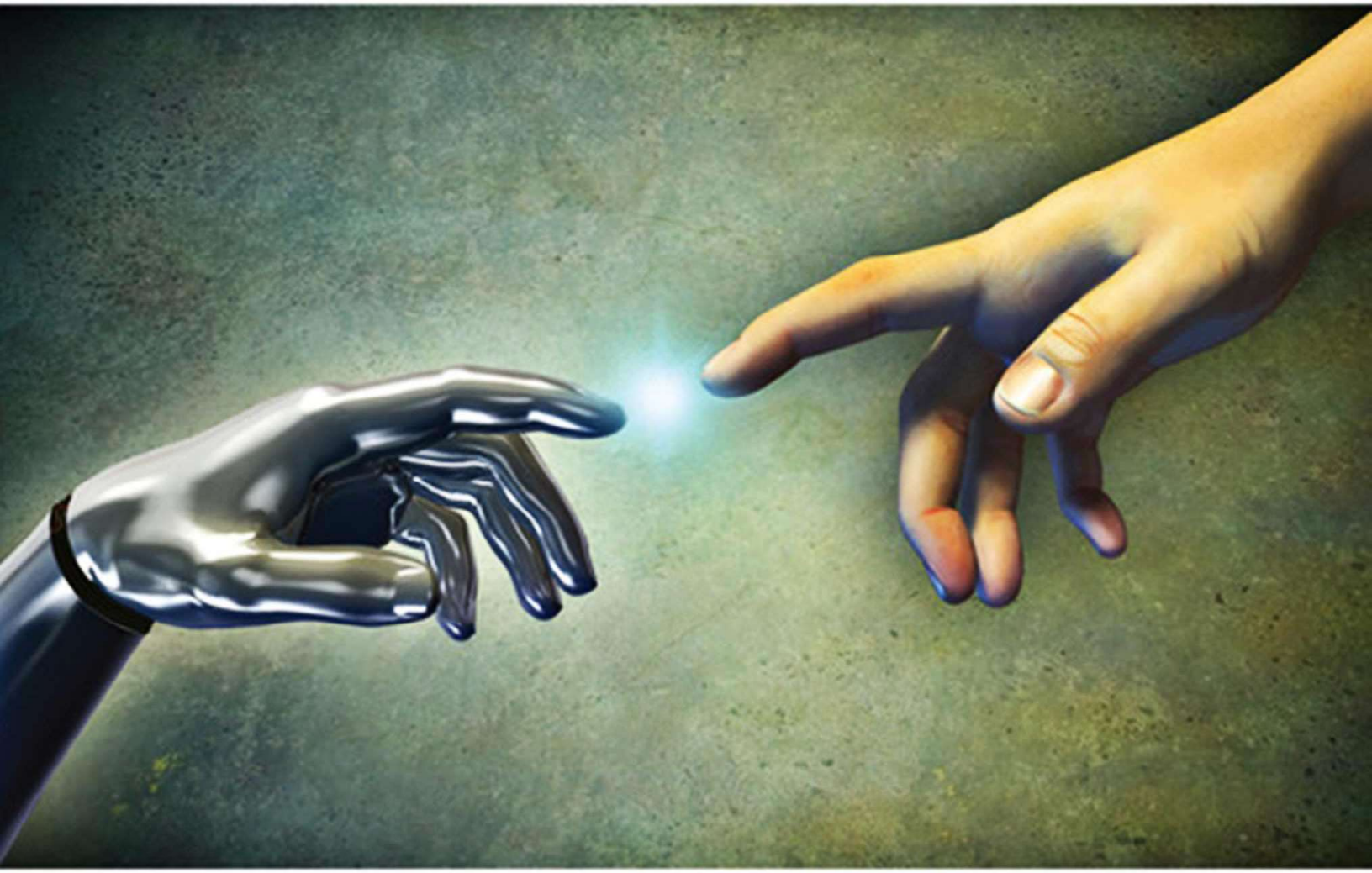
GLOBAL EDITION

Computer Science

An Overview

TWELFTH EDITION



J. Glenn Brookshear • Dennis Brylow

ALWAYS LEARNING

PEARSON

computer science

AN OVERVIEW

12th Edition

Global Edition

J. Glenn Brookshear
and
Dennis Brylow

Global Edition contributions by
Manasa S.

PEARSON

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

Contents

**Asterisks indicate suggestions for optional sections.*

CHAPTER 4

Networking and the Internet

In this chapter we discuss the area of computer science known as networking, which encompasses the study of how computers can be linked together to share information and resources. Our study will include the construction and operation of networks, applications of networks, and security issues. A prominent topic will be a particular worldwide network of networks known as the Internet.

**Asterisks indicate suggestions for optional sections.*

The need to share information and resources among different computers has led to linked computer systems, called **networks,** in which computers are connected so that data can be transferred from machine to machine. In these networks, computer users can exchange messages and share resources—such as printing capabilities, software packages, and data storage facilities—that are scattered throughout the system. The underlying software required to support such applications has grown from simple utility packages into an expanding system of network software that provides a sophisticated network-wide infrastructure. In a sense, network software is evolving into a network-wide operating system. In this chapter we will explore this expanding field of computer science.

4.1 Network Fundamentals

We begin our study of networks by introducing a variety of basic networking concepts.

Network Classifications

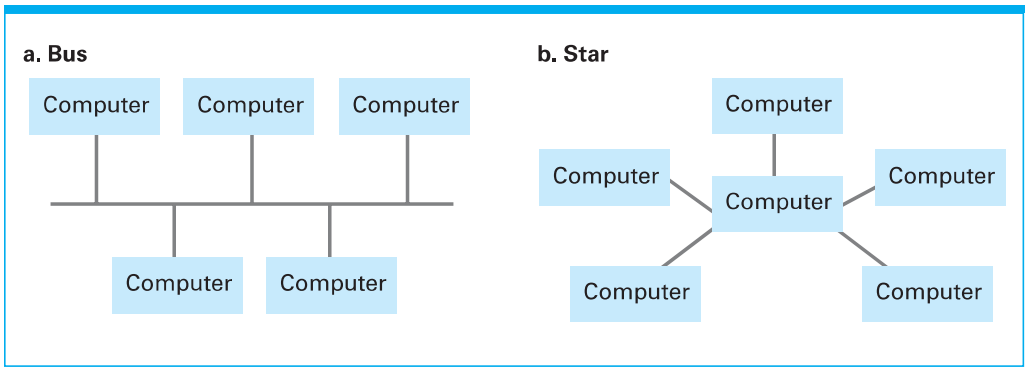
A computer network is often classified as being either a **personal area network (PAN),** a **local area network (LAN),** a **metropolitan area network (MAN),** or a **wide area network (WAN).** A PAN is normally used for short-range communications—typically less than a few meters—such as between a wireless headset and a smartphone or between a wireless mouse and its PC. In contrast, a LAN normally consists of a collection of computers in a single building or building complex. For example, the computers on a university campus or those in a manufacturing plant might be connected by a LAN. A MAN is a network of intermediate size, such as one spanning a local community. Finally, a WAN links machines over a greater distance—perhaps in neighboring cities or on opposite sides of the world.

Another means of classifying networks is based on whether the network's internal operation is based on designs that are in the public domain or on innovations owned and controlled by a particular entity such as an individual or a corporation. A network of the former type is called an **open** network; a network of the latter type is called a **closed,** or sometimes a **proprietary,** network. Open network designs are freely circulated and often grow in popularity to the point that they ultimately prevail over proprietary approaches whose applications are restricted by license fees and contract conditions.

The Internet (a popular worldwide network of networks that we will study in this chapter) is an open system. In particular, communication throughout the Internet is governed by an open collection of standards known as the TCP/IP protocol suite, which is the subject of Section 4.4. Anyone is free to use these standards without paying fees or signing license agreements. In contrast, a company such as Novell Inc. might develop proprietary systems for which it chooses to maintain ownership rights, allowing the company to draw income from selling or leasing these products.

Still another way of classifying networks is based on the topology of the network, which refers to the pattern in which the machines are connected. Two of the more popular topologies are the bus, in which the machines are all connected to a common communication line called a bus (Figure 4.1a), and the star, in which one machine serves as a central focal point to which all the others are

Figure 4.1 Two popular network topologies



connected (Figure 4.1b). The bus topology was popularized in the 1990s when it was implemented under a set of standards known as Ethernet, and Ethernet networks remain one of the most popular networking systems in use today.

The star topology has roots as far back as the 1970s. It evolved from the paradigm of a large central computer serving many users. As the simple terminals employed by these users grew into small computers themselves, a star network emerged. Today, the star configuration is popular in wireless networks where communication is conducted by means of radio broadcast and the central machine, called the **access point (AP),** serves as a focal point around which all communication is coordinated.

The difference between a bus network and a star network is not always obvious by the physical arrangement of equipment. The distinction is whether the machines in the network envision themselves as communicating directly with each other over a common bus or indirectly through an intermediary central machine. For instance, a bus network might not appear as a long bus from which computers are connected over short links as depicted in Figure 4.1. Instead, it may have a very short bus with long links to the individual machines, meaning that the network would look more like a star. Indeed, sometimes a bus network is created by running links from each computer to a central location where they are connected to a device called a **hub.** This hub is little more than a very short bus. All it does is relay any signal it receives (with perhaps some amplification) back out to all the machines connected to it. The result is a network that looks like a star network although it operates like a bus network.

Protocols

For a network to function reliably, it is important to establish rules by which activities are conducted. Such rules are called **protocols.** By developing and adopting protocol standards, vendors are able to build products for network applications that are compatible with products from other vendors. Thus, the development of protocol standards is an indispensable process in the development of networking technologies.

As an introduction to the protocol concept, let us consider the problem of coordinating the transmission of messages among computers in a network. Without rules governing this communication, all the computers might insist on transmitting messages at the same time or fail to assist other machines when that assistance is required.

In a bus network based on the Ethernet standards, the right to transmit messages is controlled by the protocol known as **Carrier Sense, Multiple Access with Collision Detection (CSMA/CD).** This protocol dictates that each message be broadcast to all the machines on the bus (Figure 4.2). Each machine monitors all the messages but keeps only those addressed to itself. To transmit a message, a machine waits until the bus is silent, and at this time it begins transmitting while continuing to monitor the bus. If another machine also begins transmitting, both machines detect the clash and pause for a brief, independently random period of time before trying to transmit again. The result is a system similar to that used by a small group of people in a conversation. If two people start to talk at once, they both stop. The difference is that people might go through a series such as, "I'm sorry, what were you going to say?", "No, no. You go first," whereas under the CSMA/CD protocol each machine merely tries again later.

Note that CSMA/CD is not compatible with wireless star networks in which all machines communicate through a central AP. This is because a machine may be unable to detect that its transmissions are colliding with those of another. For example, the machine may not hear the other because its own signal drowns out that of the other machine. Another cause might be that the signals from the different machines are blocked from each other by objects or distance even though they can all communicate with the central AP (a condition known as the **hidden terminal problem,** Figure 4.3). The result is that wireless networks adopt the policy of trying to avoid collisions rather than trying to detect them. Such policies are classified as **Carrier Sense, Multiple Access with Collision Avoidance (CSMA/CA),** many of which are standardized by IEEE (see the sidebar "Institute of Electrical and Electronics Engineers" in Chapter 7) within the protocols defined in IEEE 802.11 and commonly referred to as **WiFi.** We emphasize that collision avoidance protocols are designed to avoid collisions and may not eliminate them completely. When collisions do occur, messages must be retransmitted.

The most common approach to collision avoidance is based on giving advantage to machines that have already been waiting for an opportunity to transmit. The protocol used is similar to Ethernet's CSMA/CD. The basic difference is that when a machine first needs to transmit a message and finds the communication channel silent, it does not start transmitting immediately. Instead, it waits for a short period of time and then starts transmitting only if the channel has remained silent throughout that period. If a busy channel is experienced during this process, the machine waits for a randomly determined period before trying again. Once this period is exhausted, the machine is allowed to claim a silent channel

Figure 4.2 Communication over a bus network

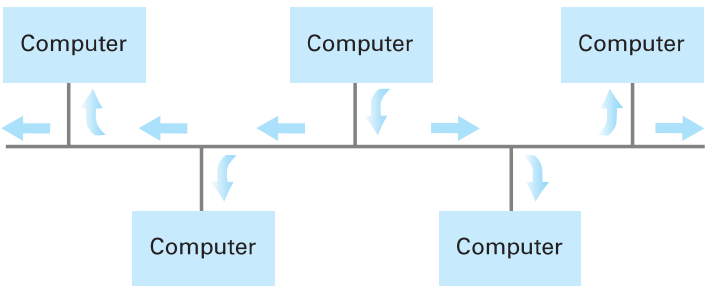
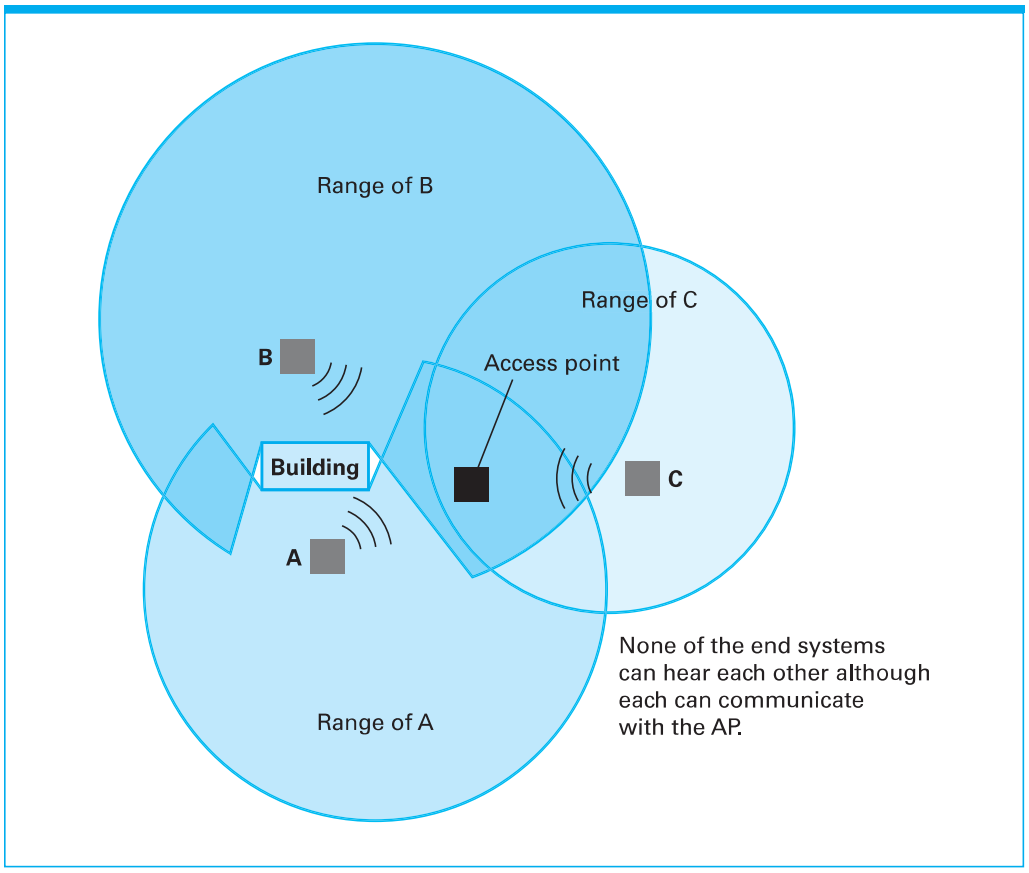


Figure 4.3 The hidden terminal problem



without hesitation. This means that collisions between "newcomers" and those that have already been waiting are avoided because a "newcomer" is not allowed to claim a silent channel until any machine that has been waiting is given the opportunity to start.

This protocol, however, does not solve the hidden terminal problem. After all, any protocol based on distinguishing between a silent or busy channel requires that each individual station be able to hear all the others. To solve this problem, some WiFi networks require that each machine send a short "request" message to the AP and wait until the AP acknowledges that request before transmitting an entire message. If the AP is busy because it is dealing with a "hidden terminal," it will ignore the request, and the requesting machine will know to wait. Otherwise, the AP will acknowledge the request, and the machine will know that it is safe to transmit. Note that all the machines in the network will hear all acknowledgments sent from the AP and thus have a good idea of whether the AP is busy at any given time, even though they may not be able to hear the transmissions taking place.

Combining Networks

Sometimes it is necessary to connect existing networks to form an extended communication system. This can be done by connecting the networks to form a larger version of the same "type" of network. For example, in the case of bus networks

based on the Ethernet protocols, it is often possible to connect the buses to form a single long bus. This is done by means of different devices known as repeaters, bridges, and switches, the distinctions of which are subtle yet informative. The simplest of these is the **repeater,** which is little more than a device that passes signals back and forth between the two original buses (usually with some form of amplification) without considering the meaning of the signals (Figure 4.4a).

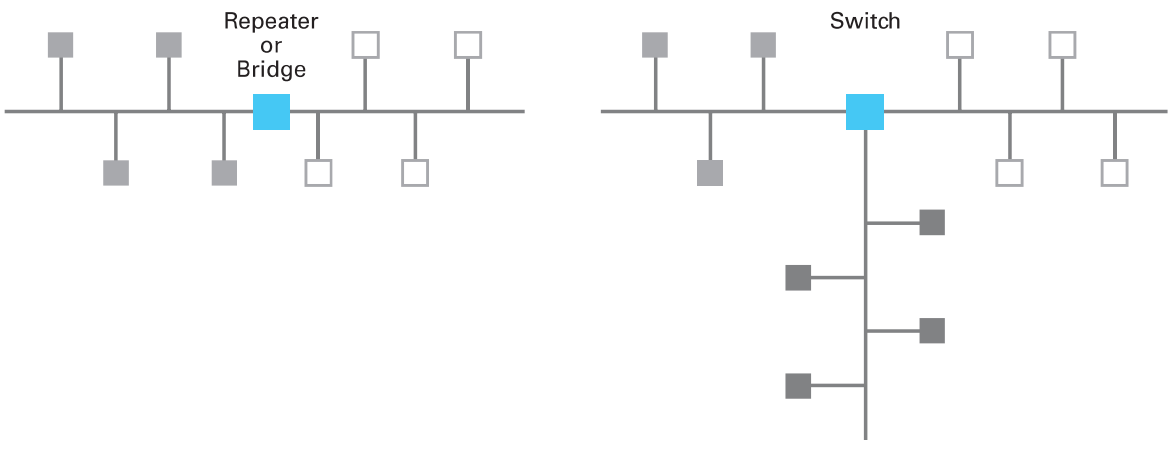
A **bridge** is similar to, but more complex than, a repeater. Like a repeater, it connects two buses, but it does not necessarily pass all messages across the connection. Instead, it looks at the destination address that accompanies each message and forwards a message across the connection only when that message is destined for a computer on the other side. Thus, two machines residing on the same side of a bridge can exchange messages without interfering with communication taking place on the other side. A bridge produces a more efficient system than that produced by a repeater.

A **switch** is essentially a bridge with multiple connections, allowing it to connect several buses rather than just two. Thus, a switch produces a network consisting of several buses extending from the switch as spokes on a wheel (Figure 4.4b). As in the case of a bridge, a switch considers the destination addresses of all messages and forwards only those messages destined for other spokes. Moreover, each message that is forwarded is relayed only into the appropriate spoke, thus minimizing the traffic in each spoke.

It is important to note that when networks are connected via repeaters, bridges, and switches, the result is a single large network. The entire system operates in the same manner (using the same protocols) as each of the original smaller networks.

Sometimes, however, the networks to be connected have incompatible characteristics. For instance, the characteristics of a WiFi network are not readily compatible with an Ethernet network. In these cases the networks must be connected in a manner that builds a network of networks, known as an **internet,** in

Figure 4.4 Building a large bus network from smaller ones



a. A repeater or bridge connecting two buses

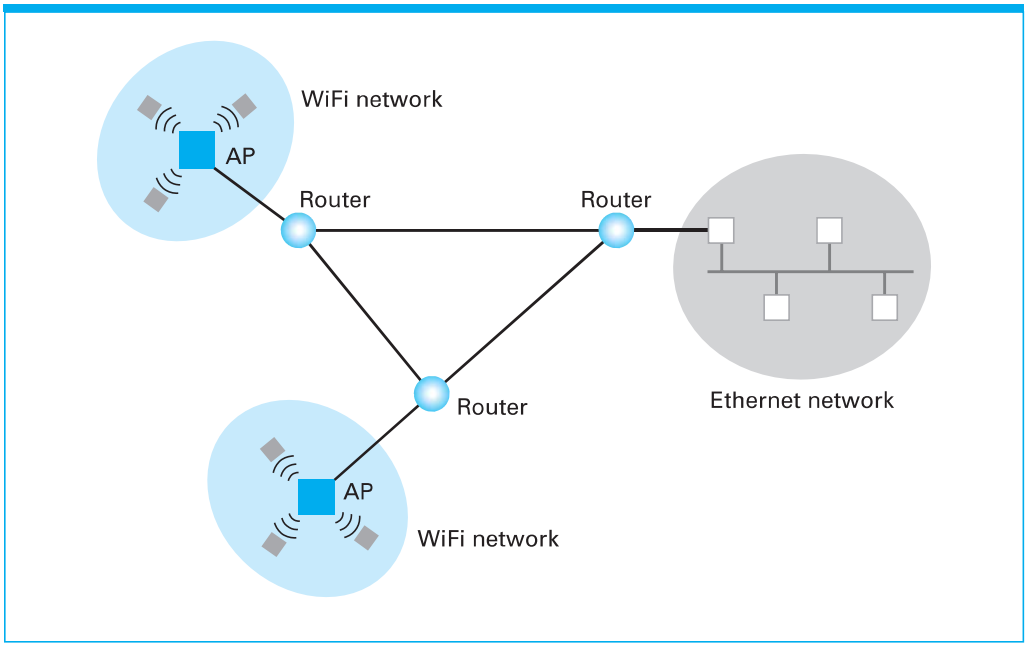
b. A switch connecting multiple buses

which the original networks maintain their individuality and continue to function as autonomous networks. (Note that the generic term *internet* is distinct from *the Internet*. The Internet, written with an uppercase *I*, refers to a particular, worldwide internet that we will study in later sections of this chapter. There are many other examples of internets. Indeed, traditional telephone communication was handled by worldwide internet systems well before the Internet was popularized.)

The connection between networks to form an internet is handled by devices known as **routers,** which are special purpose computers used for forwarding messages. Note that the task of a router is different from that of repeaters, bridges, and switches in that routers provide links between networks while allowing each network to maintain its unique internal characteristics. As an example, Figure 4.5 depicts two WiFi star networks and an Ethernet bus network connected by routers. When a machine in one of the WiFi networks wants to send a message to a machine in the Ethernet network, it first sends the message to the AP in its network. From there, the AP sends the message to its associated router, and this router forwards the message to the router at the Ethernet. There the message is given to a machine on the bus, and that machine then forwards the message to its final destination in the Ethernet.

The reason that routers are so named is that their purpose is to forward messages in their proper directions. This forwarding process is based on an internet-wide addressing system in which all the devices in an internet (including the machines in the original networks and the routers) are assigned unique addresses. (Thus, each machine in one of the original networks has two addresses: its original "local" address within its own network and its internet address.) A machine wanting to send a message to a machine in a distant network attaches the internet address of the destination to the message and directs the message to its local router. From there it is forwarded in the proper direction. For this forwarding

Figure 4.5 Routers connecting two WiFi networks and an Ethernet network to form an internet



purpose, each router maintains a **forwarding table** that contains the router's knowledge about the direction in which messages should be sent depending on their destination addresses.

The "point" at which one network is linked to an internet is often called a **gateway** because it serves as a passageway between the network and the outside world. Gateways can be found in a variety of forms, and thus the term is used rather loosely. In many cases a network's gateway is merely the router through which it communicates with the rest of the internet. In other cases the term *gateway* may be used to refer to more than just a router. For example, in most residential WiFi networks that are connected to the Internet, the term *gateway* refers collectively to both the network's AP and the router connected to the AP because these two devices are normally packaged in a single unit.

Methods of Process Communication

The various activities (or processes) executing on the different computers within a network (or even executing on the same machine via time-sharing/multitasking) must often communicate with each other to coordinate their actions and to perform their designated tasks. Such communication between processes is called **interprocess communication.**

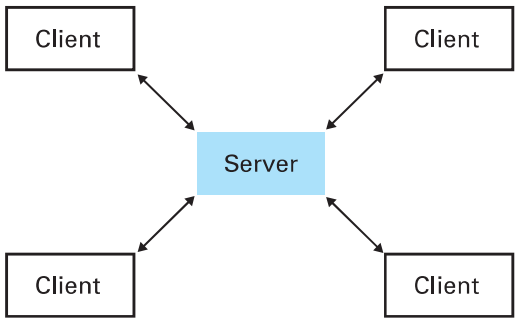
A popular convention used for interprocess communication is the **client/server** model. This model defines the basic roles played by the processes as either a **client,** which makes requests of other processes, or a **server,** which satisfies the requests made by clients.

An early application of the client/server model appeared in networks connecting all the computers in a cluster of offices. In this situation, a single, high-quality printer was attached to the network where it was available to all the machines in the network. In this case the printer played the role of a server (often called a **print server**), and the other machines were programmed to play the role of clients that sent print requests to the print server.

Another early application of the client/server model was used to reduce the cost of magnetic disk storage while also removing the need for duplicate copies of records. Here one machine in a network was equipped with a high-capacity mass storage system (usually a magnetic disk) that contained all of an organization's records. Other machines on the network then requested access to the records as they needed them. Thus the machine that actually contained the records played the role of a server (called a **file server**), and the other machines played the role of clients that requested access to the files that were stored at the file server.

Today the client/server model is used extensively in network applications, as we will see later in this chapter. However, the client/server model is not the only means of interprocess communication. Another model is the **peer-to-peer** (often abbreviated **P2P**) model. Whereas the client/server model involves one process (the server) providing a service to numerous others (clients), the peer-to-peer model involves processes that provide service to and receive service from each other (Figure 4.6). Moreover, whereas a server must execute continuously so that it is prepared to serve its clients at any time, the peer-to-peer model usually involves processes that execute on a temporary basis. For example, applications of the peer-to-peer model include instant messaging in which people carry on a written conversation over the Internet as well as situations in which people play competitive interactive games.

Figure 4.6 The client/server model compared to the peer-to-peer model



a. Server must be prepared to serve multiple clients at any time.

b. Peers communicate as equals on a one-to-one basis.

The peer-to-peer model is also a popular means of distributing files such as music recordings and motion pictures via the Internet. In this case, one peer may receive a file from another and then provide that file to other peers. The collection of peers participating in such a distribution is sometimes called a swarm. The swarm approach to file distribution is in contrast to earlier approaches that applied the client/server model by establishing a central distribution center (the server) from which clients downloaded files (or at least found sources for those files).

One reason that the P2P model is replacing the client/server model for file sharing is that it distributes the service task over many peers rather than concentrating it at one server. This lack of a centralized base of operation leads to a more efficient system. Unfortunately, another reason for the popularity of file distribution systems based on the P2P model is that, in cases of questionable legality, the lack of a central server makes legal efforts to enforce copyright laws more difficult. There are numerous cases, however, in which individuals have discovered that "difficult" does not mean "impossible" and have found themselves faced with significant liabilities due to copyright infringement violations.

You might often read or hear the term *peer-to-peer network*, which is an example of how misuse of terminology can evolve when technical terms are adopted by the nontechnical community. The term *peer-to-peer* refers to a system by which two processes communicate over a network (or internet). It is not a property of the network (or internet). A process might use the peer-to-peer model to communicate with another process and later use the client/server model to communicate with another process over the same network. Thus, it would be more accurate to speak of communicating by means of the peer-to-peer model rather than communicating over a peer-to-peer network.

Distributed Systems

With the success of networking technology, interaction between computers via networks has become common and multifaceted. Many modern software systems, such as global information retrieval systems, company-wide accounting and inventory systems, computer games, and even the software that controls a network's infrastructure itself are designed as **distributed systems,** meaning that they consist of software units that execute as processes on different computers.

Early distributed systems were developed independently from scratch. But today, research is revealing a common infrastructure running throughout these systems, including such things as communication and security systems. In turn, efforts have been made to produce prefabricated systems that provide this basic infrastructure and therefore allow distributed applications to be constructed by merely developing the part of the system that is unique to the application.

Several types of distributed computing systems are now common. **Cluster computing** describes a distributed system in which many independent computers work closely together to provide computation or services comparable to a much larger machine. The cost of these individual machines, plus the high-speed network to connect them, can be less than a higher-priced supercomputer, but with higher reliability and lower maintenance costs. Such distributed systems are used to provide **high-availability**—because it is more likely that at least one member of the cluster will be able to answer a request, even if other cluster members break down or are unavailable—and **load-balancing**—because the workload can be shifted automatically from members of the cluster that have too much to do to those that may have too little. **Grid computing** refers to distributed systems that are more loosely coupled than clusters but that still work together to accomplish large tasks. Grid computing can involve specialized software to make it easier to distribute data and algorithms to the machines participating in a grid. Examples include University of Wisconsin's Condor system, or Berkeley's Open Infrastructure for Network Computing (BOINC). Both of these systems are often installed on computers that are used for other purposes, such as PCs at work or at home, that can then volunteer computing power to the grid when the machine is not otherwise being used. Enabled by the growing connectivity of the Internet, this type of voluntary, distributed grid computing has enabled millions of home PCs to work on enormously complex mathematical and scientific problems. **Cloud computing,** whereby huge pools of shared computers on the network can be allocated for use by clients as needed, is the latest trend in distributed systems. Much as the spread of metropolitan electrical grids in the early twentieth century eliminated the need for individual factories and businesses to maintain their own generators, the Internet is making it possible for entities to entrust their data and computations to "the Cloud," which in this case refers to the enormous computing resources already available on the network. Services such as Amazon's Elastic Compute Cloud allow clients to rent virtual computers by the hour, without concern for where the computer hardware is actually located. Google Drive and Google Apps allow users to collaborate on information or build Web services without needing to know how many computers are working on the problem or where the relevant data are stored. Cloud computing services provide reasonable guarantees of reliability and scalability, but also raise concerns about privacy and security in a world where we may no longer know who owns and operates the computers that we use.

Questions & Exercises

1. What is an open network?
2. Summarize the distinction between a bridge and a switch.
3. What is a router?
4. Identify some relationships in society that conform to the client/server model.
5. Identify some protocols used in society.
6. Summarize the distinction between cluster computing and grid computing.

4.2 The Internet

The most notable example of an internet is the **Internet** (note the uppercase I), which originated from research projects going back to the early 1960s. The goal was to develop the ability to link a variety of computer networks so that they could function as a connected system that would not be disrupted by local disasters. Much of this work was sponsored by the U.S. government through the Defense Advanced Research Projects Agency (DARPA—pronounced "DAR–pa"). Over the years, the development of the Internet shifted from a government-sponsored project to an academic research project, and today it is largely a commercial undertaking that links a worldwide combination of PANs, LANs, MANs, and WANs involving millions of computers.

Internet Architecture

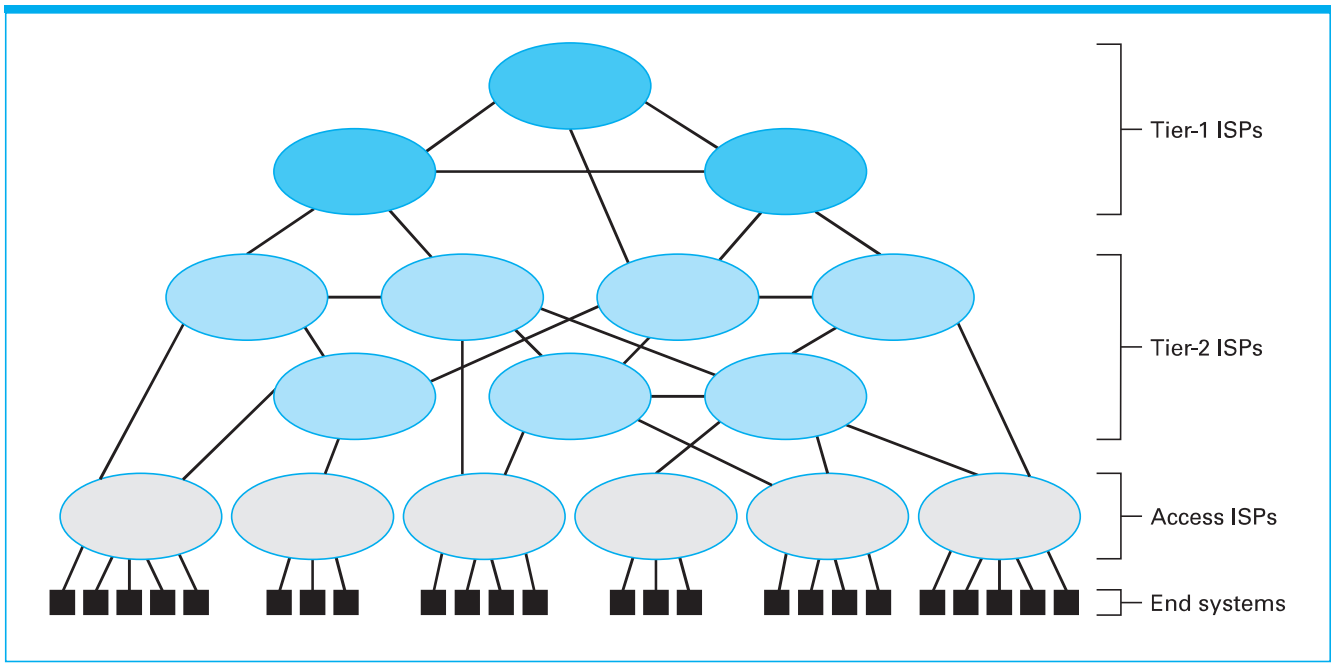
As we have already mentioned, the Internet is a collection of connected networks. In general, these networks are constructed and maintained by organizations called **Internet Service Providers (ISPs).** It is also customary to use the term ISP in reference to the networks themselves. Thus, we will speak of connecting to an ISP, when what we really mean is connecting to the network provided by an ISP.

The system of networks operated by the ISPs can be classified in a hierarchy according to the role they play in the overall Internet structure (Figure 4.7). At the top of this hierarchy are relatively few **tier-1 ISPs** that consist of very high-speed, high-capacity, international WANs. These networks are thought of as the backbone of the Internet. They are typically operated by large companies that are in the communications business. An example would be a company that originated as a traditional telephone company and has expanded its scope into providing other communication services.

Connecting to the tier-1 ISPs are the **tier-2 ISPs** that tend to be more regional in scope and less potent in their capabilities. (The distinction between the tier-1 and tier-2 ISPs is often a matter of opinion.) Again, these networks tend to be operated by companies in the communications business.

Tier-1 and tier-2 ISPs are essentially networks of routers that collectively provide the Internet's communication infrastructure. As such, they can be

Figure 4.7 Internet composition



thought of as the core of the Internet. Access to this core is usually provided by an intermediary called an **access** or **tier-3 ISP.** An access ISP is essentially an independent internet, sometimes called an **intranet,** operated by a single authority that is in the business of supplying Internet access to individual homes and businesses. Examples include cable and telephone companies that charge for their service as well as organizations such as universities or corporations that take it upon themselves to provide Internet access to individuals within their organizations.

The devices that individual users connect to the access ISPs are known as **end systems** or **hosts.** These end systems may be laptops or PCs, but increasingly range over a multitude of other devices including telephones, video cameras, automobiles, and home appliances. After all, the Internet is essentially a communications system, and thus any device that would benefit from communicating with other devices is a potential end system.

The technology by which end systems connect to larger networks is also varied. Perhaps the fastest growing are wireless connections based on WiFi technology. The strategy is to connect the AP to an access ISP and thus provide Internet access through that ISP to end systems within the AP's broadcast range. The area within the AP or group of APs' range is often called a **hot spot,** particularly when the network access is publicly available or free. Hot spots can be found in individual residences, hotel and office buildings, small businesses, parks, and in some cases span entire cities. A similar technology is used by the cellular telephone industry where hot spots are known as cells and the "routers" generating the cells are coordinated to provide continuous service as an end system moves from one cell to another.

Internet2

Now that the Internet has shifted from a research project to a household commodity, the research community has moved on to a project called Internet2. Internet2 is intended as an academic-only system and involves numerous universities working in partnership with industry and government. The goal is to conduct research in internet applications requiring high bandwidth communication, such as remote access and control of costly state-of-the-art equipment such as telescopes and medical diagnostic devices. An example of current research involves remote surgery performed by robot hands that mimic the hands of a distant surgeon who views the patient by video. You can learn more about Internet2 at http://www.internet2.org.

Other popular techniques for connecting to access ISP's use telephone lines or cable/satellite systems. These technologies may be used to provide direct connection to an end system or to a customer's router to which multiple end systems are connected. This latter tactic is popular for individual residences where a local hot spot is created by a router/AP connected to an access ISP by means of existing cable or telephone lines.

Telephone, cable television, and satellite wide area networks of the twentieth century were designed to carry analog communications, such as the human voice or pre-digital television signals. Modern networks can be designed to carry digital data directly between computers, but older, analog network infrastructure still comprises a significant portion of the Internet. Issues arising from these legacy analog linkages are often referred to collectively as the **last mile problem.** The main arteries of WANs, MANs, and many LANs are relatively easy to modernize with high-speed digital technology such as fiber optics, but it can be far costlier to replace the existing copper telephone lines and coaxial cables that connect these arteries to each individual home or office. Thus, information that originated on the Internet continents away from an end system may spend almost its entire trip on high-speed digital connections, only to traverse the "last mile" to the end system over a slow, century-old analog phone line. As alluded to in Chapter 2, several clever schemes have been developed to extend these legacy analog links to accommodate transmission of digital data. DSL modems, cable modems, satellite uplinks, and even direct fiber-optic connections to the home are used to bring broadband Internet access to end users.

Internet Addressing

As we learned in Section 4.1, an internet needs an internet-wide addressing system that assigns a unique identifying address to each computer in the system. In the Internet these addresses are known as **IP addresses.** (The term *IP* refers to "Internet Protocol," which is a term we will learn more about in Section 4.4.) Originally, each IP address was a pattern of 32 bits, but to provide a larger set of addresses, the process of converting to 128-bit addresses is currently underway (see the discussion of IPv6 in Section 4.4). Blocks of consecutively

numbered IP addresses are awarded to ISPs by the **Internet Corporation for Assigned Names and Numbers (ICANN),** which is a nonprofit corporation established to coordinate the Internet's operation. The ISPs are then allowed to allocate the addresses within their awarded blocks to machines within their region of authority. Thus, machines throughout the Internet are assigned unique IP addresses.

IP addresses are traditionally written in **dotted decimal notation** in which the bytes of the address are separated by periods and each byte is expressed as an integer represented in traditional base 10 notation. For example, using dotted decimal notation, the pattern 5.2 would represent the two-byte bit pattern 0000010100000010, which consists of the byte 00000101 (represented by 5) followed by the byte 00000010 (represented by 2), and the pattern 17.12.25 would represent the three-byte bit pattern consisting of the byte 00010001 (which is 17 written in binary notation), followed by the byte 00001100 (12 written in binary), followed by the byte 00011001 (25 written in binary). In summary, a 32-bit IP address might appear as 192.207.177.133 when expressed in dotted decimal notation.

Addresses in bit-pattern form (even when compressed using dotted decimal notation) are rarely conducive to human consumption. For this reason the Internet has an alternative addressing system in which machines are identified by mnemonic names. This addressing system is based on the concept of a **domain,** which can be thought of as a "region" of the Internet operated by a single authority such as a university, club, company, or government agency. (The word region is in quotations here because, as we will soon see, such a region may not correspond to a physical area of the Internet.) Each domain must be registered with ICANN—a process handled by companies, called **registrars,** that have been assigned this role by ICANN. As a part of this registration process, the domain is assigned a mnemonic **domain name,** which is unique among all the domain names throughout the Internet. Domain names are often descriptive of the organization registering the domain, which enhances their utility for humans.

As an example, the domain name of Marquette University is `mu.edu`. Note the suffix following the period. It is used to reflect the domain's classification, which in this case is "educational" as indicated by the `edu` suffix. These suffixes are called **top-level domains (TLDs).** Other TLDs include `com` for commercial institutions, `gov` for U.S. government institutions, `org` for nonprofit organizations, `museum` for museums, `info` for unrestricted use, and `net`, which was originally intended for ISPs but is now used on a much broader scale. In addition to these general TLDs, there are also two-letter TLDs for specific countries (called **country-code TLDs**) such as `au` for Australia and `ca` for Canada.

Once a domain's mnemonic name is registered, the organization that registered the name is free to extend the name to obtain mnemonic identifiers for individual items within the domain. For example, an individual host within Marquette University may be identified as `eagle.mu.edu`. Note that domain names are extended to the left and separated by a period. In some cases multiple extensions, called **subdomains,** are used as a means of organizing the names within a domain. These subdomains often represent different networks within the domain's jurisdiction. For example, if Yoyodyne Corporation was assigned the domain name `yoyodyne.com`, then an individual computer at Yoyodyne might have a name such as `overthruster.propulsion.yoyodyne.com`, meaning that

the computer `overthruster` is in the subdomain `propulsion` within the domain `yoyodyne` within the TLD `com`. (We should emphasize that the dotted notation used in mnemonic addresses is not related to the dotted decimal notation used to represent addresses in bit pattern form.)

Although mnemonic addresses are convenient for humans, messages are always transferred over the Internet by means of IP addresses. Thus, if a human wants to send a message to a distant machine and identifies the destination by means of a mnemonic address, the software being used must be able to convert that address into an IP address before transmitting the message. This conversion is performed with the aid of numerous servers, called **name servers,** that are essentially directories that provide address translation services to clients. Collectively, these name servers are used as an Internet-wide directory system known as the **domain name system (DNS).** The process of using DNS to perform a translation is called a **DNS lookup.**

Thus, for a machine to be accessible by means of a mnemonic domain name, that name must be represented in a name server within the DNS. In those cases in which the entity establishing the domain has the resources, it can establish and maintain its own name server containing all the names within that domain. Indeed, this is the model on which the domain system was originally based. Each registered domain represented a physical region of the Internet that was operated by a local authority such as a company, university, or government agency. This authority was essentially an access ISP that provided Internet access to its members by means of its own intranet that was linked to the Internet. As part of this system, the organization maintained its own name server that provided translation services for all the names used within its domain.

This model is still common today. However, many individuals or small organizations want to establish a domain presence on the Internet without committing the resources necessary to support it. For example, it might be beneficial for a local chess club to have a presence on the Internet as `KingsandQueens.org`, but the club would likely not have the resources to establish its own network, maintain a link from this network to the Internet, and implement its own name server. In this case, the club can contract with an access ISP to create the appearance of a registered domain using the resources already established by the ISP. Typically, the club, perhaps with the assistance of the ISP, registers the name chosen by the club and contracts with the ISP to have that name included in the ISP's name server. This means that all DNS lookups regarding the new domain name will be directed to the ISP's name server, from which the proper translation will be obtained. In this way, many registered domains can reside within a single ISP, each often occupying only a small portion of a single computer.

Internet Applications

In the earlier days of the Internet, most applications were separate, simple programs that each followed a network protocol. A newsreader application contacted servers using the **Network News Transfer Protocol (NNTP),** an application for listing and copying files across the network implemented the **File Transfer Protocol (FTP),** or an application for accessing another computer from a great distance used the **Telnet** protocol, or later the **Secure Shell (SSH)** protocol. As webservers and browsers have become more sophisticated, more and more

of these traditional network applications have come to be handled by webpages via the powerful **Hyper Text Transfer Protocol (HTTP).** Nevertheless, when examining a network protocol for an Internet application for the first time, it behooves us to begin with a few simpler examples before moving on to HTTP in the next section.

Electronic Mail A wide variety of systems now exist for exchanging messages between end users over the network; instant messaging (IM), browser-based online chatting, Twitter-based "tweets", and the Facebook "wall" are but a few. One of the oldest and most enduring uses of the Internet is the electronic mail system, or **email** for short. While many users now rely on their browser or a sophisticated application like Microsoft's Outlook, Apple's Mail, or Mozilla's Thunderbird to read and compose their email, the actual transmission of email messages from one computer to another on the Internet remains the domain of basic network protocols like SMTP.

SMTP (Simple Mail Transfer Protocol) defines a way that two computers on the network may interact when transmitting an email message from one host to the other. Consider the example case of a **mail server** `mail.skaro.gov` sending an email from end user "`dalek`" to end user "`doctor`" in the domain `tardis.edu`. First, a mail handling process on `mail.skaro.gov` contacts the mail server process on `mail.tardis.edu`. To accomplish this, it uses DNS, another network protocol, to map the human-readable destination domain name to the proper mail server name, and then to its IP address. This is not unlike looking up the phone number of an acquaintance before dialing. Similarly, when the server process at the other end answers, the protocol states that it must identify itself to the caller. The transcript of their SMTP exchange might look something like this:

```
 1  220 mail.tardis.edu SMTP Sendmail Gallifrey-1.0; Fri, 23
      Aug 2413 14:34:10
 2  HELO mail.skaro.gov
 3  250 mail.tardis.edu Hello mail.skaro.gov, pleased to meet you
 4  MAIL From: dalek@skaro.gov
 5  250 2.1.0 dalek@skaro.gov... Sender ok
 6  RCPT To: doctor@tardis.edu
 7  250 2.1.5 doctor@tardis.edu... Recipient ok
 8  DATA
 9  354 Enter mail, end with "." on a line by itself
10  Subject: Extermination.
11
12  EXTERMINATE!
13  Regards, Dalek
14  .
15  250 2.0.0 r7NJYAEl028071 Message accepted for delivery
16  QUIT
17  221 2.0.0 mail.tardis.edu closing connection
```

In line 1, the remote mail server process answers the caller by announcing its name, the protocol it speaks, and other optional information, such as the version of protocol, and the date and time. In line 2, the sending mail server

process introduces itself. In line 3, the remote server acknowledges the name of the sending server.

Most Internet protocols are not necessarily transmitted in ASCII characters so easily interpreted by a human. To be sure, the quaint politeness of "`pleased to meet you`" in line 3 is neither appreciated by the software on either end of this connection, nor essential to SMTP's proper function. However, this is actual behavior built into one popular SMTP mail server, held over from the early days of the Internet when human operators frequently needed to review SMTP transcripts to debug incompatibilities between mail servers. Untold millions of these exchanges occur on the network each day, known only to the software agents that transport email across the Internet.

In the simplest case, the remote mail server will take `mail.skaro.gov` at its word in line 2, accepting that the sending server is the machine named `mail` from the domain `skaro.gov`. SMTP is an example of a protocol originally built on trust that has subsequently been abused by spammers and other Internet miscreants. Modern mail servers must use extended versions of SMTP or their equivalent to help ensure that email is transmitted securely. We discuss concerns about network security in greater detail in the final segment of this chapter.

Returning to the transcript, in line 4 the sending server announces that it has a mail message to deliver, and identifies the sending user. In line 5, the remote server acknowledges that it will receive mail from this user at this domain. In line 6, the sending server announces the recipient on the remote server. In line 7, the remote server acknowledges that it will receive email destined for that user. In line 8, the sending server dispenses with the introductions, and announces that it is ready to send the `DATA`, the actual body of the e-mail message. In line 8, the remote server acknowledges (with code 354, in accordance with the SMTP protocol) that it is ready to receive the body of the message and includes a helpful human-readable instruction on how to conclude the message transfer.

In lines 10 through 14, the sending server conveys the text of the email message to be delivered. The remote server acknowledges acceptance in line 15, and in line 17 acknowledges the sending server's `QUIT` announcement from line 16.

The technical documents describing SMTP define each of the allowed steps in a conversation such as the transcript above. The keywords `HELO`, `MAIL`, `RCPT`, `DATA`, and `QUIT` are each precisely defined in terms of how they will be sent, what options can accompany them, and how they should be interpreted. Similarly, the remote server's numeric response codes for acknowledgment are enumerated and defined. Software designers use a protocol description to develop algorithms that will correctly implement sending and receiving email over the network.

Other protocols come into play for other aspects of transporting email. Because SMTP was initially designed for transferring text messages encoded with ASCII, additional protocols such as **MIME (Multipurpose Internet Mail Extensions)** have been developed to convert non-ASCII data to SMTP compatible form.

There are two popular protocols that may be used for accessing email that has arrived and accumulated at a user's mail server. These are **POP3 (Post Office Protocol version 3)** and **IMAP (Internet Mail Access Protocol).** POP3 (pronounced "pop-THREE") is the simpler of the two. Using POP3, a user transfers (downloads) messages to his or her local computer where they can be read, stored in various folders, edited, and otherwise manipulated as the user desires. This is done on the user's local machine using the local machine's mass storage.

IMAP (pronounced "EYE-map") allows a user to store and manipulate messages and related materials on the same machine as the mail server. In this manner, a user who must access his or her email from different computers can maintain records at the mail server that are then accessible from any remote computer to which the user may have access.

VoIP As an example of a more recent Internet application, consider **VoIP (Voice over Internet Protocol)** in which the Internet infrastructure is used to provide voice communication similar to that of traditional telephone systems. In its simplest form, VoIP consists of two processes on different machines transferring audio data via the P2P model—a process that in itself presents no significant problems. However, tasks such as initiating and receiving calls, linking VoIP with traditional telephone systems, and providing services such as emergency 911 communication are issues that extend beyond traditional Internet applications. Moreover, governments that own their country's traditional telephone companies view VoIP as a threat and have either taxed it heavily or outlawed it completely.

Existing VoIP systems come in four different forms that are competing for popularity. VoIP **soft phones** consist of P2P software that allows two or more PCs to share a call with no more special hardware than a speaker and a microphone. An example of a VoIP soft phone system is Skype, which also provides its clients with links to the traditional telephone communication system. One drawback to Skype is that it is a proprietary system, and thus much of its operational structure is not publicly known. This means that Skype users must trust the integrity of the Skype software without third-party verification. For instance, to receive calls, a Skype user must leave his or her PC connected to the Internet and available to the Skype system, which means that some of the PC's resources may be used to support other Skype communications without the PC owner's awareness—a feature that has generated some resistance.

A second form of VoIP consists of **analog telephone adapters,** which are devices that allow a user to connect his or her traditional telephone to phone service provided by an access ISP. This choice is frequently bundled with traditional Internet service and/or digital television service.

The third type of VoIP comes in the form of embedded VoIP phones, which are devices that replace a traditional telephone with an equivalent handset connected directly to a TCP/IP network. Embedded VoIP phones are becoming increasingly common for large organizations, many of whom are replacing their traditional internal copper wire telephone systems with VoIP over Ethernet to reduce costs and enhance features.

Finally, the current generation of smartphones use wireless VoIP technology. That is, earlier generations of wireless phones only communicated with the telephone company's network using that company's protocols. Access to the Internet was obtained by gateways between the company's network and the Internet, at which point signals were converted to the TCP/IP system. However, the 4G phone network is an IP-based network throughout, which means a 4G telephone is essentially just another broadband-connected host computer on the global Internet.

Internet Multimedia Streaming An enormous portion of current Internet traffic is used for transporting audio and video across the Internet in real-time, known as **streaming.** Netflix streamed more than 4 billion hours of programming to end users in the first three months of 2013 alone. Combined with YouTube,

The Generations of Wireless Telephones

Mobile phone technology has evolved rapidly since the first simple, handheld electronic units were introduced in the 1980s. An entirely new wave of phone technology has emerged roughly every 10 years since that time, leading up to our current complex, multifunction smartphones. The first generation wireless telephone network transmitted analog voice signals through the air, much like traditional telephones but without the copper wire running into the wall. In retrospect, we call these early phones "1G," or first generation. The second generation used digital signals to encode voice, providing more effective use of the airwaves, and the transmission of other kinds of digital data, like text messaging. The third generation ("**3G**") phone network provided higher data rates, allowing for mobile video calls and other bandwidth-intensive activities. The **4G** network provides even higher data rates and a fully packet-switched IP network, which has allowed smartphones to enjoy the connectivity and flexibility previously available only to broadband-enabled PCs.

these two services will consume more than half of the bandwidth of the Internet in 2014.

On the surface, Internet streaming may not seem to require special consideration. For example, one might guess that an Internet radio station could merely establish a server that would send program messages to each of the clients who requested them. This technique is known as **N-unicast.** (More precisely, *unicast* refers to one sender sending messages to one receiver, whereas *N-unicast* refers to a single sender involved with multiple unicasts.) The N-unicast approach has been applied but has the drawback of placing a substantial burden on the station's server as well as on the server's immediate Internet neighbors. Indeed, N-unicast forces the server to send individual messages to each of its clients on a real-time basis, and all these messages must be forwarded by the server's neighbors.

Most alternatives to N-unicast represent attempts to alleviate this problem. One applies the P2P model in a manner reminiscent of file-sharing systems. That is, once a peer has received data, it begins to distribute that data to those peers that are still waiting, meaning that much of the distribution problem is transferred from the data's source to the peers.

Another alternative, called **multicast,** transfers the distribution problem to the Internet routers. Using multicast, a server transmits a message to multiple clients by means of a single address and relies on the routers in the Internet to recognize the significance of that address and to produce and forward copies of the message to the appropriate destinations. Note then that applications relying on multicast require that the functionality of the Internet routers be expanded beyond their original duties. Multicast support has been implemented in small networks, but has yet to expand to the global Internet.

More importantly, most applications in this category are now **on-demand streaming,** in which the end user expects to view or listen to media at an arbitrary time of his or her choosing. This is quite a different problem from the Internet radio station example, because each end user expects to be able to start, pause, or rewind content at his or her own pace. In this case, N-unicast and multicast technologies are of little help. Each on-demand stream is effectively unicast from a media server that stores the content to the end user that wishes to retrieve it.

In order for this type of streaming to scale to thousands or even millions of simultaneous users, each with his or her own personal stream, replication of the content to many distinct servers is essential. Large-scale streaming services make use of **content delivery networks (CDNs),** groups of servers distributed strategically around the Internet that specialize in streaming copies of content to nearby end users in their network "neighborhood." In many cases, CDN machines may reside in an access ISP network, allowing customers of that access ISP to stream copies of multimedia content at high speed from a nearby server that is much closer in the network than the streaming service's central server machines. A networking technology called **anycast,** which enables an end user to automatically connect to the closest server out of a defined group of servers, helps to make CDNs practical.

Internet streaming of high-definition, on-demand video has permeated far more than traditional PCs. A broad class of embedded devices such as televisions, DVD/Blu-ray players, smartphones, and game consoles connect directly to the TCP/IP network to select viewable content from a multitude of both free and subscription servers.

Questions & Exercises

1. What is the purpose of tier-1 and tier-2 ISPs? What is the purpose of access ISPs?
2. What is DNS?
3. What bit pattern is represented by 3.6.9 in dotted decimal notation? Express the bit pattern 0001010100011100 using dotted decimal notation.
4. In what way is the structure of a mnemonic address of a computer on the Internet (such as `overthruster.propulsion.yoyodyne.com`) similar to a traditional postal address? Does this same structure occur in IP addresses?
5. Name three types of servers found on the Internet and tell what each does.
6. What aspects of network communication are described by a protocol?
7. In what way do the P2P and multicast approaches to Internet radio broadcast differ from N-unicast?
8. What criteria should one consider when choosing one of the four types of VoIP?

4.3 The World Wide Web

The World Wide Web had its origins in the work of Tim Berners-Lee who realized the potential of combining internet technology with the concept of linked-documents, called **hypertext.** His first software for implementing the Web was released in December, 1990. While this early prototype did not yet support multimedia data, it included the key components of what we now recognize as the World Wide Web: a hypertext document format for embedding **hyperlinks** to

other documents; a protocol for transferring hypertext across the network, and a server process that supplied hypertext pages upon request. From this humble beginning, the Web quickly grew to support images, audio and video, and by the mid-1990s had become the dominant application powering the growth of the Internet.

Web Implementation

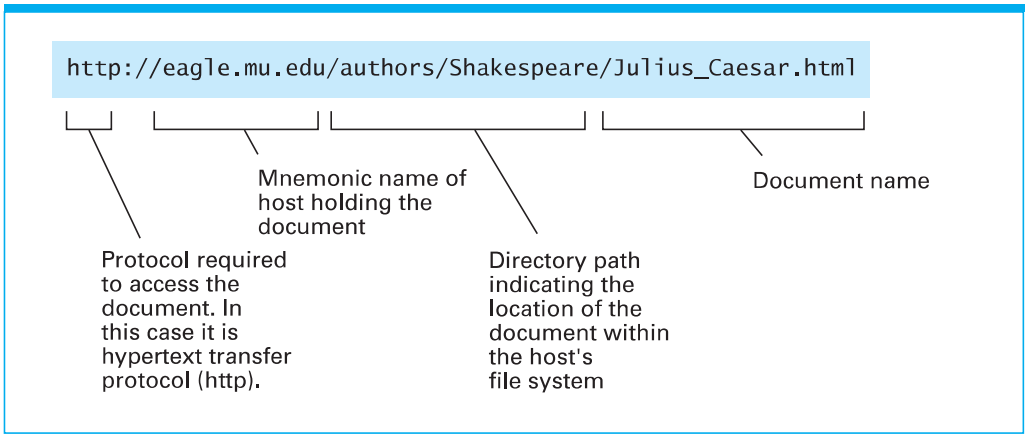
Software packages that allow users to access hypertext on the Internet fall into one of two categories: **browsers** and **webservers.** A browser resides on the user's computer and is charged with the tasks of obtaining materials requested by the user and presenting these materials to the user in an organized manner. Common Internet browsers include Firefox, Safari, and Internet Explorer. The webserver resides on a computer containing hypertext documents to be accessed. Its task is to provide access to the documents under its control as requested by clients (browsers). Hypertext documents are normally transferred between browsers and webservers using a protocol known as the Hypertext Transfer Protocol (HTTP).

In order to locate and retrieve documents on the Web, each document is given a unique address called a **Uniform Resource Locator (URL).** Each URL contains the information needed by a browser to contact the proper server and request the desired document. Thus to view a webpage, a person first provides his or her browser with the URL of the desired document and then instructs the browser to retrieve and display the document.

A typical URL is presented in Figure 4.8. It consists of four segments: the protocol to use to communicate with the server controlling access to the document, the mnemonic address of the machine containing the server, the directory path needed for the server to find the directory containing the document, and the name of the document itself. In short, the URL in Figure 4.8 tells a browser to contact the webserver on the computer known as `eagle.mu.edu` using the protocol `HTTP` and to retrieve the document named `Julius_Caesar.html` found within the subdirectory `Shakespeare` within the directory called `authors`.

Sometimes a URL might not explicitly contain all the segments shown in Figure 4.8. For example, if the server does not need to follow a directory path

Figure 4.8 A typical URL



to reach the document, no directory path will appear in the URL. Moreover, sometimes a URL will consist of only a protocol and the mnemonic address of a computer. In these cases, the webserver at that computer will return a predetermined document, typically called a home page, that usually describes the information available at that website. Such shortened URLs provide a simple means of contacting organizations. For example, the URL `http://www.google.com` will lead to the home page of Google, which contains hyperlinks to the services, products, and documents relating to the company.

To further simplify locating websites, many browsers assume that the HTTP protocol should be used if no protocol is identified. These browsers correctly retrieve the Google home page when given the "URL" consisting merely of `www.google.com`.

HTML

A traditional hypertext document is similar to a text file because its text is encoded character by character using a system such as ASCII or Unicode. The distinction is that a hypertext document also contains special symbols, called **tags,** that describe how the document should appear on a display screen, what multimedia resources (such as images) should accompany the document, and which items within the document are linked to other documents. This system of tags is known as **Hypertext Markup Language (HTML).**

Thus, it is in terms of HTML that an author of a webpage describes the information that a browser needs in order to present the page on the user's screen and to find any related documents referenced by the current page. The process is analogous to adding typesetting directions to a plain typed text (perhaps using a red pen) so that a typesetter will know how the material should appear in its final form. In the case of hypertext, the red markings are replaced by HTML tags, and a browser ultimately plays the role of the typesetter, reading the HTML tags to learn how the text is to be presented on the computer screen.

The HTML-encoded version (called the **source** version) of an extremely simple webpage is shown in Figure 4.9a. Note that the tags are delineated by the symbols < and >. The HTML source document consists of two sections—a head (surrounded by the `<head>` and `</head>` tags) and a body (surrounded by the `<body>` and `</body>` tags). The distinction between the head and body of a webpage is similar to that of the head and body of an interoffice memo. In both cases, the head contains preliminary information about the document (date, subject,

The World Wide Web Consortium

The World Wide Web Consortium (W3C) was formed in 1994 to promote the World Wide Web by developing protocol standards (known as W3C standards). W3C is headquartered at CERN, the high-energy particle physics laboratory in Geneva, Switzerland. CERN is where the original HTML markup language was developed as well as the HTTP protocol for transferring HTML documents over the Internet. Today W3C is the source of many standards (including standards for XML and numerous multimedia applications) that lead to compatibility over a wide range of Internet products. You can learn more about W3C via its website at http://www.w3c.org.

etc. in the case of a memo). The body contains the meat of the document, which in the case of a webpage is the material to be presented on the computer screen when the page is displayed.

The head of the webpage displayed in Figure 4.9a contains only the title of the document (surrounded by "title" tags). This title is only for documentation purposes; it is not part of the page that is to be displayed on the computer screen. The material that is displayed on the screen is contained in the body of the document.

The first entry in the body of the document in Figure 4.9a is a level-one heading (surrounded by the `<h1>` and `</h1>` tags) containing the text "My Web Page." Being a level-one heading means that the browser should display this text prominently on the screen. The next entry in the body is a paragraph of text (surrounded by the `<p>` and `</p>` tags) containing the text "Click here for another page." Figure 4.9b shows the page as it would be presented on a computer screen by a browser.

In its present form, the page in Figure 4.9 is not fully functional in the sense that nothing will happen when the viewer clicks on the word *here*,

Figure 4.9 A simple webpage

a. The page encoded using HTML.

Tag indicating beginning of document

Preliminaries

The part of the document that will be displayed by a browser

Tag indicating end of document

```
<html>
<head>
<title>demonstration page</title>
</head>
<body>
<h1>My Web Page</h1>
<p>Click here for another page.</p>
</body>
</html>
```

b. The page as it would appear on a computer screen.

even though the page implies that doing so will cause the browser to display another page. To cause the appropriate action, we must link the word *here* to another document.

Let us suppose that, when the word *here* is clicked, we want the browser to retrieve and display the page at the URL `http://crafty.com/demo.html`. To do so, we must first surround the word *here* in the source version of the page with the tags `<a>` and `</a>`, which are called anchor tags. Inside the opening anchor tag we insert the parameter

```
href = http://crafty.com/demo.html
```

(as shown in Figure 4.10a) indicating that the hypertext reference (`href`) associated with the tag is the URL following the equal sign (`http://crafty.com/demo.html`). Having added the anchor tags, the webpage will now appear on a computer screen as shown in Figure 4.10b. Note that this is identical to Figure 4.9b except that the word *here* is highlighted by color indicating that it is a link to another webpage. Clicking on such highlighted terms will cause the browser to retrieve and display the associated webpage. Thus, it is by means of anchor tags that webpages are linked to each other.

Finally, we should indicate how an image could be included in our simple webpage. For this purpose, let us suppose that a JPEG encoding of the image we want to include is stored as the file named `OurPic.jpg` in the directory `Images` at `Images.com` and is available via the webserver at that location. Under these conditions, we can tell a browser to display the image at the top of the webpage by inserting the image tag `<img src = "http://Images.com/Images/OurPic.jpg">` immediately after the `<body>` tag in the HTML source document. This tells the browser that the image named `OurPic.jpg` should be displayed at the beginning of the document. (The term `src` is short for "source," meaning that the information following the equal sign indicates the source of the image to be displayed.) When the browser finds this tag, it will send a message to the HTTP server at `Images.com` requesting the image called `OurPic.jpg` and then display the image appropriately.

If we moved the image `tag` to the end of the document just before the `</body>` tag, then the browser would display the image at the bottom of the webpage. There are, of course, more sophisticated techniques for positioning an image on a webpage, but these need not concern us now.

XML

HTML is essentially a notational system by which a text document along with the document's appearance can be encoded as a simple text file. In a similar manner we can also encode nontextual material as text files—an example being sheet music. At first glance the pattern of staffs, measure bars, and notes in which music is traditionally represented does not conform to the character-by-character format dictated by text files. However, we can overcome this problem by developing an alternative notation system. More precisely, we could agree to represent the start of a staff by `<staff clef = "treble">`, the end of the staff by `</staff>`, a time signature with the form `<time> 2/4 </time>`, the beginning and ending of a measure by `<measure>` and `</measure>`, respectively, a

Figure 4.10 An enhanced simple webpage

a. The page encoded using HTML.

```
<html>
<head>
<title>demonstration page</title>
</head>
<body>
<h1>My Web Page</h1>
<p>Click
   <a href="http://crafty.com/demo.html">
   here
   </a>
   for another page.</p>
</body>
</html>
```

Anchor tag containing parameter

Closing anchor tag

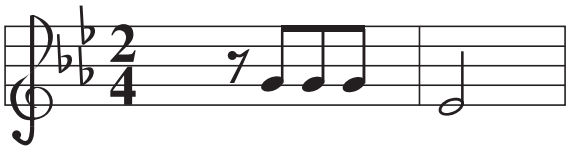
b. The page as it would appear on a computer screen.

note such as an eighth note on C as `<notes> egth C </notes>`, and so on. Then the text

```
<staff clef = "treble"> <key>C minor</key>
<time> 2/4 </time>
<measure> <rest> egth </rest> <notes> egth G,
egth G, egth G </notes></measure>
<measure> <notes> hlf E </notes></measure>
</staff>
```

could be used to encode the music shown in Figure 4.11. Using such notation, sheet music could be encoded, modified, stored, and transferred over the Internet as text files. Moreover, software could be written to present the contents of such files in the form of traditional sheet music or even to play the music on a synthesizer.

Figure 4.11 The first two bars of Beethoven's Fifth Symphony



Note that our sheet music encoding system encompasses the same style used by HTML. We chose to delineate the tags that identify components by the symbols `<` and `>`. We chose to indicate the beginning and end of structures (such as a staff, string of notes, or measure) by tags of the same name—the ending tag being designated by a slash (a `<measure>` was terminated with the tag `</measure>`). And we chose to indicate special attributes within tags by expressions such as `clef = "treble"`. This same style could also be used to develop systems for representing other formats such as mathematical expressions and graphics.

The **eXtensible Markup Language (XML)** is a standardized style (similar to that of our music example) for designing notational systems for representing data as text files. (Actually, XML is a simplified derivative of an older set of standards called the Standard Generalized Markup Language, better known as SGML.) Following the XML standard, notational systems called **markup languages** have been developed for representing mathematics, multimedia presentations, and music. In fact, HTML is the markup language based on the XML standard that was developed for representing webpages. (Actually, the original version of HTML was developed before the XML standard was solidified, and therefore some features of HTML do not strictly conform to XML. That is why you might see references to XHTML, which is the version of HTML that rigorously adheres to XML.)

XML provides a good example of how standards are designed to have wide-ranging applications. Rather than designing individual, unrelated markup languages for encoding various types of documents, the approach represented by XML is to develop a standard for markup languages in general. With this standard, markup languages can be developed for various applications. Markup languages developed in this manner possess a uniformity that allows them to be combined to obtain markup languages for complex applications such as text documents that contain segments of sheet music and mathematical expressions.

Finally, we should note that XML allows the development of new markup languages that differ from HTML in that they emphasize semantics rather than appearance. For example, with HTML the ingredients in a recipe can be marked so that they appear as a list in which each ingredient is positioned on a separate line. But if we used semantic-oriented tags, ingredients in a recipe could be marked as ingredients (perhaps using the tags `<ingredient>` and `</ingredient>`) rather than merely items in a list. The difference is subtle but important. The semantic approach would allow **search engines** (websites that assist users in locating Web material pertaining to a subject of interest) to identify recipes that contain or do not contain certain ingredients, which would be a substantial improvement over the current state of the art in which only recipes that do or do not contain certain words can be isolated. More precisely, if semantic tags are used, a search engine can identify recipes for lasagna that do not contain spinach, whereas a similar search based merely on word content would skip over a recipe that

started with the statement "This lasagna does not contain spinach." In turn, by using an Internet-wide standard for marking documents according to semantics rather than appearance, a World Wide *Semantic* Web, rather than the World Wide *Syntactic* Web we have today, would be created.

Client-Side and Server-Side Activities

Consider now the steps that would be required for a browser to retrieve the simple webpage shown in Figure 4.10 and display it on the browser's computer screen. First, playing the role of a client, the browser would use the information in a URL (perhaps obtained from the person using the browser) to contact the webserver controlling access to the page and ask that a copy of the page be transferred to it. The server would respond by sending the text document displayed in Figure 4.10a to the browser. The browser would then interpret the HTML tags in the document to determine how the page should be displayed and present the document on its computer screen accordingly. The user of the browser would see an image like that depicted in Figure 4.10b. If the user then clicked the mouse over the word *here*, the browser would use the URL in the associated anchor tag to contact the appropriate server to obtain and display another webpage. In summary, the process consists of the browser merely fetching and displaying webpages as directed by the user.

But what if we wanted a webpage involving animation or one that allows a customer to fill out an order form and submit the order? These needs would require additional activity by either the browser or the webserver. Such activities are called **client-side** activities if they are performed by a client (such as a browser) or **server-side** activities if they are performed by a server (such as a webserver).

As an example, suppose a travel agent wanted customers to be able to identify desired destinations and dates of travel, at which time the agent would present the customer with a customized webpage containing only the information pertinent to that customer's needs. In this case the travel agent's website would first provide a webpage that presents a customer with the available destinations. On the basis of this information, the customer would specify the destinations of interest and desired dates of travel (a client-side activity). This information would then be transferred back to the agent's server where it would be used to construct the appropriate customized webpage (a server-side activity), which would then be sent to the customer's browser.

Another example occurs when using the services of a search engine. In this case a user at the client specifies a topic of interest (a client-side activity), which is then transferred to the search engine where a customized webpage identifying documents of possible interest is constructed (a server-side activity) and sent back to the client. Still another example occurs in the case of Web mail—an increasingly popular means by which computer users are able to access their email by means of Web browsers. In this case, the webserver is an intermediary between the client and the client's mail server. Essentially, the webserver builds webpages that contain information from the mail server (a server-side activity) and sends those pages to the client where the client's browser displays them (a client-side activity). Conversely, the browser allows the user to create messages (a client-side activity) and sends that information to the webserver, which then forwards the messages to the mail server (a server-side activity) for mailing.

There are numerous systems for performing client- and server-side activities, each competing with the others for prominence. An early and still popular means

of controlling client-side activities is to include programs written in the language JavaScript (developed by Netscape Communications, Inc.) within the HTML source document for the webpage. From there a browser can extract the programs and follow them as needed. Another approach (developed by Sun Microsystems) is to first transfer a webpage to a browser and then transfer additional program units called applets (written in the language Java) to the browser as requested within the HTML source document. Still another approach is the system Flash (developed by Macromedia) by which extensive multimedia client-side presentations can be implemented.

An early means of controlling server-side activities was to use a set of standards called CGI (Common Gateway Interface) by which clients could request the execution of programs stored at a server. A variation of this approach (developed by Sun Microsystems) is to allow clients to cause program units called servlets to be executed at the server side. A simplified version of the servlet approach is applicable when the requested server-side activity is the construction of a customized webpage, as in our travel agent example. In this case webpage templates called JavaServer Pages (JSP) are stored at the webserver and completed using information received from a client. A similar approach is used by Microsoft, where the templates from which customized webpages are constructed are called Active Server Pages (ASP). In contrast to these proprietary systems, PHP (originally standing for Personal Home Page but now considered to mean PHP Hypertext Preprocessor) is an open source system for implementing server-side functionality.

Finally, we would be remiss if we did not recognize the security and ethical problems that arise from allowing clients and servers to execute programs on the other's machine. The fact that webservers routinely transfer programs to clients where they are executed leads to ethical questions on the server side and security questions on the client side. If the client blindly executes any program sent to it by a webserver, it opens itself to malicious activities by the server. Likewise, the fact that clients can cause programs to be executed at the server leads to ethical questions on the client side and security questions on the server side. If the server blindly executes any program sent to it by a client, security breaches and potential damage at the server could result.

Questions & Exercises

1. What is a URL? What is a browser?
2. What is a markup language?
3. What is the difference between HTML and XML?
4. What is the purpose of each of the following HTML tags?
   a. `<html>`
   b. `<head>`
   c. `</p>`
   d. `</a>`
5. To what do the terms *client side* and *server side* refer?

4.4 Internet Protocols

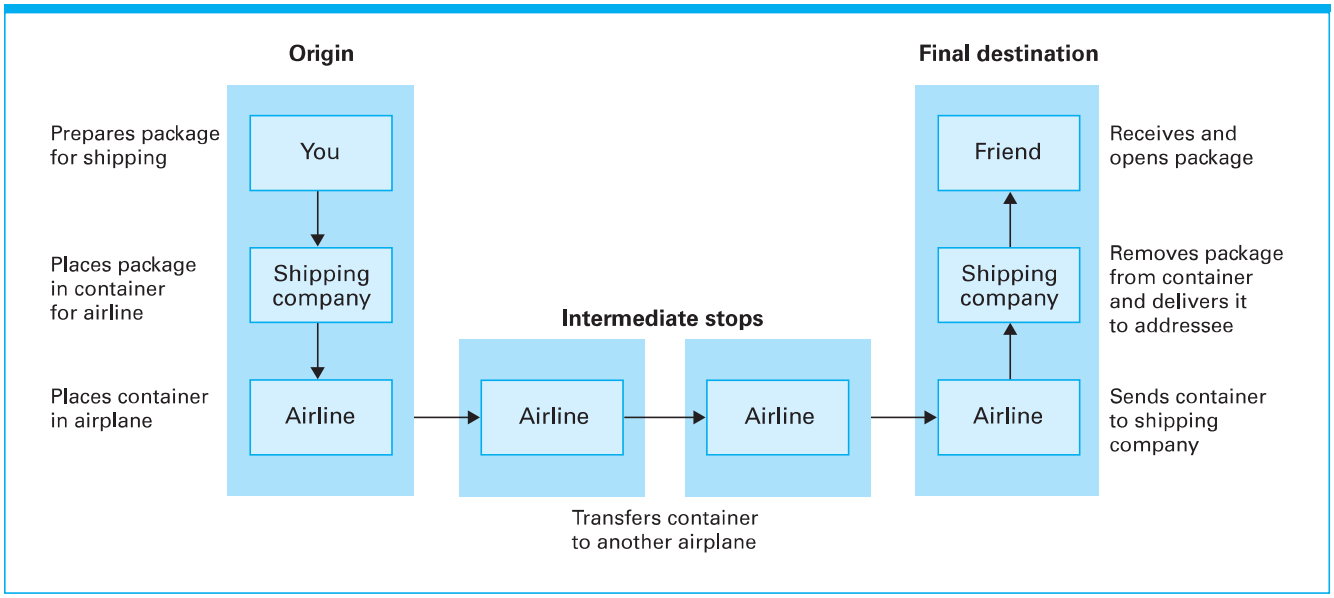
In this section we investigate how messages are transferred over the Internet. This transfer process requires the cooperation of all the computers in the system, and therefore software for controlling this process resides on every computer in the Internet. We begin by studying the overall structure of this software.

The Layered Approach to Internet Software

A principal task of networking software is to provide the infrastructure required for transferring messages from one machine to another. In the Internet, this message-passing activity is accomplished by means of a hierarchy of software units, which perform tasks analogous to those that would be performed if you were to send a gift in a package from the West Coast of the United States to a friend on the East Coast (Figure 4.12). You would first wrap the gift as a package and write the appropriate address on the outside of the package. Then, you would take the package to a shipping company such as the U.S. Postal Service. The shipping company might place the package along with others in a large container and deliver the container to an airline, whose services it has contracted. The airline would place the container in an aircraft and transfer it to the destination city, perhaps with intermediate stops along the way. At the final destination, the airline would remove the container from the aircraft and give it to the shipping company's office at the destination. In turn, the shipping company would take your package out of the container and deliver it to the addressee.

In short, the transportation of the gift would be carried out by a three-level hierarchy: (1) the user level (consisting of you and your friend), (2) the shipping company, and (3) the airline. Each level uses the next lower level as an abstract tool. (You are not concerned with the details of the shipping company, and the shipping company is not concerned with the internal operations of the airline.) Each level in the hierarchy has representatives at both the origin and the

Figure 4.12 Package-shipping example



destination, with the representatives at the destination tending to do the reverse of their counterparts at the origin.

Such is the case with software for controlling communication over the Internet, except that the Internet software has four layers rather than three, each consisting of a collection of software routines rather than people and businesses. The four layers are known as the **application layer,** the **transport layer,** the **network layer,** and the **link layer** (Figure 4.13). A message typically originates in the application layer. From there it is passed down through the transport and network layers as it is prepared for transmission, and finally it is transmitted by the link layer. The message is received by the link layer at the destination and passed back up the hierarchy until it is delivered to the application layer at the message's destination.

Let us investigate this process more thoroughly by tracing a message as it finds its way through the system (Figure 4.14). We begin our journey with the application layer.

The application layer consists of those software units such as clients and servers that use Internet communication to carry out their tasks. Although the names are similar, this layer is not restricted to software in the application classification presented in Section 3.2, but also includes many utility packages. For example, software for transferring files using FTP or for providing remote login capabilities using SSH have become so common that they are normally considered utility software.

The application layer uses the transport layer to send and receive messages over the Internet in much the same way that you would use a shipping company to send and receive packages. Just as it is your responsibility to provide an address compatible with the specifications of the shipping company, it is the application layer's responsibility to provide an address that is compatible with the Internet infrastructure. To fulfill this need, the application layer may use the services of the name servers within the Internet to translate mnemonic addresses used by humans into Internet-compatible IP addresses.

Figure 4.13 The Internet software layers

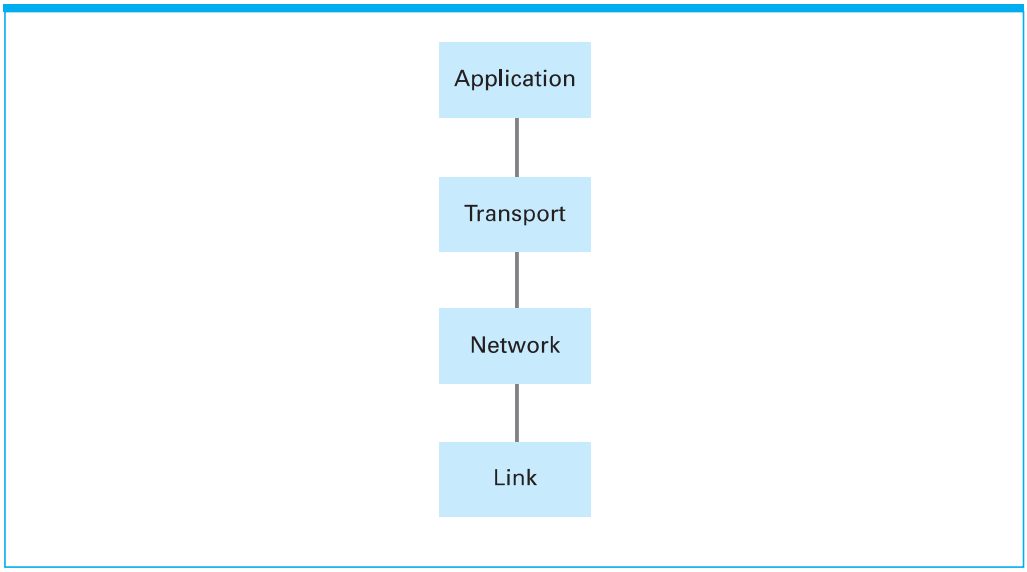
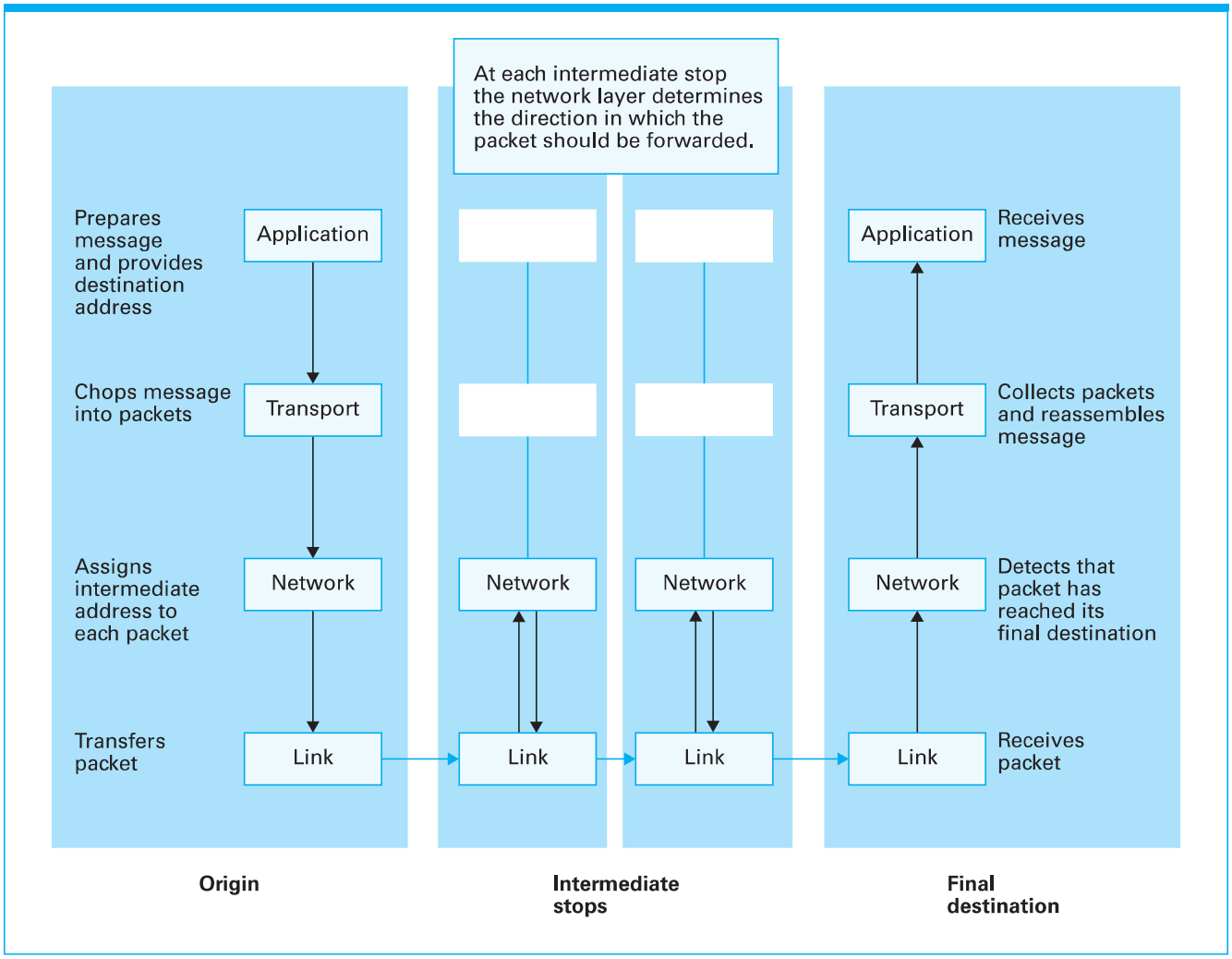


Figure 4.14 Following a message through the Internet



An important task of the transport layer is to accept messages from the application layer and to ensure that the messages are properly formatted for transmission over the Internet. Toward this latter goal, the transport layer divides long messages into small segments, which are transmitted over the Internet as individual units. This division is necessary because a single long message can obstruct the flow of other messages at the Internet routers where numerous messages cross paths. Indeed, small segments of messages can interweave at these points, whereas a long message forces others to wait while it passes (much like cars waiting for a long train to pass at a railroad crossing).

The transport layer adds sequence numbers to the small segments it produces so that the segments can be reassembled at the message's destination. Then it hands these segments, known as **packets,** to the network layer. From this point, the packets are treated as individual, unrelated messages until they reach the transport layer at their final destination. It is quite possible for the packets related to a common message to follow different paths through the Internet.

It is the network layer's job to decide in which direction a packet should be sent at each step along the packet's path through the Internet. In fact, the

combination of the network layer and the link layer below it constitutes the software residing on the Internet routers. The network layer is in charge of maintaining the router's forwarding table and using that table to determine the direction in which to forward packets. The link layer at the router is in charge of receiving and transmitting the packets.

Thus, when the network layer at a packet's origin receives the packet from the transport layer, it uses its forwarding table to determine where the packet should be sent to get it started on its journey. Having determined the proper direction, the network layer hands the packet to the link layer for actual transmission.

The link layer has the responsibility of transferring the packet. Thus the link layer must deal with the communication details particular to the individual network in which the computer resides. For instance, if that network is an Ethernet, the link layer applies CSMA/CD. If the network is a WiFi network, the link layer applies CSMA/CA.

When a packet is transmitted, it is received by the link layer at the other end of the connection. There, the link layer hands the packet up to its network layer where the packet's final destination is compared to the network layer's forwarding table to determine the direction of the packet's next step. With this decision made, the network layer returns the packet to the link layer to be forwarded along its way. In this manner each packet hops from machine to machine on its way to its final destination.

Note that only the link and network layers are involved at the intermediate stops during this journey (see again Figure 4.14), and thus these are the only layers present on routers, as previously noted. Moreover, to minimize the delay at each of these intermediate "stops," the forwarding role of the network layer within a router is closely integrated with the link layer. In turn, the time required for a modern router to forward a packet is measured in millionths of a second.

At a packet's final destination, it is the network layer that recognizes that the packet's journey is complete. In that case the network layer hands the packet to its transport layer rather than forwarding it. As the transport layer receives packets from the network layer, it extracts the underlying message segments and reconstructs the original message according to the sequence numbers that were provided by the transport layer at the message's origin. Once the message is assembled, the transport layer hands it to the appropriate unit within the application layer—thus completing the message transmission process.

Determining which unit within the application layer should receive an incoming message is an important task of the transport layer. This is handled by assigning unique **port numbers** (not related to the I/O ports discussed in Chapter 2) to the various units and requiring that the appropriate port number be appended to a message's address before starting the message on its journey. Then, once the message is received by the transport layer at the destination, the transport layer merely hands the message to the application layer software at the designated port number.

Users of the Internet rarely need to be concerned with port numbers because the common applications have universally accepted port numbers. For example, if a Web browser is asked to retrieve the document whose URL is `http://www.zoo.org/animals/frog.html`, the browser assumes that it should contact the HTTP server at `www.zoo.org` via port number 80. Likewise, when sending email, an SMTP client assumes that it should communicate with the SMTP mail server through port number 25.

In summary, communication over the Internet involves the interaction of four layers of software. The application layer deals with messages from the application's point of view. The transport layer converts these messages into segments that are compatible with the Internet and reassembles messages that are received before delivering them to the appropriate application. The network layer deals with directing the segments through the Internet. The link layer handles the actual transmission of segments from one machine to another. With all this activity, it is somewhat amazing that the response time of the Internet is measured in milliseconds, so that many transactions appear to take place instantaneously.

The TCP/IP Protocol Suite

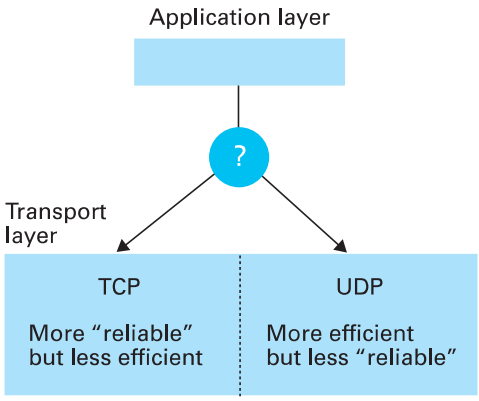
The demand for open networks has generated a need for published standards by which manufacturers can supply equipment and software that function properly with products from other vendors. One standard that has resulted is the **Open System Interconnection (OSI)** reference model, produced by the International Organization for Standardization. This standard is based on a seven-level hierarchy as opposed to the four-level hierarchy we have just described. It is an often-quoted model because it carries the authority of an international organization, but it has been slow to replace the four-level point of view, mainly because it was established after the four-level hierarchy had already become the de facto standard for the Internet.

The TCP/IP protocol suite is a collection of protocol standards used by the Internet to implement the four-level communication hierarchy. Actually, the **Transmission Control Protocol (TCP)** and the **Internet Protocol (IP)** are the names of only two of the protocols in this vast collection—so the fact that the entire collection is referred to as the TCP/IP protocol suite is rather misleading. More precisely, TCP defines a version of the transport layer. We say a *version* because the TCP/IP protocol suite provides for more than one way of implementing the transport layer; one of the other options is defined by the **User Datagram Protocol (UDP).** This diversity is analogous to the fact that when shipping a package, you have a choice of different shipping companies, each of which offers the same basic service but with its own unique characteristics. Thus, depending on the particular quality of service required, a unit within the application layer might choose to send data via a TCP or UDP version of the transport layer (Figure 4.15).

There are several differences between TCP and UDP. One is that before sending a message as requested by the application layer, a transport layer based on TCP sends its own message to the transport layer at the destination telling it that a message is about to be sent. It then waits for this message to be acknowledged before starting to send the application layer's message. In this manner, a TCP transport layer is said to establish a connection before sending a message. A transport layer based on UDP does not establish such a connection prior to sending a message. It merely sends the message to the address it was given and forgets about it. For all it knows, the destination computer might not even be operational. For this reason, UDP is called a connectionless protocol.

Another difference between TCP and UDP is that TCP transport layers at the origin and destination work together by means of acknowledgments and packet retransmissions to assure that all segments of a message are successfully transferred to the destination. For this reason TCP is called a reliable protocol, whereas UDP, which does not offer such retransmission services, is said to be an unreliable protocol.

Figure 4.15 Choosing between TCP and UDP



Still another distinction between TCP and UDP is that TCP provides for both **flow control,** meaning that a TCP transport layer at a message's origin can reduce the rate at which it transmits segments to keep from overwhelming its counterpart at the destination, as well as **congestion control,** meaning that a TCP transport layer at a message's origin can adjust its transmission rate to alleviate congestion between it and the message's destination.

All this does not mean that UDP is a poor choice. After all, a transport layer based on UDP is more streamlined than a layer based on TCP, and thus if an application is prepared to handle the potential consequences of UDP, that option might be the better choice. For example, the efficiency of UDP makes it the protocol of choice for DNS lookups and VoIP. However, because email is less time sensitive, mail servers use TCP to transfer email.

IP is the Internet's standard for implementing the tasks assigned to the network layer. We have already observed that this task consists of **forwarding,** which involves relaying packets through the Internet, and **routing,** which involves updating the layer's forwarding table to reflect changing conditions. For instance, a router may malfunction, meaning that traffic should no longer be forwarded in its direction, or a section of the Internet may become congested, meaning that traffic should be routed around the blockage. Much of the IP standard associated with routing deals with the protocols used for communication among neighboring network layers as they interchange routing information.

An interesting feature associated with forwarding is that each time an IP network layer at a message's origin prepares a packet, it appends a value called a **hop count,** or time to live, to that packet. This value is a limit to the number of times the packet should be forwarded as it tries to find its way through the Internet. Each time an IP network layer forwards a packet, it decrements that packet's hop count by one. With this information, the network layer can protect the Internet from packets circling endlessly within the system. Although the Internet continues to grow on a daily basis, an initial hop count of 64 remains more than sufficient to allow a packet to find its way through the maze of routers within today's ISPs.

For years a version of IP known as IPv4 (IP version four) has been used for implementing the network layer within the Internet. However, the Internet is

rapidly outgrowing the 32-bit internet addressing system dictated by IPv4. To solve this problem as well as to implement other improvements such as multicast, a new version of IP known as IPv6, which uses internet addresses consisting of 128 bits, has been established. The process of converting from IPv4 to IPv6 is currently underway—this is the conversion that was alluded to in our introduction of Internet addresses in Section 4.2—and it is expected that the use of 32-bit addresses within the Internet will be extinct by 2025.

Questions & Exercises

1. What layers of the Internet software hierarchy are not needed at a router?
2. What are some differences between a transport layer based on the TCP protocol and another based on the UDP protocol?
3. How does the transport layer determine which unit with the application layer should receive an incoming message?
4. What keeps a computer on the Internet from recording copies of all the messages passing through it?

4.5 Security

When a computer is connected to a network, it becomes subject to unauthorized access and vandalism. In this section we address topics associated with these problems.

Forms of Attack

There are numerous ways that a computer system and its contents can be attacked via network connections. Many of these incorporate the use of malicious software (collectively called **malware**). Such software might be transferred to, and executed on, the computer itself, or it might attack the computer from a distance. Examples of software that is transferred to, and executed on, the computer under

The Computer Emergency Response Team

In November 1988 a worm released into the Internet caused significant disruption of service. Consequently, the U.S. Defense Advanced Research Projects Agency (DARPA—pronounced "DAR–pa") formed the Computer Emergency Response Team (CERT—pronounced "SERT"), located at the CERT Coordination Center at Carnegie-Mellon University. The CERT is the Internet's security "watchdog." Among its duties are the investigation of security problems, the issuance of security alerts, and the implementation of public awareness campaigns to improve Internet security. The CERT Coordination Center maintains a website at http://www.cert.org where it posts notices of its activities.

attack include viruses, worms, Trojan horses, and spyware, whose names reflect the primary characteristic of the software.

A **virus** is software that infects a computer by inserting itself into programs that already reside in the machine. Then, when the "host" program is executed, the virus is also executed. When executed, many viruses do little more than try to transfer themselves to other programs within the computer. Some viruses, however, perform devastating actions such as degrading portions of the operating system, erasing large blocks of mass storage, or otherwise corrupting data and other programs.

A **worm** is an autonomous program that transfers itself through a network, taking up residence in computers and forwarding copies of itself to other computers. As in the case of a virus, a worm can be designed merely to replicate itself or to perform more extreme vandalism. A characteristic consequence of a worm is an explosion of the worm's replicated copies that degrades the performance of legitimate applications and can ultimately overload an entire network or internet.

A **Trojan horse** is a program that enters a computer system disguised as a desirable program, such as a game or useful utility package, that is willingly imported by the victim. Once in the computer, however, the Trojan horse performs additional activities that might have harmful effects. Sometimes these additional activities start immediately. In other instances, the Trojan horse might lie dormant until triggered by a specific event such as the occurrence of a preselected date. Trojan horses often arrive in the form of attachments to enticing email messages. When the attachment is opened (that is, when the recipient asks to view the attachment), the misdeeds of the Trojan horse are activated. Thus, email attachments from unknown sources should never be opened.

Another form of malicious software is **spyware** (sometimes called **sniffing** software), which is software that collects information about activities at the computer on which it resides and reports that information back to the instigator of the attack. Some companies use spyware as a means of building customer profiles, and in this context, it has questionable ethical merit. In other cases, spyware is used for blatantly malicious purposes such as recording the symbol sequences typed at the computer's keyboard in search of passwords or credit card numbers.

As opposed to obtaining information secretly by sniffing via spyware, **phishing** is a technique of obtaining information explicitly by simply asking for it. The term *phishing* is a play on the word *fishing* because the process involved is to cast numerous "lines" in hopes that someone will "take the bait." Phishing is often carried out via email, and in this form, it is little more than an old telephone con. The perpetrator sends email messages posing as a financial institution, a government bureau, or perhaps a law enforcement agency. The email asks the potential victim for information that is supposedly needed for legitimate purposes. However, the information obtained is used by the perpetrator for hostile purposes.

In contrast to suffering from such internal infections as viruses and spyware, a computer in a network can also be attacked by software being executed on other computers in the system. An example is a **denial of service (DoS)** attack, which is the process of overloading a computer with messages. Denial of service attacks have been launched against large commercial webservers on the Internet to disrupt the company's business and in some cases have brought the company's commercial activity to a halt.

A denial of service attack requires the generation of a large number of messages over a brief period of time. To accomplish this, an attacker usually plants

software on numerous unsuspecting computers that will generate messages when a signal is given. Then, when the signal is given, all of these computers (sometimes called a **botnet**) swamp the target with messages. Inherent, then, in denial of service attacks is the availability of unsuspecting computers to use as accomplices. This is why all PC users are discouraged from leaving their computers connected to the Internet when not in use. It has been estimated that once a PC is connected to the Internet, at least one intruder will attempt to exploit its existence within 20 minutes. In turn, an unprotected PC represents a significant threat to the integrity of the Internet.

Another problem associated with an abundance of unwanted messages is the proliferation of unwanted junk email, called **spam.** However, unlike a denial of service attack, the volume of spam is rarely sufficient to overwhelm the computer system. Instead, the effect of spam is to overwhelm the person receiving the spam. This problem is compounded by the fact that, as we have already seen, spam is a widely adopted medium for phishing and instigating Trojan horses that might spread viruses and other detrimental software.

Protection and Cures

The old adage "an ounce of prevention is worth a pound of cure" is certainly true in the context of controlling vandalism over network connections. A primary prevention technique is to filter traffic passing through a point in the network, usually with a program called a **firewall.** For instance, a firewall might be installed at the gateway of an organization's intranet to filter messages passing in and out of the region. Such firewalls might be designed to block outgoing messages with certain destination addresses or to block incoming messages from origins that are known to be sources of trouble. This latter function is a tool for terminating a denial of service attack because it provides a means of blocking traffic from the attacking computers. Another common role of a firewall at a gateway is to block all incoming messages that have origin addresses within the region accessed through the gateway because such a message would indicate that an outsider is pretending to be a member of the inside region. Masquerading as a party other than one's self is known as **spoofing.**

Firewalls are also used to protect individual computers rather than entire networks or domains. For example, if a computer is not being used as a webserver, a name server, or an email server, then a firewall should be installed at that computer to block all incoming traffic addressed to such applications. Indeed, one way an intruder might gain entry to a computer is by establishing contact through a "hole" left by a nonexistent server. In particular, one method for retrieving information gathered by spyware is to establish a clandestine server on the infected computer by which malicious clients can retrieve the spyware's findings. A properly installed firewall could block the messages from these malicious clients.

Some variations of firewalls are designed for specific purposes—an example being **spam filters,** which are firewalls designed to block unwanted email. Many spam filters use rather sophisticated techniques to distinguish between desirable email and spam. Some learn to make this distinction via a training process in which the user identifies items of spam until the filter acquires enough examples to make decisions on its own. These filters are examples of how a variety of subject areas (probability theory, artificial intelligence, etc.) can jointly contribute to developments in other fields.

Another preventative tool that has filtering connotations is the **proxy server.** A proxy server is a software unit that acts as an intermediary between a client and a server with the goal of shielding the client from adverse actions of the server. Without a proxy server, a client communicates directly with a server, meaning that the server has an opportunity to learn a certain amount about the client. Over time, as many clients within an organization's intranet deal with a distant server, that server can collect a multitude of information about the intranet's internal structure—information that can later be used for malicious activity. To counter this, an organization can establish a proxy server for a particular kind of service (FTP, HTTP, telnet, etc.). Then, each time a client within the intranet tries to contact a server of that type, the client is actually placed in contact with the proxy server. In turn, the proxy server, playing the role of a client, contacts the actual server. From then on the proxy server plays the role of an intermediary between the actual client and the actual server by relaying messages back and forth. The first advantage of this arrangement is that the actual server has no way of knowing that the proxy server is not the true client, and in fact, it is never aware of the actual client's existence. In turn, the actual server has no way of learning about the intranet's internal features. The second advantage is that the proxy server is in position to filter all the messages sent from the server to the client. For example, an FTP proxy server could check all incoming files for the presence of known viruses and block all infected files.

Still another tool for preventing problems in a network environment is auditing software that is similar to the auditing software we learned about in our discussion on operating system security (Section 3.5). Using network auditing software, a system administrator can detect a sudden increase in message traffic at various locations within the administrator's realm, monitor the activities of the system's firewalls, and analyze the pattern of requests being made by the individual computers in order to detect irregularities. In effect, auditing software is an administrator's primary tool for identifying problems before they grow out of control.

Another means of defense against invasions via network connections is software called **antivirus software,** which is used to detect and remove the presence of known viruses and other infections. (Actually, antivirus software represents a broad class of software products, each designed to detect and remove a specific type of infection. For example, whereas many products specialize in virus control, others specialize in spyware protection.) It is important for users of these packages to understand that, just as in the case of biological systems, new computer infections are constantly coming on the scene that require updated vaccines. Thus, antivirus software must be routinely maintained by downloading updates from the software's vendor. Even this, however, does not guarantee the safety of a computer. After all, a new virus must first infect some computers before it is discovered and a vaccine is produced. Thus, a wise computer user never opens email attachments from unfamiliar sources, does not download software without first confirming its reliability, does not respond to pop-up adds, and does not leave a PC connected to the Internet when such connection is not necessary.

Encryption

In some cases the purpose of network vandalism is to disrupt the system (as in denial of service attacks), but in other cases the ultimate goal is to gain access to information. The traditional means of protecting information is to control its

access through the use of passwords. However, passwords can be compromised and are of little value when data are transferred over networks and internets where messages are relayed by unknown entities. In these cases encryption can be used so that even if the data fall into unscrupulous hands, the encoded information will remain confidential. Today, many traditional Internet applications have been altered to incorporate encryption techniques, producing what are called "secure versions" of the applications.

A prime example is the secure version of HTTP, known as **HTTPS,** which is used by most financial institutions to provide customers with secure Internet access to their accounts. The backbone of HTTPS is the protocol system known as **Secure Sockets Layer (SSL),** which was originally developed by Netscape to provide secure communication links between Web clients and servers. Most browsers indicate the use of SSL by displaying a tiny padlock icon on the computer screen. (Some use the presence or absence of the icon to indicate whether SSL is being used; others display the padlock in either the locked or unlocked position.)

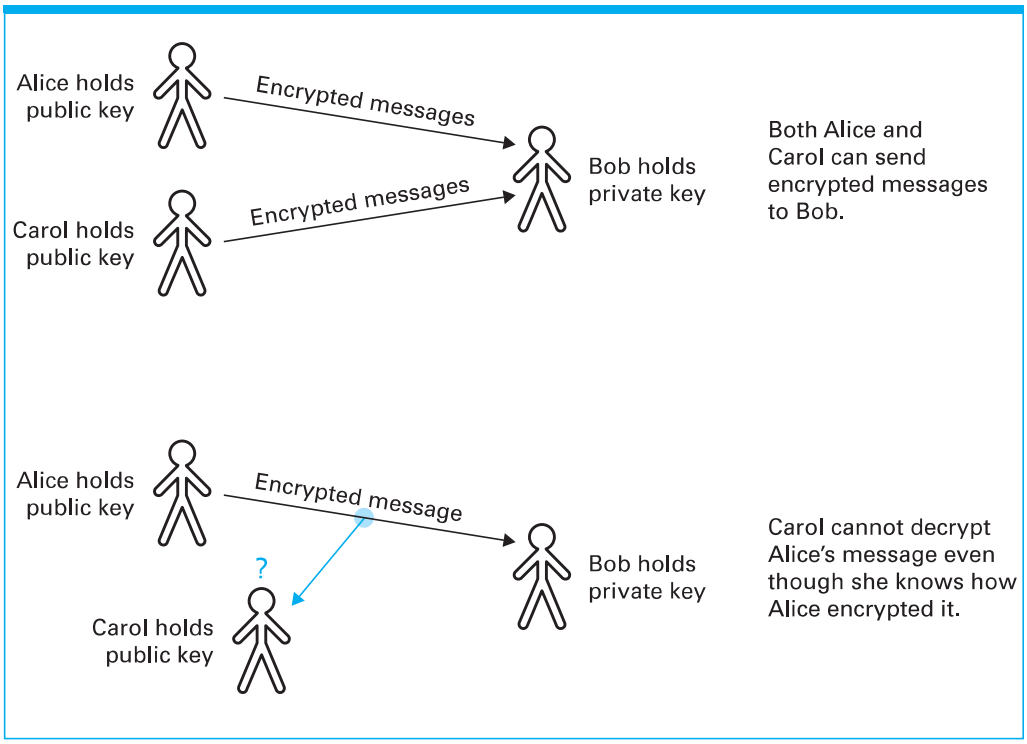
One of the more fascinating topics in the field of encryption is **public-key encryption,** which involves techniques by which encryption systems are designed so that having knowledge about how messages are encrypted does not allow one to decrypt messages. This characteristic is somewhat counterintuitive. After all, intuition would suggest that if a person knows how messages are encrypted, then that person should be able to reverse the encryption process and thus decrypt messages. But public-key encryption systems defy this intuitive logic.

A public-key encryption system involves the use of two values called keys. One key, known as the **public key,** is used to encrypt messages; the other key, known as the **private key,** is required to decrypt messages. To use the system, the public key is first distributed to those who might need to send messages to a particular destination. The private key is held in confidence at this destination. Then, the originator of a message can encrypt the message using the public key and send the message to its destination with assurance that its contents are safe, even if it is handled by intermediaries who also know the public key. Indeed, the only party that can decrypt the message is the party at the message's destination who holds the private key. Thus if Bob creates a public-key encryption system and gives both Alice and Carol the public key, then both Alice and Carol can encrypt messages to Bob, but they cannot spy on the other's communication. Indeed, if Carol intercepts a message from Alice, she cannot decrypt it even though she knows how Alice encrypted it (Figure 4.16).

There are, of course, subtle problems lurking within public-key systems. One is to ensure that the public key being used is, in fact, the proper key for the destination party. For example, if you are communicating with your bank, you want to be sure that the public key you are using for encryption is the one for the bank and not an impostor. If an impostor presents itself as the bank (an example of spoofing) and gives you its public key, the messages you encrypt and send to the "bank" would be meaningful to the impostor and not your bank. Thus, the task of associating public keys with correct parties is significant.

One approach to resolving this problem is to establish trusted Internet sites, called **certificate authorities,** whose task is to maintain accurate lists of parties and their public keys. These authorities, acting as servers, then provide reliable public-key information to their clients in packages known as **certificates.** A certificate is a package containing a party's name and that party's public key.

Figure 4.16 Public key encryption



Many commercial certificate authorities are now available on the Internet, although it is also common for organizations to maintain their own certificate authorities in order to maintain tighter control over the security of the organization's communication.

Finally, we should comment on the role public-key encryption systems play in solving problems of **authentication**—making sure that the author of a message is, in fact, the party it claims to be. The critical point here is that, in some public-key encryption systems, the roles of the encryption and decryption keys can be reversed. That is, text can be encrypted with the private key, and because only one party has access to that key, any text that is so encrypted must have originated from that party. In this manner, the holder of the private key can produce a bit pattern, called a **digital signature,** that only that party knows how to produce. By attaching that signature to a message, the sender can mark the message as being authentic. A digital signature can be as simple as the encrypted version of the message itself. All the sender must do is encrypt the message being transmitted using his or her private key (the key typically used for decrypting). When the message is received, the receiver uses the sender's public key to decrypt the signature. The message that is revealed is guaranteed to be authentic because only the holder of the private key could have produced the encrypted version.

We will return to the topic of public-key encryption at the end of Chapter 12, as an example of a complex computer algorithm.

Legal Approaches to Network Security

Another way of enhancing the security of computer networking systems is to apply legal remedies. There are, however, two obstacles to this approach. The first is that making an action illegal does not preclude the action. All it does is

provide a legal recourse. The second is that the international nature of networking means that obtaining recourse is often very difficult. What is illegal in one country might be legal in another. Ultimately, enhancing network security by legal means is an international project, and thus must be handled by international legal bodies—a potential player would be the International Court of Justice in The Hague.

Having made these disclaimers, we must admit that, although less than perfect, legal forces still have a tremendous influence, and thus it behooves us to explore some of the legal steps that are being taken to resolve conflicts in the networking arena. For this purpose, we use examples from the federal laws of the United States. Similar examples could be drawn from other government bodies such as the European Union.

We begin with the proliferation of malware. In the United States this problem is addressed by the Computer Fraud and Abuse Act, which was first passed in 1984, although it has been amended several times. It is under this act that most cases involving the introduction of worms and viruses have been prosecuted. In short, the act requires proof that the defendant knowingly caused the transmission of a program or data that intentionally caused damage. The Computer Fraud and Abuse Act also covers cases involving the theft of information. In particular, the act outlaws obtaining anything of value via the unauthorized access of a computer. Courts have tended to assign a broad interpretation to the phrase "anything of value," and thus the Computer Fraud and Abuse Act has been applied to more than the theft of information. For instance, courts have ruled that the mere use of a computer might constitute "anything of value."

The right of privacy is another, and perhaps the most controversial, networking issue facing the legal community. Questions involving an employer's right to monitor the communications of employees and the extent to which an Internet service provider is authorized to access the information being communicated by its clients have been given considerable thought. In the United States, many of these questions are addressed by the Electronic Communication Privacy Act (ECPA) of 1986, which has its origins in legislation to control wiretapping. Although the act is lengthy, its intent is captured in a few short excerpts. In particular, it states that

> Except as otherwise specifically provided in this chapter any person who intentionally intercepts, endeavors to intercept, or procures any other person to intercept or endeavor to intercept, any wire, oral, or electronic communication . . . shall be punished as provided in subsection (4) or shall be subject to suit as provided in subsection (5).

and

> . . . any person or entity providing an electronic communication service to the public shall not intentionally divulge the contents of any communication . . . on that service to any person or entity other than an addressee or intended recipient of such communication or an agent of such addressee or intended recipient.

In brief, the ECPA confirms an individual's right to private communication—it is illegal for an Internet service provider to release information about the communication of its clients, and it is illegal for unauthorized personnel to eavesdrop on another's communication. But the ECPA leaves room for debate. For example, the question regarding the rights of an employer to monitor the communication of employees becomes a question of authorization, which courts have tended to grant to employers when the communication is carried out using the employer's equipment.

Moreover, the act goes on to give some government agencies authority to monitor electronic communications under certain restrictions. These provisions have been the source of much debate. For example, in 2000 the FBI revealed the existence of its system, called Carnivore, that reports on the communication of all subscribers of an Internet service provider rather than just a court-designated target, and in 2001 in response to the terrorist attack on the World Trade Center, congress passed the controversial USA PATRIOT (Uniting and Strengthening America by Providing Appropriate Tools Required to Intercept and Obstruct Terrorism) Act that modified the restrictions under which government agencies must operate. In 2013, it was subsequently revealed that these laws had been interpreted to authorize the collection of vast amounts of telephone and Internet usage data on ordinary Americans indiscriminately.

In addition to the legal and ethical controversies raised by these developments, providing monitoring rights raises some technical problems that are more pertinent to our study. One is that to provide these capabilities, a communication system must be constructed and programmed so that communications can be monitored. To establish such capabilities was the goal of the Communications Assistance for Law Enforcement Act (CALEA). It requires telecommunication carriers to modify their equipment to accommodate law enforcement taps—a requirement that has been complex and expensive to meet.

Another controversial issue involves the clash between the government's right to monitor communications and the public's right to use encryption. If the messages being monitored are well encrypted, then tapping the communication is of limited value to law enforcement agencies. Governments in the United States, Canada, and Europe are considering systems that would require the registration of ciphering keys, but such demands are being fought by corporations. After all, due to corporate espionage it is understandable that requiring the registration of ciphering keys would make many law-abiding corporations, as well as citizens, uncomfortable. How secure can the registration system be?

Finally, as a means of recognizing the scope of legal issues surrounding the Internet, we cite the Anticybersquatting Consumer Protection Act of 1999 that is designed to protect organizations from impostors who might otherwise establish look-alike domain names (a practice known as cybersquatting). The act prohibits the use of domain names that are identical or confusingly similar to another's trademark or "common law trademark." One effect is that although the act does not outlaw domain name speculation (the process of registering potentially desirable domain names and later selling the rights to that name), it limits the practice to generic domain names. Thus, a domain name speculator might legally register a generic name such as `GreatUsedCars.com` but might not be able to claim rights to the name `BigAlUsedCars.com` if Big Al is already in the used car business. Such distinctions are often the subject of debate in lawsuits based on the Anticybersquatting Consumer Protection Act.

Questions & Exercises

1. What is *phishing*? How are computers secured against it?
2. What distinction is there between the types of firewalls that can be placed at a domain's gateway as opposed to an individual host within the domain?

3. Technically, the term *data* refers to representations of information, whereas *information* refers to the underlying meaning. Does the use of passwords protect data or information? Does the use of encryption protect data or information?
4. What advantage does public-key encryption have over more traditional encryption techniques?
5. What problems are associated with legal attempts to protect against network security problems?

Chapter Review Problems

(Asterisked problems are associated with optional sections.)

1. What is a protocol? Identify three protocols introduced in this chapter and describe the purpose of each.
2. What is interprocess communication?
3. What is meant by an intranet?
4. Describe the three types of ISP tiers.
5. What is the difference between a bus network and a star network?
6. What happens when two nodes using the CSMA/CD, encounter collision in the network?
7. Can collisions be completely eliminated by using the CSMA/CA like collision avoidance protocols?
8. What is the last mile problem? Describe a technique for solving it.
9. How does a hub differ from a switch?
10. How does a router differ from such devices as repeaters, bridges, and switches?
11. State the distinction between the client-server model and the P2P model.
12. What is the difference between congestion control and flow control?
13. Using 32-bit Internet addresses was originally thought to provide ample room for expansion, but that conjecture is not proving to be accurate. IPv6 uses 128-bit addressing. Will that prove to be adequate? Justify your answer. (For example, you might compare the number of possible addresses to the population of the world.)
14. Encode each of the following bit patterns using dotted decimal notation.
    a. 000001010001001000100011
    b. 1000000000100000
    c. 0011000000011000
15. What bit pattern is represented by each of the following dotted decimal patterns?
    a. 0.0
    b. 26.19.1
    c. 8.12.20.13
16. Suppose the address of an end system on the Internet is quoted as 134.48.4.122. What is the 32-bit address in hexadecimal notation?
17. What is on-demand streaming?
18. If a computer's mnemonic Internet address is `batman.batcave.metropolis.gov` what might you conjecture about the structure of the domain containing the machine?
19. Explain the components of the email address `kermit@animals.com`
20. In the context of VoIP, what is the difference between an analog telephone adapter and an embedded phone?
21. What is the role of a name server?
22. What is the distinction between routing and forwarding?
23. Define each of the following:
    a. Cloud computing
    b. Hot spot
    c. Top-level domains
    d. Secure Shell

24. Define each of the following:
a. Content delivery networks
b. Anycast
c. Markup languages

25. Many "lay users" of the Internet interchange the terms *Internet* and *World Wide Web*. To what do each of the terms correctly refer?

26. When viewing a simple webpage, ask your browser to display the source version of the document. Then identify the basic structure of the document. In particular, identify the head and the body of the document and list some of the statements you find in each.

27. State the difference between sniffing and phishing.

28. Modify the HTML document below so that the word "Rover" is linked to the document whose URL is `http://animals.org/pets/dogs.html`.

```
<html>
<head>
<title>Example</title>
</head>
<body>
<h1> My Pet Dog</h1>
<p>My dog's name is Rover.</p>
</body>
</html>
```

29. Draw a sketch showing how the following HTML document would appear when displayed on a computer screen.

```
<html>
<head>
<title>Example</title>
</head>
<body>
<h1> My Pet Dog</h1>
<img src = "Rover.jpg">
</body>
</html>
```

30. Using the informal XML style presented in the text, design a markup language for representing simple algebraic expressions as text files.

31. Using the informal XML style presented in the text, design a set of tags that a word processor might use for marking the underlying text. For example, how would a word processor indicate what text should be bold, italic, underlined, and so on?

32. Using the informal XML style presented in the text, design a set of tags that could be used to mark motion picture reviews according to the way the text items should appear on a printed page. Then design a set of tags that could be used to mark the reviews according to the meaning of the items in the text.

33. Using the informal XML style presented in the text, design a set of tags that could be used to mark articles about sporting events according to the way the text items should appear on a printed page. Then design a set of tags that could be used to mark the articles according to the meaning of the items in the text.

34. Identify the components of the following URL and describe the meaning of each.
`http://lifeforms.com/animals/moviestars/kermit.html`

35. Identify the components of each of the following abbreviated URLs.
a. `http://www.farmtools.org/windmills.html`
b. `http://castles.org/`
c. `www.coolstuff.com`

36. How would the action of a browser differ if you asked it to "find the document" at the URL `http://stargazer.universe.org` as opposed to `https://stargazer.universe.org`?

37. Give two examples of client-side activities on the Web. Give two examples of server-side activities on the Web.

***38.** What is the TCP/IP protocol suite?

***39.** In a network based on the bus topology, the bus is a nonsharable resource for which the machines must compete in order to transmit messages. How is deadlock (see the optional Section 3.4) controlled in this context?

***40.** List the four layers in the Internet software hierarchy and identify a task performed by each layer.

***41.** Why does the transport layer chop large messages into small packets?

*42. When an application asks the transport layer to use TCP to transmit a message, what additional messages will be sent by the transport layer in order to fulfill the application layer's request?

*43. In what way could TCP be considered a better protocol for implementing the transport layer than UDP? In what way could UDP be considered better than TCP?

*44. What does it mean to say that the TCP is a connection-oriented protocol?

*45. At what layer in the TCP/IP protocol hierarchy could a firewall be placed to filter incoming traffic by means of
a. Message content
b. Source address
c. Type of application

46. Suppose you wanted to establish a firewall to filter out email messages containing certain terms and phrases. Would this firewall be placed at your domain's gateway or at the domain's mail server? Explain your answer.

47. What is SSL and what are its benefits?

48. What is the significance of using spam filters?

49. What is the significance of certificate authorities in public-key encryption?

50. In what sense does the global nature of the Internet limit legal solutions to Internet problems?

Social Issues

The following questions are intended as a guide to the ethical/social/legal issues associated with the field of computing. The goal is not merely to answer these questions. You should also consider why you answered as you did and whether your justifications are consistent from one question to the next.

1. The ability to connect computers via networks has popularized the concept of working at home. What are some pros and cons of this movement? Will it affect the consumption of natural resources? Will it strengthen families? Will it reduce "office politics"? Will those who work at home have the same career advancement opportunities as those who work on site? Will community ties be weakened? Will reduced personal contact with peers have a positive or negative effect?
2. Ordering merchandise over the Internet is becoming an alternative to "hands-on" shopping. What effect will such a shift in shopping habits have on communities? What about shopping malls? What about small shops, such as bookstores and clothing stores, in which you like to browse without buying? To what extent is buying at the lowest possible price good or bad? Is there any moral obligation to pay more for an item in order to support a local business? Is it ethical to compare products at a local store and then order your selection at a lower price via the Internet? What are the long-term consequences of such behavior?
3. To what extent should a government control its citizens' access to the Internet (or any international network)? What about issues that involve national security? What are some security issues that might occur?
4. Electronic bulletin boards allow users of networks to post messages (often anonymously) and read messages posted by others. Should the manager of such a bulletin board be held responsible for its contents? Should a telephone company be held responsible for the contents of telephone conversations?

Should the manager of a grocery store be held responsible for the contents of a community bulletin board located in the store?

5. Should the use of the Internet be monitored? Should it be regulated? If so, by whom and to what extent?
6. How much time do you spend using the Internet? Is that time well spent? Has Internet access altered your social activities? Do you find it easier to talk to people via the Internet than in person?
7. When you buy a software package for a personal computer, the developer usually asks you to register with the developer so that you can be notified of future upgrades. This registration process is increasingly being handled via the Internet. You are usually asked to give such things as your name, address, and perhaps how you learned of the product, and then the developer's software automatically transfers this data to the developer. What ethical issues would be raised if the developer designed the registration software so that it sent additional information to the developer during the registration process? For example, the software might scan the contents of your system and report the other software packages found.
8. When you visit a website, that site has the capability of recording data, called cookies, on your computer indicating that you have visited that site. These cookies can then be used to identify return visitors and to record their previous activities so that future visits to the site can be handled more efficiently. The cookies on your computer also provide a record of the sites you have visited. Should a website have the capability to record cookies on your computer? Should a website be allowed to record cookies on your computer without your knowledge? What are possible benefits of cookies? What problems could arise from the use of cookies?
9. If corporations are required to register their encryption keys with a government agency, will they be safe?
10. In general, etiquette tells us to avoid calling a friend at his or her place of work for personal or social matters such as making arrangements for a weekend outing. Likewise, most of us would hesitate to call a customer at his or her home to describe a new product. In a similar manner, we mail wedding invitations to the guests' residences, whereas we mail announcements of business conferences to the attendees' work addresses. Is it proper to send personal email to a friend via the mail server at the friend's place of employment?
11. Suppose a PC owner leaves the PC connected to the Internet where it ultimately is used by another party to implement a denial of service attack. To what extent should the PC owner be liable? Does your answer depend on whether the owner installed proper firewalls?
12. Is it ethical for companies that produce candy or toys to provide games on their company websites that entertain children while promoting the company's products? What if the game is designed to collect information from the children? What are the boundaries between entertaining, advertising, and exploitation?

Additional Reading

Antoniou, G., P. Groth, F. van Harmelem and R. Hoekstra. *A Semantic Web Primer*, 3rd ed. Cambridge, MA: MIT Press, 2012.

Bishop, M. *Introduction to Computer Security*. Boston, MA: Addison-Wesley, 2005.

Comer, D. E. *Computer Networks and Internets*, 6th ed. Upper Saddle River, NJ: Prentice-Hall, 2014.

Comer, D. E. *Internetworking with TCP/IP, Vol. 1*, 6th ed. Upper Saddle River, NJ: Prentice-Hall, 2013.

Goldfarb, C. F., and P. Prescod. *The XML Handbook*, 5th ed. Upper Saddle River, NJ: Prentice-Hall, 2004.

Halsal, F. *Computer Networking and the Internet*, 5th ed. Boston, MA: Addison-Wesley, 2005.

Harrington, J. L. *Network Security: A Practical Approach*. San Francisco, CA: Morgan Kaufmann, 2005.

Kurose, J. F., and K. W. Ross. *Computer Networking: A Top Down Approach Featuring the Internet*, 6th ed. Boston, MA: Addison-Wesley, 2012.

Peterson, L. L., and B. S. Davie. *Computer Networks: A Systems Approach*, 5th ed. San Francisco, CA: Morgan Kaufmann, 2011.

Rosenoer, J. *CyberLaw: The Law of the Internet*. New York: Springer, 1997.

Spinello, R. A., and H. T. Tavani. *Readings in CyberEthics*, 2nd ed. Sudbury, MA: Jones and Bartlett, 2004.

Stallings, W. *Cryptography and Network Security*, 5th ed. Upper Saddle River, NJ: Prentice-Hall, 2010.

Stevens, W. R. *TCP/IP Illustrated, Vol. 1*. Boston, MA: Addison-Wesley, 1994.