

1.1 DATA COMMUNICATIONS

When we communicate, we are sharing information. This sharing can be local or remote. Between individuals, local communication usually occurs face to face, while remote communication takes place over distance. The term **telecommunication**, which includes telephony, telegraphy, and television, means communication at a distance (*tele* is Greek for "far"). The word **data** refers to information presented in whatever form is agreed upon by the parties creating and using the data.

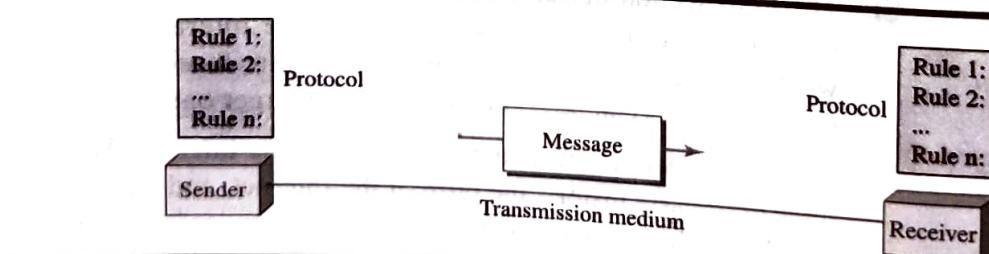
Data communications are the exchange of data between two devices via some form of transmission medium such as a wire cable. For data communications to occur, the communicating devices must be part of a communication system made up of a combination of hardware (physical equipment) and software (programs). The effectiveness of a data communications system depends on four fundamental characteristics: delivery, accuracy, timeliness, and jitter.

1. **Delivery.** The system must deliver data to the correct destination. Data must be received by the intended device or user and only by that device or user.
2. **Accuracy.** The system must deliver the data accurately. Data that have been altered in transmission and left uncorrected are unusable.
3. **Timeliness.** The system must deliver data in a timely manner. Data delivered late are useless. In the case of video and audio, timely delivery means delivering data as they are produced, in the same order that they are produced, and without significant delay. This kind of delivery is called *real-time* transmission.
4. **Jitter.** Jitter refers to the variation in the packet arrival time. It is the uneven delay in the delivery of audio or video packets. For example, let us assume that video packets are sent every 30 ms. If some of the packets arrive with 30-ms delay and others with 40-ms delay, an uneven quality in the video is the result.

1.1.1 Components

A data communications system has five components (see Figure 1.1).

Figure 1.1 Five components of data communication



1. **Message.** The **message** is the information (data) to be communicated. Popular forms of information include text, numbers, pictures, audio, and video.
2. **Sender.** The **sender** is the device that sends the data message. It can be a computer, workstation, telephone handset, video camera, and so on.

3. **Receiver.** The **receiver** is the device that receives the message. It can be a computer, workstation, telephone handset, television, and so on.
4. **Transmission medium.** The **transmission medium** is the physical path by which a message travels from sender to receiver. Some examples of transmission media include twisted-pair wire, coaxial cable, fiber-optic cable, and radio waves.
5. **Protocol.** A protocol is a set of rules that govern data communications. It represents an agreement between the communicating devices. Without a protocol, two devices may be connected but not communicating, just as a person speaking French cannot be understood by a person who speaks only Japanese.

1.1.2 Data Representation

Information today comes in different forms such as text, numbers, images, audio, and video.

Text

In data communications, text is represented as a bit pattern, a sequence of bits (0s or 1s). Different sets of bit patterns have been designed to represent text symbols. Each set is called a **code**, and the process of representing symbols is called coding. Today, the prevalent coding system is called **Unicode**, which uses 32 bits to represent a symbol or character used in any language in the world. The **American Standard Code for Information Interchange (ASCII)**, developed some decades ago in the United States, now constitutes the first 127 characters in Unicode and is also referred to as **Basic Latin**. Appendix A includes part of the Unicode.

Numbers

Numbers are also represented by bit patterns. However, a code such as ASCII is not used to represent numbers; the number is directly converted to a binary number to simplify mathematical operations. Appendix B discusses several different numbering systems.

Images

Images are also represented by bit patterns. In its simplest form, an image is composed of a matrix of pixels (picture elements), where each pixel is a small dot. The size of the pixel depends on the *resolution*. For example, an image can be divided into 1000 pixels or 10,000 pixels. In the second case, there is a better representation of the image (better resolution), but more memory is needed to store the image.

After an image is divided into pixels, each pixel is assigned a bit pattern. The size and the value of the pattern depend on the image. For an image made of only black-and-white dots (e.g., a chessboard), a 1-bit pattern is enough to represent a pixel.

If an image is not made of pure white and pure black pixels, we can increase the size of the bit pattern to include gray scale. For example, to show four levels of gray scale, we can use 2-bit patterns. A black pixel can be represented by 00, a dark gray pixel by 01, a light gray pixel by 10, and a white pixel by 11.

There are several methods to represent color images. One method is called **RGB**, so called because each color is made of a combination of three primary colors: *red*, *green*, and *blue*. The intensity of each color is measured, and a bit pattern is assigned to

it. Another method is called **YCM**, in which a color is made of a combination of three other primary colors: yellow, cyan, and magenta.

Audio

Audio refers to the recording or broadcasting of sound or music. Audio is by nature different from text, numbers, or images. It is continuous, not discrete. Even when we use a microphone to change voice or music to an electric signal, we create a continuous signal. We will learn more about audio in Chapter 26.

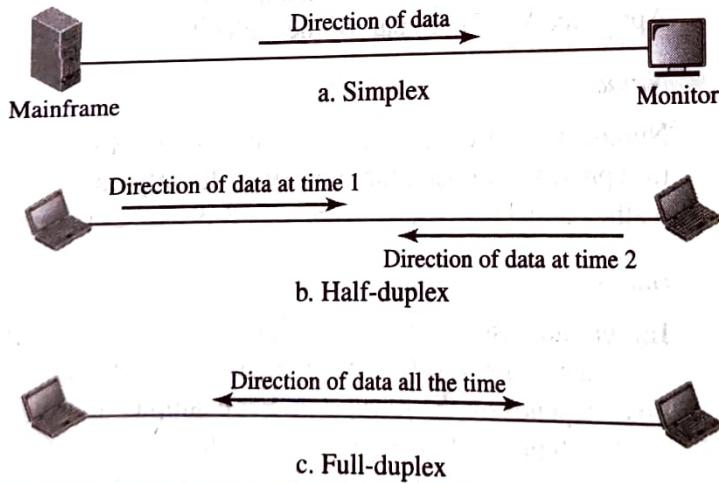
Video

Video refers to the recording or broadcasting of a picture or movie. Video can either be produced as a continuous entity (e.g., by a TV camera), or it can be a combination of images, each a discrete entity, arranged to convey the idea of motion. We will learn more about video in Chapter 26.

1.1.3 Data Flow

Communication between two devices can be simplex, half-duplex, or full-duplex as shown in Figure 1.2.

Figure 1.2 Data flow (simplex, half-duplex, and full-duplex)



Simplex

In **simplex mode**, the communication is unidirectional, as on a one-way street. Only one of the two devices on a link can transmit; the other can only receive (see Figure 1.2a).

Keyboards and traditional monitors are examples of simplex devices. The keyboard can only introduce input; the monitor can only accept output. The simplex mode can use the entire capacity of the channel to send data in one direction.

Half-Duplex

In **half-duplex mode**, each station can both transmit and receive, but not at the same time. When one device is sending, the other can only receive, and vice versa (see Figure 1.2b).

The half-duplex mode is like a one-lane road with traffic allowed in both directions. When cars are traveling in one direction, cars going the other way must wait. In a half-duplex transmission, the entire capacity of a channel is taken over by whichever of the two devices is transmitting at the time. Walkie-talkies and CB (citizens band) radios are both half-duplex systems.

The half-duplex mode is used in cases where there is no need for communication in both directions at the same time; the entire capacity of the channel can be utilized for each direction.

Full-Duplex

In **full-duplex mode** (also called *duplex*), both stations can transmit and receive simultaneously (see Figure 1.2c).

The full-duplex mode is like a two-way street with traffic flowing in both directions at the same time. In full-duplex mode, signals going in one direction share the capacity of the link with signals going in the other direction. This sharing can occur in two ways: Either the link must contain two physically separate transmission paths, one for sending and the other for receiving; or the capacity of the channel is divided between signals traveling in both directions.

One common example of full-duplex communication is the telephone network. When two people are communicating by a telephone line, both can talk and listen at the same time.

The full-duplex mode is used when communication in both directions is required all the time. The capacity of the channel, however, must be divided between the two directions.

1.2 NETWORKS

A **network** is the interconnection of a set of devices capable of communication. In this definition, a device can be a **host** (or an *end system* as it is sometimes called) such as a large computer, desktop, laptop, workstation, cellular phone, or security system. A device in this definition can also be a **connecting device** such as a router which connects the network to other networks, a switch, which connects devices together, a modem (modulator-demodulator), which changes the form of data, and so on. These devices in a network are connected using wired or wireless transmission media such as cable or air. When we connect two computers at home using a plug-and-play router, we have created a network, although very small.

1.2.1 Network Criteria

A network must be able to meet a certain number of criteria. The most important of these are performance, reliability, and security.

Performance

Performance can be measured in many ways, including transit time and response time. Transit time is the amount of time required for a message to travel from one device to another. Response time is the elapsed time between an inquiry and a response. The performance of a network depends on a number of factors, including the number of users, the type of transmission medium, the capabilities of the connected hardware, and the efficiency of the software.

Performance is often evaluated by two networking metrics: **throughput** and **delay**. We often need more throughput and less delay. However, these two criteria are often contradictory. If we try to send more data to the network, we may increase throughput but we increase the delay because of traffic congestion in the network.

Reliability

In addition to accuracy of delivery, network **reliability** is measured by the frequency of failure, the time it takes a link to recover from a failure, and the network's robustness in a catastrophe.

Security

Network **security** issues include protecting data from unauthorized access, protecting data from damage and development, and implementing policies and procedures for recovery from breaches and data losses.

1.2.2 Physical Structures

Before discussing networks, we need to define some network attributes.

Type of Connection

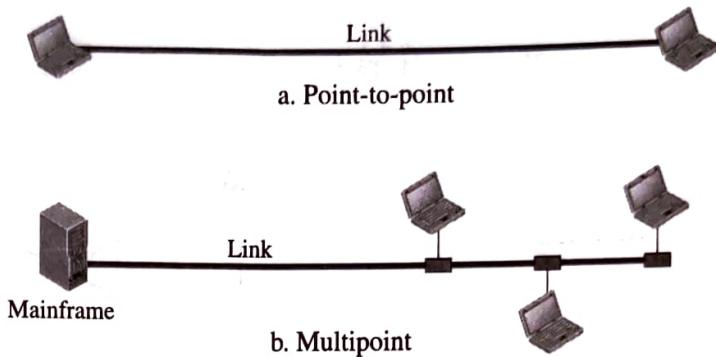
A network is two or more devices connected through links. A link is a communications pathway that transfers data from one device to another. For visualization purposes, it is simplest to imagine any link as a line drawn between two points. For communication to occur, two devices must be connected in some way to the same link at the same time. There are two possible types of connections: point-to-point and multipoint.

Point-to-Point

A **point-to-point connection** provides a dedicated link between two devices. The entire capacity of the link is reserved for transmission between those two devices. Most point-to-point connections use an actual length of wire or cable to connect the two ends, but other options, such as microwave or satellite links, are also possible (see Figure 1.3a). When we change television channels by infrared remote control, we are establishing a point-to-point connection between the remote control and the television's control system.

Multipoint

A **multipoint** (also called **multidrop**) **connection** is one in which more than two specific devices share a single link (see Figure 1.3b).

Figure 1.3 Types of connections: point-to-point and multipoint

In a multipoint environment, the capacity of the channel is shared, either spatially or temporally. If several devices can use the link simultaneously, it is a *spatially shared* connection. If users must take turns, it is a *timeshared* connection.

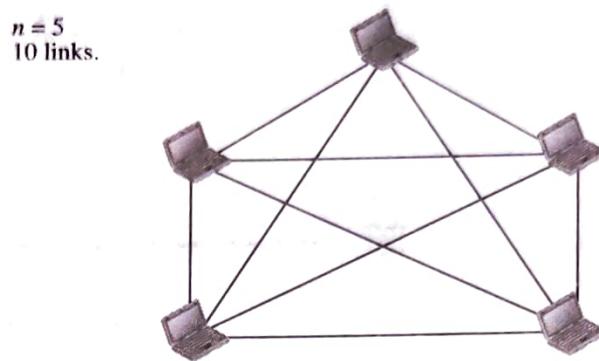
Physical Topology

The term **physical topology** refers to the way in which a network is laid out physically. Two or more devices connect to a link; two or more links form a topology. The topology of a network is the geometric representation of the relationship of all the links and linking devices (usually called **nodes**) to one another. There are four basic topologies possible: mesh, star, bus, and ring.

Mesh Topology

In a **mesh topology**, every device has a dedicated point-to-point link to every other device. The term **dedicated** means that the link carries traffic only between the two devices it connects. To find the number of physical links in a fully connected mesh network with n nodes, we first consider that each node must be connected to every other node. Node 1 must be connected to $n - 1$ nodes, node 2 must be connected to $n - 1$ nodes, and finally node n must be connected to $n - 1$ nodes. We need $n(n - 1)$ physical links. However, if each physical link allows communication in both directions (duplex mode), we can divide the number of links by 2. In other words, we can say that in a mesh topology, we need $n(n - 1) / 2$ duplex-mode links. To accommodate that many links, every device on the network must have $n - 1$ input/output (I/O) ports (see Figure 1.4) to be connected to the other $n - 1$ stations.

A mesh offers several advantages over other network topologies. First, the use of dedicated links guarantees that each connection can carry its own data load, thus eliminating the traffic problems that can occur when links must be shared by multiple devices. Second, a **mesh topology is robust**. If one link becomes unusable, it does not incapacitate the entire system. Third, there is the advantage of privacy or security. When every message travels along a dedicated line, only the intended recipient sees it. Physical boundaries prevent other users from gaining access to messages. Finally, point-to-point links make fault identification and fault isolation easy. Traffic can be routed to avoid links with suspected problems. This facility enables the network manager to discover the precise location of the fault and aids in finding its cause and solution.

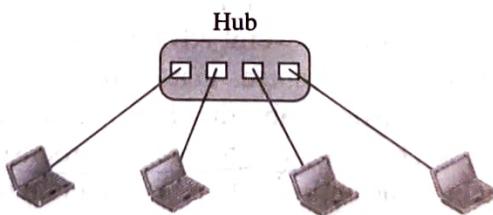
Figure 1.4 A fully connected mesh topology (five devices)

The main disadvantages of a mesh are related to the amount of cabling and the number of I/O ports required. First, because every device must be connected to every other device, installation and reconnection are difficult. Second, the sheer bulk of the wiring can be greater than the available space (in walls, ceilings, or floors) can accommodate. Finally, the hardware required to connect each link (I/O ports and cable) can be prohibitively expensive. For these reasons a mesh topology is usually implemented in a limited fashion, for example, as a backbone connecting the main computers of a hybrid network that can include several other topologies.

One practical example of a mesh topology is the connection of telephone regional offices in which each regional office needs to be connected to every other regional office.

Star Topology

In a star topology, each device has a dedicated point-to-point link only to a central controller, usually called a *hub*. The devices are not directly linked to one another. Unlike a mesh topology, a star topology does not allow direct traffic between devices. The controller acts as an exchange: If one device wants to send data to another, it sends the data to the controller, which then relays the data to the other connected device (see Figure 1.5).

Figure 1.5 A star topology connecting four stations

A star topology is less expensive than a mesh topology. In a star, each device needs only one link and one I/O port to connect it to any number of others. This factor also makes it easy to install and reconfigure. Far less cabling needs to be housed, and

additions, moves, and deletions involve only one connection: between that device and the hub.

Other advantages include robustness. If one link fails, only that link is affected. All other links remain active. This factor also lends itself to easy fault identification and fault isolation. As long as the hub is working, it can be used to monitor link problems and bypass defective links.

One big disadvantage of a star topology is the dependency of the whole topology on one single point, the hub. If the hub goes down, the whole system is dead.

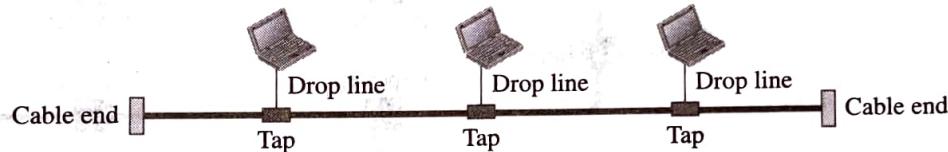
Although a star requires far less cable than a mesh, each node must be linked to a central hub. For this reason, often more cabling is required in a star than in some other topologies (such as ring or bus).

The star topology is used in local-area networks (LANs), as we will see in Chapter 13. High-speed LANs often use a star topology with a central hub.

Bus Topology

The preceding examples all describe point-to-point connections. A **bus topology**, on the other hand, is multipoint. One long cable acts as a **backbone** to link all the devices in a network (see Figure 1.6).

Figure 1.6 A bus topology connecting three stations



Nodes are connected to the bus cable by drop lines and taps. A drop line is a connection running between the device and the main cable. A tap is a connector that either splices into the main cable or punctures the sheathing of a cable to create a contact with the metallic core. As a signal travels along the backbone, some of its energy is transformed into heat. Therefore, it becomes weaker and weaker as it travels farther and farther. For this reason there is a limit on the number of taps a bus can support and on the distance between those taps.

Advantages of a bus topology include ease of installation. Backbone cable can be laid along the most efficient path, then connected to the nodes by drop lines of various lengths. In this way, a bus uses less cabling than mesh or star topologies. In a star, for example, four network devices in the same room require four lengths of cable reaching all the way to the hub. In a bus, this redundancy is eliminated. Only the backbone cable stretches through the entire facility. Each drop line has to reach only as far as the nearest point on the backbone.

Disadvantages include difficult reconnection and fault isolation. A bus is usually designed to be optimally efficient at installation. It can therefore be difficult to add new devices. Signal reflection at the taps can cause degradation in quality. This degradation can be controlled by limiting the number and spacing of devices connected to a given

length of cable. Adding new devices may therefore require modification or replacement of the backbone.

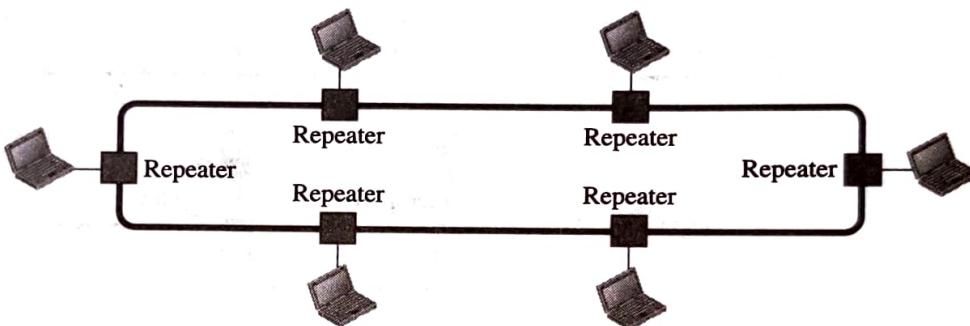
In addition, a fault or break in the bus cable stops all transmission, even between devices on the same side of the problem. The damaged area reflects signals back in the direction of origin, creating noise in both directions.

Bus topology was the one of the first topologies used in the design of early local-area networks. Traditional Ethernet LANs can use a bus topology, but they are less popular now for reasons we will discuss in Chapter 13.

Ring Topology

In a **ring topology**, each device has a dedicated point-to-point connection with only the two devices on either side of it. A signal is passed along the ring in one direction, from device to device, until it reaches its destination. Each device in the ring incorporates a repeater. When a device receives a signal intended for another device, its repeater regenerates the bits and passes them along (see Figure 1.7).

Figure 1.7 A ring topology connecting six stations



A ring is relatively easy to install and reconfigure. Each device is linked to only its immediate neighbors (either physically or logically). To add or delete a device requires changing only two connections. The only constraints are media and traffic considerations (maximum ring length and number of devices). In addition, fault isolation is simplified. Generally, in a ring a signal is circulating at all times. If one device does not receive a signal within a specified period, it can issue an alarm. The alarm alerts the network operator to the problem and its location.

However, unidirectional traffic can be a disadvantage. In a simple ring, a break in the ring (such as a disabled station) can disable the entire network. This weakness can be solved by using a dual ring or a switch capable of closing off the break.

Ring topology was prevalent when IBM introduced its local-area network, Token Ring. Today, the need for higher-speed LANs has made this topology less popular.

1.3 NETWORK TYPES

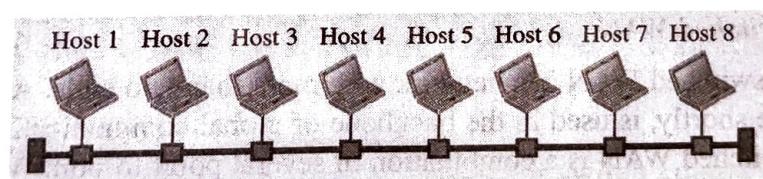
After defining networks in the previous section and discussing their physical structures, we need to discuss different types of networks we encounter in the world today. The criteria of distinguishing one type of network from another is difficult and sometimes confusing. We use a few criteria such as size, geographical coverage, and ownership to make this distinction. After discussing two types of networks, LANs and WANs, we define switching, which is used to connect networks to form an internetwork (a network of networks).

1.3.1 Local Area Network

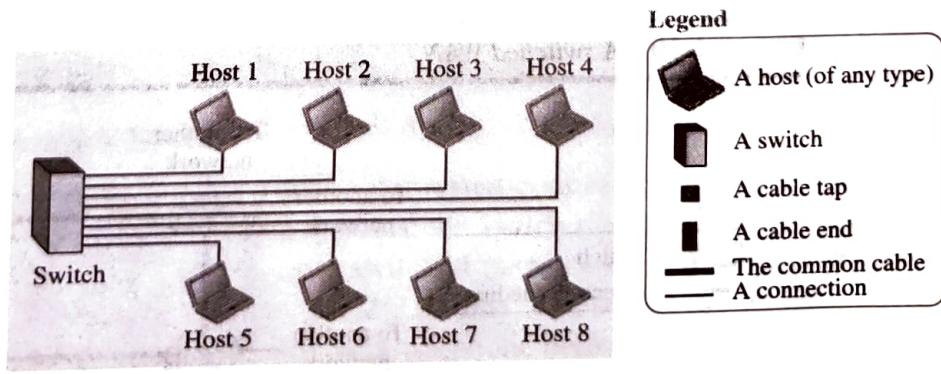
A **local area network (LAN)** is usually privately owned and connects some hosts in a single office, building, or campus. Depending on the needs of an organization, a LAN can be as simple as two PCs and a printer in someone's home office, or it can extend throughout a company and include audio and video devices. Each host in a LAN has an identifier, an address, that uniquely defines the host in the LAN. A packet sent by a host to another host carries both the source host's and the destination host's addresses.

In the past, all hosts in a network were connected through a common cable, which meant that a packet sent from one host to another was received by all hosts. The intended recipient kept the packet; the others dropped the packet. Today, most LANs use a smart connecting switch, which is able to recognize the destination address of the packet and guide the packet to its destination without sending it to all other hosts. The switch alleviates the traffic in the LAN and allows more than one pair to communicate with each other at the same time if there is no common source and destination among them. Note that the above definition of a LAN does not define the minimum or maximum number of hosts in a LAN. Figure 1.8 shows a LAN using either a common cable or a switch.

Figure 1.8 An isolated LAN in the past and today



a. LAN with a common cable (past)



b. LAN with a switch (today)

LANs are discussed in more detail in Part III of the book.

When LANs were used in isolation (which is rare today), they were designed to allow resources to be shared between the hosts. As we will see shortly, LANs today are connected to each other and to WANs (discussed next) to create communication at a wider level.

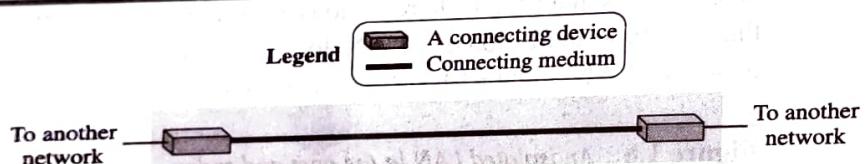
1.3.2 Wide Area Network

A **wide area network (WAN)** is also an interconnection of devices capable of communication. However, there are some differences between a LAN and a WAN. A LAN is normally limited in size, spanning an office, a building, or a campus; a WAN has a wider geographical span, spanning a town, a state, a country, or even the world. A LAN interconnects hosts; a WAN interconnects connecting devices such as switches, routers, or modems. A LAN is normally privately owned by the organization that uses it; a WAN is normally created and run by communication companies and leased by an organization that uses it. We see two distinct examples of WANs today: point-to-point WANs and switched WANs.

Point-to-Point WAN

A point-to-point WAN is a network that connects two communicating devices through a transmission media (cable or air). We will see examples of these WANs when we discuss how to connect the networks to one another. Figure 1.9 shows an example of a point-to-point WAN.

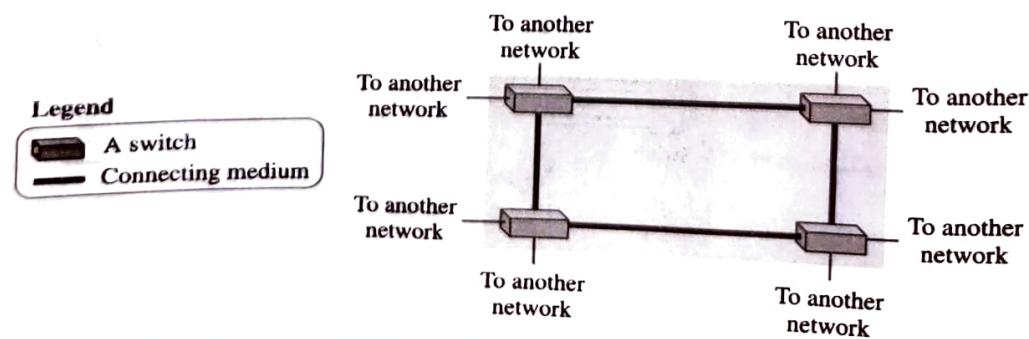
Figure 1.9 A point-to-point WAN



Switched WAN

A **switched WAN** is a network with more than two ends. A switched WAN, as we will see shortly, is used in the backbone of global communication today. We can say that a **switched WAN** is a combination of several point-to-point WANs that are connected by switches. Figure 1.10 shows an example of a switched WAN.

Figure 1.10 A switched WAN

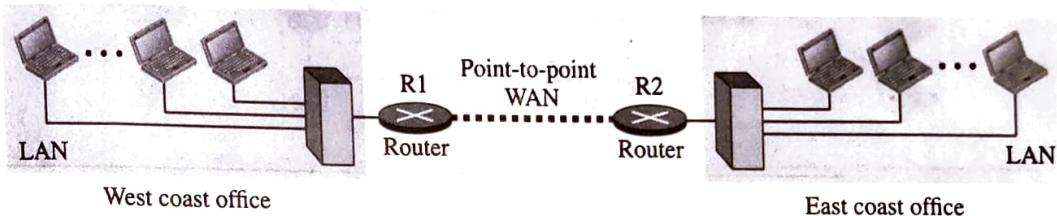


WANs are discussed in more detail in Part II of the book.

Internetwork

Today, it is very rare to see a LAN or a WAN in isolation; they are connected to one another. When two or more networks are connected, they make an **internetwork**, or **internet**. As an example, assume that an organization has two offices, one on the east coast and the other on the west coast. Each office has a LAN that allows all employees in the office to communicate with each other. To make the communication between employees at different offices possible, the management leases a point-to-point dedicated WAN from a service provider, such as a telephone company, and connects the two LANs. Now the company has an internetwork, or a private internet (with lowercase *i*). Communication between offices is now possible. Figure 1.11 shows this internet.

Figure 1.11 An internetwork made of two LANs and one point-to-point WAN



When a host in the west coast office sends a message to another host in the same office, the router blocks the message, but the switch directs the message to the destination. On the other hand, when a host on the west coast sends a message to a host on the east coast, router R1 routes the packet to router R2, and the packet reaches the destination.

Figure 1.12 (see next page) shows another internet with several LANs and WANs connected. One of the WANs is a switched WAN with four switches.

1.3.3 Switching

An internet is a **switched network** in which a switch connects at least two links together. A switch needs to forward data from a network to another network when required. The two most common types of switched networks are circuit-switched and packet-switched networks. We discuss both next.

Circuit-Switched Network

In a **circuit-switched network**, a dedicated connection, called a circuit, is always available between the two end systems; the switch can only make it active or inactive. Figure 1.13 shows a very simple switched network that connects four telephones to each end. We have used telephone sets instead of computers as an end system because circuit switching was very common in telephone networks in the past, although part of the telephone network today is a packet-switched network.

In Figure 1.13, the four telephones at each side are connected to a switch. The switch connects a telephone set at one side to a telephone set at the other side. The thick

Figure 1.12 A heterogeneous network made of four WANs and three LANs

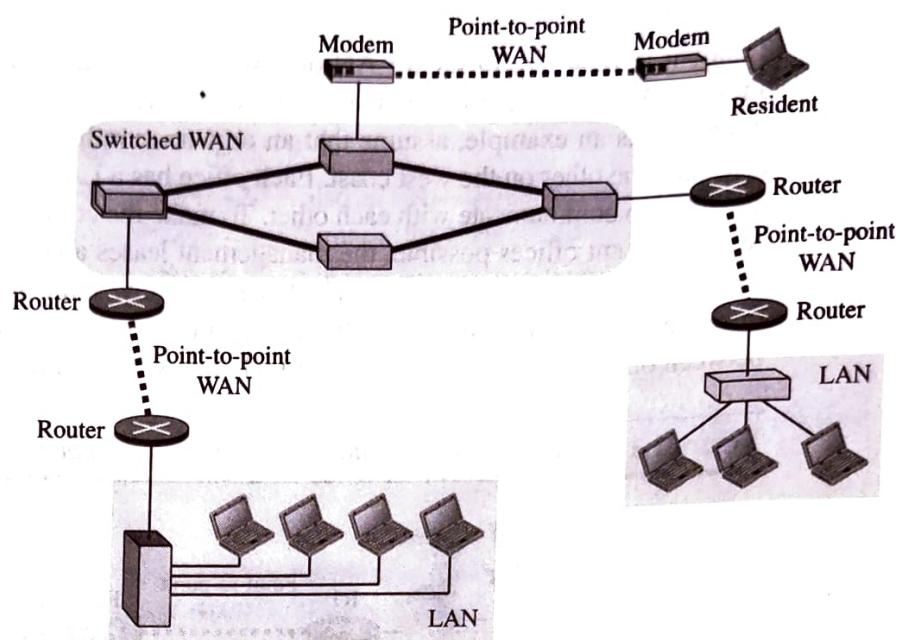
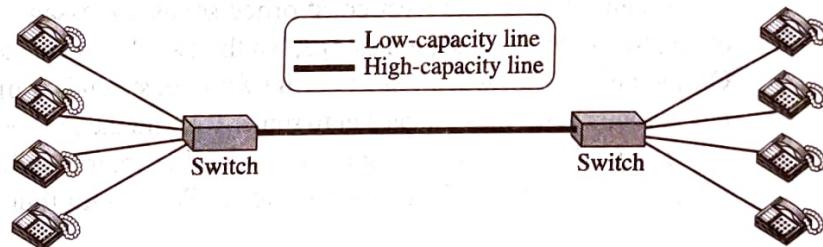


Figure 1.13 A circuit-switched network



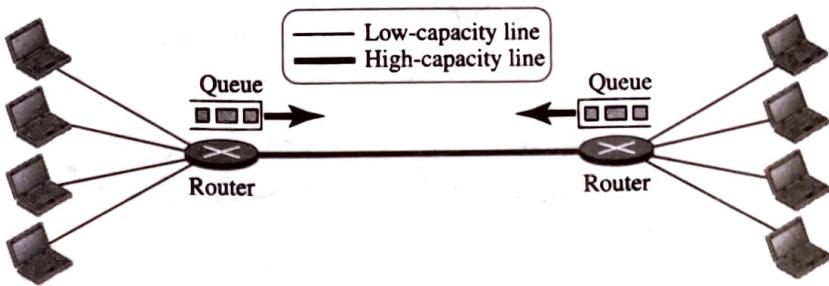
line connecting two switches is a high-capacity communication line that can handle four voice communications at the same time; the capacity can be shared between all pairs of telephone sets. The switches used in this example have forwarding tasks but no storing capability.

Let us look at two cases. In the first case, all telephone sets are busy; four people at one site are talking with four people at the other site; the capacity of the thick line is fully used. In the second case, only one telephone set at one side is connected to a telephone set at the other side; only one-fourth of the capacity of the thick line is used. This means that a circuit-switched network is efficient only when it is working at its full capacity; most of the time, it is inefficient because it is working at partial capacity. The reason that we need to make the capacity of the thick line four times the capacity of each voice line is that we do not want communication to fail when all telephone sets at one side want to be connected with all telephone sets at the other side.

Packet-Switched Network

In a computer network, the communication between the two ends is done in blocks of data called **packets**. In other words, instead of the continuous communication we see between two telephone sets when they are being used, we see the exchange of individual data packets between the two computers. This allows us to make the switches function for both storing and forwarding because a packet is an independent entity that can be stored and sent later. Figure 1.14 shows a small packet-switched network that connects four computers at one site to four computers at the other site.

Figure 1.14 A packet-switched network



A router in a packet-switched network has a queue that can store and forward the packet. Now assume that the capacity of the thick line is only twice the capacity of the data line connecting the computers to the routers. If only two computers (one at each site) need to communicate with each other, there is no waiting for the packets. However, if packets arrive at one router when the thick line is already working at its full capacity, the packets should be stored and forwarded in the order they arrived. The two simple examples show that a packet-switched network is more efficient than a circuit-switched network, but the packets may encounter some delays.

In this book, we mostly discuss packet-switched networks. In Chapter 18, we discuss packet-switched networks in more detail and discuss the performance of these networks.

1.3.4 The Internet

As we discussed before, an **internet** (note the lowercase *i*) is two or more networks that can communicate with each other. The most notable internet is called the **Internet** (uppercase *I*), and is composed of thousands of interconnected networks. Figure 1.15 shows a conceptual (not geographical) view of the Internet.

The figure shows the Internet as several backbones, provider networks, and customer networks. At the top level, the *backbones* are large networks owned by some communication companies such as Sprint, Verizon (MCI), AT&T, and NTT. The backbone networks are connected through some complex switching systems, called *peering points*. At the second level, there are smaller networks, called *provider networks*, that use the services of the backbones for a fee. The provider networks are connected to backbones and sometimes to other provider networks. The *customer networks* are

Protocol layering enables us to divide a complex task into several smaller and simpler tasks. For example, in Figure 2.2, we could have used only one machine to do the job of all three machines. However, if Maria and Ann decide that the encryption/decryption done by the machine is not enough to protect their secrecy, they would have to change the whole machine. In the present situation, they need to change only the second layer machine; the other two can remain the same. This is referred to as *modularity*. Modularity in this case means independent layers. A layer (module) can be defined as a black box with inputs and outputs, without concern about how inputs are changed to outputs. If two machines provide the same outputs when given the same inputs, they can replace each other. For example, Ann and Maria can buy the second layer machine from two different manufacturers. As long as the two machines create the same ciphertext from the same plaintext and vice versa, they do the job.

One of the advantages of protocol layering is that it allows us to separate the services from the implementation. A layer needs to be able to receive a set of services from the lower layer and to give the services to the upper layer; we don't care about how the layer is implemented. For example, Maria may decide not to buy the machine (robot) for the first layer; she can do the job herself. As long as Maria can do the tasks provided by the first layer, in both directions, the communication system works.

Another advantage of protocol layering, which cannot be seen in our simple examples but reveals itself when we discuss protocol layering in the Internet, is that communication does not always use only two end systems; there are intermediate systems that need only some layers, but not all layers. If we did not use protocol layering, we would have to make each intermediate system as complex as the end systems, which makes the whole system more expensive.

Is there any disadvantage to protocol layering? One can argue that having a single layer makes the job easier. There is no need for each layer to provide a service to the upper layer and give service to the lower layer. For example, Ann and Maria could find or build one machine that could do all three tasks. However, as mentioned above, if one day they found that their code was broken, each would have to replace the whole machine with a new one instead of just changing the machine in the second layer.

2.1.2 Principles of Protocol Layering

Let us discuss two principles of protocol layering.

First Principle

The first principle dictates that if we want bidirectional communication, we need to make each layer so that it is able to perform two opposite tasks, one in each direction. For example, the third layer task is to listen (in one direction) and *talk* (in the other direction). The second layer needs to be able to encrypt and decrypt. The first layer needs to send and receive mail.

Second Principle

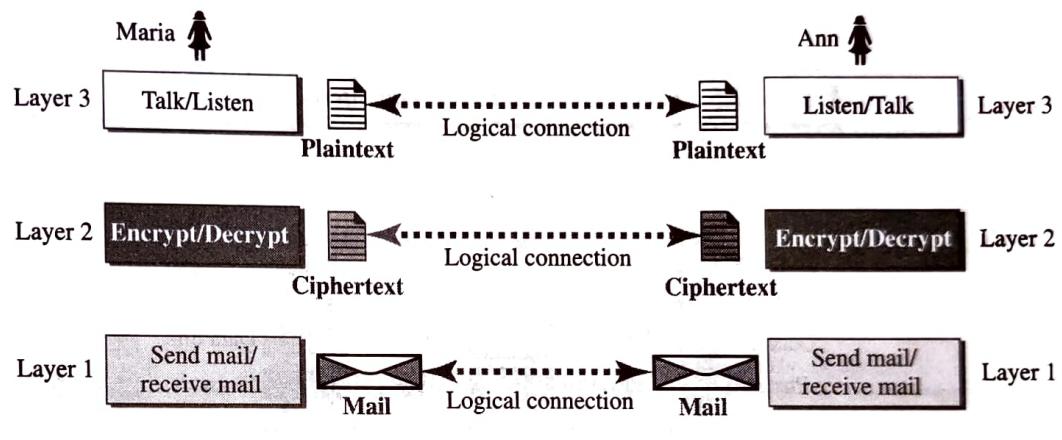
The second principle that we need to follow in protocol layering is that the two objects under each layer at both sites should be identical. For example, the object under layer 3 at both sites should be a plaintext letter. The object under layer 2 at

both sites should be a ciphertext letter. The object under layer 1 at both sites should be a piece of mail.

2.1.3 Logical Connections

After following the above two principles, we can think about logical connection between each layer as shown in Figure 2.3. This means that we have layer-to-layer communication. Maria and Ann can think that there is a logical (imaginary) connection at each layer through which they can send the object created from that layer. We will see that the concept of logical connection will help us better understand the task of layering we encounter in data communication and networking.

Figure 2.3 Logical connection between peer layers

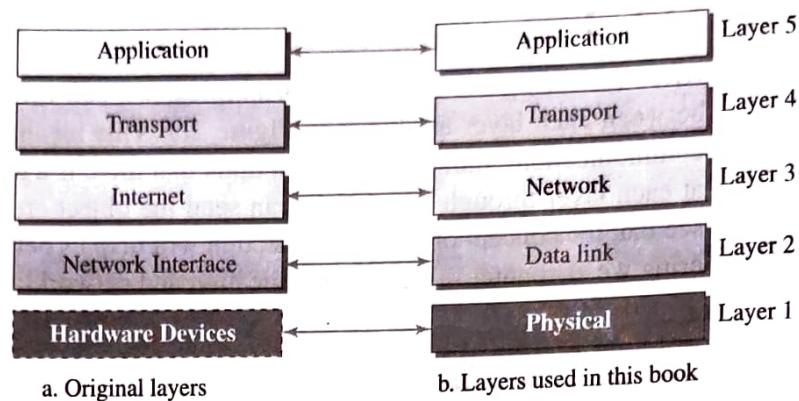
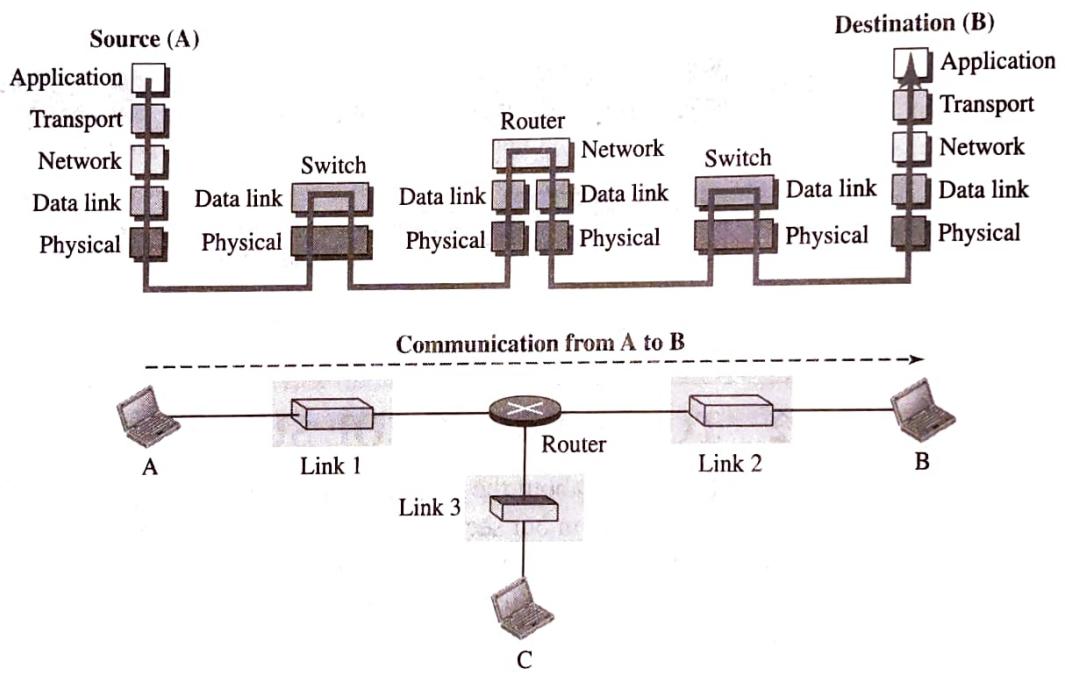


2.2 TCP/IP PROTOCOL SUITE

Now that we know about the concept of protocol layering and the logical communication between layers in our second scenario, we can introduce the TCP/IP (Transmission Control Protocol/Internet Protocol). TCP/IP is a protocol suite (a set of protocols organized in different layers) used in the Internet today. It is a hierarchical protocol made up of interactive modules, each of which provides a specific functionality. The term *hierarchical* means that each upper level protocol is supported by the services provided by one or more lower level protocols. The original TCP/IP protocol suite was defined as four software layers built upon the hardware. Today, however, TCP/IP is thought of as a five-layer model. Figure 2.4 shows both configurations.

2.2.1 Layered Architecture

To show how the layers in the TCP/IP protocol suite are involved in communication between two hosts, we assume that we want to use the suite in a small internet made up of three LANs (links), each with a link-layer switch. We also assume that the links are connected by one router, as shown in Figure 2.5.

Figure 2.4 Layers in the TCP/IP protocol suite**Figure 2.5** Communication through an internet

Let us assume that computer A communicates with computer B. As the figure shows, we have five communicating devices in this communication: source host (computer A), the link-layer switch in link 1, the router, the link-layer switch in link 2, and the destination host (computer B). Each device is involved with a set of layers depending on the role of the device in the internet. The two hosts are involved in all five layers; the source host needs to create a message in the application layer and send it down the layers so that it is physically sent to the destination host. The destination host needs to receive the communication at the physical layer and then deliver it through the other layers to the application layer.

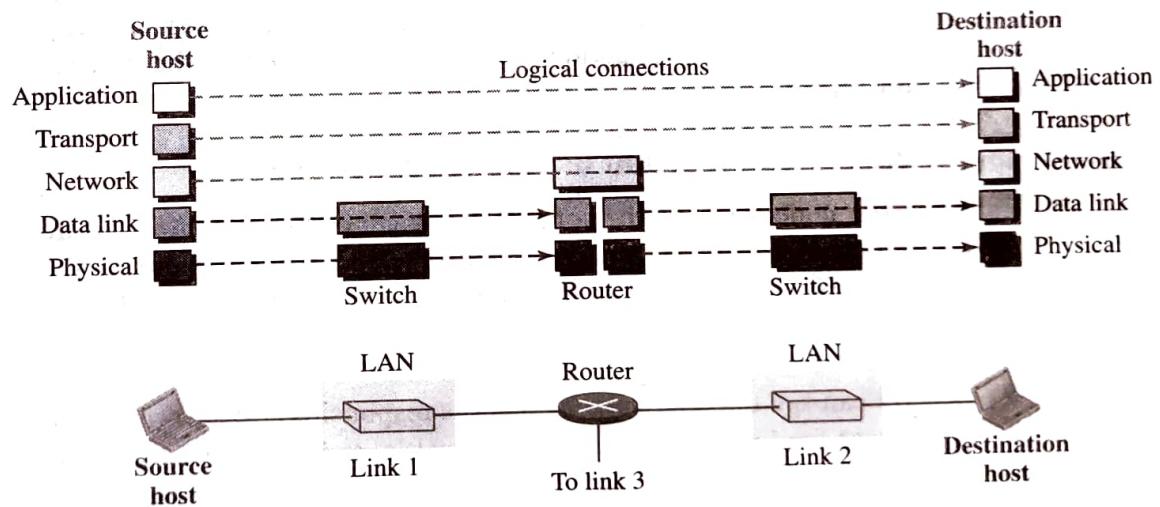
The router is involved in only three layers; there is no transport or application layer in a router as long as the router is used only for routing. Although a router is always involved in one network layer, it is involved in n combinations of link and physical layers in which n is the number of links the router is connected to. The reason is that each link may use its own data-link or physical protocol. For example, in the above figure, the router is involved in three links, but the message sent from source A to destination B is involved in two links. Each link may be using different link-layer and physical-layer protocols; the router needs to receive a packet from link 1 based on one pair of protocols and deliver it to link 2 based on another pair of protocols.

A link-layer switch in a link, however, is involved only in two layers, data-link and physical. Although each switch in the above figure has two different connections, the connections are in the same link, which uses only one set of protocols. This means that, unlike a router, a link-layer switch is involved only in one data-link and one physical layer.

2.2.2 Layers in the TCP/IP Protocol Suite

After the above introduction, we briefly discuss the functions and duties of layers in the TCP/IP protocol suite. Each layer is discussed in detail in the next five parts of the book. To better understand the duties of each layer, we need to think about the logical connections between layers. Figure 2.6 shows logical connections in our simple internet.

Figure 2.6 Logical connections between layers of the TCP/IP protocol suite



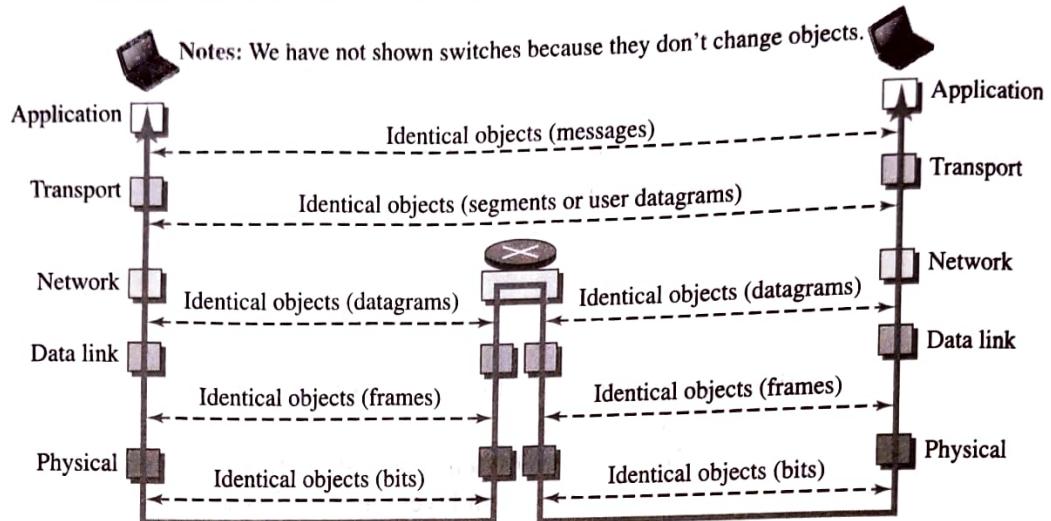
Using logical connections makes it easier for us to think about the duty of each layer. As the figure shows, the duty of the application, transport, and network layers is end-to-end. However, the duty of the data-link and physical layers is hop-to-hop, in which a hop is a host or router. In other words, the domain of duty of the top three layers is the internet, and the domain of duty of the two lower layers is the link.

Another way of thinking of the logical connections is to think about the data unit created from each layer. In the top three layers, the data unit (packets) should not be

changed by any router or link-layer switch. In the bottom two layers, the packet created by the host is changed only by the routers, not by the link-layer switches.

Figure 2.7 shows the second principle discussed previously for protocol layering. We show the identical objects below each layer related to each device.

Figure 2.7 Identical objects in the TCP/IP protocol suite



Note that, although the logical connection at the network layer is between the two hosts, we can only say that identical objects exist between two hops in this case because a router may fragment the packet at the network layer and send more packets than received (see fragmentation in Chapter 19). Note that the link between two hops does not change the object.

2.2.3 Description of Each Layer

After understanding the concept of logical communication, we are ready to briefly discuss the duty of each layer. Our discussion in this chapter will be very brief, but we come back to the duty of each layer in next five parts of the book.

Physical Layer

We can say that the physical layer is responsible for carrying individual bits in a frame across the link. Although the physical layer is the lowest level in the TCP/IP protocol suite, the communication between two devices at the physical layer is still a logical communication because there is another, hidden layer, the transmission media, under the physical layer. Two devices are connected by a transmission medium (cable or air).

We need to know that the transmission medium does not carry bits; it carries electrical or optical signals. So the bits received in a frame from the data-link layer are transformed and sent through the transmission media, but we can think that the logical unit between two physical layers in two devices is a bit. There are several protocols that transform a bit to a signal. We discuss them in Part II when we discuss the physical layer and the transmission media.

Data-link Layer

We have seen that an internet is made up of several links (LANs and WANs) connected by routers. There may be several overlapping sets of links that a datagram can travel from the host to the destination. The routers are responsible for choosing the *best* links. However, when the next link to travel is determined by the router, the data-link layer is responsible for taking the datagram and moving it across the link. The link can be a wired LAN with a link-layer switch, a wireless LAN, a wired WAN, or a wireless WAN. We can also have different protocols used with any link type. In each case, the data-link layer is responsible for moving the packet through the link.

TCP/IP does not define any specific protocol for the data-link layer. It supports all the standard and proprietary protocols. Any protocol that can take the datagram and carry it through the link suffices for the network layer. The data-link layer takes a datagram and encapsulates it in a packet called a *frame*.

Each link-layer protocol may provide a different service. Some link-layer protocols provide complete error detection and correction, some provide only error correction. We discuss wired links in Chapters 13 and 14 and wireless links in Chapters 15 and 16.

Network Layer

The network layer is responsible for creating a connection between the source computer and the destination computer. The communication at the network layer is host-to-host. However, since there can be several routers from the source to the destination, the routers in the path are responsible for choosing the best route for each packet. We can say that the network layer is responsible for host-to-host communication and routing the packet through possible routes. Again, we may ask ourselves why we need the network layer. We could have added the routing duty to the transport layer and dropped this layer. One reason, as we said before, is the separation of different tasks between different layers. The second reason is that the routers do not need the application and transport layers. Separating the tasks allows us to use fewer protocols on the routers.

The network layer in the Internet includes the main protocol, Internet Protocol (IP), that defines the format of the packet, called a datagram at the network layer. IP also defines the format and the structure of addresses used in this layer. IP is also responsible for routing a packet from its source to its destination, which is achieved by each router forwarding the datagram to the next router in its path.

IP is a connectionless protocol that provides no flow control, no error control, and no congestion control services. This means that if any of these services is required for an application, the application should rely only on the transport-layer protocol. The network layer also includes unicast (one-to-one) and multicast (one-to-many) routing protocols. A routing protocol does not take part in routing (it is the responsibility of IP), but it creates forwarding tables for routers to help them in the routing process.

The network layer also has some auxiliary protocols that help IP in its delivery and routing tasks. The Internet Control Message Protocol (ICMP) helps IP to report some problems when routing a packet. The Internet Group Management Protocol (IGMP) is another protocol that helps IP in multitasking. The Dynamic Host Configuration Protocol (DHCP) helps IP to get the network-layer address for a host. The Address Resolution Protocol (ARP) is a protocol that helps IP to find the link-layer address of a host or

a router when its network-layer address is given. ARP is discussed in Chapter 9, ICMP in Chapter 19, and IGMP in Chapter 21.

Transport Layer

The logical connection at the transport layer is also end-to-end. The transport layer at the source host gets the message from the application layer, encapsulates it in a transport-layer packet (called a *segment* or a *user datagram* in different protocols) and sends it, through the logical (imaginary) connection, to the transport layer at the destination host. In other words, the transport layer is responsible for giving services to the application layer: to get a message from an application program running on the source host and deliver it to the corresponding application program on the destination host. We may ask why we need an end-to-end transport layer when we already have an end-to-end application layer. The reason is the separation of tasks and duties, which we discussed earlier. The transport layer should be independent of the application layer. In addition, we will see that we have more than one protocol in the transport layer, which means that each application program can use the protocol that best matches its requirement.

As we said, there are a few transport-layer protocols in the Internet, each designed for some specific task. The main protocol, Transmission Control Protocol (TCP), is a connection-oriented protocol that first establishes a logical connection between transport layers at two hosts before transferring data. It creates a logical pipe between two TCPs for transferring a stream of bytes. TCP provides flow control (matching the sending data rate of the source host with the receiving data rate of the destination host to prevent overwhelming the destination), error control (to guarantee that the segments arrive at the destination without error and resending the corrupted ones), and congestion control to reduce the loss of segments due to congestion in the network. The other common protocol, User Datagram Protocol (UDP), is a connectionless protocol that transmits user datagrams without first creating a logical connection. In UDP, each user datagram is an independent entity without being related to the previous or the next one (the meaning of the term *connectionless*). UDP is a simple protocol that does not provide flow, error, or congestion control. Its simplicity, which means small overhead, is attractive to an application program that needs to send short messages and cannot afford the retransmission of the packets involved in TCP, when a packet is corrupted or lost. A new protocol, Stream Control Transmission Protocol (SCTP) is designed to respond to new applications that are emerging in the multimedia. We will discuss UDP, TCP, and SCTP in Chapter 24.

Application Layer

As Figure 2.6 shows, the logical connection between the two application layers is end-to-end. The two application layers exchange *messages* between each other as though there were a bridge between the two layers. However, we should know that the communication is done through all the layers.

Communication at the application layer is between two *processes* (two programs running at this layer). To communicate, a process sends a request to the other process and receives a response. Process-to-process communication is the duty of the application layer. The application layer in the Internet includes many predefined protocols, but

a router when its network-layer address is given. ARP is discussed in Chapter 9, ICMP in Chapter 19, and IGMP in Chapter 21.

Transport Layer

The logical connection at the transport layer is also end-to-end. The transport layer at the source host gets the message from the application layer, encapsulates it in a transport-layer packet (called a *segment* or a *user datagram* in different protocols) and sends it, through the logical (imaginary) connection, to the transport layer at the destination host. In other words, the transport layer is responsible for giving services to the application layer: to get a message from an application program running on the source host and deliver it to the corresponding application program on the destination host. We may ask why we need an end-to-end transport layer when we already have an end-to-end application layer. The reason is the separation of tasks and duties, which we discussed earlier. The transport layer should be independent of the application layer. In addition, we will see that we have more than one protocol in the transport layer, which means that each application program can use the protocol that best matches its requirement.

As we said, there are a few transport-layer protocols in the Internet, each designed for some specific task. The main protocol, Transmission Control Protocol (TCP), is a connection-oriented protocol that first establishes a logical connection between transport layers at two hosts before transferring data. It creates a logical pipe between two TCPs for transferring a stream of bytes. TCP provides flow control (matching the sending data rate of the source host with the receiving data rate of the destination host to prevent overwhelming the destination), error control (to guarantee that the segments arrive at the destination without error and resending the corrupted ones), and congestion control to reduce the loss of segments due to congestion in the network. The other common protocol, User Datagram Protocol (UDP), is a connectionless protocol that transmits user datagrams without first creating a logical connection. In UDP, each user datagram is an independent entity without being related to the previous or the next one (the meaning of the term *connectionless*). UDP is a simple protocol that does not provide flow, error, or congestion control. Its simplicity, which means small overhead, is attractive to an application program that needs to send short messages and cannot afford the retransmission of the packets involved in TCP, when a packet is corrupted or lost. A new protocol, Stream Control Transmission Protocol (SCTP) is designed to respond to new applications that are emerging in the multimedia. We will discuss UDP, TCP, and SCTP in Chapter 24.

Application Layer

As Figure 2.6 shows, the logical connection between the two application layers is end-to-end. The two application layers exchange *messages* between each other as though there were a bridge between the two layers. However, we should know that the communication is done through all the layers.

Communication at the application layer is between two *processes* (two programs running at this layer). To communicate, a process sends a request to the other process and receives a response. Process-to-process communication is the duty of the application layer. The application layer in the Internet includes many predefined protocols, but

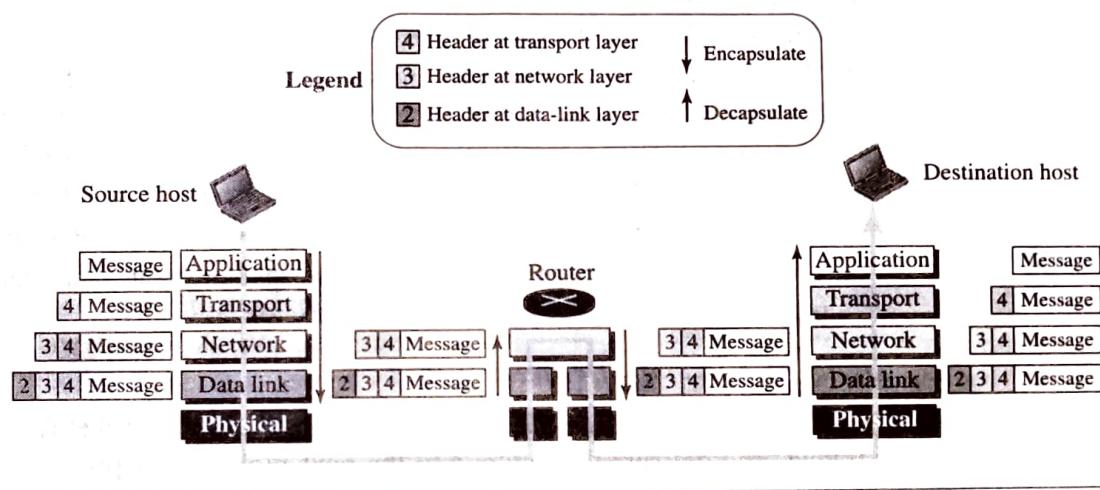
a user can also create a pair of processes to be run at the two hosts. In Chapter 25, we explore this situation.

The Hypertext Transfer Protocol (HTTP) is a vehicle for accessing the World Wide Web (WWW). The Simple Mail Transfer Protocol (SMTP) is the main protocol used in electronic mail (e-mail) service. The File Transfer Protocol (FTP) is used for transferring files from one host to another. The Terminal Network (TELNET) and Secure Shell (SSH) are used for accessing a site remotely. The Simple Network Management Protocol (SNMP) is used by an administrator to manage the Internet at global and local levels. The Domain Name System (DNS) is used by other protocols to find the network-layer address of a computer. The Internet Group Management Protocol (IGMP) is used to collect membership in a group. We discuss most of these protocols in Chapter 26 and some in other chapters.

2.2.4 Encapsulation and Decapsulation

One of the important concepts in protocol layering in the Internet is encapsulation/decapsulation. Figure 2.8 shows this concept for the small internet in Figure 2.5.

Figure 2.8 Encapsulation/Decapsulation



We have not shown the layers for the link-layer switches because no encapsulation/decapsulation occurs in this device. In Figure 2.8, we show the encapsulation in the source host, decapsulation in the destination host, and encapsulation and decapsulation in the router.

Encapsulation at the Source Host

At the source, we have only encapsulation.

1. At the application layer, the data to be exchanged is referred to as a *message*. A message normally does not contain any header or trailer, but if it does, we refer to the whole as the message. The message is passed to the transport layer.
2. The transport layer takes the message as the payload, the load that the transport layer should take care of. It adds the transport layer header to the payload, which contains the identifiers of the source and destination application programs that

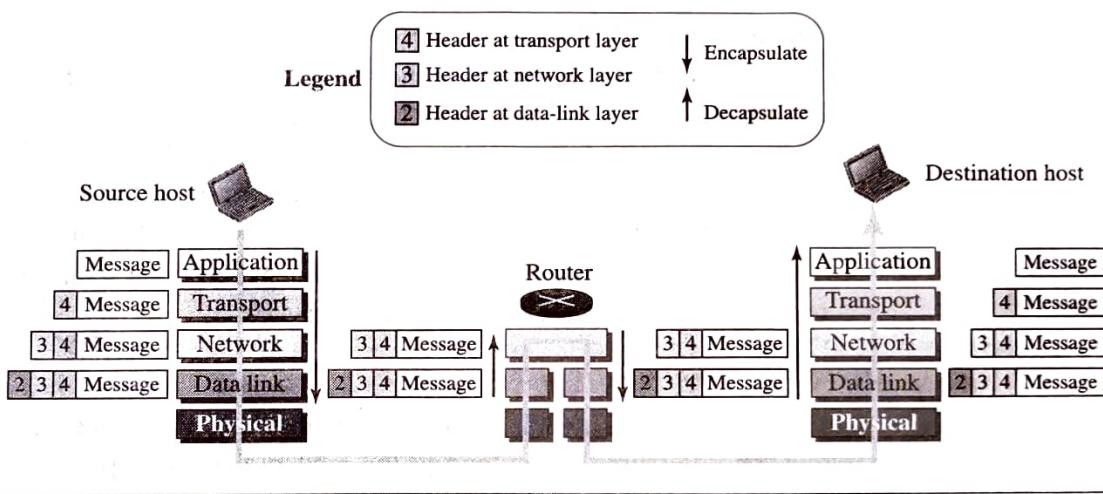
a user can also create a pair of processes to be run at the two hosts. In Chapter 25, we explore this situation.

The Hypertext Transfer Protocol (HTTP) is a vehicle for accessing the World Wide Web (WWW). The Simple Mail Transfer Protocol (SMTP) is the main protocol used in electronic mail (e-mail) service. The File Transfer Protocol (FTP) is used for transferring files from one host to another. The Terminal Network (TELNET) and Secure Shell (SSH) are used for accessing a site remotely. The Simple Network Management Protocol (SNMP) is used by an administrator to manage the Internet at global and local levels. The Domain Name System (DNS) is used by other protocols to find the network-layer address of a computer. The Internet Group Management Protocol (IGMP) is used to collect membership in a group. We discuss most of these protocols in Chapter 26 and some in other chapters.

2.2.4 Encapsulation and Decapsulation

One of the important concepts in protocol layering in the Internet is encapsulation/decapsulation. Figure 2.8 shows this concept for the small internet in Figure 2.5.

Figure 2.8 Encapsulation/Decapsulation



We have not shown the layers for the link-layer switches because no encapsulation/decapsulation occurs in this device. In Figure 2.8, we show the encapsulation in the source host, decapsulation in the destination host, and encapsulation and decapsulation in the router.

Encapsulation at the Source Host

At the source, we have only encapsulation.

1. At the application layer, the data to be exchanged is referred to as a *message*. A message normally does not contain any header or trailer, but if it does, we refer to the whole as the message. The message is passed to the transport layer.
2. The transport layer takes the message as the payload, the load that the transport layer should take care of. It adds the transport layer header to the payload, which contains the identifiers of the source and destination application programs that

want to communicate plus some more information that is needed for the end-to-end delivery of the message, such as information needed for flow, error control, or congestion control. The result is the transport-layer packet, which is called the *segment* (in TCP) and the *user datagram* (in UDP). The transport layer then passes the packet to the network layer.

3. The network layer takes the transport-layer packet as data or payload and adds its own header to the payload. The header contains the addresses of the source and destination hosts and some more information used for error checking of the header, fragmentation information, and so on. The result is the network-layer packet, called a *datagram*. The network layer then passes the packet to the data-link layer.
4. The data-link layer takes the network-layer packet as data or payload and adds its own header, which contains the link-layer addresses of the host or the next hop (the router). The result is the link-layer packet, which is called a *frame*. The frame is passed to the physical layer for transmission.

Decapsulation and Encapsulation at the Router

At the router, we have both decapsulation and encapsulation because the router is connected to two or more links.

1. After the set of bits are delivered to the data-link layer, this layer decapsulates the datagram from the frame and passes it to the network layer.
2. The network layer only inspects the source and destination addresses in the datagram header and consults its forwarding table to find the next hop to which the datagram is to be delivered. The contents of the datagram should not be changed by the network layer in the router unless there is a need to fragment the datagram if it is too big to be passed through the next link. The datagram is then passed to the data-link layer of the next link.
3. The data-link layer of the next link encapsulates the datagram in a frame and passes it to the physical layer for transmission.

Decapsulation at the Destination Host

At the destination host, each layer only decapsulates the packet received, removes the payload, and delivers the payload to the next-higher layer protocol until the message reaches the application layer. It is necessary to say that decapsulation in the host involves error checking.

2.2.5 Addressing

It is worth mentioning another concept related to protocol layering in the Internet, *addressing*. As we discussed before, we have logical communication between pairs of layers in this model. Any communication that involves two parties needs two addresses: source address and destination address. Although it looks as if we need five pairs of addresses, one pair per layer, we normally have only four because the physical layer does not need addresses; the unit of data exchange at the physical layer is a bit, which definitely cannot have an address. Figure 2.9 shows the addressing at each layer.

As the figure shows, there is a relationship between the layer, the address used in that layer, and the packet name at that layer. At the application layer, we normally use names to define the site that provides services, such as *someorg.com*, or the e-mail

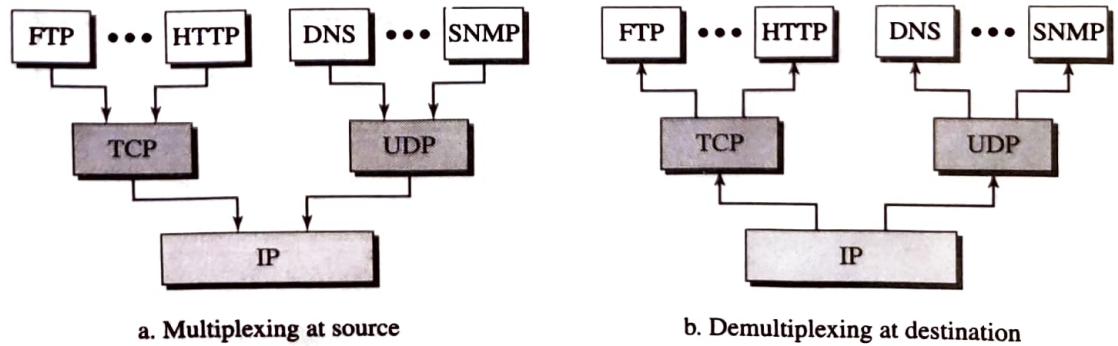
Figure 2.9 Addressing in the TCP/IP protocol suite

Packet names	Layers	Addresses
Message	Application layer	Names
Segment / User datagram	Transport layer	Port numbers
Datagram	Network layer	Logical addresses
Frame	Data-link layer	Link-layer addresses
Bits	Physical layer	

address, such as *somebody@coldmail.com*. At the transport layer, addresses are called port numbers, and these define the application-layer programs at the source and destination. Port numbers are local addresses that distinguish between several programs running at the same time. At the network-layer, the addresses are global, with the whole Internet as the scope. A network-layer address uniquely defines the connection of a device to the Internet. The link-layer addresses, sometimes called MAC addresses, are locally defined addresses, each of which defines a specific host or router in a network (LAN or WAN). We will come back to these addresses in future chapters.

2.2.6 Multiplexing and Demultiplexing

Since the TCP/IP protocol suite uses several protocols at some layers, we can say that we have multiplexing at the source and demultiplexing at the destination. Multiplexing in this case means that a protocol at a layer can encapsulate a packet from several next-higher layer protocols (one at a time); demultiplexing means that a protocol can decapsulate and deliver a packet to several next-higher layer protocols (one at a time). Figure 2.10 shows the concept of multiplexing and demultiplexing at the three upper layers.

Figure 2.10 Multiplexing and demultiplexing

To be able to multiplex and demultiplex, a protocol needs to have a field in its header to identify to which protocol the encapsulated packets belong. At the transport

layer, either UDP or TCP can accept a message from several application-layer protocols. At the network layer, IP can accept a segment from TCP or a user datagram from UDP. IP can also accept a packet from other protocols such as ICMP, IGMP, and so on. At the data-link layer, a frame may carry the payload coming from IP or other protocols such as ARP (see Chapter 9).

2.3 THE OSI MODEL

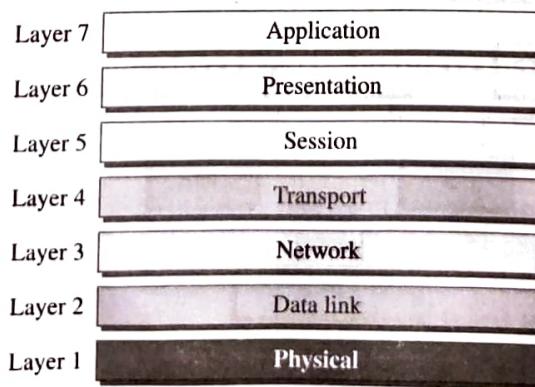
Although, when speaking of the Internet, everyone talks about the TCP/IP protocol suite, this suite is not the only suite of protocols defined. Established in 1947, the **International Organization for Standardization (ISO)** is a multinational body dedicated to worldwide agreement on international standards. Almost three-fourths of the countries in the world are represented in the ISO. An ISO standard that covers all aspects of network communications is the **Open Systems Interconnection (OSI) model**. It was first introduced in the late 1970s.

ISO is the organization; OSI is the model.

An *open system* is a set of protocols that allows any two different systems to communicate regardless of their underlying architecture. The purpose of the OSI model is to show how to facilitate communication between different systems without requiring changes to the logic of the underlying hardware and software. The OSI model is not a protocol; it is a model for understanding and designing a network architecture that is flexible, robust, and interoperable. The OSI model was intended to be the basis for the creation of the protocols in the OSI stack.

The OSI model is a layered framework for the design of network systems that allows communication between all types of computer systems. It consists of seven separate but related layers, each of which defines a part of the process of moving information across a network (see Figure 2.11).

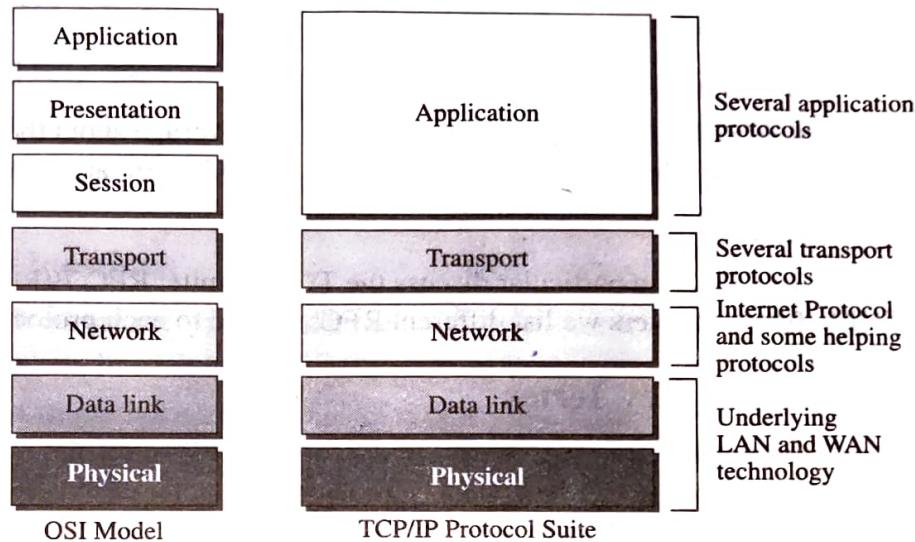
Figure 2.11 The OSI model



2.3.1 OSI versus TCP/IP

When we compare the two models, we find that two layers, session and presentation, are missing from the TCP/IP protocol suite. These two layers were not added to the TCP/IP protocol suite after the publication of the OSI model. The application layer in the suite is usually considered to be the combination of three layers in the OSI model, as shown in Figure 2.12.

Figure 2.12 TCP/IP and OSI model



Two reasons were mentioned for this decision. First, TCP/IP has more than one transport-layer protocol. Some of the functionalities of the session layer are available in some of the transport-layer protocols. Second, the application layer is not only one piece of software. Many applications can be developed at this layer. If some of the functionalities mentioned in the session and presentation layers are needed for a particular application, they can be included in the development of that piece of software.

2.3.2 Lack of OSI Model's Success

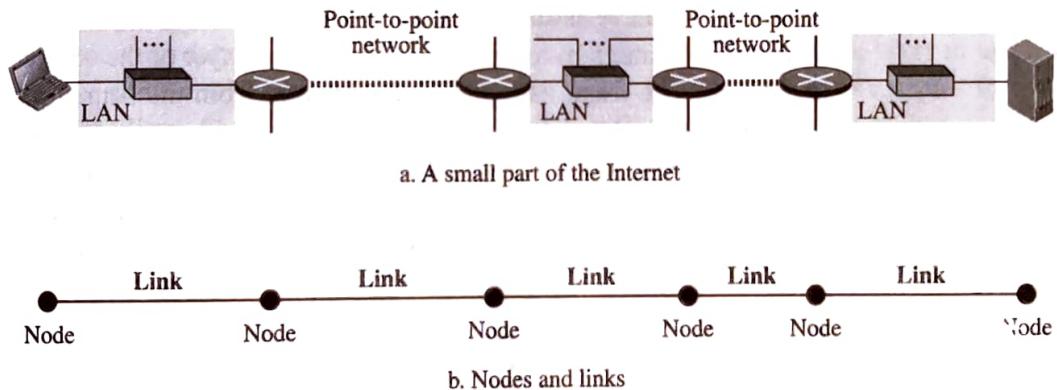
The OSI model appeared after the TCP/IP protocol suite. Most experts were at first excited and thought that the TCP/IP protocol would be fully replaced by the OSI model. This did not happen for several reasons, but we describe only three, which are agreed upon by all experts in the field. First, OSI was completed when TCP/IP was fully in place and a lot of time and money had been spent on the suite; changing it would cost a lot. Second, some layers in the OSI model were never fully defined. For example, although the services provided by the presentation and the session layers were listed in the document, actual protocols for these two layers were not fully defined, nor were they fully described, and the corresponding software was not fully

and so on. Finally, the data-link layer at router R7 communicates with the data-link layer at Bob's computer. Only one data-link layer is involved at the source or the destination, but two data-link layers are involved at each router. The reason is that Alice's and Bob's computers are each connected to a single network, but each router takes input from one network and sends output to another network. Note that although switches are also involved in the data-link-layer communication, for simplicity we have not shown them in the figure.

9.1.1 Nodes and Links

Communication at the data-link layer is node-to-node. A data unit from one point in the Internet needs to pass through many networks (LANs and WANs) to reach another point. These LANs and WANs are connected by routers. It is customary to refer to the two end hosts and the routers as **nodes** and the networks in between as **links**. Figure 9.2 is a simple representation of links and nodes when the path of the data unit is only six nodes.

Figure 9.2 *Nodes and Links*



The first node is the source host; the last node is the destination host. The other four nodes are four routers. The first, the third, and the fifth links represent the three LANs; the second and the fourth links represent the two WANs.

9.1.2 Services

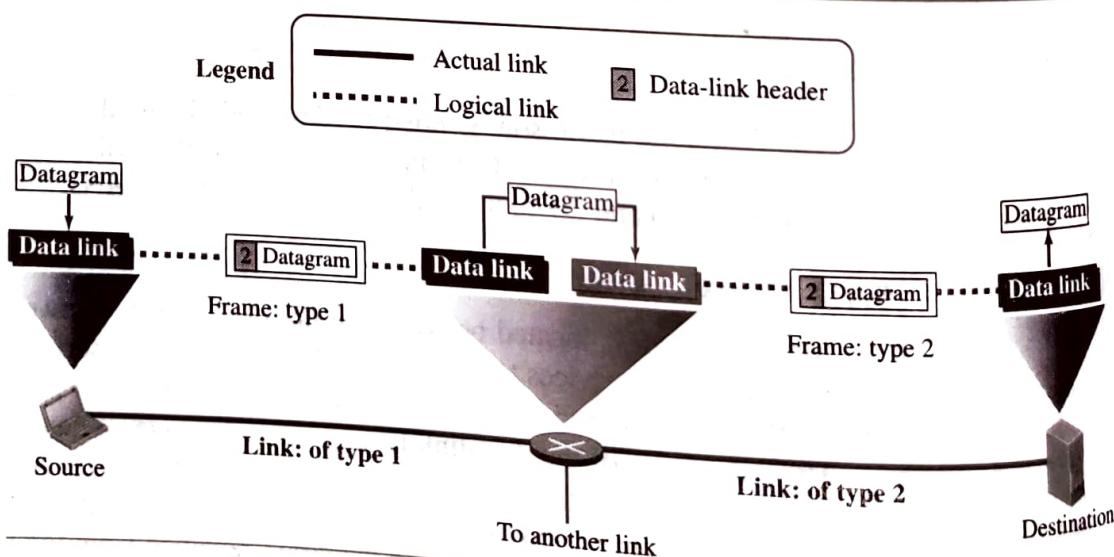
The data-link layer is located between the physical and the network layers. The data-link layer provides services to the network layer; it receives services from the physical layer. Let us discuss services provided by the data-link layer.

The duty scope of the data-link layer is node-to-node. When a packet is travelling in the Internet, the data-link layer of a node (host or router) is responsible for delivering a datagram to the next node in the path. For this purpose, the data-link layer of the sending node needs to encapsulate the datagram received from the network in a frame, and the data-link layer of the receiving node needs to decapsulate the datagram from the frame. In other words, the data-link layer of the source host needs only to

encapsulate, the data-link layer of the destination host needs to decapsulate, but each intermediate node needs to both encapsulate and decapsulate. One may ask why we need encapsulation and decapsulation at each intermediate node. The reason is that each link may be using a different protocol with a different frame format. Even if one link and the next are using the same protocol, encapsulation and decapsulation are needed because the link-layer addresses are normally different. An analogy may help in this case. Assume a person needs to travel from her home to her friend's home in another city. The traveller can use three transportation tools. She can take a taxi to go to the train station in her own city, then travel on the train from her own city to the city where her friend lives, and finally reach her friend's home using another taxi. Here we have a source node, a destination node, and two intermediate nodes. The traveller needs to get into the taxi at the source node, get out of the taxi and get into the train at the first intermediate node (train station in the city where she lives), get out of the train and get into another taxi at the second intermediate node (train station in the city where her friend lives), and finally get out of the taxi when she arrives at her destination. A kind of encapsulation occurs at the source node, encapsulation and decapsulation occur at the intermediate nodes, and decapsulation occurs at the destination node. Our traveller is the same, but she uses three transporting tools to reach the destination.

Figure 9.3 shows the encapsulation and decapsulation at the data-link layer. For simplicity, we have assumed that we have only one router between the source and destination. The datagram received by the data-link layer of the source host is encapsulated in a frame. The frame is logically transported from the source host to the router. The frame is decapsulated at the data-link layer of the router and encapsulated at another frame. The new frame is logically transported from the router to the destination host. Note that, although we have shown only two data-link layers at the router, the router actually has three data-link layers because it is connected to three physical links.

Figure 9.3 A communication with only three nodes



With the contents of the above figure in mind, we can list the services provided by a data-link layer as shown below.

Framing

Definitely, the first service provided by the data-link layer is **framing**. The data-link layer at each node needs to encapsulate the datagram (packet received from the network layer) in a **frame** before sending it to the next node. The node also needs to decapsulate the datagram from the frame received on the logical channel. Although we have shown only a header for a frame, we will see in future chapters that a frame may have both a header and a trailer. Different data-link layers have different formats for framing.

A packet at the data-link layer is normally called a *frame*.

Flow Control

Whenever we have a producer and a consumer, we need to think about flow control. If the producer produces items that cannot be consumed, accumulation of items occurs. The sending data-link layer at the end of a link is a producer of frames; the receiving data-link layer at the other end of a link is a consumer. If the rate of produced frames is higher than the rate of consumed frames, frames at the receiving end need to be buffered while waiting to be consumed (processed). Definitely, we cannot have an unlimited buffer size at the receiving side. We have two choices. The first choice is to let the receiving data-link layer drop the frames if its buffer is full. The second choice is to let the receiving data-link layer send a feedback to the sending data-link layer to ask it to stop or slow down. Different data-link-layer protocols use different strategies for flow control. Since flow control also occurs at the transport layer, with a higher degree of importance, we discuss this issue in Chapter 23 when we talk about the transport layer.

Error Control

At the sending node, a frame in a data-link layer needs to be changed to bits, transformed to electromagnetic signals, and transmitted through the transmission media. At the receiving node, electromagnetic signals are received, transformed to bits, and put together to create a frame. Since electromagnetic signals are susceptible to error, a frame is susceptible to error. The error needs first to be detected. After detection, it needs to be either corrected at the receiver node or discarded and retransmitted by the sending node. Since error detection and correction is an issue in every layer (node-to-node or host-to-host), we have dedicated all of Chapter 10 to this issue.

Congestion Control

Although a link may be congested with frames, which may result in frame loss, most data-link-layer protocols do not directly use a congestion control to alleviate congestion, although some wide-area networks do. In general, congestion control is considered an issue in the network layer or the transport layer because of its end-to-end nature. We will discuss congestion control in the network layer and the transport layer in later chapters.

9.1.3 Two Categories of Links

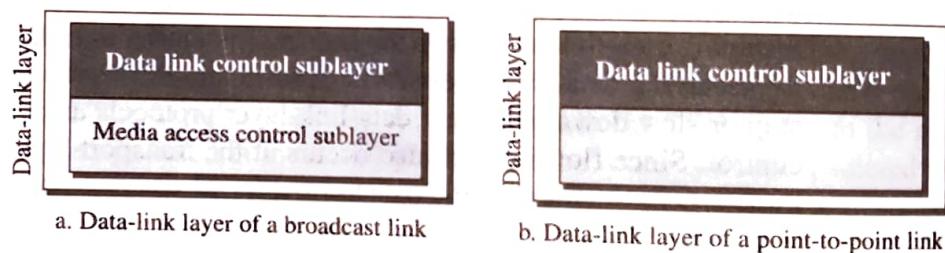
Although two nodes are physically connected by a transmission medium such as cable or air, we need to remember that the data-link layer controls how the medium is used. We can have a data-link layer that uses the whole capacity of the medium; we can also

have a data-link layer that uses only part of the capacity of the link. In other words, we can have a *point-to-point link* or a *broadcast link*. In a point-to-point link, the link is dedicated to the two devices; in a broadcast link, the link is shared between several pairs of devices. For example, when two friends use the traditional home phones to chat, they are using a point-to-point link; when the same two friends use their cellular phones, they are using a broadcast link (the air is shared among many cell phone users).

9.1.4 Two Sublayers

To better understand the functionality of and the services provided by the link layer, we can divide the data-link layer into two sublayers: **data link control (DLC)** and **media access control (MAC)**. This is not unusual because, as we will see in later chapters, LAN protocols actually use the same strategy. The data link control sublayer deals with all issues common to both point-to-point and broadcast links; the media access control sublayer deals only with issues specific to broadcast links. In other words, we separate these two types of links at the data-link layer, as shown in Figure 9.4.

Figure 9.4 Dividing the data-link layer into two sublayers



We discuss the DLC and MAC sublayers later, each in a separate chapter. In addition, we discuss the issue of error detection and correction, a duty of the data-link and other layers, also in a separate chapter.

9.2 LINK-LAYER ADDRESSING

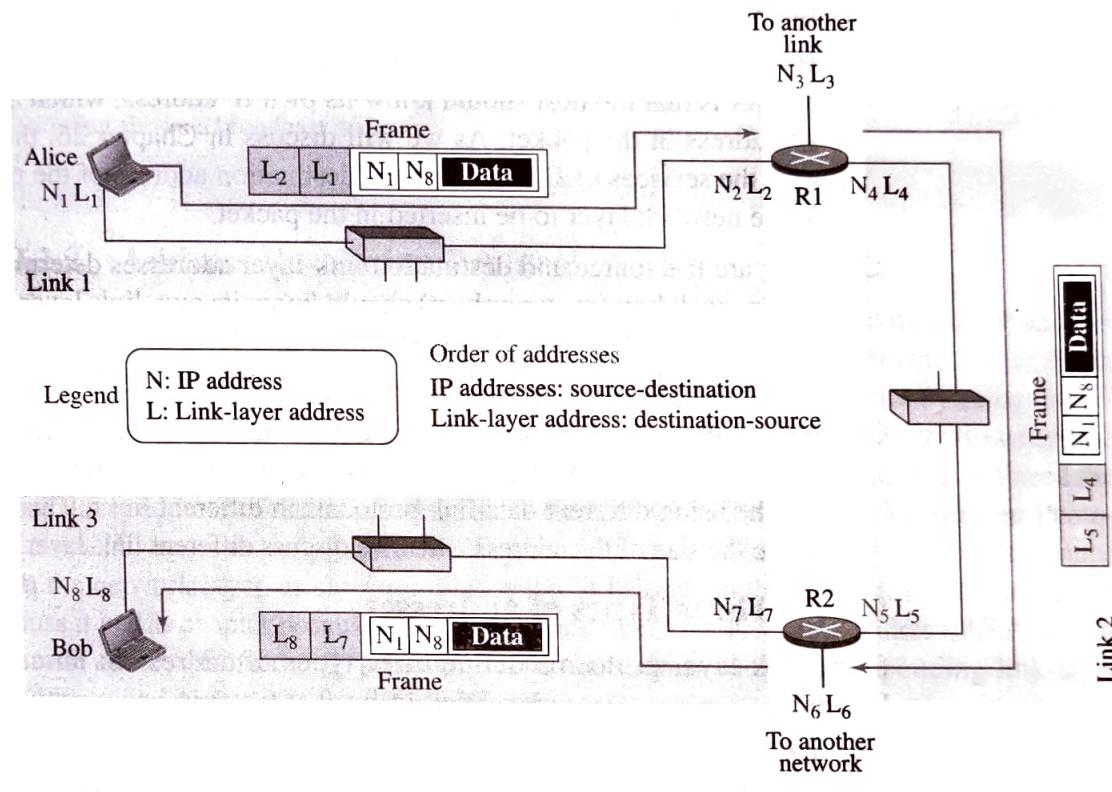
The next issue we need to discuss about the data-link layer is the link-layer addresses. In Chapter 18, we will discuss IP addresses as the identifiers at the network layer that define the exact points in the Internet where the source and destination hosts are connected. However, in a connectionless internetwork such as the Internet we cannot make a datagram reach its destination using only IP addresses. The reason is that each datagram in the Internet, from the same source host to the same destination host, may take a different path. The source and destination IP addresses define the two ends but cannot define which links the datagram should pass through.

We need to remember that the IP addresses in a datagram should not be changed. If the destination IP address in a datagram changes, the packet never reaches its destination; if the source IP address in a datagram changes, the destination host or a router can never communicate with the source if a response needs to be sent back or an error needs to be reported back to the source (see ICMP in Chapter 19).

The above discussion shows that we need another addressing mechanism in a connectionless internetwork: the link-layer addresses of the two nodes. A *link-layer address* is sometimes called a *link address*, sometimes a *physical address*, and sometimes a *MAC address*. We use these terms interchangeably in this book.

Since a link is controlled at the data-link layer, the addresses need to belong to the data-link layer. When a datagram passes from the network layer to the data-link layer, the datagram will be encapsulated in a frame and two data-link addresses are added to the frame header. These two addresses are changed every time the frame moves from one link to another. Figure 9.5 demonstrates the concept in a small internet.

Figure 9.5 IP addresses and link-layer addresses in a small internet



In the internet in Figure 9.5, we have three links and two routers. We also have shown only two hosts: Alice (source) and Bob (destination). For each host, we have shown two addresses, the IP addresses (N) and the link-layer addresses (L). Note that a router has as many pairs of addresses as the number of links the router is connected to. We have shown three frames, one in each link. Each frame carries the same datagram with the same source and destination addresses (N₁ and N₈), but the link-layer addresses of the frame change from link to link. In link 1, the link-layer addresses are L₁ and L₂. In link 2, they are L₄ and L₅. In link 3, they are L₇ and L₈. Note that the IP addresses and the link-layer addresses are not in the same order. For IP addresses, the source address comes before the destination address; for link-layer addresses, the destination address comes before the source. The datagrams and

frames are designed in this way, and we follow the design. We may raise several questions:

- ❑ If the IP address of a router does not appear in any datagram sent from a source to a destination, why do we need to assign IP addresses to routers? The answer is that in some protocols a router may act as a sender or receiver of a datagram. For example, in routing protocols we will discuss in Chapters 20 and 21, a router is a sender or a receiver of a message. The communications in these protocols are between routers.
- ❑ Why do we need more than one IP address in a router, one for each interface? The answer is that an interface is a connection of a router to a link. We will see that an IP address defines a point in the Internet at which a device is connected. A router with n interfaces is connected to the Internet at n points. This is the situation of a house at the corner of a street with two gates; each gate has the address related to the corresponding street.
- ❑ How are the source and destination IP addresses in a packet determined? The answer is that the host should know its own IP address, which becomes the source IP address in the packet. As we will discuss in Chapter 26, the application layer uses the services of DNS to find the destination address of the packet and passes it to the network layer to be inserted in the packet.
- ❑ How are the source and destination link-layer addresses determined for each link? Again, each hop (router or host) should know its own link-layer address, as we discuss later in the chapter. The destination link-layer address is determined by using the Address Resolution Protocol, which we discuss shortly.
- ❑ What is the size of link-layer addresses? The answer is that it depends on the protocol used by the link. Although we have only one IP protocol for the whole Internet, we may be using different data-link protocols in different links. This means that we can define the size of the address when we discuss different link-layer protocols.

9.2.1 Three Types of addresses

Some link-layer protocols define three types of addresses: unicast, multicast, and broadcast.

Unicast Address

Each host or each interface of a router is assigned a unicast address. Unicasting means one-to-one communication. A frame with a unicast address destination is destined only for one entity in the link.

Example 9.1

As we will see in Chapter 13, the unicast link-layer addresses in the most common LAN, Ethernet, are 48 bits (six bytes) that are presented as 12 hexadecimal digits separated by colons; for example, the following is a link-layer address of a computer.

A3:34:45:11:92:F1

Multicast Address

Some link-layer protocols define multicast addresses. Multicasting means one-to-many communication. However, the jurisdiction is local (inside the link).

Example 9.2

As we will see in Chapter 13, the multicast link-layer addresses in the most common LAN, Ethernet, are 48 bits (six bytes) that are presented as 12 hexadecimal digits separated by colons. The second digit, however, needs to be an even number in hexadecimal. The following shows a multicast address:

A2:34:45:11:92:F1

Broadcast Address

Some link-layer protocols define a broadcast address. Broadcasting means one-to-all communication. A frame with a destination broadcast address is sent to all entities in the link.

Example 9.3

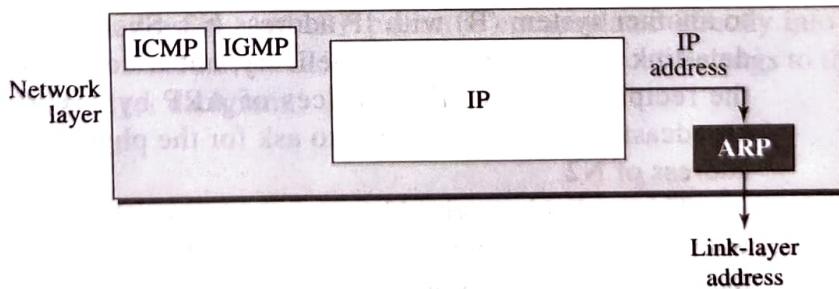
As we will see in Chapter 13, the broadcast link-layer addresses in the most common LAN, Ethernet, are 48 bits, all 1s, that are presented as 12 hexadecimal digits separated by colons. The following shows a broadcast address:

FF:FF:FF:FF:FF:FF

9.2.2 Address Resolution Protocol (ARP)

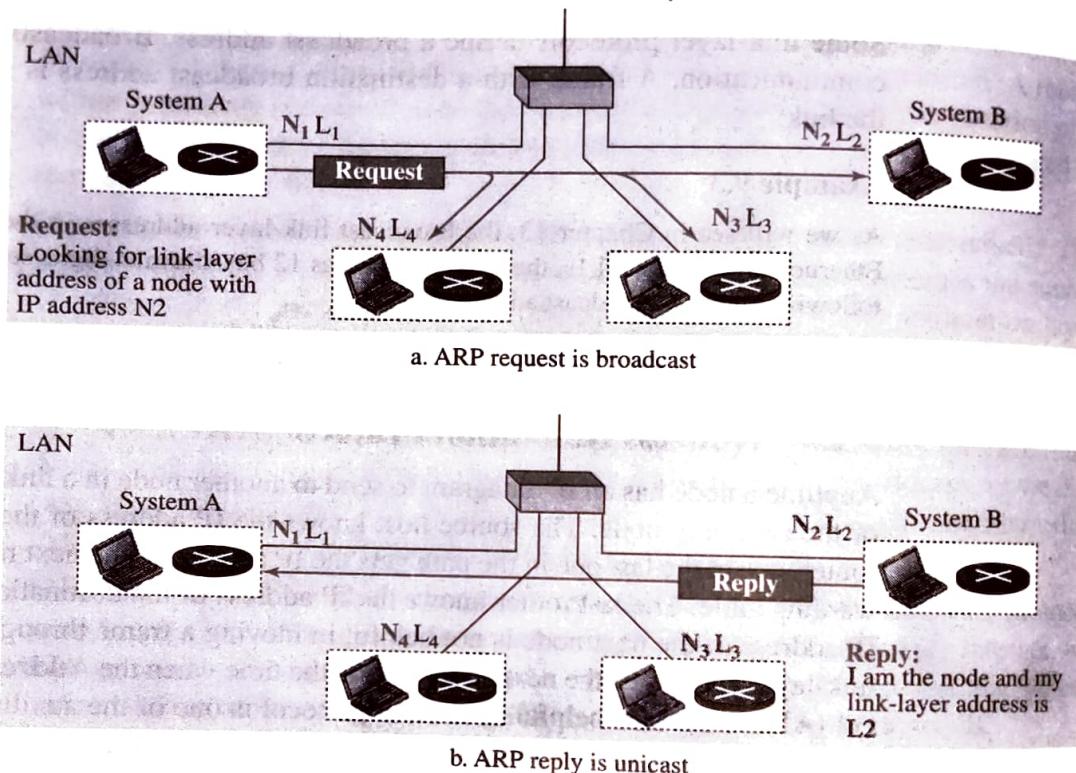
Anytime a node has an IP datagram to send to another node in a link, it has the IP address of the receiving node. The source host knows the IP address of the default router. Each router except the last one in the path gets the IP address of the next router by using its forwarding table. The last router knows the IP address of the destination host. However, the IP address of the next node is not helpful in moving a frame through a link; we need the link-layer address of the next node. This is the time when the **Address Resolution Protocol (ARP)** becomes helpful. The ARP protocol is one of the auxiliary protocols defined in the network layer, as shown in Figure 9.6. It belongs to the network layer, but we discuss it in this chapter because it maps an IP address to a logical-link address. ARP accepts an IP address from the IP protocol, maps the address to the corresponding link-layer address, and passes it to the data-link layer.

Figure 9.6 Position of ARP in TCP/IP protocol suite



Anytime a host or a router needs to find the link-layer address of another host or router in its network, it sends an ARP request packet. The packet includes the link-layer and IP addresses of the sender and the IP address of the receiver. Because the sender does not know the link-layer address of the receiver, the query is broadcast over the link using the link-layer broadcast address, which we discuss for each protocol later (see Figure 9.7).

Figure 9.7 ARP operation



Every host or router on the network receives and processes the ARP request packet, but only the intended recipient recognizes its IP address and sends back an ARP response packet. The response packet contains the recipient's IP and link-layer addresses. The packet is unicast directly to the node that sent the request packet.

In Figure 9.7a, the system on the left (A) has a packet that needs to be delivered to another system (B) with IP address N₂. System A needs to pass the packet to its data-link layer for the actual delivery, but it does not know the physical address of the recipient. It uses the services of ARP by asking the ARP protocol to send a broadcast ARP request packet to ask for the physical address of a system with an IP address of N₂.

This packet is received by every system on the physical network, but only system B will answer it, as shown in Figure 9.7b. System B sends an ARP reply packet that includes its physical address. Now system A can send all the packets it has for this destination using the physical address it received.

Caching

A question that is often asked is this: If system A can broadcast a frame to find the link-layer address of system B, why can't system A send the datagram for system B using a broadcast frame? In other words, instead of sending one broadcast frame (ARP request), one unicast frame (ARP response), and another unicast frame (for sending the datagram), system A can encapsulate the datagram and send it to the network. System B receives it and keep it; other systems discard it.

To answer the question, we need to think about the efficiency. It is probable that system A has more than one datagram to send to system B in a short period of time. For example, if system B is supposed to receive a long e-mail or a long file, the data do not fit in one datagram.

Let us assume that there are 20 systems connected to the network (link): system A, system B, and 18 other systems. We also assume that system A has 10 datagrams to send to system B in one second.

- a. Without using ARP, system A needs to send 10 broadcast frames. Each of the 18 other systems need to receive the frames, decapsulate the frames, remove the datagram and pass it to their network-layer to find out the datagrams do not belong to them. This means processing and discarding 180 broadcast frames.
- b. Using ARP, system A needs to send only one broadcast frame. Each of the 18 other systems need to receive the frames, decapsulate the frames, remove the ARP message and pass the message to their ARP protocol to find that the frame must be discarded. This means processing and discarding only 18 (instead of 180) broadcast frames. After system B responds with its own data-link address, system A can store the link-layer address in its cache memory. The rest of the nine frames are only unicast. Since processing broadcast frames is expensive (time consuming), the first method is preferable.

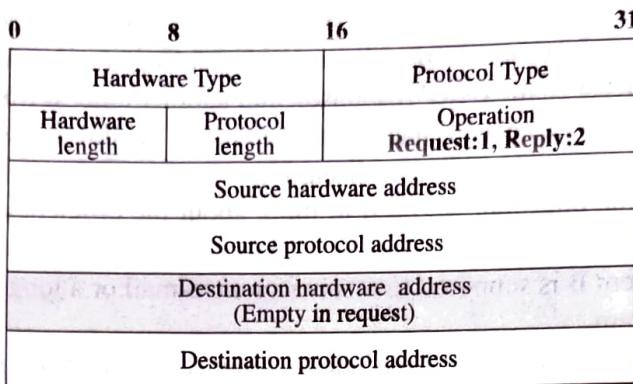
Packet Format

Figure 9.8 shows the format of an ARP packet. The names of the fields are self-explanatory. The *hardware type* field defines the type of the link-layer protocol; Ethernet is given the type 1. The *protocol type* field defines the network-layer protocol: IPv4 protocol is $(0800)_{16}$. The source hardware and source protocol addresses are variable-length fields defining the link-layer and network-layer addresses of the sender. The destination hardware address and destination protocol address fields define the receiver link-layer and network-layer addresses. An ARP packet is encapsulated directly into a data-link frame. The frame needs to have a field to show that the payload belongs to the ARP and not to the network-layer datagram.

Example 9.4

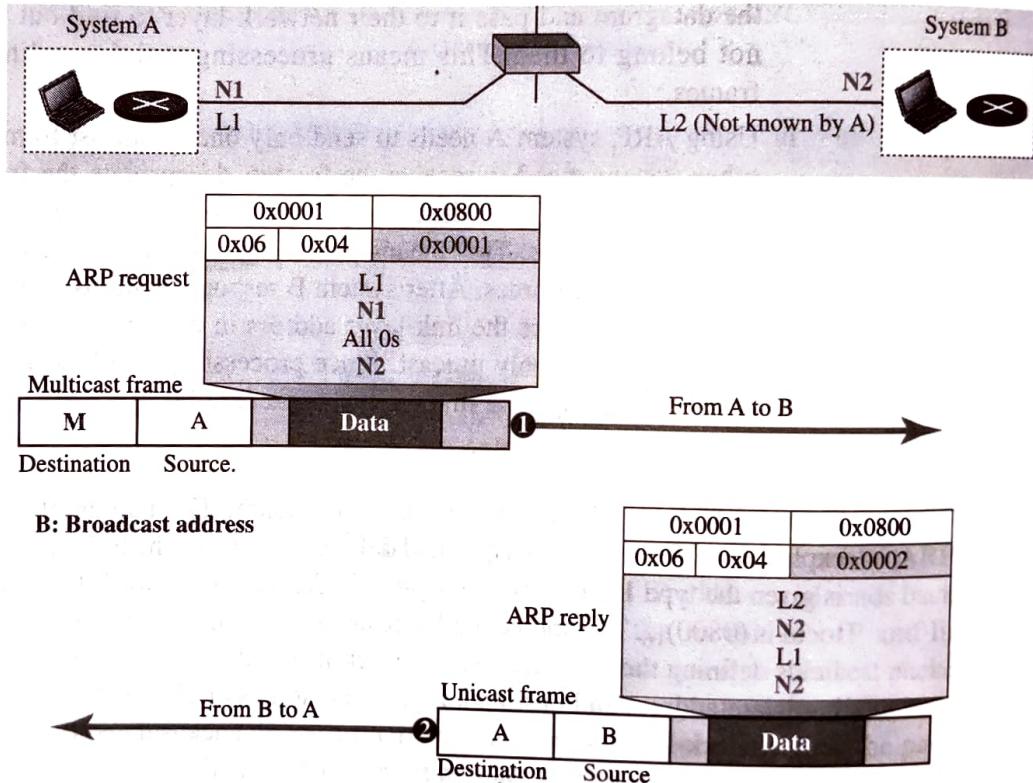
A host with IP address **N1** and MAC address **L1** has a packet to send to another host with IP address **N2** and physical address **L2** (which is unknown to the first host). The two hosts are on the same network. Figure 9.9 shows the ARP request and response messages.

Figure 9.8 ARP packet



Hardware: LAN or WAN protocol
Protocol: Network-layer protocol

Figure 9.9 Example 9.4

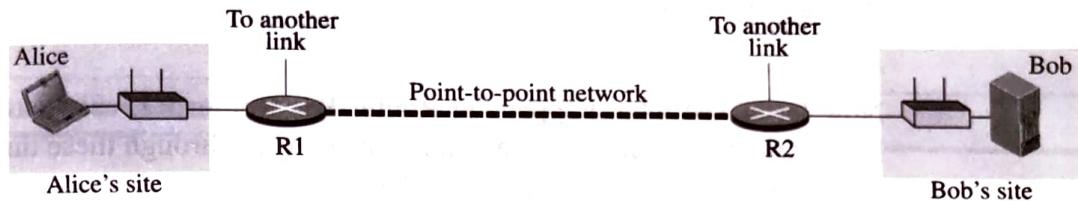


9.2.3 An Example of Communication

To show how communication is done at the data-link layer and how link-layer addresses are found, let us go through a simple example. Assume Alice needs to send a datagram to Bob, who is three nodes away in the Internet. How Alice finds the network-layer address of Bob is what we discover in Chapter 26 when we discuss DNS. For the moment, assume that Alice knows the network-layer (IP) address of Bob. In other words, Alice's host is given the data to be sent, the IP address of Bob, and the

IP address of Alice's host (each host needs to know its IP address). Figure 9.10 shows the part of the internet for our example.

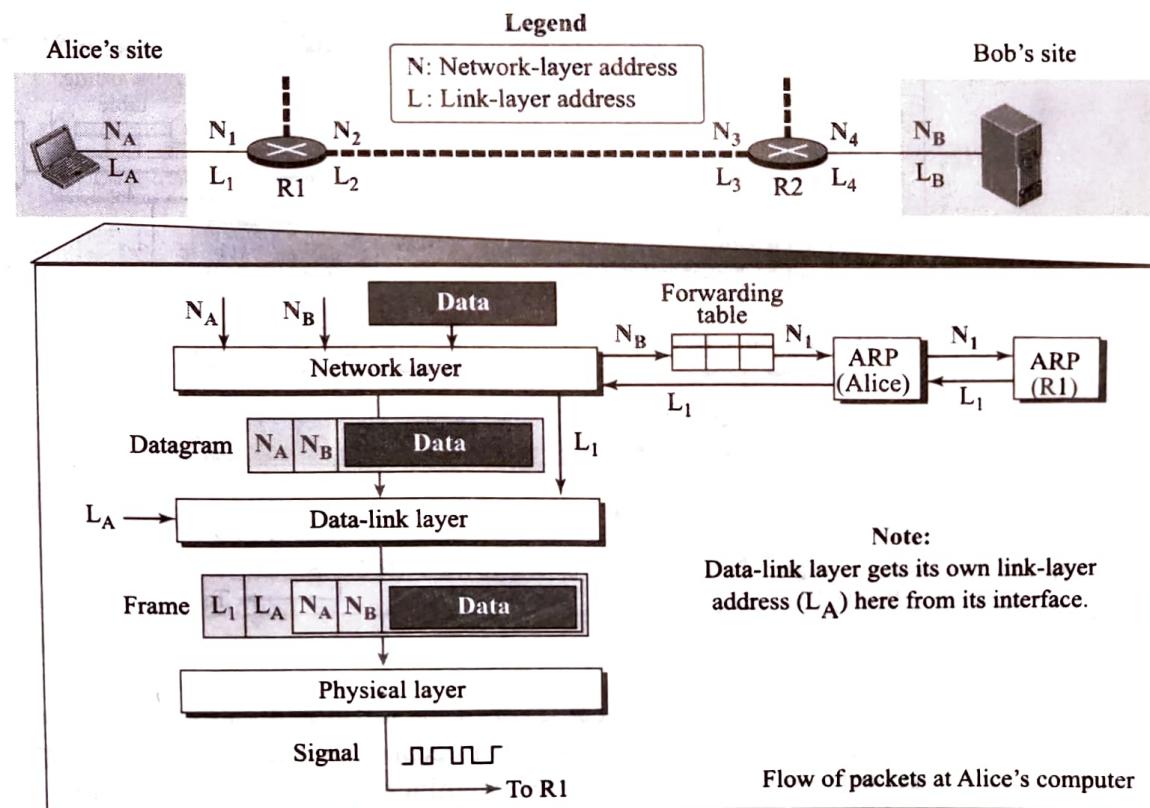
Figure 9.10 The internet for our example



Activities at Alice's Site

We will use symbolic addresses to make the figures more readable. Figure 9.11 shows what happens at Alice's site.

Figure 9.11 Flow of packets at Alice's computer



The network layer knows it's given N_A , N_B , and the packet, but it needs to find the link-layer address of the next node. The network layer consults its routing table and tries to find which router is next (the default router in this case) for the destination N_B . As we will discuss in Chapter 18, the routing table gives N_1 , but the network layer

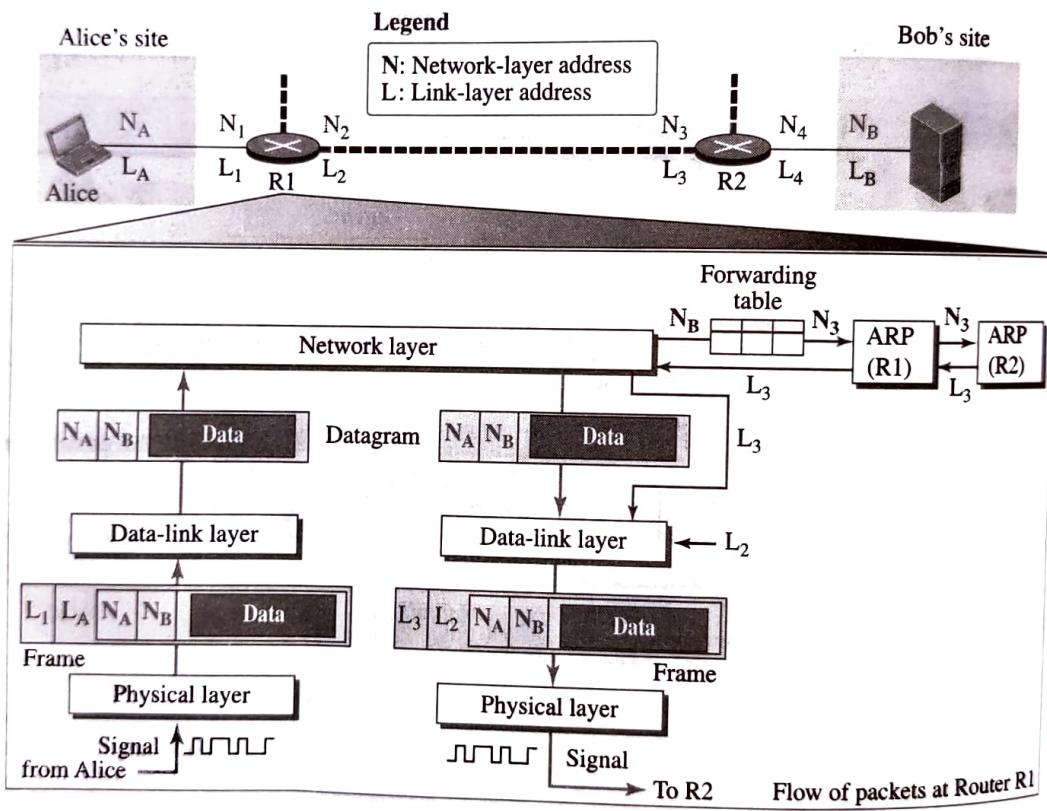
needs to find the link-layer address of router R1. It uses its ARP to find the link-layer address L_1 . The network layer can now pass the datagram with the link-layer address to the data-link layer.

The data-link layer knows its own link-layer address, L_A . It creates the frame and passes it to the physical layer, where the address is converted to signals and sent through the media.

Activities at Router R1

Now let us see what happens at Router R1. Router R1, as we know, has only three lower layers. The packet received needs to go up through these three layers and come down. Figure 9.12 shows the activities.

Figure 9.12 Flow of activities at router R1



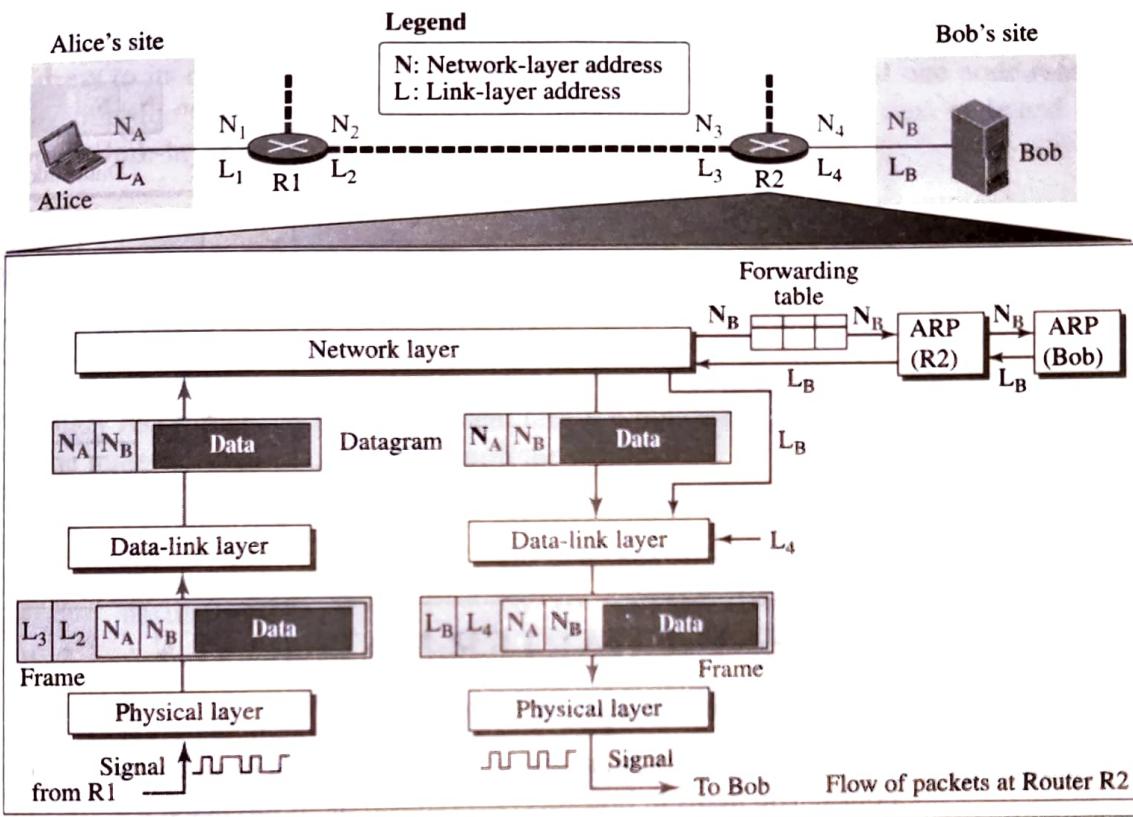
At arrival, the physical layer of the left link creates the frame and passes it to the data-link layer. The data-link layer decapsulates the datagram and passes it to the network layer. The network layer examines the network-layer address of the datagram and finds that the datagram needs to be delivered to the device with IP address N_B . The network layer consults its routing table to find out which is the next node (router) in the path to N_B . The forwarding table returns N_3 . The IP address of router R2 is in the same link with R1. The network layer now uses the ARP to find the link-layer address of this router, which comes up as L_3 . The network layer passes the datagram and L_3 to the data-link layer belonging to the link at the right side. The link layer

encapsulates the datagram, adds **L3** and **L2** (its own link-layer address), and passes the frame to the physical layer. The physical layer encodes the bits to signals and sends them through the medium to R2.

Activities at Router R2

Activities at router R2 are almost the same as in R1, as shown in Figure 9.13.

Figure 9.13 Activities at router R2.

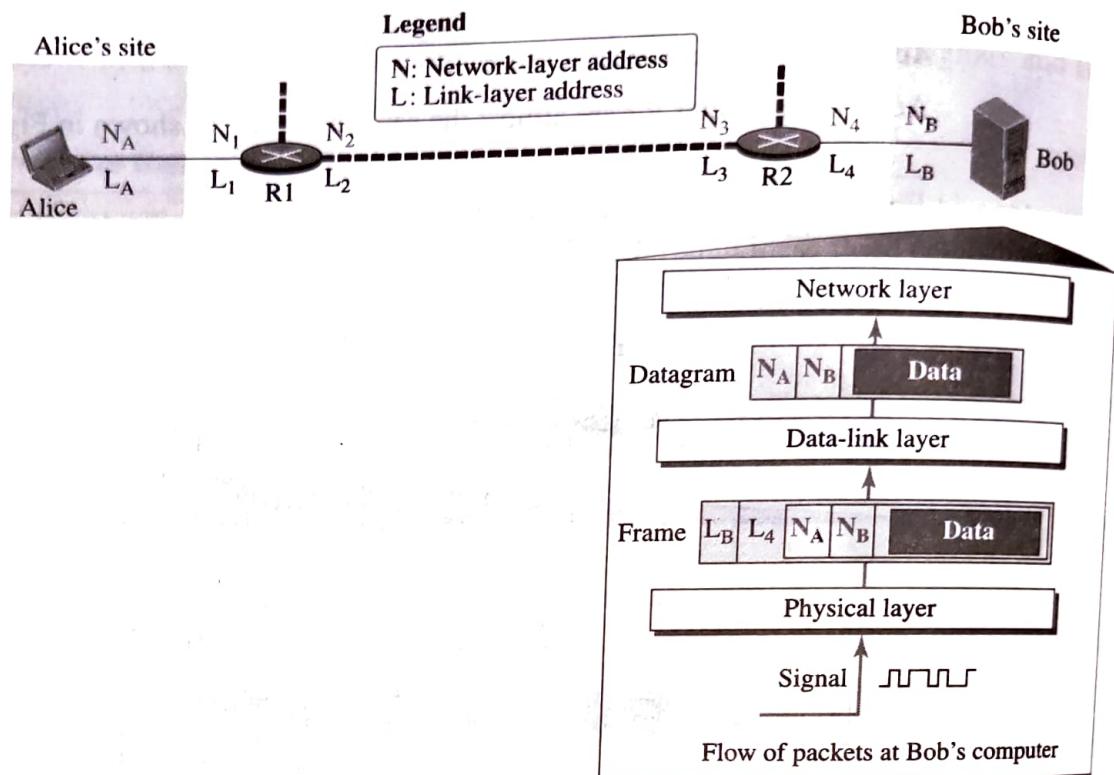


Activities at Bob's Site

Now let us see what happens at Bob's site. Figure 9.14 shows how the signals at Bob's site are changed to a message. At Bob's site there are no more addresses or mapping needed. The signal received from the link is changed to a frame. The frame is passed to the data-link layer, which decapsulates the datagram and passes it to the network layer. The network layer decapsulates the message and passes it to the transport layer.

Changes in Addresses

This example shows that the source and destination network-layer addresses, N_A and N_B , have not been changed during the whole journey. However, all four network-layer addresses of routers R1 and R2 (N_1 , N_2 , N_3 , and N_4) are needed to transfer a datagram from Alice's computer to Bob's computer.

Figure 9.14 Activities at Bob's site

9.3 END-CHAPTER MATERIALS

9.3.1 Recommended Reading

For more details about subjects discussed in this chapter, we recommend the following books. The items in brackets [...] refer to the reference list at the end of the text.

Books

Several books discuss link-layer issues. Among them we recommend [Ham 80], [Zar 02], [Ror 96], [Tan 03], [GW 04], [For 03], [KMK 04], [Sta 04], [Kes 02], [PD 03], [Kei 02], [Spu 00], [KCK 98], [Sau 98], [Izz 00], [Per 00], and [WV 00].

9.3.2 Key Terms

Address Resolution Protocol (ARP)
data link control (DLC)
frame
framing

links
media access control (MAC)
nodes

9.3.3 Summary

The Internet is made of many hosts, networks, and connecting devices such as routers. The hosts and connecting devices are referred to as *nodes*; the networks are referred to

11.1 DLC SERVICES

The **data link control (DLC)** deals with procedures for communication between two adjacent nodes—node-to-node communication—no matter whether the link is dedicated or broadcast. Data link control functions include *framing* and *flow and error control*. In this section, we first discuss framing, or how to organize the bits that are carried by the physical layer. We then discuss flow and error control.

11.1.1 Framing

Data transmission in the physical layer means moving bits in the form of a signal from the source to the destination. The physical layer provides bit synchronization to ensure that the sender and receiver use the same bit durations and timing. We discussed the physical layer in Part II of the book.

The data-link layer, on the other hand, needs to pack bits into frames, so that each frame is distinguishable from another. Our postal system practices a type of framing. The simple act of inserting a letter into an envelope separates one piece of information from another; the envelope serves as the delimiter. In addition, each envelope defines the sender and receiver addresses, which is necessary since the postal system is a many-to-many carrier facility.

Framing in the data-link layer separates a message from one source to a destination by adding a sender address and a destination address. The destination address defines where the packet is to go; the sender address helps the recipient acknowledge the receipt.

Although the whole message could be packed in one frame, that is not normally done. One reason is that a frame can be very large, making flow and error control very inefficient. When a message is carried in one very large frame, even a single-bit error would require the retransmission of the whole frame. When a message is divided into smaller frames, a single-bit error affects only that small frame.

Frame Size

Frames can be of fixed or variable size. In *fixed-size framing*, there is no need for defining the boundaries of the frames; the size itself can be used as a delimiter. An example of this type of framing is the ATM WAN, which uses frames of fixed size called *cells*. We discuss ATM in Chapter 14.

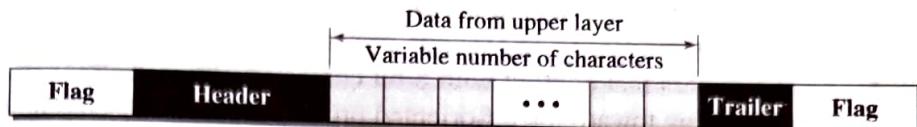
Our main discussion in this chapter concerns *variable-size framing*, prevalent in local-area networks. In variable-size framing, we need a way to define the end of one frame and the beginning of the next. Historically, two approaches were used for this purpose: a character-oriented approach and a bit-oriented approach.

Character-Oriented Framing

In *character-oriented (or byte-oriented) framing*, data to be carried are 8-bit characters from a coding system such as ASCII (see Appendix A). The header, which normally carries the source and destination addresses and other control information, and the trailer, which carries error detection redundant bits, are also multiples of 8 bits. To separate one frame from the next, an 8-bit (1-byte) **flag** is added at the beginning and the end of a frame. The flag, composed of protocol-dependent special characters, signals the

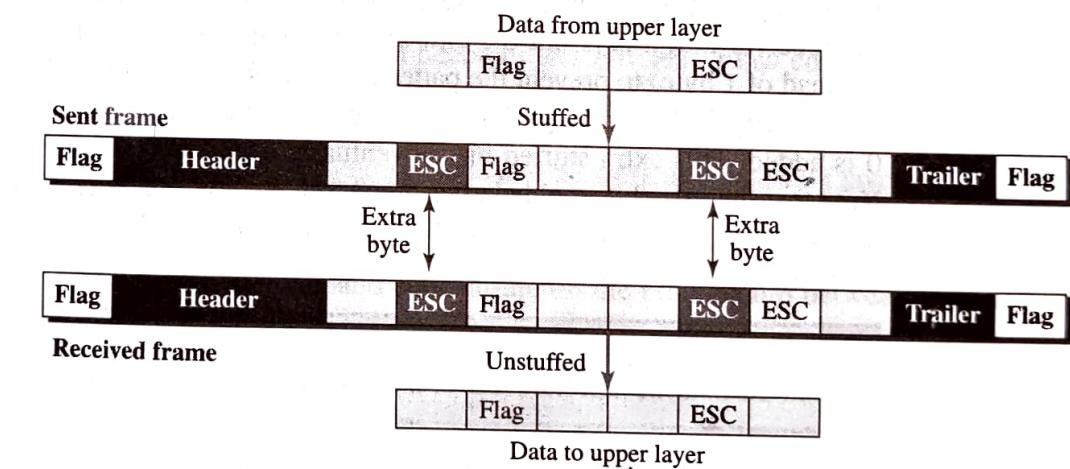
start or end of a frame. Figure 11.1 shows the format of a frame in a character-oriented protocol.

Figure 11.1 A frame in a character-oriented protocol



Character-oriented framing was popular when only text was exchanged by the data-link layers. The flag could be selected to be any character not used for text communication. Now, however, we send other types of information such as graphs, audio, and video; any character used for the flag could also be part of the information. If this happens, the receiver, when it encounters this pattern in the middle of the data, thinks it has reached the end of the frame. To fix this problem, a byte-stuffing strategy was added to character-oriented framing. In **byte stuffing** (or character stuffing), a special byte is added to the data section of the frame when there is a character with the same pattern as the flag. The data section is stuffed with an extra byte. This byte is usually called the *escape character* (*ESC*) and has a predefined bit pattern. Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not as a delimiting flag. Figure 11.2 shows the situation.

Figure 11.2 Byte stuffing and unstuffing



Byte stuffing is the process of adding one extra byte whenever there is a flag or escape character in the text.

Byte stuffing by the escape character allows the presence of the flag in the data section of the frame, but it creates another problem. What happens if the text contains one or more escape characters followed by a byte with the same pattern as the flag? The

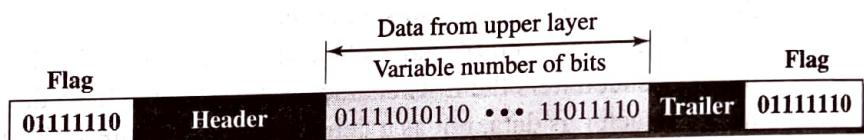
receiver removes the escape character, but keeps the next byte, which is incorrectly interpreted as the end of the frame. To solve this problem, the escape characters that are part of the text must also be marked by another escape character. In other words, if the escape character is part of the text, an extra one is added to show that the second one is part of the text.

Character-oriented protocols present another problem in data communications. The universal coding systems in use today, such as Unicode, have 16-bit and 32-bit characters that conflict with 8-bit characters. We can say that, in general, the tendency is moving toward the bit-oriented protocols that we discuss next.

Bit-Oriented Framing

In *bit-oriented framing*, the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, and so on. However, in addition to headers (and possible trailers), we still need a delimiter to separate one frame from the other. Most protocols use a special 8-bit pattern flag, 01111110, as the delimiter to define the beginning and the end of the frame, as shown in Figure 11.3.

Figure 11.3 A frame in a bit-oriented protocol



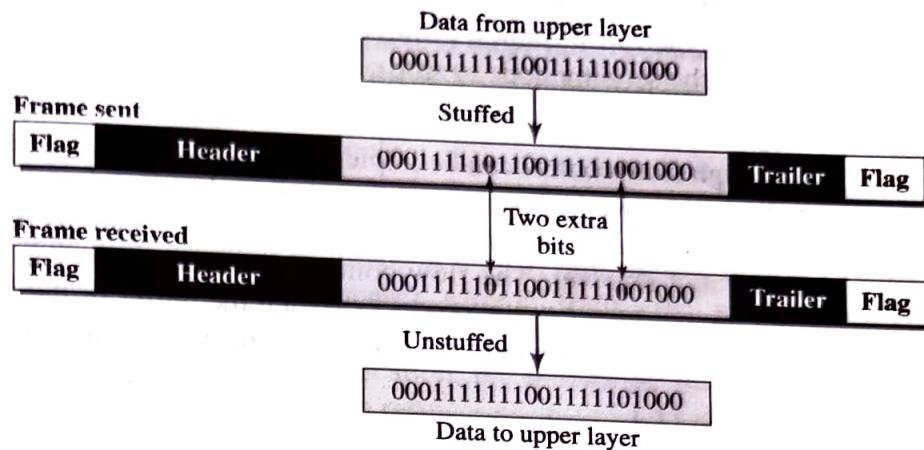
This flag can create the same type of problem we saw in the character-oriented protocols. That is, if the flag pattern appears in the data, we need to somehow inform the receiver that this is not the end of the frame. We do this by stuffing 1 single bit (instead of 1 byte) to prevent the pattern from looking like a flag. The strategy is called **bit stuffing**. In bit stuffing, if a 0 and five consecutive 1 bits are encountered, an extra 0 is added. This extra stuffed bit is eventually removed from the data by the receiver. Note that the extra bit is added after one 0 followed by five 1s regardless of the value of the next bit. This guarantees that the flag field sequence does not inadvertently appear in the frame.

Bit stuffing is the process of adding one extra 0 whenever five consecutive 1s follow a 0 in the data, so that the receiver does not mistake the pattern 01111110 for a flag.

Figure 11.4 shows bit stuffing at the sender and bit removal at the receiver. Note that even if we have a 0 after five 1s, we still stuff a 0. The 0 will be removed by the receiver.

This means that if the flaglike pattern 01111110 appears in the data, it will change to 011111010 (stuffed) and is not mistaken for a flag by the receiver. The real flag 01111110 is not stuffed by the sender and is recognized by the receiver.

Figure 11.4 Bit stuffing and unstuffing



11.1.2 Flow and Error Control

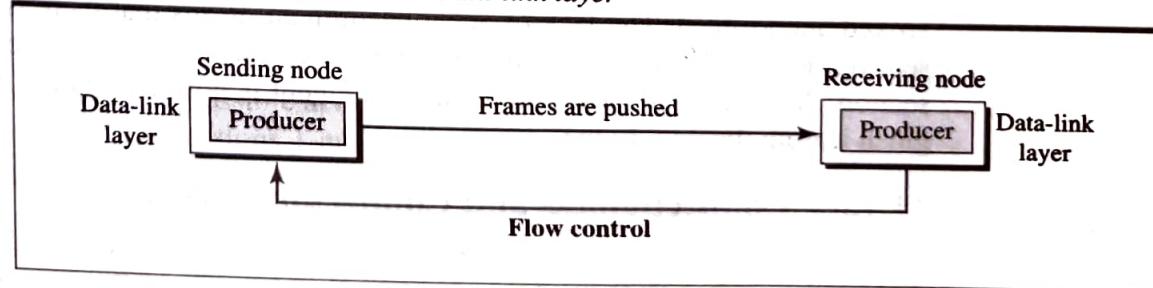
We briefly defined flow and error control in Chapter 9; we elaborate on these two issues here. One of the responsibilities of the data-link control sublayer is flow and error control at the data-link layer.

Flow Control

Whenever an entity produces items and another entity consumes them, there should be a balance between production and consumption rates. If the items are produced faster than they can be consumed, the consumer can be overwhelmed and may need to discard some items. If the items are produced more slowly than they can be consumed, the consumer must wait, and the system becomes less efficient. Flow control is related to the first issue. We need to prevent losing the data items at the consumer site.

In communication at the data-link layer, we are dealing with four entities: network and data-link layers at the sending node and network and data-link layers at the receiving node. Although we can have a complex relationship with more than one producer and consumer (as we will see in Chapter 23), we ignore the relationships between networks and data-link layers and concentrate on the relationship between two data-link layers, as shown in Figure 11.5.

Figure 11.5 Flow control at the data-link layer



The figure shows that the data-link layer at the sending node tries to push frames toward the data-link layer at the receiving node. If the receiving node cannot process and deliver the packet to its network at the same rate that the frames arrive, it becomes overwhelmed with frames. Flow control in this case can be feedback from the receiving node to the sending node to stop or slow down pushing frames.

Buffers

Although flow control can be implemented in several ways, one of the solutions is normally to use two *buffers*; one at the sending data-link layer and the other at the receiving data-link layer. A buffer is a set of memory locations that can hold packets at the sender and receiver. The flow control communication can occur by sending signals from the consumer to the producer. When the buffer of the receiving data-link layer is full, it informs the sending data-link layer to stop pushing frames.

Example 11.1

The above discussion requires that the consumers communicate with the producers on two occasions: when the buffer is full and when there are vacancies. If the two parties use a buffer with only one slot, the communication can be easier. Assume that each data-link layer uses one single memory slot to hold a frame. When this single slot in the receiving data-link layer is empty, it sends a note to the network layer to send the next frame.

Error Control

Since the underlying technology at the physical layer is not fully reliable, we need to implement error control at the data-link layer to prevent the receiving node from delivering corrupted packets to its network layer. Error control at the data-link layer is normally very simple and implemented using one of the following two methods. In both methods, a CRC is added to the frame header by the sender and checked by the receiver.

- In the first method, if the frame is corrupted, it is silently discarded; if it is not corrupted, the packet is delivered to the network layer. This method is used mostly in wired LANs such as Ethernet.
- In the second method, if the frame is corrupted, it is silently discarded; if it is not corrupted, an acknowledgment is sent (for the purpose of both flow and error control) to the sender.

Combination of Flow and Error Control

Flow and error control can be combined. In a simple situation, the acknowledgment that is sent for flow control can also be used for error control to tell the sender the packet has arrived uncorrupted. The lack of acknowledgment means that there is a problem in the sent frame. We show this situation when we discuss some simple protocols in the next section. A frame that carries an acknowledgment is normally called an *ACK* to distinguish it from the data frame.

11.1.3 Connectionless and Connection-Oriented

A DLC protocol can be either connectionless or connection-oriented. We will discuss this issue very briefly here, but we return to this topic in the network and transport layer.

Connectionless Protocol

In a connectionless protocol, frames are sent from one node to the next without any relationship between the frames; each frame is independent. Note that the term *connectionless* here does not mean that there is no physical connection (transmission medium) between the nodes; it means that there is no *connection* between frames. The frames are not numbered and there is no sense of ordering. Most of the data-link protocols for LANs are connectionless protocols.

Connection-Oriented Protocol

In a connection-oriented protocol, a logical connection should first be established between the two nodes (setup phase). After all frames that are somehow related to each other are transmitted (transfer phase), the logical connection is terminated (teardown phase). In this type of communication, the frames are numbered and sent in order. If they are not received in order, the receiver needs to wait until all frames belonging to the same set are received and then deliver them in order to the network layer. Connection-oriented protocols are rare in wired LANs, but we can see them in some point-to-point protocols, some wireless LANs, and some WANs.

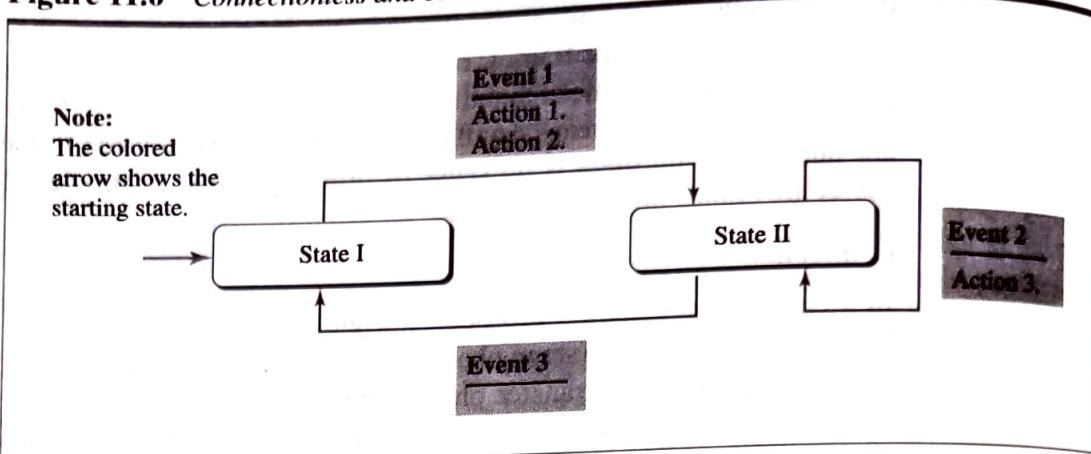
11.2 DATA-LINK LAYER PROTOCOLS

Traditionally four protocols have been defined for the data-link layer to deal with flow and error control: Simple, Stop-and-Wait, Go-Back-N, and Selective-Repeat. Although the first two protocols still are used at the data-link layer, the last two have disappeared. We therefore briefly discuss the first two protocols in this chapter, in which we need to understand some wired and wireless LANs. We postpone the discussion of all four, in full detail, to Chapter 23, where we discuss the transport layer.

The behavior of a data-link-layer protocol can be better shown as a **finite state machine (FSM)**. An FSM is thought of as a machine with a finite number of states. The machine is always in one of the states until an *event* occurs. Each event is associated with two reactions: defining the list (possibly empty) of actions to be performed and determining the next state (which can be the same as the current state). One of the states must be defined as the initial state, the state in which the machine starts when it turns on. In Figure 11.6, we show an example of a machine using FSM. We have used rounded-corner rectangles to show states, colored text to show events, and regular black text to show actions. A horizontal line is used to separate the event from the actions, although later we replace the horizontal line with a slash. The arrow shows the movement to the next state.

The figure shows a machine with three states. There are only three possible events and three possible actions. The machine starts in state I. If event 1 occurs, the machine performs actions 1 and 2 and moves to state II. When the machine is in state II, two events may occur. If event 1 occurs, the machine performs action 3 and remains in the same state, state II. If event 3 occurs, the machine performs no action, but move to state I.

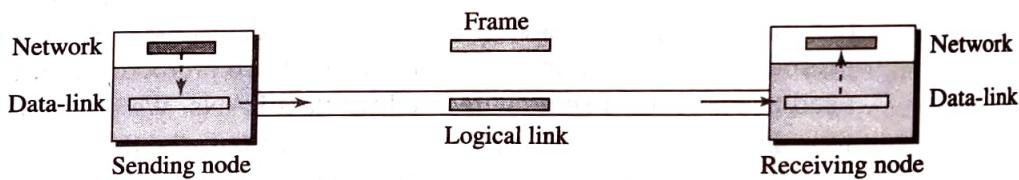
Figure 11.6 Connectionless and connection-oriented service represented as FSMs



11.2.1 Simple Protocol

Our first protocol is a **simple protocol** with neither flow nor error control. We assume that the receiver can immediately handle any frame it receives. In other words, the receiver can never be overwhelmed with incoming frames. Figure 11.7 shows the layout for this protocol.

Figure 11.7 Simple protocol



The data-link layer at the sender gets a packet from its network layer, makes a frame out of it, and sends the frame. The data-link layer at the receiver receives a frame from the link, extracts the packet from the frame, and delivers the packet to its network layer. The data-link layers of the sender and receiver provide transmission services for their network layers.

FSMs

The sender site should not send a frame until its network layer has a message to send. The receiver site cannot deliver a message to its network layer until a frame arrives. We can show these requirements using two FSMs. Each FSM has only one state, the *ready state*. The sending machine remains in the ready state until a request comes from the process in the network layer. When this event occurs, the sending machine encapsulates the message in a frame and sends it to the receiving machine. The receiving machine remains in the ready state until a frame arrives from the sending machine. When this event occurs, the receiving machine decapsulates the message out of the frame and delivers it to the process at the network layer. Figure 11.8 shows the FSMs for the simple protocol. We'll see more in Chapter 23, which uses this protocol.

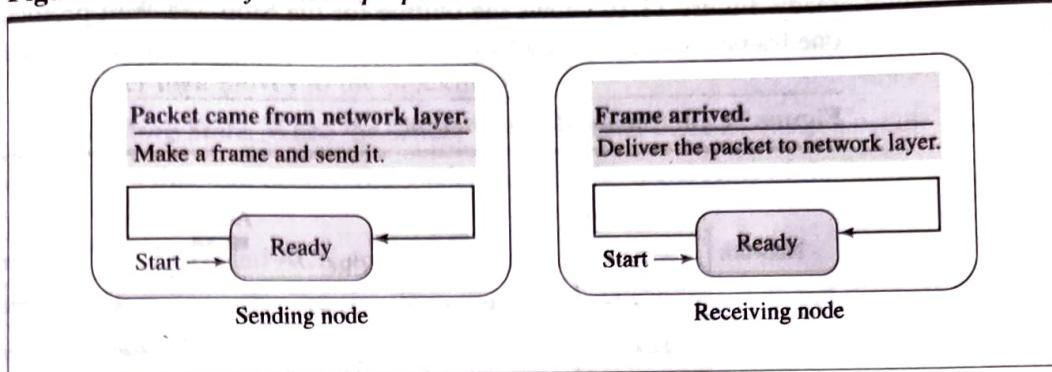
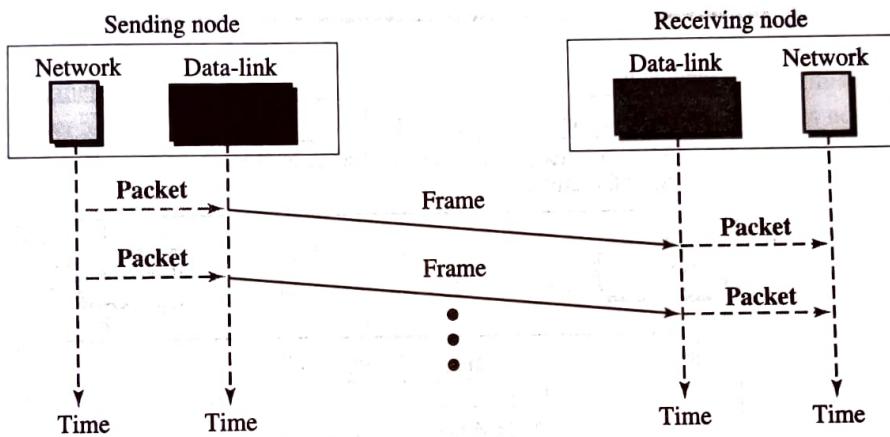
Figure 11.8 FSMs for the simple protocol**Example 11.2**

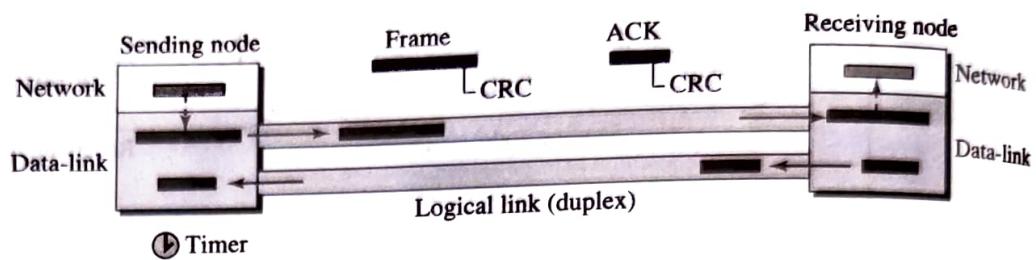
Figure 11.9 shows an example of communication using this protocol. It is very simple. The sender sends frames one after another without even thinking about the receiver.

Figure 11.9 Flow diagram for Example 11.2**11.2.2 Stop-and-Wait Protocol**

Our second protocol is called the **Stop-and-Wait protocol**, which uses both flow and error control. We show a primitive version of this protocol here, but we discuss the more sophisticated version in Chapter 23 when we have learned about sliding windows. In this protocol, the sender sends one frame at a time and waits for an acknowledgment before sending the next one. To detect corrupted frames, we need to add a CRC (see Chapter 10) to each data frame. When a frame arrives at the receiver site, it is checked. If its CRC is incorrect, the frame is corrupted and silently discarded. The silence of the receiver is a signal for the sender that a frame was either corrupted or lost. Every time the sender sends a frame, it starts a timer. If an acknowledgment arrives before the timer expires, the timer is stopped and the sender sends the next frame (if it has one to send). If the timer expires, the sender resends the previous frame, assuming that the frame was either lost or corrupted. This means that the sender needs to keep a copy of the frame until its acknowledgment arrives. When the corresponding

acknowledgment arrives, the sender discards the copy and sends the next frame if it is ready. Figure 11.10 shows the outline for the Stop-and-Wait protocol. Note that only one frame and one acknowledgment can be in the channels at any time.

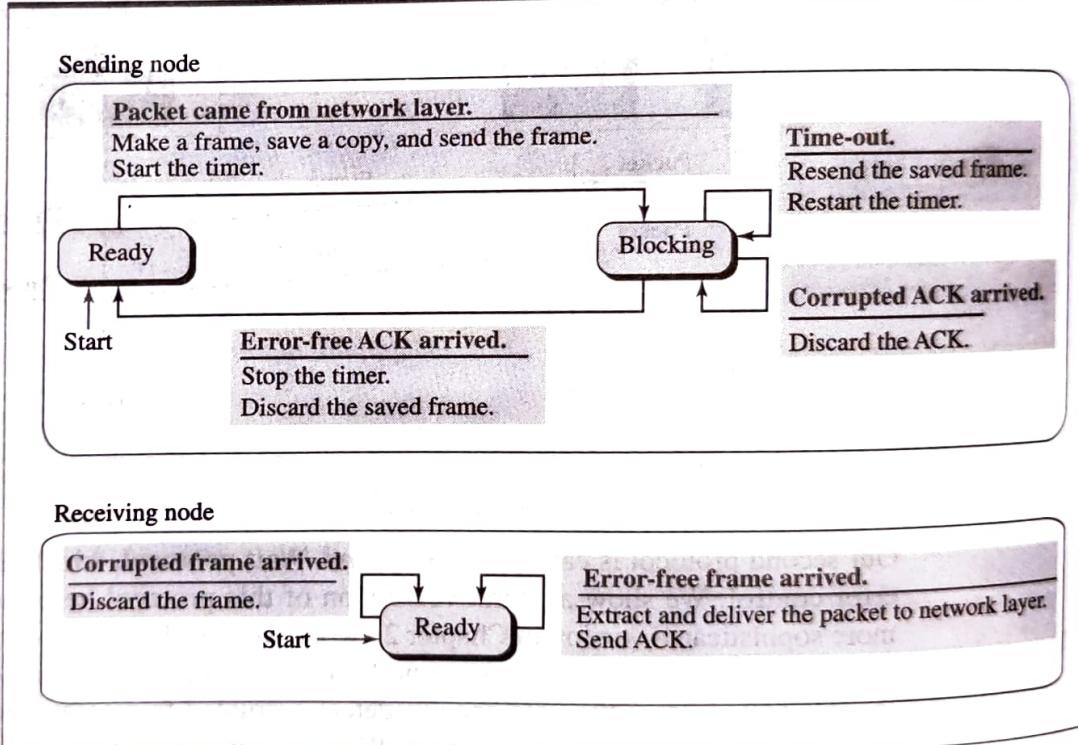
Figure 11.10 Stop-and-Wait protocol



FSMs

Figure 11.11 shows the FSMs for our primitive Stop-and-Wait protocol.

Figure 11.11 FSM for the Stop-and-Wait protocol



We describe the sender and receiver states below.

Sender States

The sender is initially in the ready state, but it can move between the ready and blocking state.

- **Ready State.** When the sender is in this state, it is only waiting for a packet from the network layer. If a packet comes from the network layer, the sender creates a frame, saves a copy of the frame, starts the only timer and sends the frame. The sender then moves to the blocking state.
- **Blocking State.** When the sender is in this state, three events can occur:
 - a. If a time-out occurs, the sender resends the saved copy of the frame and restarts the timer.
 - b. If a corrupted ACK arrives, it is discarded.
 - c. If an error-free ACK arrives, the sender stops the timer and discards the saved copy of the frame. It then moves to the ready state.

Receiver

The receiver is always in the *ready* state. Two events may occur:

- a. If an error-free frame arrives, the message in the frame is delivered to the network layer and an ACK is sent.
- b. If a corrupted frame arrives, the frame is discarded.

Example 11.3

Figure 11.12 shows an example. The first frame is sent and acknowledged. The second frame is sent, but lost. After time-out, it is resent. The third frame is sent and acknowledged, but the acknowledgment is lost. The frame is resent. However, there is a problem with this scheme. The network layer at the receiver site receives two copies of the third packet, which is not right. In the next section, we will see how we can correct this problem using sequence numbers and acknowledgment numbers.

Sequence and Acknowledgment Numbers

We saw a problem in Example 11.3 that needs to be addressed and corrected. Duplicate packets, as much as corrupted packets, need to be avoided. As an example, assume we are ordering some item online. If each packet defines the specification of an item to be ordered, duplicate packets mean ordering an item more than once. To correct the problem in Example 11.3, we need to add **sequence numbers** to the data frames and **acknowledgment numbers** to the ACK frames. However, numbering in this case is very simple. Sequence numbers are 0, 1, 0, 1, 0, 1, . . . ; the acknowledgment numbers can also be 1, 0, 1, 0, 1, 0, . . . In other words, the sequence numbers start with 0, the acknowledgment numbers start with 1. An acknowledgment number always defines the sequence number of the next frame to receive.

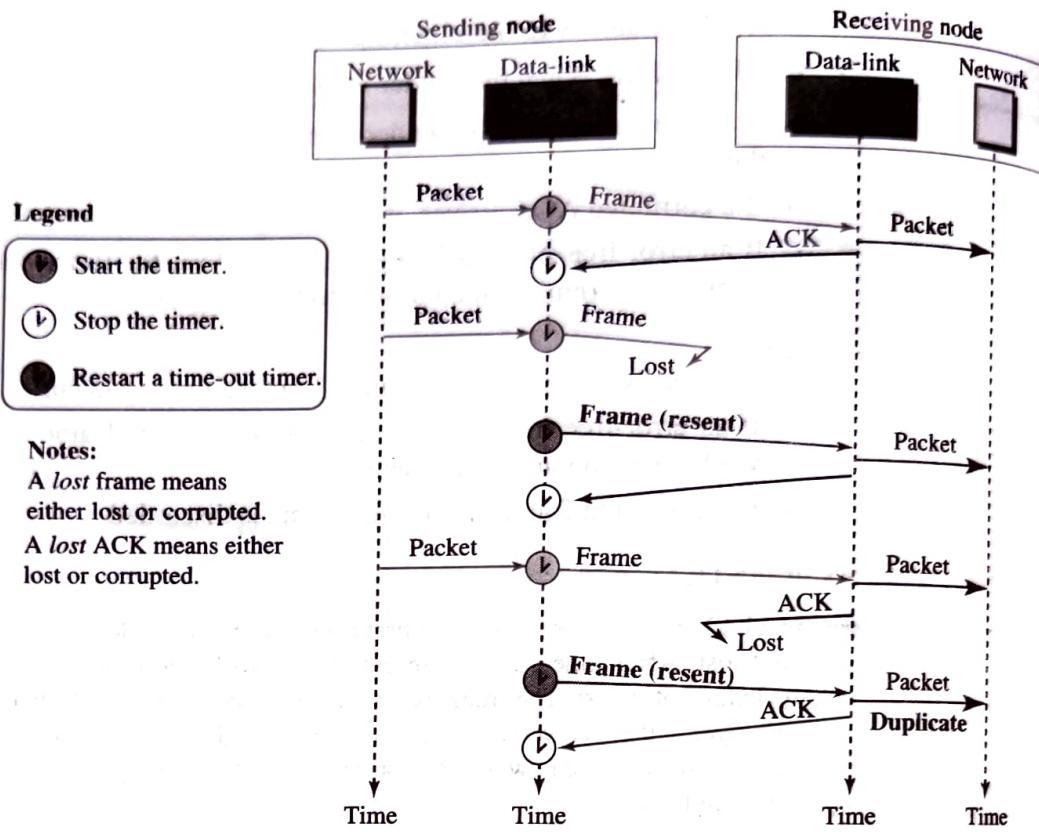
Example 11.4

Figure 11.13 shows how adding sequence numbers and acknowledgment numbers can prevent duplicates. The first frame is sent and acknowledged. The second frame is sent, but lost. After time-out, it is resent. The third frame is sent and acknowledged, but the acknowledgment is lost. The frame is resent.

FSMs with Sequence and Acknowledgment Numbers

We can change the FSM in Figure 11.11 to include the sequence and acknowledgment numbers, but we leave this as a problem at the end of the chapter.

Figure 11.12 Flow diagram for Example 11.3

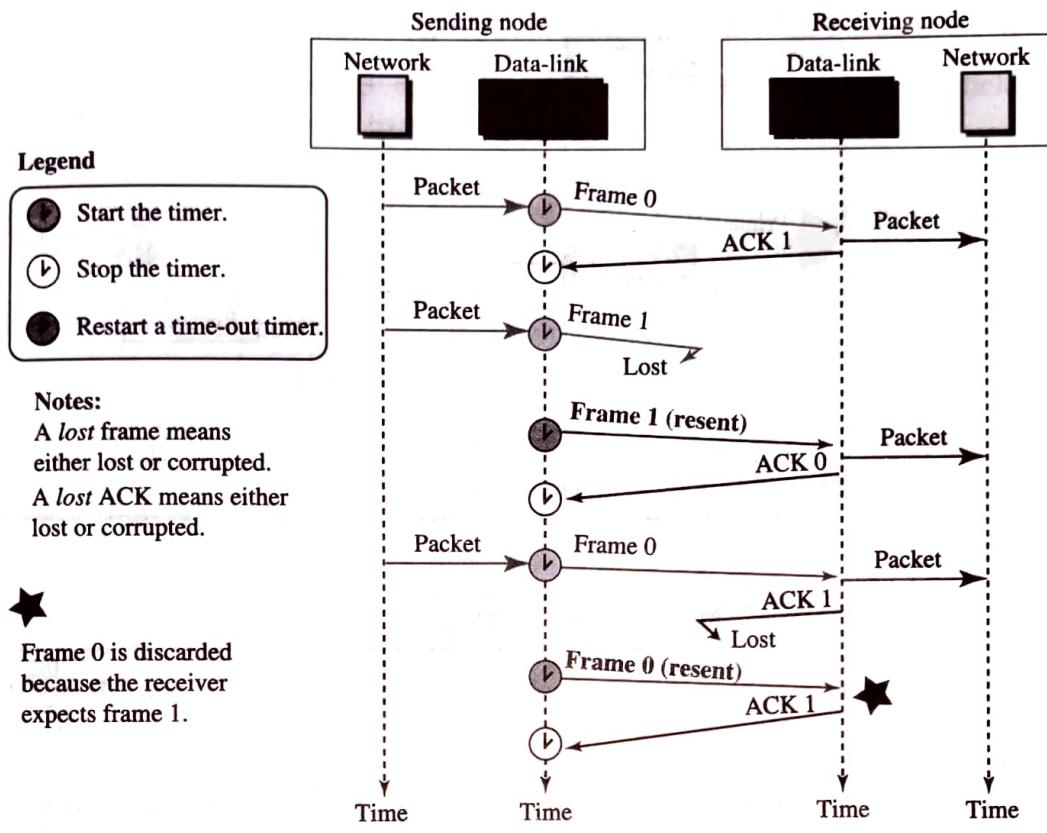


11.2.3 Piggybacking

The two protocols we discussed in this section are designed for unidirectional communication, in which data is flowing only in one direction although the acknowledgment may travel in the other direction. Protocols have been designed in the past to allow data to flow in both directions. However, to make the communication more efficient, the data in one direction is piggybacked with the acknowledgment in the other direction. In other words, when node A is sending data to node B, Node A also acknowledges the data received from node B. Because piggybacking makes communication at the data-link layer more complicated, it is not a common practice. We discuss two-way communication and piggybacking in more detail in Chapter 23.

11.3 HDLC

High-level Data Link Control (HDLC) is a bit-oriented protocol for communication over point-to-point and multipoint links. It implements the Stop-and-Wait protocol we discussed earlier. Although this protocol is more a theoretical issue than practical, most of the concept defined in this protocol is the basis for other practical protocols such as PPP, which we discuss next, or the Ethernet protocol, which we discuss in wired LANs (Chapter 13), or in wireless LANs (Chapter 15).

Figure 11.13 Flow diagram for Example 11.4

11.3.1 Configurations and Transfer Modes

HDLC provides two common transfer modes that can be used in different configurations: *normal response mode (NRM)* and *asynchronous balanced mode (ABM)*. In *normal response mode (NRM)*, the station configuration is unbalanced. We have one primary station and multiple secondary stations. A *primary station* can send commands; a *secondary station* can only respond. The NRM is used for both point-to-point and multipoint links, as shown in Figure 11.14.

In ABM, the configuration is balanced. The link is point-to-point, and each station can function as a primary and a secondary (acting as peers), as shown in Figure 11.15. This is the common mode today.

11.3.2 Framing

To provide the flexibility necessary to support all the options possible in the modes and configurations just described, HDLC defines three types of frames: *information frames (I-frames)*, *supervisory frames (S-frames)*, and *unnumbered frames (U-frames)*. Each type of frame serves as an envelope for the transmission of a different type of message. I-frames are used to data-link user data and control information relating to user data (piggy-backing). S-frames are used only to transport control information. U-frames are reserved for system management. Information carried by U-frames is intended for managing the

Figure 11.14 Normal response mode

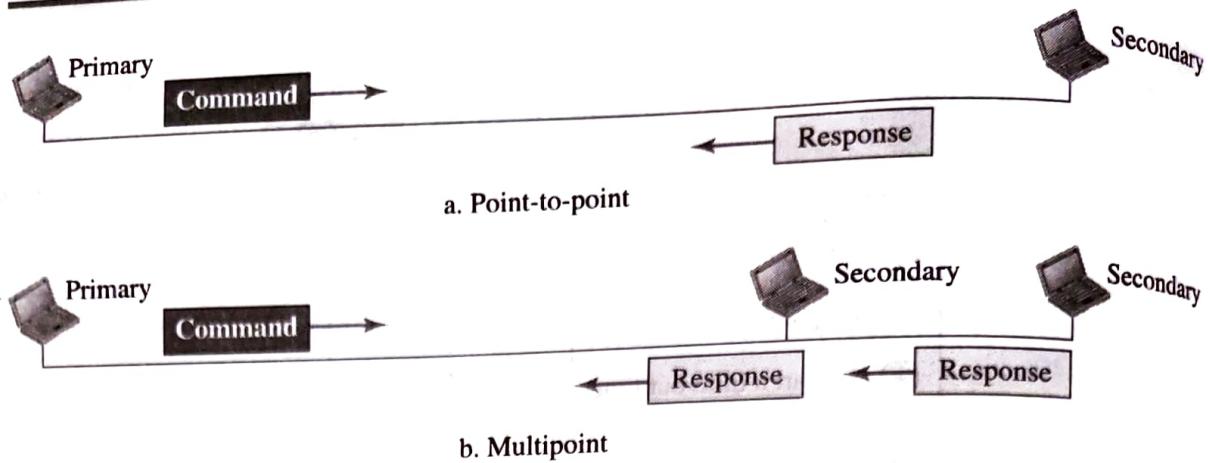
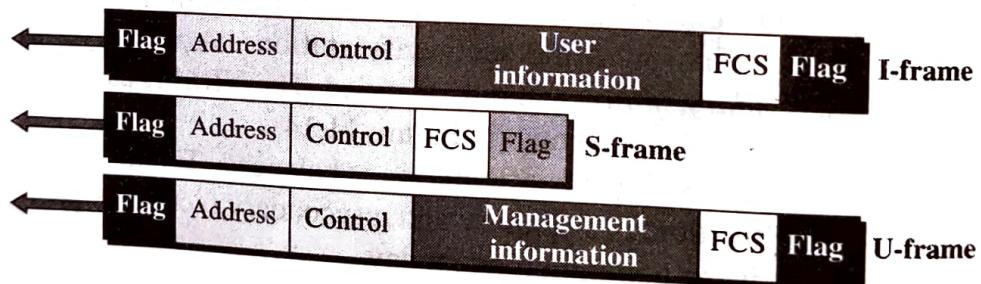


Figure 11.15 Asynchronous balanced mode



link itself. Each frame in HDLC may contain up to six fields, as shown in Figure 11.16: a beginning flag field, an address field, a control field, an information field, a frame check sequence (FCS) field, and an ending flag field. In multiple-frame transmissions, the ending flag of one frame can serve as the beginning flag of the next frame.

Figure 11.16 HDLC frames



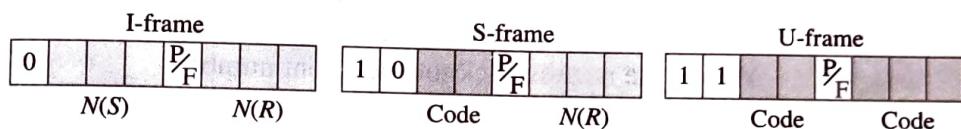
Let us now discuss the fields and their use in different frame types.

- ❑ **Flag field.** This field contains synchronization pattern 01111110, which identifies both the beginning and the end of a frame.
- ❑ **Address field.** This field contains the address of the secondary station. If a primary station created the frame, it contains a *to* address. If a secondary station creates the frame, it contains a *from* address. The address field can be one byte or several bytes long, depending on the needs of the network.

- ❑ **Control field.** The control field is one or two bytes used for flow and error control. The interpretation of bits are discussed later.
- ❑ **Information field.** The information field contains the user's data from the network layer or management information. Its length can vary from one network to another.
- ❑ **FCS field.** The frame check sequence (FCS) is the HDLC error detection field. It can contain either a 2- or 4-byte CRC.

The control field determines the type of frame and defines its functionality. So let us discuss the format of this field in detail. The format is specific for the type of frame, as shown in Figure 11.17.

Figure 11.17 Control field format for the different frame types



Control Field for I-Frames

I-frames are designed to carry user data from the network layer. In addition, they can include flow- and error-control information (piggybacking). The subfields in the control field are used to define these functions. The first bit defines the type. If the first bit of the control field is 0, this means the frame is an I-frame. The next 3 bits, called $N(S)$, define the sequence number of the frame. Note that with 3 bits, we can define a sequence number between 0 and 7. The last 3 bits, called $N(R)$, correspond to the acknowledgment number when piggybacking is used. The single bit between $N(S)$ and $N(R)$ is called the P/F bit. The P/F field is a single bit with a dual purpose. It has meaning only when it is set (bit = 1) and can mean poll or final. It means *poll* when the frame is sent by a primary station to a secondary (when the address field contains the address of the receiver). It means *final* when the frame is sent by a secondary to a primary (when the address field contains the address of the sender).

Control Field for S-Frames

Supervisory frames are used for flow and error control whenever piggybacking is either impossible or inappropriate. S-frames do not have information fields. If the first 2 bits of the control field are 10, this means the frame is an S-frame. The last 3 bits, called $N(R)$, correspond to the acknowledgment number (ACK) or negative acknowledgment number (NAK), depending on the type of S-frame. The 2 bits called *code* are used to define the type of S-frame itself. With 2 bits, we can have four types of S-frames, as described below:

- ❑ **Receive ready (RR).** If the value of the code subfield is 00, it is an RR S-frame. This kind of frame acknowledges the receipt of a safe and sound frame or group of frames. In this case, the value of the $N(R)$ field defines the acknowledgment number.

- ❑ **Receive not ready (RNR).** If the value of the code subfield is 10, it is an RNR S-frame. This kind of frame is an RR frame with additional functions. It acknowledges the receipt of a frame or group of frames, and it announces that the receiver is busy and cannot receive more frames. It acts as a kind of congestion-control mechanism by asking the sender to slow down. The value of $N(R)$ is the acknowledgment number.
- ❑ **Reject (REJ).** If the value of the code subfield is 01, it is an REJ S-frame. This is a NAK frame, but not like the one used for Selective Repeat ARQ. It is a NAK that can be used in Go-Back-N ARQ to improve the efficiency of the process by informing the sender, before the sender timer expires, that the last frame is lost or damaged. The value of $N(R)$ is the negative acknowledgment number.
- ❑ **Selective reject (SREJ).** If the value of the code subfield is 11, it is an SREJ S-frame. This is a NAK frame used in Selective Repeat ARQ. Note that the HDLC Protocol uses the term *selective reject* instead of *selective repeat*. The value of $N(R)$ is the negative acknowledgment number.

Control Field for U-Frames

Unnumbered frames are used to exchange session management and control information between connected devices. Unlike S-frames, U-frames contain an information field, but one used for system management information, not user data. As with S-frames, however, much of the information carried by U-frames is contained in codes included in the control field. U-frame codes are divided into two sections: a 2-bit prefix before the P/F bit and a 3-bit suffix after the P/F bit. Together, these two segments (5 bits) can be used to create up to 32 different types of U-frames.

Control Field for U-Frames

Unnumbered frames are used to exchange session management and control information between connected devices. Unlike S-frames, U-frames contain an information field, but one used for system management information, not user data. As with S-frames, however, much of the information carried by U-frames is contained in codes included in the control field. U-frame codes are divided into two sections: a 2-bit prefix before the P/F bit and a 3-bit suffix after the P/F bit. Together, these two segments (5 bits) can be used to create up to 32 different types of U-frames.

Example 11.5

Figure 11.18 shows how U-frames can be used for connection establishment and connection release. Node A asks for a connection with a set asynchronous balanced mode (SABM) frame; node B gives a positive response with an unnumbered acknowledgment (UA) frame. After these two exchanges, data can be transferred between the two nodes (not shown in the figure). After data transfer, node A sends a DISC (disconnect) frame to release the connection; it is confirmed by node B responding with a UA (unnumbered acknowledgment).

Example 11.6

Figure 11.19 shows two exchanges using piggybacking. The first is the case where no error has occurred; the second is the case where an error has occurred and some frames are discarded.