

## CHAPTER 12

# *Multiple Access*

In Chapter 11 we discussed data link control, a mechanism which provides a link with reliable communication. In the protocols we described, we assumed that there is an available dedicated link (or channel) between the sender and the receiver. This assumption may or may not be true. If, indeed, we have a dedicated link, as when we connect to the Internet using PPP as the data link control protocol, then the assumption is true and we do not need anything else.

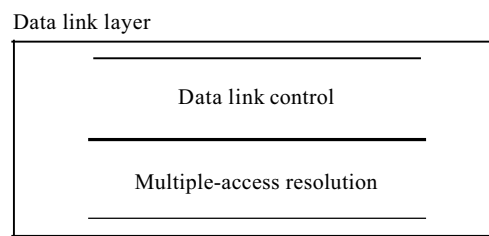
On the other hand, if we use our cellular phone to connect to another cellular phone, the channel (the band allocated to the vendor company) is not dedicated. A person a few feet away from us may be using the same channel to talk to her friend.

We can consider the data link layer as two sublayers. The upper sublayer is responsible for data link control, and the lower sublayer is responsible for resolving access to the shared media. If the channel is dedicated, we do not need the lower sublayer. Figure 12.1 shows these two sublayers in the data link layer.

---

Figure 12.1 *Data link layer divided into two functionality-oriented sublayers*

---



---

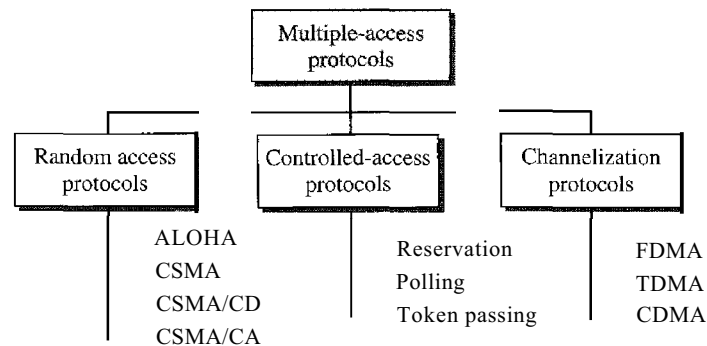
We will see in Chapter 13 that the IEEE has actually made this division for LANs. The upper sublayer that is responsible for flow and error control is called the logical link control (LLC) layer; the lower sublayer that is mostly responsible for multiple-access resolution is called the media access control (MAC) layer.

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.

The situation is similar for multipoint networks. Many formal protocols have been devised to handle access to a shared link. We categorize them into three groups. Protocols belonging to each group are shown in Figure 12.2.

Figure 12.2 Taxonomy of multiple-access protocols discussed in this chapter



## 12.1 RANDOM ACCESS

In random access or contention methods, no station is superior to another station and none is assigned the control over another. No station permits, or does not permit, another station to send. 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 the testing of 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. This method later evolved into two parallel methods: carrier sense multiple access with collision detection (CSMA/CD) and carrier sense multiple access with collision avoidance (CSMA/CA). CSMA/CD tells the station what to do when a collision is detected. CSMA/CA tries to avoid the collision.

## ALOHA

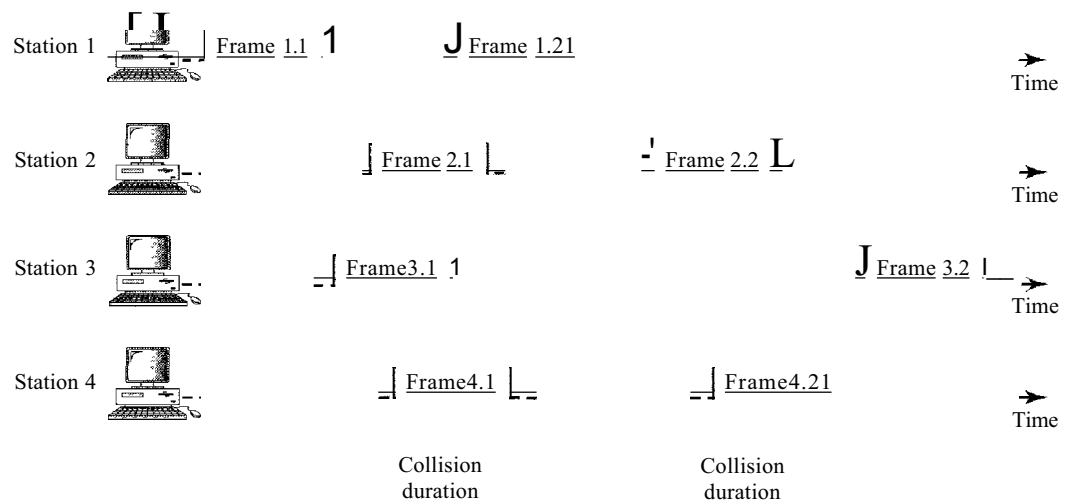
ALOHA, the earliest random access method, was developed at the University of Hawaii in early 1970. It was designed for a radio (wireless) LAN, but it can be used on any shared medium.

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. However, since there is only one channel to share, there is the possibility of collision between frames from different stations. Figure 12.3 shows an example of frame collisions in pure ALOHA.

Figure 12.3 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 12.3 shows that only

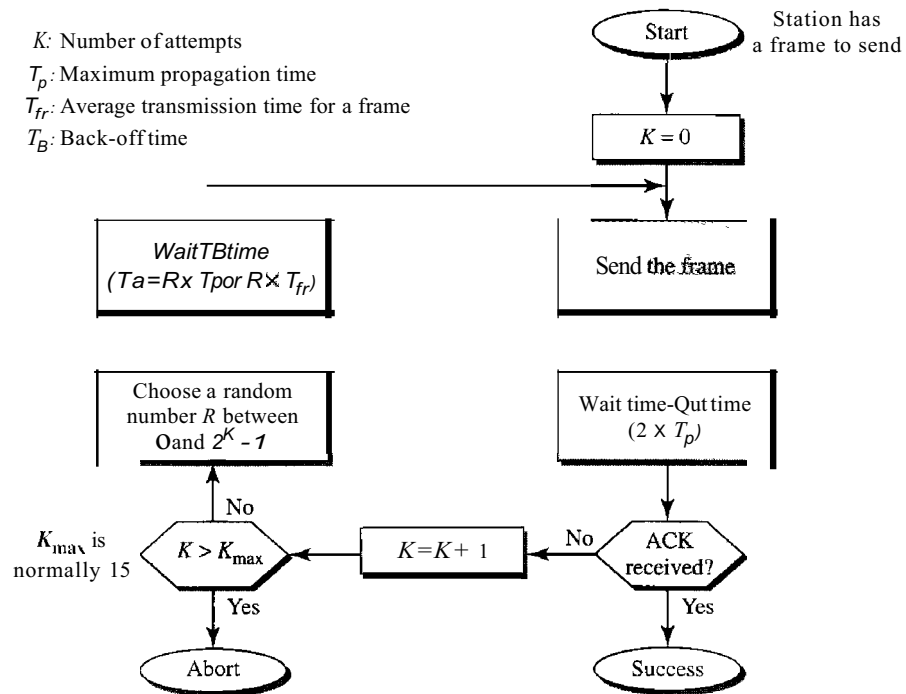
two frames survive: frame 1.1 from station 1 and frame 3.2 from station 3. We need to mention that even if one bit of a frame coexists on the channel with one bit from another frame, there is a collision and both will be destroyed.

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 back-off 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 12.4 shows the procedure for pure ALOHA based on the above strategy.

Figure 12.4 Procedure for pure ALOHA protocol



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 back-off 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 back-off**. In this

method, for each retransmission, a multiplier in the range 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.

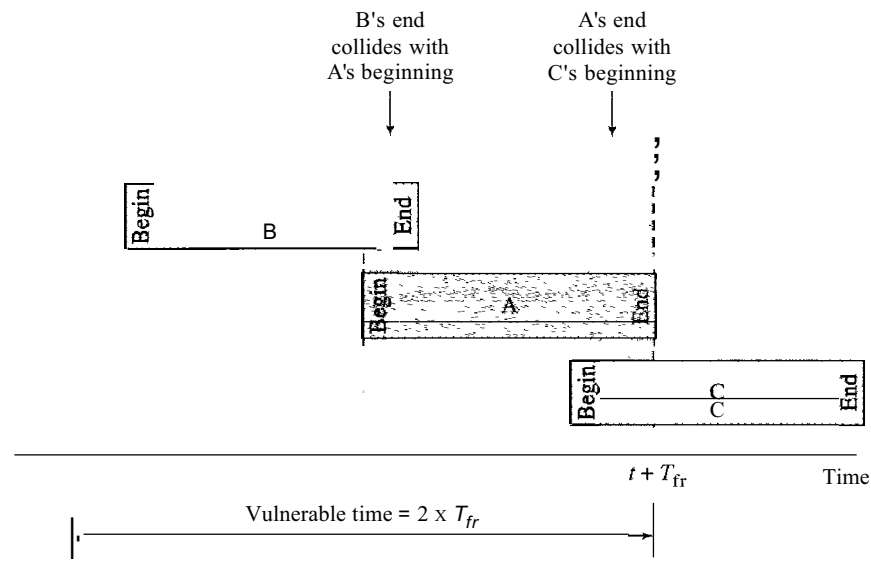
### Example 12.1

The stations on a wireless ALOHA network are a maximum of 600 km apart. If we assume that signals propagate at  $3 \times 10^8$  m/s, we find  $T_p = (600 \times 10^3) / (3 \times 10^8) = 2$  ms. Now we can find the value of  $T_B$  for different values of  $K$ .

- For  $K = 1$ , the range is  $\{0, 1\}$ . The station needs to generate a random number with a value of 0 or 1. This means that  $T_B$  is either 0 ms ( $0 \times 2$ ) or 2 ms ( $1 \times 2$ ), based on the outcome of the random variable.
- For  $K = 2$ , the range 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.
- For  $K = 3$ , the range is  $\{0, 1, 2, 3, 4, 5, 6, 7\}$ . This means that  $T_B$  can be 0, 2, 4, ..., 14 ms, based on the outcome of the random variable.
- We need to mention that if  $K > 10$ , it is normally set to 10.

**Vulnerable time** Let us find the length of time, the **vulnerable time**, in which there is a possibility of collision. We assume that the stations send fixed-length frames with each frame taking  $T_{fr}$  s to send. Figure 12.5 shows the vulnerable time for station A.

**Figure 12.5** Vulnerable time for pure ALOHA protocol



Station A sends a frame at time  $t$ . Now imagine station B has already sent a frame between  $t - T_{fr}$  and  $t$ . This leads to a collision between the frames from station A and station B. The end of B's frame collides with the beginning of A's frame. On the other hand, suppose that station C sends a frame between  $t$  and  $t + T_{fr}$ . Here, there is a collision between frames from station A and station C. The beginning of C's frame collides with the end of A's frame.

Looking at Figure 12.5, we see that the vulnerable time, during which a collision may occur in pure ALOHA, is 2 times the frame transmission time.

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

### 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  $2 \times 1 \text{ ms} = 2 \text{ ms}$ . This means no station should send later than 1 ms before this station starts transmission and no station should start sending during the one 1-ms period 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 proved that the average number of successful transmissions for pure ALOHA is  $S = G \times e^{-2G}$ . The maximum throughput  $S_{max}$  is 0.184, for  $G = \frac{1}{2}$ . In other words, if one-half a frame is generated during one frame transmission time (in other words, one frame during two frame transmission times), then 18.4 percent of these frames reach their destination successfully. This is an expected result 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.

---


$$\text{The throughput for pure ALOHA is } S = G \times e^{-2G}.$$

$$\text{The maximum throughput } S_{max} = 0.184 \text{ when } G = (1/2).$$


---

### 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.

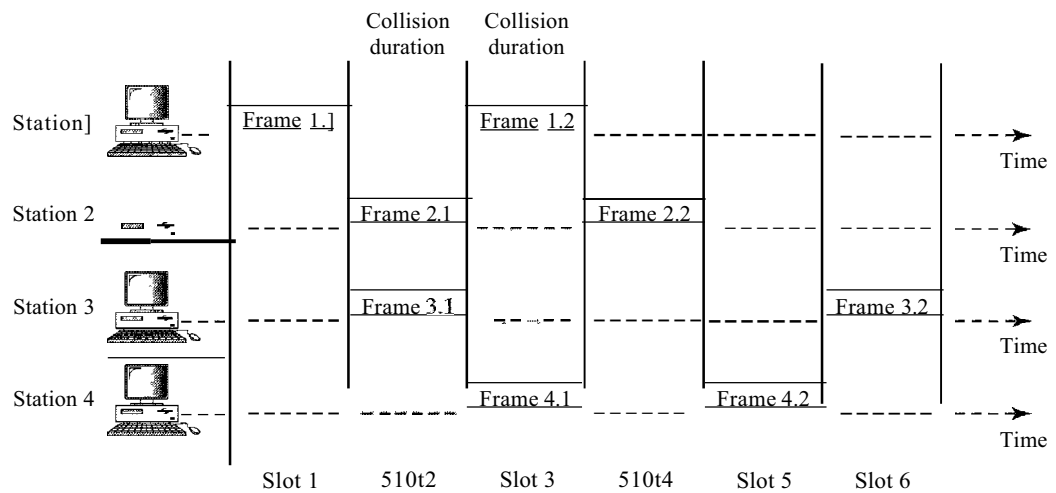
- a. If the system creates 1000 frames per second, this is 1 frame per millisecond. The load is 1. In this case  $S = G \times e^{-2G}$  or  $S = 0.135$  (13.5 percent). This means that the throughput is  $1000 \times 0.135 = 135$  frames. Only 135 frames out of 1000 will probably survive.
- b. If the system creates 500 frames per second, this is (1/2) frame per millisecond. The load is (1/2). In this case  $S = G \times e^{-2G}$  or  $S = 0.184$  (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, this is (1/4) frame per millisecond. The load is (1/4). In this case  $S = G \times e^{-2G}$  or  $S = 0.152$  (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 soon 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  $T_{fr}$  s and force the station to send only at the beginning of the time slot. Figure 12.6 shows an example of frame collisions in slotted ALOHA.

Figure 12.6 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 12.7 shows the situation.

Figure 12.7 shows that the vulnerable time for slotted ALOHA is one-half that of pure ALOHA.

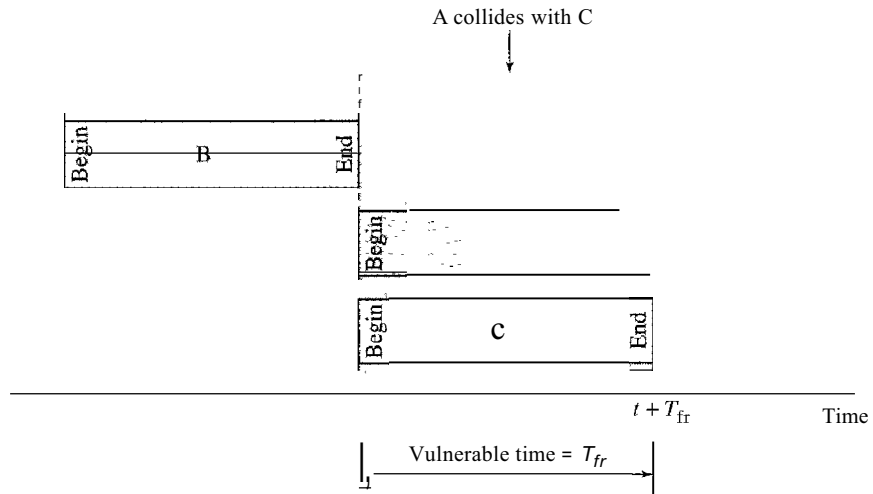
$$\text{Slotted ALOHA vulnerable time} = T_{fr}$$

**Throughput** It can be proved 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 a frame is generated during one frame transmission time, then 36.8 percent of these frames reach their destination successfully. This result can be expected 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  $G = 1$ .

Figure 12.7 Vulnerable time for slotted ALOHA protocol



*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  $S = G \times e^{-G}$  or  $S = 0.368$  (36.8 percent). This means that the throughput is  $1000 \times 0.368 = 368$  frames. Only 368 out of 1000 frames will probably survive. Note that this is the maximum throughput case, percentagewise.
- b. Here  $G$  is  $\frac{1}{2}$ . In this case  $S = G \times e^{-G}$  or  $S = 0.303$  (30.3 percent). This means that the throughput is  $500 \times 0.303 = 151$ . Only 151 frames out of 500 will probably survive.
- c. Now  $G$  is  $\frac{1}{4}$ . In this case  $S = G \times e^{-G}$  or  $S = 0.195$  (19.5 percent). This means that the throughput is  $250 \times 0.195 = 49$ . Only 49 frames out of 250 will probably survive.

**Carrier Sense Multiple Access (CSMA)**

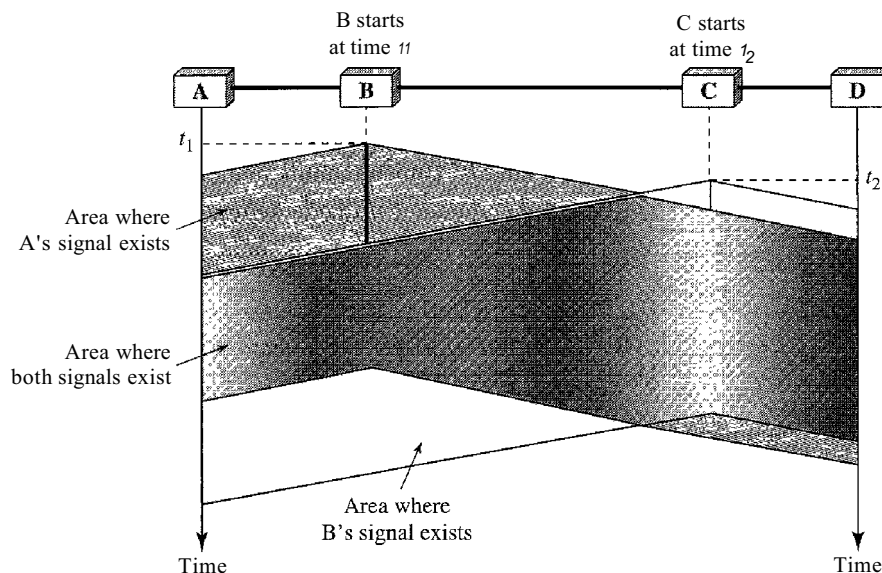
To minimize the chance of collision and, therefore, increase the performance, the CSMA method was developed. The chance of collision can be reduced if a station senses the medium before trying to use it. Carrier sense multiple access (CSMA) requires that each station first listen to the medium (or check 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 12.8, 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



Figure 12.8 Space/time model of the collision in CSMA



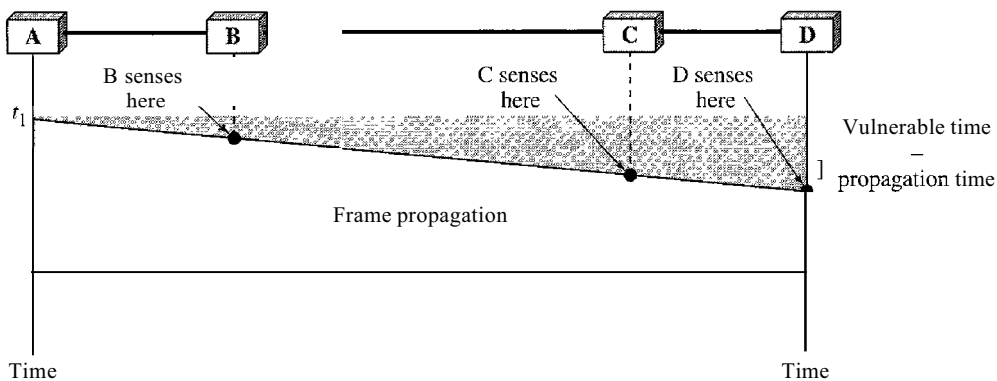
and for every station to sense it. In other words, a station may sense the medium and find it idle, only because the first bit sent by another station has not yet been received.

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 12.9 shows the worst

Figure 12.9 Vulnerable time in CSMA



case. The leftmost station A sends a frame at time  $t'$  which reaches the rightmost station D at time  $t' + T_p$ . The gray area shows the vulnerable area in time and space.

*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 I-persistent method, the nonpersistent method, and the p-persistent method. Figure 12.10 shows the behavior of three persistence methods when a station finds a channel busy.

**Figure 12.10** Behavior of three persistence methods

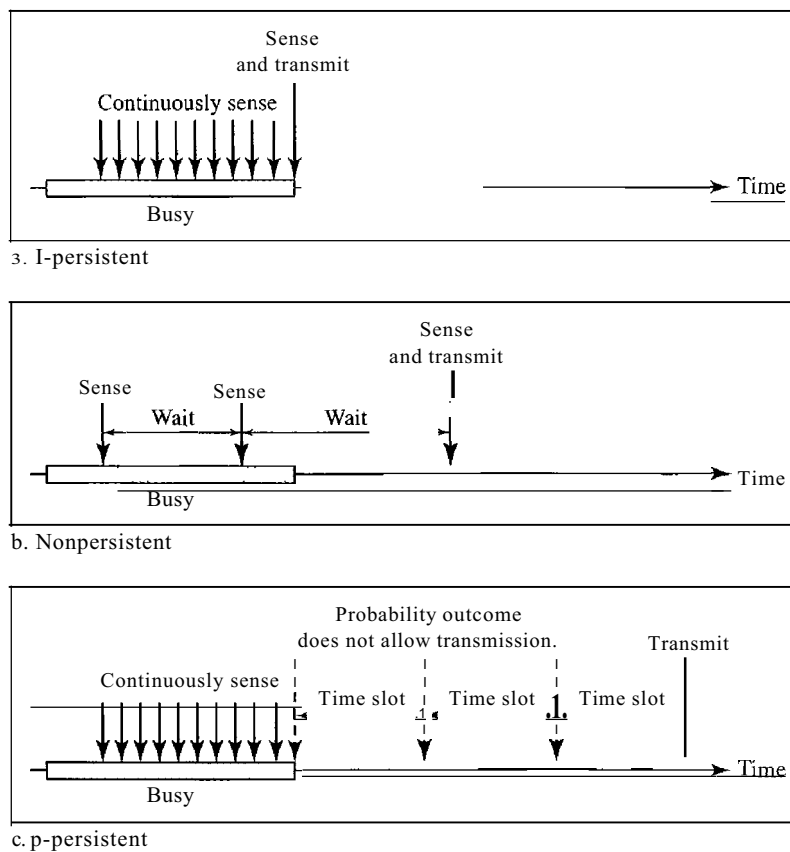
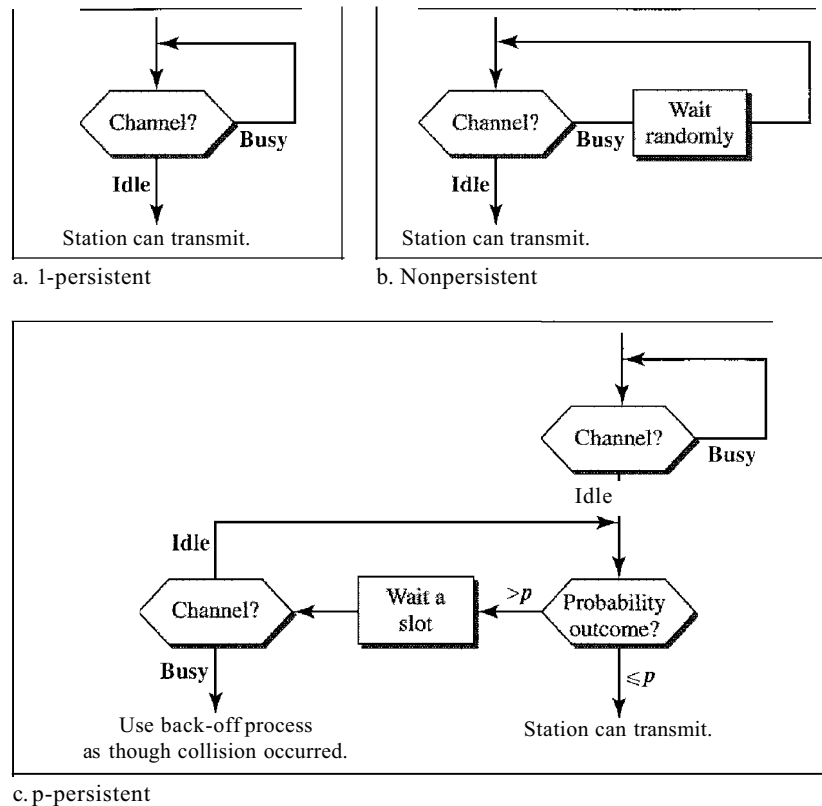


Figure 12.11 shows the flow diagrams for these methods.

**I-Persistent** The **I-persistent method** is simple and straightforward. In this method, after the station finds the line idle, it sends its frame immediately (with probability I). 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 in Chapter 13 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

**Figure 12.11** Flow diagram for three persistence methods

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.
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.

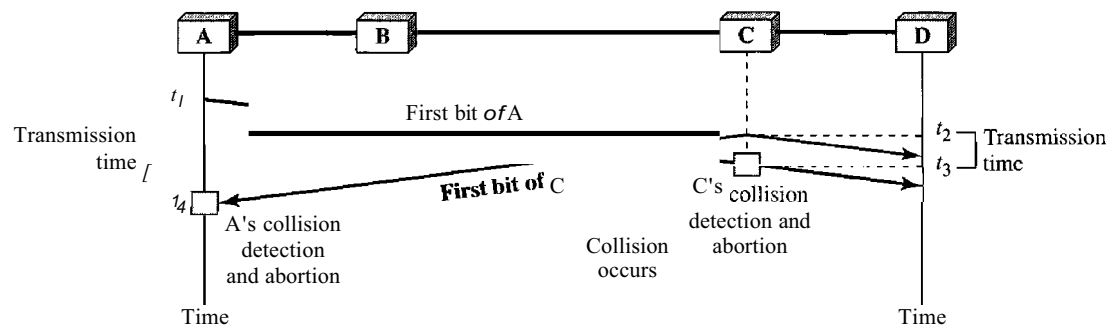
### Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

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 12.12, stations A and C are involved in the collision.

Figure 12.12 Collision of the first bit in CSMA/CD



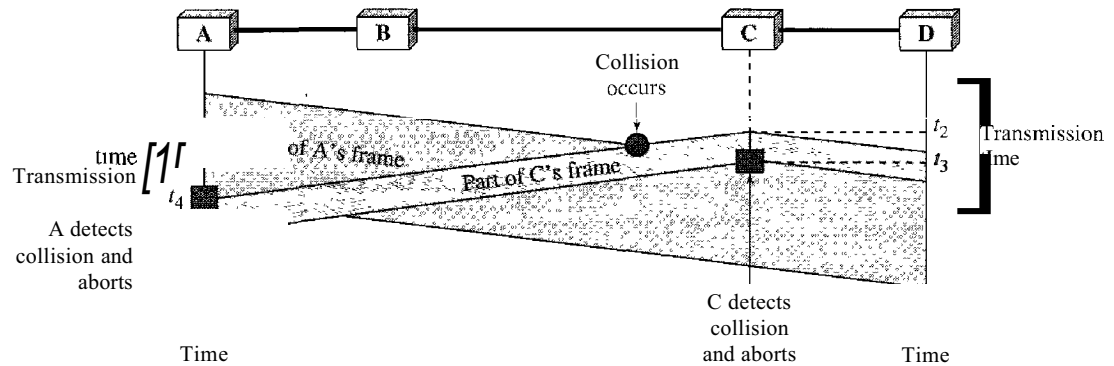
At time  $t_1$ , station A has executed its persistence procedure and starts sending the bits of its frame. At time  $t_2$ , station C has not yet sensed the first bit sent by A. Station C executes its persistence procedure and starts sending the bits in its frame, which propagate both to the left and 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  $t_3 - t_2$ . Later we show that, for the protocol to work, the length of any frame divided by the bit rate in this protocol must be more than either of these durations. At time  $t_4$ , the transmission of A's frame, though incomplete, is aborted; at time  $t_3$ , the transmission of B's frame, though incomplete, is aborted.

Now that we know the time durations for the two transmissions, we can show a more complete graph in Figure 12.13.

### 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  $2T_p$ .

Figure 12.13 Collision and abortion in CSMA/CD



### 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  $25.6 \mu\text{s}$ , what is the minimum size of the frame?

### Solution

The frame transmission time is  $T_{fr} = 2 \times T_p = 51.2 \mu\text{s}$ . This means, in the worst case, a station needs to transmit for a period of  $51.2 \mu\text{s}$  to detect the collision. The minimum size of the frame is  $10 \text{ Mbps} \times 51.2 \mu\text{s} = 512 \text{ bits}$  or 64 bytes. This is actually the minimum size of the frame for Standard Ethernet, as we will see in Chapter 13.

### Procedure

Now let us look at the flow diagram for *CSMA/CD* in Figure 12.14. It is similar to the one for the ALOHA protocol, but there are differences.

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 12.11.

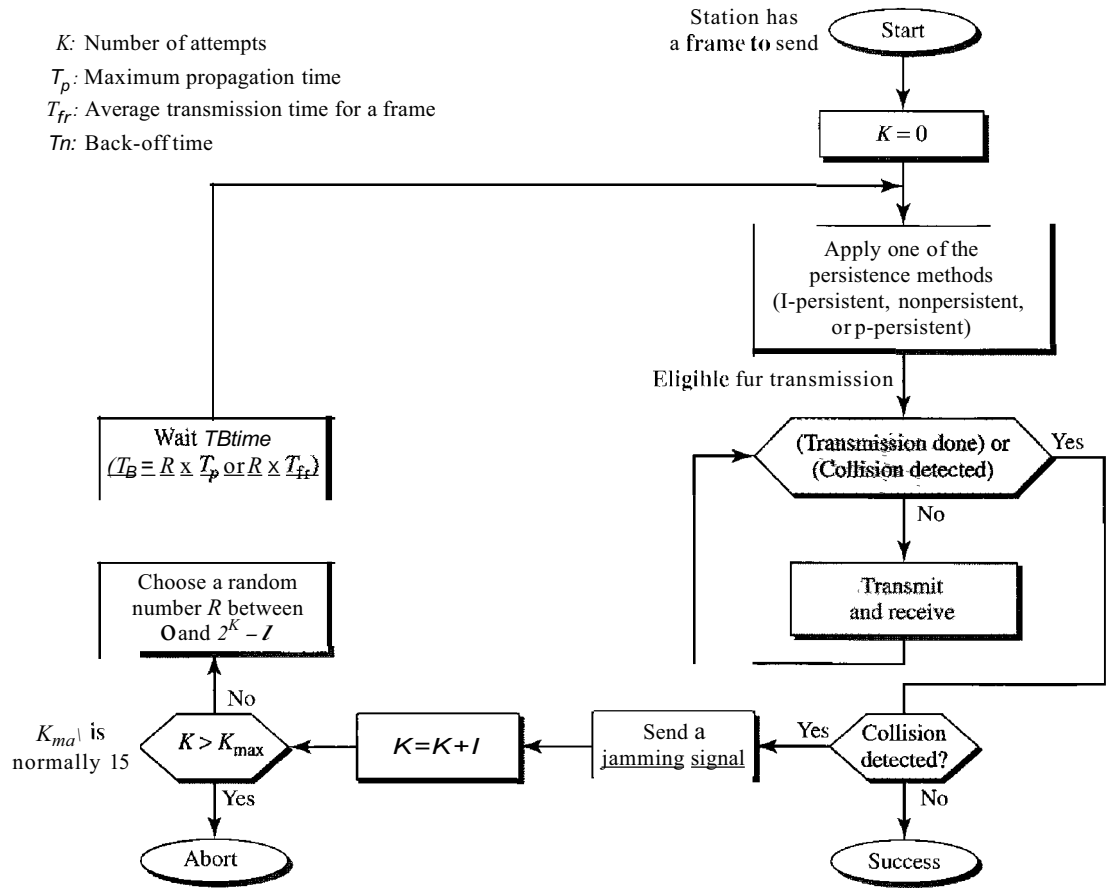
The second difference is the frame transmission. In ALOHA, we first transmit the entire frame and then wait for an acknowledgment. In *CSMA/CD*, transmission and collision detection is a continuous process. We do not send the entire frame and then look for a collision. The station transmits and receives continuously and simultaneously (using two different ports). 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 that enforces the collision in case other stations have not yet sensed the collision.

### Energy Level

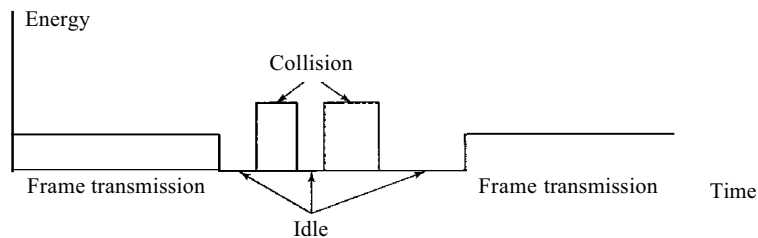
We can say that the level of energy in a channel can have three values: zero, normal, and abnormal. At the zero level, the channel is idle. At the normal level, a station has

**Figure 12.14** Flow diagram for the CSMA/CD



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 12.15 shows the situation.

**Figure 12.15** 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 I-persistent method the maximum throughput is around 50 percent when  $G = 1$ . For nonpersistent method, the maximum throughput can go up to 90 percent when  $G$  is between 3 and 8.

## Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)

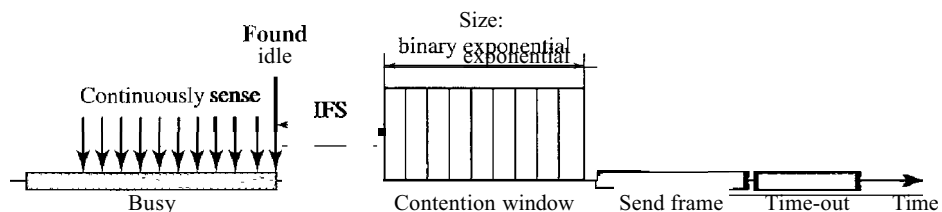
The basic idea behind *CSMA/CD* is that a station needs to be able to receive while transmitting to detect a collision. When there is no collision, the station receives one signal: its own signal. When there is a collision, the station receives two signals: its own signal and the signal transmitted by a second station. To distinguish between these two cases, the received signals in these two cases must be significantly different. In other words, the signal from the second station needs to add a significant amount of energy to the one created by the first station.

In a wired network, the received signal has almost the same energy as the sent signal because either the length of the cable is short or there are repeaters that amplify the energy between the sender and the receiver. This means that in a collision, the detected energy almost doubles.

However, in a wireless network, much of the sent energy is lost in transmission. The received signal has very little energy. Therefore, a collision may add only 5 to 10 percent additional energy. This is not useful for effective collision detection.

We need to avoid collisions on wireless networks because they cannot be detected. Carrier sense multiple access with collision avoidance (*CSMA/CA*) was invented for this network. Collisions are avoided through the use of *CSMA/CA*'s three strategies: the inter-frame space, the contention window, and acknowledgments, as shown in Figure 12.16.

Figure 12.16 Timing in *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. If after the IFS time the channel is still idle, the station can send, but it still needs to wait a time equal to the contention time (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.

---

In CSMA/CA, the IFS can also be used to define  
the priority of a station or a frame.

---

### *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 back-off 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.

---

In CSMA/CA, if the station finds the channel busy,  
it does not restart the timer of the contention window;  
it stops the timer and restarts it when the channel becomes idle.

---

### *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.

### *Procedure*

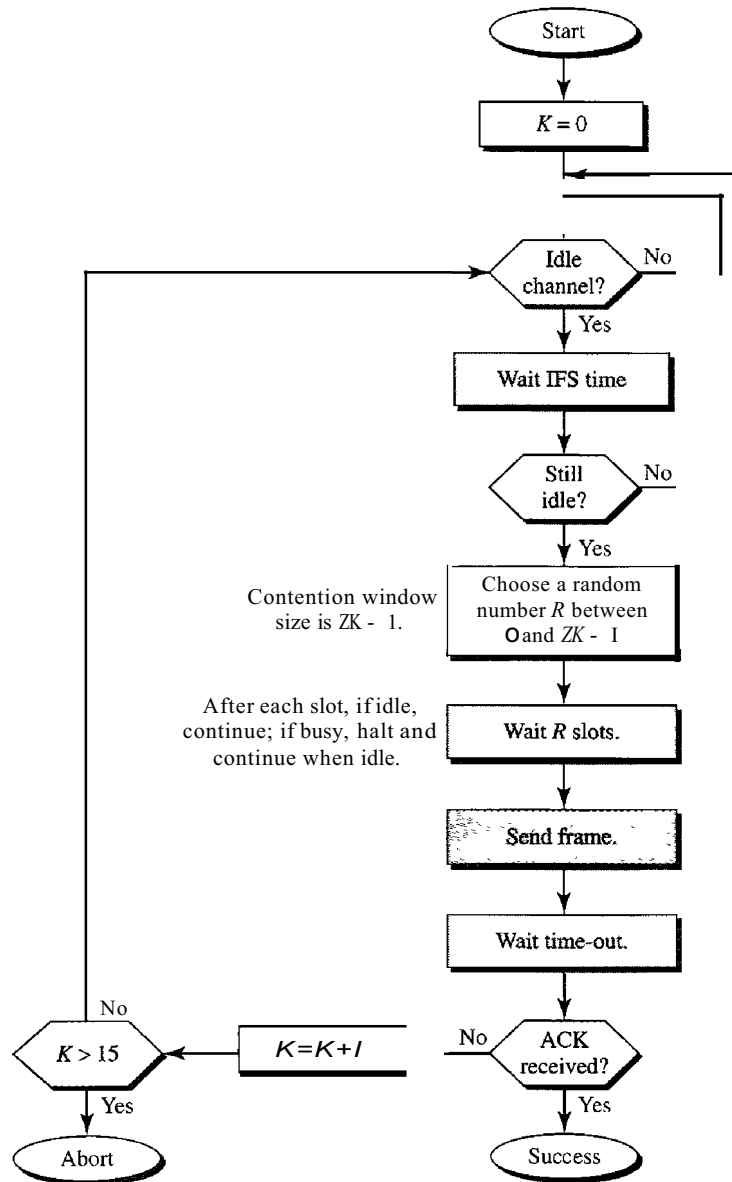
Figure 12.17 shows the procedure. Note that the channel needs to be sensed before and after the IFS. The channel also needs to be sensed during the contention time. For each time slot of the contention window, the channel is sensed. If it is found idle, the timer continues; if the channel is found busy, the timer is stopped and continues after the timer becomes idle again.

### *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 hand-shaking features. The use of CSMA/CA in wireless networks will be discussed in Chapter 14.



Figure 12.17 Flow diagram for CSMA/CA



## 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 popular controlled-access methods.

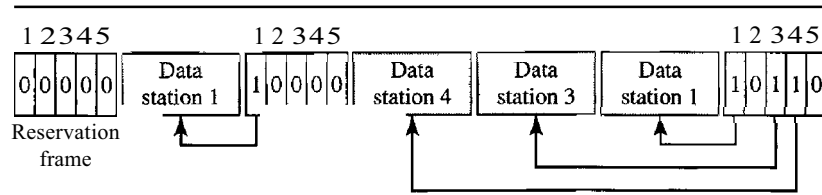
### Reservation

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.

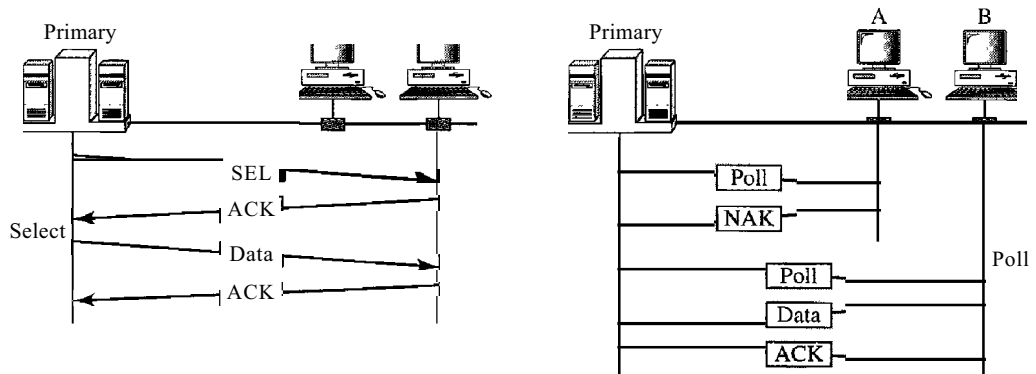
Figure 12.18 Reservation access method



### 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).

Figure 12.19 Select and poll functions in polling access method



If the primary wants to receive data, it asks the secondaries if they have anything to send; this is called poll function. If the primary wants to send data, it tells the secondary to get ready to receive; this is called select function.

*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.

## 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.

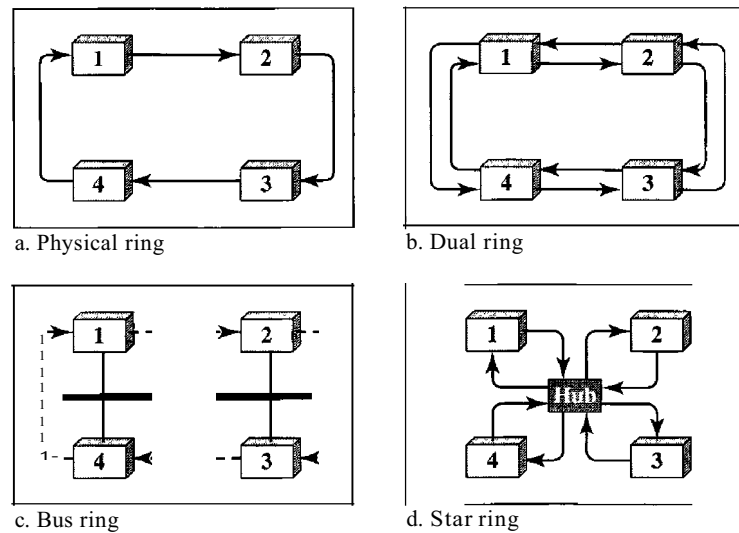
But how is the right to access the channel passed from one station to another? In this method, a special packet called a token circulates through the ring. The possession of the token gives the station the right to access the channel and send its data. When a station has some data to send, it waits until 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 show 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 is a multiple-access method in which the available bandwidth of a link is shared in time, frequency, or through code, between different stations. In this section, we discuss three channelization protocols: FDMA, TDMA, and CDMA.

---

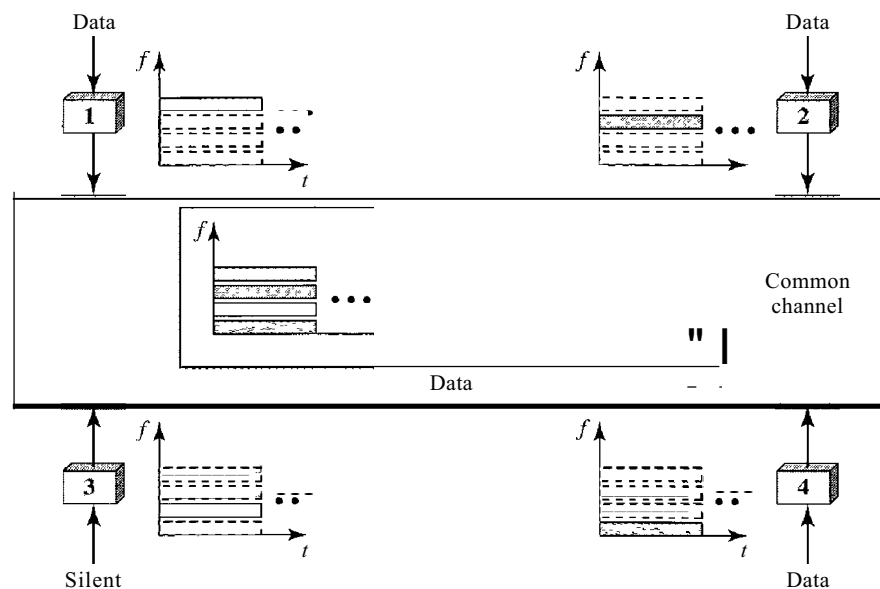
We see the application of all these methods in Chapter 16  
when we discuss cellular phone systems.

---

### Frequency-Division Multiple Access (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.

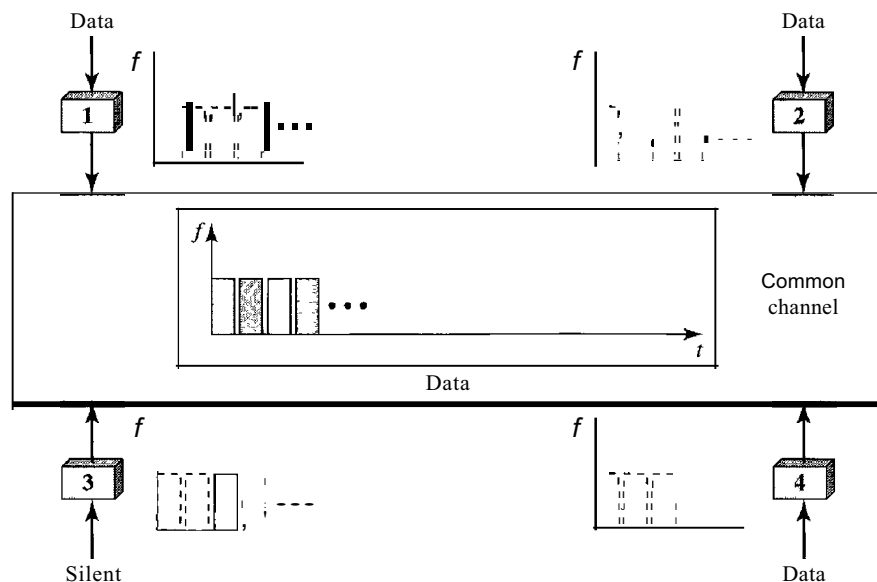
We need to emphasize that although FDMA and FDM conceptually seem similar, there are differences between them. FDM, as we saw in Chapter 6, is a physical layer technique that combines the loads from low-bandwidth channels and transmits them by using a high-bandwidth channel. The channels that are combined 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.

### Time-Division Multiple Access (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 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.

### Code-Division Multiple Access (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 because only one channel occupies the entire bandwidth of the link. It differs from TDMA because 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 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*

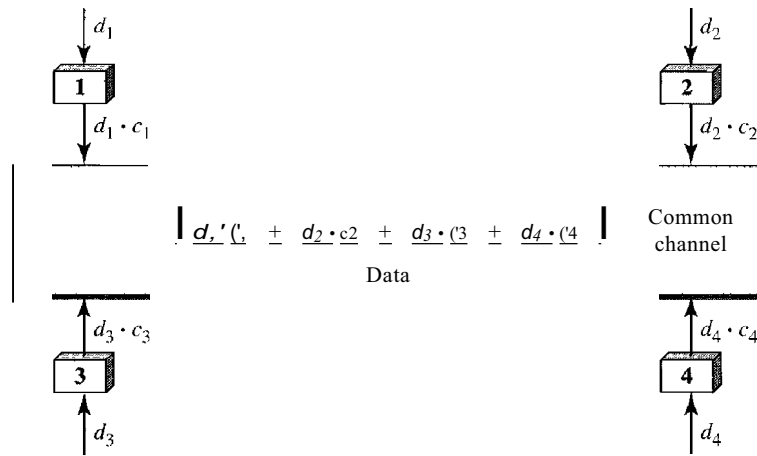
Let us assume we have four stations 1, 2, 3, and 4 connected to the same channel. The data from station 1 are  $d_1$ , from station 2 are  $d_2$ , and so on. The code assigned to the first station is  $c_1$ , to the second is  $c_2$ , and so on. We assume that the assigned codes have two properties.

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

Figure 12.23 Simple idea of communication with code



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.

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

**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.

Figure 12.24 Chip sequences



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:

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,

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

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 the each sequence. This is called the inner product of two equal sequences. For example,

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

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

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

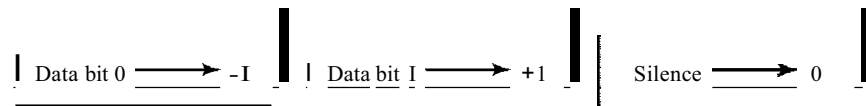
5. Adding two sequences means adding the corresponding elements. The result is another sequence. For example,

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

### 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



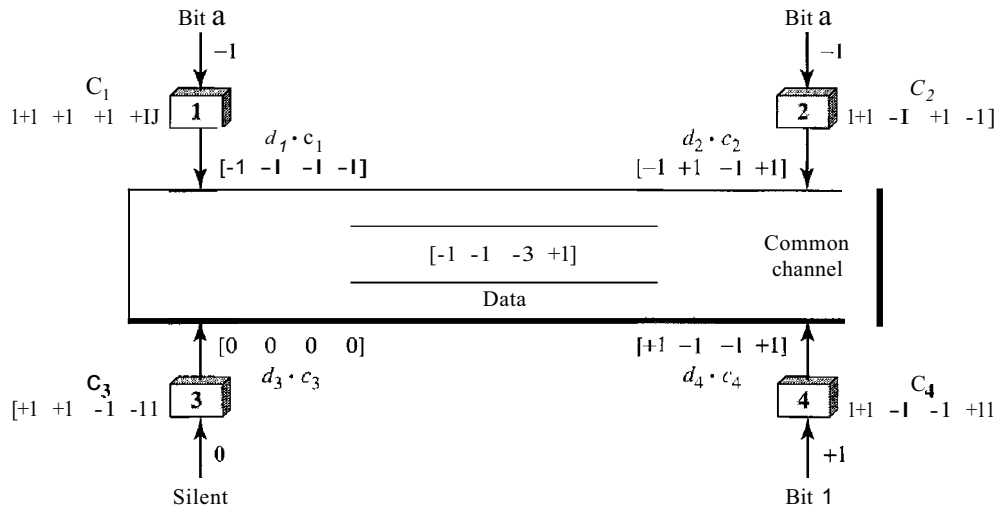
### Encoding and Decoding

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 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

$$[-1 -1 -3 +1] \cdot [+1 -1 +1 -1] = -4/4 = -1 \rightarrow \text{bit 1}$$

Figure 12.26 Sharing channel in CDMA



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 12.27 Digital signal created by four stations in CDMA

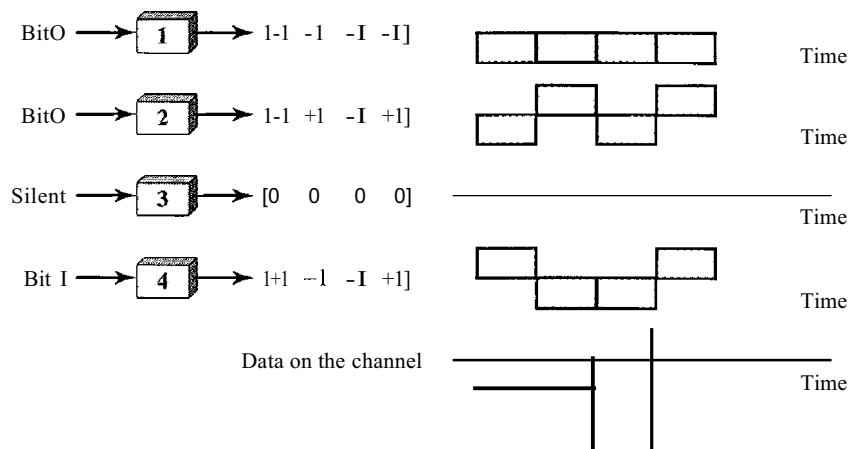
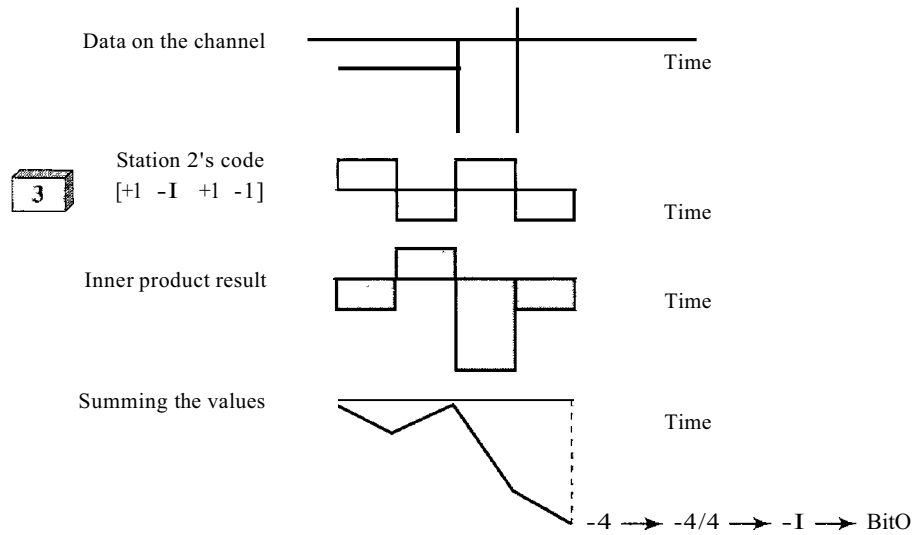


Figure 12.28 shows how station 3 can detect the data sent by station 2 by using the code for station 2. The total data on the channel are multiplied (inner product operation) by the signal representing station 2 chip code to get a new signal. The station then integrates and adds the area under the signal,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

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



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

$$W_1 = \begin{bmatrix} +1 \end{bmatrix} \qquad W_{2N} = \begin{bmatrix} W_N & W_N \\ W_N & \overline{W_N} \end{bmatrix}$$

a. Two basic rules

$$W_1 = \begin{bmatrix} +1 \end{bmatrix} \qquad W_2 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix} \qquad W_4 = \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{bmatrix}$$

b. Generation of  $W_1$ ,  $W_2$ , and  $W_4$

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  $W_j$ 's, with the last one the complement of  $W_1$ . After  $W_2$  is generated,  $W_4$  can be made of four  $W_2$ 's, with the last one the complement of  $W_2$ . Of course,  $W_8$  is composed of four  $W_4$ 's, and so on. Note that after  $W_N$  is made, each station is assigned a chip corresponding to a row.

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  $D = d_1 \cdot (1 + d_2 \cdot (2 + d_3 \cdot (3 + d_4 \cdot (4)))$ . The receiver which wants to get the data sent by station 1 multiplies these data by  $c_1$ :

$$\begin{aligned} D \cdot c_1 &= (d_1 \cdot (1 + d_2 \cdot (2 + d_3 \cdot (3 + d_4 \cdot (4))) \cdot c_1 \\ &= d_1 \cdot c_1 \cdot (1 + d_2 \cdot (2 \cdot (1 + d_3 \cdot (3 \cdot (1 + d_4 \cdot (4 \cdot c_1) \\ &= d_1 \times N + d_2 \times 0 + d_3 \times 0 + d_4 \times 0 \\ &= d_1 \times N \end{aligned}$$

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

---

## 12.4 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.