Chapter Three

Data Link Control

3.1 Introduction

Our discussion so far has concerned *sending signals over a transmission link. For effective* digital data communications, much more is needed to control and manage the exchange. To achieve the necessary control, a layer of logic is added above the physical interfacing discussed in pervious chapter; this logic is referred to as *data link layer* or a *data link control protocol.* When a data link control protocol is used, the transmission medium between systems is referred to as a *data link.* In this chapter we will study the design principles for layer 2, the data link layer. This study deals with the algorithms for achieving reliable, efficient communication between two adjacent machines at the data link layer. By adjacent, we mean that the two machines are connected by a communication channel that acts conceptually like a wire (e.g., a coaxial cable, telephone line, or point-to-point wireless channel). The essential property of a channel that makes it "wirelike" is that the bits are delivered in exactly the same order in which they are sent.

To see the need for data link control, some list of the requirements and objectives for effective data communication between two directly connected transmitting- receiving stations are:

Frame synchronization: Data are sent in blocks called frames. The beginning and end of each frame must be recognizable.

Flow control: The sending station must not send frames at a rate faster then the receiving station can absorb them.

Error control: Any bit errors introduced by the transmission system must be corrected.

Addressing: On a multipoint line, such as a local area network (LAN), the identity of the two stations involved in a transmission must be specified.

Control and data on same link: It is usually not desirable to have a physically separate communications path for control information. Accordingly, the receiver must be able to distinguish control information from the data being transmitted.

Link management: The initiation, maintenance, and termination of a sustained data exchange require a fair amount of coordination and cooperation among stations. Procedures for the management of this exchange are required.

To accomplish these goals, the data link layer takes the packets it gets from the network layer and encapsulates them into frames for transmission. Each frame contains a frame header, a payload field for holding the packet, and a frame trailer, as illustrated in figure 4.1. Frame management forms the heart of what the data link layer does. In the following sections we will examine all the above-mentioned issues in some detail.

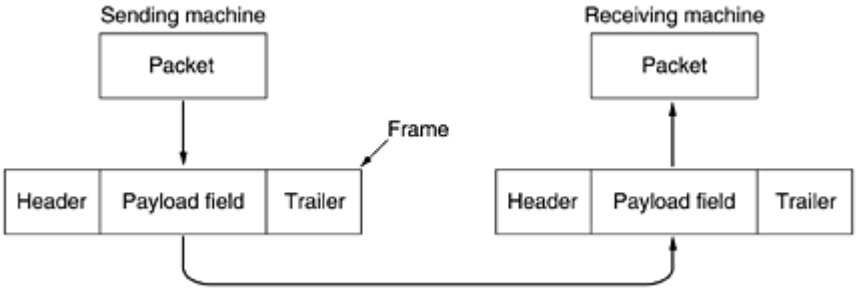

Figure 3.1 structure of a frame

3.2 Services Provided to the Network Layer

The function of the data link layer is to provide services to the network layer. The principal service is transferring data from the network layer on the source machine to the network layer on the destination machine. On the source machine is an entity, call it a process, in the network layer that hands some bits to the data link layer for transmission to the destination. The job of the data link layer is to transmit the bits to the destination machine so they can be handed over to the network layer there. The data link layer can be designed to offer various services. The actual services offered can vary from system to system. Three reasonable possibilities that are commonly provided are

- Unacknowledged connectionless service.
- Acknowledged connectionless service.
- Acknowledged connection-oriented service.

Unacknowledged connectionless service consists of having the source machine send independent frames to the destination machine without having the destination machine acknowledge them. No logical connection is established beforehand or released afterward. If a frame is lost due to noise on the line, no attempt is made to detect the loss or recover from it in the data link layer. This class of service is appropriate when the error rate is very low so that recovery is left to higher layers. It is also appropriate for real-time traffic, such as voice, in

which late data are worse than bad data. Most LANs use unacknowledged connectionless service in the data link layer.

The next step up in terms of reliability is ***acknowledged connectionless service.*** When this service is offered, there are still no logical connections used, but each frame sent is individually acknowledged. In this way, the sender knows whether a frame has arrived correctly. If it has not arrived within a specified time interval, it can be sent again. This service is useful over unreliable channels, such as wireless systems.

The most sophisticated service the data link layer can provide to the network layer is ***connection-oriented service***. With this service, the source and destination machines establish a connection before any data are transferred. Each frame sent over the connection is numbered, and the data link layer guarantees that each frame sent is indeed received. Furthermore, it guarantees that each frame is received exactly once and that all frames are received in the right order. With connectionless service, in contrast, it is conceivable that a lost acknowledgement causes a packet to be sent several times and thus received several times. Connection-oriented service, in contrast, provides the network layer processes with the equivalent of a reliable bit stream.

When connection-oriented service is used, transfers go through three distinct phases. In the first phase, the connection is established by having both sides initialize variables and counters needed to keep track of which frames have been received and which ones have not. In the second phase, one or more frames are actually transmitted. In the third and final phase, the connection is released, freeing up the variables, buffers, and other resources used to maintain the connection.

Consider a typical example: a WAN subnet consisting of routers connected by point-to-point leased telephone lines. When a frame arrives at a router, the hardware checks it for errors (using techniques we will study late in this chapter), then passes the frame to the data link layer software (which might be embedded in a chip on the network interface card). The data link layer software checks to see if this is the frame expected, and if so, gives the packet contained in the payload field to the routing software. The routing software then chooses the appropriate outgoing line and passes the packet back down to the data link layer software, which then transmits it.

3.3 Fragmentation

To provide service to the network layer, the data link layer must use the service provided to it by the physical layer. What the physical layer does is accept a raw bit stream and attempt to deliver it to the destination. This bit stream is not guaranteed to be error free. The number of bits received may be less than, equal to, or more than the number of bits transmitted, and they may have different values. It is up to the data link layer to detect and, if necessary, correct errors. The usual approach is for the data link layer to break the bit stream up into discrete frames and compute the checksum for each frame. (Checksum algorithms will be discussed later in this chapter.) When a frame arrives at the destination, the checksum is recomputed. If the newly-computed checksum is different from the one contained in the frame, the data link layer knows that an error has occurred and takes steps to deal with it (e.g., discarding the bad frame and possibly also sending back an error report). Why we have to break up a large block of data into smaller blocks and transmit the data in many frames? This is done for the following reasons:

- The buffer size of the receiver may be limited.
- The longer the transmission, the more likely that there will be an error, necessitating retransmission of the entire frame. With smaller frames, errors are detected sooner, and a smaller amount of data needs to be retransmitted.
- On a shared medium, such as a LAN, it is usually desirable not to permit one station to occupy the medium for an extended period, as this causes long delays at the other sending stations.

Breaking the bit stream up into frames is more difficult than it at first appears. One way to achieve this framing is to insert time gaps between frames, much like the spaces between words in ordinary text. However, networks rarely make any guarantees about timing, so it is possible these gaps might be squeezed out or other gaps might be inserted during transmission.

Since it is too risky to count on timing to mark the start and end of each frame, other methods have been devised. One of such methods is **bit stuffing.** This technique allows data frames to contain an arbitrary number of bits and allows character codes with an arbitrary number of bits per character. It works like this. Each frame begins and ends with a special bit pattern, 01111110. Whenever the sender's data link layer encounters five consecutive 1s in the data, it automatically stuffs a 0 bit into the outgoing bit stream. When the receiver sees five consecutive incoming 1 bits, followed by a 0 bit, it automatically destuffs (i.e., deletes) the 0 bit. If the user

data contain the flag pattern, 01111110, this flag is transmitted as 011111010 but stored in the receiver's memory as 01111110. **The other methods (Character count and Flag Bytes with byte stuffing) is left to you as reading assignment**.

3.4 Error Detection

In pervious chapter, we talked about transmission impairments and the effect of data rate. Regardless of the design of the transmission system, there will be errors, resulting in the change of one or more bits in a transmitted frame.

In digital transmission systems, an error occurs when a bit is altered between transmission and reception; that is, a binary 1 is transmitted and a binary 0 is received, or a binary 0 is transmitted and a binary 1 is received. Two general types of errors can occur: ***single-bit errors and burst errors***. A single-bit error is an isolated error condition that alters one bit but does not affect nearby bits. A burst error of length B is a contiguous sequence of B bits in which the first and last bits and any number of intermediate bits are received in error.

As a result of the physical processes that generate them, errors on some media (e.g., radio) tend to come in bursts rather than singly. Having the errors come in bursts has both advantages and disadvantages over isolated single-bit errors. On the advantage side, computer data are always sent in blocks of bits. Suppose that the block size is 1000 bits and the error rate is 0.001 per bit. If errors were independent, most blocks would contain an error. If the errors came in bursts of 100 however, only one or two blocks in 100 would be affected, on average. The disadvantage of burst errors is that they are much harder to correct than are isolated errors.

Network designers have developed two basic strategies for dealing with errors. One way is to include enough redundant information along with each block of data sent, to enable the receiver to deduce what the transmitted data must have been. The other way is to include only enough redundancy to allow the receiver to deduce that an error occurred, but not which error, and have it request a retransmission. The former strategy uses error-correcting codes and the latter uses error-detecting codes.

3.4.1 Error Detection

For a given frame of bits, additional bits that constitute an error-detecting code are added by the transmitter. This code is calculated as a function of the other transmitted bits. The receiver performs the same calculation and compares the two results. A detected error occurs if and only if there is a mismatch. The following paragraphs discuss two of such methods.

Parity Check

The simplest error-detection scheme is to append a parity bit to the end of a block of data. A typical example is ASCII transmission, in which a parity bit is attached to each 7-bit ASCII character. The value of this bit is selected so that the character has an even number of 1s (even parity) or an odd number of 1s (odd parity). So, for example, if the transmitter is transmitting an ASCII G (1110001) and using odd parity, it will append a 1 and transmit 11100011. The receiver examines the received character and, if the total number of 1s is odd, assumes that no error has occurred. If one bit (or any odd number of bits) is erroneously inverted during transmission (for example, 11QO0011), then the receiver will detect an error. It is used to detect a single bit error. Note, however, that if two (or any even number) of bits are inverted due to error, an undetected error occurs. The use of the parity bit is not foolproof, as noise impulses are often long enough to destroy more than one bit, particularly at high data rates.

Cyclic Redundancy Check (CRC)

One of the most common and one of the most powerful, error-detecting codes is the cyclic redundancy check (CRC) (also known as Polynomial Code), which can be described as follows. Given a k bit block of bits, or message, the transmitter generates an n-bit sequence, known as a frame check sequence (FCS), so that the resulting frame, consisting of k + n bits, is exactly divisible by some predetermined number. The receiver then divides the incoming frame by that number and, if there is no remainder, assumes there was no error.

Example:

Original message (k) : 1101011011

Poly (predetermined number) : 10011

We append n zeros to the original message

Message after appending n (4 for this example) zeros: 11010110110000

Now we simply divide the augmented message by the poly using CRC arithmetic, which is binary division.

```
            1100001010 = Quotient (nobody cares about the quotient)
          _______________
10011 ) 11010110110000 = Augmented message (1101011011 + 0000)
=Poly  10011,,.,,....
```

```
-----,,.,,....
 10011,.,,....
 10011,.,,....
 -----,.,,....
  00001.,,....
  00000.,,....
  -----.,,....
   00010,,....
   00000,,....
   -----,,....
    00101,....
    00000,....
    -----,....
     01011....
     00000....
     -----....
      10110...
      10011...
      -----...
       01010..
       00000..
       -----..
        10100.
        10011.
        -----.
         01110
         00000
         -----
          1110 = Remainder = THE CHECKSUM!!!!
```

The division yields a quotient, which we throw away, and a remainder, which is the calculated checksum. This ends the calculation. Usually, the checksum is then appended to the message and the result transmitted. In this case the transmission would be: 11010110111110. At the receiving end, the transmitted message (original message + checksum) will be divided by the checksum. If the remainder is zero, there is no error. Otherwise there is an error.

3.5 Flow Control

Flow control is a technique for assuring that a transmitting entity does not overwhelm a receiving entity with data. The receiving entity typically allocates a data buffer of some maximum length for a transfer. When data are received, the receiver must do a certain amount of processing before passing the data to the higher-level software. In the absence of flow control, the receiver's buffer may fill up and overflow while it is processing old data. There are different mechanisms of enforcing flow control at data link layer. Some of them are discussed below.

3.5.1 Stop-and-Wait Flow Control

The simplest form of flow control, known as stop-and-wait flow control, works as follows. A source entity transmits a frame. After reception, the destination entity indicates its willingness to accept another frame by sending back an acknowledgment to the frame just received. The source must wait until it receives the acknowledgment before sending the next frame. The destination can thus stop the flow of data by simply withholding acknowledgment. This procedure works fine and, indeed, can hardly be improved upon when a message is sent in a few large frames.

With the use of multiple frames for a single message, the stop-and-wait procedure may be inadequate. The essence of the problem is that only one frame at a time can be in transit. In essence, for very high data rates, or for very long distances between sender and receiver, stop-and-wait flow control provides inefficient line utilization.

3.5.2 Sliding-Window Flow Control

The essence of the problem described in stop and wait is that only one frame at a time can be in transit. In situations where the bit length of the link is greater than the frame length, serious inefficiencies result. Efficiency can be greatly improved by allowing multiple frames to be in transit at the same time.

Let us examine how this might work for two stations, A and B, connected via a full-duplex link. Station B allocates buffer space for *n* frames. Thus, *B* can accept n frames, and A is allowed to send *n* frames without waiting for any acknowledgments. To keep track of which frames have been acknowledged, each is labeled with a sequence number. B acknowledges a frame by sending an acknowledgment that includes the sequence number of the next frame expected. This acknowledgment also implicitly announces that B is prepared to receive the next n frames, beginning with the number specified. This scheme can also be used to acknowledge multiple

frames. For example, B could receive frames 2,3, and 4, but withhold acknowledgment until frame 4 has arrived: by then returning an acknowledgment with sequence number *5*, B acknowledges frames 2,3, and 4 at one time. A maintains a list of sequence numbers that it is allowed to send, and B maintains a list of sequence numbers that it is prepared to receive. Each of these lists can be thought of as a *window* of frames. The operation is referred to as sliding-window flow control.

3.6 Error Control (correction)

Error control refers to mechanisms to detect and correct errors that occur in the transmission of frames. Data are sent as a sequence of frames; frames arrive in the same order in which they are sent; and each transmitted frame suffers an arbitrary and variable amount of delay before reception. In addition, we admit the possibility of two types of errors:

Lost frame: A frame fails to arrive at the other side. For example, a noise burst may damage a frame to the extent that the receiver is not aware that a frame has been transmitted.

Damaged frame: A recognizable frame does arrive, but some of the bits are in error (have been altered during transmission). The most common techniques for error control are based on some or all of the following ingredients:

Error detection: As discussed in the preceding section.

Positive acknowledgment: The destination returns a positive acknowledgment to successfully received, error-free frames.

Retransmission after timeout: The source retransmits a frame that has not been acknowledged after a predetermined amount of time.

Negative acknowledgment and retransmission; The destination returns a negative acknowledgment to frames in which an error is detected. The source retransmits such frames.

Collectively, these mechanisms are all referred to as **automatic repeat request** (ARQ); the effect of ARQ is to turn an unreliable data link into a reliable one. Three versions of ARQ have been standardized:

- Stop-and-wait ARQ
- Go-back-N ARQ
- Selective-reject ARQ

All of these forms are based on the use of the flow control technique discussed earlier

3.6.1 Stop-and-Wait ARQ

Stop-and-wait ARQ is based on the stop-and-wait flow-control technique outlined previously. The source station transmits a single frame and then must await an acknowledgment (ACK). No other data frames can be sent until the destination station's reply arrives at the source station. Two sorts of errors could occur. First, the frame that arrives at the destination could be damaged; the receiver detects this by using the error detection technique referred to earlier and simply discards the frame. To account for this possibility, the source station is equipped with a timer. After a frame is transmitted, the source station waits for an acknowledgment. If no acknowledgment is received by the time the timer expires, then the same frame is sent again. Note that this method requires that the transmitter maintain a copy of a transmitted frame until an acknowledgment is received for that frame. The second sort of error is a damaged acknowledgment. Consider the following situation. Station ***A*** sends a frame. The frame is received correctly by station B, which responds with an acknowledgment (ACK). The ACK is damaged in transit and is not recognizable by ***A,*** which will therefore time-out and resend the same frame. This duplicate frame arrives and is accepted by B, which has therefore accepted two copies of the same frame as if they were separate. To avoid this problem, frames are alternately labeled with 0 or 1, and positive acknowledgments are of the form ACKO and ACK1. In keeping with the sliding-window convention, an ACKO acknowledges receipt of a frame numbered 1 and indicates that the receiver is ready for a frame numbered 0. The principal advantage of stop-and-wait ARQ is its simplicity. Its principal disadvantage is that stop-and-wait is an inefficient mechanism.

3.6.2 Go-back-N ARQ and Selective Repeat ARQ

Go-Back-N ARQ is a specific instance of ARQ Protocol, in which the sending process continues to send a number of frames specified by a *window size* without receiving an acknowledgment packet from the receiver.

The receiver keeps track of the sequence number of the next frame it expects to receive, and sends that number with every acknowledgment it sends. If a frame from the sender does not reach the receiver or if it is damaged, the receiver will stop acknowledging received frames. Once the sender has sent all of the frames in its *window*, it will detect that all of the frames since the first lost frame are *outstanding*, and will go back to sequence number of the last acknowledgment it received from the receiver process and fill its window starting with that

frame and continue the process over again. This approach can waste a lot of bandwidth if the error rate is high.

Lets see the case in which one frame (frame 2) is lost while the receiver's window is large. Frames 0 and 1 are correctly received and acknowledged. Frame 2, however, is damaged or lost. The sender, unaware of this problem, continues to send frames until the timer for frame 2 expires. Then it backs up to frame 2 and starts all over with it, sending 2, 3, 4, etc. all over again.

The other general strategy for handling errors is called selective repeat. **Selective Repeat ARQ** is a specific instance of ARQ Protocol, in which the sending process continues to send a number of frames specified by a *window size* even after a frame loss. Unlike Go-Back-N ARQ, the receiving process will continue to accept and acknowledge frames sent after an initial error. In other words, when it is used, a bad frame that is received is discarded, but good frames received after it are buffered. When the sender times out, only the oldest unacknowledged frame is retransmitted. If that frame arrives correctly, the receiver can deliver to the network layer, in sequence, all the frames it has buffered. Selective repeat is often combined with having the receiver send a negative acknowledgement (NAK) when it detects an error, for example, when it receives a checksum error or a frame out of sequence. NAKs stimulate retransmission before the corresponding timer expires and thus improve performance.

3.7 Access methods

In any broadcast network, the key issue is how to determine who gets to use the channel when there is competition for it. To make this point clearer, consider a conference call in which six people, on six different telephones, are all connected so that each one can hear and talk to all the others. It is very likely that when one of them stops speaking, two or more will start talking at once, leading to chaos. In a face-to-face meeting, chaos is avoided by external means, for example, at a meeting, people raise their hands to request permission to speak. When only a single channel is available, determining who should go next is much harder. Many protocols for solving the problem are known. In the literature, broadcast channels are sometimes referred to as *multiaccess channels or random access channels*.

The methods that can be used to determine how the shared media is accessed are called **Access methods**. The protocols used to determine who goes next on a multiaccess channel belong to a sublayer of the data link layer called the MAC (***Medium Access Control***) sublayer. The MAC

sublayer is especially important in LANs, many of which use a multiaccess channel as the basis for communication. WANs, in contrast, use point-to-point links, except for satellite networks. As you can understand the above paragraph the traditional data link layer is divided in to LLC and MAC. The separation is done because the logic required to manage access to a shared-access medium is not found in traditional layer-2 data link control. The following paragraphs discuss different access methods that are used in LAN.

3.7.1 ALOHA

The earliest of access methods, known as ALOHA, was developed for packet radio networks. However, it is applicable to any shared transmission medium. ALOHA, or pure ALOHA as it is sometimes called, is a true free-for-all. Whenever a station has a frame to send, it does so. The station then listens for an amount of time equal to the maximum possible round-trip propagation delay on the network (twice the time it takes to send a frame between the two most widely separated stations) plus a small fixed time increment. If the station hears an acknowledgment during that time, fine; otherwise, it resends the frame. If the station fails to receive an acknowledgment after repeated transmissions, it gives up. A receiving station determines the correctness of an incoming frame by examining a frame check- sequence field. If the frame is valid and if the destination address in the frame header matches the receiver's address, the station immediately sends an acknowledgment. The frame may be invalid due to noise on the channel or because another station transmitted a frame at about the same time. In the latter case, the two frames may interfere with each other at the receiver so that neither gets through; this is known as a ***collision.*** If a received frame is determined to be invalid, the receiving station simply ignores the frame.

ALOHA is as simple as can be, and pays a penalty for it. Because the number of collisions rises rapidly with increased load, the maximum utilization of the channel is only about 18%. To improve efiiciency, a modification of ALOHA, known as slotted ALOHA, was developed. In this scheme, time on the channel is organized into uniform slots whose size equals the frame transmission time. Some central clock or other technique is needed to synchronize all stations. Transmission is permitted to begin only at a slot boundary. Thus, frames that do overlap will do so totally. This increases the maximum utilization of the system to about 37%.

3.7.2 Carrier Sense Multiple Access (CSMA)

Both ALOHA and slotted ALOHA exhibit poor utilization. Both fail to take advantage of one of the key properties of both packet radio and LANs, which is that propagation delay between stations is usually very small compared to frame transmission time. Consider the following observations. If the station-to-station propagation time is large compared to the frame transmission time, then, after a station launches a frame, it will be a long time before other stations know about it. During that time, one of the other stations may transmit a frame; the two frames may interfere with each other and neither gets through. Indeed, if the distances are great enough, many stations may begin transmitting, one after the other, and none of their frames get through safe. Suppose, however, that the propagation time is small compared to frame transmission time. In that case, when a station launches a frame, all the other stations know it almost immediately. So, if they had any sense, they would not try transmitting until the first station was done. Collisions would be rare because they would occur only when two stations began to transmit almost simultaneously. Another way to look at it is that a short delay time provides the stations with better feedback about the state of the network; this information can be used to improve efficiency.

The foregoing observations led to the development of carrier-sense multiple access (CSMA). With CSMA, a station wishing to transmit first listens to the medium to determine if another transmission is in progress (carrier sense). If the medium is in use, the station must wait. If the medium is idle, the station may transmit. It may happen that two or more stations attempt to transmit at about the same time. If this happens, there will be a collision; the data from both transmissions will be garbled and not received successfully. To account for this, a station waits a reasonable amount of time, after transmitting, for an acknowledgment, taking into account the maximum round-trip propagation delay, and the fact that the acknowledging station must also contend for the channel in order to respond. If there is no acknowledgment, the station assumes that a collision has occurred and retransmits.

One can see how this strategy would be effective for networks in which the average frame transmission time is much longer than the propagation time. Collisions can occur only when more than one user begins transmitting within a short time (the period of the propagation delay). If a station begins to transmit a frame, and there are no collisions during the time it takes for the

leading edge of the packet to propagate to the farthest station, then there will be no collision for this frame because all other stations are now aware of the transmission.

The maximum utilization achievable using CSMA can far exceed that of ALOHA or slotted ALOHA. The maximum utilization depends on the length of the frame and on the propagation time; the longer the frames or the shorter the propagation time, the higher the utilization.

With CSMA, an algorithm is needed to specify what a station should do if the medium is found busy. The most common approach, and the one used in IEEE 802.3, is the *1-persistent technique.* A station wishing to transmit listens to the medium and obeys the following rules:

1. If the medium is idle, transmit; otherwise, go to step 2.
2. If the medium is busy, continue to listen until the channel is sensed idle; then transmit immediately.

If two or more stations are waiting to transmit, a collision is guaranteed. Things get sorted out only after the collision.

3.7.3 CSMA/CD

CSMA, although more efficient than ALOHA or slotted ALOHA, still has one glaring inefficiency: When two frames collide, the medium remains unusable for the duration of transmission of both damaged frames. For long frames, compared to propagation time, the amount of wasted capacity can be considerable. This waste can be reduced if a station continues to listen to the medium while transmitting. This leads to the following rules for CSMA/CD:

1. If the medium is idle, transmit; otherwise, go to step 2.

2. If the medium is busy, continue to listen until the channel is idle, then transmit immediately.

3. If a collision is detected during transmission, transmit a brief jamming signal to assure that all stations know that there has been a collision and then cease transmission.

4. After transmitting the jamming signal, wait a random amount of time, then attempt to transmit again. (Repeat from step 1.)

3.7.4 CSMA/CA

CSMA/CA belongs to a class of CSMA protocols. CSMA/CA stands for: **Carrier Sense Multiple Access** With **Collision Avoidance**. In CSMA, a station wishing to transmit has to first listen to the channel for a predetermined amount of time so as to check for any activity on the

channel. If the channel is sensed "idle" then the station is permitted to transmit. If the channel is sensed as "busy" the station has to defer its transmission. This is the essence of both CSMA/CA and CSMA/CD. In CSMA/CA , once the channel is clear, a station sends a signal telling all other stations not to transmit, and then sends its packet. In Ethernet 802.3, the station continues to wait for a time, and checks to see if the channel is still free. If it is free, the station transmits, and waits for an acknowledgment signal that the packet was received. Collision avoidance is used to improve the performance of **CSMA** by attempting to be less "greedy" on the channel. If the channel is sensed busy before transmission then the transmission is deferred for a "random" interval. This reduces the probability of collisions on the channel.

CSMA/CA is used where CSMA/CD cannot be implemented due to the nature of the channel. CSMA/CA is used in 802.11 based wireless LANs. One of the problems of wireless LANs is that it is not possible to listen while sending, therefore collision detection is not possible. Another reason is the hidden terminal problem, whereby a node A, in range of the receiver R, is not in range of the sender S, and therefore cannot know that S is transmitting to R.

3.7.5 Token Passing

A token is a special control frame on token ring, token bus, and FDDI (Fiber Distributed Data Interface) networks that determines which stations can transmit data on a shared network. In token-passing networks a token is passed around the network from device to device. The node that has the token can transmit. When the data transmission is complete, the token is released so that other devices may use the network media. Unlike contention-based networks, such as Ethernet, workstations on token-based networks do not compete for access to the network. Only the station that obtains the token can transmit. Other stations wait for the token rather than trying to access the network on their own. On Ethernet networks (that uses CSMA/CD and CSMA/CA), "collisions" occur when two or more workstations attempt to access the network at the same time. They must back off and try again later, which reduces performance-especially as the number of workstations attached to a network segment increases.

In token ring networks, a station takes possession of a token and changes one bit, converting the token to a SFS (start-of-frame sequence). A field exists in the token in which workstations can indicate the type of priority required for the transmission. The priority setting is basically a request to other stations for future use of the token. The other stations compare a workstation's

request for priority with their own priority levels. If the workstation's priority is higher, the other stations will grant the workstation access to the token for an extended period.

The main advantage of token-passing networks is that they are deterministic. In other words, it is easy to calculate the maximum time that will pass before a device has the opportunity to send data. This explains the popularity of token-passing networks in some real-time environments such as factories, where machinery must be capable of communicating at a determinable interval.

3.8 Ethernet

Ethernet (also known as *IEEE 802.3 standard*) is a data transmission standard for local area networks based on the following principle:

"All machines on an Ethernet network are connected to the same communication line, made up of cylindrical cables"

It is developed at the PARC in Palo Alto CA in the early 70's by Robert Metcalf. It is a Carrier Sense Multiple Access transmission method. This means if the channel is available then any device connected to the medium can transmit. It is the most common LAN technology today well suited for any and all types of applications and the cost of such a network is not very high . Different variants of Ethernet technologies are distinguished according to the type and diameter of the cables used. The following table summarizes different Ethernet technologies.

Table 3.1 Different types of Ethernet Technologies

Abbreviation	Name	Cable	Connector	Speed	Ports
10Base2	Thin Ethernet	Coaxial cable (50 Ohms) with a thin diameter	BNC	10 Mb/s	185m
10Base5	Thick Ethernet	Coaxial cable with a thick diameter (0.4 inch)	BNC	10Mb/s	500m
10Base-T	Standard Ethernet	Twisted pair (category 3)	RJ-45	10 Mb/s	100m
100Base-TX	Fast Ethernet	Double twisted pair (category 5)	RJ-45	100 Mb/s	100m
100Base-FX	Fast Ethernet	Multimode fibre optic (type 62.5/125)		100 Mb/s	2 km

1000Base-T	Gigabit Ethernet	Double twisted pair (category 5)	RJ-45	1000 Mb/s	100m
1000Base-LX	Gigabit Ethernet	Monomode or multimode fibre optic		1000 Mb/s	550m
1000Base-SX	Gigabit Ethernet	Multimode fibre optic		1000 Mbit/s	550m
10GBase-SR	10Gigabit Ethernet	Multimode fibre optic		10 Gbit/s	500m
10GBase-LX4	10Gigabit Ethernet	Multimode fibre optic		10 Gbit/s	500m

Ethernet Addressing

Ethernet addressing uses the *Media Access Control (MAC) address* burned into each and every Ethernet Network Interface Card (NIC). The MAC, or hardware address, is a 48-bit (6 byte) address written in a hexadecimal format.

3.9 Token Ring

The *Token Ring* architecture was developed in the mid-1980s by IBM. It is the preferred method of networking by IBM and is therefore found primarily in large IBM mini- and mainframe installations.

The goal of IBM's version of Token Ring was to facilitate a simple wiring structure using twisted-pair cable that connects a computer to the network through a wall socket, with the main wiring located in a centralized location.

Their token-passing ring access method, more than their physical cable layout, distinguishes Token Ring networks from other networks.

The architecture of a typical Token Ring network begins with a physical ring. However, in its IBM implementation, a star-wired ring, computers on the network are connected to a central hub. The logical ring represents the token's path between computers. The actual physical ring of cable is in the hub. Users are part of a ring, but they connect to it through a hub. A Token Ring network includes the following features:

- Star-wired ring topology
- Token-passing access method
- Shielded and unshielded twisted-pair (IBM Types 1, 2, and 3) cabling
- Transfer rates of 4 and 16 Mbps

- Baseband transmission

3.10 FDDI

FDDI is short for *Fiber Distributed Data Interface*. It is based on fiberoptic transmission. It's also based on a ring topology and token passing. It's advanced technology in the form of token ring over fiber. FDDI was developed for two primary reasons: to support and help extend the capabilities of older LANs, such as Ethernet and Token Ring, and to provide a reliable infrastructure for businesses moving even *mission-critical* applications to networks. FDDI is suitable for use as a backbone connecting a number of smaller LANs, and it can provide the core of a network as large as a Metropolitan Area Network (MAN). In that sense, FDDI is more than LAN but less than WAN. In addition, because FDDI transfers information extremely quickly (100 Mbps), it is often used to connect high-end devices, such as mainframes, minicomputers, and peripherals, or to connect high-performance devices within a LAN. Engineering or video/graphics workstations, for instance, benefit from FDDI because they need considerable bandwidth in order to transfer large amounts of data at satisfactorily high speeds.

As its name indicates, FDDI was developed around the idea of using optical cable. This is, in fact, the type of cable used, especially when high-speed transmission is needed over relatively long distances (2000 to 10,000 meters). However, over shorter distances (about 100 meters, or 330 feet), FDDI can also be implemented on less expensive copper cable. In all, FDDI supports different types of cable:

- **Multimode fiberoptic cable.** This type of cable can be used over a maximum of 2000 meters and uses LEDs as a light source.
- **Single mode fiberoptic cable.** This can be used over a maximum of 10,000 meters and uses lasers as a light source. Single mode cable is thinner at the core than multimode, but it provides greater bandwidth because of the way the light impulse travels through the cable.
- **Category 5 Unshielded Twisted Pair copper wiring.** This cable contains eight wires and can be used over distances up to 30 meters.
- **IBM Type 1 Shielded Twisted Pair copper wiring.** This is a shielded cable that contains two pairs of twisted wires, with each pair also shielded.

FDDI (Fiber Distributed Data Interface)

FDDI topology and operation are similar to Token Ring, *except* that FDDI is primarily based on optical transmission. In addition, FDDI is characterized by two *counter-rotating* rings (known as a *dual-ring topology*).

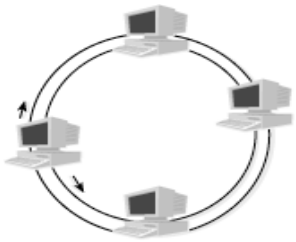

Figure 3.2 : FDDI

Why two rings? The second one is there mostly for insurance. Normally in a FDDI network, one ring (known as the primary ring) actually carries the tokens and data, and the secondary ring remains idle and is used as a backup for fault tolerance—insurance. Because the secondary ring is available if needed, whenever a nonfunctioning node causes a break in the primary ring, traffic can "wrap" around the problem node and continue carrying data, only in the opposite direction and on the secondary ring. That way, even if a node goes down, the network continues to function.

Sometimes, however, both rings are used for data. In this case, the data travels in one direction (clockwise) on one ring, and in the other direction (counterclockwise) on the other ring. Using both rings to carry data means that twice as many frames can circulate at the same time and, therefore, the speed of the network can double—from 100 Mbps to 200 Mbps.

FDDI token passing

Token passing on a FDDI network works much the way it does on a Token Ring network. That is, nodes pass a token around the ring, and only the node with the token is allowed to transmit a frame. There is a twist to this, however, that's related to FDDI's fault tolerance. When a node on the ring detects a problem, it doesn't simply sit around. Instead, it generates a frame known as a *beacon* and sends it on to the network. As neighboring nodes detect the beacon, they too begin to transmit beacons, and so it goes around the ring. When the node that started the process eventually receives its own beacon back—usually after the network has switched to the secondary ring—it then assumes that the problem has been isolated or resolved, generates a new token, and starts once again.

A FDDI network, as already mentioned, cannot include rings longer than 100 kilometers apiece. Another restriction on a FDDI network is that it cannot support more than 500 nodes per ring. Although the overall network topology must conform to a logical ring, the network doesn't actually have to look like a circle. It can include stars connected to hubs or concentrators, and it can even include trees—collections of hubs connected in a hierarchy. As long as the stars and trees connect in a logical ring, the FDDI network is happy.

Generally FDDI is a high-speed, high-bandwidth network based on optical transmissions. It is relatively expensive to implement, although the cost can be held down by the mixing of fiberoptic with copper cabling. Because it has been around for a few years, however, it has been fine-tuned to a high level of stability. It is most often used as a network backbone, for connecting high-end computers (mainframes, minicomputers, and peripherals), and for LANs connecting high-performance engineering, graphics, and other workstations that demand rapid transfer of large amounts of data.