Chapter 3

Multiple Access Techniques I

3.1 Definition

The data link layer is divided into two sub-layers: Logical link control (LLC) and Medium access control (MAC) sub layers. The LLC is responsible for flow and error control, and the lower sublayer, MAC sublayer, is responsible for resolving access to the shared media. If the channel is dedicated, we do not need the MAC sublayer. When 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. 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 Fig. 3.1.

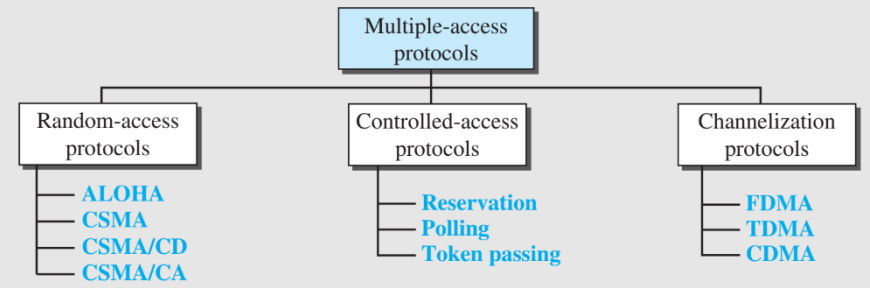

Figure 3.1 Taxonomy of multiple access protocols

There are some advantages of using shared medium applying MAC protocols. These include:

- low cost infrastructure since multiple devices use the same medium
- all stations attached to the medium hear
- transmission from any other station ⇒ routing not necessary

However, such techniques pose some challenges to be dealt with for ensuring effective sharing of the transmission medium:

- access of multiple sending and receiving nodes to the shared medium must be coordinated
- stations should not be transmitting simultaneously or interrupting each other
- stations should not be able to 'monopolize' the transmission/shared medium

Next, each categories of MAC protocols will be discussed in details.

3.2 Channelization Techniques

Channelization (also known as channel partition) 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.

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*. Depending on applications, this guard band can consumes up to 40% of the available bandwidth which ultimately reduces throughput. 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. Fig. 3.2 shows the idea of FDMA.

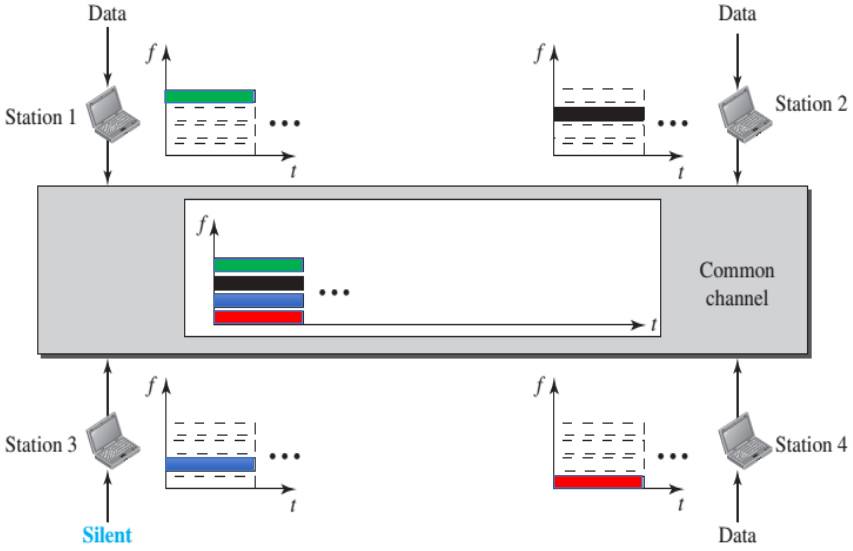

Figure 3.2 Illustration of FDMA

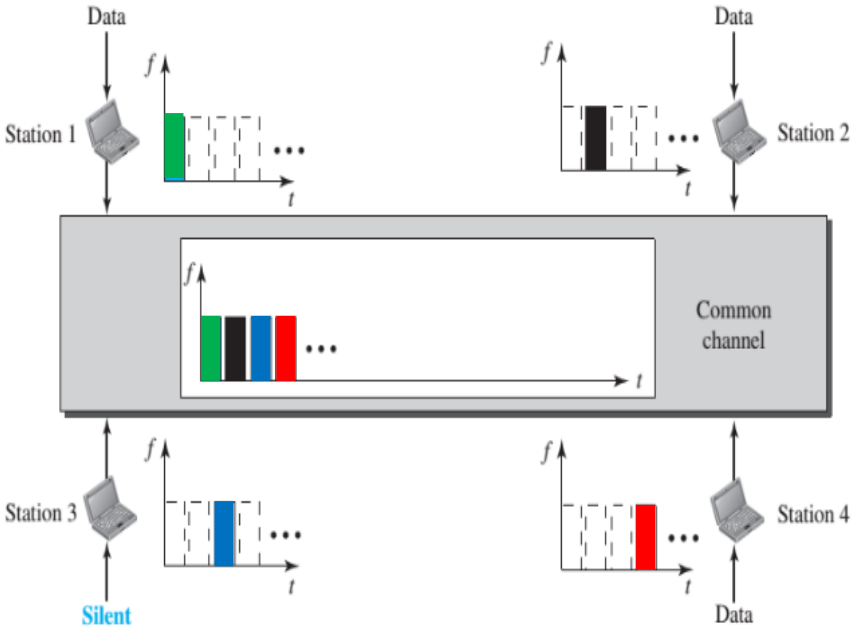

Figure 3.3 Illustration of TDMA

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. 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. The concept of TDMA is illustrated in Fig. 3.3.

CDMA

In code division multiple access (CDMA), all stations can send their data using the whole bandwidth all the time. 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. Instead of time or bandwidth, each user is allocated a unique code. The codes must be orthogonal in that if we multiply any two codes, the result will be zero. Suppose that we have four users to send their data using CDMA. Then, CDMA procedure can be summarized as follows:

1. Each user is given a unique code.
2. Each user multiplies the bit he wants to send by his code.
3. The result of multiplications of all users are added together and transmitted.

4. At the receiving end, if any other user wants to decode (recovering the original bit) the bit sent from any of the send, he multiplies the received sequence by the unique code of the sender.
5. Then, he divides the result of the multiplication done in step 4 by the length of the code, where the length of a code is the number of elements in the code. The result of the division is the bit sent from the intended sender.

Let's see an example to properly comprehend the CDMA technique.
First of all we have to generate the orthogonal code. We are going to generate Walsh-Hadamard code. Let's start with a one-element matrix (!), H.

$$H = [1]$$

Using this matrix, next create a four elements (2×2) matrix, H^2, where each element of the matrix is taken from the already created one element matrix, H.

$$H^2 = \begin{bmatrix} H^1 & H^1 \\ H^1 & -H^1 \end{bmatrix}$$

$$H^2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Then, create a 4×4 matrix, using H^2.

$$H^4 = \begin{bmatrix} H^2 & H^2 \\ H^2 & -H^2 \end{bmatrix}$$

$$H^4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Similarly, we can create H^8 using H^4.

$$H^8 = \begin{bmatrix} H^4 & H^4 \\ H^4 & -H^4 \end{bmatrix}$$

$$H^8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

Each row of each of the matrix represents a unique code. If we multiply any two codes (rows) of a matrix using vector multiplications, the results will be zero. For example, if we multiply first two rows of H^4, we get 1×1+1×-1+1×1+1×-1=0. If we have two users, we can allocate code to each user from H^2: one row for each user. If we have three or four users, we can allocate codes from H^4 and so on.

Suppose there are five users A, B, C, D and E. They want to send 0, +1, +1, -1, -1, respectively. Here, 0 indicates A is not going to send any data. Whereas +1 indicates binary 1, -1 indicates binary 0. To find the transmit sequence, first allocate code to each user. Since we have five users, we have to allocate code from H^8. Suppose that we allocate first five rows to the users sequentially; i.e., First row (R1) to A, second row (R2) to B. Similarly, R3, R4 and R5 to C, D and E, respectively.

Next, multiply the code of each user by the data he wants to send. Then, add all the results.

$$
\begin{aligned}
0 \times R1 = \quad & [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0] \\
1 \times R2 = \quad & [1 \quad -1 \quad 1 \quad -1 \quad 1 \quad -1 \quad 1 \quad -1] \\
1 \times R3 = \quad & [1 \quad 1 \quad -1 \quad -1 \quad 1 \quad 1 \quad -1 \quad -1] \\
-1 \times R4 = \quad & [-1 \quad 1 \quad 1 \quad -1 \quad -1 \quad 1 \quad 1 \quad -1] \\
-1 \times R5 = \quad & [-1 \quad -1 \quad -1 \quad -1 \quad 1 \quad 1 \quad 1 \quad 1] \\
\hline
& [0 \quad 0 \quad 0 \quad -4 \quad 2 \quad 2 \quad 2 \quad -2]
\end{aligned}
$$

The result of the multiplication and addition is $[0 \quad 0 \quad 0 \quad -4 \quad 2 \quad 2 \quad 2 \quad -2]$. This vector will be transmitted. This vector contains the transmitted information from each user. However, it is impossible to tell which user is sending which value just observing the transmit sequence. The question is how a recipient can recover the bit sent from a particular user.

Suppose that a user named G is talking to user B. So G needs to recover the value sent from user B. Like all other recipients, G also receives the same sequence,
$[0 \quad 0 \quad 0 \quad -4 \quad 2 \quad 2 \quad 2 \quad -2]$. To recover the value sent from B, G will multiply the received sequence by the code of B (Please notice that G is going to use the code of send B, NOT his own code).

$$Received\ sequence \times Desired\ user's\ (B's)\ code$$

$$[0 \quad 0 \quad 0 \quad -4 \quad 2 \quad 2 \quad 2 \quad -2] \times [1 \quad -1 \quad 1 \quad -1 \quad 1 \quad -1 \quad 1 \quad -1]$$

$$=[0 \quad 0 \quad 0 \quad -4 \quad 2 \quad 2 \quad 2 \quad -2] \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$$

$$= 0 \times 1 + 0 \times -1 + 0 \times 1 - 4 \times -1 + 2 \times 1 + 2 \times -1 + 2 \times 1 - 2 \times -1$$

$$= 0 + 0 + 0 + 4 + 2 - 2 + 2 + 2$$

$$= 8$$

The result will be multiplied by the length of the code which is 8 as the code consists of eight elements.

The bit sent from user B $= Received\ sequence \times B's\ code\ /(length\ of\ the\ code)$

$=8/8= 1$

Thus, the G recovers the value sent from B correctly. Similarly, any recipient can recover the data sent from any sender as long as he has the code of the sender whose data he wants to recover.

3.3 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.

Reservation Technique

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. Fig. 3.4 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.

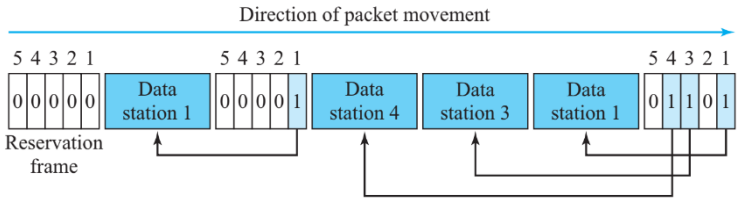

Figure 3.4 Reserveration technique

Polling Technique

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 3.5). This method uses poll and select functions to prevent collisions. However, the drawback is if the primary station fails, the system goes down.

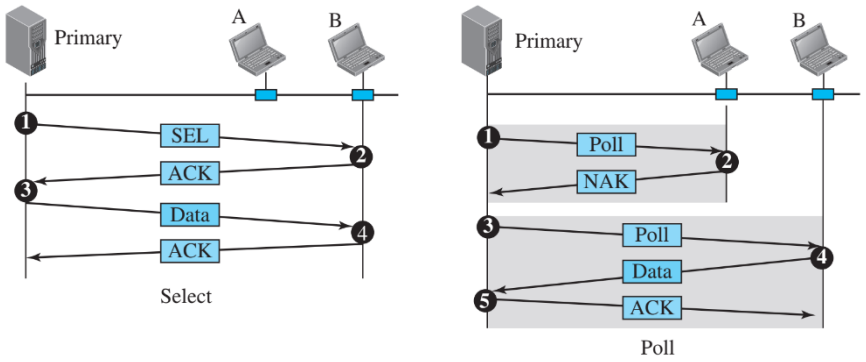

Figure 3.5 Select and poll functions in polling-access method

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.

Token Passing Technique

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.

In a token-passing network, stations do not have to be physically connected in a ring; the ring can be a logical one. Figure 3.6 shows four different physical topologies that can create a logical ring.

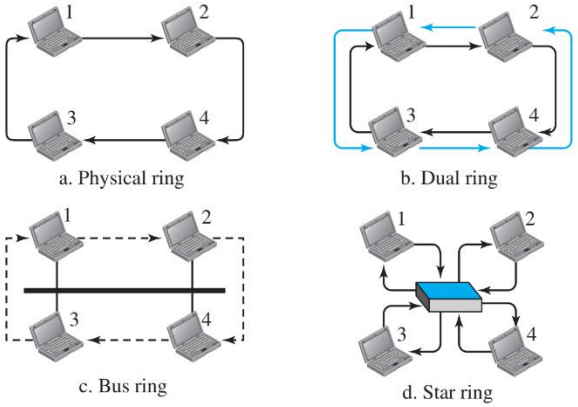

Figure 3.6 Logical rings and physical topology in a token-passing access method

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.