FUTURE VISION BIE

One Stop for All Study Materials

& Lab Programs

Future Vision

By K B Hemanth Raj

Scan the QR Code to Visit the Web Page

Or

Visit : https://hemanthrajhemu.github.io

Gain Access to All Study Materials according to VTU,
CSE – Computer Science Engineering,
ISE – Information Science Engineering,
ECE - Electronics and Communication Engineering
& MORE...

Join Telegram to get Instant Updates: https://bit.ly/VTU_TELEGRAM

Contact: MAIL: futurevisionbie@gmail.com

INSTAGRAM: www.instagram.com/hemanthraj_hemu/

INSTAGRAM: www.instagram.com/futurevisionbie/

WHATSAPP SHARE: https://bit.ly/FVBIESHARE

CHAPTER 9

Introduction to Data-Link Layer

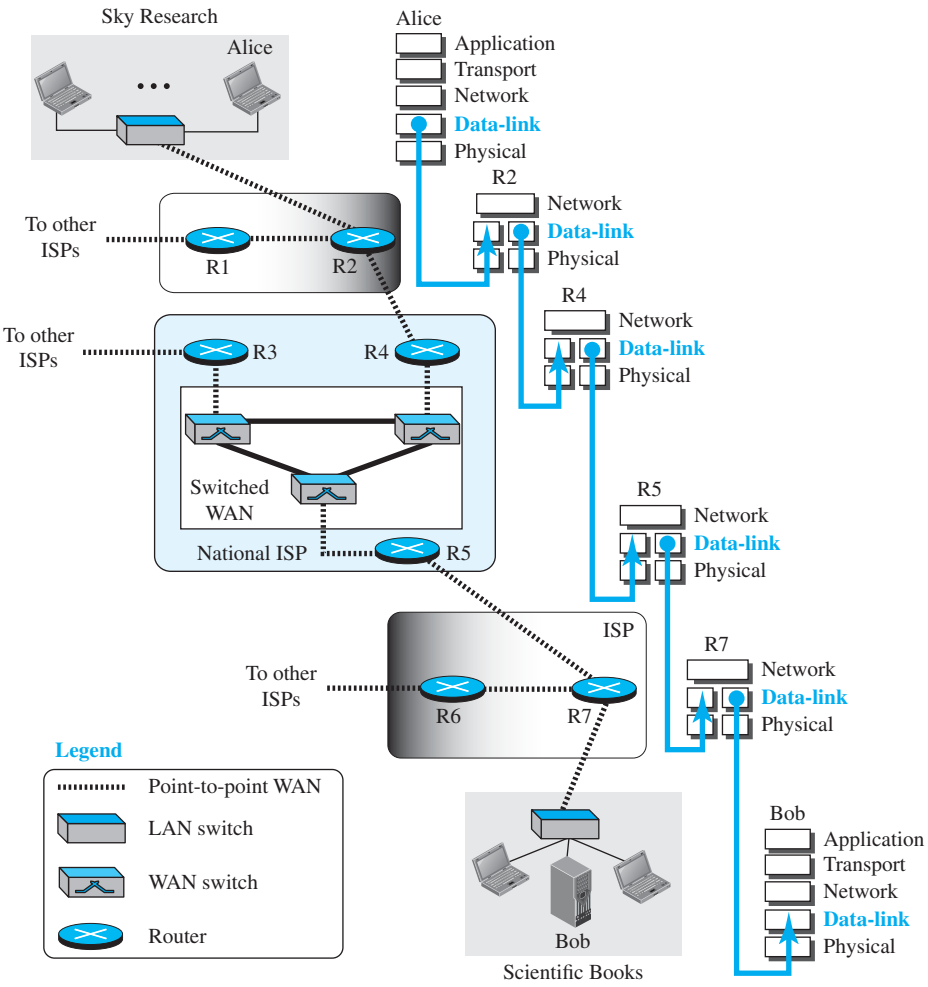
The TCP/IP protocol suite does not define any protocol in the data-link layer or physical layer. These two layers are territories of networks that when connected make up the Internet. These networks, wired or wireless, provide services to the upper three layers of the TCP/IP suite. This may give us a clue that there are several standard protocols in the market today. For this reason, we discuss the data-link layer in several chapters. This chapter is an introduction that gives the general idea and common issues in the data-link layer that relate to all networks.

- The first section introduces the data-link layer. It starts with defining the concept of links and nodes. The section then lists and briefly describes the services provided by the data-link layer. It next defines two categories of links: point-to-point and broadcast links. The section finally defines two sublayers at the data-link layer that will be elaborated on in the next few chapters.
- The second section discusses link-layer addressing. It first explains the rationale behind the existence of an addressing mechanism at the data-link layer. It then describes three types of link-layer addresses to be found in some link-layer protocols. The section discusses the Address Resolution Protocol (ARP), which maps the addresses at the network layer to addresses at the data-link layer. This protocol helps a packet at the network layer find the link-layer address of the next node for delivery of the frame that encapsulates the packet. To show how the network layer helps us to find the data-link-layer addresses, a long example is included in this section that shows what happens at each node when a packet is travelling through the Internet.

9.1 INTRODUCTION

The Internet is a combination of networks glued together by connecting devices (routers or switches). If a packet is to travel from a host to another host, it needs to pass through these networks. Figure 9.1 shows the same scenario we discussed in Chapter 3, but we are now interested in communication at the data-link layer. Communication at the data-link layer is made up of five separate logical connections between the data-link layers in the path.

Figure 9.1 *Communication at the data-link layer*



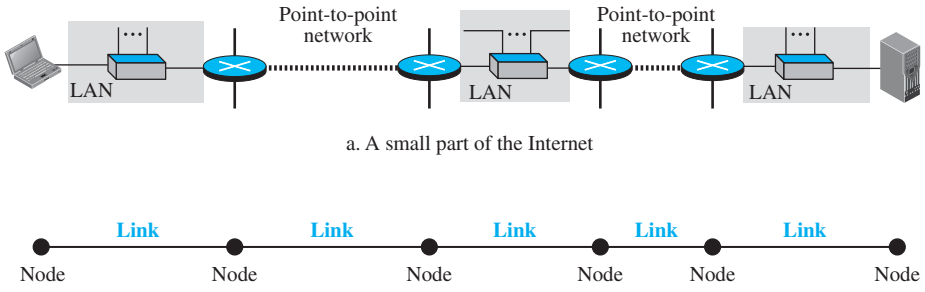
The data-link layer at Alice's computer communicates with the data-link layer at router R2. The data-link layer at router R2 communicates with the data-link layer at router R4,

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. Theses 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*



a. A small part of the Internet

b. 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

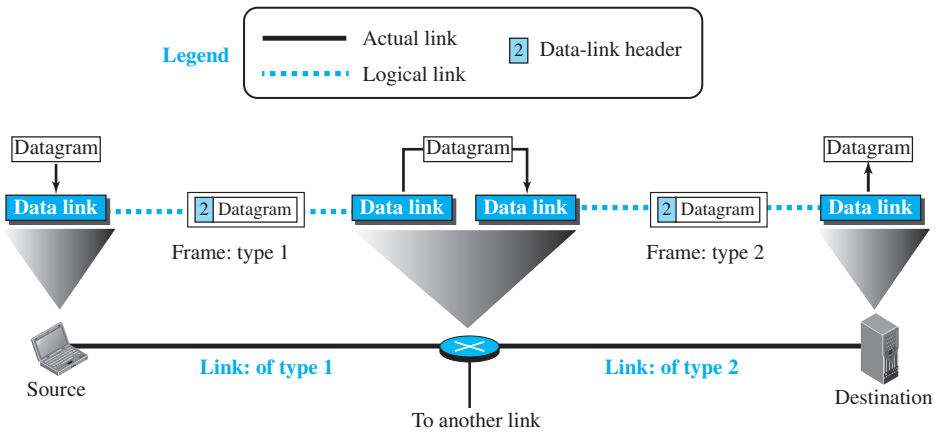
The data-link layer is located between the physical and the network layers. The data-link layer provides services to the network layer; it receives services from the physical layer. Let us discuss services provided by the data-link layer.

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

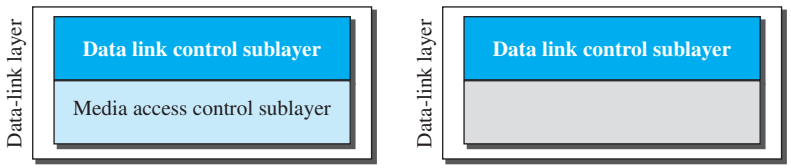
Although two nodes are physically connected by a transmission medium such as cable or air, we need to remember that the data-link layer controls how the medium is used. We can have a data-link layer that uses the whole capacity of the medium; we can also

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*



a. Data-link layer of a broadcast link b. Data-link layer of a point-to-point link

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

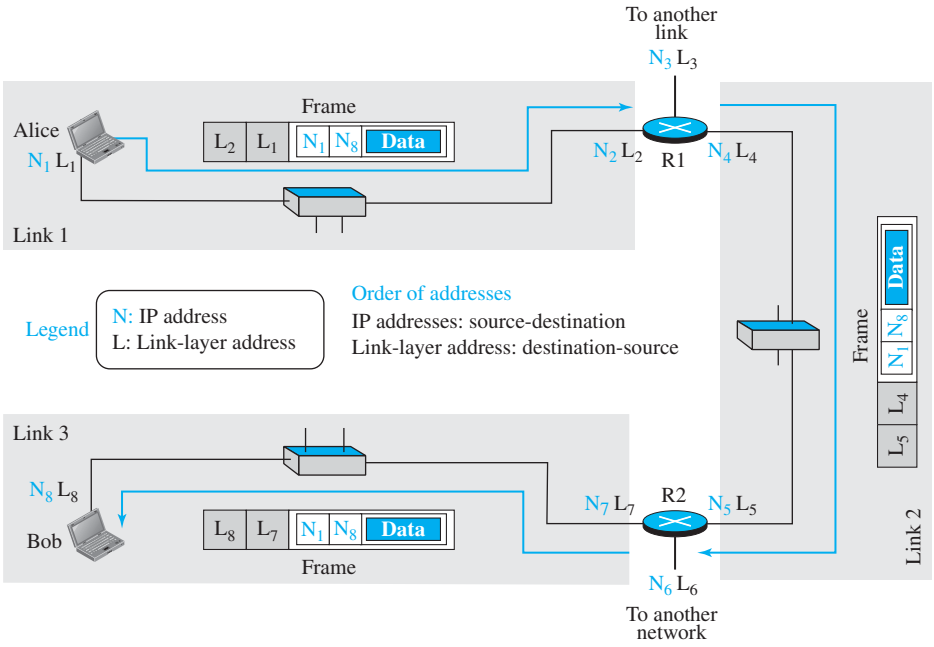
The next issue we need to discuss about the data-link layer is the link-layer addresses. In Chapter 18, we will discuss IP addresses as the identifiers at the network layer that define the exact points in the Internet where the source and destination hosts are connected. However, in a connectionless 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 L_1 and L_2. In link 2, they are L_4 and L_5. In link 3, they are L_7 and L_8. 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

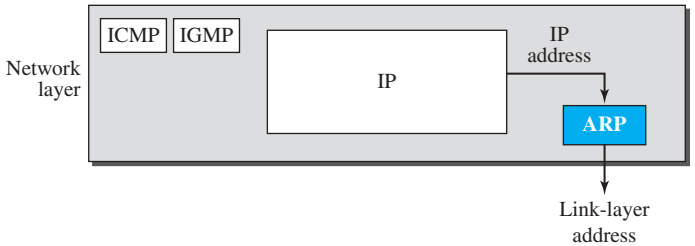
As we will see in Chapter 13, the broadcast link-layer addresses in the most common LAN, Ethernet, are 48 bits, all 1s, that are presented as 12 hexadecimal digits separated by colons. The following shows a broadcast address:

FF:FF:FF:FF:FF:FF

9.2.2 Address Resolution Protocol (ARP)

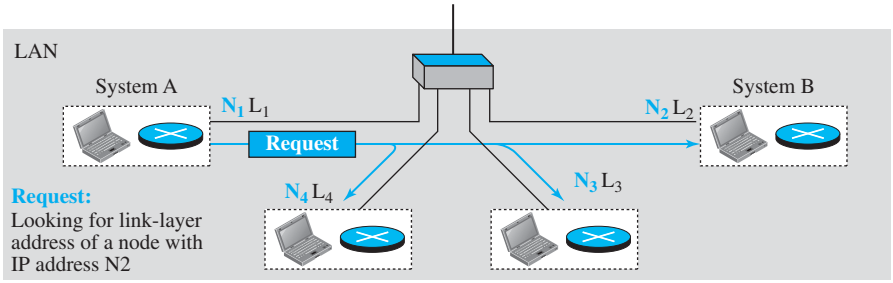
Anytime a node has an IP datagram to send to another node in a link, it has the IP address of the receiving node. The source host knows the IP address of the default router. Each router except the last one in the path gets the IP address of the next router by using its forwarding table. The last router 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*

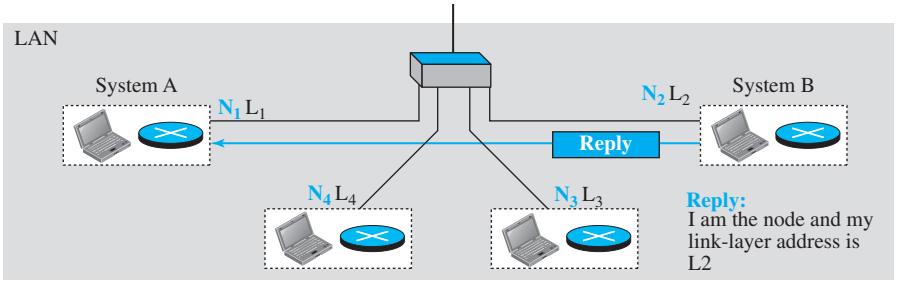


Anytime a host or a router needs to find the link-layer address of another host or router in its network, it sends an ARP request packet. The packet includes the link-layer and IP addresses of the sender and the IP address of the receiver. Because the sender does not know the link-layer address of the 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*



a. ARP request is broadcast



b. ARP reply is unicast

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*

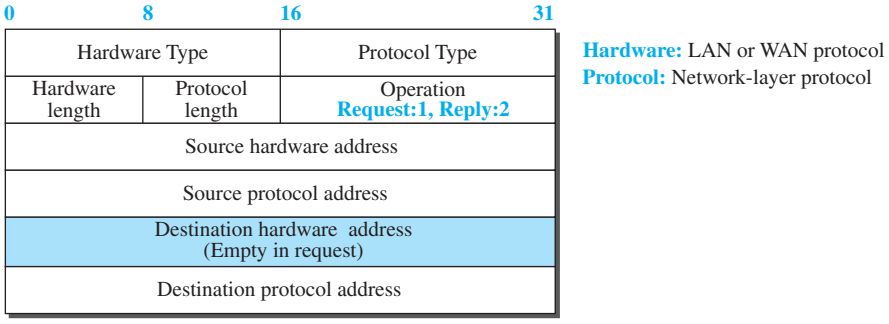
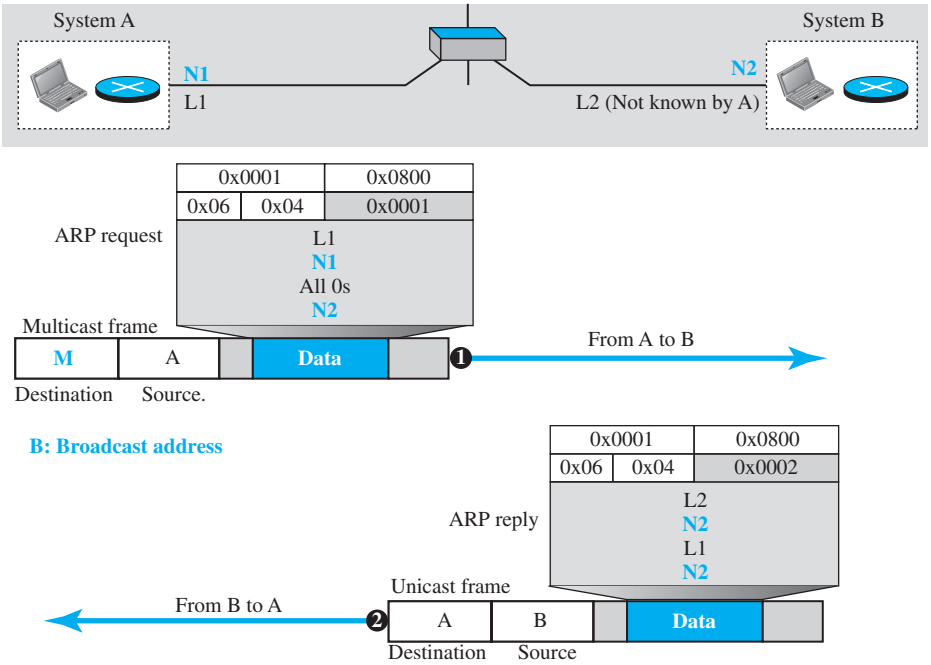


Figure 9.9 *Example 9.4*

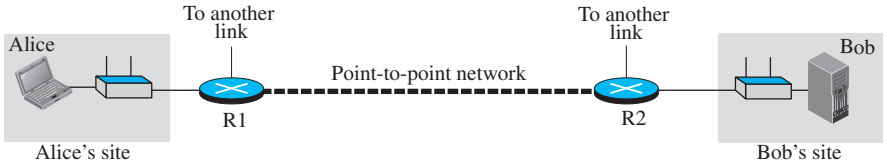


9.2.3 An Example of Communication

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.

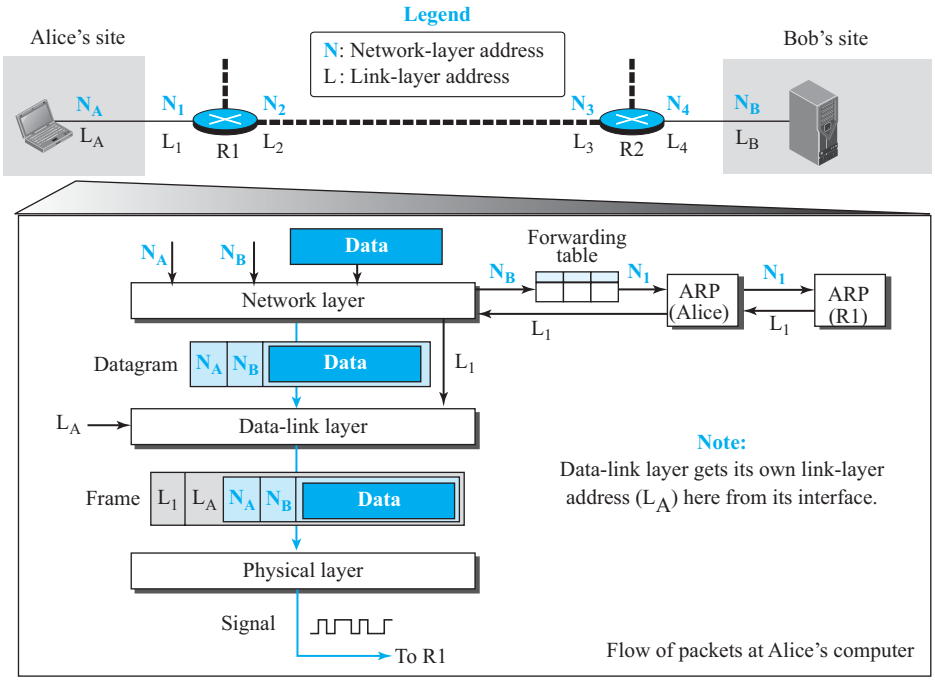
Figure 9.10 *The internet for our example*



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

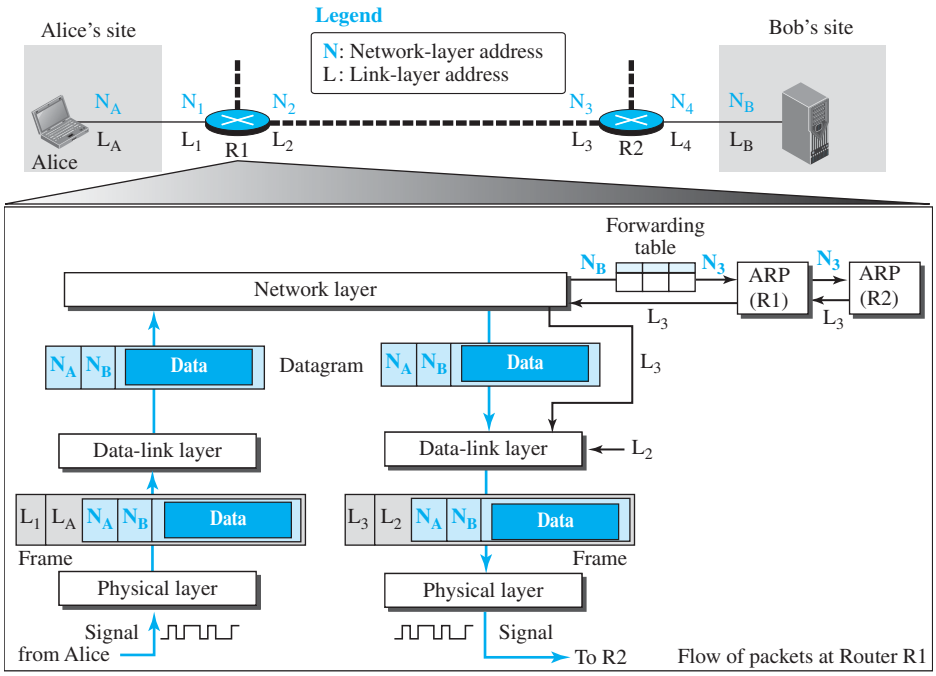
needs to find the link-layer address of router R1. It uses its ARP to find the link-layer address $\mathbf{L_1}$. The network layer can now pass the datagram with the link-layer address to the data-link layer.

The data-link layer knows its own link-layer address, $\mathbf{L_A}$. 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*



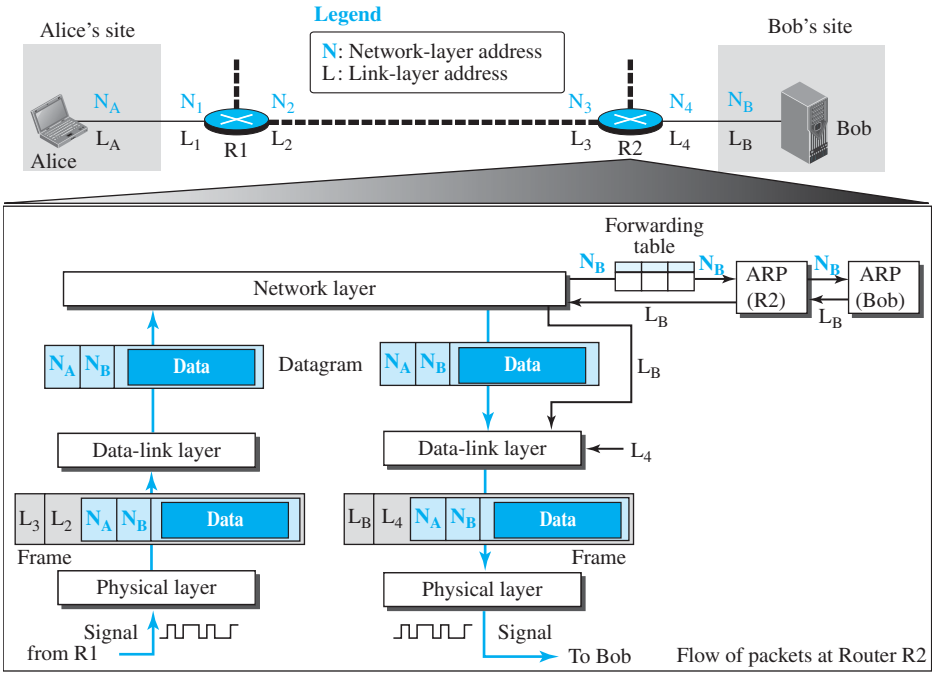
At arrival, the physical layer of the left link creates the frame and passes it to the data-link layer. The data-link layer decapsulates the datagram and passes it to the network layer. The network layer examines the network-layer address of the datagram and finds that the datagram needs to be delivered to the device with IP address $\mathbf{N_B}$. The network layer consults its routing table to find out which is the next node (router) in the path to $\mathbf{N_B}$. The forwarding table returns $\mathbf{N_3}$. 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 $\mathbf{L_3}$. The network layer passes the datagram and $\mathbf{L_3}$ 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.*



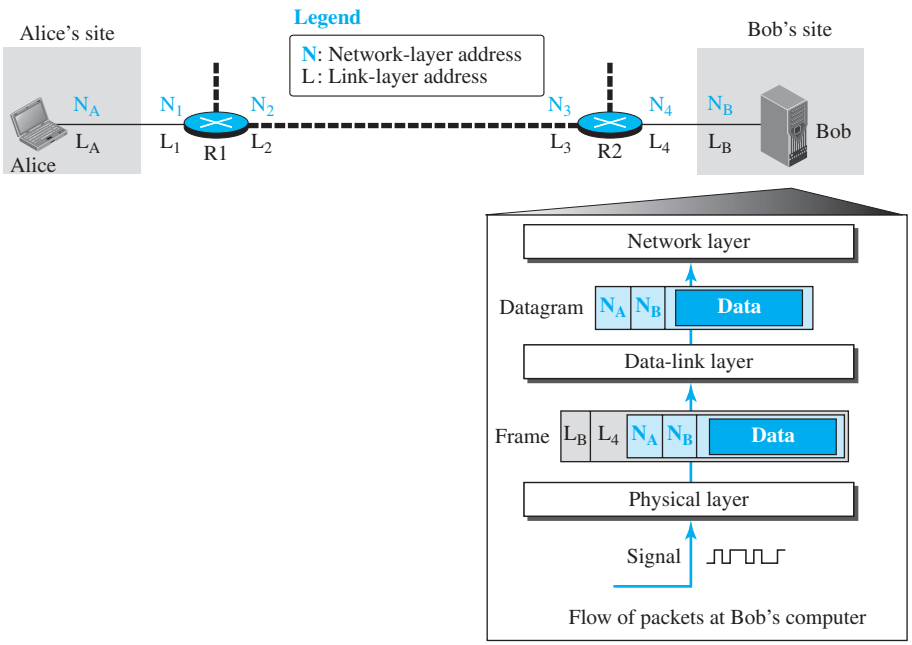
Activities at Bob's Site

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, NA and NB, have not been changed during the whole journey. However, all four network-layer addresses of routers R1 and R2 (N1, N2, N3, and N4) 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

as *links*. A path in the Internet from a source host to a destination host is a set of nodes and links through which a packet should travel.

The data-link layer is responsible for the creation and delivery of a frame to another node, along the link. It is responsible for packetizing (framing), flow control, error control, and congestion control along the link. Two data-link layers at the two ends of a link coordinate to deliver a frame from one node to the next.

As with any delivery between a source and destination in which there are many paths, we need two types of addressing. The end-to-end addressing defines the source and destination; the link-layer addressing defines the addresses of the nodes that the packet should pass through. To avoid including the link-layer addresses of all of these nodes in the frame, the Address Resolution Protocol (ARP) was devised to map an IP address to its corresponding link-layer address. When a packet is at one node ready to be sent to the next, the forwarding table finds the IP address of the next node and ARP find its link-layer address.

9.4 PRACTICE SET

9.4.1 Quizzes

A set of interactive quizzes for this chapter can be found on the book website. It is strongly recommended that the student take the quizzes to check his/her understanding of the materials before continuing with the practice set.

9.4.2 Questions

Q9-1. Distinguish between communication at the network layer and communication at the data-link layer.

Q9-2. Distinguish between a point-to-point link and a broadcast link.

Q9-3. Can two hosts in two different networks have the same link-layer address? Explain.

Q9-4. Is the size of the ARP packet fixed? Explain.

Q9-5. What is the size of an ARP packet when the protocol is IPv4 and the hardware is Ethernet?

Q9-6. Assume we have an isolated link (not connected to any other link) such as a private network in a company. Do we still need addresses in both the network layer and the data-link layer? Explain.

Q9-7. In Figure 9.9, why is the destination hardware address all 0s in the ARP request message?

Q9-8. In Figure 9.9, why is the destination hardware address of the frame from A to B a broadcast address?

Q9-9. In Figure 9.9, how does system A know what the link-layer address of system B is when it receives the ARP reply?

Q9-10. When we talkabout the broadcast address in a link, do we mean sending a message to all hosts and routers in the link or to all hosts and routers in the Internet? In other words, does a broadcast address have a local jurisdiction or a universal jurisdiction? Explain.

Q9-11. Why does a host or a router need to run the ARP program all of the time in the background?

Q9-12. Why does a router normally have more than one interface?

Q9-13. Why is it better not to change an end-to-end address from the source to the destination?

Q9-14. How many IP addresses and how many link-layer addresses should a router have when it is connected to five links?

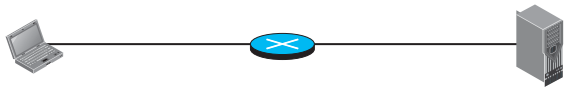
9.4.3 Problems

P9-1. Assume we have an internet (a private small internet) in which all hosts are connected in a mesh topology. Do we need routers in this internet? Explain.

P9-2. In the previous problem, do we need both network and data-link layers?

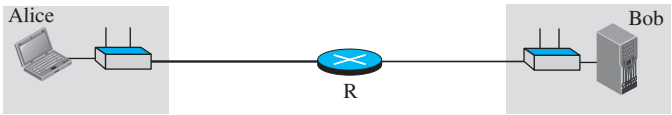
P9-3. Explain why we do not need the router in Figure 9.15.

Figure 9.15 *Problem 9-3*



P9-4. Explain why we may need a router in Figure 9.16.

Figure 9.16 *Problem 9-4*



P9-5. Is the current Internet using circuit-switching or packet-switching at the data-link layer? Explain.

P9-6. Assume Alice is travelling from 2020 Main Street in Los Angeles to 1432 American Boulevard in Chicago. If she is travelling by air from Los Angeles Airport to Chicago Airport,

a. find the end-to-end addresses in this scenario.

b. find the link-layer addresses in this scenario.

P9-7. In the previous problem, assume Alice cannot find a direct flight from the Los Angeles to the Chicago. If she needs to change flights in Denver,

a. find the end-to-end addresses in this scenario.

b. find the link-layer addresses in this scenario.

P9-8. When we send a letter using the services provided by the post office, do we use an end-to-end address? Does the post office necessarily use an end-to-end address to deliver the mail? Explain.

P9-9. In Figure 9.5, assume Link 2 is broken. How can Alice communicate with Bob?

P9-10. In Figure 9.5, show the process of frame change in routers R1 and R2.

P9-11. In Figure 9.7, assume system B is not running the ARP program. What would happen?

P9-12. In Figure 9.7, do you think that system A should first check its cache for mapping from N2 to L2 before even broadcasting the ARP request?

P9-13. Assume the network in Figure 9.7 does not support broadcasting. What do you suggest for sending the ARP request in this network?

P9-14. In Figures 9.11 to 9.13, both the forwarding table and ARP are doing a kind of mapping. Show the difference between them by listing the input and output of mapping for a forwarding table and ARP.

P9-15. Figure 9.7 shows a system as either a host or a router. What would be the actual entity (host or router) of system A and B in each of the following cases:

a. If the link is the first one in the path?

b. If the link is the middle one in the path?

c. If the link is the last one in the path?

d. If there is only one link in the path (local communication)?

https://hemanthrajhemu.github.io

CHAPTER 11

Data Link Control (DLC)

As we discussed in Chapter 9, the data-link layer is divided into two sublayers. In this chapter, we discuss the upper sublayer of the data-link layer (DLC). The lower sublayer, multiple access control (MAC) will be discussed in Chapter 12. We have already discussed error detection and correction, an issue that is encountered in several layers, in Chapter 10.

This chapter is divided into four sections.

- The first section discusses the general services provided by the DLC sublayer. It first describes framing and two types of frames used in this sublayer. The section then discusses flow and error control. Finally, the section explains that a DLC protocol can be either connectionless or connection-oriented.
- The second section discusses some simple and common data-link protocols that are implemented at the DLC sublayer. The section first describes the Simple Protocol. It then explains the Stop-and-Wait Protocol.
- The third section introduces HDLC, a protocol that is the basis of all common data-link protocols in use today such as PPP and Ethernet. The section first talks about configurations and transfer modes. It then describes framing and three different frame formats used in this protocol.
- The fourth section discusses PPP, a very common protocol for point-to-point access. It first introduces the services provided by the protocol. The section also describes the format of the frame in this protocol. It then describes the transition mode in the protocol using an FSM. The section finally explains multiplexing in PPP.

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.

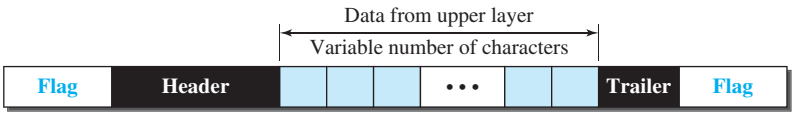
Our main discussion in this chapter concerns *variable-size framing,* prevalent in local-area networks. In variable-size framing, we need a way to define the end of one frame and the beginning of the next. Historically, two approaches were used for this purpose: a character-oriented approach and a bit-oriented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

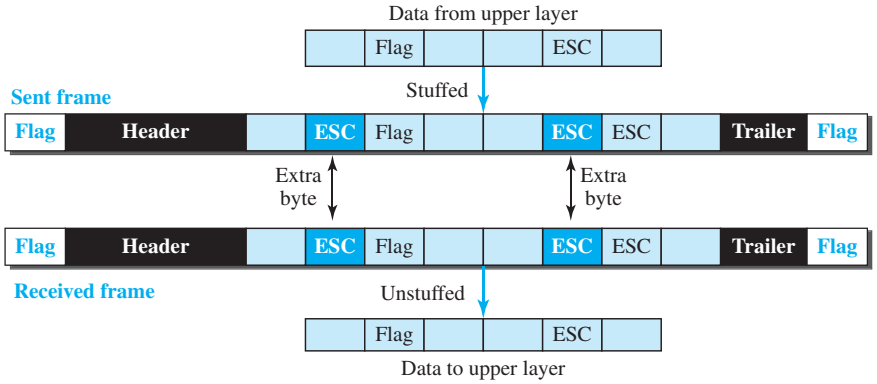
start or end of a frame. Figure 11.1 shows the format of a frame in a character-oriented protocol.

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

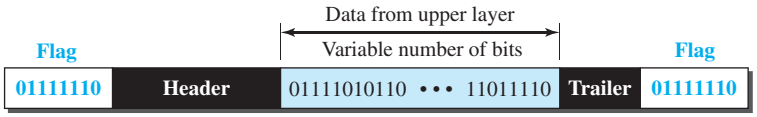
receiver removes the escape character, but keeps the next byte, which is incorrectly interpreted as the end of the frame. To solve this problem, the escape characters that are part of the text must also be marked by another escape character. In other words, if the escape character is part of the text,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



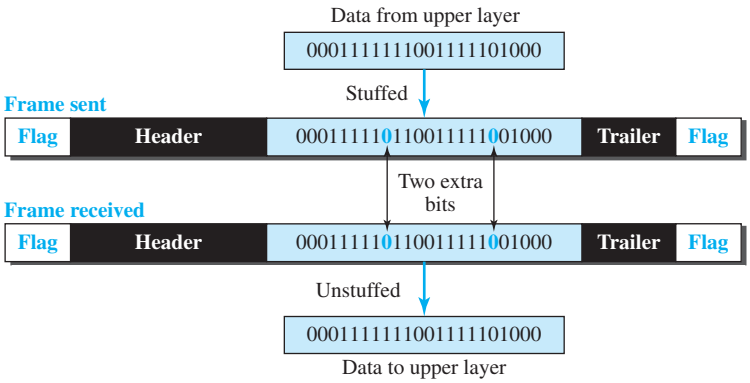
This flag can create the same type of problem we saw in the character-oriented protocols. That is, if the flag pattern appears in the data, we need to somehow inform the receiver that this is not the end of the frame. We do this by stuffing 1 single bit (instead of 1 byte) to prevent the pattern from 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 0111110 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

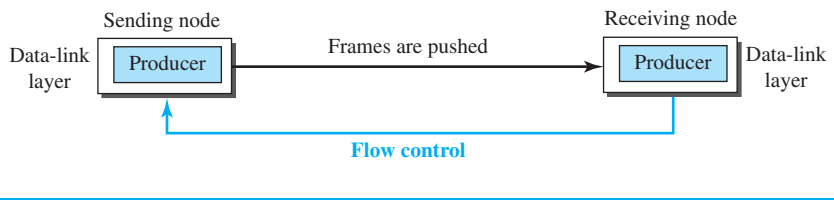
We briefly defined flow and error control in Chapter 9; we elaborate on these two issues here. One of the responsibilities of the data-link control sublayer is flow and error control at the data-link layer.

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.

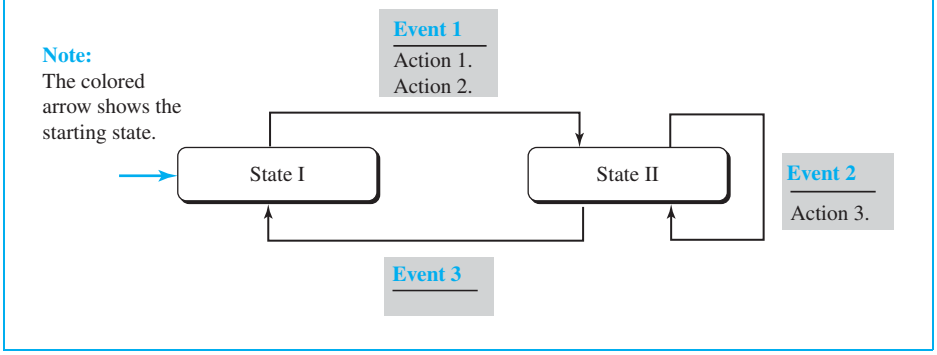
11.2 DATA-LINK LAYER PROTOCOLS

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.

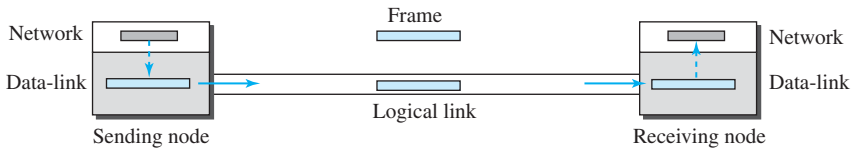
Figure 11.6 *Connectionless and connection-oriented service represented as FSMs*



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*

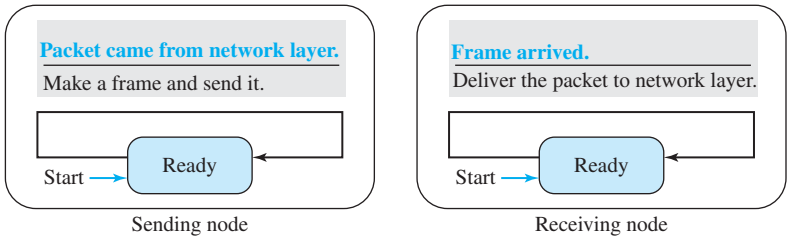


The data-link layer at the sender gets a packet from its network layer, makes a frame out of it, and sends the frame. The data-link layer at the receiver receives a frame from the link, extracts the packet from the frame, and delivers the packet to its network layer. The data-link layers of the sender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

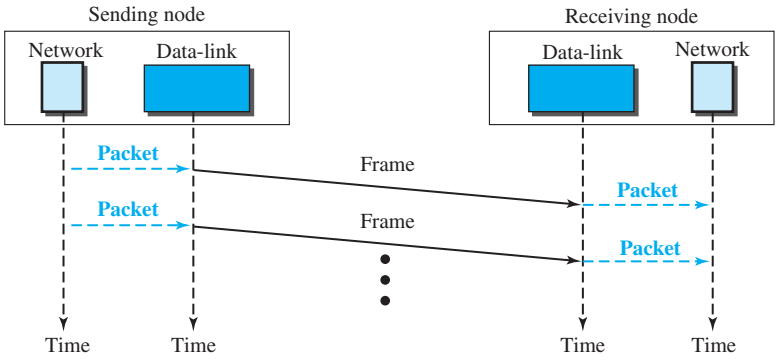
Figure 11.8 *FSMs for the simple protocol*



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*

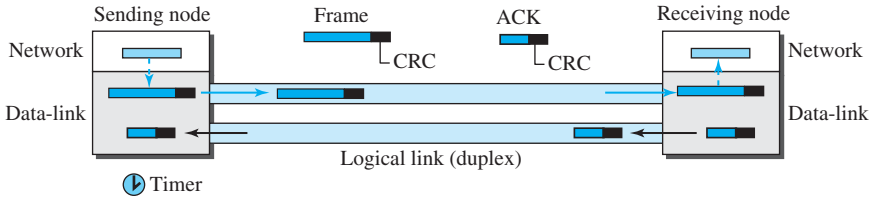


11.2.2 Stop-and-Wait Protocol

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.

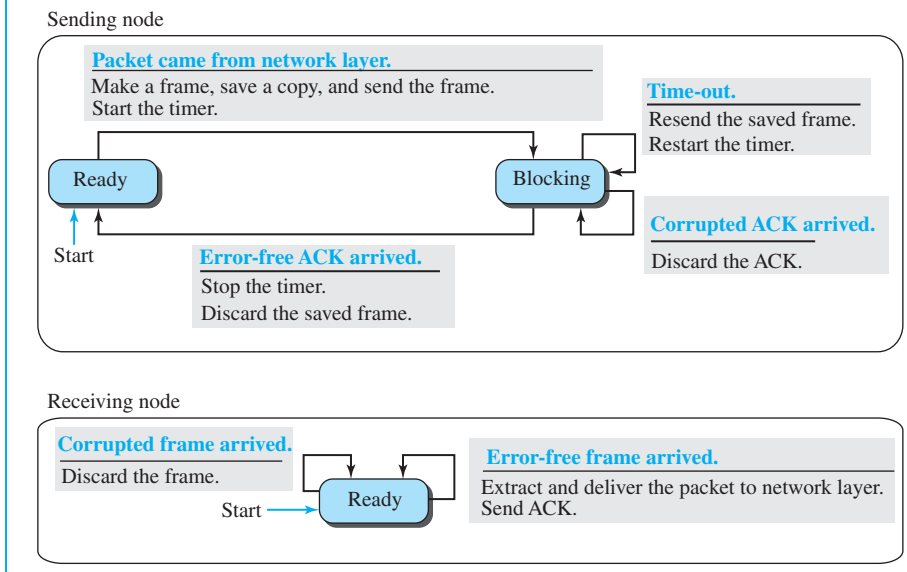
Figure 11.10 *Stop-and-Wait protocol*



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.

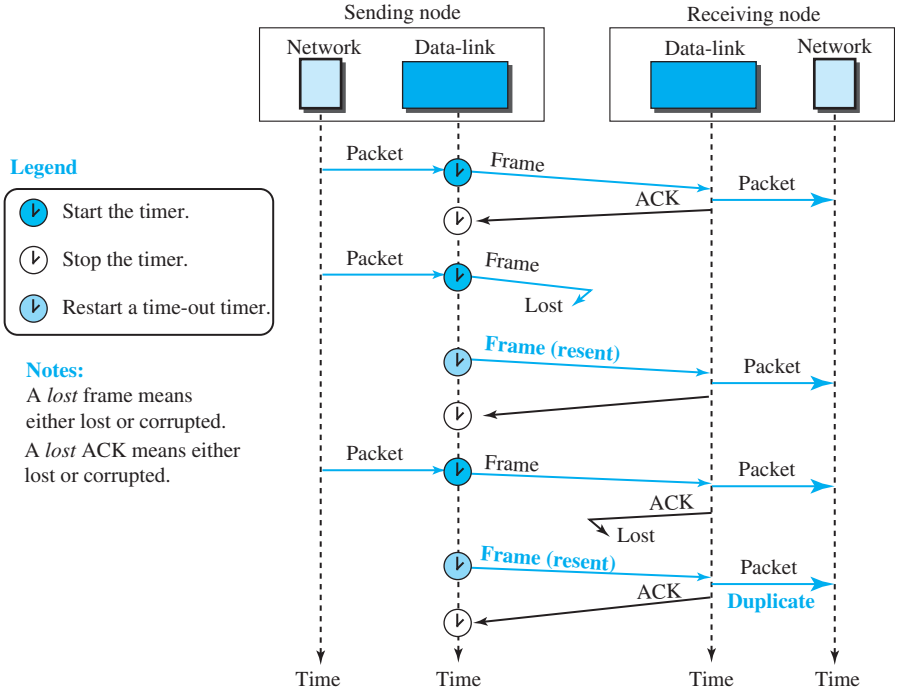
Example 11.4

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*



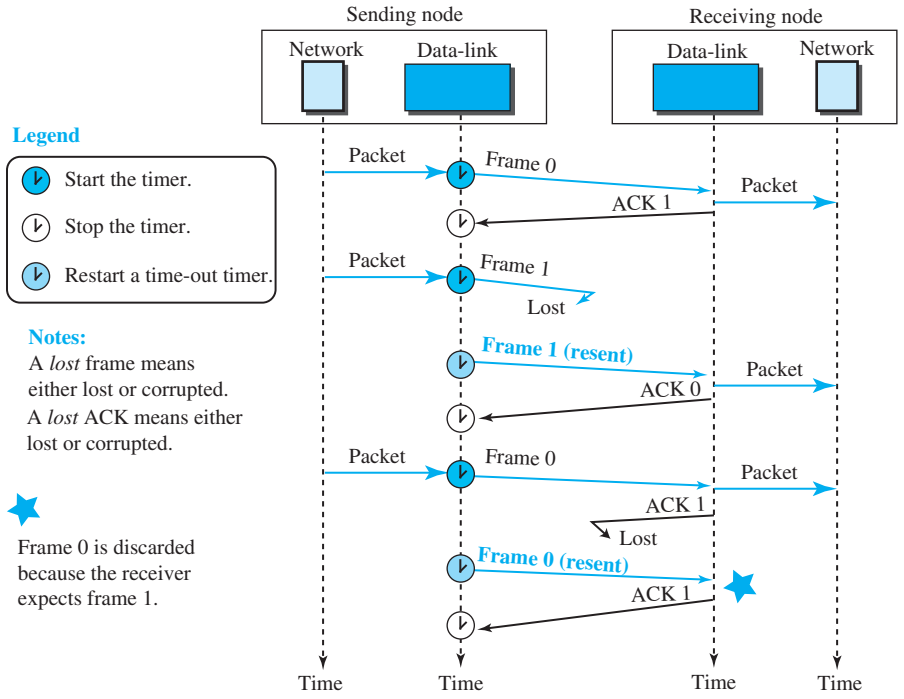
11.2.3 Piggybacking

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 (piggybacking). 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*

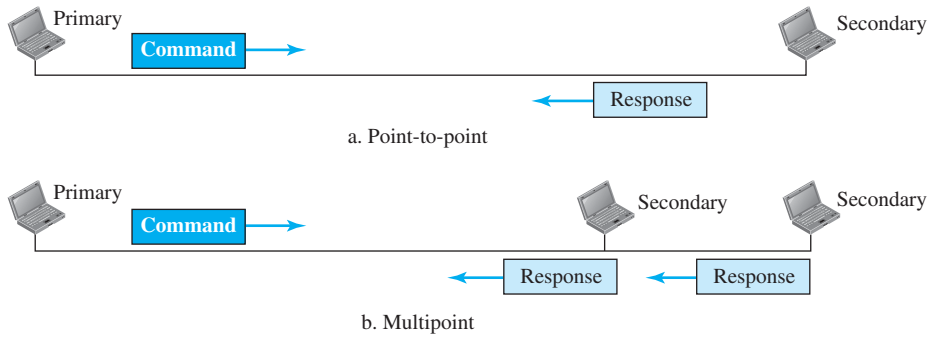
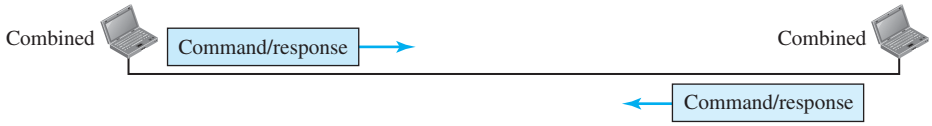
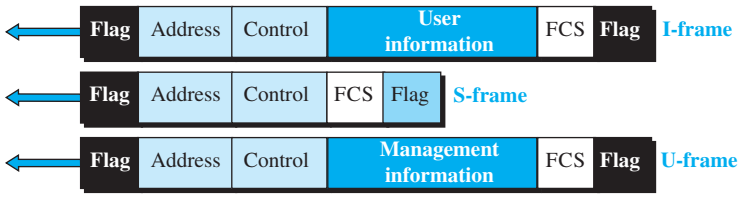


Figure 11.15 *Asynchronous balanced mode*



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*



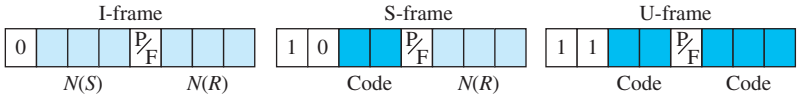
Let us now discuss the fields and their use in different frame types.

- ***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.

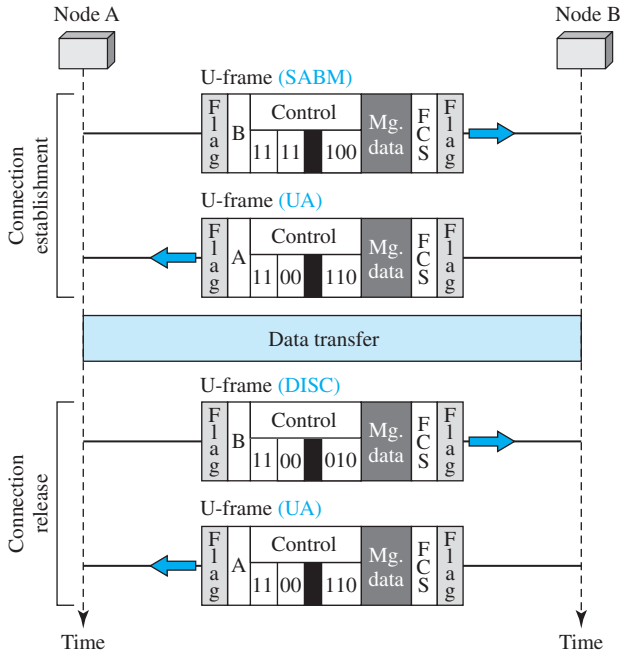
Example 11.5

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.

Figure 11.18 *Example of connection and disconnection*



11.4 POINT-TO-POINT PROTOCOL (PPP)

One of the most common protocols for point-to-point access is the **Point-to-Point Protocol (PPP).** Today, millions of Internet users who need to connect their home computers to the server of an Internet service provider use PPP. The majority of these users have a traditional modem; they are connected to the Internet through a telephone line, which provides the services of the physical layer. But to control and manage the transfer of data, there is a need for a point-to-point protocol at the data-link layer. PPP is by far the most common.

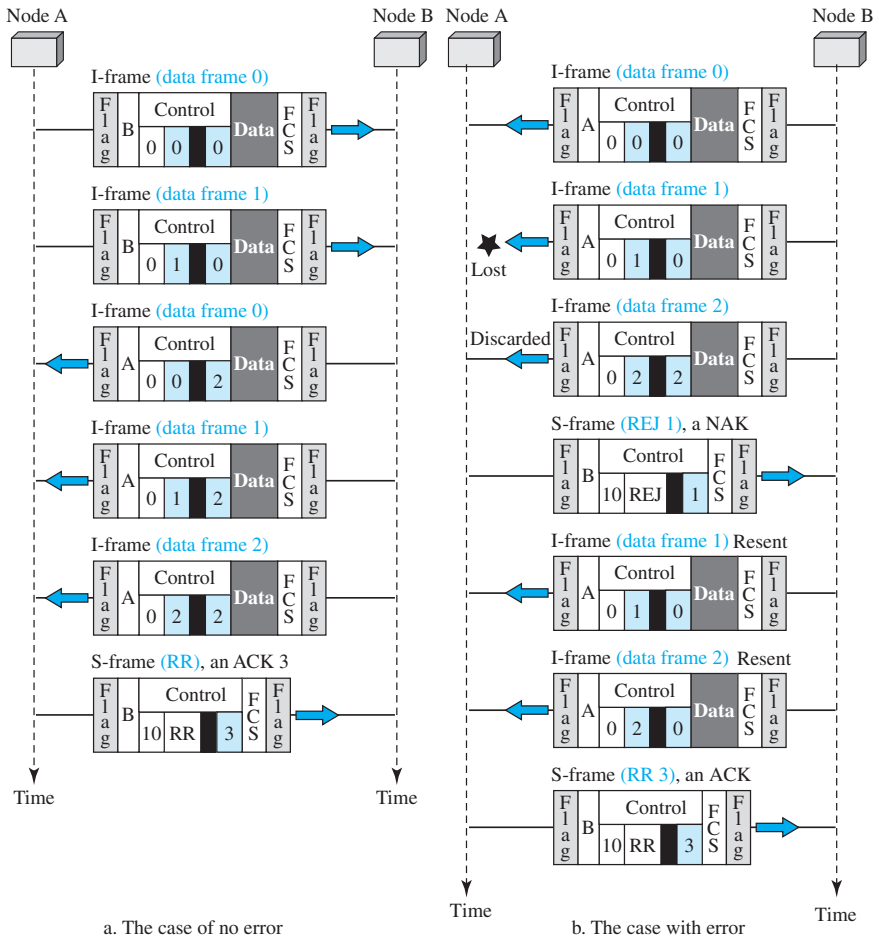
11.4.1 Services

The designers of PPP have included several services to make it suitable for a point-to-point protocol, but have ignored some traditional services to make it simple.

Services Provided by PPP

PPP defines the format of the frame to be exchanged between devices. It also defines how two devices can negotiate the establishment of the link and the exchange of data. PPP is designed to accept payloads from several network layers (not only IP). Authentication is also provided in the protocol, but it is optional. The new version of PPP, called *Multilink PPP,* provides connections over multiple links. One interesting feature of PPP is that it provides network address configuration. This is particularly useful when a home user needs a temporary network address to connect to the Internet.

Figure 11.19 *Example of piggybacking with and without error*



Services Not Provided by PPP

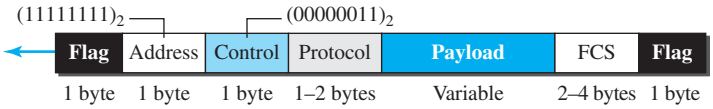
PPP does not provide flow control. A sender can send several frames one after another with no concern about overwhelming the receiver. PPP has a very simple mechanism for error control. A CRC field is used to detect errors. If the frame is corrupted, it is silently discarded; the upper-layer protocol needs to take care of the problem. Lack of error control and sequence numbering may cause a packet to be received out of order. PPP does not provide a sophisticated addressing mechanism to handle frames in a multipoint configuration.

11.4.2 Framing

PPP uses a character-oriented (or byte-oriented) frame. Figure 11.20 shows the format of a PPP frame. The description of each field follows:

- ***Flag.*** A PPP frame starts and ends with a 1-byte flag with the bit pattern 01111110.

Figure 11.20 *PPP frame format*



- ***Address.*** The address field in this protocol is a constant value and set to 11111111 (broadcast address).
- ***Control.*** This field is set to the constant value 00000011 (imitating unnumbered frames in HDLC). As we will discuss later, PPP does not provide any flow control. Error control is also limited to error detection.
- ***Protocol.*** The protocol field defines what is being carried in the data field: either user data or other information. This field is by default 2 bytes long, but the two parties can agree to use only 1 byte.
- ***Payload field.*** This field carries either the user data or other information that we will discuss shortly. The data field is a sequence of bytes with the default of a maximum of 1500 bytes; but this can be changed during negotiation. The data field is byte-stuffed if the flag byte pattern appears in this field. Because there is no field defining the size of the data field, padding is needed if the size is less than the maximum default value or the maximum negotiated value.
- ***FCS.*** The frame check sequence (FCS) is simply a 2-byte or 4-byte standard CRC.

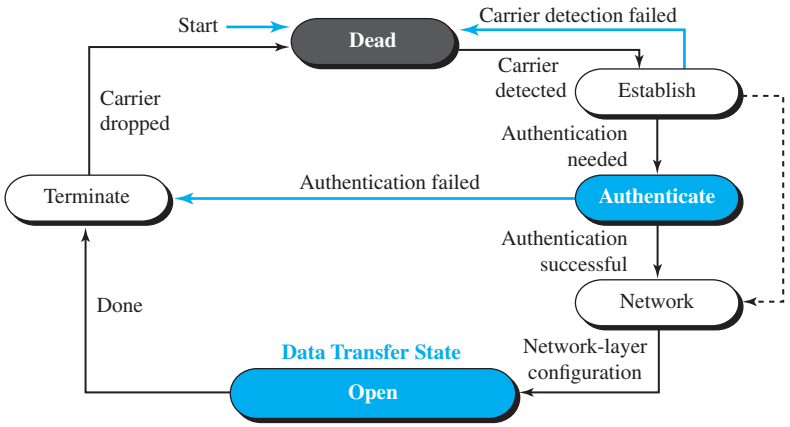
Byte Stuffing

Since PPP is a byte-oriented protocol, the flag in PPP is a byte that needs to be escaped whenever it appears in the data section of the frame. The escape byte is 01111101, which means that every time the flaglike pattern appears in the data, this extra byte is stuffed to tell the receiver that the next byte is not a flag. Obviously, the escape byte itself should be stuffed with another escape byte.

11.4.3 Transition Phases

A PPP connection goes through phases which can be shown in a *transition phase* diagram (see Figure 11.21). The transition diagram, which is an FSM, starts with the *dead* state. In this state, there is no active carrier (at the physical layer) and the line is quiet. When one of the two nodes starts the communication, the connection goes into the *establish* state. In this state, options are negotiated between the two parties. If the two parties agree that they need authentication (for example, if they do not know each other), then the system needs to do authentication (an extra step); otherwise, the parties can simply start communication. The link-control protocol packets, discussed shortly, are used for this purpose. Several packets may be exchanged here. Data transfer takes place in the *open* state. When a connection reaches this state, the exchange of data packets can be started. The connection remains in this state until one of the endpoints wants to terminate the connection. In this case, the system goes to the *terminate* state. The system remains in this state until the carrier (physical-layer signal) is dropped, which moves the system to the *dead* state again.

Figure 11.21 *Transition phases*



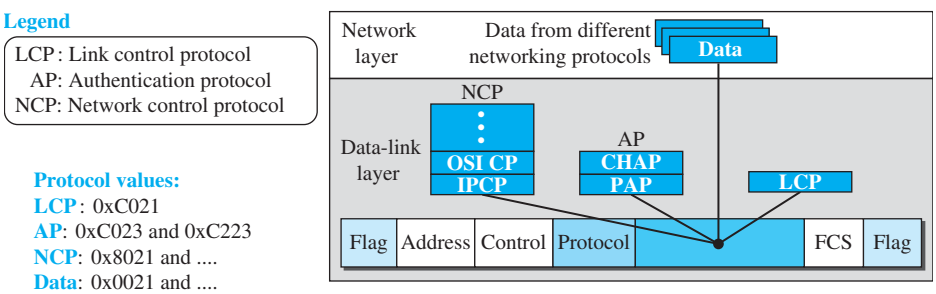
11.4.4 Multiplexing

Although PPP is a link-layer protocol, it uses another set of protocols to establish the link, authenticate the parties involved, and carry the network-layer data. Three sets of protocols are defined to make PPP powerful: the Link Control Protocol (LCP), two Authentication Protocols (APs), and several Network Control Protocols (NCPs). At any moment, a PPP packet can carry data from one of these protocols in its data field, as shown in Figure 11.22. Note that there are one LCP, two APs, and several NCPs. Data may also come from several different network layers.

Link Control Protocol

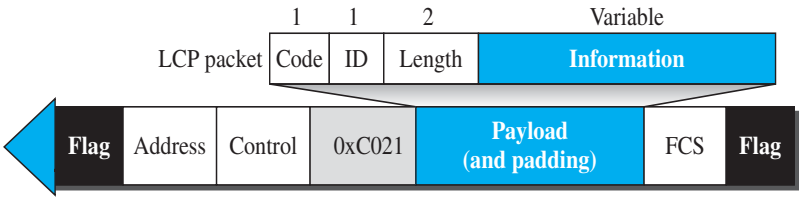
The **Link Control Protocol (LCP)** is responsible for establishing, maintaining, configuring, and terminating links. It also provides negotiation mechanisms to set options between the two endpoints. Both endpoints of the link must reach an agreement about the options before the link can be established. See Figure 11.21.

Figure 11.22 *Multiplexing in PPP*



All LCP packets are carried in the payload field of the PPP frame with the protocol field set to C021 in hexadecimal (see Figure 11.23).

Figure 11.23 *LCP packet encapsulated in a frame*



The code field defines the type of LCP packet. There are 11 types of packets, as shown in Table 11.1.

Table 11.1 *LCP packets*

Code	Packet Type	Description
0x01	Configure-request	Contains the list of proposed options and their values
0x02	Configure-ack	Accepts all options proposed
0x03	Configure-nak	Announces that some options are not acceptable
0x04	Configure-reject	Announces that some options are not recognized
0x05	Terminate-request	Request to shut down the line
0x06	Terminate-ack	Accept the shutdown request
0x07	Code-reject	Announces an unknown code
0x08	Protocol-reject	Announces an unknown protocol
0x09	Echo-request	A type of hello message to check if the other end is alive
0x0A	Echo-reply	The response to the echo-request message
0x0B	Discard-request	A request to discard the packet

There are three categories of packets. The first category, comprising the first four packet types, is used for link configuration during the establish phase. The second category, comprising packet types 5 and 6, is used for link termination during the termination phase. The last five packets are used for link monitoring and debugging.

The ID field holds a value that matches a request with a reply. One endpoint inserts a value in this field, which will be copied into the reply packet. The length field defines the length of the entire LCP packet. The information field contains information, such as options, needed for some LCP packets.

There are many options that can be negotiated between the two endpoints. Options are inserted in the information field of the configuration packets. In this case, the

information field is divided into three fields: option type, option length, and option data. We list some of the most common options in Table 11.2.

Table 11.2 *Common options*

Option	*Default*
Maximum receive unit (payload field size)	1500
Authentication protocol	None
Protocol field compression	Off
Address and control field compression	Off

Authentication Protocols

Authentication plays a very important role in PPP because PPP is designed for use over dial-up links where verification of user identity is necessary. *Authentication* means validating the identity of a user who needs to access a set of resources. PPP has created two protocols for authentication: Password Authentication Protocol and Challenge Handshake Authentication Protocol. Note that these protocols are used during the authentication phase.

PAP

The **Password Authentication Protocol (PAP)** is a simple authentication procedure with a two-step process:

a. The user who wants to access a system sends an authentication identification (usually the user name) and a password.

b. The system checks the validity of the identification and password and either accepts or denies connection.

Figure 11.24 shows the three types of packets used by PAP and how they are actually exchanged. When a PPP frame is carrying any PAP packets, the value of the protocol field is 0xC023. The three PAP packets are authenticate-request, authenticate-ack, and authenticate-nak. The first packet is used by the user to send the user name and password. The second is used by the system to allow access. The third is used by the system to deny access.

CHAP

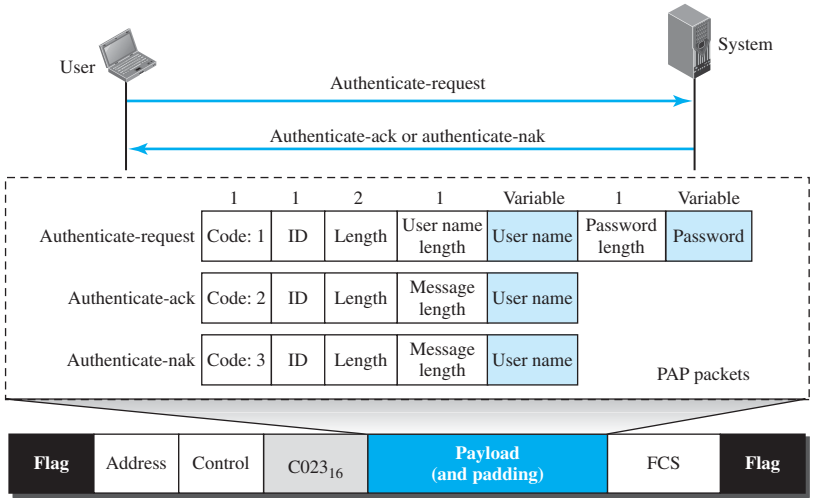
The **Challenge Handshake Authentication Protocol (CHAP)** is a three-way handshaking authentication protocol that provides greater security than PAP. In this method, the password is kept secret; it is never sent online.

a. The system sends the user a challenge packet containing a challenge value, usually a few bytes.

b. The user applies a predefined function that takes the challenge value and the user's own password and creates a result. The user sends the result in the response packet to the system.

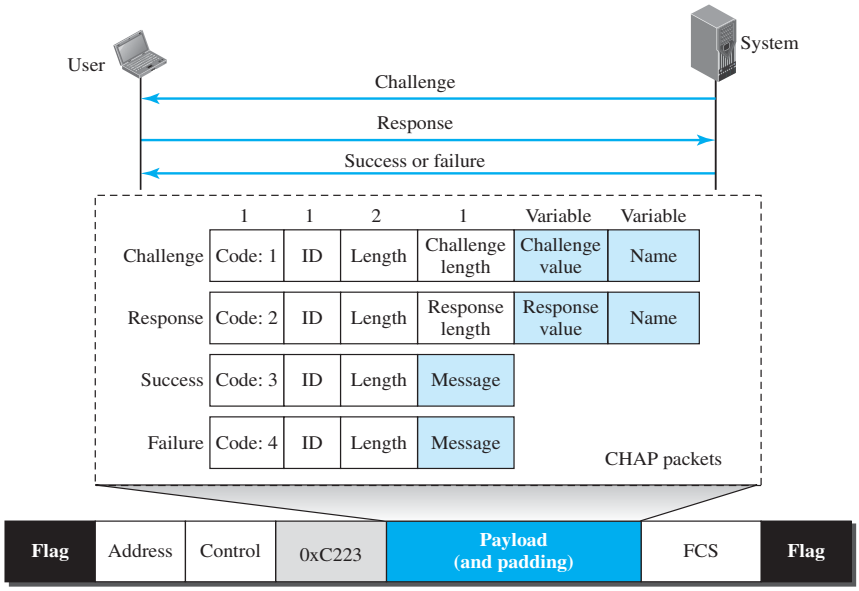
c. The system does the same. It applies the same function to the password of the user (known to the system) and the challenge value to create a result. If the

Figure 11.24 *PAP packets encapsulated in a PPP frame*



result created is the same as the result sent in the response packet, access is granted; otherwise, it is denied. CHAP is more secure than PAP, especially if the system continuously changes the challenge value. Even if the intruder learns the challenge value and the result, the password is still secret. Figure 11.25 shows the packets and how they are used.

Figure 11.25 *CHAP packets encapsulated in a PPP frame*



CHAP packets are encapsulated in the PPP frame with the protocol value C223 in hexadecimal. There are four CHAP packets: challenge, response, success, and failure. The first packet is used by the system to send the challenge value. The second is used by the user to return the result of the calculation. The third is used by the system to allow access to the system. The fourth is used by the system to deny access to the system.

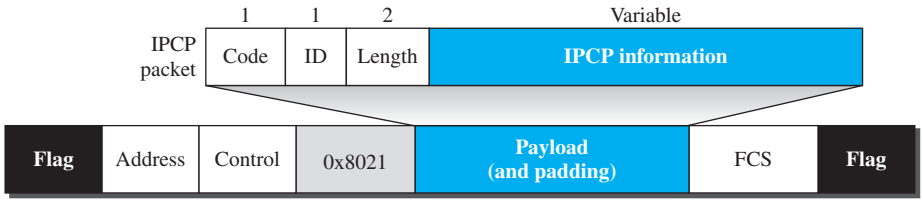
Network Control Protocols

PPP is a multiple-network-layer protocol. It can carry a network-layer data packet from protocols defined by the Internet, OSI, Xerox, DECnet, AppleTalk, Novel, and so on. To do this, PPP has defined a specific Network Control Protocol for each network protocol. For example, IPCP (Internet Protocol Control Protocol) configures the link for carrying IP data packets. Xerox CP does the same for the Xerox protocol data packets, and so on. Note that none of the NCP packets carry network-layer data; they just configure the link at the network layer for the incoming data.

IPCP

One NCP protocol is the **Internet Protocol Control Protocol (IPCP).** This protocol configures the link used to carry IP packets in the Internet. IPCP is especially of interest to us. The format of an IPCP packet is shown in Figure 11.26. Note that the value of the protocol field in hexadecimal is 8021.

Figure 11.26 *IPCP packet encapsulated in PPP frame*



IPCP defines seven packets, distinguished by their code values, as shown in Table 11.3.

Table 11.3 *Code value for IPCP packets*

Code	*IPCP Packet*
0x01	Configure-request
0x02	Configure-ack
0x03	Configure-nak
0x04	Configure-reject
0x05	Terminate-request
0x06	Terminate-ack
0x07	Code-reject

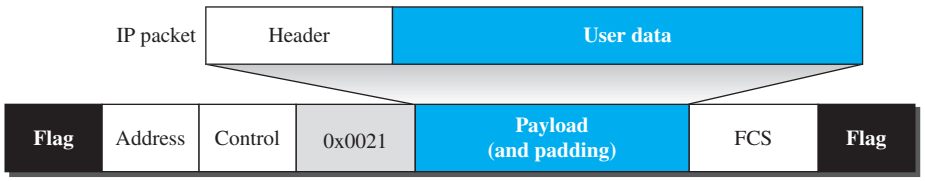
Other Protocols

There are other NCP protocols for other network-layer protocols. The OSI Network Layer Control Protocol has a protocol field value of 8023; the Xerox NS IDP Control Protocol has a protocol field value of 8025; and so on.

Data from the Network Layer

After the network-layer configuration is completed by one of the NCP protocols, the users can exchange data packets from the network layer. Here again, there are different protocol fields for different network layers. For example, if PPP is carrying data from the IP network layer, the field value is 0021 (note that the three rightmost digits are the same as for IPCP). If PPP is carrying data from the OSI network layer, the value of the protocol field is 0023, and so on. Figure 11.27 shows the frame for IP.

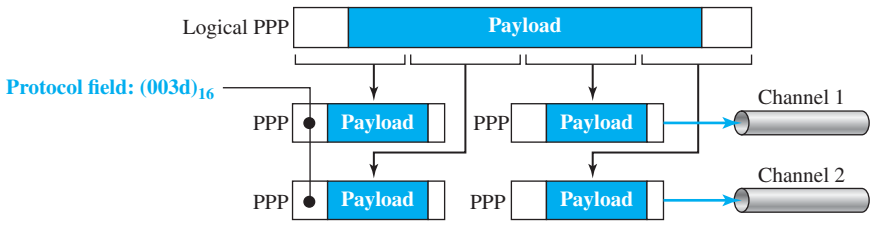
Figure 11.27 *IP datagram encapsulated in a PPP frame*



Multilink PPP

PPP was originally designed for a single-channel point-to-point physical link. The availability of multiple channels in a single point-to-point link motivated the development of Multilink PPP. In this case, a logical PPP frame is divided into several actual PPP frames. A segment of the logical frame is carried in the payload of an actual PPP frame, as shown in Figure 11.28. To show that the actual PPP frame is carrying a fragment of a logical PPP frame, the protocol field is set to $(003d)_{16}$. This new development adds complexity. For example, a sequence number needs to be added to the actual PPP frame to show a fragment's position in the logical frame.

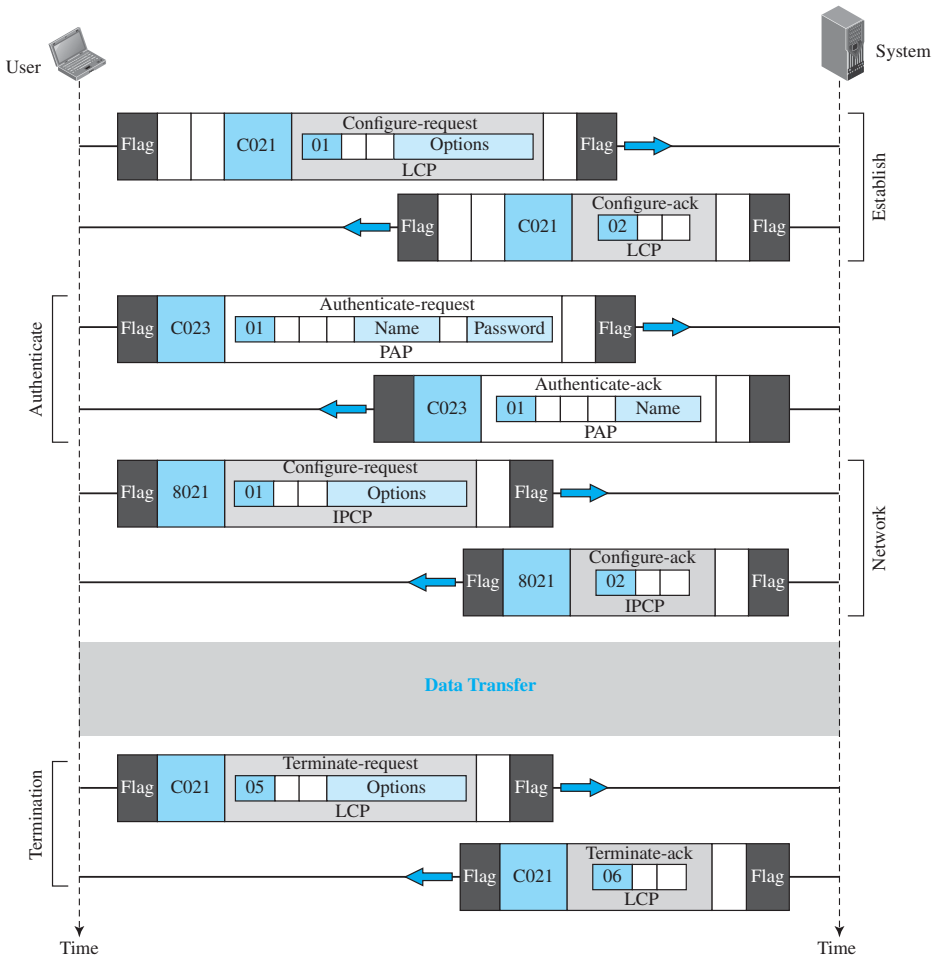
Figure 11.28 *Multilink PPP*



Example 11.7

Let us go through the phases followed by a network layer packet as it is transmitted through a PPP connection. Figure 11.29 shows the steps. For simplicity, we assume unidirectional movement of data from the user site to the system site (such as sending an e-mail through an ISP).

Figure 11.29 *An example*



The first two frames show link establishment. We have chosen two options (not shown in the figure): using PAP for authentication and suppressing the address control fields. Frames 3 and 4 are for authentication. Frames 5 and 6 establish the network layer connection using IPCP.

The next several frames show that some IP packets are encapsulated in the PPP frame. The system (receiver) may have been running several network layer protocols, but it knows that the incoming data must be delivered to the IP protocol because the NCP protocol used before the data transfer was IPCP.

After data transfer, the user then terminates the data-link connection, which is acknowledged by the system. Of course the user or the system could have chosen to terminate the network-layer IPCP and keep the data-link layer running if it wanted to run another NCP protocol.

11.5 END-CHAPTER MATERIALS

11.5.1 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].

11.5.2 Key Terms

acknowledgment number
bit stuffing
byte stuffing
Challenge Handshake Authentication Protocol (CHAP)
data link control (DLC)
finite state machine (FSM)
flag
High-level Data Link Control (HDLC)
Internet Protocol Control Protocol (IPCP)
Link Control Protocol (LCP)
Password Authentication Protocol (PAP)
piggybacking
Point-to-Point Protocol (PPP)
sequence number
Simple Protocol
Stop-and-Wait Protocol

11.5.3 Summary

Data link control deals with the design and procedures for communication between two adjacent nodes: node-to-node communication. Framing in the data-link layer separates one packet from another. In fixed-size framing, there is no need for defining the boundaries of frames; in variable-size framing, we need a delimiter (flag) to define the boundary of two frames. Variable-size framing uses two categories of protocols: byte-oriented (or character-oriented) and bit-oriented. In a byte-oriented protocol, the data section of a frame is a sequence of bytes; in a bit-oriented protocol, the data section of a frame is a sequence of bits. In byte-oriented protocols, we use byte stuffing; in bit-oriented protocols, we use bit stuffing.

Another duty of DLC is flow and error control. At the data-link layer, flow control means creating a balance between the frames sent by a node and the frames that can be handled by the next node. Error control at the data-link layer is normally implemented very simply. Corrupted frames are silently discarded; uncorrupted frames are accepted with or without sending acknowledgments to the sender.

A DLC protocol can be either connectionless or connection-oriented. In a connectionless protocol, frames are sent from one node to the next without any relationship between the frames; each frame is independent. In a connection-oriented protocol, a logical connection should first be established between the two nodes before sending the data frames. After all related frames are transmitted, the logical connection is terminated.

Data-link protocols have been designed to handle communication between two nodes. We discussed two protocols in this chapter. In the Simple Protocol, there is no flow and error control. In the Stop-and-Wait Protocol, there are both flow and error controls, but communication is a frame at a time.

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. It is the basis of many protocols in practice today. HDLC defines three types of frames: information frames, supervisory frames, and unnumbered frames. The informational frames are used to carry data frames. Supervisory frames are used only to transport control information for flow and error control. Unnumbered frames are reserved for system management and provide connection-oriented service.

One of the most common protocols for point-to-point access is the Point-to-Point Protocol (PPP). PPP uses only one type of frame, but allows multiplexing of different payloads to achieve a kind of connection-oriented service authentication. Encapsulating different packets in a frame allows PPP to move to different states to provide necessary services.

11.6 PRACTICE SET

11.6.1 Quizzes

A set of interactive quizzes for this chapter can be found on the book website. It is strongly recommended that the student take the quizzes to check his/her understanding of the materials before continuing with the practice set.

11.6.2 Questions

Q11-1. Define *framing* and give the reason it is needed.

Q11-2. Explain why flags are needed when we use variable-size frames.

Q11-3. Assume a new character-oriented protocol is using the 16-bit Unicode as the character set. What should the size of the flag be in this protocol?

Q11-4. Compare and contrast byte-oriented and bit-oriented protocols.

Q11-5. Compare and contrast byte-stuffing and bit-stuffing.

Q11-6. In a byte-oriented protocol, should we first unstuff the extra bytes and then remove the flags or reverse the process?

Q11-7. In a bit-oriented protocol, should we first unstuff the extra bits and then remove the flags or reverse the process?

Q11-8. Compare and contrast flow control and error control.

Q11-9. In the Stop-and-Wait Protocol, assume that the sender has only one slot in which to keep the frame to send or the copy of the sent frame. What happens if the network layer delivers a packet to the data-link layer at this moment?

Q11-10. In Example 11.3 (Figure 11.12) how many frames are in transit at the same time?

Q11-11. In Example 11.4 (Figure 11.13) how many frames are in transit at the same time?

Q11-12. In the traditional Ethernet protocol (Chapter 13), the frames are sent with the CRC. If the frame is corrupted, the receiving node just discards it. Is this an example of a Simple Protocol or the Stop-and-Wait Protocol? Explain.

Q11-13. In Figure 11.11, do the ready and blocking states use the same timer? Explain.

Q11-14. Explain why there is no need for CRC in the Simple Protocol.

Q11-15. In Figure 11.9, we show the packet path as a horizontal line, but the frame path as a diagonal line. Can you explain the reason?

Q11-16. In Figure 11.12, explain why we need a timer at the sending site, but none at the receiving site.

Q11-17. Does the duplex communication in Figure 11.10 necessarily mean we need two separate media between the two nodes? Explain.

Q11-18. Define *piggybacking* and its benefit.

Q11-19. In Figure 11.16, which frame type can be used for acknowledgment?

Q11-20. Compare Figure 11.6 and Figure 11.21. If both are FSMs, why are there no event/action pairs in the second?

Q11-21. In PPP, we normally talk about *user* and *system* instead of *sending and receiving nodes;* explain the reason.

Q11-22. Compare and contrast HDLC with PPP.

Q11-23. Compare the flag byte and the escape byte in PPP. Are they are the same? Explain.

Q11-24. In Figure 11.20, explain why we need only one address field. Explain why the address is set to the predefined value of $(11111111)_2$.

11.6.3 Problems

P11-1. Byte-stuff the following frame payload in which E is the escape byte, F is the flag byte, and D is a data byte other than an escape or a flag character.

D	E	D	D	F	D	D	E	E	D	F	D

P11-2. Unstuff the following frame payload in which E is the escape byte, F is the flag byte, and D is a data byte other than an escape or a flag character.

E	E	D	E	F	D	D	E	F	E	E	D	D	D	

P11-3. Bit-stuff the following frame payload:

```
000111111100111110100011111111110000111
```

P11-4. Unstuff the following frame payload:

```
0001111100000111110111010011101111000001111
```

P11-5. Assume we change the Stop-and-Wait Protocol to include a NAK (negative feedback), which is used only when a corrupted frame arrives and is discarded. Redraw Figure 11.9 to show this change.

P11-6. In Example 11.4 (Figure 11.13), assume the round trip time for a frame is 40 milliseconds. Explain what will happen if we set the time-out in each of the following cases.

a. 35 milliseconds **b.** 45 milliseconds **c.** 40 milliseconds

P11-7. Redraw Figure 11.12 using the following scenario:

a. The first frame is sent and acknowledged.

b. The second frame is sent and acknowledged, but the acknowledgment is lost.

c. The second frame is resent, but it is timed-out.

d. The second frame is resent and acknowledged.

P11-8. Redraw Figure 11.2 using the following scenario:

a. Frame 0 is sent, but lost.

b. Frame 0 is resent and acknowledged.

c. Frame 1 is sent and acknowledged, but the acknowledgment is lost.

d. Frame 1 is resent and acknowledged.

P11-9. In Figure 11.11, show what happens in each of the following cases:

a. The sender is at the ready state and an error-free ACK arrives.

b. The sender is at the blocking state and a time-out occurs.

c. The sender is at the ready state and a time-out occurs.

P11-10. In Figure 11.11, show what happens in each of the following cases:

a. The receiver is in the ready state and a packet comes from the network layer.

b. The receiver is in the ready state and a corrupted frame arrives.

c. The receiver is in the ready state and an acknowledgment arrives.

P11-11. Using the following specifications, draw a finite state machine with three states (I, II, and III), five events, and six actions:

a. If the machine is in state I, two events can occur. If event 1 occurs, the machine moves to state II. If event 2 occurs, the machine performs actions 1 and 2 and moves to state III.

b. If the machine is in state II, two events can occur. If event 3 occurs, the machine remains in state II. If event 4 occurs, the machine moves to state III.

c. If the machine is in state III, three events can occur. If event 2 occurs, the machine remains in state III. If event 3 occurs, the machine performs actions 1, 2, 4, and 5 moves to state II. If event 5 occurs, the machine performs actions 1, 2, and 6 and moves to state I.

P11-12. Using the following specifications, draw a finite state machine with three states (I, II, and III), six events, and four actions:

a. If the machine is in state I, two events can occur. If event 1 occurs, the machine moves to state III. If event 3 occurs, the machine performs actions 2 and 4 and moves to state II.

b. If the machine is in state II, two events can occur. If event 4 occurs, the machine remains in state II. If event 6 occurs, the machine performs actions 1 and 2 and moves to state III.

c. If the machine is in state III, three events can occur. If event 2 occurs, the machine remains in state III. If event 6 occurs, the machine performs actions 2, 3, 4, and 5 moves to state I. If event 4 occurs, the machine performs actions 1 and 2 and moves to state I.

P11-13. Redraw Figure 11.11 using a variable to hold the one-bit sequence number and a variable to hold the one-bit acknowledgment number.

P11-14. Redraw Figure 11.10 using piggybacking.

P11-15. Assume PPP is in the established phase; show payload encapsulated in the frame.

P11-16. Redraw Figure 11.21 with the system not using authentication.

P11-17. Assume PPP is in the authentication phase, show payload exchanged between the nodes if PPP is using

a. PAP

b. CHAP

P11-18. Assume the only computer in the residence uses PPP to communicate with the ISP. If the user sends 10 network-layer packets to ISP, how many frames are exchanged in each of the following cases:

a. Using no authentication?

b. Using PAP for authentication?

c. Using CHAP for authentication?

11.7 SIMULATION EXPERIMENTS

11.7.1 Applets

We have created some Java applets to show some of the main concepts discussed in this chapter. It is strongly recommended that the students activate these applets on the book website and carefully examine the protocols in action.

11.8 PROGRAMMING ASSIGNMENTS

For each of the following assignments, write a program in the programming language you are familiar with.

Prg11-1. Write and test a program that simulates the byte stuffing and byte unstuffing as shown in Figure 11.2.

Prg11-2. Write and test a program that simulates the bit stuffing and bit unstuffing as shown in Figure 11.4.

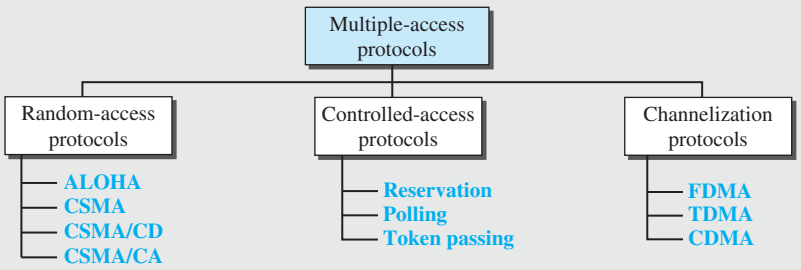
https://hemanthrajhemu.github.io

CHAPTER 12

Media Access Control (MAC)

When nodes or stations are connected and use a common link, called a *multipoint* or *broadcast link,* we need a multiple-access protocol to coordinate access to the link. The problem of controlling the access to the medium is similar to the rules of speaking in an assembly. The procedures guarantee that the right to speak is upheld and ensure that two people do not speak at the same time, do not interrupt each other, do not monopolize the discussion, and so on. Many protocols have been devised to handle access to a shared link. All of these protocols belong to a sublayer in the data-link layer called *media access control (MAC).* We categorize them into three groups, as shown in Figure 12.1.

Figure 12.1 *Taxonomy of multiple-access protocols*



This chapter is divided into three sections:

- The first section discusses random-access protocols. Four protocols, ALOHA, CSMA, CSMA/CD, and CSMA/CA, are described in this section. These protocols are mostly used in LANs and WANs, which we discuss in future chapters.
- The second section discusses controlled-access protocols. Three protocols, reservation, polling, and token-passing, are described in this section. Some of these protocols are used in LANs, but others have some historical value.
- The third section discusses channelization protocols. Three protocols, FDMA, TDMA, and CDMA are described in this section. These protocols are used in cellular telephony, which we discuss in Chapter 16.

12.1 RANDOM ACCESS

In **random-access** or **contention** methods, no station is superior to another station and none is assigned control over another. At each instance, a station that has data to send uses a procedure defined by the protocol to make a decision on whether or not to send. This decision depends on the state of the medium (idle or busy). In other words, each station can transmit when it desires on the condition that it follows the predefined procedure, including testing the state of the medium.

Two features give this method its name. First, there is no scheduled time for a station to transmit. Transmission is random among the stations. That is why these methods are called *random access*. Second, no rules specify which station should send next. Stations compete with one another to access the medium. That is why these methods are also called *contention* methods.

In a random-access method, each station has the right to the medium without being controlled by any other station. However, if more than one station tries to send, there is an access conflict—***collision***—and the frames will be either destroyed or modified. To avoid access conflict or to resolve it when it happens, each station follows a procedure that answers the following questions:

- When can the station access the medium?
- What can the station do if the medium is busy?
- How can the station determine the success or failure of the transmission?
- What can the station do if there is an access conflict?

The random-access methods we study in this chapter have evolved from a very interesting protocol known as *ALOHA*, which used a very simple procedure called ***multiple access (MA).*** The method was improved with the addition of a procedure that forces the station to sense the medium before transmitting. This was called *carrier sense multiple access* (*CSMA*). This method later evolved into two parallel methods: *carrier sense multiple access with collision detection (CSMA/CD),* which tells the station what to do when a collision is detected, and *carrier sense multiple access with collision avoidance (CSMA/CA),* which tries to avoid the collision.

12.1.1 ALOHA

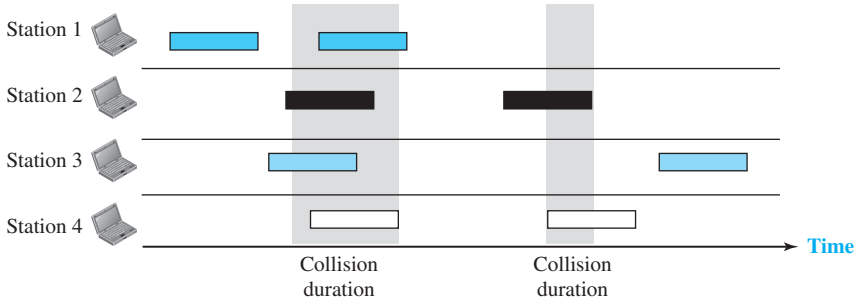
ALOHA, the earliest random access method, was developed at the University of Hawaii in early 1970. It was designed for a radio (wireless) LAN, but it can be used on any shared medium.

It is obvious that there are potential collisions in this arrangement. The medium is shared between the stations. When a station sends data, another station may attempt to do so at the same time. The data from the two stations collide and become garbled.

Pure ALOHA

The original ALOHA protocol is called ***pure ALOHA.*** This is a simple but elegant protocol. The idea is that each station sends a frame whenever it has a frame to send (multiple access). However, since there is only one channel to share, there is the possibility of collision between frames from different stations. Figure 12.2 shows an example of frame collisions in pure ALOHA.

Figure 12.2 *Frames in a pure ALOHA network*



There are four stations (unrealistic assumption) that contend with one another for access to the shared channel. The figure shows that each station sends two frames; there are a total of eight frames on the shared medium. Some of these frames collide because multiple frames are in contention for the shared channel. Figure 12.2 shows that only two frames survive: one frame from station 1 and one frame from station 3. We need to mention that even if one bit of a frame coexists on the channel with one bit from another frame, there is a collision and both will be destroyed. It is obvious that we need to resend the frames that have been destroyed during transmission.

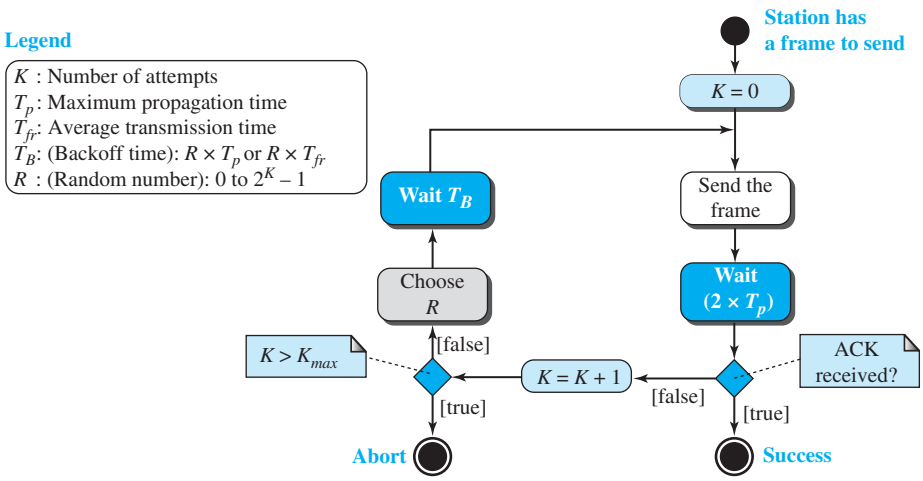
The pure ALOHA protocol relies on acknowledgments from the receiver. When a station sends a frame, it expects the receiver to send an acknowledgment. If the acknowledgment does not arrive after a time-out period, the station assumes that the frame (or the acknowledgment) has been destroyed and resends the frame.

A collision involves two or more stations. If all these stations try to resend their frames after the time-out, the frames will collide again. Pure ALOHA dictates that when the time-out period passes, each station waits a random amount of time before resending its frame. The randomness will help avoid more collisions. We call this time the *backoff time* T_B.

Pure ALOHA has a second method to prevent congesting the channel with retransmitted frames. After a maximum number of retransmission attempts K_{max}, a station must give up and try later. Figure 12.3 shows the procedure for pure ALOHA based on the above strategy.

The time-out period is equal to the maximum possible round-trip propagation delay, which is twice the amount of time required to send a frame between the two most widely separated stations ($2 \times T_p$). The backoff time T_B is a random value that normally depends on K (the number of attempted unsuccessful transmissions). The formula for T_B depends on the implementation. One common formula is the ***binary exponential backoff.*** In this method, for each retransmission, a multiplier $R = 0$ to $2^K - 1$ is randomly chosen and multiplied by T_p (maximum propagation time) or T_{fr} (the average time required to send out a frame) to find T_B. Note that in this procedure, the range of the random numbers increases after each collision. The value of K_{max} is usually chosen as 15.

Figure 12.3 *Procedure for pure ALOHA protocol*



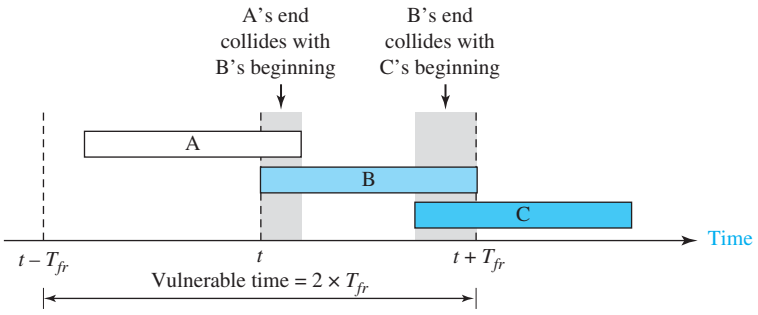
Example 12.1

The stations on a wireless ALOHA network are a maximum of 600 km apart. If we assume that signals propagate at 3×10^8 m/s, we find $T_p = (600 \times 10^3) / (3 \times 10^8) = 2$ ms. For $K = 2$, the range of R is {0, 1, 2, 3}. This means that T_B can be 0, 2, 4, or 6 ms, based on the outcome of the random variable R.

Vulnerable time

Let us find the ***vulnerable time,*** the length of time in which there is a possibility of collision. We assume that the stations send fixed-length frames with each frame taking T_{fr} seconds to send. Figure 12.4 shows the vulnerable time for station B.

Figure 12.4 *Vulnerable time for pure ALOHA protocol*



Station B starts to send a frame at time t. Now imagine station A has started to send its frame after $t - T_{fr}$. This leads to a collision between the frames from station B and

station A. On the other hand, suppose that station C starts to send a frame before time $t + T_{fr}$. Here, there is also a collision between frames from station B and station C.

Looking at Figure 12.4, we see that the vulnerable time during which a collision may occur in pure ALOHA is 2 times the frame transmission time.

$$\textbf{Pure ALOHA vulnerable time} = 2 \times T_{fr}$$

Example 12.2

A pure ALOHA network transmits 200-bit frames on a shared channel of 200 kbps. What is the requirement to make this frame collision-free?

Solution

Average frame transmission time T_{fr} is 200 bits/200 kbps or 1 ms. The vulnerable time is 2×1 ms = 2 ms. This means no station should send later than 1 ms before this station starts transmission and no station should start sending during the period (1 ms) that this station is sending.

Throughput

Let us call G the average number of frames generated by the system during one frame transmission time. Then it can be proven that the average number of successfully transmitted frames for pure ALOHA is $S = G \times e^{-2G}$. The maximum throughput S_{max} is 0.184, for $G = 1/2$. (We can find it by setting the derivative of S with respect to G to 0; see Exercises.) In other words, if one-half a frame is generated during one frame transmission time (one frame during two frame transmission times), then 18.4 percent of these frames reach their destination successfully. We expect $G = 1/2$ to produce the maximum throughput because the vulnerable time is 2 times the frame transmission time. Therefore, if a station generates only one frame in this vulnerable time (and no other stations generate a frame during this time), the frame will reach its destination successfully.

The throughput for pure ALOHA is $S = G \times e^{-2G}$.
The maximum throughput $S_{max} = 1/(2e) = 0.184$ when $G = (1/2)$.

Example 12.3

A pure ALOHA network transmits 200-bit frames on a shared channel of 200 kbps. What is the throughput if the system (all stations together) produces

a. 1000 frames per second?
b. 500 frames per second?
c. 250 frames per second?

Solution

The frame transmission time is 200/200 kbps or 1 ms.

a. If the system creates 1000 frames per second, or 1 frame per millisecond, then $G = 1$. In this case $S = G \times e^{-2G} = 0.135$ (13.5 percent). This means that the throughput is $1000 \times 0.135 = 135$ frames. Only 135 frames out of 1000 will probably survive.

b. If the system creates 500 frames per second, or 1/2 frames per millisecond, then $G = 1/2$. In this case $S = G \times e^{-2G} = 0.184$ (18.4 percent). This means that the throughput is $500 \times 0.184 = 92$ and that only 92 frames out of 500 will probably survive. Note that this is the *maximum* throughput case, percentagewise.

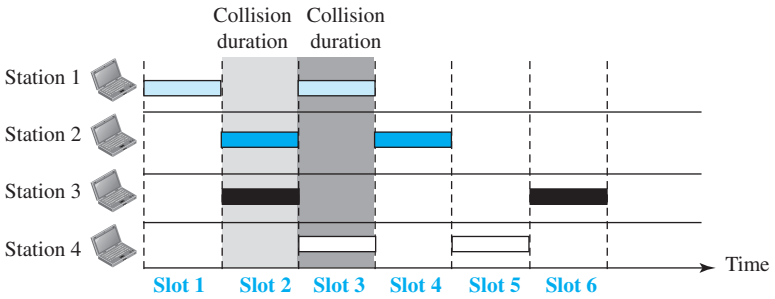
c. If the system creates 250 frames per second, or 1/4 frames per millisecond, then $G = 1/4$. In this case $S = G \times e^{-2G} = 0.152$ (15.2 percent). This means that the throughput is $250 \times 0.152 = 38$. Only 38 frames out of 250 will probably survive.

Slotted ALOHA

Pure ALOHA has a vulnerable time of $2 \times T_{fr}$. This is so because there is no rule that defines when the station can send. A station may send soon after another station has started or just before another station has finished. Slotted ALOHA was invented to improve the efficiency of pure ALOHA.

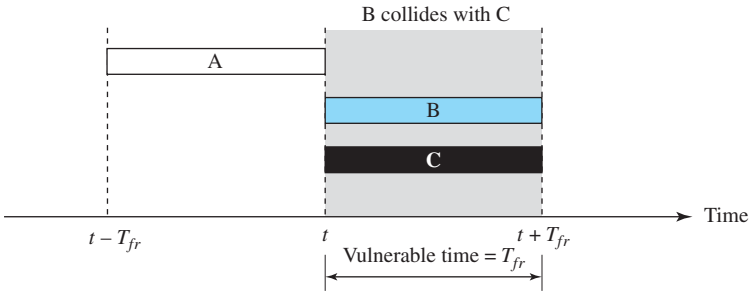
In **slotted ALOHA** we divide the time into slots of T_{fr} seconds and force the station to send only at the beginning of the time slot. Figure 12.5 shows an example of frame collisions in slotted ALOHA.

Figure 12.5 *Frames in a slotted ALOHA network*



Because a station is allowed to send only at the beginning of the synchronized time slot, if a station misses this moment, it must wait until the beginning of the next time slot. This means that the station which started at the beginning of this slot has already finished sending its frame. Of course, there is still the possibility of collision if two stations try to send at the beginning of the same time slot. However, the vulnerable time is now reduced to one-half, equal to T_{fr}. Figure 12.6 shows the situation.

Figure 12.6 *Vulnerable time for slotted ALOHA protocol*



Slotted ALOHA vulnerable time = T_{fr}

Throughput

It can be proven that the average number of successful transmissions for slotted ALOHA is $S = G \times e^{-G}$. The maximum throughput S_{max} is 0.368, when $G = 1$. In other words, if one frame is generated during one frame transmission time, then 36.8 percent of these frames reach their destination successfully. We expect $G = 1$ to produce maximum throughput because the vulnerable time is equal to the frame transmission time. Therefore, if a station generates only one frame in this vulnerable time (and no other station generates a frame during this time), the frame will reach its destination successfully.

The throughput for slotted ALOHA is $S = G \times e^{-G}$.
The maximum throughput $S_{max} = 0.368$ when $G = 1$.

Example 12.4

A slotted ALOHA network transmits 200-bit frames using a shared channel with a 200-kbps bandwidth. Find the throughput if the system (all stations together) produces

a. 1000 frames per second.

b. 500 frames per second.

c. 250 frames per second.

Solution

This situation is similar to the previous exercise except that the network is using slotted ALOHA instead of pure ALOHA. The frame transmission time is 200/200 kbps or 1 ms.

a. In this case G is 1. So $S = G \times e^{-G} = 0.368$ (36.8 percent). This means that the throughput is $1000 \times 0.0368 = 368$ frames. Only 368 out of 1000 frames will probably survive. Note that this is the maximum throughput case, percentagewise.

b. Here G is 1/2. In this case $S = G \times e^{-G} = 0.303$ (30.3 percent). This means that the throughput is $500 \times 0.0303 = 151$. Only 151 frames out of 500 will probably survive.

c. Now G is 1/4. In this case $S = G \times e^{-G} = 0.195$ (19.5 percent). This means that the throughput is $250 \times 0.195 = 49$. Only 49 frames out of 250 will probably survive.

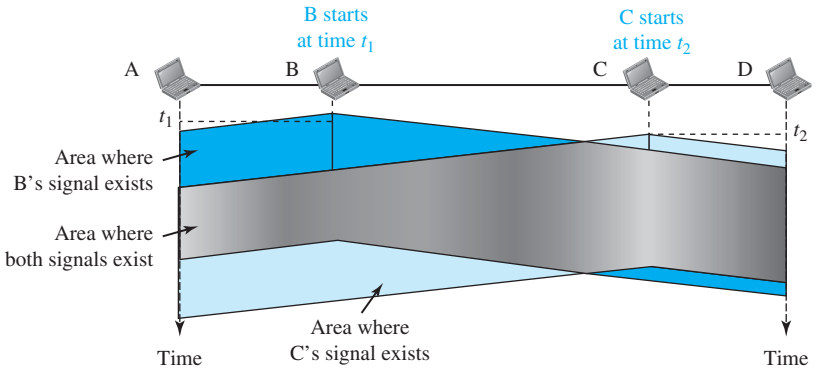
12.1.2 CSMA

To minimize the chance of collision and, therefore, increase the performance, the CSMA method was developed. The chance of collision can be reduced if a station senses the medium before trying to use it. **Carrier sense multiple access (CSMA)** requires that each station first listen to the medium (or check the state of the medium) before sending. In other words, CSMA is based on the principle "sense before transmit" or "listen before talk."

CSMA can reduce the possibility of collision, but it cannot eliminate it. The reason for this is shown in Figure 12.7, a space and time model of a CSMA network. Stations are connected to a shared channel (usually a dedicated medium).

The possibility of collision still exists because of propagation delay; when a station sends a frame, it still takes time (although very short) for the first bit to reach every station and for every station to sense it. In other words, a station may sense the medium and find it idle, only because the first bit sent by another station has not yet been received.

Figure 12.7 *Space/time model of a collision in CSMA*

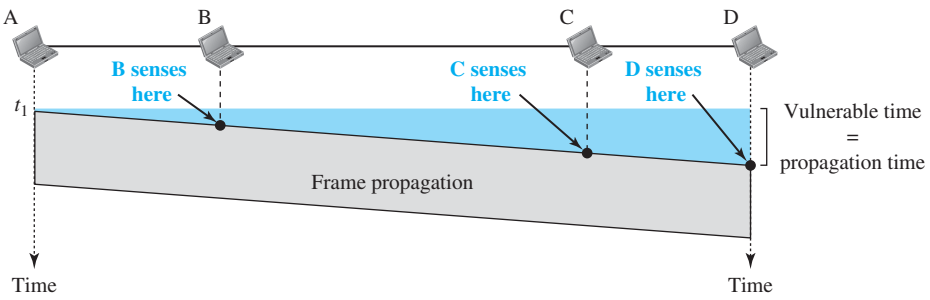


At time t_1, station B senses the medium and finds it idle, so it sends a frame. At time t_2 ($t_2 > t_1$), station C senses the medium and finds it idle because, at this time, the first bits from station B have not reached station C. Station C also sends a frame. The two signals collide and both frames are destroyed.

Vulnerable Time

The vulnerable time for CSMA is the ***propagation time*** T_p. This is the time needed for a signal to propagate from one end of the medium to the other. When a station sends a frame and any other station tries to send a frame during this time, a collision will result. But if the first bit of the frame reaches the end of the medium, every station will already have heard the bit and will refrain from sending. Figure 12.8 shows the worst case. The leftmost station, A, sends a frame at time t_1, which reaches the rightmost station, D, at time $t_1 + T_p$. The gray area shows the vulnerable area in time and space.

Figure 12.8 *Vulnerable time in CSMA*



Persistence Methods

What should a station do if the channel is busy? What should a station do if the channel is idle? Three methods have been devised to answer these questions: the **1-persistent method,** the **nonpersistent method,** and the ***p*-persistent method.** Figure 12.9 shows the behavior of three persistence methods when a station finds a channel busy.

Figure 12.9 *Behavior of three persistence methods*

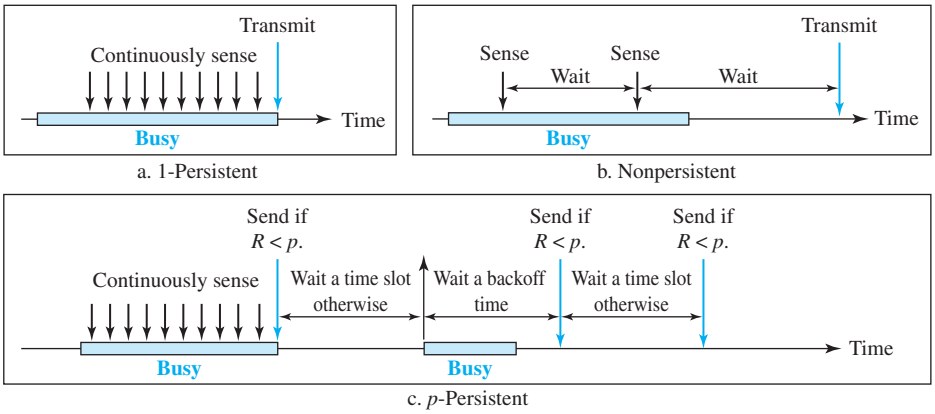


Figure 12.10 shows the flow diagrams for these methods.

1-Persistent

The *1-persistent method* is simple and straightforward. In this method, after the station finds the line idle, it sends its frame immediately (with probability 1). This method has the highest chance of collision because two or more stations may find the line idle and send their frames immediately. We will see later that Ethernet uses this method.

Nonpersistent

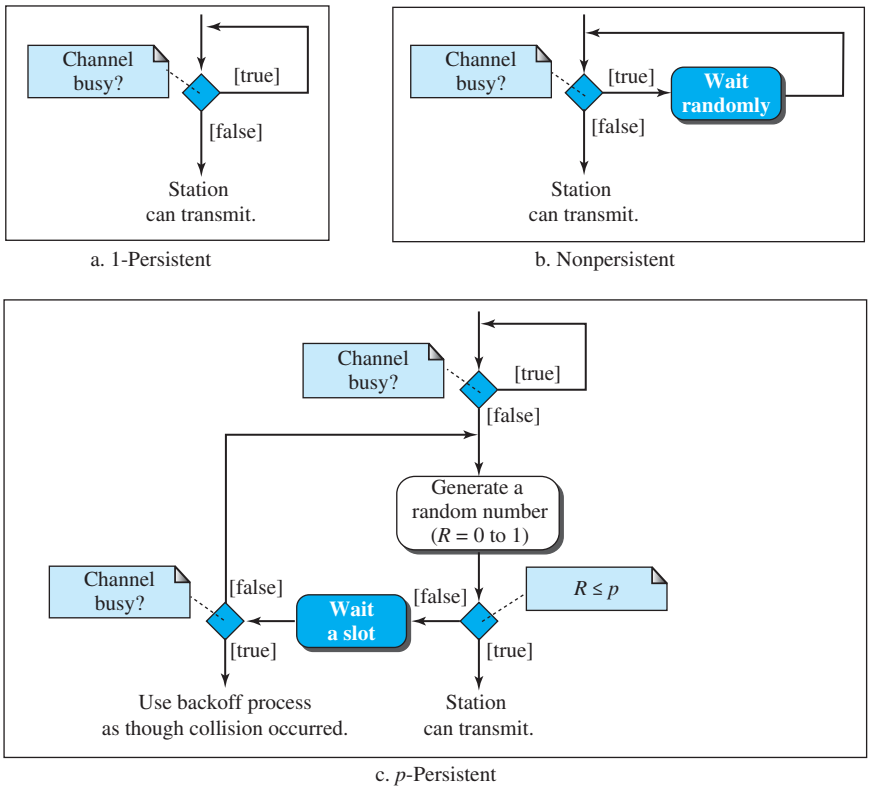
In the *nonpersistent method,* a station that has a frame to send senses the line. If the line is idle, it sends immediately. If the line is not idle, it waits a random amount of time and then senses the line again. The nonpersistent approach reduces the chance of collision because it is unlikely that two or more stations will wait the same amount of time and retry to send simultaneously. However, this method reduces the efficiency of the network because the medium remains idle when there may be stations with frames to send.

p-*Persistent*

The p-*persistent method* is used if the channel has time slots with a slot duration equal to or greater than the maximum propagation time. The *p*-persistent approach combines the advantages of the other two strategies. It reduces the chance of collision and improves efficiency. In this method, after the station finds the line idle it follows these steps:

1. With probability *p*, the station sends its frame.

Figure 12.10 *Flow diagram for three persistence methods*



2. With probability $q = 1 - p$, the station waits for the beginning of the next time slot and checks the line again.
   a. If the line is idle, it goes to step 1.
   b. If the line is busy, it acts as though a collision has occurred and uses the back-off procedure.

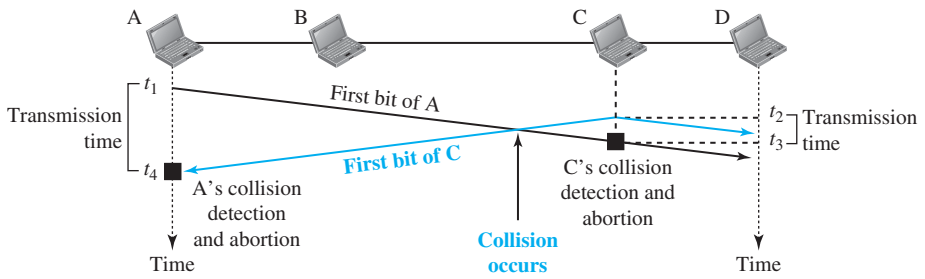
12.1.3 CSMA/CD

The CSMA method does not specify the procedure following a collision. **Carrier sense multiple access with collision detection (CSMA/CD)** augments the algorithm to handle the collision.

In this method, a station monitors the medium after it sends a frame to see if the transmission was successful. If so, the station is finished. If, however, there is a collision, the frame is sent again.

To better understand CSMA/CD, let us look at the first bits transmitted by the two stations involved in the collision. Although each station continues to send bits in the frame until it detects the collision, we show what happens as the first bits collide. In Figure 12.11, stations A and C are involved in the collision.

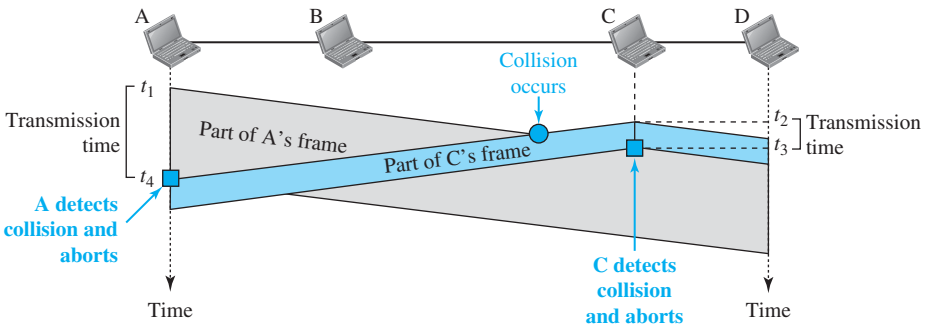
Figure 12.11 *Collision of the first bits in CSMA/CD*



At time t_1, station A has executed its persistence procedure and starts sending the bits of its frame. At time t_2, station C has not yet sensed the first bit sent by A. Station C executes its persistence procedure and starts sending the bits in its frame, which propagate both to the left and to the right. The collision occurs sometime after time t_2. Station C detects a collision at time t_3 when it receives the first bit of A's frame. Station C immediately (or after a short time, but we assume immediately) aborts transmission. Station A detects collision at time t_4 when it receives the first bit of C's frame; it also immediately aborts transmission. Looking at the figure, we see that A transmits for the duration $t_4 - t_1$; C transmits for the duration $t_3 - t_2$.

Now that we know the time durations for the two transmissions, we can show a more complete graph in Figure 12.12.

Figure 12.12 *Collision and abortion in CSMA/CD*



Minimum Frame Size

For CSMA/CD to work, we need a restriction on the frame size. Before sending the last bit of the frame, the sending station must detect a collision, if any, and abort the transmission. This is so because the station, once the entire frame is sent, does not keep a copy of

the frame and does not monitor the line for collision detection. Therefore, the frame transmission time T_{fr} must be at least two times the maximum propagation time T_p. To understand the reason, let us think about the worst-case scenario. If the two stations involved in a collision are the maximum distance apart, the signal from the first takes time T_p to reach the second, and the effect of the collision takes another time T_P to reach the first. So the requirement is that the first station must still be transmitting after $2T_p$.

Example 12.5

A network using CSMA/CD has a bandwidth of 10 Mbps. If the maximum propagation time (including the delays in the devices and ignoring the time needed to send a jamming signal, as we see later) is 25.6 μs, what is the minimum size of the frame?

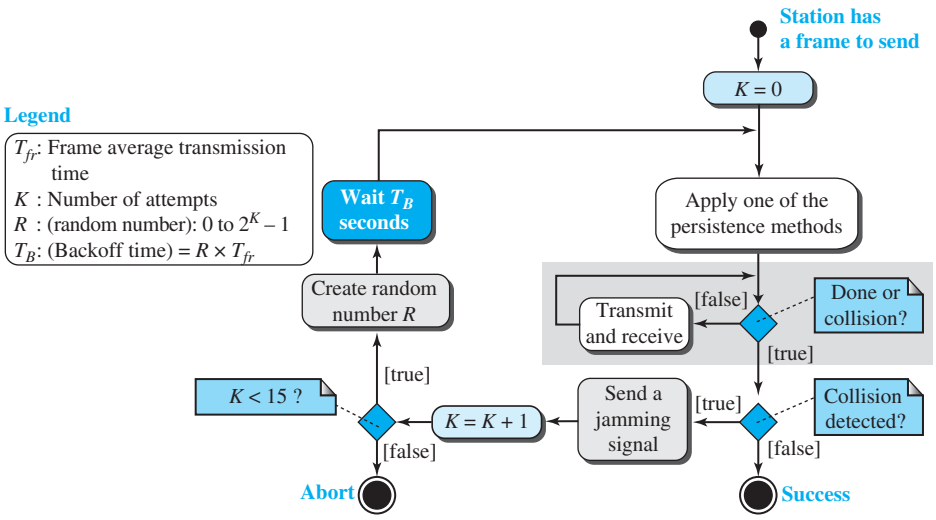
Solution

The minimum frame transmission time is $T_{fr} = 2 \times T_p = 51.2$ μs. This means, in the worst case, a station needs to transmit for a period of 51.2 μs to detect the collision. The minimum size of the frame is 10 Mbps × 51.2 μs = 512 bits or 64 bytes. This is actually the minimum size of the frame for Standard Ethernet, as we will see later in the chapter.

Procedure

Now let us look at the flow diagram for CSMA/CD in Figure 12.13. It is similar to the one for the ALOHA protocol, but there are differences.

Figure 12.13 *Flow diagram for the CSMA/CD*



The first difference is the addition of the persistence process. We need to sense the channel before we start sending the frame by using one of the persistence processes we discussed previously (nonpersistent, 1-persistent, or *p*-persistent). The corresponding box can be replaced by one of the persistence processes shown in Figure 12.10.

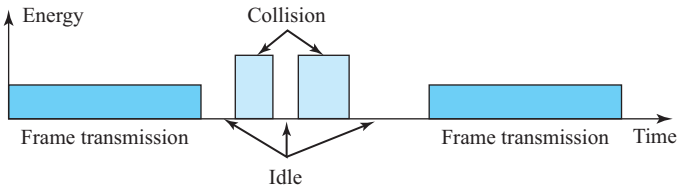
The second difference is the frame transmission. In ALOHA, we first transmit the entire frame and then wait for an acknowledgment. In CSMA/CD, transmission and collision detection are continuous processes. We do not send the entire frame and then look for a collision. The station transmits and receives continuously and simultaneously (using two different ports or a bidirectional port). We use a loop to show that transmission is a continuous process. We constantly monitor in order to detect one of two conditions: either transmission is finished or a collision is detected. Either event stops transmission. When we come out of the loop, if a collision has not been detected, it means that transmission is complete; the entire frame is transmitted. Otherwise, a collision has occurred.

The third difference is the sending of a short **jamming signal** to make sure that all other stations become aware of the collision.

Energy Level

We can say that the level of energy in a channel can have three values: zero, normal, and abnormal. At the zero level, the channel is idle. At the normal level, a station has successfully captured the channel and is sending its frame. At the abnormal level, there is a collision and the level of the energy is twice the normal level. A station that has a frame to send or is sending a frame needs to monitor the energy level to determine if the channel is idle, busy, or in collision mode. Figure 12.14 shows the situation.

Figure 12.14 *Energy level during transmission, idleness, or collision*



Throughput

The throughput of CSMA/CD is greater than that of pure or slotted ALOHA. The maximum throughput occurs at a different value of G and is based on the persistence method and the value of p in the p-persistent approach. For the 1-persistent method, the maximum throughput is around 50 percent when $G = 1$. For the nonpersistent method, the maximum throughput can go up to 90 percent when G is between 3 and 8.

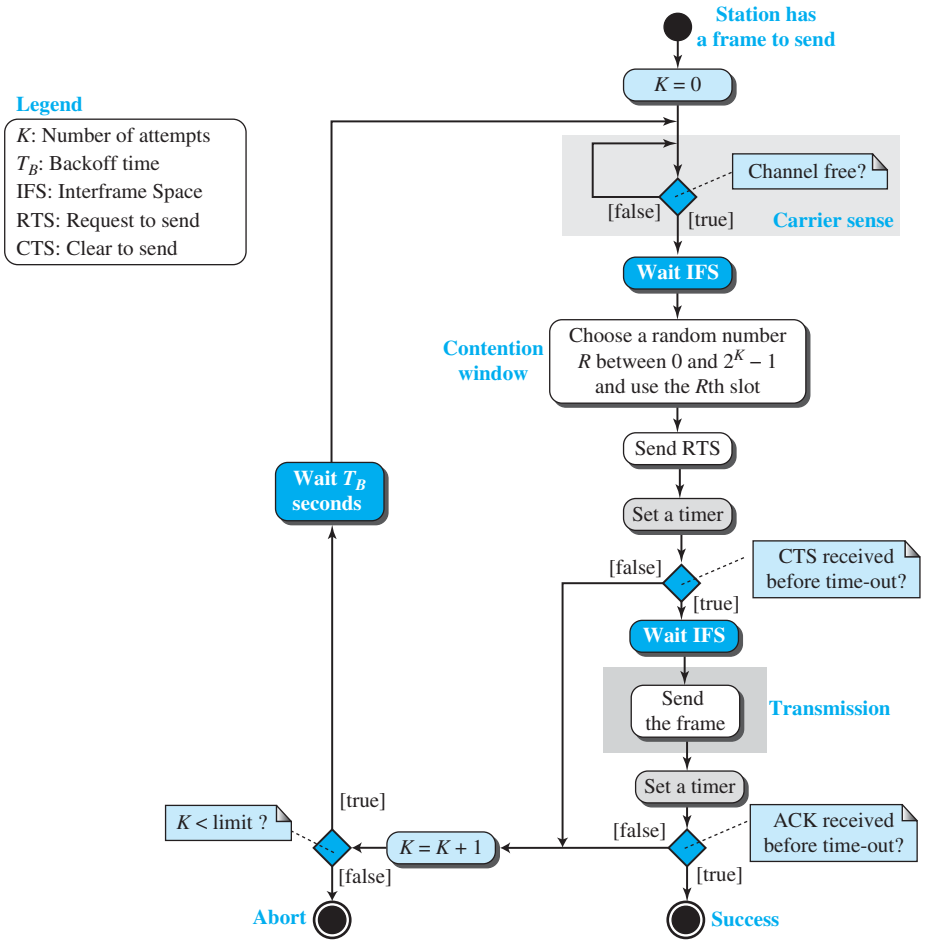
Traditional Ethernet

One of the LAN protocols that used CSMA/CD is the traditional Ethernet with the data rate of 10 Mbps. We discuss the Ethernet LANs in Chapter 13, but it is good to know that the traditional Ethernet was a broadcast LAN that used the 1-persistence method to control access to the common media. Later versions of Ethernet try to move from CSMA/CD access methods for the reason that we discuss in Chapter 13.

12.1.4 CSMA/CA

Carrier sense multiple access with collision avoidance (CSMA/CA) was invented for wireless networks. Collisions are avoided through the use of CSMA/CA's three strategies: the interframe space, the contention window, and acknowledgments, as shown in Figure 12.15. We discuss RTS and CTS frames later.

Figure 12.15 *Flow diagram of CSMA/CA*

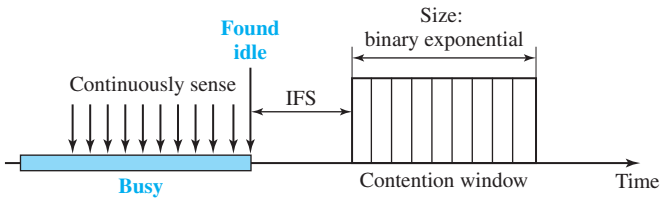


- ***Interframe Space (IFS).*** First, collisions are avoided by deferring transmission even if the channel is found idle. When an idle channel is found, the station does not send immediately. It waits for a period of time called the ***interframe space*** or ***IFS.*** Even though the channel may appear idle when it is sensed, a distant station may have already started transmitting. The distant station's signal has not yet reached this

station. The IFS time allows the front of the transmitted signal by the distant station to reach this station. After waiting an IFS time, if the channel is still idle, the station can send, but it still needs to wait a time equal to the contention window (described next). The IFS variable can also be used to prioritize stations or frame types. For example, a station that is assigned a shorter IFS has a higher priority.

- ***Contention Window.*** The **contention window** is an amount of time divided into slots. A station that is ready to send chooses a random number of slots as its wait time. The number of slots in the window changes according to the binary exponential backoff strategy. This means that it is set to one slot the first time and then doubles each time the station cannot detect an idle channel after the IFS time. This is very similar to the *p*-persistent method except that a random outcome defines the number of slots taken by the waiting station. One interesting point about the contention window is that the station needs to sense the channel after each time slot. However, if the station finds the channel busy, it does not restart the process; it just stops the timer and restarts it when the channel is sensed as idle. This gives priority to the station with the longest waiting time. See Figure 12.16.

Figure 12.16 *Contention window*



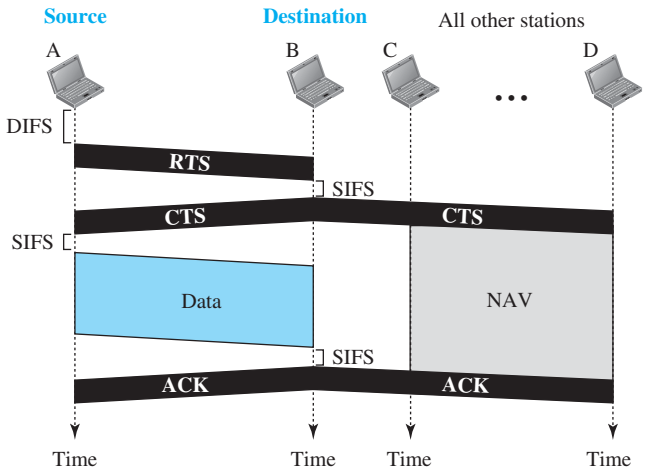
- ***Acknowledgment.*** With all these precautions, there still may be a collision resulting in destroyed data. In addition, the data may be corrupted during the transmission. The positive acknowledgment and the time-out timer can help guarantee that the receiver has received the frame.

Frame Exchange Time Line

Figure 12.17 shows the exchange of data and control frames in time.

1. Before sending a frame, the source station senses the medium by checking the energy level at the carrier frequency.
   a. The channel uses a persistence strategy with backoff until the channel is idle.
   b. After the station is found to be idle, the station waits for a period of time called the ***DCF interframe space (DIFS);*** then the station sends a control frame called the *request to send (RTS)*.
2. After receiving the RTS and waiting a period of time called the ***short interframe space (SIFS),*** the destination station sends a control frame, called the *clear to send (CTS),* to the source station. This control frame indicates that the destination station is ready to receive data.

Figure 12.17 *CSMA/CA and NAV*



3. The source station sends data after waiting an amount of time equal to SIFS.
4. The destination station, after waiting an amount of time equal to SIFS, sends an acknowledgment to show that the frame has been received. Acknowledgment is needed in this protocol because the station does not have any means to check for the successful arrival of its data at the destination. On the other hand, the lack of collision in CSMA/CD is a kind of indication to the source that data have arrived.

Network Allocation Vector

How do other stations defer sending their data if one station acquires access? In other words, how is the *collision avoidance* aspect of this protocol accomplished? The key is a feature called *NAV.*

When a station sends an RTS frame, it includes the duration of time that it needs to occupy the channel. The stations that are affected by this transmission create a timer called a **network allocation vector (NAV)** that shows how much time must pass before these stations are allowed to check the channel for idleness. Each time a station accesses the system and sends an RTS frame, other stations start their NAV. In other words, each station, before sensing the physical medium to see if it is idle, first checks its NAV to see if it has expired. Figure 12.17 shows the idea of NAV.

Collision During Handshaking

What happens if there is a collision during the time when RTS or CTS control frames are in transition, often called the *handshaking period?* Two or more stations may try to send RTS frames at the same time. These control frames may collide. However, because there is no mechanism for collision detection, the sender assumes there has been a collision if it has not received a CTS frame from the receiver. The backoff strategy is employed, and the sender tries again.

Hidden-Station Problem

The solution to the hidden station problem is the use of the handshake frames (RTS and CTS). Figure 12.17 also shows that the RTS message from B reaches A, but not C. However, because both B and C are within the range of A, the CTS message, which contains the duration of data transmission from B to A, reaches C. Station C knows that some hidden station is using the channel and refrains from transmitting until that duration is over.

CSMA/CA and Wireless Networks

CSMA/CA was mostly intended for use in wireless networks. The procedure described above, however, is not sophisticated enough to handle some particular issues related to wireless networks, such as hidden terminals or exposed terminals. We will see how these issues are solved by augmenting the above protocol with handshaking features. The use of CSMA/CA in wireless networks will be discussed in Chapter 15.

12.2 CONTROLLED ACCESS

In **controlled access,** the stations consult one another to find which station has the right to send. A station cannot send unless it has been authorized by other stations. We discuss three controlled-access methods.

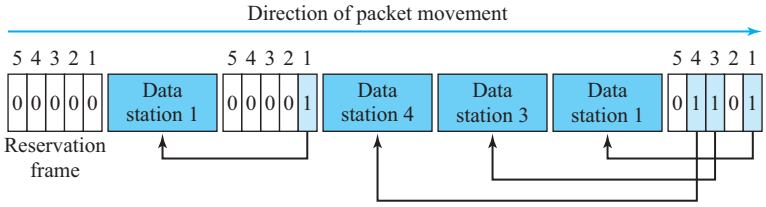
12.2.1 Reservation

In the **reservation** method, a station needs to make a reservation before sending data. Time is divided into intervals. In each interval, a reservation frame precedes the data frames sent in that interval.

If there are *N* stations in the system, there are exactly *N* reservation minislots in the reservation frame. Each minislot belongs to a station. When a station needs to send a data frame, it makes a reservation in its own minislot. The stations that have made reservations can send their data frames after the reservation frame.

Figure 12.18 shows a situation with five stations and a five-minislot reservation frame. In the first interval, only stations 1, 3, and 4 have made reservations. In the second interval, only station 1 has made a reservation.

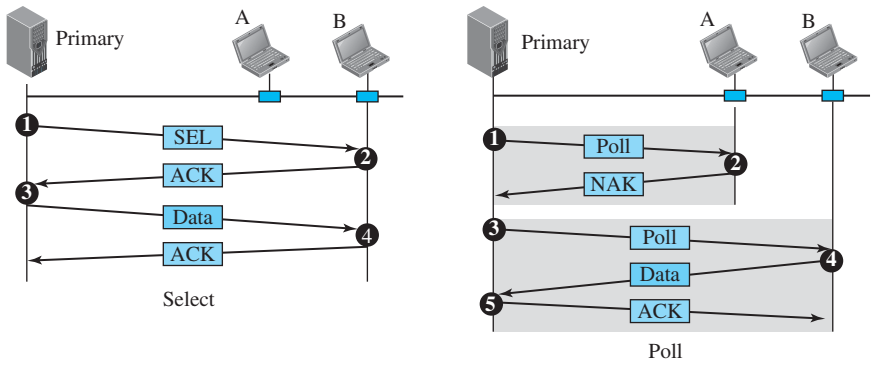
Figure 12.18 *Reservation access method*



12.2.2 Polling

Polling works with topologies in which one device is designated as a ***primary station*** and the other devices are ***secondary stations.*** All data exchanges must be made through the primary device even when the ultimate destination is a secondary device. The primary device controls the link; the secondary devices follow its instructions. It is up to the primary device to determine which device is allowed to use the channel at a given time. The primary device, therefore, is always the initiator of a session (see Figure 12.19). This method uses poll and select functions to prevent collisions. However, the drawback is if the primary station fails, the system goes down.

Figure 12.19 *Select and poll functions in polling-access method*



Select

The *select* function is used whenever the primary device has something to send. Remember that the primary controls the link. If the primary is neither sending nor receiving data, it knows the link is available. If it has something to send, the primary device sends it. What it does not know, however, is whether the target device is prepared to receive. So the primary must alert the secondary to the upcoming transmission and wait for an acknowledgment of the secondary's ready status. Before sending data, the primary creates and transmits a select (SEL) frame, one field of which includes the address of the intended secondary.

Poll

The *poll* function is used by the primary device to solicit transmissions from the secondary devices. When the primary is ready to receive data, it must ask (poll) each device in turn if it has anything to send. When the first secondary is approached, it responds either with a NAK frame if it has nothing to send or with data (in the form of a data frame) if it does. If the response is negative (a NAK frame), then the primary polls the next secondary in the same manner until it finds one with data to send. When the response is positive (a data frame), the primary reads the frame and returns an acknowledgment (ACK frame), verifying its receipt.

12.2.3 Token Passing

In the **token-passing** method, the stations in a network are organized in a logical ring. In other words, for each station, there is a *predecessor* and a *successor*. The predecessor is the station which is logically before the station in the ring; the successor is the station which is after the station in the ring. The current station is the one that is accessing the channel now. The right to this access has been passed from the predecessor to the current station. The right will be passed to the successor when the current station has no more data to send.

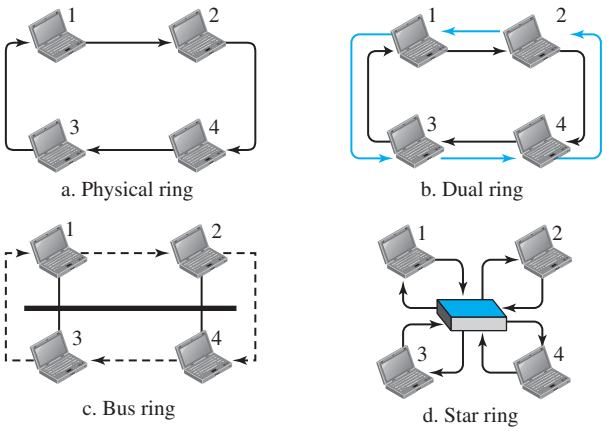
But how is the right to access the channel passed from one station to another? In this method, a special packet called a ***token*** circulates through the ring. The possession of the token gives the station the right to access the channel and send its data. When a station has some data to send, it waits until it receives the token from its predecessor. It then holds the token and sends its data. When the station has no more data to send, it releases the token, passing it to the next logical station in the ring. The station cannot send data until it receives the token again in the next round. In this process, when a station receives the token and has no data to send, it just passes the data to the next station.

Token management is needed for this access method. Stations must be limited in the time they can have possession of the token. The token must be monitored to ensure it has not been lost or destroyed. For example, if a station that is holding the token fails, the token will disappear from the network. Another function of token management is to assign priorities to the stations and to the types of data being transmitted. And finally, token management is needed to make low-priority stations release the token to high-priority stations.

Logical Ring

In a token-passing network, stations do not have to be physically connected in a ring; the ring can be a logical one. Figure 12.20 shows four different physical topologies that can create a logical ring.

Figure 12.20 *Logical ring and physical topology in token-passing access method*



In the physical ring topology, when a station sends the token to its successor, the token cannot be seen by other stations; the successor is the next one in line. This means that the token does not have to have the address of the next successor. The problem with this topology is that if one of the links—the medium between two adjacent stations—fails, the whole system fails.

The dual ring topology uses a second (auxiliary) ring which operates in the reverse direction compared with the main ring. The second ring is for emergencies only (such as a spare tire for a car). If one of the links in the main ring fails, the system automatically combines the two rings to form a temporary ring. After the failed link is restored, the auxiliary ring becomes idle again. Note that for this topology to work, each station needs to have two transmitter ports and two receiver ports. The high-speed Token Ring networks called *FDDI (Fiber Distributed Data Interface)* and *CDDI (Copper Distributed Data Interface)* use this topology.

In the bus ring topology, also called a token bus, the stations are connected to a single cable called a *bus.* They, however, make a logical ring, because each station knows the address of its successor (and also predecessor for token management purposes). When a station has finished sending its data, it releases the token and inserts the address of its successor in the token. Only the station with the address matching the destination address of the token gets the token to access the shared media. The Token Bus LAN, standardized by IEEE, uses this topology.

In a star ring topology, the physical topology is a star. There is a hub, however, that acts as the connector. The wiring inside the hub makes the ring; the stations are connected to this ring through the two wire connections. This topology makes the network less prone to failure because if a link goes down, it will be bypassed by the hub and the rest of the stations can operate. Also adding and removing stations from the ring is easier. This topology is still used in the Token Ring LAN designed by IBM.

12.3 CHANNELIZATION

Channelization (or *channel partition,* as it is sometimes called) is a multiple-access method in which the available bandwidth of a link is shared in time, frequency, or through code, among different stations. In this section, we discuss three channelization protocols: FDMA, TDMA, and CDMA.

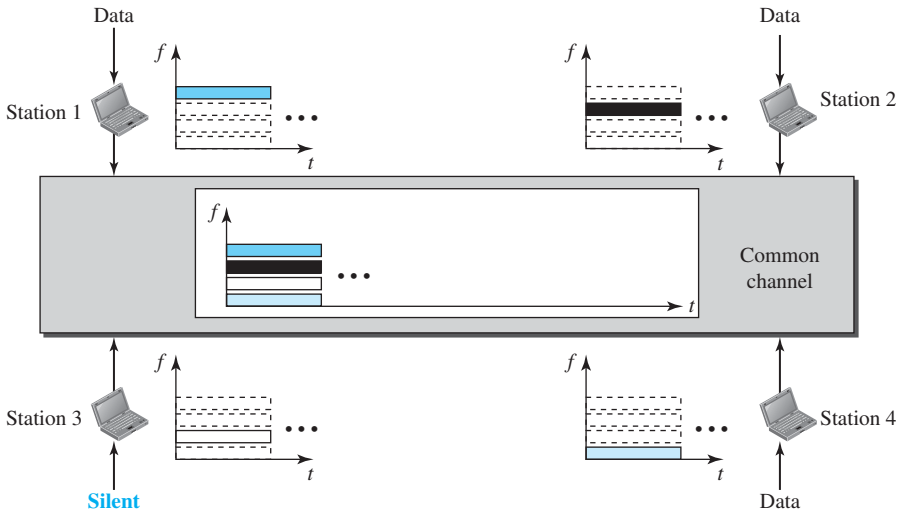
We see the application of all these methods in Chapter 16 when we discuss cellular phone systems.

12.3.1 FDMA

In **frequency-division multiple access (FDMA),** the available bandwidth is divided into frequency bands. Each station is allocated a band to send its data. In other words, each band is reserved for a specific station, and it belongs to the station all the time. Each station also uses a bandpass filter to confine the transmitter frequencies. To prevent

station interferences, the allocated bands are separated from one another by small *guard bands*. Figure 12.21 shows the idea of FDMA.

Figure 12.21 *Frequency-division multiple access (FDMA)*



In FDMA, the available bandwidth of the common channel is divided into bands that are separated by guard bands.

FDMA specifies a predetermined frequency band for the entire period of communication. This means that stream data (a continuous flow of data that may not be packetized) can easily be used with FDMA. We will see in Chapter 16 how this feature can be used in cellular telephone systems.

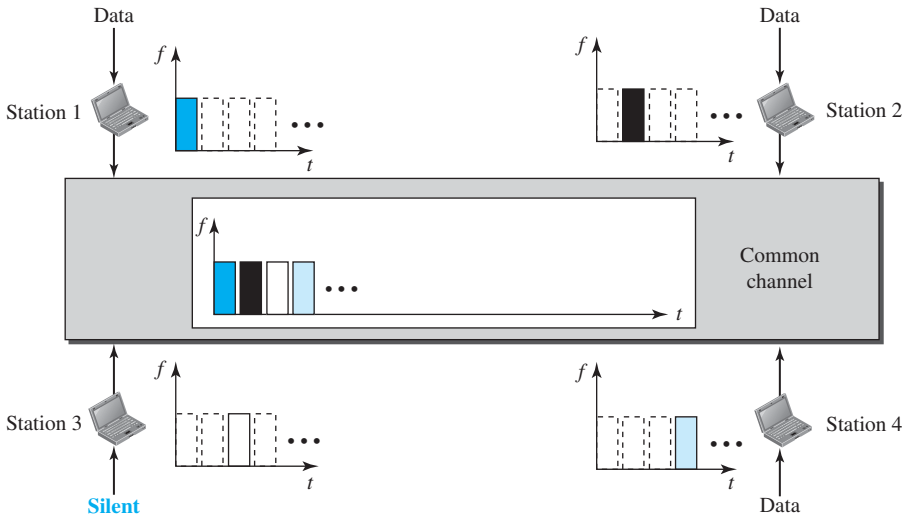
We need to emphasize that although FDMA and frequency-division multiplexing (FDM) conceptually seem similar, there are differences between them. FDM, as we saw in Chapter 6, is a physical layer technique that combines the loads from low-bandwidth channels and transmits them by using a high-bandwidth channel. The channels that are combined are low-pass. The multiplexer modulates the signals, combines them, and creates a bandpass signal. The bandwidth of each channel is shifted by the multiplexer.

FDMA, on the other hand, is an access method in the data-link layer. The data-link layer in each station tells its physical layer to make a bandpass signal from the data passed to it. The signal must be created in the allocated band. There is no physical multiplexer at the physical layer. The signals created at each station are automatically bandpass-filtered. They are mixed when they are sent to the common channel.

12.3.2 TDMA

In **time-division multiple access (TDMA),** the stations share the bandwidth of the channel in time. Each station is allocated a time slot during which it can send data. Each station transmits its data in its assigned time slot. Figure 12.22 shows the idea behind TDMA.

Figure 12.22 *Time-division multiple access (TDMA)*



The main problem with TDMA lies in achieving synchronization between the different stations. Each station needs to know the beginning of its slot and the location of its slot. This may be difficult because of propagation delays introduced in the system if the stations are spread over a large area. To compensate for the delays, we can insert *guard times*. Synchronization is normally accomplished by having some synchronization bits (normally referred to as *preamble bits*) at the beginning of each slot.

In TDMA, the bandwidth is just one channel that is timeshared between different stations.

We also need to emphasize that although TDMA and time-division multiplexing (TDM) conceptually seem the same, there are differences between them. TDM, as we saw in Chapter 6, is a physical layer technique that combines the data from slower channels and transmits them by using a faster channel. The process uses a physical multiplexer that interleaves data units from each channel.

TDMA, on the other hand, is an access method in the data-link layer. The data-link layer in each station tells its physical layer to use the allocated time slot. There is no physical multiplexer at the physical layer.

12.3.3 CDMA

Code-division multiple access (CDMA) was conceived several decades ago. Recent advances in electronic technology have finally made its implementation possible. CDMA differs from FDMA in that only one channel occupies the entire bandwidth of the link. It differs from TDMA in that all stations can send data simultaneously; there is no timesharing.

In CDMA, one channel carries all transmissions simultaneously.

Analogy

Let us first give an analogy. CDMA simply means communication with different codes. For example, in a large room with many people, two people can talk privately in English if nobody else understands English. Another two people can talk in Chinese if they are the only ones who understand Chinese, and so on. In other words, the common channel, the space of the room in this case, can easily allow communication between several couples, but in different languages (codes).

Idea

Let us assume we have four stations, 1, 2, 3, and 4, connected to the same channel. The data from station 1 are d_1, from station 2 are d_2, and so on. The code assigned to the first station is c_1, to the second is c_2, and so on. We assume that the assigned codes have two properties.

1. If we multiply each code by another, we get 0.
2. If we multiply each code by itself, we get 4 (the number of stations).

With these two properties in mind, let us see how the above four stations can send data using the same common channel, as shown in Figure 12.23.

Station 1 multiplies (a special kind of multiplication, as we will see) its data by its code to get $d_1 \cdot c_1$. Station 2 multiplies its data by its code to get $d_2 \cdot c_2$, and so on. The data that go on the channel are the sum of all these terms, as shown in the box. Any station that wants to receive data from one of the other three multiplies the data on the channel by the code of the sender. For example, suppose stations 1 and 2 are talking to each other. Station 2 wants to hear what station 1 is saying. It multiplies the data on the channel by c_1, the code of station 1.

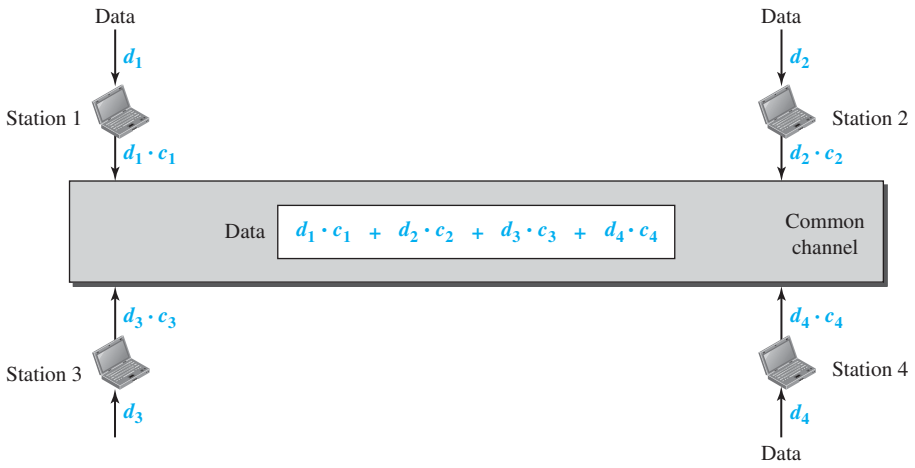
Because $(c_1 \cdot c_1)$ is 4, but $(c_2 \cdot c_1)$, $(c_3 \cdot c_1)$, and $(c_4 \cdot c_1)$ are all 0s, station 2 divides the result by 4 to get the data from station 1.

Chips

CDMA is based on coding theory. Each station is assigned a code, which is a sequence of numbers called *chips,* as shown in Figure 12.24. The codes are for the previous example.

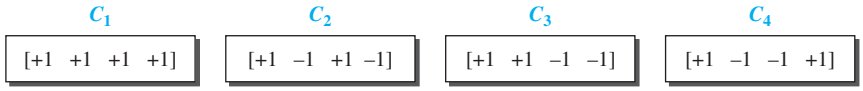
Later in this chapter we show how we chose these sequences. For now, we need to know that we did not choose the sequences randomly; they were carefully selected. They are called ***orthogonal sequences*** and have the following properties:

Figure 12.23 *Simple idea of communication with code*



$$\text{data} = (d_1 \cdot c_1 + d_2 \cdot c_2 + d_3 \cdot c_3 + d_4 \cdot c_4) \cdot c_1$$
$$= d_1 \cdot c_1 \cdot c_1 + d_2 \cdot c_2 \cdot c_1 + d_3 \cdot c_3 \cdot c_1 + d_4 \cdot c_4 \cdot c_1 = 4 \times d_1$$

Figure 12.24 *Chip sequences*



1. Each sequence is made of N elements, where N is the number of stations.
2. If we multiply a sequence by a number, every element in the sequence is multiplied by that element. This is called multiplication of a sequence by a scalar. For example,

$$2 \bullet [+1 +1 -1 -1] = [+2 +2 -2 -2]$$

3. If we multiply two equal sequences, element by element, and add the results, we get N, where N is the number of elements in each sequence. This is called the ***inner product*** of two equal sequences. For example,

$$[+1 +1 -1 -1] \bullet [+1 +1 -1 -1] = 1 + 1 + 1 + 1 = 4$$

4. If we multiply two different sequences, element by element, and add the results, we get 0. This is called the *inner product* of two different sequences. For example,

$$[+1 +1 -1 -1] \bullet [+1 +1 +1 +1] = 1 + 1 - 1 - 1 = 0$$

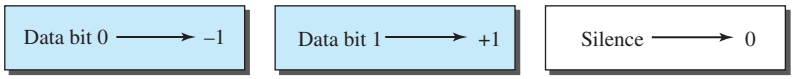
5. Adding two sequences means adding the corresponding elements. The result is another sequence. For example,

$$[+1\ +1\ -1\ -1] + [+1\ +1\ +1\ +1] = [+2\ +2\ \ 0\ \ 0]$$

Data Representation

We follow these rules for encoding: If a station needs to send a 0 bit, it encodes it as −1; if it needs to send a 1 bit, it encodes it as +1. When a station is idle, it sends no signal, which is interpreted as a 0. These are shown in Figure 12.25.

Figure 12.25 *Data representation in CDMA*



Encoding and Decoding

As a simple example, we show how four stations share the link during a 1-bit interval. The procedure can easily be repeated for additional intervals. We assume that stations 1 and 2 are sending a 0 bit and channel 4 is sending a 1 bit. Station 3 is silent. The data at the sender site are translated to −1, −1, 0, and +1. Each station multiplies the corresponding number by its chip (its orthogonal sequence), which is unique for each station. The result is a new sequence which is sent to the channel. For simplicity, we assume that all stations send the resulting sequences at the same time. The sequence on the channel is the sum of all four sequences as defined before. Figure 12.26 shows the situation.

Now imagine that station 3, which we said is silent, is listening to station 2. Station 3 multiplies the total data on the channel by the code for station 2, which is [+1 −1 +1 −1], to get

$$[-1\ -1\ -3\ +1] \bullet [+1\ -1\ +1\ -1] = -4/4 = -1 \quad \rightarrow \quad \textbf{bit 1}$$

Signal Level

The process can be better understood if we show the digital signal produced by each station and the data recovered at the destination (see Figure 12.27). The figure shows the corresponding signals for each station (using NRZ-L for simplicity) and the signal that is on the common channel.

Figure 12.28 shows how station 3 can detect the data sent by station 2 by using the code for station 2. The total data on the channel are multiplied (inner product operation) by the signal representing station 2 chip code to get a new signal. The station then integrates and adds the area under the signal, to get the value −4, which is divided by 4 and interpreted as bit 0.

Sequence Generation

To generate chip sequences, we use a **Walsh table,** which is a two-dimensional table with an equal number of rows and columns, as shown in Figure 12.29.

Figure 12.26 *Sharing channel in CDMA*

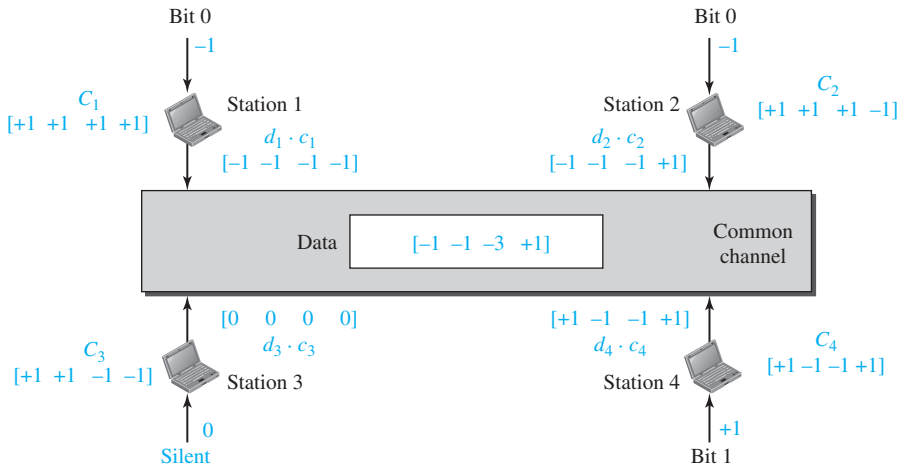
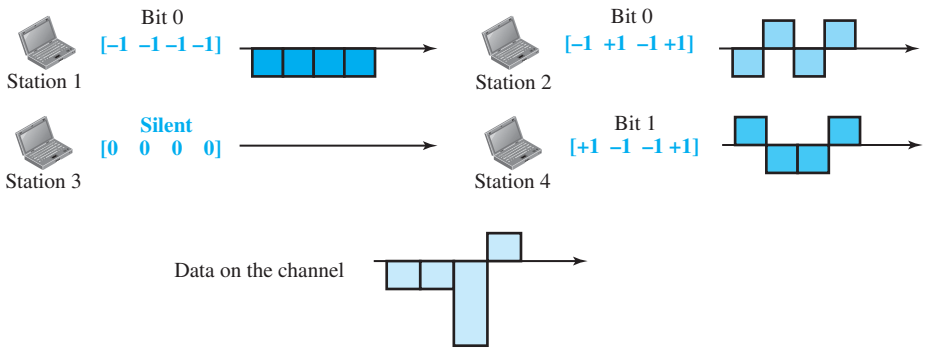


Figure 12.27 *Digital signal created by four stations in CDMA*



In the Walsh table, each row is a sequence of chips. W_1 for a one-chip sequence has one row and one column. We can choose −1 or +1 for the chip for this trivial table (we chose +1). According to Walsh, if we know the table for N sequences W_N, we can create the table for $2N$ sequences W_{2N}, as shown in Figure 12.29. The W_N with the overbar $\overline{W_N}$ stands for the complement of W_N, where each +1 is changed to −1 and vice versa. Figure 12.29 also shows how we can create W_2 and W_4 from W_1. After we select W_1, W_2 can be made from four W_1s, with the last one the complement of W_1. After W_2 is generated, W_4 can be made of four W_2s, with the last one the complement of W_2. Of course, W_8 is composed of four W_4s, and so on. Note that after W_N is made, each station is assigned a chip corresponding to a row.

Figure 12.28 *Decoding of the composite signal for one in CDMA*

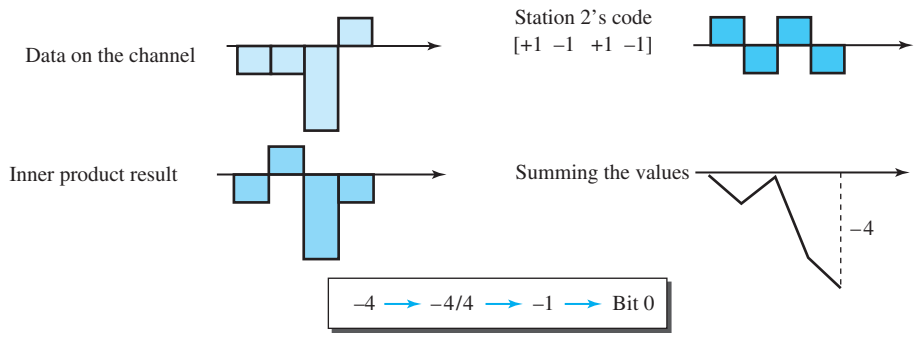
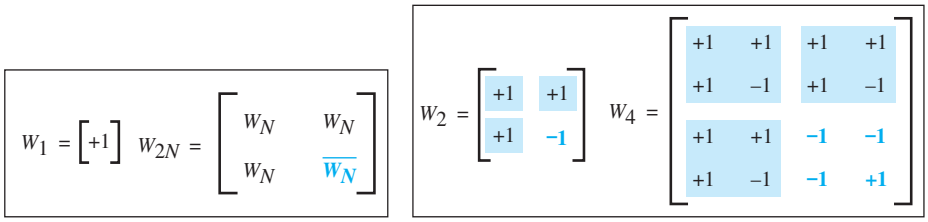


Figure 12.29 *General rule and examples of creating Walsh tables*



a. Two basic rules b. Generation of W_2 and W_4

Something we need to emphasize is that the number of sequences, N, needs to be a power of 2. In other words, we need to have $N = 2^m$.

The number of sequences in a Walsh table needs to be $N = 2^m$.

Example 12.6

Find the chips for a network with

a. Two stations

b. Four stations

Solution

We can use the rows of W_2 and W_4 in Figure 12.29:

a. For a two-station network, we have [+1 +1] and [+1 –1].

b. For a four-station network we have [+1 +1 +1 +1], [+1 –1 +1 –1], [+1 +1 –1 –1], and [+1 –1 –1 +1].

Example 12.7

What is the number of sequences if we have 90 stations in our network?

Solution

The number of sequences needs to be 2^m. We need to choose $m = 7$ and $N = 2^7$ or 128. We can then use 90 of the sequences as the chips.

Example 12.8

Prove that a receiving station can get the data sent by a specific sender if it multiplies the entire data on the channel by the sender's chip code and then divides it by the number of stations.

Solution

Let us prove this for the first station, using our previous four-station example. We can say that the data on the channel $D = (d_1 \cdot c_1 + d_2 \cdot c_2 + d_3 \cdot c_3 + d_4 \cdot c_4)$. The receiver that wants to get the data sent by station 1 multiplies these data by c_1.

$$
\begin{aligned}
D \cdot c_1 &= (d_1 \cdot c_1 + d_2 \cdot c_2 + d_3 \cdot c_3 + d_4 \cdot c_4) \quad c_1 \\
&= d_1 \cdot c_1 \cdot c_1 + d_2 \cdot c_2 \cdot c_1 + d_3 \cdot c_3 \cdot c_1 + d_4 \cdot c_4 \cdot c_1 \\
&= d_1 \times N + d_2 \times 0 + d_3 \times 0 + d_4 \times 0 \\
&= d_1 \times N
\end{aligned}
$$

When we divide the result by N, we get d_1.

12.4 END-CHAPTER MATERIALS

12.4.1 Recommended Reading

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

Books

Several excellent 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].

RFCs

A discussion of the use of the checksum in the Internet can be found in RFC 1141.

12.4.2 Key Terms

1-persistent method
ALOHA
binary exponential backoff
carrier sense multiple access (CSMA)
carrier sense multiple access with collision avoidance (CSMA/CA)
carrier sense multiple access with collision detection (CSMA/CD)
channelization
code-division multiple access (CDMA)
collision
contention
contention window
controlled access
DCF interframe space (DIFS)
frequency-division multiple access (FDMA)
inner product
interface space (IFS)
jamming signal
media access control (MAC)
multiple access (MA)
network allocation vector (NAV)
nonpersistent method
orthogonal sequence
p-persistent method
polling
primary station
propagation time
pure ALOHA
random access
reservation
secondary station
short interframe space (SIFS)
slotted ALOHA
time-division multiple access (TDMA)
token
token passing
vulnerable time
Walsh table

12.4.3 Summary

Many formal protocols have been devised to handle access to a shared link. We categorize them into three groups: random access protocols, controlled access protocols, and channelization protocols.

In random access or contention methods, no station is superior to another station and none is assigned the control over another. ALOHA allows multiple access (MA) to the shared medium.There are potential collisions in this arrangement. To minimize the chance of collision and, therefore, increase the performance, the CSMA method was developed. The chance of collision can be reduced if a station senses the medium before trying to use it. Carrier sense multiple access (CSMA) requires that each station first listen to the medium before sending. Carrier sense multiple access with collision detection (CSMA/CD) augments the CSMA algorithm to handle collision. In this method, a station monitors the medium after it sends a frame to see if the transmission was successful. If so, the station is finished. If, however, there is a collision, the frame is sent again. To avoid collisions on wireless networks, carrier sense multiple access with collision avoidance (CSMA/CA) was invented. Collisions are avoided through the use of three strategies: the interframe space, the contention window, and acknowledgments.

In controlled access, the stations consult one another to find which station has the right to send. A station cannot send unless it has been authorized by other stations. We discussed three popular controlled-access methods: reservation, polling, and token passing. In the reservation access method, a station needs to make a reservation before sending data. Time is divided into intervals. In each interval, a reservation frame precedes the data frames sent in that interval. In the polling method, all data exchanges must be made through the primary device even when the ultimate destination is a secondary device. The primary device controls the link; the secondary devices follow its

instructions. In the token-passing method, the stations in a network are organized in a logical ring. Each station has a predecessor and a successor. A special packet called a *token* circulates through the ring.

Channelization is a multiple-access method in which the available bandwidth of a link is shared in time, frequency, or through code, between different stations. We discussed three channelization protocols: FDMA, TDMA, and CDMA. In frequency-division multiple access (FDMA), the available bandwidth is divided into frequency bands. Each station is allocated a band to send its data. In other words, each band is reserved for a specific station, and it belongs to the station all the time. In time-division multiple access (TDMA), the stations share the bandwidth of the channel in time. Each station is allocated a time slot during which it can send data. Each station transmits its data in its assigned time slot. In code-division multiple access (CDMA), the stations use different codes to achieve multiple access. CDMA is based on coding theory and uses sequences of numbers called *chips*. The sequences are generated using orthogonal codes such as the Walsh tables.

12.5 PRACTICE SET

12.5.1 Quizzes

A set of interactive quizzes for this chapter can be found on the book website. It is strongly recommended that the student take the quizzes to check his/her understanding of the materials before continuing with the practice set.

12.5.2 Questions

Q12-1. Which of the following is a random-access protocol?

a. CSMA/CD **b.** Polling **c.** TDMA

Q12-2. Which of the following is a controlled-access protocol?

a. Token-passing **b.** Polling **c.** FDMA

Q12-3. Which of the following is a channelization protocol?

a. ALOHA **b.** Token-passing **c.** CDMA

Q12-4. Stations in a pure Aloha network send frames of size 1000 bits at the rate of 1 Mbps. What is the vulnerable time for this network?

Q12-5. Stations in an slotted Aloha network send frames of size 1000 bits at the rate of 1 Mbps. What is the vulnerable time for this network?

Q12-6. In a pure Aloha network with $G = 1/2$, how is the throughput affected in each of the following cases?

a. G is increased to 1. **b.** G is decreased to 1/4.

Q12-7. In a slotted Aloha network with $G = 1/2$, how is the throughput affected in each of the following cases?

a. G is increased to 1. **b.** G is decreased to 1/4.

Q12-8. To understand the uses of K in Figure 12.3, find the probability that a station can send immediately in each of the following cases:

a. After one failure. **b.** After three failures.

Q12-9. To understand the uses of K in Figure 12.13, find the probability that a station can send immediately in each of the following cases:

a. After one failure. **b.** After four failures.

Q12-10. To understand the uses of K in Figure 12.15, find the probability that a station can send immediately in each of the following cases:

a. After two failures. **b.** After five failures.

Q12-11. Based on Figure 12.3, how do we interpret *success* in an Aloha network?

Q12-12. Based on Figure 12.13, how do we interpret *success* in an Aloha network?

Q12-13. Based on Figure 12.15, how do we interpret *success* in an Aloha network?

Q12-14. Assume the propagation delay in a broadcast network is 5 μs and the frame transmission time is 10 μs.

a. How long does it take for the first bit to reach the destination?

b. How long does it take for the last bit to reach the destination after the first bit has arrived?

c. How long is the network involved with this frame (vulnerable to collision)?

Q12-15. Assume the propagation delay in a broadcast network is 3 μs and the frame transmission time is 5 μs. Can the collision be detected no matter where it occurs?

Q12-16. Assume the propagation delay in a broadcast network is 6 μs and the frame transmission time is 4 μs. Can the collision be detected no matter where it occurs?

Q12-17. Explain why collision is an issue in random access protocols but not in controlled access protocols.

Q12-18. Explain why collision is an issue in random access protocols but not in channelization protocols.

Q12-19. Assume the propagation delay in a broadcast network is 5 μs and the frame transmission time is 10 μs.

a. How long does it take for the first bit to reach the destination?

b. How long does it take for the last bit to reach the destination after the first bit has arrived?

c. How long is the network involved with this frame (vulnerable to collision)?

Q12-20. Assume the propagation delay in a broadcast network is 12 μs and the frame transmission time is 8 μs.

a. How long does it take for the first bit to reach the destination?

b. How long does it take for the last bit to reach the destination after the first bit has arrived?

c. How long is the network involved with this frame (vulnerable to collision)?

Q12-21. List some strategies in CSMA/CA that are used to avoid collision.

Q12-22. In a wireless LAN, station A is assigned IFS = 5 milliseconds and station B is assigned IFS = 7 milliseconds. Which station has a higher priority? Explain.

Q12-23. There is no acknowledgment mechanism in CSMA/CD, but we need this mechanism in CSMA/CA. Explain the reason.

Q12-24. What is the purpose of NAV in CSMA/CA?

12.5.3 Problems

P12-1. To formulate the performance of a multiple-access network, we need a mathematical model. When the number of stations in a network is very large, the Poisson distribution, $p[x] = (e^{-\lambda} \times \lambda^x)/(x!)$, is used. In this formula, $p[x]$ is the probability of generating x number of frames in a period of time and λ is the average number of generated frames during the same period of time. Using the Poisson distribution:

- **a.** Find the probability that a pure Aloha network generates x number of frames during the vulnerable time. Note that the vulnerable time for this network is two times the frame transmission time (T_{fr}).
- **b.** Find the probability that a slotted Aloha network generates x number of frames during the vulnerable time. Note that the vulnerable time for this network is equal to the frame transmission time (T_{fr}).

P12-2. In the previous problem, we used the Poisson distribution to find the probability of generating x number of frames, in a certain period of time, in a pure or slotted Aloha network as $p[x] = (e^{-\lambda} \times \lambda^x)/(x!)$. In this problem, we want to find the probability that a frame in such a network reaches its destination without colliding with other frames. For this purpose, it is simpler to think that we have G stations, each sending an average of one frame during the frame transmission time (instead of having N frames, each sending an average of G/N frames during the same time). Then, the probability of success for a station is the probability that no other station sends a frame during the vulnerable time.

- **a.** Find the probability that a station in a pure Aloha network can successfully send a frame during a vulnerable time.
- **b.** Find the probability that a station in a slotted Aloha network can successfully send a frame during a vulnerable time.

P12-3. In the previous problem, we found that the probability of a station (in a G-station network) successfully sending a frame in a vulnerable time is $P = e^{-2G}$ for a pure Aloha and $P = e^{-G}$ for a slotted Aloha network. In this problem, we want to find the throughput of these networks, which is the probability that any station (out of G stations) can successfully send a frame during the vulnerable time.

- **a.** Find the throughput of a pure Aloha network.
- **b.** Find the throughput of a slotted Aloha network.

P12-4. In the previous problem, we showed that the throughput is $S = Ge^{-2G}$ for a pure Aloha network and $S = Ge^{-G}$ for a slotted Aloha network. In this problem, we want to find the value of G in each network that makes the throughput maximum and find the value of the maximum throughput. This can be

done if we find the derivative of S with respect to G and set the derivative to zero.

a. Find the value of G that makes the throughput maximum, and find the value of the maximum throughput for a pure Aloha network.

b. Find the value of G that makes the throughput maximum, and find the value of the maximum throughput for a slotted Aloha network.

P12-5. A multiple access network with a large number of stations can be analyzed using the Poisson distribution. When there is a limited number of stations in a network, we need to use another approach for this analysis. In a network with N stations, we assume that each station has a frame to send during the frame transmission time (T_{fr}) with probability p. In such a network, a station is successful in sending its frame if the station has a frame to send during the vulnerable time and no other station has a frame to send during this period of time.

a. Find the probability that a station in a pure Aloha network can successfully send a frame during the vulnerable time.

b. Find the probability that a station in a slotted Aloha network can successfully send a frame during the vulnerable time.

P12-6. In the previous problem, we found the probability of success for a station to send a frame successfully during the vulnerable time. The throughput of a network with a limited number of stations is the probability that any station (out of N stations) can send a frame successfully. In other words, the throughput is the sum of N success probabilities.

a. Find the throughput of a pure Aloha network.

b. Find the throughput of a slotted Aloha network.

P12-7. In the previous problem, we found the throughputs of a pure and a slotted Aloha network as $S = Np\ (1-p)^{2(N-1)}$ and $S = Np\ (1-p)^{(N-1)}$ respectively. In this problem we want to find the maximum throughput with respect to p.

a. Find the value of p that maximizes the throughput of a pure Aloha network, and calculate the maximum throughput when N is a very large number.

b. Find the value of p that maximizes the throughput of a slotted Aloha network, and calculate the maximum throughput when N is a very large number.

P12-8. There are only three active stations in a slotted Aloha network: A, B, and C. Each station generates a frame in a time slot with the corresponding probabilities $p_A = 0.2$, $p_B = 0.3$, and $p_C = 0.4$ respectively.

a. What is the throughput of each station?

b. What is the throughput of the network?

P12-9. There are only three active stations in a slotted Aloha network: A, B, and C. Each station generates a frame in a time slot with the corresponding probabilities $p_A = 0.2$, $p_B = 0.3$, and $p_C = 0.4$ respectively.

a. What is the probability that any station can send a frame in the first slot?

b. What is the probability that station A can successfully send a frame for the first time in the second slot?

c. What is the probability that station C can successfully send a frame for the first time in the third slot?

P12-10. A slotted Aloha network is working with maximum throughput.

a. What is the probability that a slot is empty?

b. How many slots, n, on average, should pass before getting an empty slot?

P12-11. One of the useful parameters in a LAN is the number of bits that can fit in one meter of the medium ($n_{b/m}$). Find the value of $n_{b/m}$ if the data rate is 100 Mbps and the medium propagation speed is 2×10^8 m/s.

P12-12. Another useful parameter in a LAN is the bit length of the medium (L_b), which defines the number of bits that the medium can hold at any time. Find the bit length of a LAN if the data rate is 100 Mbps and the medium length in meters (L_m) for a communication between two stations is 200 m. Assume the propagation speed in the medium is 2×10^8 m/s.

P12-13. We have defined the parameter $\boldsymbol{a}$ as the number of frames that can fit the medium between two stations, or $\boldsymbol{a} = (T_p)/(T_{fr})$. Another way to define this parameter is $\boldsymbol{a} = L_b/F_b$, in which L_b is the bit length of the medium and F_b is the frame length of the medium. Show that the two definitions are equivalent.

P12-14. In a bus CSMA/CD network with a data rate of 10 Mbps, a collision occurs 20 μs after the first bit of the frame leaves the sending station. What should the length of the frame be so that the sender can detect the collision?

P12-15. Assume that there are only two stations, A and B, in a bus CSMA/CD network. The distance between the two stations is 2000 m and the propagation speed is 2×10^8 m/s. If station A starts transmitting at time t_1:

a. Does the protocol allow station B to start transmitting at time $t_1 + 8$ μs? If the answer is yes, what will happen?

b. Does the protocol allow station B to start transmitting at time $t_1 + 11$ μs? If the answer is yes, what will happen?

P12-16. There are only two stations, A and B, in a bus 1-persistence CSMA/CD network with $T_p = 25.6$ μs and $T_{fr} = 51.2$ μs. Station A has a frame to send to station B. The frame is unsuccessful two times and succeeds on the third try. Draw a time line diagram for this problem. Assume that the R is 1 and 2 respectively and ignore the time for sending a jamming signal (see Figure 12.13).

P12-17. To understand why we need to have a minimum frame size $T_{fr} = 2 \times T_p$ in a CDMA/CD network, assume we have a bus network with only two stations, A and B, in which $T_{fr} = 40$ μs and $T_p = 25$ μs. Station A starts sending a frame at time $t = 0.0$ μs and station B starts sending a frame at $t = 23.0$ μs. Answer the following questions:

a. Do frames collide?

b. If the answer to part *a* is yes, does station A detect collision?

c. If the answer to part *a* is yes, does station B detect collision?

P12-18. In a bus 1-persistence CSMA/CD with $T_p = 50$ μs and $T_{fr} = 120$ μs, there are two stations, A and B. Both stations start sending frames to each other at the same time. Since the frames collide, each station tries to retransmit. Station A

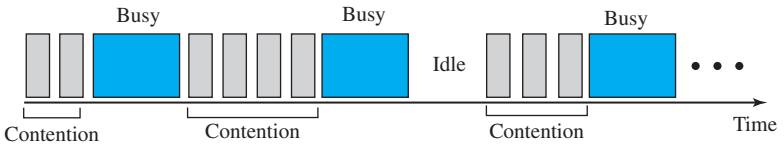
comes out with $R = 0$ and station B with $R = 1$. Ignore any other delay including the delay for sending jamming signals. Do the frames collide again? Draw a time-line diagram to prove your claim. Does the generation of a random number help avoid collision in this case?

P12-19. The random variable R (Figure 12.13) is designed to give stations different delays when a collision has occurred. To alleviate the collision, we expect that different stations generate different values of R. To show the point, find the probability that the value of R is the same for two stations after

a. the first collision. **b.** the second collision.

P12-20. Assume we have a slotted CSMA/CD network. Each station in this network uses a contention period, in which the station contends for access to the shared channel before being able to send a frame. We assume that the contention period is made of contention slots. At the beginning of each slot, the station senses the channel. If the channel is free, the station sends its frame; if the channel is busy, the station refrains from sending and waits until the beginning of the next slot. In other words, the station waits, on average, for k slots before sending its frame, as shown in Figure 12.30. Note that the channel is either in the contention state, the transmitting state, or the idle state (when no station has a frame to send). However, if N is a very large number, the idle state actually disappears.

Figure 12.30 *Problem P12-20*



a. What is the probability of a free slot (P_{free}) if the number of stations is N and each station has a frame to send with probability p?

b. What is the maximum of this probability when N is a very large number?

c. What is the probability that the jth slot is free?

d. What is the average number of slots, k, that a station should wait before getting a free slot?

e. What is the value of k when N (the number of stations) is very large?

P12-21. Although the throughput calculation of a CSMA/CD is really involved, we can calculate the maximum throughput of a slotted CSMA/CD with the specification we described in the previous problem. We found that the average number of contention slots a station needs to wait is $k = e$ slots. With this assumption, the throughput of a slotted CSMA/CD is

$$S = (T_{fr})/(\text{time the channel is busy for a frame})$$

The time the channel is busy for a frame is the time to wait for a free slot plus the time to transmit the frame plus the propagation delay to receive the good news about the lack of collision. Assume the duration of a contention slot is $2 \times (T_p)$ and $a = (T_p)/(T_{fr})$. Note that the parameter a is the number of frames that occupy the transmission media. Find the throughput of a slotted CSMA/CD in terms of the parameter a.

P12-22. We have a pure ALOHA network with a data rate of 10 Mbps. What is the maximum number of 1000-bit frames that can be successfully sent by this network?

P12-23. Check to see if the following set of chips can belong to an orthogonal system.

$$[+1, +1] \quad \text{and} \quad [+1, -1]$$

P12-24. Check to see if the following set of chips can belong to an orthogonal system.

$$[+1, +1, +1, +1] \ , \ [+1, -1, -1, +1] \ , \ [-1, +1, +1, -1] \ , \ [+1, -1, -1, +1]$$

P12-25. Alice and Bob are experimenting with CSMA using a W_2 Walsh table (see Figure 12.29). Alice uses the code [+1, +1] and Bob uses the code [+1, −1]. Assume that they simultaneously send a hexadecimal digit to each other. Alice sends $(6)_{16}$ and Bob sends $(B)_{16}$. Show how they can detect what the other person has sent.

12.6 SIMULATION EXPERIMENTS

12.6.1 Applets

We have created some Java applets to show some of the main concepts discussed in this chapter. It is strongly recommended that the students activate these applets on the book website and carefully examine the protocols in action.

12.7 PROGRAMMING ASSIGNMENTS

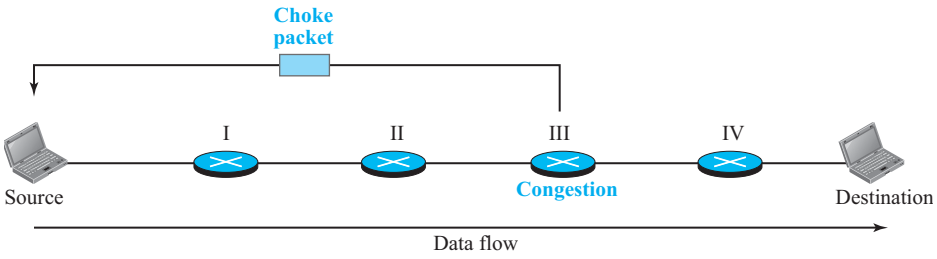
For each of the following assignments, write a program in the programming language you are familiar with.

Prg12-1. Write and test a program to simulate the flow diagram of CSMA/CD in Figure 12.13.

Prg12-2. Write and test a program to simulate the flow diagram of CSMA/CA in Figure 12.15.

Choke Packet A **choke packet** is a packet sent by a node to the source to inform it of congestion. Note the difference between the backpressure and choke-packet methods. In backpressure, the warning is from one node to its upstream node, although the warning may eventually reach the source station. In the choke-packet method, the warning is from the router, which has encountered congestion, directly to the source station. The intermediate nodes through which the packet has traveled are not warned. We will see an example of this type of control in ICMP (discussed in Chapter 19). When a router in the Internet is overwhelmed with IP datagrams, it may discard some of them, but it informs the source host, using a source quench ICMP message. The warning message goes directly to the source station; the intermediate routers do not take any action. Figure 18.15 shows the idea of a choke packet.

Figure 18.15 *Choke packet*



Implicit Signaling In implicit signaling, there is no communication between the congested node or nodes and the source. The source guesses that there is congestion somewhere in the network from other symptoms. For example, when a source sends several packets and there is no acknowledgment for a while, one assumption is that the network is congested. The delay in receiving an acknowledgment is interpreted as congestion in the network; the source should slow down. We saw this type of signaling when we discuss TCP congestion control in Chapter 24.

Explicit Signaling The node that experiences congestion can explicitly send a signal to the source or destination. The explicit-signaling method, however, is different from the choke-packet method. In the choke-packet method, a separate packet is used for this purpose; in the explicit-signaling method, the signal is included in the packets that carry data. Explicit signaling can occur in either the forward or the backward direction. This type of congestion control can be seen in an ATM network, discussed in Chapter 14.

18.4 IPV4 ADDRESSES

The identifier used in the IP layer of the TCP/IP protocol suite to identify the connection of each device to the Internet is called the Internet address or IP address. An IPv4 address is a 32-bit address that uniquely and universally defines the connection of a host or a router to the Internet. The IP address is the address of the connection, not the

host or the router, because if the device is moved to another network, the IP address may be changed.

IPv4 addresses are unique in the sense that each address defines one, and only one, connection to the Internet. If a device has two connections to the Internet, via two networks, it has two IPv4 addresses. IPv4 addresses are universal in the sense that the addressing system must be accepted by any host that wants to be connected to the Internet.

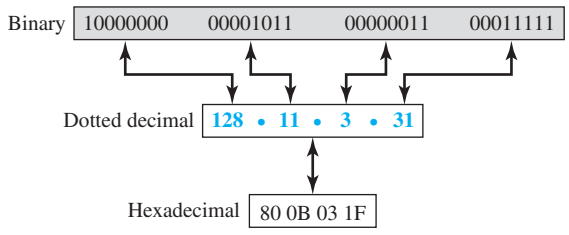
18.4.1 Address Space

A protocol like IPv4 that defines addresses has an address space. An **address space** is the total number of addresses used by the protocol. If a protocol uses b bits to define an address, the address space is 2^b because each bit can have two different values (0 or 1). IPv4 uses 32-bit addresses, which means that the address space is 2^{32} or 4,294,967,296 (more than four billion). If there were no restrictions, more than 4 billion devices could be connected to the Internet.

Notation

There are three common notations to show an IPv4 address: binary notation (base 2), dotted-decimal notation (base 256), and hexadecimal notation (base 16). In *binary notation,* an IPv4 address is displayed as 32 bits. To make the address more readable, one or more spaces are usually inserted between each octet (8 bits). Each octet is often referred to as a byte. To make the IPv4 address more compact and easier to read, it is usually written in decimal form with a decimal point (dot) separating the bytes. This format is referred to as *dotted-decimal notation*. Note that because each byte (octet) is only 8 bits, each number in the dotted-decimal notation is between 0 and 255. We sometimes see an IPv4 address in hexadecimal notation. Each hexadecimal digit is equivalent to four bits. This means that a 32-bit address has 8 hexadecimal digits. This notation is often used in network programming. Figure 18.16 shows an IP address in the three discussed notations.

Figure 18.16 *Three different notations in IPv4 addressing*



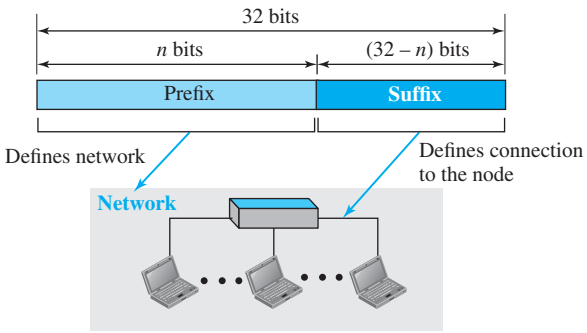
Hierarchy in Addressing

In any communication network that involves delivery, such as a telephone network or a postal network, the addressing system is hierarchical. In a postal network, the postal address (mailing address) includes the country, state, city, street, house number, and the

name of the mail recipient. Similarly, a telephone number is divided into the country code, area code, local exchange, and the connection.

A 32-bit IPv4 address is also hierarchical, but divided only into two parts. The first part of the address, called the *prefix*, defines the network; the second part of the address, called the *suffix*, defines the node (connection of a device to the Internet). Figure 18.17 shows the prefix and suffix of a 32-bit IPv4 address. The prefix length is n bits and the suffix length is $(32 - n)$ bits.

Figure 18.17 *Hierarchy in addressing*



A prefix can be fixed length or variable length. The network identifier in the IPv4 was first designed as a fixed-length prefix. This scheme, which is now obsolete, is referred to as classful addressing. The new scheme, which is referred to as classless addressing, uses a variable-length network prefix. First, we briefly discuss classful addressing; then we concentrate on classless addressing.

18.4.2 Classful Addressing

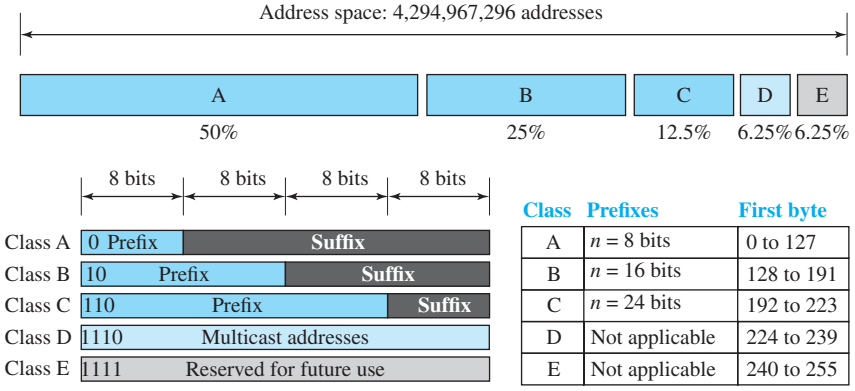
When the Internet started, an IPv4 address was designed with a fixed-length prefix, but to accommodate both small and large networks, three fixed-length prefixes were designed instead of one ($n = 8$, $n = 16$, and $n = 24$). The whole address space was divided into five classes (class A, B, C, D, and E), as shown in Figure 18.18. This scheme is referred to as **classful addressing.** Although classful addressing belongs to the past, it helps us to understand classless addressing, discussed later.

In class A, the network length is 8 bits, but since the first bit, which is 0, defines the class, we can have only seven bits as the network identifier. This means there are only $2^7 = 128$ networks in the world that can have a class A address.

In class B, the network length is 16 bits, but since the first two bits, which are $(10)_2$, define the class, we can have only 14 bits as the network identifier. This means there are only $2^{14} = 16{,}384$ networks in the world that can have a class B address.

All addresses that start with $(110)_2$ belong to class C. In class C, the network length is 24 bits, but since three bits define the class, we can have only 21 bits as the network identifier. This means there are $2^{21} = 2{,}097{,}152$ networks in the world that can have a class C address.

Figure 18.18 *Occupation of the address space in classful addressing*



Class	Prefixes	First byte
A	n = 8 bits	0 to 127
B	n = 16 bits	128 to 191
C	n = 24 bits	192 to 223
D	Not applicable	224 to 239
E	Not applicable	240 to 255

Class D is not divided into prefix and suffix. It is used for multicast addresses. All addresses that start with 1111 in binary belong to class E. As in Class D, Class E is not divided into prefix and suffix and is used as reserve.

Address Depletion

The reason that classful addressing has become obsolete is address depletion. Since the addresses were not distributed properly, the Internet was faced with the problem of the addresses being rapidly used up, resulting in no more addresses available for organizations and individuals that needed to be connected to the Internet. To understand the problem, let us think about class A. This class can be assigned to only 128 organizations in the world, but each organization needs to have a single network (seen by the rest of the world) with 16,777,216 nodes (computers in this single network). Since there may be only a few organizations that are this large, most of the addresses in this class were wasted (unused). Class B addresses were designed for midsize organizations, but many of the addresses in this class also remained unused. Class C addresses have a completely different flaw in design. The number of addresses that can be used in each network (256) was so small that most companies were not comfortable using a block in this address class. Class E addresses were almost never used, wasting the whole class.

Subnetting and Supernetting

To alleviate address depletion, two strategies were proposed and, to some extent, implemented: subnetting and supernetting. In subnetting, a class A or class B block is divided into several subnets. Each subnet has a larger prefix length than the original network. For example, if a network in class A is divided into four subnets, each subnet has a prefix of $n_{sub} = 10$. At the same time, if all of the addresses in a network are not used, subnetting allows the addresses to be divided among several organizations. This idea did not work because most large organizations were not happy about dividing the block and giving some of the unused addresses to smaller organizations.

While subnetting was devised to divide a large block into smaller ones, supernetting was devised to combine several class C blocks into a larger block to be attractive to

organizations that need more than the 256 addresses available in a class C block. This idea did not work either because it makes the routing of packets more difficult.

Advantage of Classful Addressing

Although classful addressing had several problems and became obsolete, it had one advantage: Given an address, we can easily find the class of the address and, since the prefix length for each class is fixed, we can find the prefix length immediately. In other words, the prefix length in classful addressing is inherent in the address; no extra information is needed to extract the prefix and the suffix.

18.4.3 Classless Addressing

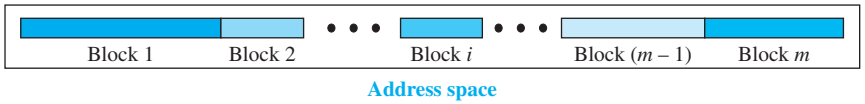
Subnetting and supernetting in classful addressing did not really solve the address depletion problem. With the growth of the Internet, it was clear that a larger address space was needed as a long-term solution. The larger address space, however, requires that the length of IP addresses also be increased, which means the format of the IP packets needs to be changed. Although the long-range solution has already been devised and is called IPv6 (discussed later), a short-term solution was also devised to use the same address space but to change the distribution of addresses to provide a fair share to each organization. The short-term solution still uses IPv4 addresses, but it is called *classless addressing*. In other words, the class privilege was removed from the distribution to compensate for the address depletion.

There was another motivation for classless addressing. During the 1990s, Internet Service Providers (ISPs) came into prominence. An ISP is an organization that provides Internet access for individuals, small businesses, and midsize organizations that do not want to create an Internet site and become involved in providing Internet services (such as electronic mail) for their employees. An ISP can provide these services. An ISP is granted a large range of addresses and then subdivides the addresses (in groups of 1, 2, 4, 8, 16, and so on), giving a range of addresses to a household or a small business. The customers are connected via a dial-up modem, DSL, or cable modem to the ISP. However, each customer needs some IPv4 addresses.

In 1996, the Internet authorities announced a new architecture called **classless addressing.** In classless addressing, variable-length blocks are used that belong to no classes. We can have a block of 1 address, 2 addresses, 4 addresses, 128 addresses, and so on.

In classless addressing, the whole address space is divided into variable length blocks. The prefix in an address defines the block (network); the suffix defines the node (device). Theoretically, we can have a block of $2^0, 2^1, 2^2, \ldots, 2^{32}$ addresses. One of the restrictions, as we discuss later, is that the number of addresses in a block needs to be a power of 2. An organization can be granted one block of addresses. Figure 18.19 shows the division of the whole address space into nonoverlapping blocks.

Figure 18.19 *Variable-length blocks in classless addressing*



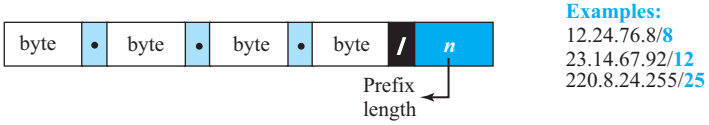
Unlike classful addressing, the prefix length in classless addressing is variable. We can have a prefix length that ranges from 0 to 32. The size of the network is inversely proportional to the length of the prefix. A small prefix means a larger network; a large prefix means a smaller network.

We need to emphasize that the idea of classless addressing can be easily applied to classful addressing. An address in class A can be thought of as a classless address in which the prefix length is 8. An address in class B can be thought of as a classless address in which the prefix is 16, and so on. In other words, classful addressing is a special case of classless addressing.

Prefix Length: Slash Notation

The first question that we need to answer in classless addressing is how to find the prefix length if an address is given. Since the prefix length is not inherent in the address, we need to separately give the length of the prefix. In this case, the prefix length, *n*, is added to the address, separated by a slash. The notation is informally referred to as *slash notation* and formally as ***classless interdomain routing*** or ***CIDR*** (pronounced cider) strategy. An address in classless addressing can then be represented as shown in Figure 18.20.

Figure 18.20 *Slash notation (CIDR)*



In other words, an address in classless addressing does not, per se, define the block or network to which the address belongs; we need to give the prefix length also.

Extracting Information from an Address

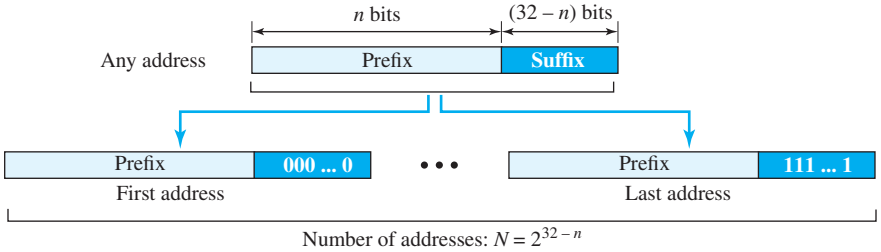
Given any address in the block, we normally like to know three pieces of information about the block to which the address belongs: the number of addresses, the first address in the block, and the last address. Since the value of prefix length, *n*, is given, we can easily find these three pieces of information, as shown in Figure 18.21.

1. The number of addresses in the block is found as $N = 2^{32-n}$.
2. To find the first address, we keep the *n* leftmost bits and set the (32 – *n*) rightmost bits all to 0s.
3. To find the last address, we keep the *n* leftmost bits and set the (32 – *n*) rightmost bits all to 1s.

Example 18.1

A classless address is given as 167.199.170.82/27. We can find the above three pieces of information as follows. The number of addresses in the network is $2^{32-n} = 2^5 = 32$ addresses.

Figure 18.21 *Information extraction in classless addressing*



The first address can be found by keeping the first 27 bits and changing the rest of the bits to 0s.

Address: 167.199.170.82/27	10100111	11000111	10101010	01010010
First address: 167.199.170.64/27	10100111	11000111	10101010	01000000

The last address can be found by keeping the first 27 bits and changing the rest of the bits to 1s.

Address: 167.199.170.82/27	10100111	11000111	10101010	01011111
Last address: 167.199.170.95/27	10100111	11000111	10101010	01011111

Address Mask

Another way to find the first and last addresses in the block is to use the address mask. The address mask is a 32-bit number in which the n leftmost bits are set to 1s and the rest of the bits $(32 - n)$ are set to 0s. A computer can easily find the address mask because it is the complement of $(2^{32-n} - 1)$. The reason for defining a mask in this way is that it can be used by a computer program to extract the information in a block, using the three bit-wise operations NOT, AND, and OR.

1. The number of addresses in the block N = **NOT** (mask) + 1.
2. The first address in the block = (Any address in the block) **AND** (mask).
3. The last address in the block = (Any address in the block) **OR** [(**NOT** (mask)].

Example 18.2

We repeat Example 18.1 using the mask. The mask in dotted-decimal notation is 256.256.256.224. The AND, OR, and NOT operations can be applied to individual bytes using calculators and applets at the book website.

Number of addresses in the block:	N = **NOT** (mask) + 1 = 0.0.0.31 + 1 = 32 addresses
First address:	First = (address) **AND** (mask) = 167.199.170.82
Last address:	Last = (address) **OR** (**NOT** mask) = 167.199.170.255

Example 18.3

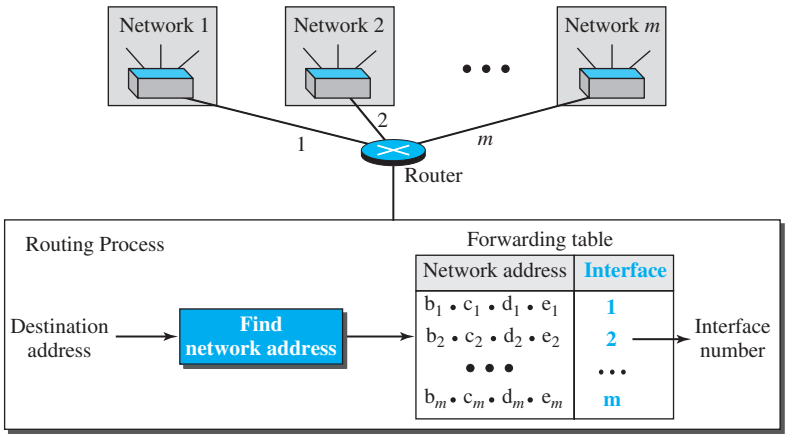
In classless addressing, an address cannot per se define the block the address belongs to. For example, the address 230.8.24.56 can belong to many blocks. Some of them are shown below with the value of the prefix associated with that block.

Prefix length:16	→	Block:	230.8.0.0	to	230.8.255.255
Prefix length:20	→	Block:	230.8.16.0	to	230.8.31.255
Prefix length:26	→	Block:	230.8.24.0	to	230.8.24.63
Prefix length:27	→	Block:	230.8.24.32	to	230.8.24.63
Prefix length:29	→	Block:	230.8.24.56	to	230.8.24.63
Prefix length:31	→	Block:	230.8.24.56	to	230.8.24.57

Network Address

The above examples show that, given any address, we can find all information about the block. The first address, the **network address,** is particularly important because it is used in routing a packet to its destination network. For the moment, let us assume that an internet is made of *m* networks and a router with *m* interfaces. When a packet arrives at the router from any source host, the router needs to know to which network the packet should be sent: from which interface the packet should be sent out. When the packet arrives at the network, it reaches its destination host using another strategy that we discuss later. Figure 18.22 shows the idea. After the network address has been

Figure 18.22 *Network address*



found, the router consults its forwarding table to find the corresponding interface from which the packet should be sent out. The network address is actually the identifier of the network; each network is identified by its network address.

Block Allocation

The next issue in classless addressing is block allocation. How are the blocks allocated? The ultimate responsibility of block allocation is given to a global authority called the Internet Corporation for Assigned Names and Numbers (ICANN). However, ICANN does not normally allocate addresses to individual Internet users. It assigns a large block of addresses to an ISP (or a larger organization that is considered an ISP in this case). For the proper operation of the CIDR, two restrictions need to be applied to the allocated block.

1. The number of requested addresses, N, needs to be a power of 2. The reason is that $N = 2^{32-n}$ or $n = 32 - \log_2 N$. If N is not a power of 2, we cannot have an integer value for n.
2. The requested block needs to be allocated where there is an adequate number of contiguous addresses available in the address space. However, there is a restriction on choosing the first address in the block. The first address needs to be divisible by the number of addresses in the block. The reason is that the first address needs to be the prefix followed by $(32 - n)$ number of 0s. The decimal value of the first address is then

$$\text{first address} = (\text{prefix in decimal}) \times 2^{32-n} = (\text{prefix in decimal}) \times N.$$

Example 18.4

An ISP has requested a block of 1000 addresses. Since 1000 is not a power of 2, 1024 addresses are granted. The prefix length is calculated as $n = 32 - \log_2 1024 = 22$. An available block, 18.14.12.0/**22**, is granted to the ISP. It can be seen that the first address in decimal is 302,910,464, which is divisible by 1024.

Subnetting

More levels of hierarchy can be created using subnetting. An organization (or an ISP) that is granted a range of addresses may divide the range into several subranges and assign each subrange to a subnetwork (or subnet). Note that nothing stops the organization from creating more levels. A subnetwork can be divided into several sub-subnetworks. A sub-subnetwork can be divided into several sub-sub-subnetworks, and so on.

Designing Subnets

The subnetworks in a network should be carefully designed to enable the routing of packets. We assume the total number of addresses granted to the organization is N, the prefix length is n, the assigned number of addresses to each subnetwork is N_{sub}, and the prefix length for each subnetwork is n_{sub}. Then the following steps need to be carefully followed to guarantee the proper operation of the subnetworks.

- The number of addresses in each subnetwork should be a power of 2.
- The prefix length for each subnetwork should be found using the following formula:

$$n_{sub} = 32 - \log_2 N_{sub}$$

- The starting address in each subnetwork should be divisible by the number of addresses in that subnetwork. This can be achieved if we first assign addresses to larger subnetworks.

Finding Information about Each Subnetwork

After designing the subnetworks, the information about each subnetwork, such as first and last address, can be found using the process we described to find the information about each network in the Internet.

Example 18.5

An organization is granted a block of addresses with the beginning address 14.24.74.0/24. The organization needs to have 3 subblocks of addresses to use in its three subnets: one subblock of 10 addresses, one subblock of 60 addresses, and one subblock of 120 addresses. Design the subblocks.

Solution

There are $2^{32-24} = 256$ addresses in this block. The first address is 14.24.74.0/24; the last address is 14.24.74.255/24. To satisfy the third requirement, we assign addresses to subblocks, starting with the largest and ending with the smallest one.

a. The number of addresses in the largest subblock, which requires 120 addresses, is not a power of 2. We allocate 128 addresses. The subnet mask for this subnet can be found as $n_1 = 32 - \log_2 128 = 25$. The first address in this block is 14.24.74.0/25; the last address is 14.24.74.127/25.

b. The number of addresses in the second largest subblock, which requires 60 addresses, is not a power of 2 either. We allocate 64 addresses. The subnet mask for this subnet can be found as $n_2 = 32 - \log_2 64 = 26$. The first address in this block is 14.24.74.128/26; the last address is 14.24.74.191/26.

c. The number of addresses in the smallest subblock, which requires 10 addresses, is not a power of 2 either. We allocate 16 addresses. The subnet mask for this subnet can be found as $n_3 = 32 - \log_2 16 = 28$. The first address in this block is 14.24.74.192/28; the last address is 14.24.74.207/28.

If we add all addresses in the previous subblocks, the result is 208 addresses, which means 48 addresses are left in reserve. The first address in this range is 14.24.74.208. The last address is 14.24.74.255. We don't know about the prefix length yet. Figure 18.23 shows the configuration of blocks. We have shown the first address in each block.

Address Aggregation

One of the advantages of the CIDR strategy is **address aggregation** (sometimes called *address summarization* or *route summarization*). When blocks of addresses are combined to create a larger block, routing can be done based on the prefix of the larger block. ICANN assigns a large block of addresses to an ISP. Each ISP in turn divides its assigned block into smaller subblocks and grants the subblocks to its customers.

Example 18.6

Figure 18.24 shows how four small blocks of addresses are assigned to four organizations by an ISP. The ISP combines these four blocks into one single block and advertises the larger block to the rest of the world. Any packet destined for this larger block should be sent to this ISP. It is the responsibility of the ISP to forward the packet to the appropriate organization. This is similar to

Figure 18.23 *Solution to Example 18.5*

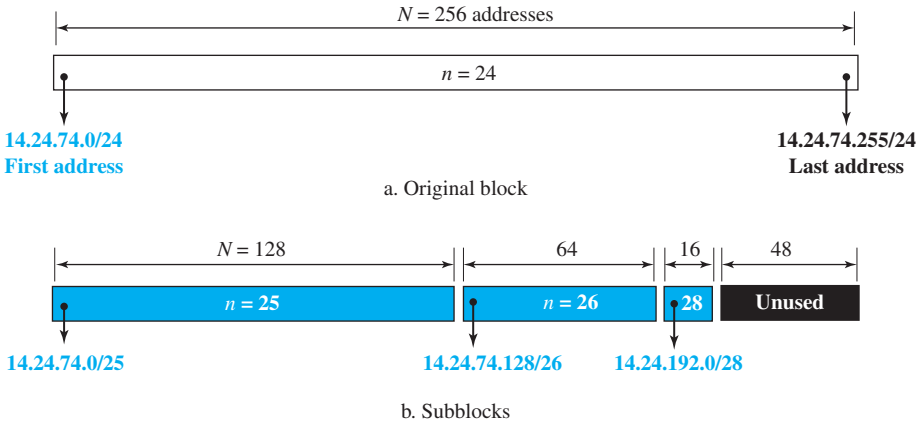
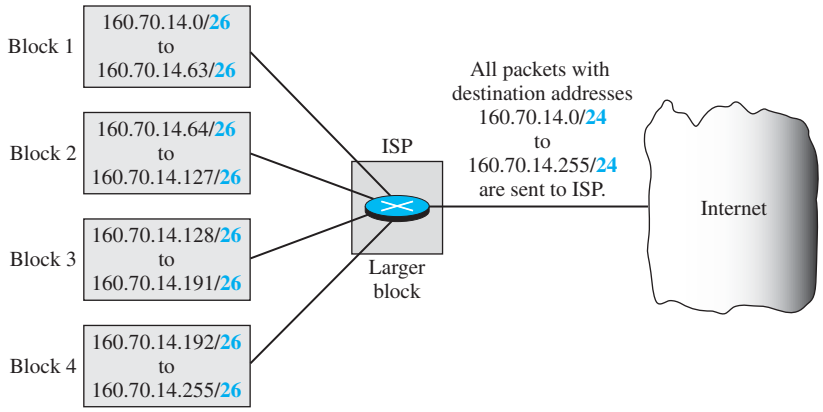


Figure 18.24 *Example of address aggregation*



routing we can find in a postal network. All packages coming from outside a country are sent first to the capital and then distributed to the corresponding destination.

Special Addresses

Before finishing the topic of addresses in IPv4, we need to mention five special addresses that are used for special purposes: *this-host* address, *limited-broadcast* address, *loopback* address, *private* addresses, and *multicast* addresses.

This-host Address

The only address in the block **0.0.0.0/32** is called the *this-host* address. It is used whenever a host needs to send an IP datagram but it does not know its own address to use as the source address. We will see an example of this case in the next section.

Limited-broadcast Address

The only address in the block **255.255.255.255/32** is called the *limited-broadcast* address. It is used whenever a router or a host needs to send a datagram to all devices in a network. The routers in the network, however, block the packet having this address as the destination; the packet cannot travel outside the network.

Loopback Address

The block **127.0.0.0/8** is called the *loopback* address. A packet with one of the addresses in this block as the destination address never leaves the host; it will remain in the host. Any address in the block is used to test a piece of software in the machine. For example, we can write a client and a server program in which one of the addresses in the block is used as the server address. We can test the programs using the same host to see if they work before running them on different computers.

Private Addresses

Four blocks are assigned as private addresses: 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, and 169.254.0.0/16. We will see the applications of these addresses when we discuss NAT later in the chapter.

Multicast Addresses

The block 224.0.0.0/4 is reserved for multicast addresses. We discuss these addresses later in the chapter.

18.4.4 Dynamic Host Configuration Protocol (DHCP)

We have seen that a large organization or an ISP can receive a block of addresses directly from ICANN and a small organization can receive a block of addresses from an ISP. After a block of addresses are assigned to an organization, the network administration can manually assign addresses to the individual hosts or routers. However, address assignment in an organization can be done automatically using the **Dynamic Host Configuration Protocol (DHCP).** DHCP is an application-layer program, using the client-server paradigm, that actually helps TCP/IP at the network layer.

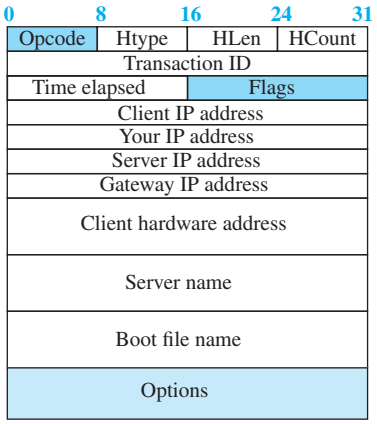
DHCP has found such widespread use in the Internet that it is often called a *plug-and-play protocol*. In can be used in many situations. A network manager can configure DHCP to assign permanent IP addresses to the host and routers. DHCP can also be configured to provide temporary, on demand, IP addresses to hosts. The second capability can provide a temporary IP address to a traveller to connect her laptop to the Internet while she is staying in the hotel. It also allows an ISP with 1000 granted addresses to provide services to 4000 households, assuming not more than one-forth of customers use the Internet at the same time.

In addition to its IP address, a computer also needs to know the network prefix (or address mask). Most computers also need two other pieces of information, such as the address of a default router to be able to communicate with other networks and the address of a name server to be able to use names instead of addresses, as we will see in Chapter 26. In other words, four pieces of information are normally needed: the computer address, the prefix, the address of a router, and the IP address of a name server. DHCP can be used to provide these pieces of information to the host.

DHCP Message Format

DHCP is a client-server protocol in which the client sends a request message and the server returns a response message. Before we discuss the operation of DHCP, let us show the general format of the DHCP message in Figure 18.25. Most of the fields are explained in the figure, but we need to discuss the option field, which plays a very important role in DHCP.

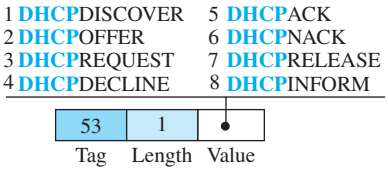
Figure 18.25 *DHCP message format*



Fields:
Opcode: Operation code, request (1) or reply (2)
Htype: Hardware type (Ethernet, ...)
HLen: Length of hardware address
HCount: Maximum number of hops the packet can travel
Transaction ID: An integer set by the client and repeated by the server
Time elapsed: The number of seconds since the client started to boot
Flags: First bit defines unicast (0) or multicast (1); other 15 bits not used
Client IP address: Set to 0 if the client does not know it
Your IP address: The client IP address sent by the server
Server IP address: A broadcast IP address if client does not know it
Gateway IP address: The address of default router
Server name: A 64-byte domain name of the server
Boot file name: A 128-byte file name holding extra information
Options: A 64-byte field with dual purpose described in text

The 64-byte option field has a dual purpose. It can carry either additional information or some specific vendor information. The server uses a number, called a **magic cookie,** in the format of an IP address with the value of 99.130.83.99. When the client finishes reading the message, it looks for this magic cookie. If present, the next 60 bytes are options. An option is composed of three fields: a 1-byte tag field, a 1-byte length field, and a variable-length value field. There are several tag fields that are mostly used by vendors. If the tag field is 53, the value field defines one of the 8 message types shown in Figure 18.26. We show how these message types are used by DHCP.

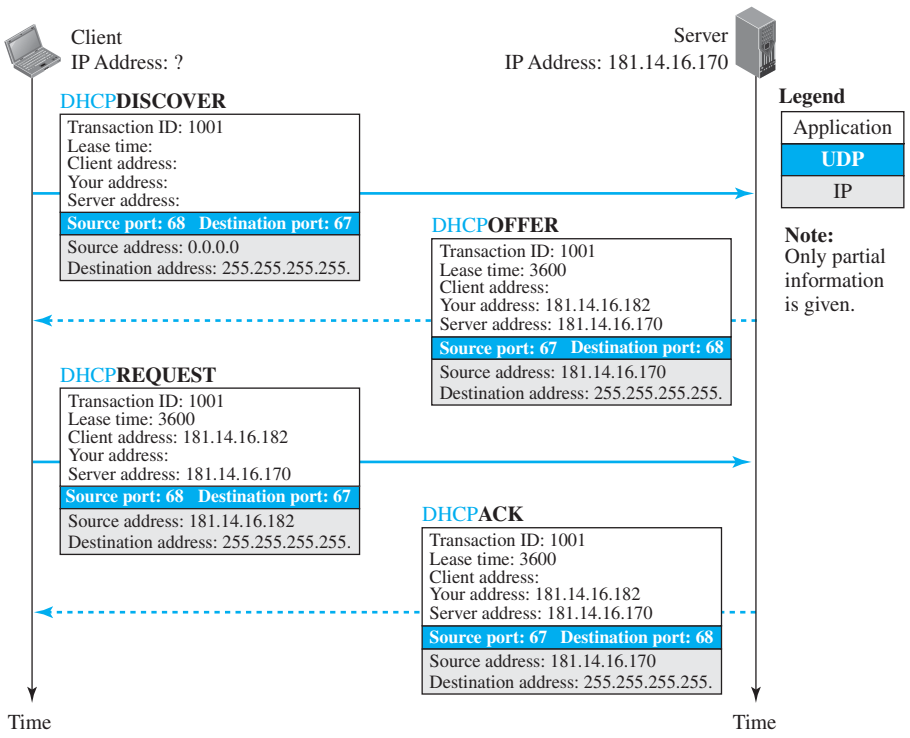
Figure 18.26 *Option format*



DHCP Operation

Figure 18.27 shows a simple scenario.

Figure 18.27 *Operation of DHCP*



1. The joining host creates a DHCPDISCOVER message in which only the transaction-ID field is set to a random number. No other field can be set because the host has no knowledge with which to do so. This message is encapsulated in a UDP user datagram with the source port set to 68 and the destination port set to 67. We will discuss the reason for using two well-known port numbers later. The user datagram is encapsulated in an IP datagram with the source address set to **0.0.0.0** ("this host") and the destination address set to **255.255.255.255** (broadcast address). The reason is that the joining host knows neither its own address nor the server address.
2. The DHCP server or servers (if more than one) responds with a DHCPOFFER message in which the your address field defines the offered IP address for the joining host and the server address field includes the IP address of the server. The message also includes the lease time for which the host can keep the IP address. This message is encapsulated in a user datagram with the same port numbers, but in the reverse order. The user datagram in turn is encapsulated in a datagram with the server address as the source IP address, but the destination address is a broadcast address, in which the server allows other DHCP servers to receive the offer and give a better offer if they can.

3. The joining host receives one or more offers and selects the best of them. The joining host then sends a **DHCP**REQUEST message to the server that has given the best offer. The fields with known value are set. The message is encapsulated in a user datagram with port numbers as the first message. The user datagram is encapsulated in an IP datagram with the source address set to the new client address, but the destination address still is set to the broadcast address to let the other servers know that their offer was not accepted.
4. Finally, the selected server responds with a **DHCP**ACK message to the client if the offered IP address is valid. If the server cannot keep its offer (for example, if the address is offered to another host in between), the server sends a **DHCP**NACK message and the client needs to repeat the process. This message is also broadcast to let other servers know that the request is accepted or rejected.

Two Well-Known Ports

We said that the DHCP uses two well-known ports (68 and 67) instead of one well-known and one ephemeral. The reason for choosing the well-known port 68 instead of an ephemeral port for the client is that the response from the server to the client is broadcast. Remember that an IP datagram with the limited broadcast message is delivered to every host on the network. Now assume that a DHCP client and a DAYTIME client, for example, are both waiting to receive a response from their corresponding server and both have accidentally used the same temporary port number (56017, for example). Both hosts receive the response message from the DHCP server and deliver the message to their clients. The DHCP client processes the message; the DAYTIME client is totally confused with a strange message received. Using a well-known port number prevents this problem from happening. The response message from the DHCP server is not delivered to the DAYTIME client, which is running on the port number 56017, not 68. The temporary port numbers are selected from a different range than the well-known port numbers.

The curious reader may ask what happens if two DHCP clients are running at the same time. This can happen after a power failure and power restoration. In this case the messages can be distinguished by the value of the transaction ID, which separates each response from the other.

Using FTP

The server does not send all of the information that a client may need for joining the network. In the **DHCP**ACK message, the server defines the pathname of a file in which the client can find complete information such as the address of the DNS server. The client can then use a file transfer protocol to obtain the rest of the needed information.

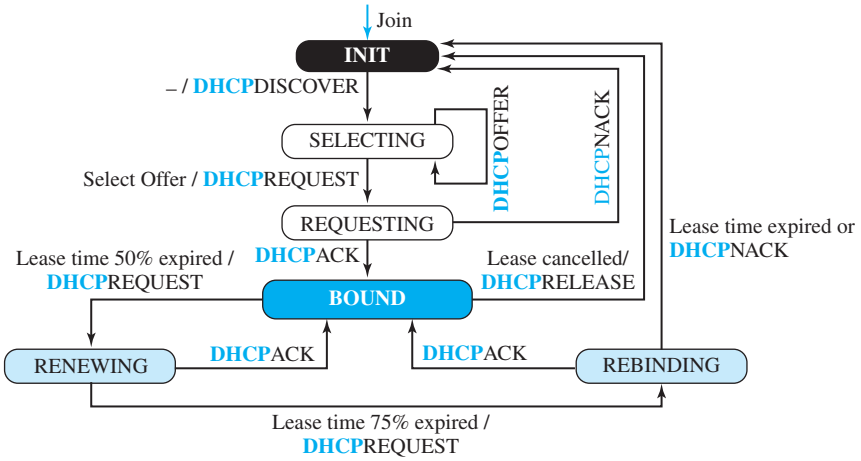
Error Control

DHCP uses the service of UDP, which is not reliable. To provide error control, DHCP uses two strategies. First, DHCP requires that UDP use the checksum. As we will see in Chapter 24, the use of the checksum in UDP is optional. Second, the DHCP client uses timers and a retransmission policy if it does not receive the DHCP reply to a request. However, to prevent a traffic jam when several hosts need to retransmit a request (for example, after a power failure), DHCP forces the client to use a random number to set its timers.

Transition States

The previous scenarios we discussed for the operation of the DHCP were very simple. To provide dynamic address allocation, the DHCP client acts as a state machine that performs transitions from one state to another depending on the messages it receives or sends. Figure 18.28 shows the transition diagram with the main states.

Figure 18.28 *FSM for the DHCP client*



When the DHCP client first starts, it is in the INIT state (initializing state). The client broadcasts a discover message. When it receives an offer, the client goes to the SELECTING state. While it is there, it may receive more offers. After it selects an offer, it sends a request message and goes to the REQUESTING state. If an ACK arrives while the client is in this state, it goes to the BOUND state and uses the IP address. When the lease is 50 percent expired, the client tries to renew it by moving to the RENEWING state. If the server renews the lease, the client moves to the BOUND state again. If the lease is not renewed and the lease time is 75 percent expired, the client moves to the REBINDING state. If the server agrees with the lease (ACK message arrives), the client moves to the BOUND state and continues using the IP address; otherwise, the client moves to the INIT state and requests another IP address. Note that the client can use the IP address only when it is in the BOUND, RENEWING, or REBINDING state. The above procedure requires that the client uses three timers: *renewal timer* (set to 50 percent of the lease time), *rebinding timer* (set to 75 percent of the lease time), and *expiration timer* (set to the lease time).

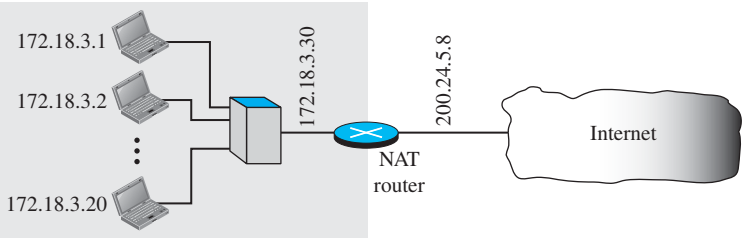
18.4.5 Network Address Resolution (NAT)

The distribution of addresses through ISPs has created a new problem. Assume that an ISP has granted a small range of addresses to a small business or a household. If the business grows or the household needs a larger range, the ISP may not be able to grant the demand because the addresses before and after the range may have already been allocated to other networks. In most situations, however, only a portion of computers in

a small network need access to the Internet simultaneously. This means that the number of allocated addresses does not have to match the number of computers in the network. For example, assume that in a small business with 20 computers the maximum number of computers that access the Internet simultaneously is only 4. Most of the computers are either doing some task that does not need Internet access or communicating with each other. This small business can use the TCP/IP protocol for both internal and universal communication. The business can use 20 (or 25) addresses from the private block addresses (discussed before) for internal communication; five addresses for universal communication can be assigned by the ISP.

A technology that can provide the mapping between the private and universal addresses, and at the same time support virtual private networks, which we discuss in Chapter 32, is **Network Address Translation (NAT).** The technology allows a site to use a set of private addresses for internal communication and a set of global Internet addresses (at least one) for communication with the rest of the world. The site must have only one connection to the global Internet through a NAT-capable router that runs NAT software. Figure 18.29 shows a simple implementation of NAT.

Figure 18.29 *NAT*



As the figure shows, the private network uses private addresses. The router that connects the network to the global address uses one private address and one global address. The private network is invisible to the rest of the Internet; the rest of the Internet sees only the NAT router with the address 200.24.5.8.

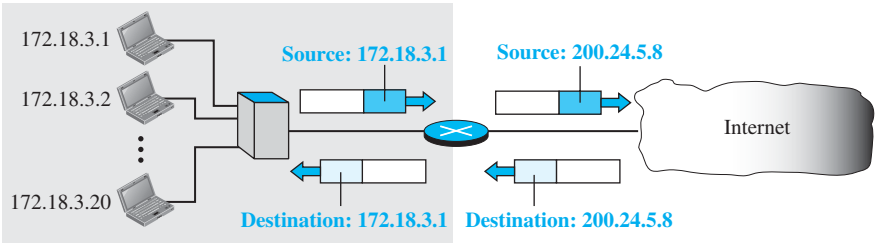
Address Translation

All of the outgoing packets go through the NAT router, which replaces the source address in the packet with the global NAT address. All incoming packets also pass through the NAT router, which replaces the destination address in the packet (the NAT router global address) with the appropriate private address. Figure 18.30 shows an example of address translation.

Translation Table

The reader may have noticed that translating the source addresses for an outgoing packet is straightforward. But how does the NAT router know the destination address for a packet coming from the Internet? There may be tens or hundreds of private IP

Figure 18.30 *Address translation*

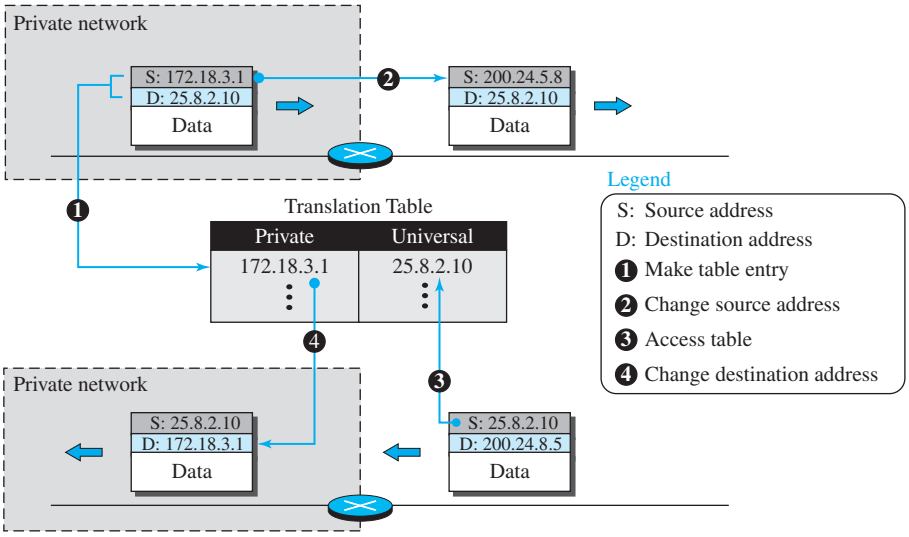


addresses, each belonging to one specific host. The problem is solved if the NAT router has a translation table.

Using One IP Address

In its simplest form, a translation table has only two columns: the private address and the external address (destination address of the packet). When the router translates the source address of the outgoing packet, it also makes note of the destination address—where the packet is going. When the response comes back from the destination, the router uses the source address of the packet (as the external address) to find the private address of the packet. Figure 18.31 shows the idea.

Figure 18.31 *Translation*



In this strategy, communication must always be initiated by the private network. The NAT mechanism described requires that the private network start the communication.

As we will see, NAT is used mostly by ISPs that assign a single address to a customer. The customer, however, may be a member of a private network that has many private addresses. In this case, communication with the Internet is always initiated from the customer site, using a client program such as HTTP, TELNET, or FTP to access the corresponding server program. For example, when e-mail that originates from outside the network site is received by the ISP e-mail server, it is stored in the mailbox of the customer until retrieved with a protocol such as POP.

Using a Pool of IP Addresses

The use of only one global address by the NAT router allows only one private-network host to access a given external host. To remove this restriction, the NAT router can use a pool of global addresses. For example, instead of using only one global address (200.24.5.8), the NAT router can use four addresses (200.24.5.8, 200.24.5.9, 200.24.5.10, and 200.24.5.11). In this case, four private-network hosts can communicate with the same external host at the same time because each pair of addresses defines a separate connection. However, there are still some drawbacks. No more than four connections can be made to the same destination. No private-network host can access two external server programs (e.g., HTTP and TELNET) at the same time. And, likewise, two private-network hosts cannot access the same external server program (e.g., HTTP or TELNET) at the same time.

Using Both IP Addresses and Port Addresses

To allow a many-to-many relationship between private-network hosts and external server programs, we need more information in the translation table. For example, suppose two hosts inside a private network with addresses 172.18.3.1 and 172.18.3.2 need to access the HTTP server on external host 25.8.3.2. If the translation table has five columns, instead of two, that include the source and destination port addresses and the transport-layer protocol, the ambiguity is eliminated. Table 18.1 shows an example of such a table.

Table 18.1 *Five-column translation table*

Private address	*Private port*	*External address*	*External port*	*Transport protocol*
172.18.3.1	1400	25.8.3.2	80	TCP
172.18.3.2	1401	25.8.3.2	80	TCP
⋮	⋮	⋮	⋮	⋮

Note that when the response from HTTP comes back, the combination of source address (25.8.3.2) and destination port address (1401) defines the private network host to which the response should be directed. Note also that for this translation to work, the ephemeral port addresses (1400 and 1401) must be unique.

18.5 FORWARDING OF IP PACKETS

We discussed the concept of forwarding at the network layer earlier in this chapter. In this section, we extend the concept to include the role of IP addresses in forwarding. As we discussed before, forwarding means to place the packet in its route to its destination.