address and indexes its table with this address to find the appropriate outbound link. The switch then sends the packet into this outbound link.

The whole routing process is also analogous to the car driver who does not use maps but instead prefers to ask for directions. For example, suppose Joe is driving from Philadelphia to 156 Lakeside Drive in Orlando, Florida. Joe first drives to his neighborhood gas station and asks how to get to 156 Lakeside Drive in Orlando, Florida. The gas station attendant extracts the Florida portion of the address and tells Joe that he needs to get onto the interstate highway I-95 South, which has an entrance just next to the gas station. He also tells Joe that once he enters Florida he should ask someone else there. Joe then takes I-95 South until he gets to Jacksonville, Florida, at which point he asks another gas station attendant for directions. The attendant extracts the Orlando portion of the address and tells Joe that he should continue on I-95 to Daytona Beach and then ask someone else. In Daytona Beach another gas station attendant also extracts the Orlando portion of the address and tells Joe that he should take I-4 directly to Orlando. Joe takes I-4 and gets off at the Orlando exit. Joe goes to another gas station attendant, and this time the attendant extracts the Lakeside Drive portion of the address and tells Joe the road he must follow to get to Lakeside Drive. Once Joe reaches Lakeside Drive he asks a kid on a bicycle how to get to his destination. The kid extracts the 156 portion of the address and points to the house. Joe finally reaches his ultimate destination.

We will be discussing routing in datagram networks in great detail in this book. But for now we mention that, in contrast with VC networks, *datagram networks do not maintain connection-state information in their switches.* In fact, a switch in a pure datagram network is completely oblivious to any flows of traffic that may be passing through it—it makes routing decisions for each individual packet. Because VC networks must maintain connection-state information in their switches, opponents of VC networks argue that VC networks are overly complex. These opponents include most researchers and engineers in the Internet community. Proponents of VC networks feel that VCs can offer applications a wider variety of networking services.

How would you like to actually see the route that packets take in the Internet? We now invite you to get your hands dirty by interacting with the Traceroute program, using the interface (http://www.awl.com/kurose-ross) provided on the Web site for this book.

Network Taxonomy

We have now introduced several important networking concepts: circuit switching, packet switching, message switching, virtual circuits, connectionless service, and connection-oriented service. How does it all fit together?

First, in our simple view of the world, a telecommunications network either employs circuit switching or packet switching (see Figure 1.13). A link in a circuit-switched network can employ either FDM or TDM (see Figure 1.14). Packet-switched networks are either virtual circuit networks or datagram networks.

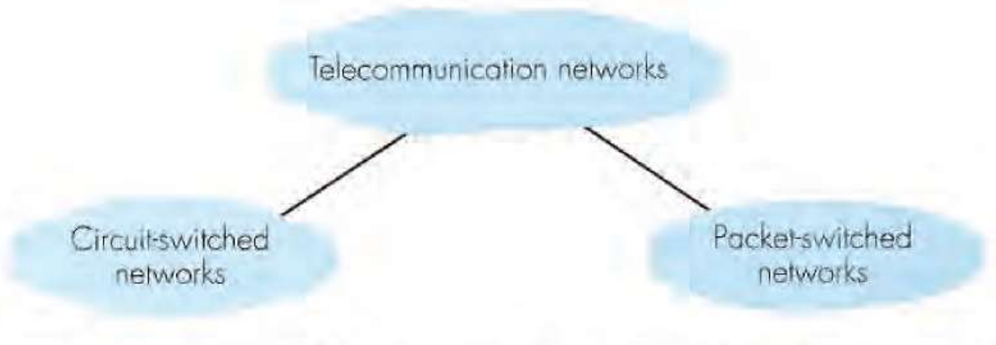


Figure 1.13 ◆ Highest-level distinction among telecommunication networks: Circuit-switched or packet-switched?

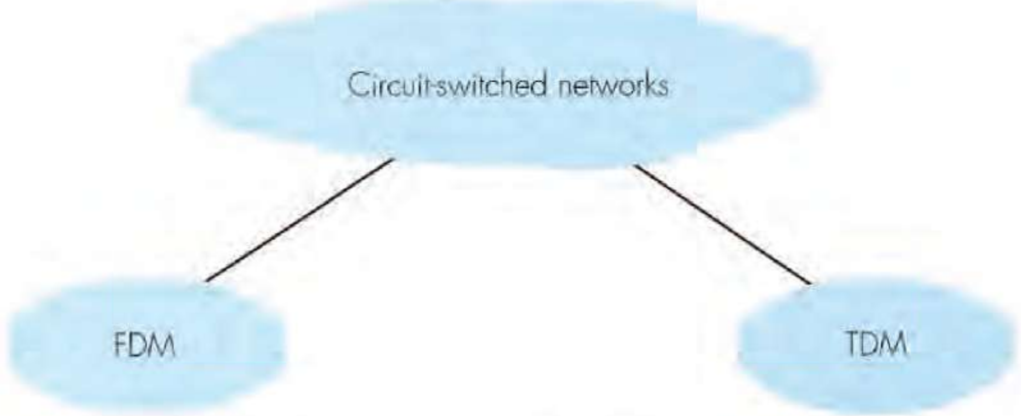


Figure 1.14 ◆ Circuit-switching implementation: FDM or TDM?

Switches in virtual circuit networks route packets according to the packets' VC numbers and maintain connection state. Switches in datagram networks route packets according to the packets' destination addresses and do not maintain connection state (see Figure 1.15).

Examples of packet-switched networks that use VCs include X.25, frame relay, and ATM. A packet-switched network either (1) uses VCs for all of its message routing, or (2) uses destination addresses for all of its message routing. It doesn't employ both routing techniques. (This last statement is a bit of a white lie, as there are networks that use datagram routing "on top of" VC routing. This is the case for "IP over ATM," as we shall cover later in the book.)

A datagram network is *not*, however, either a connectionless or a connection-oriented network. Indeed, a datagram network can provide the connectionless service to some of its applications and the connection-oriented service to other applications.

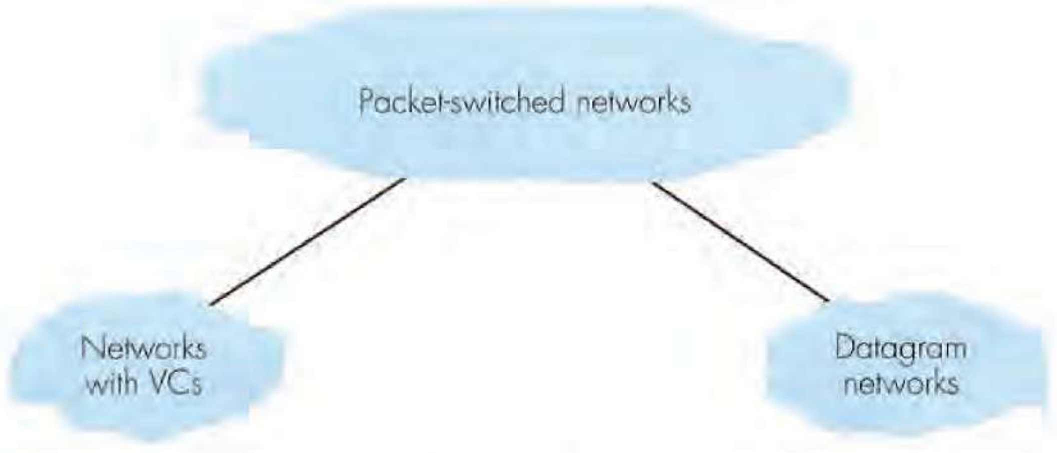


Figure 1.15 ✦ Packet-switching implementation: Virtual circuits or datagrams?

For example, the Internet, which is a datagram network, provides both connectionless and connection-oriented service to its applications. We saw in Section 1.3 that these services are provided in the Internet by the UDP and TCP protocols, respectively. Networks with VCs—such as X.25, Frame Relay, and ATM—are always, however, connection-oriented.

1.5 ✦ Access Networks and Physical Media

In Sections 1.3 and 1.4 we have examined the roles of end systems and routers in a network architecture. In this section we consider the **access network**—the physical link(s) that connect an end system to its **edge router**—that is, to the first router on a path from the end system to any other distant end system. Since access network technology is closely tied to physical media technology (fiber, coaxial pair, twisted-pair telephone wire, radio spectrum), we consider these two topics together in this section.

1.5.1 Access Networks

Figure 1.16 shows the access networks' links highlighted in thick, shaded lines.

Access networks can be loosely divided into three categories:

- **Residential access networks,** connecting a home end system into the network
- **Institutional access networks,** connecting an end system in a business or educational institution into the network
- **Mobile access networks,** connecting a mobile end system into the network

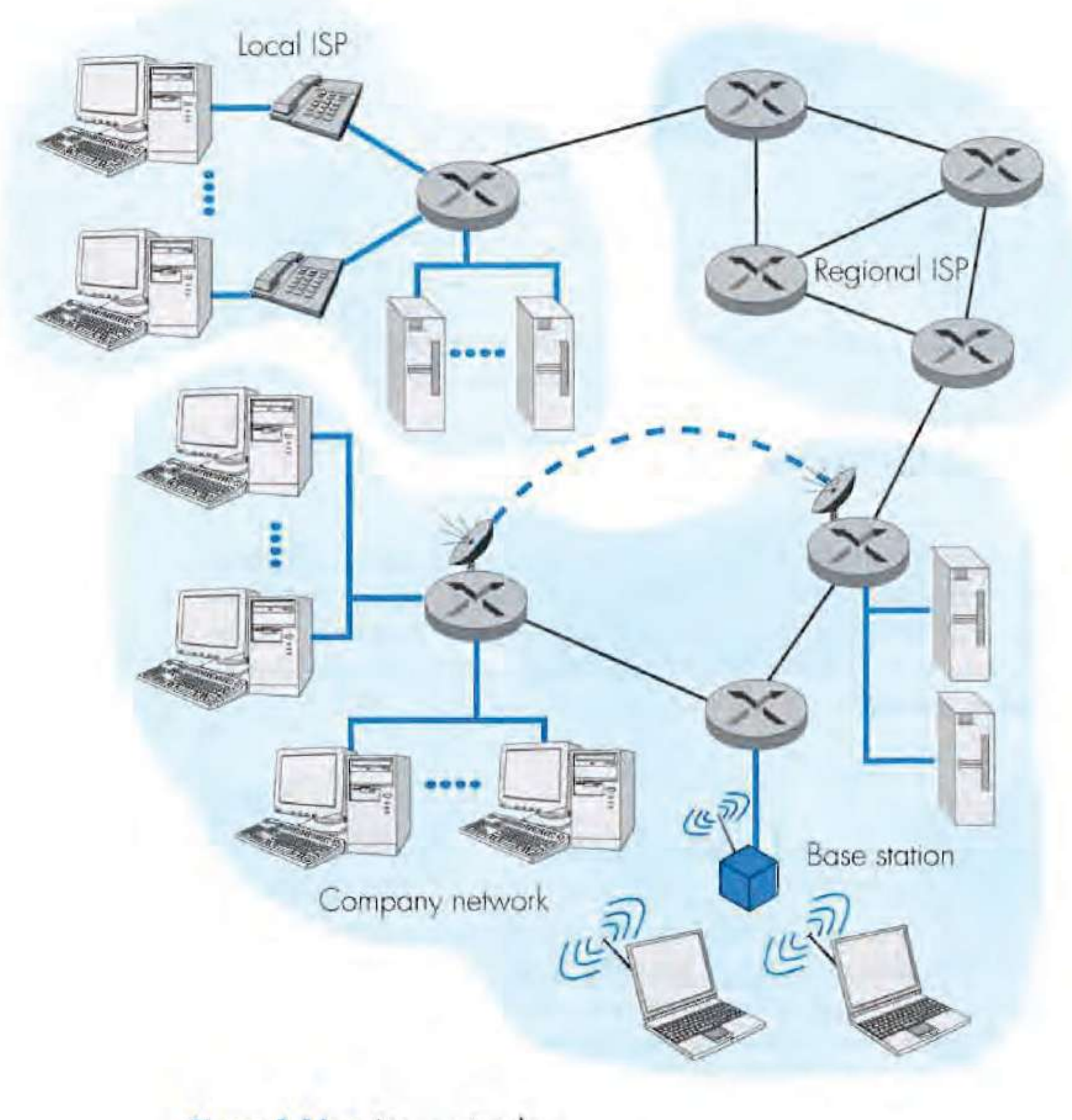


Figure 1.16 ◆ Access networks

These categories are not hard and fast; some corporate end systems may well use the access network technology that we ascribe to residential access networks, and vice versa. The following descriptions are meant to hold for the common (if not every) case.

Residential Access Networks

A residential access network connects a home end system (typically a PC, but perhaps a Web TV or other residential system) to an edge router. Probably the most common form of home access is by use of a **modem** over a POTS (plain old telephone system) dialup line to an Internet service provider (ISP). The home modem converts the digital output of the PC into analog format for transmission over the analog phone line. A modem in the ISP converts the analog signal back into digital form for input to the ISP router. In this case, the access network is simply a point-to-point dialup link into an edge router. The point-to-point link is your ordinary twisted-pair phone line. (We will discuss twisted pair later in this section.) Today's modem speeds allow dialup access at rates up to 56 Kbps. However, due to the poor quality of twisted-pair line between many homes and ISPs, many users get an effective rate significantly less than 56 Kbps.

Whereas dialup modems require conversion of the end system's digital data into analog form for transmission, so-called narrowband **ISDN** technology (Integrated Services Digital Network) [Pacific Bell 1998] allows for all-digital transmission of data from a home end system over ISDN "telephone" lines to a phone company central office. Although ISDN was originally conceived as a way to carry digital data from one end of the phone system to another, it is also an important network access technology that provides higher speed access (for example, 128 Kbps) from the home into a data network such as the Internet. In this case, ISDN can be thought of simply as a "better modem" [NAS 1995]. A good source for additional Web information on ISDN is Dan Kegel's ISDN page [Kegel 1999].

Dialup modems and narrowband ISDN are already widely deployed technologies. Two new technologies, **asymmetric digital subscriber line (ADSL)** [ADSL 1998] and **hybrid fiber coaxial cable (HFC)** [Cable 1998] are currently being deployed. ADSL is conceptually similar to dialup modems: It is a new modem technology again running over existing twisted-pair telephone lines, but it can transmit at rates of up to about 8 Mbps from the ISP router to a home end system. The data rate in the reverse direction, from the home end system to the central office router, is less than 1 Mbps. The asymmetry in the access speeds gives rise to the term *asymmetric* in ADSL. The asymmetry in the data rates reflects the belief that a home user is more likely to be a consumer of information (bringing data into the home) than a producer of information.

ADSL uses frequency division multiplexing, as described in the previous section. In particular, ADSL divides the communication link between the home and the ISP into three nonoverlapping frequency bands:

- A high-speed downstream channel, in the 50 kHz to 1 MHz band
- A medium-speed upstream channel, in the 4 kHz to 50 kHz band
- An ordinary POTS two-way telephone channel, in the 0 to 4 KHz band

One of the features of ADSL is that the service allows the user to make an ordinary telephone call, using the POTS channel, while simultaneously surfing the Web. This feature is not available with standard dialup modems. The actual amount of downstream and upstream bandwidth available to the user is a function of the distance

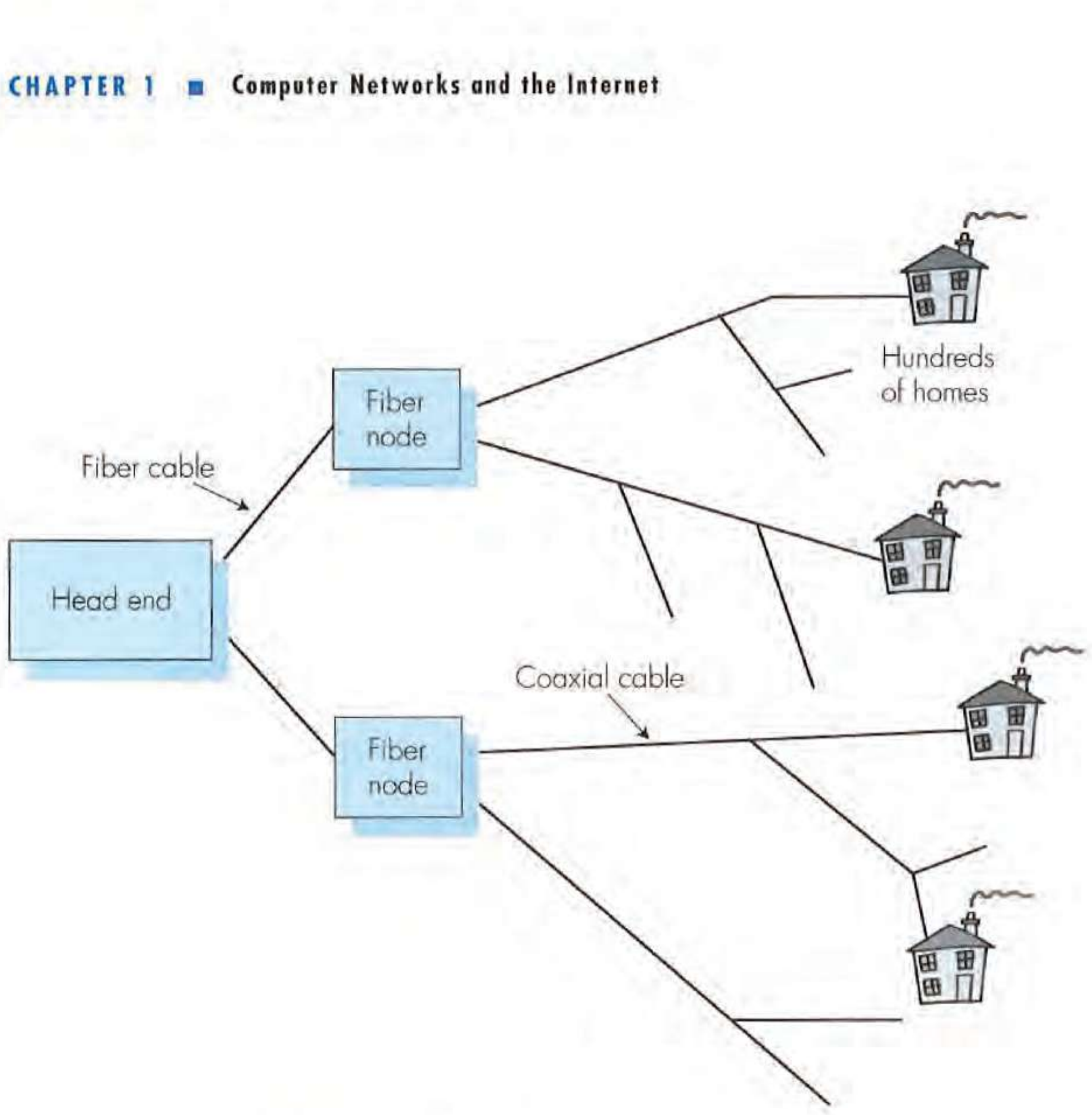


Figure 1.17 ♦ A hybrid fiber-coax access network

between the home modem and the ISP modem, the gauge of the twisted-pair line, and the degree of electrical interference. For a high-quality line with negligible electrical interference, an 8 Mbps downstream transmission rate is possible if the distance between the home and the ISP is less than 3,000 meters; the downstream transmission rate drops to about 2 Mbps for a distance of 6,000 meters. The upstream rate ranges from 16 Kbps to 1 Mbps.

ADSL, ISDN, and dialup modems all use ordinary phone lines, but HFC access networks are extensions of the current cable network used for broadcasting cable television. In a traditional cable system, a cable head end station broadcasts through a distribution of coaxial cable and amplifiers to residences. (We discuss coaxial cable later in this chapter.) As illustrated in Figure 1.17, fiber optics (also to be discussed soon) connect the cable head end to neighborhood-level junctions, from which traditional coaxial cable is then used to reach individual houses and apartments. Each neighborhood juncture typically supports 500 to 5,000 homes.

As with ADSL, HFC requires special modems, called cable modems. Companies that provide cable Internet access require their customers to either purchase or lease a modem. One such company is Cyber Cable, which uses Motorola's Cyber Surfer Cable Modem and provides high-speed Internet access to most of the neighborhoods in Paris. Typically, the cable modem is an external device and connects to the home PC through a 10-BaseT Ethernet port. (We will discuss Ethernet in great detail in Chapter 5.) Cable modems divide the HFC network into two channels, a downstream and an upstream channel. As with ADSL, the downstream channel is typically allocated more bandwidth and hence a larger transmission rate. For example, the downstream rate of the Cyber Cable system is 10 Mbps and the upstream rate is 768 Kbps. However, with HFC (and not with ADSL), these rates are shared among the homes, as we discuss next.

One important characteristic of HFC is that it is a shared broadcast medium. In particular, every packet sent by the head end travels downstream on every link to every home; and every packet sent by a home travels on the upstream channel to the head end. For this reason, if several users are receiving different Internet videos on the downstream channel, the actual rate at which each user receives its video will be significantly less than the downstream rate. On the other hand, if all the active users are Web surfing, then each of the users may actually receive Web pages at the full downstream rate, as a small collection of users will rarely request a Web page at exactly the same time. Because the upstream channel is also shared, packets sent by two different homes at the same time will collide, which further decreases the effective upstream bandwidth. (We will discuss this collision issue in some detail when we discuss Ethernet in Chapter 5.) Advocates of ADSL are quick to point out that ADSL is a point-to-point connection between the home and ISP, and therefore all the ADSL bandwidth is dedicated rather than shared. Cable advocates, however, argue that a reasonably dimensioned HFC network provides higher bandwidths than ADSL [@Home 1998]. The battle between ADSL and HFC for high speed residential access has clearly begun; see for example, [@Home 1998].

Company Access Networks

In company access networks, a local area network (LAN) is used to connect an end system to an edge router. As we will see in Chapter 5, there are many different types of LAN technology. However, Ethernet technology is currently by far the most prevalent access technology in company networks. Ethernet operates at 10 Mbps or 100 Mbps (and now even at 1 Gbps). It uses either twisted-pair copper wire or coaxial cable to connect a number of end systems with each other and with an edge router. The edge router is responsible for routing packets that have destinations outside of that LAN. Like HFC, Ethernet uses a shared medium, so that end users share the transmission rate of the LAN. More recently, shared Ethernet technology has been migrating towards switched Ethernet technology. Switched Ethernet uses multiple coaxial cable or twisted-pair Ethernet segments connected at a "switch" to allow the full bandwidth of an Ethernet to be delivered to different users on the same LAN simultaneously [Cisco LAN 1998]. We will explore shared and switched Ethernet in some detail in Chapter 5.

Mobile Access Networks

Mobile access networks use the radio spectrum to connect a mobile end system (for example, a laptop PC or a PDA with a wireless modem) to a base station. This base station, in turn, is connected to an edge router of a data network.

An emerging standard for wireless data networking is **cellular digital packet data (CDPD)** [Wireless 1998]. As the name suggests, a CDPD network operates as an overlay network (that is, as a separate, smaller virtual network, as a piece of the larger network) within the cellular telephone network. A CDPD network thus uses the same radio spectrum as the cellular phone system, and operates at speeds in the tens of Kbits per second. As with cable-based access networks and shared Ethernet, CDPD end systems must share the transmission media with other CDPD end systems within the cell covered by a base station. A media access control (MAC) protocol is used to arbitrate channel sharing among the CDPD end systems; we will cover MAC protocols in detail in Chapter 5.

The CDPD system supports the IP protocol and thus allows an IP end system to exchange IP packets over the wireless channel with an IP base station. CDPD does not provide for any protocols above the network layer. From an Internet perspective, CDPD can be viewed as extending the Internet dialtone (that is, the ability to transfer IP packets) across a wireless link between a mobile end system and an Internet router. An excellent introduction to CDPD is [Waung 1998].

1.5.2 Physical Media

In the previous subsection, we gave an overview of some of the most important access network technologies in the Internet. As we described these technologies, we also indicated the physical media used. For example, we said that HFC uses a combination of fiber cable and coaxial cable. We said that ordinary modems, ISDN, and ADSL use twisted-pair copper wire. And we said that mobile access networks use the radio spectrum. In this subsection we provide a brief overview of these and other transmission media that are commonly employed in the Internet.

In order to define what is meant by a physical medium, let us reflect on the brief life of a bit. Consider a bit traveling from one end system, through a series of links and routers, to another end system. This poor bit gets transmitted many, many times! The source end system first transmits the bit, and shortly thereafter the first router in the series receives the bit; the first router then transmits the bit, and shortly afterwards the second router receives the bit, and so on. Thus our bit, when traveling from source to destination, passes through a series of transmitter–receiver pairs. For each transmitter–receiver pair, the bit is sent by propagating electromagnetic waves or optical pulses across a **physical medium.** The physical medium can take many shapes and forms and does not have to be of the same type for each transmitter–receiver pair along the path. Examples of physical media include twisted-pair copper wire, coaxial cable, multimode fiber-optic cable, terrestrial radio spectrum, and satellite radio spectrum. Physical media fall into two categories: **guided media** and **unguided media.** With guided media, the waves are guided along a solid medium, such as a fiber-optic cable, a twisted-pair copper wire, or a coaxial cable. With unguided media, the waves propagate in the atmosphere and in outer space, such as in a digital satellite channel or in a CDPD system.

Some Popular Physical Media

Suppose you want to wire a building to allow computers to access the Internet or an intranet. Should you use twisted-pair copper wire, coaxial cable, or fiber optics? Which of these media gives the highest bit rates over the longest distances? We shall address these questions below.

But before we get into the characteristics of the various guided medium types, let us say a few words about their costs. The actual cost of the physical link (copper wire, fiber-optic cable, and so on) is often relatively minor compared with the other networking costs. In particular, the labor cost associated with the installation of the physical link can be orders of magnitude higher than the cost of the material. For this reason, many builders install twisted pair, optical fiber, and coaxial cable to every room in a building. Even if only one medium is initially used, there is a good chance that another medium could be used in the near future, and so money is saved by not having to lay additional wires.

Twisted-Pair Copper Wire. The least-expensive and most commonly used transmission medium is twisted-pair copper wire. For over one-hundred years it has been used by telephone networks. In fact, more than 99 percent of the wired connections from the telephone handset to the local telephone switch use twisted-pair copper wire. Most of us have seen twisted-pair in our homes and work environments. Twisted pair consists of two insulated copper wires, each about 1 mm thick, arranged in a regular spiral pattern (see Figure 1.18). The wires are twisted together to reduce the electrical interference from similar pairs close by. Typically, a number of pairs are bundled together in a cable by wrapping the pairs in a protective shield. A wire pair constitutes a single communication link.

Figure 1.18 ✦ Twisted pair

Unshielded twisted pair (UTP) is commonly used for computer networks within a building, that is, for local area networks (LANs). Data rates for LANs using twisted pair today range from 10 Mbps to 100 Mbps. The data rates that can be achieved depend on the thickness of the wire and the distance between transmitter and receiver. Two types of UTP are common in LANs: category 3 and category 5. Category 3 corresponds to voice-grade twisted pair, commonly found in office buildings. Office buildings are often prewired with two or more parallel pairs of category 3 twisted pair; one pair is used for telephone communication, and the additional pairs can be used for additional telephone lines or for LAN networking. 10 Mbps Ethernet, one of the most prevalent LAN types, can use category 3 UTP. Category 5, with its more twists per centimeter and Teflon™ insulation, can handle higher bit rates. 100 Mbps Ethernet running on category 5 UTP has become very popular in recent years. In recent years, category 5 UTP has become common for preinstallation in new office buildings.

When fiber-optic technology emerged in the 1980s, many people disparaged twisted pair because of its relatively low bit rates. Some people even felt that fiber-optic technology would completely replace twisted pair. But twisted pair did not give up so easily. Modern twisted-pair technology, such as category 5 UTP, can achieve data rates of 100 Mbps for distances up to a few hundred meters. Even higher rates are possible over shorter distances. In the end, twisted pair has emerged as the dominant solution for high-speed LAN networking.

As discussed in the section on access networks, twisted pair is also commonly used for residential Internet access. We saw that dialup modem technology enables access at rates of up to 56 Kbps over twisted pair. We also saw that ISDN is available in many communities, providing access rates of about 128 Kbps over twisted pair. We also saw that ADSL (asymmetric digital subscriber loop) technology has enabled residential users to access the Internet at rates in excess of 6 Mbps over twisted pair.

Coaxial Cable. Like twisted pair, coaxial cable consists of two copper conductors, but the two conductors are concentric rather than parallel. With this construction and a special insulation and shielding, coaxial cable can have higher bit rates than twisted pair. Coaxial cable comes in two varieties: **baseband coaxial cable** and **broadband coaxial cable.**

Baseband coaxial cable, also called 50-ohm cable, is about a centimeter thick, lightweight, and easy to bend. It is commonly used in LANs; in fact, the computer you use at work or at school is probably connected to a LAN with either baseband coaxial cable or with UTP. Take a look at the connection to your computer's interface card. If you see a telephone-like jack and some wire that resembles telephone wire, you are using UTP; if you see a T-connector and a cable running out of both sides of the T-connector, you are using baseband coaxial cable. The term *baseband* comes from the fact that the stream of bits is dumped directly into the cable, without shifting the signal to a different frequency band. 10 Mbps Ethernets can use either UTP or baseband coaxial cable. As we will discuss in Chapter 5, it is a little more expensive to use UTP for 10 Mbps Ethernet, as UTP requires an additional networking device, called a **hub.**

Broadband coaxial cable, also called 75-ohm cable, is quite a bit thicker, heavier, and stiffer than the baseband variety. It was once commonly used in LANs and can still be found in some older installations. For LANs, baseband cable is now preferable since it is less expensive, easier to physically handle, and does not require attachment cables. Broadband cable, however, is quite common in cable television systems. As we saw earlier, cable television systems have recently been coupled with cable modems to provide residential users with Web access at rates of 10 Mbps or higher. With broadband coaxial cable, the transmitter shifts the digital signal to a specific frequency band, and the resulting analog signal is sent from the transmitter to one or more receivers. Both baseband and broadband coaxial cable can be used as a guided **shared medium.** Specifically, a number of end systems can be connected directly to the cable, and all the end systems receive whatever any one of the computers transmits. We will look at this issue in more detail in Chapter 5.

Fiber Optics. An optical fiber is a thin, flexible medium that conducts pulses of light, with each pulse representing a bit. A single optical fiber can support tremendous bit rates, up to tens or even hundreds of gigabits per second. They are immune to electromagnetic interference, have very low signal attenuation up to 100 kilometers, and are very hard to tap. These characteristics have made fiber optics the preferred long-haul guided transmission media, particularly for overseas links. Many of the long-distance telephone networks in the United States and elsewhere now use fiber optics exclusively. Fiber optics is also prevalent in the backbone of the Internet. However, the high cost of optical devices—such as transmitters, receivers, and switches—has hindered their deployment for short-haul transport, such as in a LAN or into the home in a residential access network. AT&T Labs provides a site [AT&T Optics 1999] on fiber optics, including several nice animations. [Ramaswami 1998] and [Green 1992] provide in-depth coverage of optical networks.

Terrestrial and Satellite Radio Channels. Radio channels carry signals in the electromagnetic spectrum. They are an attractive media because they require no physical wire to be installed, can penetrate walls, provide connectivity to a mobile user, and can potentially carry a signal for long distances. The characteristics of a radio channel depend significantly on the propagation environment and the distance over which a signal is to be carried. Environmental considerations determine path loss and shadow fading (which decrease in signal strength as the signal travels over a distance and around/through obstructing objects), multipath fading (due to signal reflection off of interfering objects), and interference (due to other radio channels or electromagnetic signals).

Terrestrial radio channels can be broadly classified into two groups: those that operate as local area networks (typically spanning from ten to a few hundred meters) and wide-area radio channels that are used for mobile data services (typically operating within a metropolitan region). A number of wireless LAN products are on the market, operating in the range of from one to tens of Mbps. Mobile data services (such as the CDPD standard we touched on in Section 1.5.1), typically provide channels that operate at tens of Kbps. See [Goodman 1997] for a survey and discussion of the technology and products.

A communication satellite links two or more Earth-based microwave transmitter/receivers, known as ground stations. The satellite receives transmissions on one frequency band, regenerates the signal using a repeater (discussed below), and transmits the signal on another frequency. Satellites can provide bandwidths in the gigabit per second range. Two types of satellites are used in communications: geostationary satellites and low-altitude satellites.

Geostationary satellites permanently remain above the same spot on Earth. This stationary presence is achieved by placing the satellite in orbit at 36,000 kilometers above Earth's surface. This huge distance from ground station through satellite back to ground station introduces a substantial signal propagation delay of 250 milliseconds. Nevertheless, satellite links, which can operate at speeds of hundreds of Mbps, are often used in telephone networks and in the backbone of the Internet.

Low-altitude satellites are placed much closer to Earth and do not remain permanently above one spot on Earth. They rotate around Earth just as the Moon does. To provide continuous coverage to an area, many satellites need to be placed in orbit. There are currently many low-altitude communication systems in development. The Iridium system, for example, consists of 66 low-altitude satellites. Lloyd's satellite constellation systems Web page [Wood 1999] provides and collects information on Iridium as well as other satellite constellation systems. The low-altitude satellite technology may be used for Internet access sometime in the future.

1.6 ✦ Delay and Loss in Packet-Switched Networks

Having now briefly considered the major pieces of the Internet architecture—the applications, end systems, end-to-end transport protocols, routers, and links—let us now consider what can happen to a packet as it travels from its source to its destination. Recall that a packet starts in a host (the source), passes through a series of routers, and ends its journey in another host (the destination). As a packet travels from one node (host or router) to the subsequent node (host or router) along this path, the packet suffers from several different types of delays at *each* node along the path. The most important of these delays are the **nodal processing delay, queuing delay, transmission delay,** and **propagation delay;** together, these delays accumulate to give a **total nodal delay.** In order to acquire a deep understanding of packet switching and computer networks, we must understand the nature and importance of these delays.

1.6.1 Types of Delay

Let us explore these delays in the context of Figure 1.19. As part of its end-to-end route between source and destination, a packet is sent from the upstream node through router A to router B. Our goal is to characterize the nodal delay at router A.

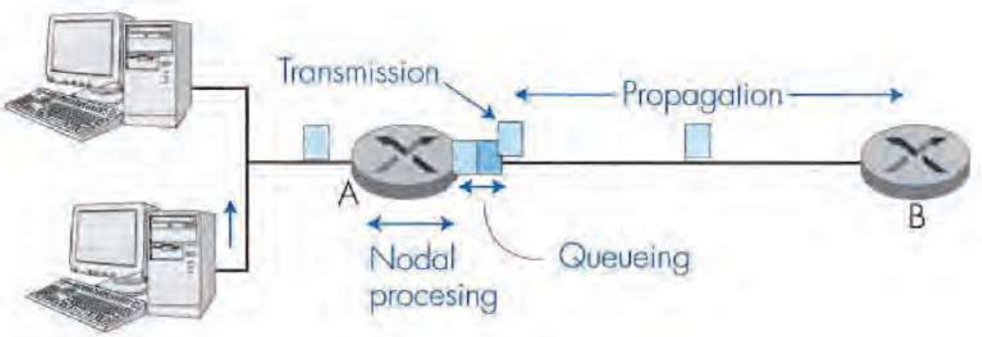


Figure 1.19 ✦ The delay through router A

Note that router A has an outbound link leading to router B. This link is preceded by a queue (also known as a buffer). When the packet arrives at router A from the upstream node, router A examines the packet's header to determine the appropriate outbound link for the packet, and then directs the packet to the link. In this example, the outbound link for the packet is the one that leads to router B. A packet can be transmitted on a link only if there is no other packet currently being transmitted on the link and if there are no other packets preceding it in the queue; if the link is currently busy or if there are other packets already queued for the link, the newly arriving packet will then join the queue.

Processing Delay

The time required to examine the packet's header and determine where to direct the packet is part of the **processing delay.** The processing delay can also include other factors, such as the time needed to check for bit-level errors in the packet that occurred in transmitting the packet's bits from the upstream router to router A. Processing delays in high-speed routers are typically on the order of microseconds or less. After this nodal processing, the router directs the packet to the queue that precedes the link to router B. (In Section 4.6 we will study the details of how a router operates.)

Queuing Delay

At the queue, the packet experiences a **queuing delay** as it waits to be transmitted onto the link. The queuing delay of a specific packet will depend on the number of other, earlier-arriving packets that are queued and waiting for transmission across the link. The delay of a given packet can vary significantly from packet to packet. If the queue is empty and no other packet is currently being transmitted, then our packet's queuing delay is zero. On the other hand, if the traffic is heavy and many other packets are also waiting to be transmitted, the queuing delay will be long. We will see shortly that the number of packets that an arriving packet might expect to find on arrival is a function of the intensity and nature of the traffic arriving to the queue. Queuing delays can be on the order of milliseconds to microseconds in practice.

Transmission Delay

Assuming that packets are transmitted in first-come-first-serve manner, as is common in the Internet, our packet can be transmitted once all the packets that have arrived before it have been transmitted. Denote the length of the packet by L bits, and denote the transmission rate of the link from router A to router B by R bits/sec. The rate R is determined by transmission rate of the link to router B. For example, for a 10-Mbps Ethernet link, the rate is $R = 10$ Mbps; for a 100-Mbps Ethernet link, the rate is $R = 100$ Mbps. The **transmission delay** (also called the store-and-forward delay, as discussed in Section 1.4) is L/R. This is the amount of time required to transmit all of the packet's bits into the link. Transmission delays are typically on the order of microseconds or less in practice.

Propagation Delay

Once a bit is pushed onto the link, it needs to propagate to router B. The time required to propagate from the beginning of the link to router B is the **propagation delay.** The bit propagates at the propagation speed of the link. The propagation speed depends on the physical medium of the link (that is, multimode fiber, twisted-pair copper wire, and so on) and is in the range of

$$2 \cdot 10^8 \text{ meters/sec to } 3 \cdot 10^8 \text{ meters/sec}$$

which is equal to, or a little less than, the speed of light. The propagation delay is the distance between two routers divided by the propagation speed. That is, the propagation delay is d/s, where d is the distance between router A and router B and s is the propagation speed of the link. Once the last bit of the packet propagates to node B, it and all the preceding bits of the packet are stored in router B. The whole process then continues with router B now performing the forwarding. In wide-area networks, propagation delays are on the order of milliseconds.

Comparing Transmission and Propagation Delay

Newcomers to the field of computer networking sometimes have difficulty understanding the difference between transmission delay and propagation delay. The difference is subtle but important. The transmission delay is the amount of time required for the router to push out the packet; it is a function of the packet's length and the transmission rate of the link, but has nothing to do with the distance between the two routers. The propagation delay, on the other hand, is the time it takes a bit to propagate from one router to the next; it is a function of the distance between the two routers, but has nothing to do with the packet's length or the transmission rate of the link.

An analogy might clarify the notions of transmission and propagation delay. Consider a highway that has a toll booth every 100 kilometers. You can think of the highway segments between toll booths as links and the toll booths as routers. Suppose that cars travel (that is, propagate) on the highway at a rate of 100 km/ hour (that is, when a car leaves a toll booth, it instantaneously accelerates to 100 km/hour and maintains that speed between toll booths). Suppose that there is a caravan of 10 cars that are traveling together, and that these 10 cars follow each other in a fixed order. You can think of each car as a bit and the caravan as a packet. Also suppose that each toll booth services (that is, transmits) a car at a rate of one car per 12 seconds, and that it is late at night so that the caravan's cars are the only cars on the highway. Finally, suppose that whenever the first car of the caravan arrives at a toll booth, it waits at the entrance until the nine other cars have arrived and lined up behind it. (Thus the entire caravan must be "stored" at the toll booth before it can begin to be "forwarded.") The time required for the toll booth to push the entire caravan onto the highway is 10/(5 cars/minute) = 2 minutes. This time is analogous to the transmission delay in a router. The time required for a car to travel from the exit of one toll booth to the next toll booth is 100 km/(100 km/hour) = 1 hour. This time is analogous to propagation delay. Therefore, the time from when the caravan is "stored" in front of a toll booth until the caravan is "stored" in front of the next toll booth is the sum of "transmission delay" and "the propagation delay"—in this example, 62 minutes.

Let's explore this analogy a bit more. What would happen if the toll-booth service time for a caravan were greater than the time for a car to travel between toll booths? For example, suppose cars travel at rate 1,000 km/hr and the toll booth services cars at rate one car per minute. Then the traveling delay between toll booths is 6 minutes and the time to serve a caravan is 10 minutes. In this case, the first few cars in the caravan will arrive at the second toll booth before the last cars in caravan leave the first toll booth. This situation also arises in packet-switched networks—the first bits in a packet can arrive at a router while many of the remaining bits in the packet are still waiting to be transmitted by the preceding router.

If we let d_{proc}, d_{queue}, d_{trans}, and d_{prop} denote the processing, queuing, transmission, and propagation delays, then the total nodal delay is given by

$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

The contribution of these delay components can vary significantly. For example, d_{prop} can be negligible (for example, a couple of microseconds) for a link connecting two routers on the same university campus; however, d_{prop} is hundreds of milliseconds for two routers interconnected by a geostationary satellite link, and can be the dominant term in d_{nodal}. Similarly, d_{trans} can range from negligible to significant. Its contribution is typically negligible for transmission rates of 10 Mbps and higher (for example, for LANs); however, it can be hundreds of milliseconds for large Internet packets sent over 28.8 Kbps modem links. The processing delay, d_{proc}, is often negligible; however, it strongly influences a router's maximum throughput, which is the maximum rate at which a router can forward packets.

Queuing Delay

The most complicated and interesting component of nodal delay is the queuing delay, d_{queue}. In fact, queuing delay is so important and interesting in computer networking that thousands of papers and numerous books have been written about it [Bertsekas 1991; Daigle 1991; Kleinrock 1975, 1976; Ross 1995]! We give only a high-level, intuitive discussion of queuing delay here; the more curious reader may want to browse through some of the books (or even eventually write a Ph.D. thesis on the subject!). Unlike the other three delays (namely, d_{proc}, d_{trans}, and d_{prop}), the queuing delay can vary from packet to packet. For example, if 10 packets arrive at an empty queue at the same time, the first packet transmitted will suffer no queuing delay, while the last packet transmitted will suffer a relatively large queuing delay (while it waits for the other nine packets to be transmitted). Therefore, when characterizing queuing delay, one typically uses statistical measures, such as average queuing delay, variance of queuing delay, and the probability that the queuing delay exceeds some specified value.

When is the queuing delay large and when is it insignificant? The answer to this question depends largely on the rate at which traffic arrives to the queue, the transmission rate of the link, and the nature of the arriving traffic, that is, whether the traffic arrives periodically or whether it arrives in bursts. To gain some insight here, let a denote the average rate at which packets arrive to the queue (a is in units of packets/sec). Recall that R is the transmission rate, that is, it is the rate (in bits/sec) at which bits are pushed out of the queue. Also suppose, for simplicity, that all packets consist of L bits. Then the average rate at which bits arrive to the queue is La bits/sec. Finally, assume that the queue is very big, so that it can hold essentially an infinite number of bits. The ratio La/R, called the **traffic intensity**, often plays an important role in estimating the extent of the queuing delay. If $La/R > 1$, then the average rate at which bits arrive to the queue exceeds the rate at which the bits can be transmitted from the queue. In this unfortunate situation, the queue will tend to increase without bound and the queuing delay will approach infinity! Therefore, one of the golden rules in traffic engineering is: *Design your system so that the traffic intensity is no greater than 1.*

Now consider the case $La/R \leq 1$. Here, the nature of the arriving traffic impacts the queuing delay. For example, if packets arrive periodically, that is, one packet arrives every L/R seconds, then every packet will arrive to an empty queue and there will be no queuing delay. On the other hand, if packets arrive in bursts but periodically, there can be a significant average queuing delay. For example, suppose

N packets arrive at the same time every $(L/R)N$ seconds. Then the first packet transmitted has no queuing delay; the second packet transmitted has a queuing delay of L/R seconds; and more generally, the nth packet transmitted has a queuing delay of $(n - 1)L/R$ seconds. We leave it as an exercise for the reader to calculate the average queuing delay in this example.

The two examples described above of periodic arrivals are a bit academic. Typically, the arrival process to a queue is *random*, that is, the arrivals do not follow any pattern; packets are spaced apart by random amounts of time. In this more realistic case, the quantity La/R is not usually sufficient to fully characterize the delay statistics. Nonetheless, it is useful in gaining an intuitive understanding of the extent of the queuing delay. In particular, if traffic intensity is close to zero, then packet arrivals are few and far between and it is unlikely that an arriving packet will find another packet in the queue. Hence, the average queuing delay will be close to zero. On the other hand, when the traffic intensity is close to 1, there will be intervals of time when the arrival rate exceeds the transmission capacity (due to the burstiness of arrivals), and a queue will form. As the traffic intensity approaches 1, the average queue length gets larger and larger. The qualitative dependence of average queuing delay on the traffic intensity is shown in Figure 1.20.

One important aspect of Figure 1.20 is the fact that as the traffic intensity approaches 1, the average queuing delay increases rapidly. A small percentage increase in the intensity will result in a much larger percentage-wise increase in delay. Perhaps you have experienced this phenomenon on the highway. If you regularly drive on a road that is typically congested, the fact that the road is typically congested means that its traffic intensity is close to 1. If some event causes an even slightly larger-than-usual amount of traffic, the delays you experience can be huge.

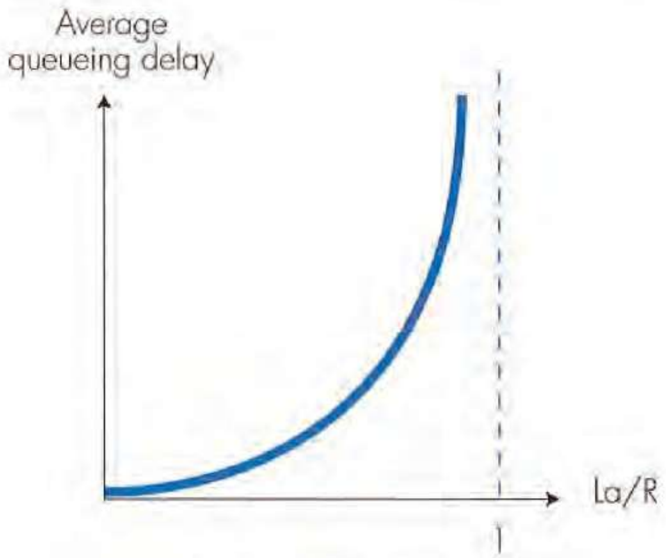


Figure 1.20 ♦ Dependence of average queuing delay on traffic intensity

Packet Loss

In our discussions above, we have assumed that the queue is capable of holding an infinite number of packets. In reality a queue preceding a link has finite capacity, although the queuing capacity greatly depends on the switch design and cost. Because the queue capacity is finite, packet delays do not really approach infinity as the traffic intensity approaches 1. Instead, a packet can arrive to find a full queue. With no place to store such a packet, a router will **drop** that packet; that is, the packet will be **lost.** From an end-system viewpoint, this will look like a packet having been transmitted into the network core, but never emerging from the network at the destination. The fraction of lost packets increases as the traffic intensity increases. Therefore, performance at a node is often measured not only in terms of delay, but also in terms of the probability of packet loss. As we shall discuss in the subsequent chapters, a lost packet may be retransmitted on an end-to-end basis, by either the application or by the transport layer protocol.

End-to-End Delay

Our discussion up to this point has been focused on the nodal delay, that is, the delay at a single router. Let us conclude our discussion by briefly considering the delay from source to destination. To get a handle on this concept, suppose there are $Q-1$ routers between the source host and the destination host. Let us also suppose that the network is uncongested (so that queuing delays are negligible), the processing delay at each router and at the source host is d_{proc}, the transmission rate out of each router and out of the source host is R bits/sec, and the propagation delay between each pair or routers and between the source host and the first router is d_{prop}. The nodal delays accumulate and give an end-to-end delay,

$$d_{\text{end-end}} = Q\,(d_{\text{proc}} + d_{\text{trans}} + d_{\text{prop}})$$

where, once again, $d_{\text{trans}} = L/R$, where L is the packet size. We leave it to the reader to generalize this formula to the case of heterogeneous delays at the nodes and to the presence of an average queuing delay at each node.

1.7 ✦ Protocol Layers and Their Service Models

From our discussion thus far, it is apparent that the Internet is an *extremely* complicated system. We have seen that there are many pieces to the Internet: numerous applications and protocols, various types of end systems and connections between end systems, routers, and various types of link-level media. Given this enormous complexity, is there any hope of organizing network architecture, or at least our discussion of network architecture? Fortunately, the answers to both questions is yes.

1.7.1 Layered Architecture

Before attempting to organize our thoughts on Internet architecture, let's look for a human analogy. Actually, we deal with complex systems all the time in our everyday life. Imagine if someone asked *you* to describe, for example, the airline system. How would you find the structure to describe this complex system that has ticketing agents, baggage checkers, gate personnel, pilots, airplanes, air traffic control, and a worldwide system for routing airplanes? One way to describe this system might be to describe the series of actions you take (or others take for you) when you fly on an airline. You purchase your ticket, check your bags, go to the gate, and eventually get loaded onto the plane. The plane takes off and is routed to its destination. After your plane lands, you de-plane at the gate and claim your bags. If the trip was bad, you complain about the flight to the ticket agent (getting nothing for your effort). This scenario is shown in Figure 1.21.

Already, we can see some analogies here with computer networking: You are being shipped from source to destination by the airline; a packet is shipped from source host to destination host in the Internet. But this is not quite the analogy we are after. We are looking for some *structure* in Figure 1.21. Looking at Figure 1.21, we note that there is a ticketing function at each end; there is also a baggage function for already-ticketed passengers, and a gate function for already-ticketed and already-baggage-checked passengers. For passengers who have made it through the gate (that is, passengers who are already ticketed, baggage-checked, and through the gate), there is a takeoff and landing function, and while in flight, there is an airplane routing function. This suggests that we can look at the functionality in Figure 1.21 in a *horizontal* manner, as shown in Figure 1.22.

Figure 1.22 has divided the airline functionality into layers, providing a framework in which we can discuss airline travel. Now, when we want to describe a part of airline travel, we can talk about a specific, well-defined component of airline

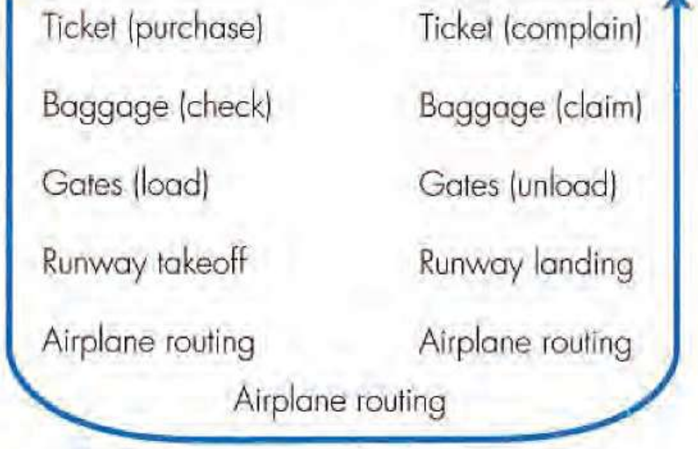


Figure 1.21 ♦ Taking an airplane trip: Actions

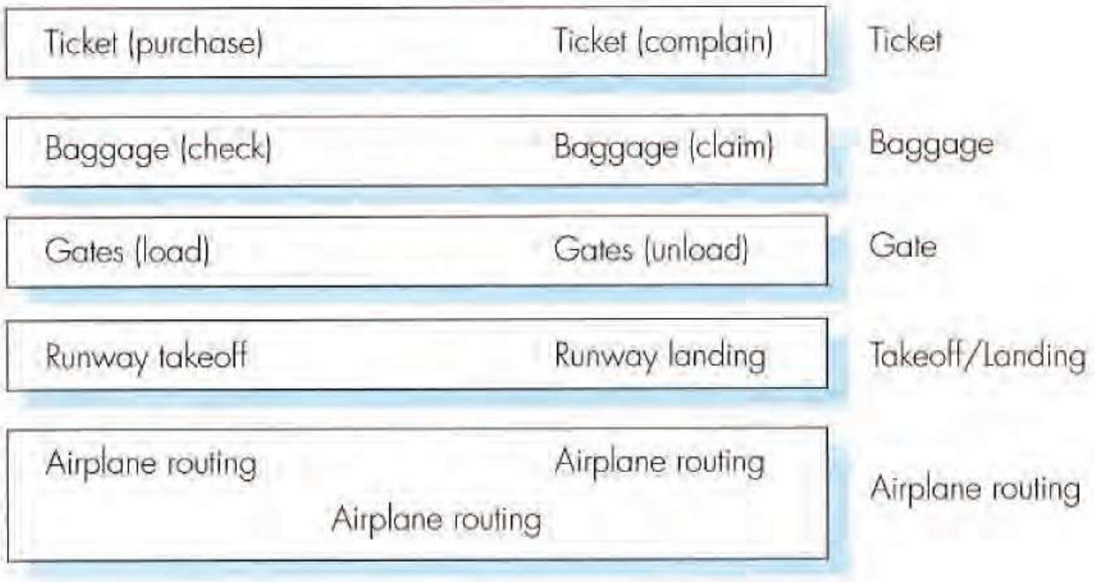


Figure 1.22 ✦ Horizontal layering of airline functionality

travel. For example, when we discuss gate functionality, we know we are discussing functionality that sits "below" baggage handling, and "above" takeoff and landing. We note that each layer, combined with the layers below it, implement some functionality, some *service*. At the ticketing layer and below, airline-counter-to-airline-counter transfer of a person is accomplished. At the baggage layer and below, baggage-check-to-baggage-claim transfer of a person and bags is accomplished. Note that the baggage layer provides this service only to an already-ticketed person. At the gate layer, departure-gate-to-arrival-gate transfer of a person and bags is accomplished. At the takeoff/landing layer, runway-to-runway transfer of people and their bags is accomplished. Each layer provides its service by (1) performing certain actions within that layer (for example, at the gate layer, loading and unloading people from an airplane) and by (2) using the services of the layer directly below it (for example, in the gate layer, using the runway-to-runway passenger transfer service of the takeoff/landing layer).

As noted above, a layered architecture allows us to discuss a well-defined, specific part of a large and complex system. This simplification itself is of considerable value. When a system has a layered structure it is also much easier to change the *implementation* of the service provided by the layer. As long as the layer provides the same service to the layer above it, and uses the same services from the layer below it, the remainder of the system remains unchanged when a layer's implementation is changed. (Note that changing the implementation of a service is very different from changing the service itself!) For example, if the gate functions were changed (for example, to have people board and disembark by height), the remainder of the airline system would remain unchanged since the gate layer still provides the same function (loading and unloading people); it simply implements that function in a different manner after the change. For large and complex systems that are constantly being updated, the ability to change the implementation of a service without affecting other components of the system is another important advantage of layering.

But enough with airlines. Let's now turn our attention to network protocols. To reduce design complexity, network designers organize protocols—and the network hardware and software that implements the protocols—in **layers.** With a layered protocol architecture, each protocol belongs to one of the layers. It's important to realize that a protocol in layer n is *distributed* among the network entities (including end systems and packet switches) that implement that protocol, just as the functions in our layered airline architecture were distributed between the departing and arriving airports. In other words, there's a piece of layer n in each of the network entities. These pieces communicate with each other by exchanging layer-n messages. These messages are called layer-n protocol data units, or more commonly ***n*-PDUs.** The contents and format of an n-PDU, as well as the manner in which the n-PDUs are exchanged among the network elements, are defined by a layer-n protocol. When taken together, the protocols of the various layers are called the **protocol stack.**

When layer n of Host A sends an n-PDU to layer n of Host B, layer n of Host A *passes* the n-PDU to layer $n-1$ and then lets layer $n-1$ deliver the n-PDU to layer n of B; thus layer n is said to *rely* on layer $n-1$ to deliver its n-PDU to the destination. A key concept is that of the **service model** of a layer. Layer $n-1$ is said to offer **services** to layer n. For example, layer $n-1$ might guarantee that the n-PDU will arrive without error at layer n in the destination within one second, or it might only guarantee that the n-PDU will eventually arrive at the destination without any assurances about error.

Protocol Layering

The concept of protocol layering is fairly abstract and is sometimes difficult to grasp at first. This concept will become clear as we study the Internet layers and their constituent protocols in greater detail. But let us now try to shed some insight on protocol layering and protocol stacks with an example. Consider a network that organizes its communication protocols in four layers. Because there are four layers, there are four types of PDUs: 1-PDUs, 2-PDUs, 3-PDUs, and 4-PDUs. As shown in Figure 1.23, the application, operating at the highest layer, layer 4, creates a message, M. Any message created at this highest layer is a 4-PDU. The message M itself may consist of many different fields (in much the same way as a structure or record in a programming language may contain different fields); it is up to the application to define and interpret the fields in the message. The fields might contain the name of the sender, a code indicating the type of the message, and some additional data.

Within the source host, the contents of the entire message M is then "passed" down the protocol stack to layer 3. In the example in Figure 1.23, layer 3 in the

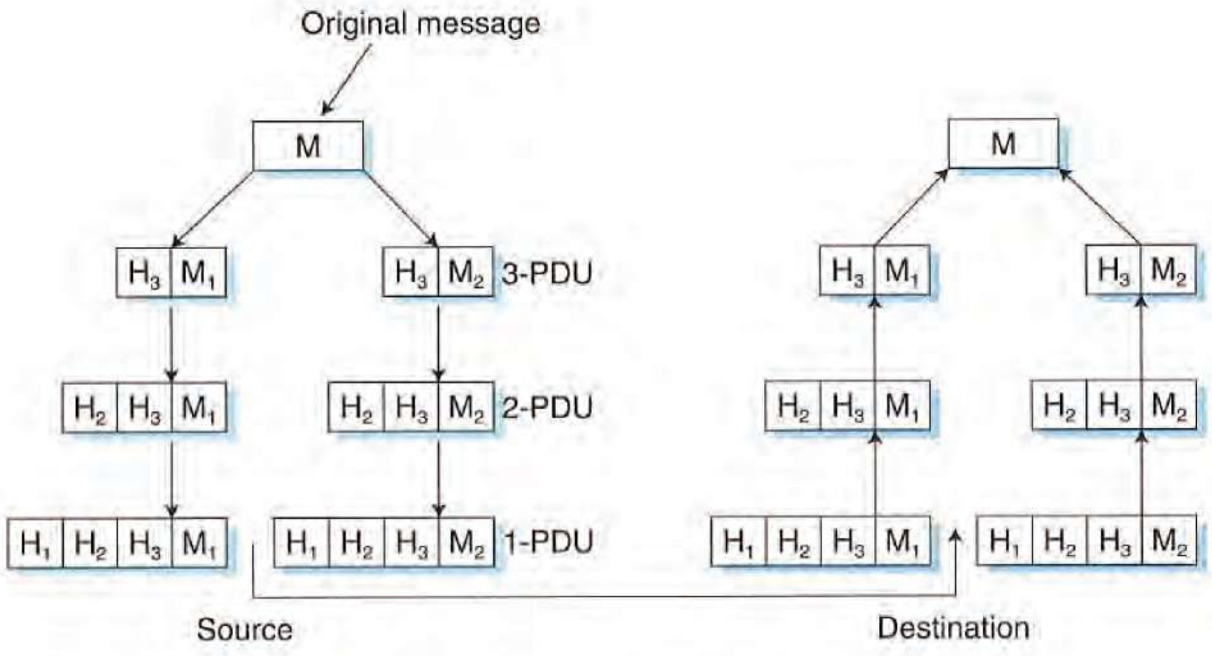


Figure 1.23 ♦ Different PDUs at different layers in the protocol architecture

source host divides a 4-PDU, M, into two parts, M_1 and M_2. The layer 3 in the source host then adds to M_1 and M_2 so-called **headers** to create two layer-3 PDUs. Headers contain the additional information needed by the sending and receiving sides of layer 3 to implement the service that layer 3 provides to layer 4. The procedure continues in the source, adding more header at each layer, until the 1-PDUs are created. The 1-PDUs are sent out of the source host onto a physical link. At the other end, the destination host receives 1-PDUs and directs them up the protocol stack. At each layer, the corresponding header is removed. Finally, M is reassembled from M_1 and M_2 and then passed on to the application.

Note that in Figure 1.23, layer n uses the services of layer $n-1$. For example, once layer 4 creates the message M, it passes the message down to layer 3 and relies on layer 3 to deliver the message to layer 4 at the destination.

Interestingly enough, this notion of relying on lower-layer services is prevalent in many other forms of communication. For example, consider ordinary postal mail. When you write a letter, you include envelope information such as the destination address and the return address with the letter. The letter, along with the address information, can be considered a PDU at the highest layer of the protocol stack. You then drop the PDU in a mailbox. At this point, the letter is out of your hands. The postal service may then add some of its own internal information onto your letter, essentially adding a header to your letter. For example, in the United States a barcode is often printed on your letter.

Once you drop your envelope into a mailbox, you *rely* on the services of the postal service to deliver the letter to the correct destination in a timely manner. For example, you don't worry about whether a postal truck will break down while carrying the letter. Instead the postal service takes care of this, presumably with well-defined plans to recover from such failures. Furthermore, within the postal service itself there are layers, and the protocols at one layer rely on and use the services of the layer below.

In order for one layer to interoperate with the layer below it, the interfaces between the two layers must be precisely defined. Standards bodies define precisely the interfaces between adjacent layers (for example, the format of the PDUs passed between the layers) and permit the developers of networking software and hardware to implement the interior of the layers as they please. Therefore, if a new and improved implementation of a layer is released, the new implementation can replace the old implementation and, in theory, the layers will continue to interoperate.

Layer Functions

In a computer network, each layer may perform one or more of the following generic set of tasks:

- **Error control,** which makes the logical channel between the layers in two peer network elements more reliable
- **Flow control,** which avoids overwhelming a slower peer with PDUs
- **Segmentation and reassembly,** which at the transmitting side divides large data chunks into smaller pieces and at the receiving side reassembles the smaller pieces into the original large chunk
- **Multiplexing,** which allows several higher-level sessions to share a single lower-level connection
- **Connection setup,** which provides handshaking with a peer

Protocol layering has conceptual and structural advantages. We mention, however, that some researchers and networking engineers are vehemently opposed to layering [Wakeman 1992]. One potential drawback of layering is that one layer may duplicate lower-layer functionality. For example, many protocol stacks provide error recovery on both a link basis and an end-to-end basis. A second potential drawback is that functionality at one layer may need information (for example, a timestamp value) that is present only in another layer; this violates the goal of separation of layers.

1.7.2 The Internet Protocol Stack

The Internet stack consists of five layers: the physical, data link, network, transport, and application layers. Rather than use the cumbersome terminology *n*-PDU for each of the five layers, we instead give special names to the PDUs in four of the five layers: frame, datagram, segment, and message. We don't name a data unit for the physical layer, as no name is commonly used at this layer. The Internet stack and the corresponding PDU names are illustrated in Figure 1.24.

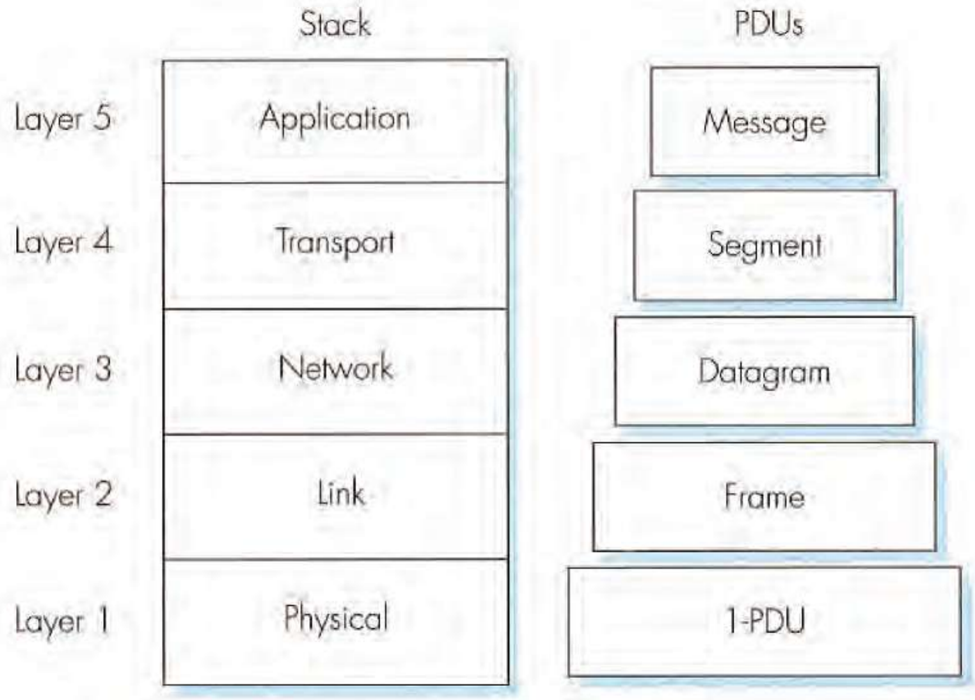


Figure 1.24 ◆ **The Internet protocol stack, and protocol data units**

A protocol layer can be implemented in software, in hardware, or using a combination of the two. Application-layer protocols—such as HTTP and SMTP—are almost always implemented in software in the end systems; so are transport-layer protocols. Because the physical layer and data link layers are responsible for handling communication over a specific link, they are typically implemented in a network interface card (for example, Ethernet or ATM interface cards) associated with a given link. The network layer is often a mixed implementation of hardware and software. We now summarize the Internet layers and the services they provide:

Application Layer

The application layer is responsible for supporting network applications. The application layer includes many protocols, including HTTP to support the Web, SMTP to support electronic mail, and FTP to support file transfer. We shall see in Chapter 2 that it is very easy to create our own new application-layer protocols.

Transport Layer

The transport layer provides the service of transporting application-layer messages between the client and server sides of an application. In the Internet there are two transport protocols, TCP and UDP, either of which can transport application-layer messages. TCP provides a connection-oriented service to its applications. This service includes guaranteed delivery of application-layer messages to the destination and flow control (that is, sender/receiver speed matching). TCP also segments long messages into shorter segments and provides a congestion control mechanism, so that a source throttles its transmission rate when the network is congested. The UDP protocol provides its applications a connectionless service, which (as we saw in Section 1.3) is very much a no-frills service.

Network Layer

The network layer is responsible for routing datagrams from one host to another. The Internet's network layer has two principle components. It has a protocol that defines the fields in the IP datagram as well as how the end systems and routers act on these fields. This protocol is the celebrated IP protocol. There is only one IP protocol, and all Internet components that have a network layer must run the IP protocol. The Internet's network layer also contains routing protocols that determine the routes that datagrams take between sources and destinations. The Internet has many routing protocols. As we saw in Section 1.4, the Internet is a network of networks, and within a network, the network administrator can run any routing protocol desired. Although the network layer contains both the IP protocol and numerous routing protocols, it is often simply referred to as the IP layer, reflecting the fact that IP is the glue that binds the Internet together.

The Internet transport layer protocols (TCP and UDP) in a source host passes a transport-layer segment and a destination address to the IP layer, just as you give the postal service a letter with a destination address. The IP layer then provides the service of routing the segment to its destination. When the packet arrives at the destination, IP passes the segment to the transport layer within the destination.

Link Layer

The network layer routes a packet through a series of packet switches (called routers, in the Internet) between the source and destination. To move a packet from one node (host or packet switch) to the next node in the route, the network layer must rely on the services of the link layer. In particular, at each node IP passes the datagram to the link layer, which delivers the datagram to the next node along the route. At this next node, the link layer passes the IP datagram to the network layer. The process is analogous to the postal worker at a mailing center who puts a letter into a plane that will deliver the letter to the next postal center along the route. The services provided at the link layer depend on the specific link-layer protocol that is employed over the link. For example, some protocols provide reliable delivery on a link basis, that is, from transmitting node, over one link, to receiving node. Note that this reliable delivery service is different from the reliable delivery service of TCP, which provides reliable delivery from one end system to another. Examples of link layers include Ethernet and PPP; in some contexts, ATM and frame relay can be considered link layers. As datagrams typically need to traverse several links to travel from source to destination, a datagram may be handled by different link-layer protocols at different links along its route. For example, a datagram may be handled by Ethernet on one link and then PPP on the next link. IP will receive a different service from each of the different link-layer protocols.

Physical Layer

While the job of the link layer is to move entire frames from one network element to an adjacent network element, the job of the physical layer is to move the *individual bits* within the frame from one node to the next. The protocols in this layer are again link dependent, and further depend on the actual transmission medium of the link (for example, twisted-pair copper wire, single-mode fiber optics). For example, Ethernet has many physical layer protocols: one for twisted-pair copper wire, another for coaxial cable, another for fiber, and so on. In each case, a bit is moved across the link in a different way.

If you examine the Table of Contents, you will see that we have roughly organized this book using the layers of the Internet protocol stack. We take a **top-down approach,** first covering the application layer and then preceding downwards.

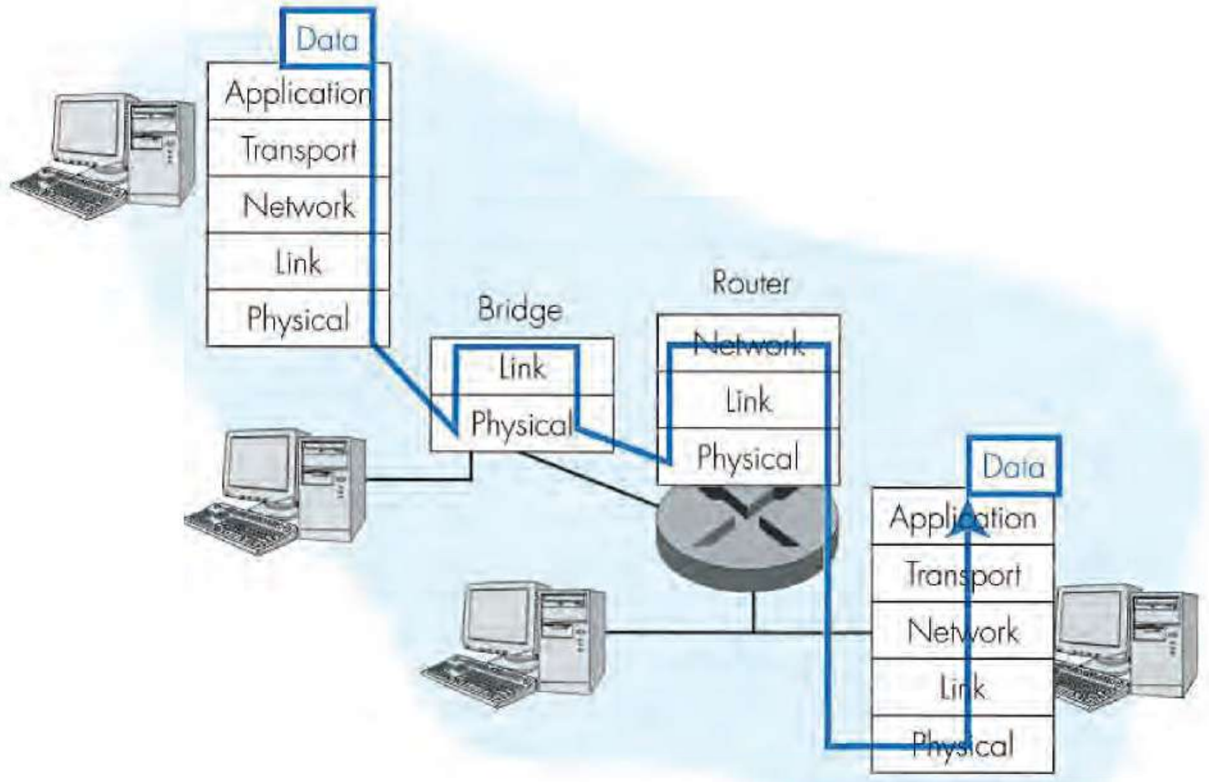


Figure 1.25 ◆ Hosts, routers, and bridges; each contains a different set of layers, reflecting their differences in functionality

1.7.3 Network Entities and Layers

The most important network entities are end systems and packet switches. As we discuss later in this book, there are two types of packet switches: routers and bridges. We presented an overview of routers in the earlier sections. Bridges will be discussed in detail in Chapter 5 whereas routers will be covered in more detail in Chapter 4. Similar to end systems, routers and bridges organize the networking hardware and software into layers. But routers and bridges do not implement *all* of the layers in the protocol stack; they typically implement only the bottom layers. As shown in Figure 1.25, bridges implement layers 1 and 2; routers implement layers 1 through 3. This means, for example, that Internet routers are capable of implementing the IP protocol (a layer 3 protocol), while bridges are not. We will see later that while bridges do not recognize IP addresses, they are capable of recognizing layer 2 addresses, such as Ethernet addresses. Note that hosts implement all five layers; this is consistent with the view that the Internet architecture puts much of its complexity at the "edges" of the network.

1.8 ✦ Internet Backbones, NAPs, and ISPs

Our discussion of layering in the previous section has perhaps given the impression that the Internet is a carefully organized and highly intertwined structure. This is certainly true in the sense that all of the network entities (end systems, routers, and bridges) use a common set of protocols, enabling the entities to communicate with each other. However, from a topological perspective, to many people the Internet seems to be growing in a chaotic manner, with new sections, branches, and wings popping up in random places on a daily basis. Indeed, unlike the protocols, the Internet's topology can grow and evolve without approval from a central authority. Let us now try to get a grip on the seemingly nebulous Internet topology.

As we mentioned at the beginning of this chapter, the topology of the Internet is loosely hierarchical. Roughly speaking, from bottom-to-top the hierarchy consists of end systems (PCs, workstations, and so on) connected to local Internet service providers (ISPs). The local ISPs are in turn connected to regional ISPs, which are in turn connected to national and international ISPs. The national and international ISPs are connected together at the highest tier in the hierarchy. New tiers and branches can be added just as a new piece of Lego can be attached to an existing Lego construction.

In this section we describe the topology of the Internet in the United States as of 2000. Let's begin at the top of the hierarchy and work our way down. Residing at the very top of the hierarchy are the national ISPs, which are called **national service providers (NSPs).** The NSPs form independent backbone networks that span North America (and typically extend abroad as well). Just as there are multiple long-distance telephone companies in the United States, there are multiple NSPs that compete with each other for traffic and customers. The existing NSPs include internetMCI, SprintLink, PSINet, UUNet Technologies, and AGIS. The NSPs typically have high-bandwidth transmission links, with bandwidths ranging from 1.5 Mbps to 622 Mbps and higher. Each **NSP** also has numerous hubs that interconnect its links and at which **regional ISPs** can tap into the NSP.

The NSPs themselves must be interconnected to each other. To see this, suppose one regional ISP, say MidWestnet, is connected to the MCI NSP and another regional ISP, say EastCoastnet, is connected to Sprint's NSP. How can traffic be sent from MidWestnet to EastCoastnet? The solution is to introduce switching centers, called **network access points (NAPs),** which interconnect the NSPs, thereby allowing each regional ISP to pass traffic to any other regional ISP. To keep us all confused, some of the NAPs are not referred to as NAPs but instead as MAEs (metropolitan area exchanges). In the United States, many of the NAPs are run by RBOCs (regional Bell operating companies); for example, PacBell has a NAP in San Francisco and Ameritech has a NAP in Chicago. For a list of major NSPs (those connected into at least three MAPs/MAE's), see [Haynal 1999]. In addition to connecting to each other at NAPs, NSPs can connect to each other through so-called private peering points; see Figure 1.26. For a discussion of NAPs as well as private peering among NSPs, see [Huston 1999a].

Because the NAPs relay and switch tremendous volumes of Internet traffic, they are typically in themselves complex high-speed switching networks concentrated in a small geographical area (for example, a single building). Often the NAPs use high-speed ATM switching technology in the heart of the NAP, with IP riding on top of

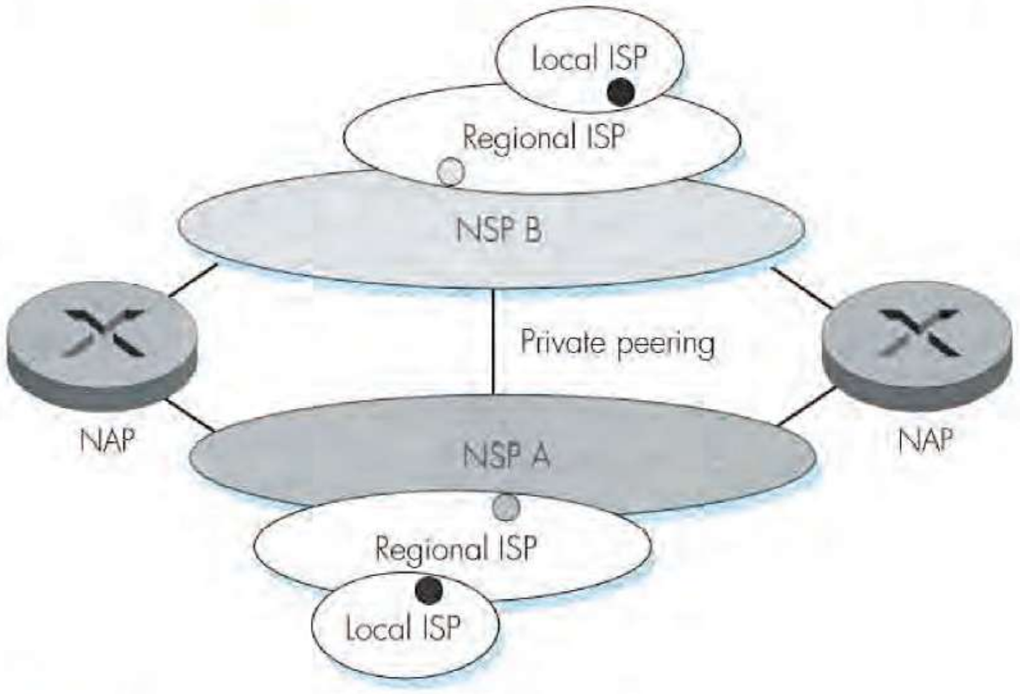


Figure 1.26 ✦ Internet structure: Network of networks

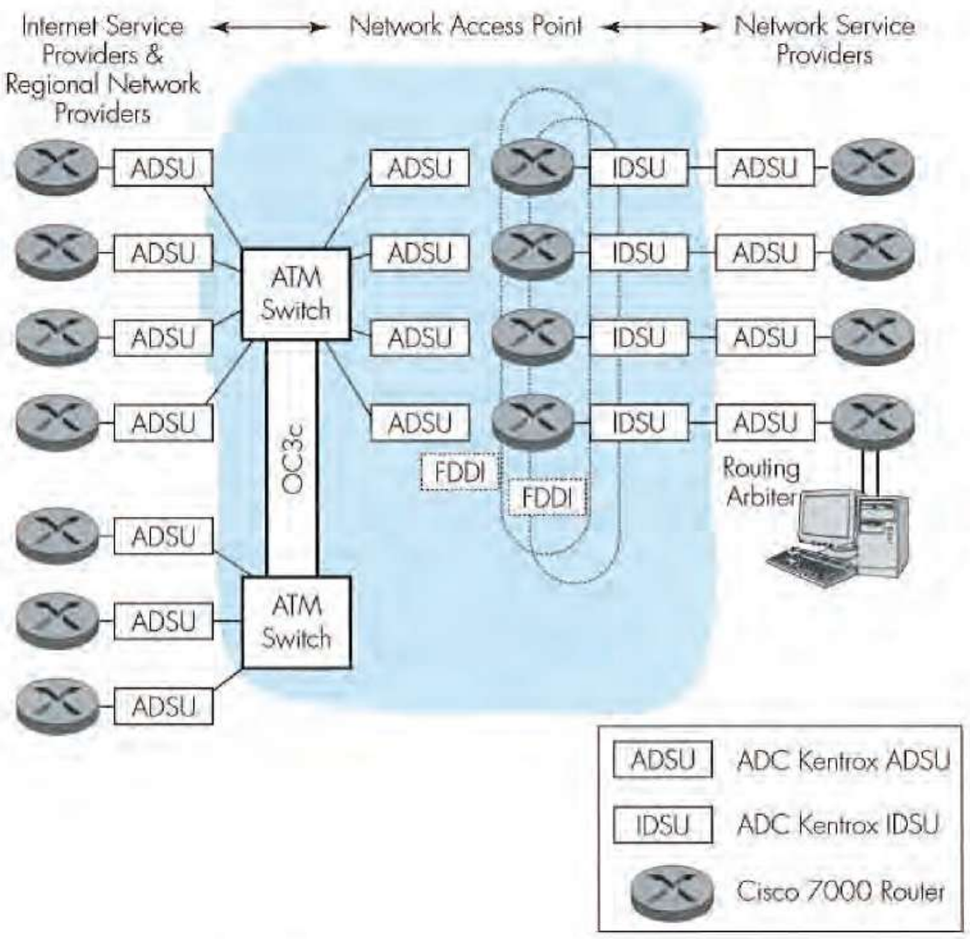


Figure 1.27 ✦ The PacBell NAP architecture (courtesy of the Pacific Bell Web site)

ATM. Figure 1.27 illustrates PacBell's San Francisco NAP. The details of Figure 1.27 are unimportant for us now; it is worthwhile to note, however, that the NSP hubs can themselves be complex data networks.

Running an NSP is not cheap. In June 1996, the cost of leasing 45 Mbps fiber optics from coast to coast, as well as the additional hardware required, was approximately \$150,000 per month. And the fees that an NSP pays the NAPs to connect to the NAPs can exceed \$300,000 annually. NSPs and NAPs also have significant capital costs in equipment for high-speed networking. An NSP earns money by charging a monthly fee to the regional ISPs that connect to it. The fee that an NSP charges to a regional ISP typically depends on the bandwidth of the connection between the regional ISP and the NSP; clearly a 1.5 Mbps connection would be charged less than a 45 Mbps connection. Once the fixed-bandwidth connection is in place, the regional ISP can pump and receive as much data as it pleases, up to the bandwidth of the connection, at no additional cost. If an NSP has significant revenues from the regional ISPs that connect to it, it may be able to cover the high capital and monthly costs of setting up and maintaining an NSP. For a discussion of the current practice of financial settlement among interconnected network providers, see [Huston 1999b].

A regional ISP is also a complex network, consisting of routers and transmission links with rates ranging from 64 Kbps upward. A regional ISP typically taps into an NSP (at an NSP hub), but it can also tap directly into a NAP, in which case the regional ISP pays a monthly fee to a NAP instead of to an NSP. A regional ISP can also tap into the Internet backbone at two or more distinct points (for example, at an NSP hub or at a NAP). How does a regional ISP cover its costs? To answer this question, let's jump to the bottom of the hierarchy.

End systems gain access to the Internet by connecting to a local ISP. Universities and corporations can act as local ISPs, but backbone service providers can also serve as a local ISP. Many local ISPs are small "mom and pop" companies, however. A popular Web site known simply as "The List" contains links to nearly 8,000 local, regional, and backbone ISPs [List 1999]. The local ISPs tap into one of the regional ISPs in its region. Analogous to the fee structure between the regional ISP and the NSP, the local ISP pays a monthly fee to its regional ISP that depends on the bandwidth of the connection. Finally, the local ISP charges its customers (typically) a flat, monthly fee for Internet access: the higher the transmission rate of the connection, the higher the monthly fee.

We conclude this section by mentioning that any one of us can become a local ISP as soon as we have an Internet connection. All we need to do is purchase the necessary equipment (for example, router and modem pool) that is needed to allow other users to connect to our so-called point of presence. Thus, new tiers and branches can be added to the Internet topology just as a new piece of Lego can be attached to an existing Lego construction.

1.9 ✦ A Brief History of Computer Networking and the Internet

Sections 1.1–1.8 presented an overview of technology of computer networking and the Internet. You should know enough now to impress your family and friends. However, if you really want to be a big hit at the next cocktail party, you should sprinkle your discourse with tidbits about the fascinating history of the Internet [Segaller 1998].

1.9.1 Development and Demonstration of Early Packet Switching Principles: 1961–1972

The fields of computer networking and today's Internet trace their beginnings back to the early 1960s, a time at which the telephone network was the world's dominant communication network. Recall from Section 1.4 that the telephone network uses circuit switching to transmit information from a sender to receiver—an appropriate choice given that voice is transmitted at a constant rate between sender and receiver. Given the increasing importance (and great expense) of computers in the early 1960s and the advent of timeshared computers, it was perhaps natural (at least with perfect hindsight!) to consider the question of how to hook computers together so that they could be shared among geographically distributed users. The traffic generated by such users was likely to be "bursty"—intervals of activity, such as the sending of a command to a remote computer, followed by periods of inactivity while waiting for a reply or while contemplating the received response.

Three research groups around the world, all unaware of the others' work [Leiner 1998], began inventing the notion of packet switching as an efficient and robust alternative to circuit switching. The first published work on packet-switching techniques was that of Leonard Kleinrock [Kleinrock 1961, Kleinrock 1964], at that time a graduate student at MIT. Using queuing theory, Kleinrock's work elegantly demonstrated the effectiveness of the packet-switching approach for bursty traffic sources. In 1964, Paul Baran [Baran 1964] at the Rand Institute had begun investigating the use of packet switching for secure voice over military networks, and at the National Physical Laboratory in England, Donald Davies and Roger Scantlebury were also developing their ideas on packet switching.

The work at MIT, Rand, and NPL laid the foundations for today's Internet. But the Internet also has a long history of a let's-build-it-and-demonstrate-it attitude that also dates back to the early 1960s. J.C.R. Licklider [DEC 1990] and Lawrence Roberts, both colleagues of Kleinrock's at MIT, went on to lead the computer science program at the Advanced Research Projects Agency (ARPA) in the United States. Roberts published an overall plan for the so-called ARPAnet [Roberts 1967], the first packet-switched computer network and a direct ancestor of today's public Internet. The early packet switches were known as interface message processors (IMPs) and the contract to build these switches was awarded to the BBN company. On Labor Day in 1969, the first IMP was installed at UCLA under Kleinrock's supervision, with three additional IMPs being installed shortly thereafter at the Stanford Research Institute (SRI), UC Santa Barbara, and the University of Utah (Figure 1.28). The fledgling precursor to the Internet was four nodes large by the end of 1969. Kleinrock recalls the very first use of the network to perform a remote login from UCLA to SRI, crashing the system [Kleinrock 1998].

By 1972, ARPAnet had grown to approximately 15 nodes, and was given its first public demonstration by Robert Kahn at the 1972 International Conference on Computer Communications. The first host-to-host protocol between ARPAnet end systems known as the network-control protocol (NCP) was completed [RFC 001]. With an end-to-end protocol available, applications could now be written. The first e-mail program was written by Ray Tomlinson at BBN in 1972.

1.9.2 Internetworking, and New and Proprietary Networks: 1972–1980

The initial ARPAnet was a single, closed network. In order to communicate with an ARPAnet host, one had to actually be attached to another ARPAnet IMP. In the early to mid 1970s, additional packet-switching networks besides ARPAnet came into

Figure 1.28 ◆ The first interface message processor (IMP), with L. Kleinrock

being: ALOHAnet, a microwave network linking together universities on the Hawaiian islands [Abramson 1970]; Telenet, a BBN commercial packet-switching network based on ARPAnet technology; Tymnet; and Transpac, a French packet-switching network. The number of networks was beginning to grow. In 1973, Robert Metcalfe's Ph.D. thesis laid out the principle of Ethernet, which would later lead to a huge growth in so-called local area networks (LANs) that operated over a small distance based on the Ethernet protocol.

Once again, with perfect hindsight one might now see that the time was ripe for developing an encompassing architecture for connecting networks together. Pioneering work on interconnecting networks (once again under the sponsorship of DARPA—Defense Advanced Research Projects Agency), in essence creating a *network of networks,* was done by Vinton Cerf and Robert Kahn [Cerf 1974]; the term "internetting" was coined to describe this work.

These architectural principles were embodied in the TCP protocol. The early versions of TCP, however, were quite different from today's TCPs. The early ver-

Internet Design Principles

The architectural principles that Cerf and Kahn [Cerf 1974] articulated for creating a so-called "open network architecture" are the foundation on which today's Internet is built [Leiner 1998]:

- *Minimalism, autonomy:* A network should be able to operate on its own, with no internal changes required for it to be internetworked with other networks.
- *Best-effort service:* Internetworked networks would provide best-effort, end-to-end service. If reliable communication was required, this could be accomplished by retransmitting lost messages from the sending host.
- *Stateless routers:* The routers in the internetworked networks would not maintain any per-flow state about any ongoing connection.
- *Decentralized control:* There would be no global control over the internetworked networks.

These principles continue to serve as the architectural foundation for today's Internet, even 25 years later—a testament to insight of the early Internet designers. For an interesting retrospective look at the Internet design philosophy, see [Clark 1988].

sions of TCP combined a reliable in-sequence delivery of data via end-system retransmission (still part of today's TCP) with forwarding functions (which today are performed by IP). Early experimentation with TCP, combined with the recognition of the importance of an unreliable, non-flow-controlled end-end transport service for applications such as packetized voice, led to the separation of IP out of TCP and the development of the UDP protocol. The three key Internet protocols that we see today—TCP, UDP, and IP—were conceptually in place by the end of the 1970s.

In addition to the DARPA Internet-related research, many other important networking activities were underway. In Hawaii, Norman Abramson was developing ALOHAnet, a packet-based radio network that allowed multiple remote sites on the Hawaiian islands to communicate with each other. The ALOHA protocol [Abramson 1970] was the first so-called multiple-access protocol, allowing geographically distributed users to share a single broadcast communication medium (a radio frequency). Abramson's work on multiple-access protocols was built upon by Metcalfe and Boggs in the development of the Ethernet protocol [Metcalfe 1976] for wire-based shared broadcast networks; see Figure 1.29. Interestingly, Metcalfe and Boggs' Ethernet protocol was motivated by the need to connect multiple PCs, printers, and shared disks together [Perkins 1994]. Twenty-five years ago, well before the PC revolution and the explosion of networks, Metcalfe and Boggs were laying the foundation for today's PC LANs. Ethernet technology represented an important step for internetworking as well. Each Ethernet local area network was itself a network, and

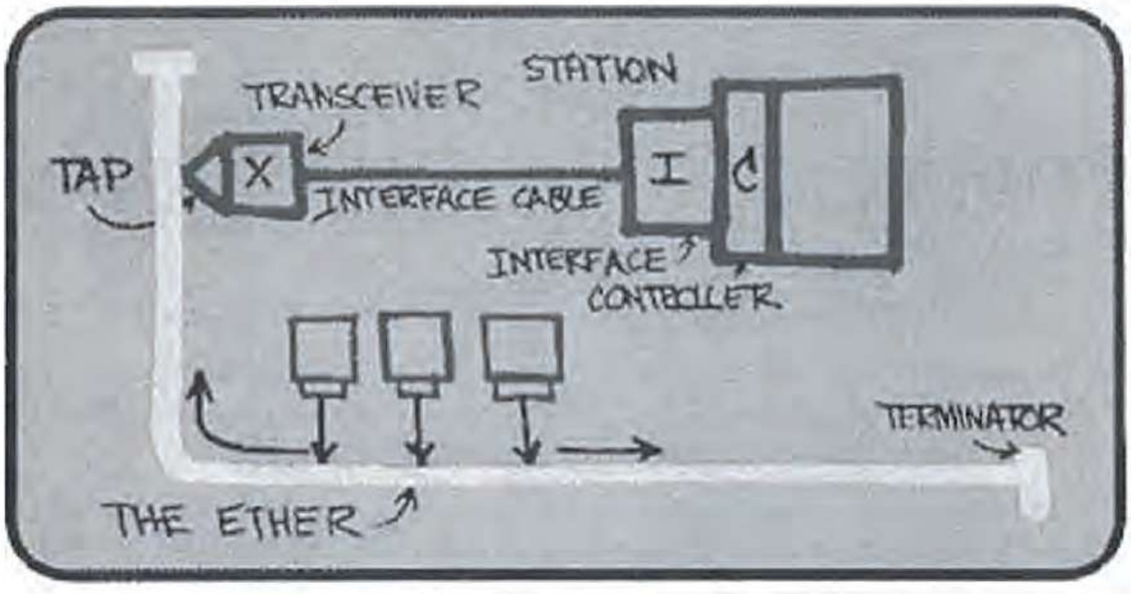


Figure 1.29 ♦ Metcalf's original conception of the Ethernet.

as the number of LANs proliferated, the need to internetwork these LANs together became increasingly important. We discuss Ethernet, Aloha, and other LAN technologies in detail in Chapter 5.

In addition to the DARPA internetworking efforts and the Aloha/Ethernet multiple-access networks, a number of companies were developing their own proprietary network architectures. Digital Equipment Corporation (Digital) released the first version of the DECnet in 1975, allowing two PDP-11 minicomputers to communicate with each other. DECnet has continued to evolve since then, with significant portions of the OSI protocol suite being based on ideas pioneered in DECnet. Other important players during the 1970s were Xerox (with the XNS architecture) and IBM (with the SNA architecture). Each of these early networking efforts would contribute to the knowledge base that would drive networking in the 80s and 90s.

It is important to note here that in the 1980s (and even before), researchers such as [Fraser 1983, 1993] and [Turner 1986] were also developing a competitor technology to the Internet architecture. These efforts have contributed to the development of the ATM architecture, a connection-oriented approach based on the use of fixed-size packets, known as cells. We will examine portions of the ATM architecture throughout this book.

1.9.3 A Proliferation of Networks: 1980–1990

By the end of the 1970s, approximately 200 hosts were connected to the ARPAnet. By the end of the 1980s the number of hosts connected to the public Internet, a confederation of networks looking much like today's Internet, would reach 100,000. The 1980s would be a time of tremendous growth.

Much of the growth in the early 1980s resulted from several distinct efforts to create computer networks linking universities together. BITnet (because it's their network) provided e-mail and file transfers among several universities in the Northeast. CSNET (computer science network) was formed to link together university researchers without access to ARPAnet. In 1986, NSFNET was created to provide access to NSF-sponsored supercomputing centers. Starting with an initial backbone speed of 56 Kbps, NSFNET's backbone would be running at 1.5 Mbps by the end of the decade, and would be serving as a primary backbone linking together regional networks.

In the ARPAnet community, many of the final pieces of today's Internet architecture were falling into place. January 1, 1983, saw the official deployment of TCP/IP as the new standard host protocol for ARPAnet (replacing the NCP protocol). The transition [RFC 801] from NCP to TCP/IP was a "flag day" type event—all hosts were required to transfer over to TCP/IP as of that day. In the late 1980s, important extensions were made to TCP to implement host-based congestion control [Jacobson 1988]. The Domain Name System, used to map between a human-readable Internet name (for example, gaia.cs.umass.edu) and its 32-bit IP address, was also developed [RFC 1034].

Paralleling this development of the ARPAnet (which was for the most part a United States effort), in the early 1980s the French launched the Minitel project, an ambitious plan to bring data networking into everyone's home. Sponsored by the French government, the Minitel system consisted of a public packet-switched network (based on the X.25 protocol suite, which uses virtual circuits), Minitel servers, and inexpensive terminals with built-in low speed modems. The Minitel became a huge success in 1984 when the French government gave away a free Minitel terminal to each French household that wanted one. Minitel sites included free sites—such as a telephone directory site—as well as private sites, which collected a usage-based fee from each user. At its peak in the mid 1990s, it offered more than 20,000 different services, ranging from home banking to specialized research databases. It was used by over 20% of France's population, generated more than $1 billion each year, and created 10,000 jobs. The Minitel was in a large proportion of French homes 10 years before most Americans had ever heard of the Internet. It still enjoys widespread use in France, but is increasingly facing stiff competition from the Internet.

1.9.4 Commercialization and the Web: The 1990s

The 1990s were ushered in with two events that symbolized the continued evolution and the soon-to-arrive commercialization of the Internet. First, ARPAnet, the progenitor of the Internet ceased to exist. MILNET and the Defense Data Network had grown in the 1980s to carry most of the U.S. Department-of-Defense-related traffic and NSFnet had begun to serve as a backbone network connecting regional networks in the United States and national networks overseas. In 1991, NSFNET lifted its restrictions on use of NSFNET for commercial purposes. NSFNET itself would be decommissioned in 1995, with Internet backbone traffic being carried by commercial Internet service providers.

The main event of the 1990s, however, was to be the release of the World Wide Web, which brought the Internet into the homes and businesses of millions and millions of people worldwide. The Web also served as a platform for enabling and deploying hundreds of new applications, including online stock trading and banking, streamed multimedia services, and information retrieval services. For a brief history of the early days of the Web, see [W3C 1995].

The Web was invented at CERN by Tim Berners-Lee in 1989–1991 [Berners-Lee 1989], based on ideas originating in earlier work on hypertext from the 1940s by Bush [Bush 1945] and since the 1960s by Ted Nelson [Ziff-Davis 1998]. Berners-Lee and his associates developed initial versions of HTML, HTTP, a Web server, and a browser—the four key components of the Web. The original CERN browsers only provided a line-mode interface. Around the end of 1992 there were about 200 Web servers in operation, this collection of servers being the tip of the iceberg for what was about to come. At about this time several researchers were developing Web browsers with GUI interfaces, including Marc Andreesen, who led the development of the popular GUI browser Mosaic for X. Andreesen and his colleagues released an alpha version of his browser in 1993, and in 1994 he and James Baker formed Mosaic Communications, which later became Netscape Communications Corporation [Cusumano 1998; Quittner 1998]. By 1995, university students were using Mosaic and Netscape browsers to surf the Web on a daily basis. At about this time companies—big and small—began to operate Web servers and transact commerce over the Web. In 1996, Microsoft got into the Web business in a big way.

During the 1990s, networking research and development also made significant advances in the areas of high-speed routers and routing (see Chapter 4) and local area networks (see Chapter 5). The technical community struggled with the problems of defining and implementing an Internet service model for traffic requiring real-time constraints, such as continuous media applications (see Chapter 6). The need to secure and manage Internet infrastructure (see Chapters 7 and 8) also became of paramount importance as e-commerce applications proliferated and the Internet became a central component of the world's telecommunications infrastructure.

1.10 Summary

In this chapter we've covered a tremendous amount of material! We've looked at the various pieces of hardware and software that make up the Internet in particular, and computer networks in general. We started at the "edge" of the network, looking at end systems and applications, and at the transport service provided to the applications running on the end systems. Using network-based distributed applications as examples, we introduced the notion of a protocol—a key concept in networking. We then dove deeper inside the network, into the network core, identifying packet switching and circuit switching as the two basic approaches for transporting data through a telecommunication network, and we examined the strengths and weaknesses of each approach. We then looked at the lowest (from an architectural standpoint) parts of the network—the link-layer technologies and physical media typically found in the access network.

In the second part of this introductory chapter, we took the broader view on networking. From a performance standpoint, we identified the causes of packet delay and packet loss in the Internet. We identified key architectural principles (layering, service models) in networking. We then examined the structure of today's Internet. We finished our introduction to networking with a brief history of computer networking. The first chapter in itself constitutes a mini-course in computer networking.

So, we have indeed covered a tremendous amount of ground in this first chapter! If you're a bit overwhelmed, don't worry. In the following chapters we will revisit all of these ideas, covering them in much more detail (that's a promise, not a threat!). At this point, we hope you leave this chapter with a still-developing intuition for the pieces that make up a network, a still-developing command for the vocabulary of networking (don't be shy to refer back to this chapter), and an ever-growing desire to learn more about networking. That's the task ahead of us for the rest of this book.

Roadmapping This Book

Before starting any trip, we should always glance at a roadmap in order to become familiar with the major roads and junctures that lie between us and our ultimate destination. For the trip we are about to embark on, the ultimate destination is a deep understanding of the how, what, and why of computer networks. Our roadmap is the sequence of chapters of this book:

1. Computer Networks and the Internet
2. Application Layer
3. Transport Layer
4. Network Layer and Routing
5. Link Layer and Local Area Networks
6. Multimedia Networking
7. Security in Computer Networks
8. Network Management

Taking a look at this roadmap, we identify Chapters 2 through 5 as the four core chapters of this book. You should notice that there is one chapter for each of the top four layers of the Internet protocol stack. Further note that our journey will begin at the top of the Internet protocol stack, namely, the application layer, and will work its way downward. The rationale behind this top-down journey is that once we understand the applications, we can then understand the network services needed to support these applications. We can then, in turn, examine the various ways in which such services might be implemented by a network architecture. Covering applications early thus provides motivation for the remainder of the text.

The second half of the book—Chapters 6 through 8—zoom in on three enormously important (and somewhat independent) topics in modern computer networking. In Chapter 6 (Multimedia Networking), we examine audio and video applications such as Internet phone, video conferencing, and streaming of stored media. We also look at how a packet-switched network can be designed to provide consistent quality of service to audio and video applications. In Chapter 7 (Security in Computer Networks), we first look at the underpinnings of encryption and network security, and then we examine how the basic theory is being applied in a broad range of Internet contexts, including electronic mail and Internet commerce. The last chapter (Network Management) examines the key issues in network management as well as the Internet protocols that address these issues.

Homework Problems and Questions

Chapter 1 ■ Review Questions

Sections 1.1–1.4

1. What are the two types of services that the Internet provides to its applications? What are some characteristics of each of these services?
2. It has been said that flow control and congestion control are equivalent. Is this true for the Internet's connection-oriented service? Are the objectives of flow control and congestion control the same?
3. Briefly describe how the Internet's connection-oriented service provides reliable transport.
4. What advantage does a circuit-switched network have over a packet-switched network? What advantages does TDM have over FDM in a circuit-switched network?
5. Suppose that between a sending host and a receiving host there is exactly one packet switch. The transmission rates between the sending host and the switch

and between the switch and the receiving host are R_1 and R_2, respectively. Assuming that the router uses store-and-forward packet switching, what is the total end-to-end delay to send a packet of length L? (Ignore queuing, propagation delay, and processing delay.)

6. What are some of the networking technologies that use virtual circuits? (Find good URLs that discuss and explain these technologies.)
7. What is meant by connection state information in a virtual circuit network?
8. Suppose you are developing a standard for a new type of network. You need to decide whether your network will use VCs or datagram routing. What are the pros and cons for using VCs?

Sections 1.5–1.7

9. Is HFC bandwidth dedicated or shared among users? Are collisions possible in a downstream HFC channel? Why or why not?
10. What is the transmission rate of Ethernet LANs? For a given transmission rate, can each user on the LAN continuously transmit at that rate?
11. What are some of the physical media that Ethernet can run over?
12. Dial-up modems, ISDN, HFC, and ADSL are all used for residential access. For each of these access technologies, provide a range of transmission rates and comment on whether the bandwidth is shared or dedicated.
13. Consider sending a series of packets from a sending host to a receiving host over a fixed route. List the delay components in the end-to-end delay for a single packet. Which of these delays are constant and which are fixed?
14. Review the car-caravan analogy in Section 1.6. Again assume a propagation speed of 100 km/hour.
    a. Suppose the caravan travels 200 km, beginning in front of one toll booth, passing through a second toll booth, and finishing just before a third toll booth. What is the end-to-end delay?
    b. Repeat (a), now assuming that there are seven cars in the caravan instead of 10.
15. List five tasks that a layer can perform. It is possible that one (or more) of these tasks could be performed by two (or more) layers?
16. What are the five layers in the Internet protocol stack? What are the principal responsibilities for each of these layers?
17. Which layers in the Internet protocol stack does a router process?

Problems

1. Design and describe an application-level protocol to be used between an automatic teller machine and a bank's centralized computer. Your protocol should allow a user's card and password to be verified, the account balance (which is maintained at the centralized computer) to be queried, and an account withdrawal to be made (that is, money disbursed to the user). Your protocol entities should be able to handle the all-too-common case in which there is not enough money in the account to cover the withdrawal. Specify your protocol by listing the messages exchanged and the action taken by the automatic teller machine or the bank's centralized computer on transmission and receipt of messages. Sketch the operation of your protocol for the case of a simple withdrawal with no errors, using a diagram similar to that in Figure 1.2. Explicitly state the assumptions made by your protocol about the underlying end-to-end transport service.
2. Consider an application that transmits data at a steady rate (for example, the sender generates a N-bit unit of data every k time units, where k is small and fixed). Also, when such an application starts, it will stay on for a relatively long period of time. Answer the following questions, briefly justifying your answer:
   a. Would a packet-switched network or a circuit-switched network be more appropriate for this application? Why?
   b. Suppose that a packet-switching network is used and the only traffic in this network comes from such applications as described above. Furthermore, assume that the sum of the application data rates is less than the capacities of each and every link. Is some form of congestion control needed? Why?
3. Consider sending a file of $F = M \cdot L$ bits over a path of Q links. Each link transmits at R bps. The network is lightly loaded so that there are no queuing delays. When a form of packet switching is used, the $M \cdot L$ bits are broken up into M packets, each packet with L bits. Propagation delay is negligible.
   a. Suppose the network is a packet-switched virtual circuit network. Denote the VC set-up time by t_s seconds. Suppose the sending layers add a total of h bits of header to each packet. How long does it take to send the file from source to destination?
   b. Suppose the network is a packet-switched datagram network and a connectionless service is used. Now suppose each packet has $2h$ bits of header. How long does it take to send the file?
   c. Repeat (b), but assume message switching is used (that is, $2h$ bits are added to the message, and the message is not segmented).

d. Finally, suppose that the network is a circuit-switched network. Further suppose that the transmission rate of the circuit between source and destination is R bps. Assuming t_s set-up time and h bits of header appended to the entire file, how long does it take to send the file?

4. Experiment with the message-switching Java applet in this chapter. Do the delays in the applet correspond to the delays in the previous question? How do link propagation delays affect the overall end-to-end delay for packet switching and for message switching?
5. Consider sending a large file of F bits from Host A to Host B. There are two links (and one switch) between A and B, and the links are uncongested (that is, no queuing delays). Host A segments the file into segments of S bits each and adds 40 bits of header to each segment, forming packets of $L = 40 + S$ bits. Each link has a transmission rate of R bps. Find the value of S that minimizes the delay of moving the packet from Host A to Host B. Neglect propagation delay.
6. This elementary problem begins to explore propagation delay and transmission delay, two central concepts in data networking. Consider two hosts, Hosts A and B, connected by a single link of rate R bps. Suppose that the two hosts are separated by m meters, and suppose the propagation speed along the link is s meters/sec. Host A is to send a packet of size L bits to Host B.

   a. Express the propagation delay, d_{prop} in terms of m and s.

   b. Determine the transmission time of the packet, d_{trans} in terms of L and R.

   c. Ignoring processing and queuing delays, obtain an expression for the end-to-end delay.

   d. Suppose Host A begins to transmit the packet at time $t = 0$. At time $t = d_{\text{trans}}$, where is the last bit of the packet?

   e. Suppose d_{prop} is greater than d_{trans}. At time $t = d_{\text{trans}}$, where is the first bit of the packet?

   f. Suppose d_{prop} is less than d_{trans}. At time $t = d_{\text{trans}}$, where is the first bit of the packet?

   g. Suppose $s = 2.5 \cdot 10^8$, $L = 100$ bits, and $R = 28$ kbps. Find the distance m so that d_{prop} equals d_{trans}.
7. In this problem we consider sending voice from Host A to Host B over a packet-switched network (for example, Internet phone). Host A converts on-the-fly analog voice to a digital 64-Kbps bit stream. Host A then groups the bits into 48-byte packets. There is one link between host A and B; its transmission rate is 1 Mbps and its propagation delay is 2 msec. As soon as Host A gathers a packet, it sends it to Host B. As soon as Host B receives an entire packet, it converts the packet's bits to an analog signal. How much time elapses

from the time a bit is created (from the original analog signal at A) until a bit is decoded (as part of the analog signal at B)?

8. Suppose users share a 1-Mbps link. Also suppose each user requires 100 Kbps when transmitting, but each user only transmits 10 percent of the time. (See the discussion "Packet Switching versus Circuit Switching.")
   a. When circuit switching is used, how many users can be supported?
   b. For the remainder of this problem, suppose packet switching is used. Find the probability that a given user is transmitting.
   c. Suppose there are 40 users. Find the probability that at any given time, n users are transmitting simultaneously.
   d. Find the probability that there are 11 or more users transmitting simultaneously.
9. Consider the queuing delay in a router buffer (preceding an outbound link). Suppose all packets are L bits, the transmission rate is R bps, and that N packets arrive to the buffer every LN/R seconds. Find the average queuing delay of a packet.
10. Consider the queuing delay in a router buffer. Let I denote traffic intensity, that is, $I = La/R$. Suppose that the queuing delay takes the form $IL/R\ (1-I)$ for $I < 1$.
    a. Provide a formula for the total delay, that is, the queuing delay plus the transmission delay.
    b. Plot the total delay as a function of L/R.
11. a. Generalize the end-to-end delay formula in Section 1.6 for heterogeneous processing rates, transmission rates, and propagation delays.
    b. Repeat (a), but now also suppose that there is an average queuing delay of d_{queue} at each node.
12. Perform a *traceroute* between source and destination on the same continent at three different hours of the day. Find the average and standard deviation of the delays. Do the same for a source and destination on different continents.

Discussion Questions

1. Write a one-paragraph description for each of three major projects currently under way at the World Wide Web Consortium (W3C).
2. What is Internet phone? Describe some of the existing products for Internet phone. Find some of the Web sites of companies that are in the Internet phone business.
3. What is streaming of stored audio? Describe some of the existing products for Internet audio streaming. Find some of the Web sites of companies that are in

the Internet audio-streaming business. Find some Web sites that provide streaming content.

4. What is Internet video conferencing? Describe some of the existing products for Internet video conferencing. Find some of the Web sites of companies that are in the Internet video-conferencing business.
5. Surf the Web to find a company that is offering HFC Internet access. What is the transmission rate of the cable modem? Is this rate always guaranteed for each user on the network?
6. Suppose you are developing an application for the Internet. Would you have your application run over TCP or UDP? Elaborate. (We will explore this question in some detail in subsequent chapters. For now, appeal to your intuition to answer the question.)
7. What does the current topological structure of the Internet (that is, backbone ISPs, regional ISPs, and local ISPs) have in common with the topological structure of the telephone networks in the United States? How is pricing in the Internet the same as or different from pricing in the phone system?

Interview

Leonard Kleinrock

Leonard Kleinrock is a Professor of Computer Science at the University of California, Los Angeles. In 1969, his computer at UCLA became the first node of the Internet. His creation of packet switching principles became the technology behind the Internet. Leonard is also the chairman and founder of Nomadix, Inc., a company whose technology provides greater accessibility of broadband Internet service providers to consumers. He received his BEE from the City College of New York (CCNY) and his Masters and Ph.D. in Electrical Engineering from MIT.

✦ What made you decide to specialize in networking/Internet technology?

As a Ph.D. student at MIT in 1959, I looked around and found that most of my classmates were doing research in the area of information theory and coding theory. At MIT, there was the great researcher Claude Shannon who had launched these fields and had solved most of the important problems already. The research problems that were left were hard and of lesser consequence. So I decided to launch out in a new area that no one else had yet conceived of. Remember that at MIT I was surrounded by lots of computers, and it was clear to me that soon these machines would need to communicate with each other. At the time, there was no effective way for them to do so, so I decided to develop the technology that would permit efficient data networks to be created.

✦ What was your first job in the computer industry? What did it entail?

I went to the evening session at CCNY from 1951–1957 for my bachelor's degree in electrical engineering. During the day, I worked first as a technician and then as an engineer at a small, industrial electronics firm called Photobell. While there, I introduced digital technology to their product line. Essentially, we were using photoelectric devices to detect the presence of certain items (boxes, people, etc.) and the use of a circuit known then as a bistable multivibrator was just the kind of technology we needed to bring digital processing into this field of detection. These circuits happen to be the building blocks for computers.

✦ What is a typical day like for you?

At Nomadix, I spend the day working on the company's vision for the next few years as well as on business opportunities and interviews with the press and public media. At UCLA, I continue to supervise Ph.D. students' research in networking. At night, I continue with my work and my never-ending e-mail, exercise (karate, jogging, swimming), and I read.

✦ In what direction do you see the future of networking?

The clearest part of my vision is that of nomadic computing and smart spaces. The availability of lightweight, inexpensive, high-performance, portable computing devices plus the ubiquity of the Internet has enabled us to become nomads. Nomadic computing refers to the technology that enables end users who travel from place to place to gain access to Internet services in a transparent fashion, no matter where they travel. However, nomadic computing is only one step. The next step will enable us to move out from the netherworld of cyberspace to the physical world of smart spaces. Our environments (desks, walls, vehicles, watches, belts, and so on) will come alive with technology, through actuators, sensors, logic, processing, storage, cameras, microphones, speakers, displays, and communication. This embedded technology will allow our environment to provide the IP services we want. When I walk into a room, the room will know I entered. I will be able to communicate with my environment naturally, as in spoken English; my requests will generate replies that present Web pages to me from wall displays, my eyeglasses, as speech, holograms, and so forth. The Internet will essentially be a pervasive global nervous system.

✦ What people have inspired you professionally?

By far, it was Claude Shannon from MIT, a brilliant researcher who had the ability to relate his mathematical ideas to the physical world in highly intuitive ways. He was on my Ph.D. thesis committee.

✦ Do you have any advice for students entering the networking/ Internet field?

The Internet and all that it enables is a vast new frontier, full of amazing challenges. There is room for great innovation. Don't be constrained by today's technology. Reach out and imagine what could be and then make it happen.

✦ What was going through your mind when you sent the first host-to-host message (from UCLA to the Stanford Research Institute)?

The first host-to-host message was a bit of an anticlimax. In my mind, the more impressive first event took place on September 2, 1969, when the first piece of networking equipment (the IMP) connected to the first operational system in the outside world (my host computer at UCLA). That's when the Internet was born. Earlier that year, I was quoted in a UCLA press release saying that once the network was up and running, it would be possible to gain access to computer utilities from our homes and offices as easily as we gain access to electricity and telephone connectivity. But I never anticipated that my 92-year-old mother would be on the Internet today.