UNIT- II

DATA LINK LAYER FUNCTIONS (SERVICES)

1. **Providing services to the network layer:**

   1 Unacknowledged connectionless service.

   Appropriate for low error rate and real-time traffic. Ex: Ethernet

   2. Acknowledged connectionless service.

   Useful in unreliable channels, WiFi. Ack/Timer/Resend

   3. Acknowledged connection-oriented service.

   Guarantee frames are received exactly once and in the right order. Appropriate over long, unreliable links such as a satellite channel or a long-distance telephone circuit

2. **Framing:** Frames are the streams of bits received from the network layer into manageable data units. This division of stream of bits is done by Data Link Layer.
3. **Physical Addressing**: The Data Link layer adds a header to the frame in order to define physical address of the sender or receiver of the frame, if the frames are to be distributed to different systems on the network.
4. **Flow Control:** A receiving node can receive the frames at a faster rate than it can process the frame. Without flow control, the receiver's buffer can overflow, and frames can get lost. To overcome this problem, the data link layer uses the flow control to prevent the sending node on one side of the link from overwhelming the receiving node on another side of the link. This prevents traffic jam at the receiver side.
5. **Error Control:** Error control is achieved by adding a trailer at the end of the frame. Duplication of frames are also prevented by using this mechanism. Data Link Layers adds mechanism to prevent duplication of frames.

   Error detection: Errors can be introduced by signal attenuation and noise. Data Link Layer protocol provides a mechanism to detect one or more errors. This is achieved by adding error detection bits in the frame and then receiving node can perform an error check.

   Error correction: Error correction is similar to the Error detection, except that receiving node not only detects the errors but also determine where the errors have occurred in the frame.
6. **Access Control**: Protocols of this layer determine which of the devices has control over the link at any given time, when two or more devices are connected to the same link**.**
7. **Reliable delivery**: Data Link Layer provides a reliable delivery service, i.e., transmits the network layer datagram without any error. A reliable delivery service is accomplished with transmissions and acknowledgements. A data link layer mainly provides the reliable delivery

service over the links as they have higher error rates and they can be corrected locally, link at which an error occurs rather than forcing to retransmit the data.

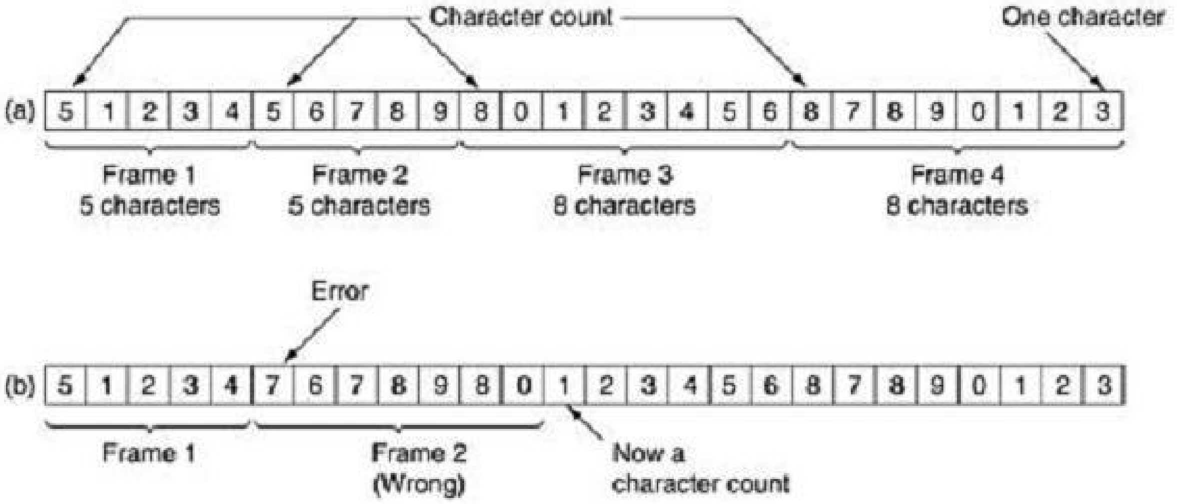
8. **Half-Duplex & Full-Duplex**: In a Full-Duplex mode, both the nodes can transmit the data at the same time. In a Half-Duplex mode, only one node can transmit the data at the same time.

FRAMING:

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 **(framing**). 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).We will look at four framing methods:

1. Character count.
2. Flag bytes with byte stuffing.
3. Starting and ending flags, with bit stuffing.
4. Physical layer coding violations.

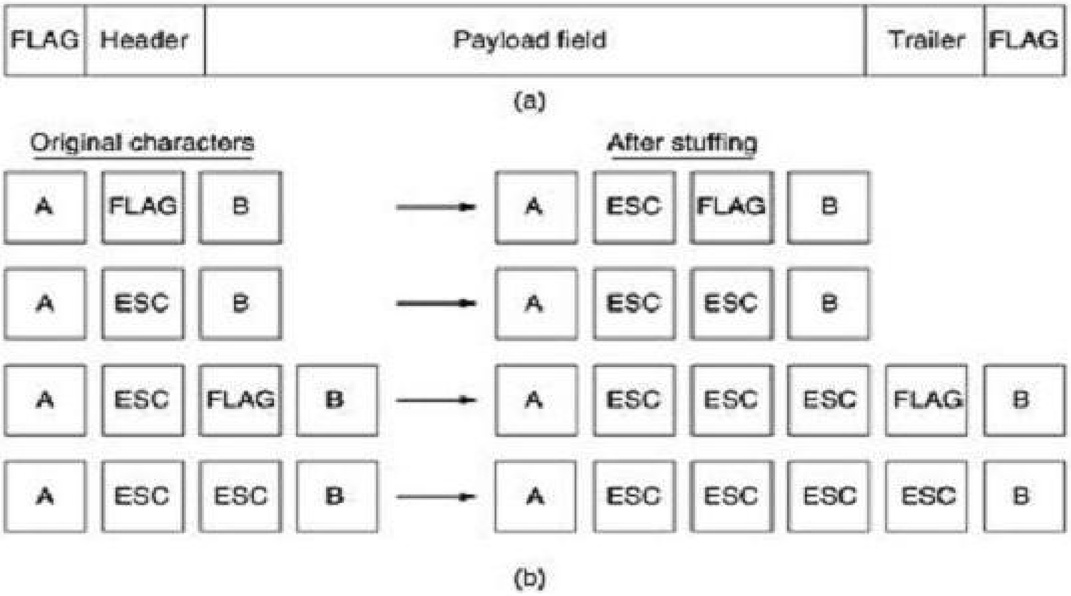
Character count method uses a field in the header to specify the number of characters in the frame. When the data link layer at the destination sees the character count, it knows how many characters follow and hence where the end of the frame is. This technique is shown in Fig. (a) For four frames of sizes 5, 5, 8, and 8 characters, respectively.



A character stream. (a) Without errors. (b) With one error

The trouble with this algorithm is that the count can be garbled by a transmission error. For example, if the character count of 5 in the second frame of Fig. (b) becomes a 7, the destination will get out of synchronization and will be unable to locate the start of the next frame. Even if the checksum is incorrect so the destination knows that the frame is bad, it still has no way of telling where the next frame starts. Sending a frame back to the source asking for a retransmission does not help either, since the destination does not know how many characters to skip over to get to the start of the retransmission. For this reason, the character count method is rarely used anymore.

Flag bytes with byte stuffing method gets around the problem of resynchronization after an error by having each frame start and end with special bytes. In the past, the starting and ending bytes were different, but in recent years most protocols have used the same byte, called a flag byte, as both the starting and ending delimiter, as shown in Fig. (a) as FLAG. In this way, if the receiver ever loses synchronization, it can just search for the flag byte to find the end of the current frame. Two consecutive flag bytes indicate the end of one frame and start of the next one.



(a) A frame delimited by flag bytes (b) Four examples of byte sequences before and after byte stuffing

It may easily happen that the flag byte's bit pattern occurs in the data. This situation will usually interfere with the framing. One way to solve this problem is to have the sender's data link layer insert a special escape byte (ESC) just before each ''accidental'' flag byte in the data. The data link layer on the receiving end removes the escape byte before the data are given to the network layer. This technique is called byte stuffing or character stuffing.

Thus, a framing flag byte can be distinguished from one in the data by the absence or presence of an escape byte before it.

What happens if an escape byte occurs in the middle of the data? The answer is that, it too is stuffed with an escape byte. Thus, any single escape byte is part of an escape sequence, whereas a doubled one indicates that a single escape occurred naturally in the data. Some examples are shown in Fig. (b). In all cases, the byte sequence delivered after de stuffing is exactly the same as the original byte sequence.

A major disadvantage of using this framing method is that it is closely tied to the use of 8-bit characters. Not all character codes use 8-bit characters. For example UNICODE uses 16-bit characters, so a new technique had to be developed to allow arbitrary sized characters

Starting and ending flags, with bit stuffing 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 (in fact, a flag byte). 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. This bit stuffing is analogous to byte stuffing, in which an escape byte is stuffed into the outgoing character stream before a flag byte in the data.

When the receiver sees five consecutive incoming 1 bits, followed by a 0 bit, it automatically de- stuffs (i.e., deletes) the 0 bit. Just as byte stuffing is completely transparent to the network layer in both computers, so is bit stuffing. If the user data contain the flag pattern, 01111110, this flag is transmitted as 011111010 but stored in the receiver's memory as 01111110.

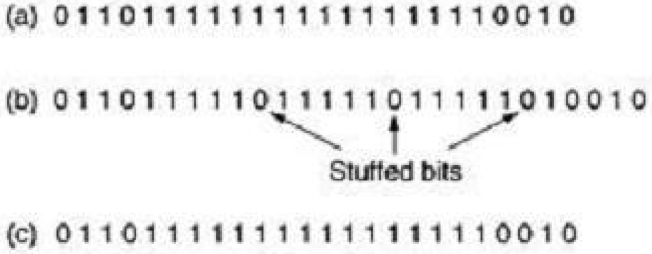
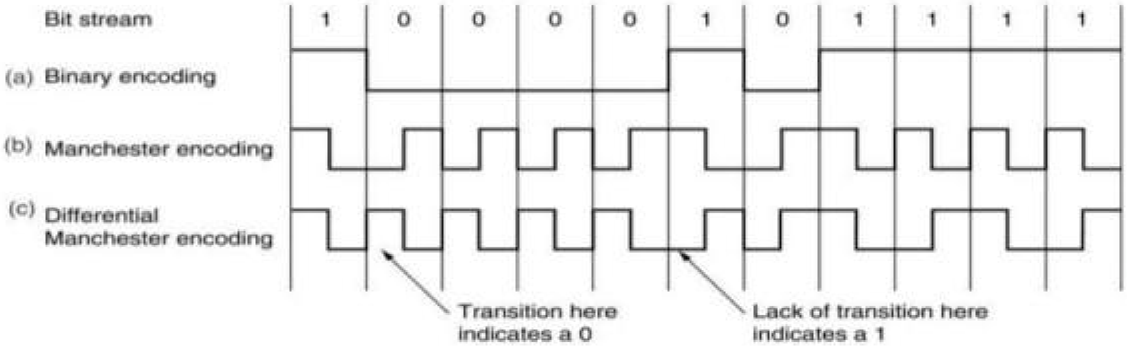

(a) 011011111111111111110010
(b) 011011111011111011111010010
Stuffed bits
(c) 011011111111111111110010

Fig:Bit stuffing. (a) The original data. (b) The data as they appear on the line. (c) The data as they are stored in the receiver's memory after destuffing.

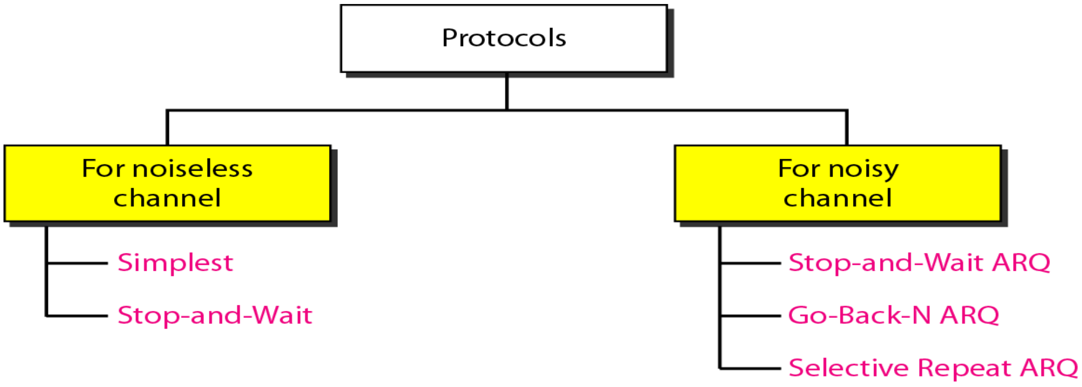
With bit stuffing, the boundary between two frames can be unambiguously recognized by the flag pattern. Thus, if the receiver loses track of where it is, all it has to do is scan the input for flag sequences, since they can only occur at frame boundaries and never within the data.

Physical layer coding violations method of framing is only applicable to networks in which the encoding on the physical medium contains some redundancy. For example, some LANs encode 1 bit of data by using 2 physical bits. Normally, a 1 bit is a high-low pair and a 0 bit is a low-high pair. The scheme means that every data bit has a transition in the middle, making it easy for the receiver to locate the bit boundaries. The combinations high-

high and low-low are not used for data but are used for delimiting frames in some protocols.

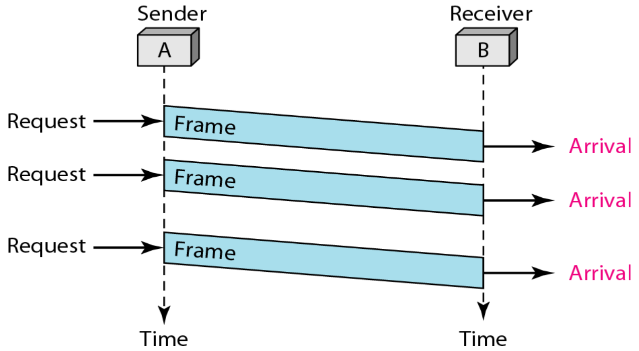

Bit stream
1
0
0
0
0
1
0
1
1
1
1
(a) Binary encoding
(b) Manchester encoding
(c) Differential Manchester encoding
Transition here indicates a 0
Lack of transition here indicates a 1

As a final note on framing, many data link protocols use combination of a character count with one of the other methods for extra safety. When a frame arrives, the count field is used to locate the end of the frame. Only if the appropriate delimiter is present at that position and the checksum is correct is the frame accepted as valid. Otherwise, the input stream is scanned for the next delimiter



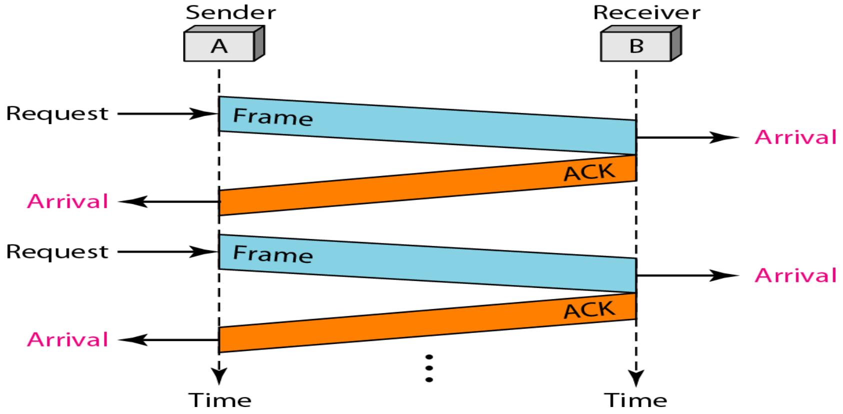
ELEMENTARY DATA LINK PROTOCOLS

Simplest Protocol



It is very simple. The sender sends a sequence of frames without even thinking about the receiver. Data are transmitted in one direction only. Both sender & receiver always ready. Processing time can be ignored. Infinite buffer space is available. And best of all, the communication channel between the data link layers never damages or loses frames. This thoroughly unrealistic protocol, which we will nickname ‘‘Utopia,’’ .The utopia protocol is unrealistic because **it does not handle either flow control or error correction**

Stop-and-wait Protocol



It is still very simple. The sender sends one frame and waits for feedback from the receiver. When the ACK arrives, the sender sends the next frame

It is Stop-and-Wait Protocol because the sender sends one frame, stops until it receives confirmation from the receiver (okay to go ahead), and then sends the next frame. We still have unidirectional communication for data frames, but auxiliary ACK frames (simple tokens of acknowledgment) travel from the other direction. We add flow control to our previous protocol.

NOISY CHANNELS

Although the Stop-and-Wait Protocol gives us an idea of how to add flow control to its predecessor, noiseless channels are nonexistent. We can ignore the error (as we sometimes do), or we need to add error control to our protocols. We discuss three protocols in this section that use error control.

Sliding Window Protocols:

1 Stop-and-Wait Automatic Repeat Request

2 Go-Back-N Automatic Repeat Request

3 Selective Repeat Automatic Repeat Request

1 Stop-and-Wait Automatic Repeat Request

To detect and correct corrupted frames, we need to add redundancy bits to our data frame. When the frame arrives at the receiver site, it is checked and if it is corrupted, it is silently discarded. The detection of errors in this protocol is manifested by the silence of the receiver.

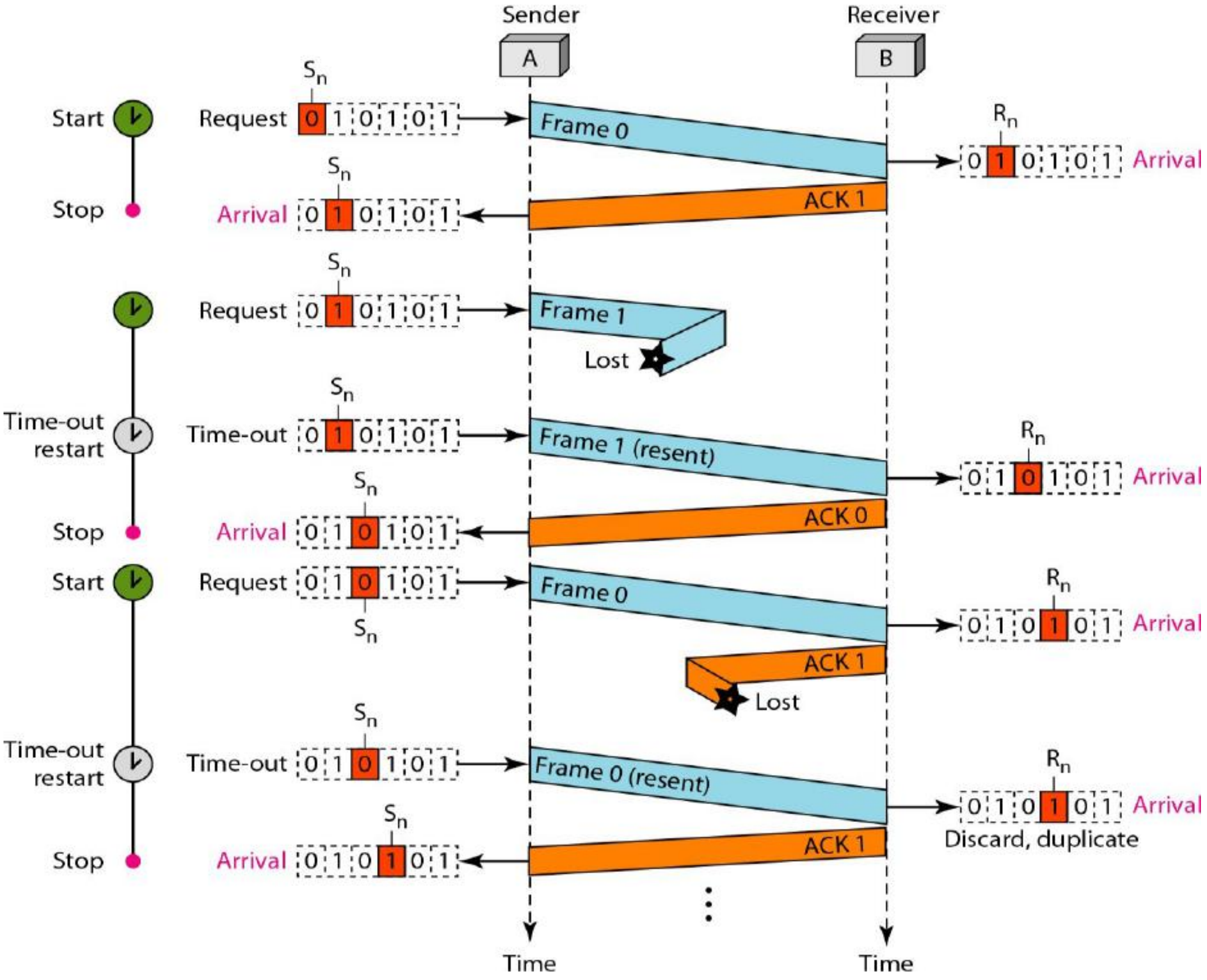
Lost frames are more difficult to handle than corrupted ones. In our previous protocols, there was no way to identify a frame. The received frame could be the correct one, or a duplicate, or a frame out of order. The solution is to number the frames. When the receiver receives a data frame that is out of order, this means that frames were either lost or duplicated

The lost frames need to be resent in this protocol. If the receiver does not respond when there is an error, how can the sender know which frame to resend? To remedy this problem, the sender keeps a copy of the sent frame. At the same time, it starts a timer. If the timer expires and there is no ACK for the sent frame, the frame is resent, the copy is held, and the timer is restarted. Since the protocol uses the stop-and-wait mechanism, there is only one specific frame that needs an ACK

Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and retransmitting of the frame when the timer expires

In Stop-and-Wait ARQ, we use sequence numbers to number the frames. The sequence numbers are based on modulo-2 arithmetic.

In Stop-and-Wait ARQ, the acknowledgment number always announces in modulo-2 arithmetic the sequence number of the next frame expected.



Bandwidth Delay Product:

Assume that, in a Stop-and-Wait ARQ system, the bandwidth of the line is 1 Mbps, and 1 bit takes 20 ms to make a round trip. What is the bandwidth-delay product? If the system data frames are 1000 bits in length, what is the utilization percentage of the link?

$$\boxed{(1 \times 10^6) \times (20 \times 10^{-3}) = 20{,}000 \text{ bits}}$$

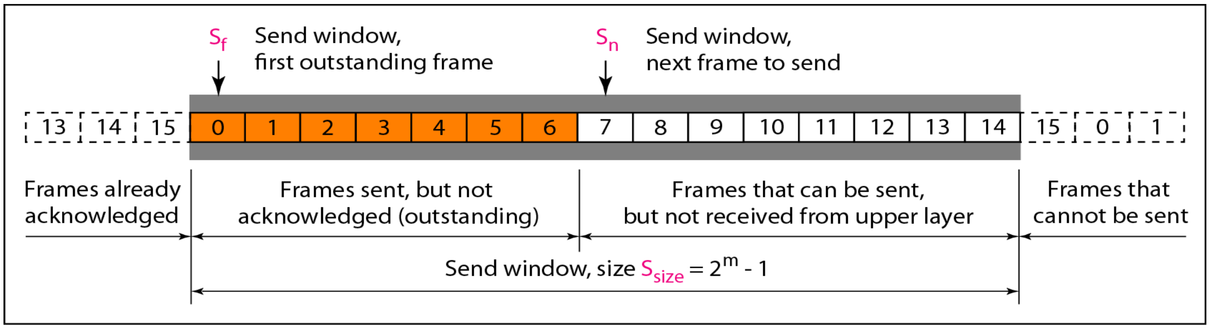
The link utilization is only 1000/20,000, or 5 percent. For this reason, for a link with a high bandwidth or long delay, the use of Stop-and-Wait ARQ wastes the capacity of the link.

2. Go-Back-N Automatic Repeat Request

To improve the efficiency of transmission (filling the pipe), multiple frames must be in transition while waiting for acknowledgment. In other words, we need to let more than one frame be outstanding to keep the channel busy while the sender is waiting for acknowledgment.

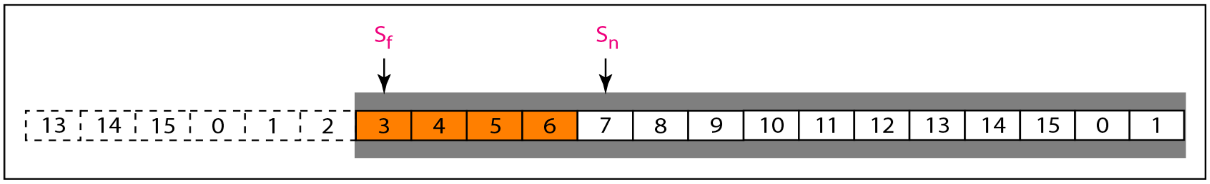
The first is called Go-Back-N Automatic Repeat. In this protocol we can send several frames before receiving acknowledgments; we keep a copy of these frames until the acknowledgments arrive.

In the Go-Back-N Protocol, the sequence numbers are modulo 2^m, where m is the size of the sequence number field in bits. The sequence numbers range from 0 to *2 power m*- 1. For example, if *m* is 4, the only sequence numbers are 0 through 15 inclusive.



S_f
Send window,
first outstanding frame
S_n
Send window,
next frame to send
13 14 15 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
Frames already acknowledged
Frames sent, but not acknowledged (outstanding)
Frames that can be sent, but not received from upper layer
Frames that cannot be sent
Send window, size $S_{size} = 2^m - 1$

a. Send window before sliding



b. Send window after sliding

The **sender window** at any time divides the possible sequence numbers into four regions.

The first region, from the far left to the left wall of the window, defines the sequence numbers belonging to frames that are already acknowledged. The sender does not worry about these frames and keeps no copies of them.

The second region, colored in Figure (a), defines the range of sequence numbers belonging to the frames that are sent and have an unknown status. The sender needs to wait to find out if these frames have been received or were lost. We call these outstanding frames.

The third range, white in the figure, defines the range of sequence numbers for frames that can be sent; however, the corresponding data packets have not yet been received from the network layer.

Finally, the fourth region defines sequence numbers that cannot be used until the window slides

The send window is an abstract concept defining an imaginary box of size $2^m - 1$ with three variables: S_f, S_n, and S_{size}. The variable *Sf* defines the sequence number of the first (oldest) outstanding frame. The variable *Sn* holds the sequence number that will be assigned to the next frame to be sent. Finally, the variable Ssize defines the size of the window.

Figure (b) shows how a send window can slide one or more slots to the right when an acknowledgment arrives from the other end. The acknowledgments in this protocol are cumulative, meaning that more than one frame can be acknowledged by an ACK frame. In Figure, frames 0, I, and 2 are acknowledged, so the window has slide to the right three slots. Note that the value of *Sf* is 3 because frame 3 is now the first outstanding frame.**The send window can slide one or more slots when a valid acknowledgment arrives.**

Receiver window: variable *Rn* (receive window, next frame expected) . The sequence numbers to the left of the window belong to the frames already received and acknowledged; the sequence numbers to the right of this window define the frames that cannot be received. Any received frame with a sequence number in these two regions is discarded. Only a frame with a sequence number matching the value of *Rn* is accepted and acknowledged. The receive window also slides, but only one slot at a time. When a correct frame is received (and a frame is received only one at a time), the window slides.(see below figure for receiving window)

The receive window is an abstract concept defining an imaginary box of size 1 with one single variable Rn. The window slides when a correct frame has arrived; sliding occurs one slot at a time

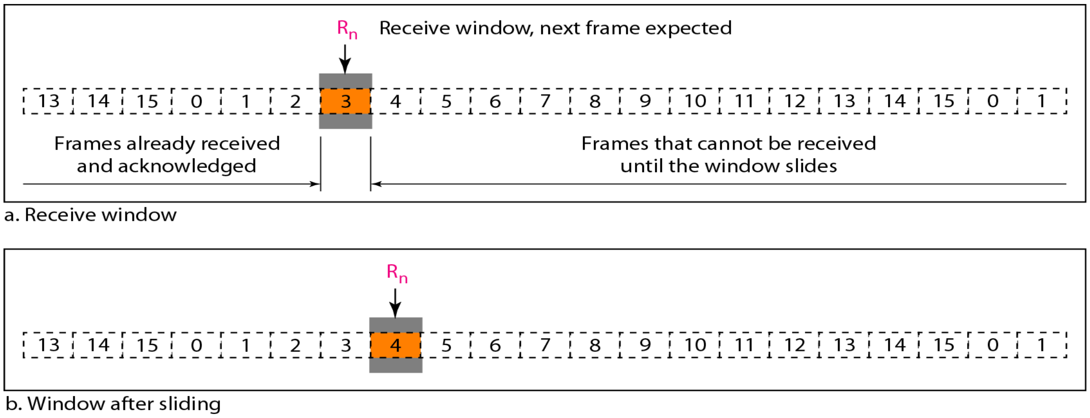


Fig: Receiver window (before sliding (a), After sliding (b))

Timers

Although there can be a timer for each frame that is sent, in our protocol we use only one. The reason is that the timer for the first outstanding frame always expires first; we send all outstanding frames when this timer expires.

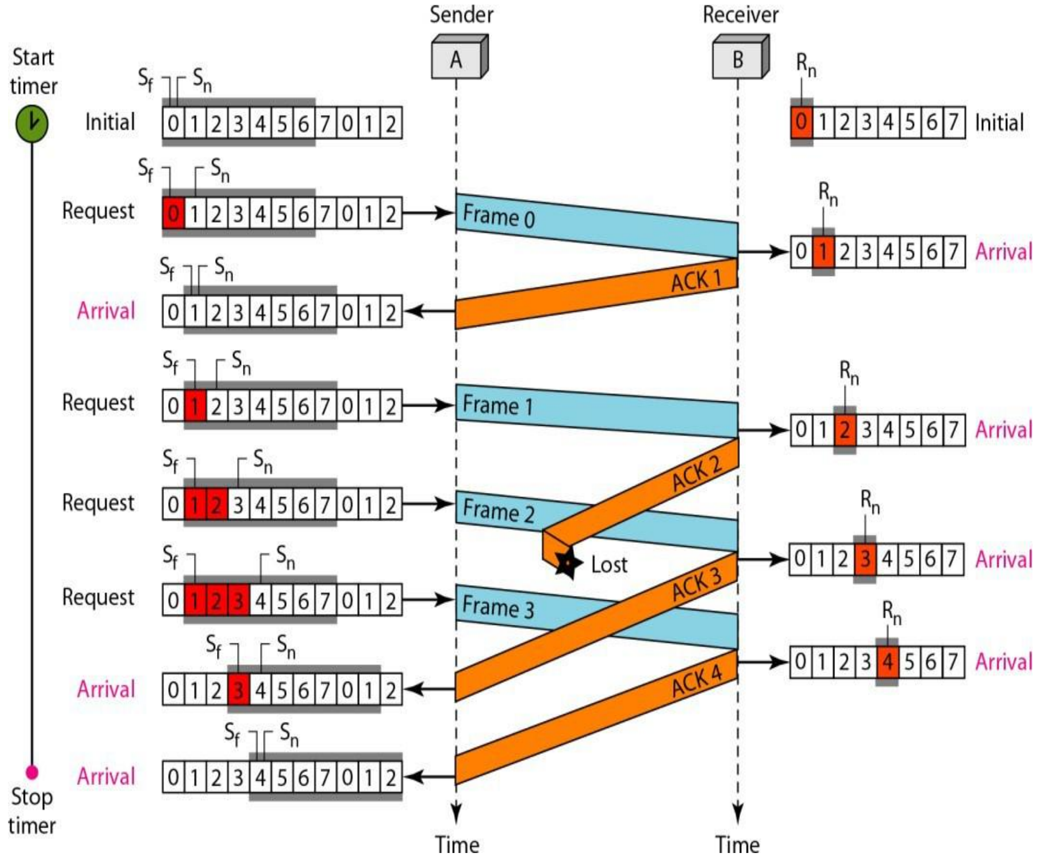
Acknowledgment

The receiver sends a positive acknowledgment if a frame has arrived safe and sound and in order. If a frame is damaged or is received out of order, the receiver is silent and will discard all subsequent frames until it receives the one it is expecting. The silence of the receiver causes the timer of the unacknowledged frame at the sender side to expire. This, in turn, causes the sender to go back and resend all frames, beginning with the one with the expired timer. The receiver does not have to acknowledge each frame received. It can send one cumulative acknowledgment for several frames.

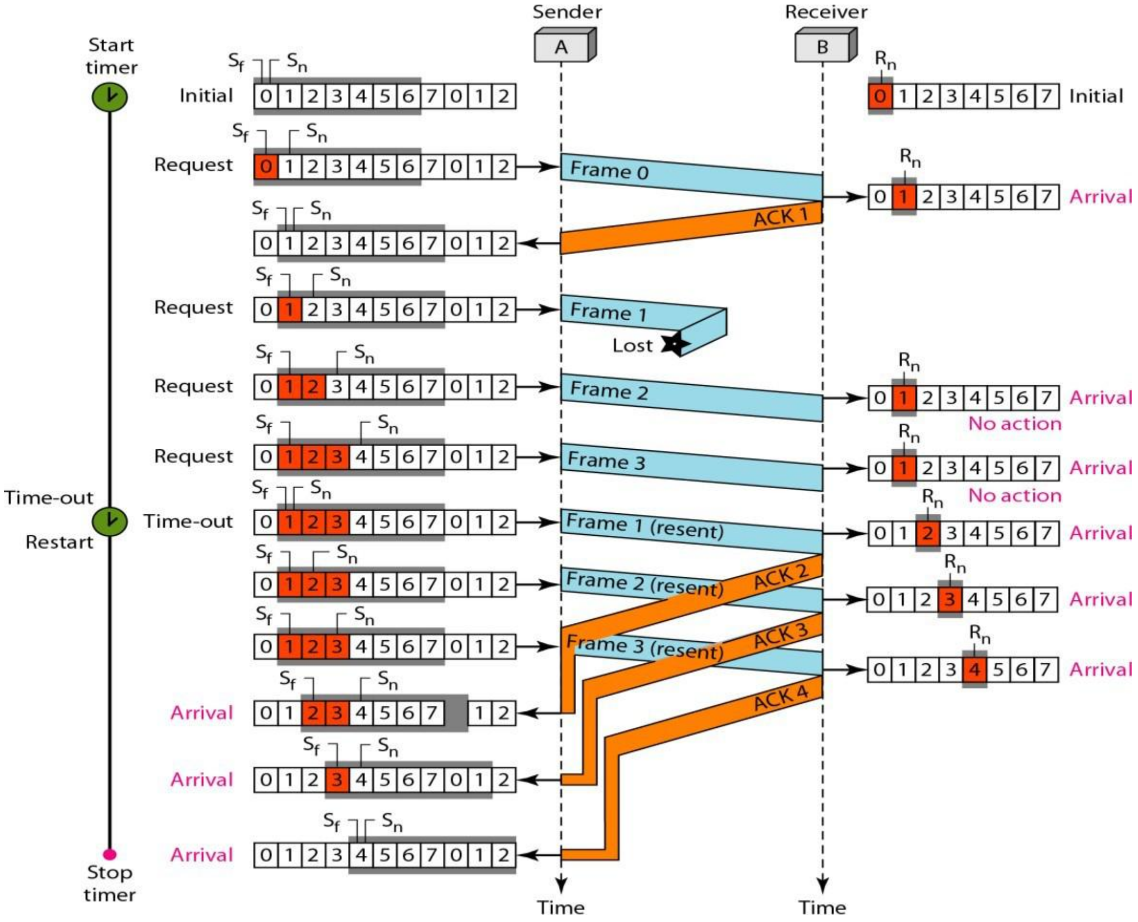
Resending a Frame

When the timer expires, the sender resends all outstanding frames. For example, suppose the sender has already sent frame 6, but the timer for frame 3 expires. This means that frame 3 has not been acknowledged; the sender goes back and sends frames 3,4,5, and 6 again. That is why the protocol is called *Go-Back-N* ARQ.

Below figure is an example(if ack lost) of a case where the forward channel is reliable, but the reverse is not. No data frames are lost, but some ACKs are delayed and one is lost. The example also shows how cumulative acknowledgments can help if acknowledgments are delayed or lost



Below figure is an example(if frame lost)



Stop-and-Wait ARQ is a special case of Go-Back-N ARQ in which the size of the send window is 1.

3 Selective Repeat Automatic Repeat Request

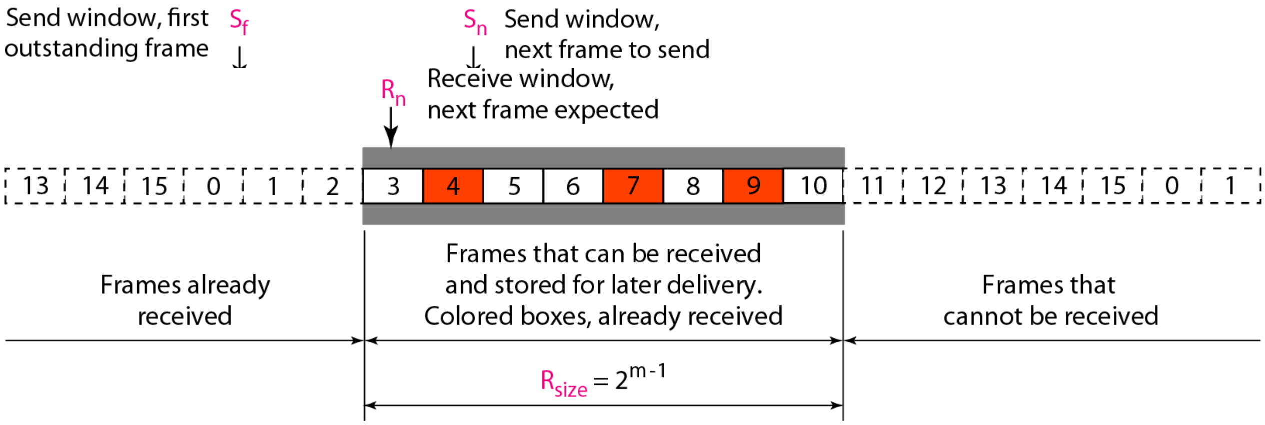
In Go-Back-N ARQ, The receiver keeps track of only one variable, and there is no need to buffer out-of- order frames; they are simply discarded. However, this protocol is very inefficient for a noisy link.

In a noisy link a frame has a higher probability of damage, which means the resending of multiple frames. This resending uses up the bandwidth and slows down the transmission.

For noisy links, there is another mechanism that does not resend *N* frames when just one frame is damaged; only the damaged frame is resent. This mechanism is called Selective Repeat ARQ.

It is more efficient for noisy links, but the processing at the receiver is more complex.

Sender Window (explain go-back N sender window concept (before & after sliding.) The only difference in sender window between Go-back N and Selective Repeat is Window size)


Send window, first outstanding frame
S_f
S_n
Send window, next frame to send
R_n
Receive window, next frame expected
13
14
15
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
0
1
Frames already received
Frames that can be received and stored for later delivery. Colored boxes, already received
Frames that cannot be received
$R_{size} = 2^{m-1}$

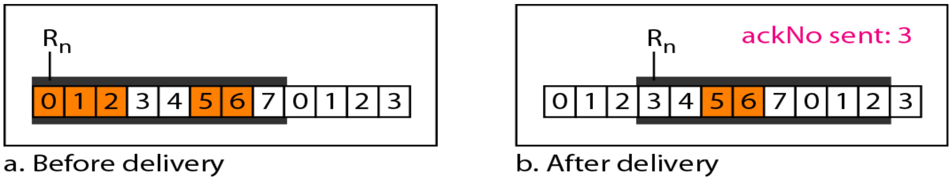
Receiver window

The receiver window in Selective Repeat is totally different from the one in Go Back-N. First, the size of the receive window is the same as the size of the send window *(2^{m-1}).*

The Selective Repeat Protocol allows as many frames as the size of the receiver window to arrive out of order and be kept until there is a set of in-order frames to be delivered to the network layer. Because the sizes of the send window and receive window are the same, all the frames in the send frame can arrive out of order and be stored until they can be delivered. However the receiver never delivers packets out of order to the network layer. Above Figure shows the receive window. Those slots inside the window that are colored define frames that have arrived out of order and are waiting for their neighbors to arrive before delivery to the network layer.

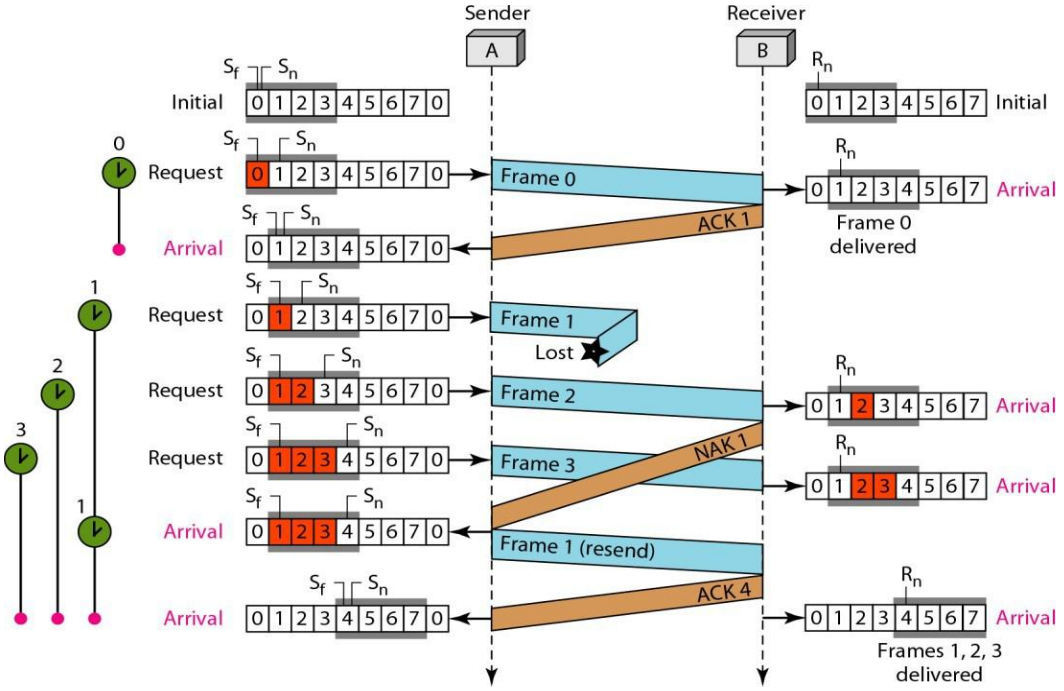
In Selective Repeat ARQ, the size of the sender and receiver window must be at most one-half of 2^m

Delivery of Data in Selective Repeat ARQ:



a. Before delivery b. After delivery

Flow Diagram



Differences between Go-Back N & Selective Repeat

One main difference is the number of timers. Here, each frame sent or resent needs a timer, which means that the timers need to be numbered (0, 1,2, and 3). The timer for frame 0 starts at the first request, but stops when the ACK for this frame arrives.

There are two conditions for the delivery of frames to the network layer: First, a set of consecutive frames must have arrived. Second, the set starts from the beginning of the window. After the first arrival, there was only one frame and it started from the beginning of the window. After the last arrival, there are three frames and the first one starts from the beginning of the window.

Another important point is that a NAK is sent.

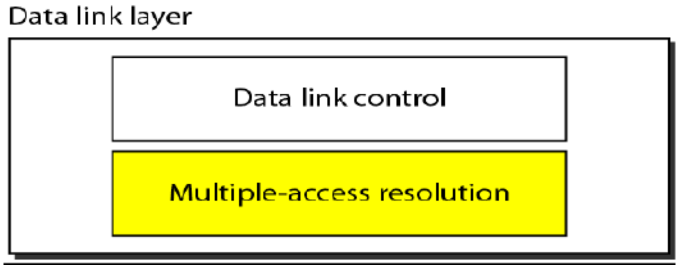
The next point is about the ACKs. Notice that only two ACKs are sent here. The first one acknowledges only the first frame; the second one acknowledges three frames. In Selective Repeat, ACKs are sent when data are delivered to the network layer. If the data belonging to *n* frames are delivered in one shot, only one ACK is sent for all of them.

Piggybacking

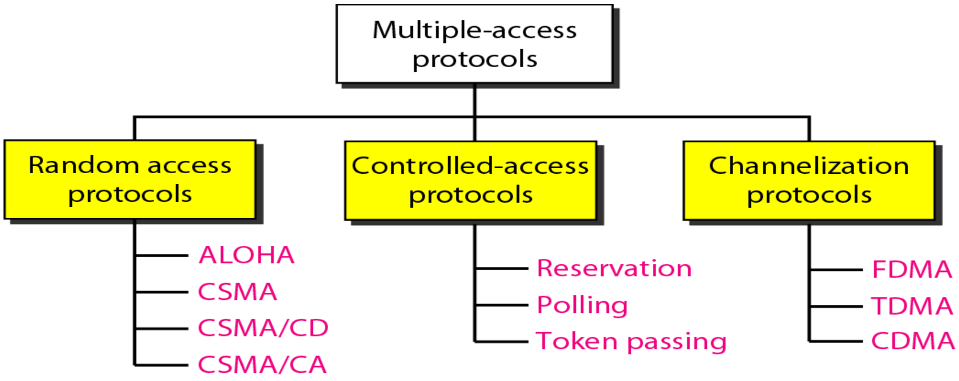
A technique called **piggybacking** is used to improve the efficiency of the bidirectional protocols. When a frame is carrying data from A to B, it can also carry control information about arrived (or lost) frames from B; when a frame is carrying data from B to A, it can also carry control information about the arrived (or lost) frames from A.

RANDOM ACCESS PROTOCOLS

We can consider the data link layer as two sub layers. The upper sub layer is responsible for data link control, and the lower sub layer is responsible for resolving access to the shared media



The upper sub layer that is responsible for flow and error control is called the logical link control (LLC) layer; the lower sub layer 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.



Taxonomy of multiple-access protocols

RANDOM ACCESS

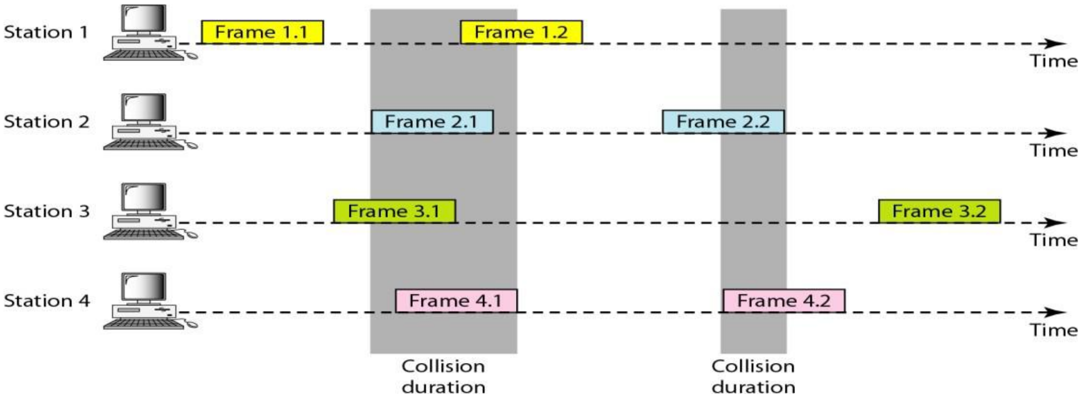
In random access or contention methods, no station is superior to another station and none is assigned the control over another.

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.

ALOHA

1 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. Below Figure shows an example of frame collisions in pure ALOHA.

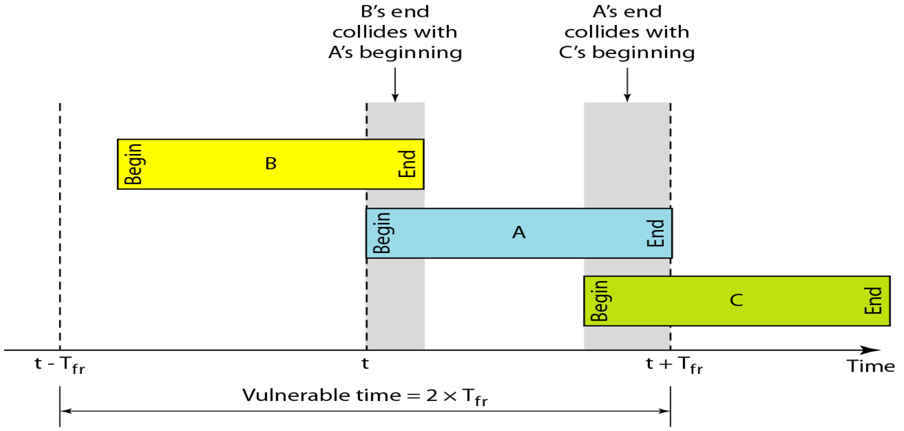


Frames in a pure ALOHA network

In pure ALOHA, the stations transmit frames whenever they have data to send.

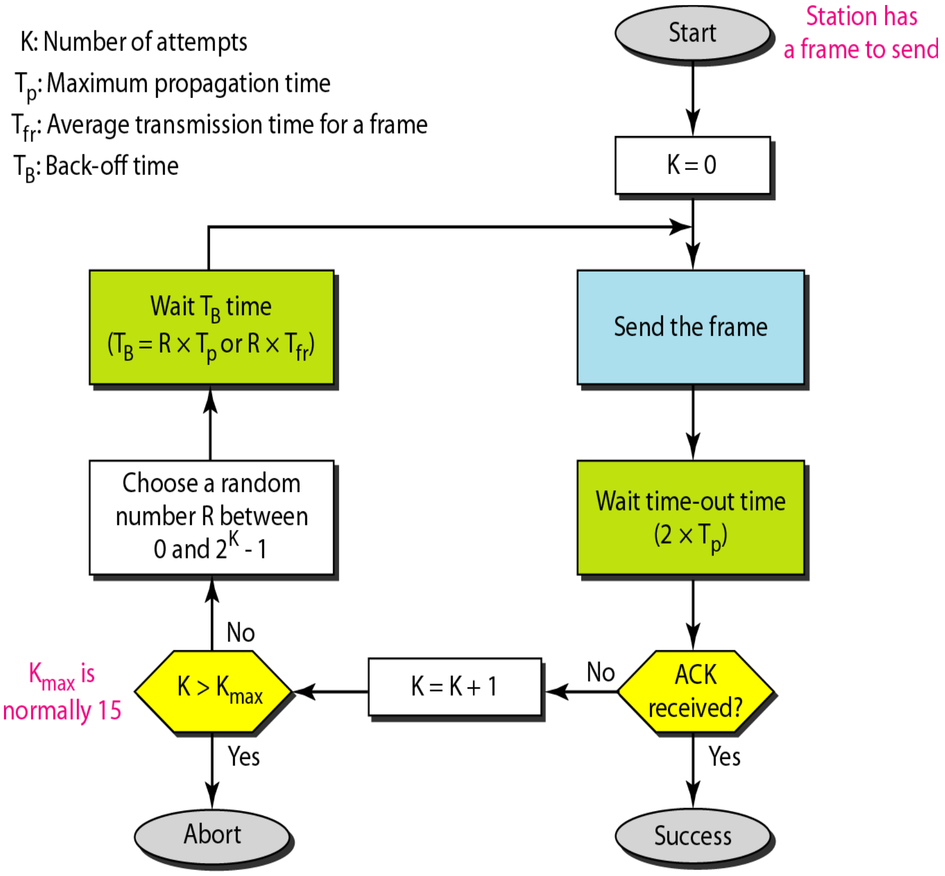
- When two or more stations transmit simultaneously, there is collision and the frames are destroyed.
- In pure ALOHA, whenever any station transmits a frame, it expects the acknowledgement from the receiver.
- If acknowledgement is not received within specified time, the station assumes that the frame (or acknowledgement) has been destroyed.
- If the frame is destroyed because of collision the station waits for a random amount of time and sends it again. This waiting time must be random otherwise same frames will collide again and again.
- Therefore pure ALOHA dictates that when time-out period passes, each station must wait for a random amount of time before resending its frame. This randomness will help avoid more collisions.

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 *Tfr* S to send. Below Figure shows the vulnerable time for station A.


B's end collides with A's beginning
A's end collides with C's beginning
Begin
B
End
Begin
A
End
Begin
C
End
t - T_fr
t
t + T_fr
Time
Vulnerable time = 2 × T_fr

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* + *Tfr .* 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, we see that the vulnerable time, during which a collision may occur in pure ALOHA, is 2 times the frame transmission time. Pure ALOHA vulnerable time = 2 x Tfr


K: Number of attempts
T_p: Maximum propagation time
T_{fr}: Average transmission time for a frame
T_B: Back-off time
Start
Station has a frame to send
K = 0
Wait T_B time ($T_B = R \times T_p$ or $R \times T_{fr}$)
Send the frame
Choose a random number R between 0 and 2^K - 1
Wait time-out time ($2 \times T_p$)
No
K_{max} is normally 15
$K > K_{max}$
K = K + 1
No
ACK received?
Yes
Yes
Abort
Success

Procedure for pure ALOHA protocol

Example

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 *Tfr* is 200 bits/200 kbps or 1 ms. The vulnerable time is 2 x 1 ms =2 ms. This means no station should send later than 1 ms before this station starts transmission and no station should start sending during the one I-ms period that this station is sending.

The throughput for pure ALOHA is S = G × e −2G . The maximum throughput Smax = 0.184 when G= (1/2).

PROBLEM

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**.** 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× $e^{-2\ G}$ or S = 0.135 (13.5 percent). This means that the throughput is 1000 × 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 × e^{-2G} or S = 0.184 (18.4 percent). This means that the throughput is 500 × 0.184 = 92 and that only 92 frames out of 500 will probably survive. Note that this is the maximum throughput case, percentage wise.

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 × $e\ -^{2G}$ or S = 0.152 (15.2 percent). This means that the throughput is 250 × 0.152 = 38. Only 38 frames out of 250 will probably survive.

2 Slotted ALOHA

Pure ALOHA has a vulnerable time of 2 x *Tfr .* 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 Tfr s and force the station to send only at the beginning of the time slot. Figure 3 shows an example of frame collisions in slotted ALOHA

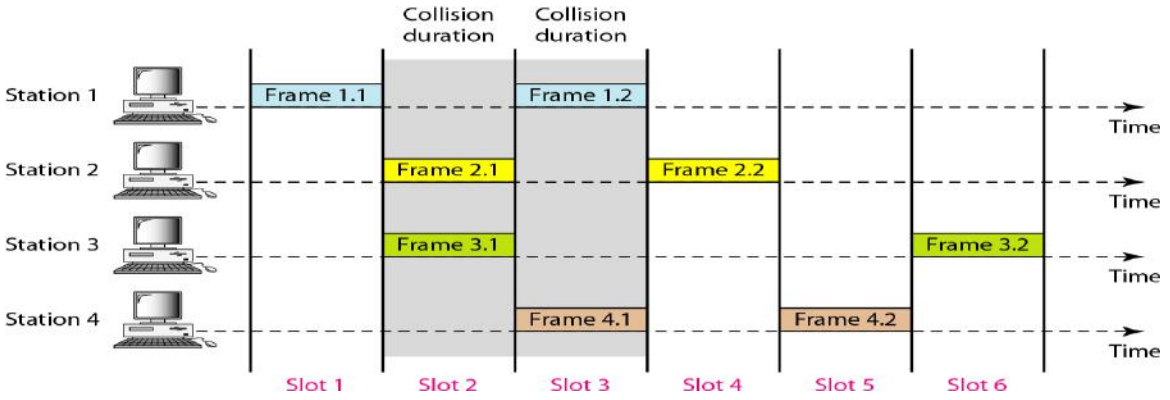
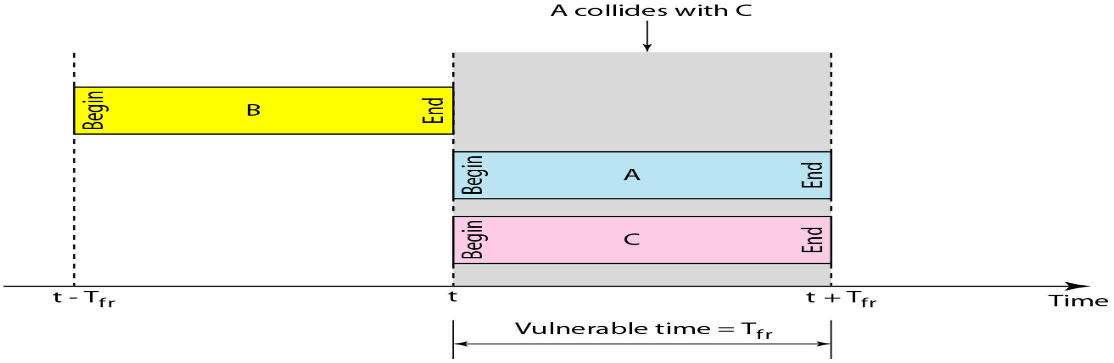


FIG:3

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 *Tfr* Figure 4 shows the situation

Below fig shows that the vulnerable time for slotted ALOHA is one-half that of pure ALOHA. <u>Slotted ALOHA vulnerable time = Tfr</u>



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

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

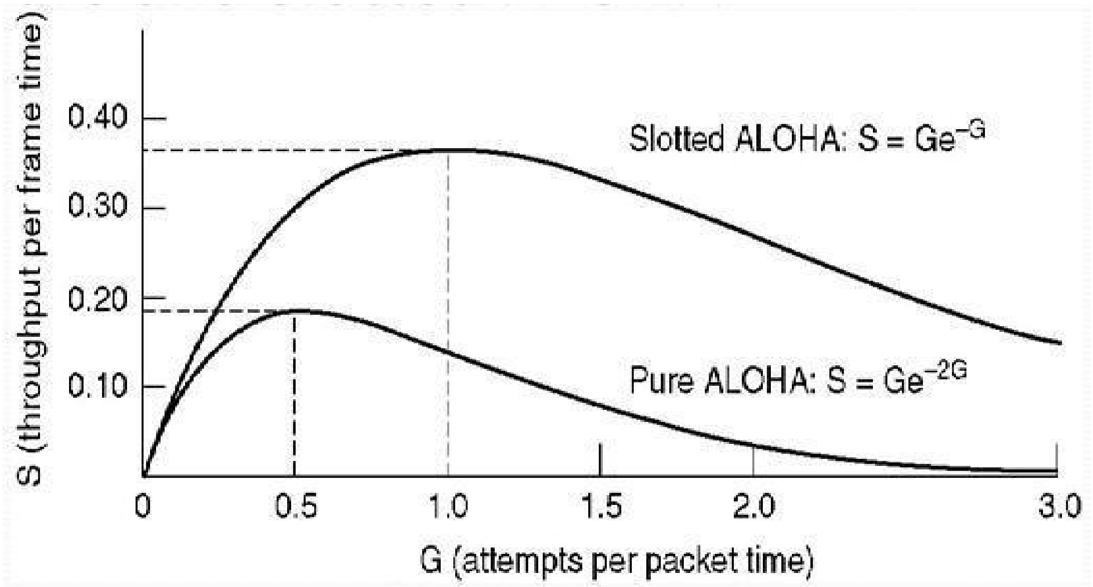
This situation is similar to the previous exercise except that the network is using slotted ALOHA instead of pure ALOHA. The frame transmission time is *200/200* kbps or 1 ms.

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

b. Here G is 1/2 In this case S =G x *e-G* or S =0.303 (30.3 percent). This means that the throughput is 500 x 0.0303 =151. Only 151 frames out of 500 will probably survive.

c. Now G is 1/4. In this case S =G x *e-G* or S =0.195 (19.5 percent). This means that the throughput is 250 x 0.195 = 49. Only 49 frames out of 250 will probably survive

Comparison between Pure Aloha & Slotted Aloha



Carrier Sense Multiple Access (CSMA)

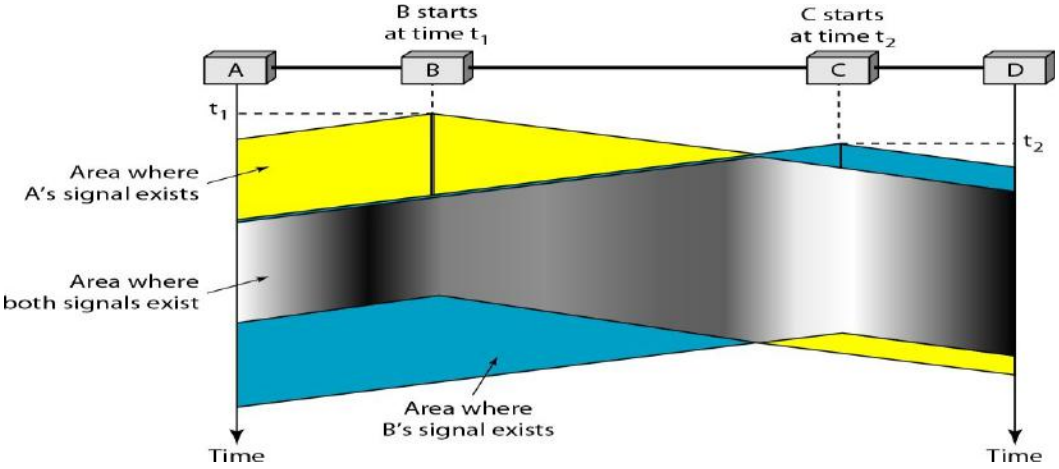
To minimize the chance of collision and, therefore, increase the performance, the CSMA method was developed. The chance of collision can be reduced if a station senses the medium before trying to use it. Carrier sense multiple access (CSMA) requires that each station first listen to the medium (or check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

The possibility of collision still exists because of propagation delay; 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 *tI*' station B senses the medium and finds it idle, so it sends a frame. At time *t2 (t2> tI)*' station C senses the medium and finds it idle because, at this time, the first bits from station B

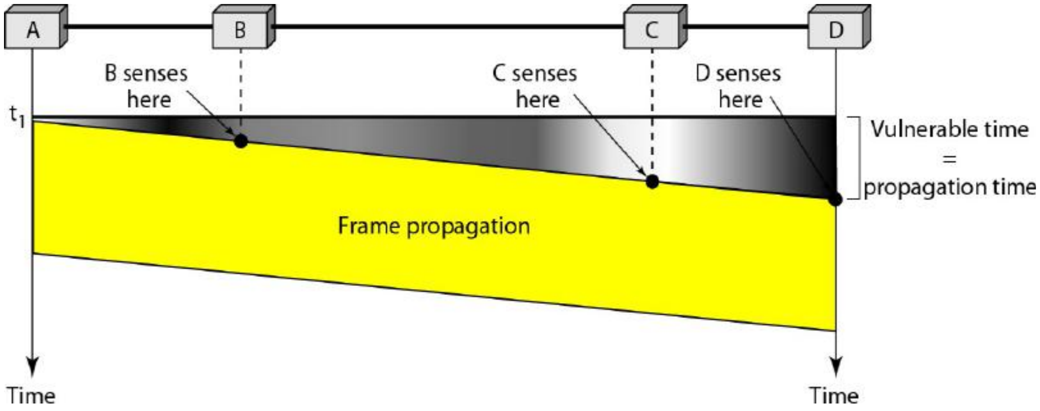
have not reached station C. Station C also sends a frame. The two signals collide and both frames are destroyed.



Space/time model of the collision in CSMA

Vulnerable Time

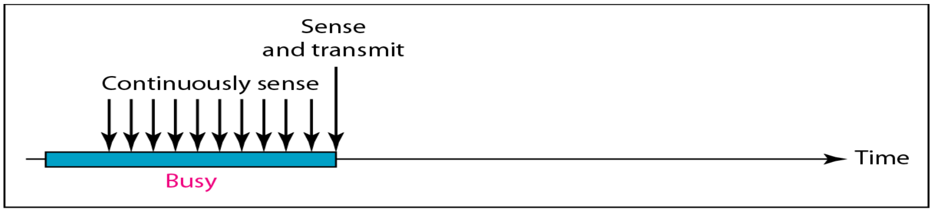
The vulnerable time for CSMA is the propagation time Tp . This is the time needed for a signal to propagate from one end of the medium to the other. When a station sends a frame, and any other station tries to send a frame during this time, a collision will result. But if the first bit of the frame reaches the end of the medium, every station will already have heard the bit and will refrain from sending



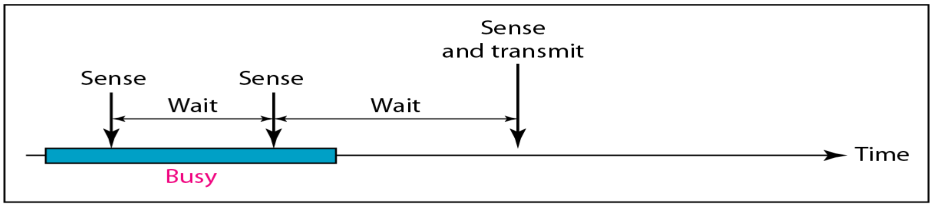
Vulnerable time in CSMA

Persistence Methods

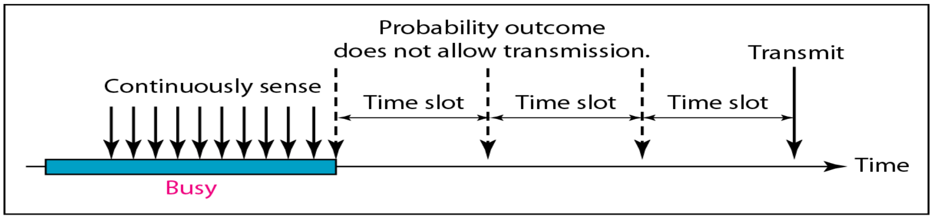
What should a station do if the channel is busy? What should a station do if the channel is idle? Three methods have been devised to answer these questions: the 1-persistent method, the non-persistent method, and the p-persistent method



a. 1-persistent



b. Nonpersistent



c. p-persistent

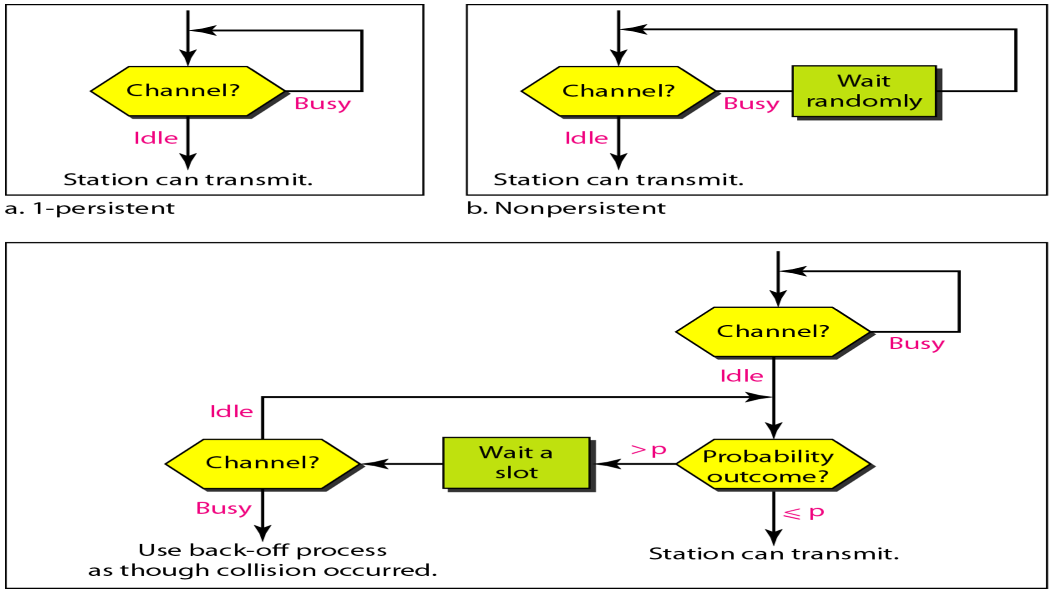
1-Persistent: In this method, after the station finds the line idle, it sends its frame immediately (with probability 1). This method has the highest chance of collision because two or more stations may find the line idle and send their frames immediately.

Non-persistent: a station that has a frame to send senses the line. If the line is idle, it sends immediately. If the line is not idle, it waits a random amount of time and then senses the line again. This 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: This 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 backoff procedure.



a. 1-persistent

b. Nonpersistent

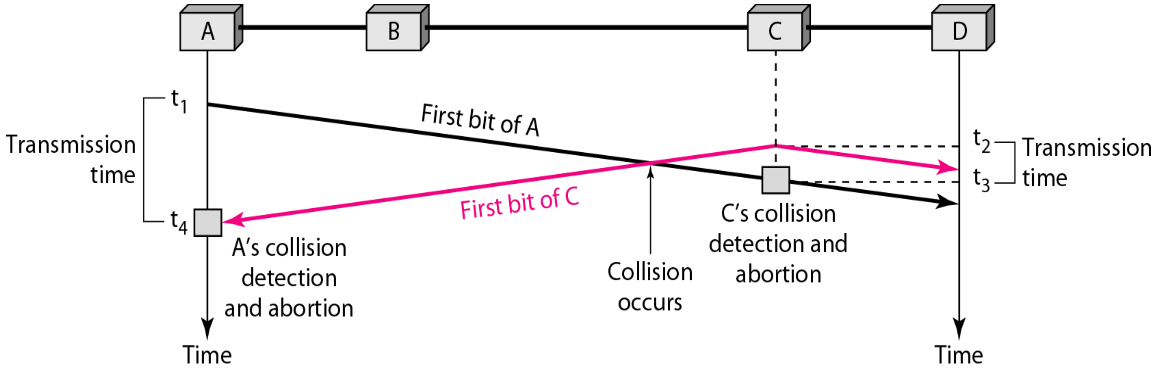
a. c. p-persistent

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 below Figure, stations A and C are involved in the collision.



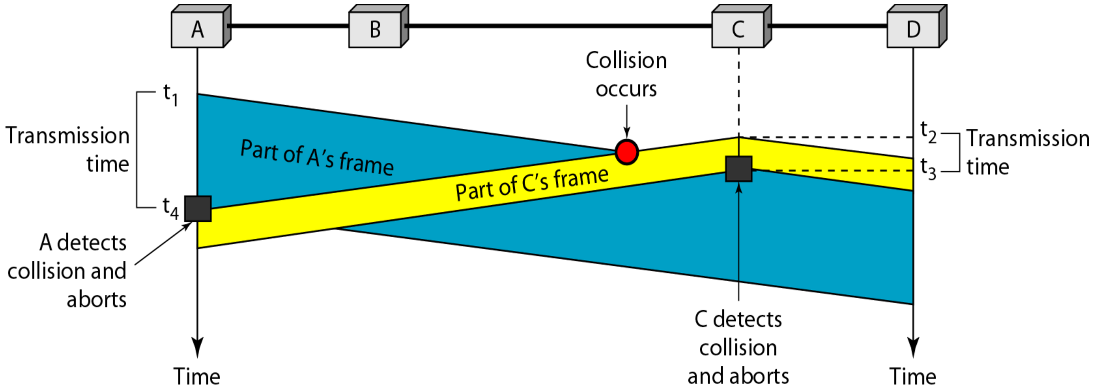
Collision of the first bit in CSMA/CD

At time *t* 1, station A has executed its persistence procedure and starts sending the bits of its frame. At time *t2,* 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 *t2.*Station C detects a collision at time *t3* when it receives the first bit of A's frame. Station C immediately (or after a short time, but we assume immediately) aborts transmission.

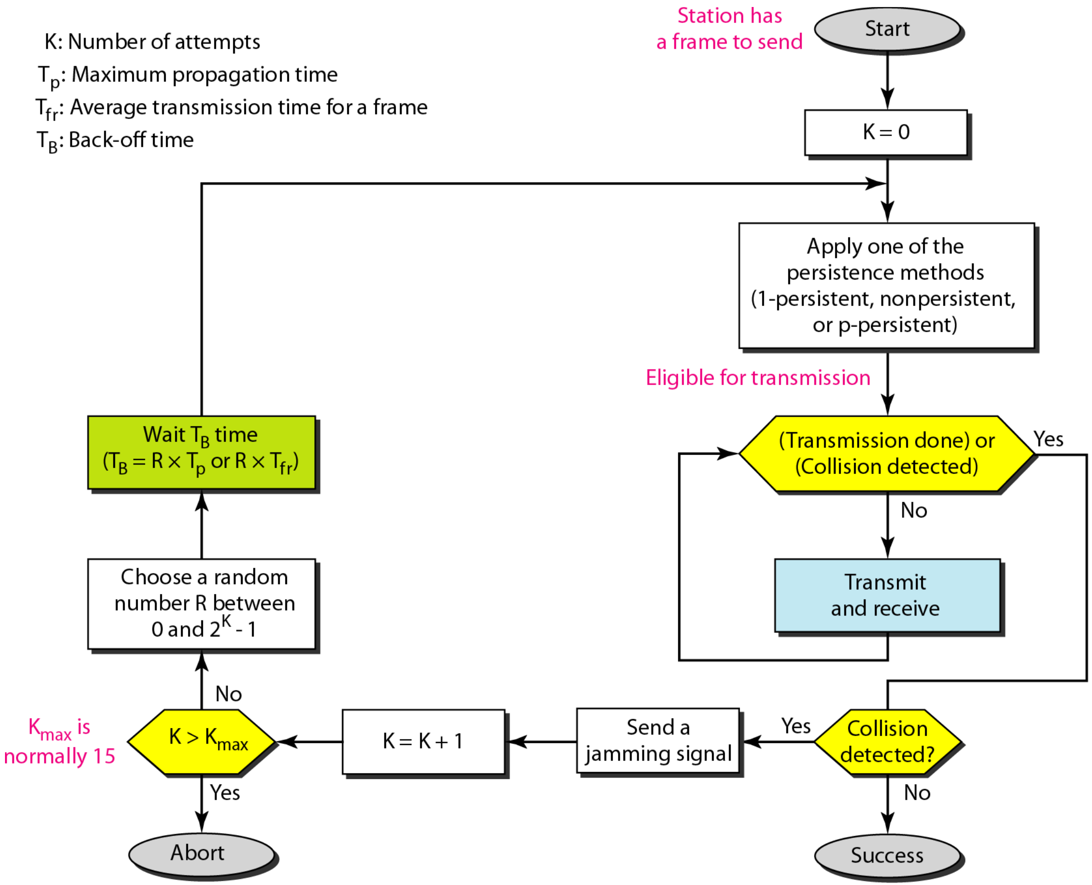
Station A detects collision at time *t4* when it receives the first bit of C's frame; it also immediately aborts transmission. Looking at the figure, we see that A transmits for the duration *t4 - tl;* C transmits for the duration *t3 - t2.*

Minimum Frame Size

For *CSMAlCD* 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 *Tp.* 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 *Tp* to reach the second, and the effect of the collision takes another time *Tp* to reach the first. So the requirement is that the first station must still be transmitting after *2Tp* .



Collision and abortion in CSMA/CD


K: Number of attempts
T_p: Maximum propagation time
T_{fr}: Average transmission time for a frame
T_B: Back-off time
Station has a frame to send
Start
K = 0
Apply one of the persistence methods (1-persistent, nonpersistent, or p-persistent)
Eligible for transmission
(Transmission done) or (Collision detected)
Yes
No
Transmit and receive
Wait T_B time ($T_B = R \times T_p$ or $R \times T_{fr}$)
Choose a random number R between 0 and $2^K - 1$
No
K_{max} is normally 15
$K > K_{max}$
Yes
Abort
K = K + 1
Send a jamming signal
Yes
Collision detected?
No
Success

Flow diagram for the CSMA/CD

PROBLEM

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

SOL

The frame transmission time is Tfr = 2 × Tp = 51.2 μs. This means, in the worst case, a station needs to transmit for a period of 51.2 μs to detect the collision. The minimum size of the frame is 10 Mbps × 51.2 μs = 512 bits or 64 bytes. This is actually the minimum size of the frame for Standard Ethernet**.**

DIFFERENCES BETWEEN ALOHA & CSMA/CD

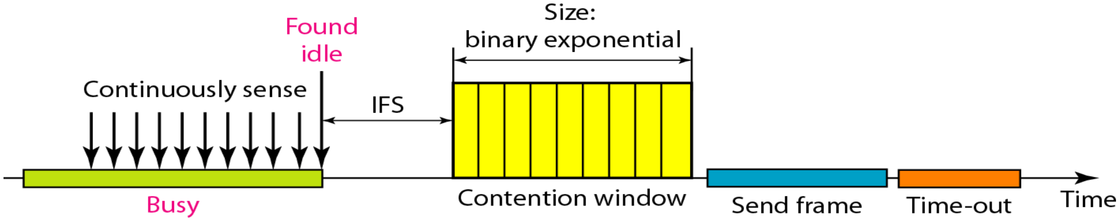
The first difference is the addition of the persistence process. We need to sense the channel before we start sending the frame by using one of the persistence processes

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

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.

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

We need to avoid collisions on wireless networks because they cannot be detected. Carrier sense multiple access with collision avoidance *(CSMAlCA)* was invented for wirelesss 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



Timing in CSMA/CA

Inter frame 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 inter frame 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. The IFS variable can also be used to prioritize stations or frame types. For example, a station that is assigned 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

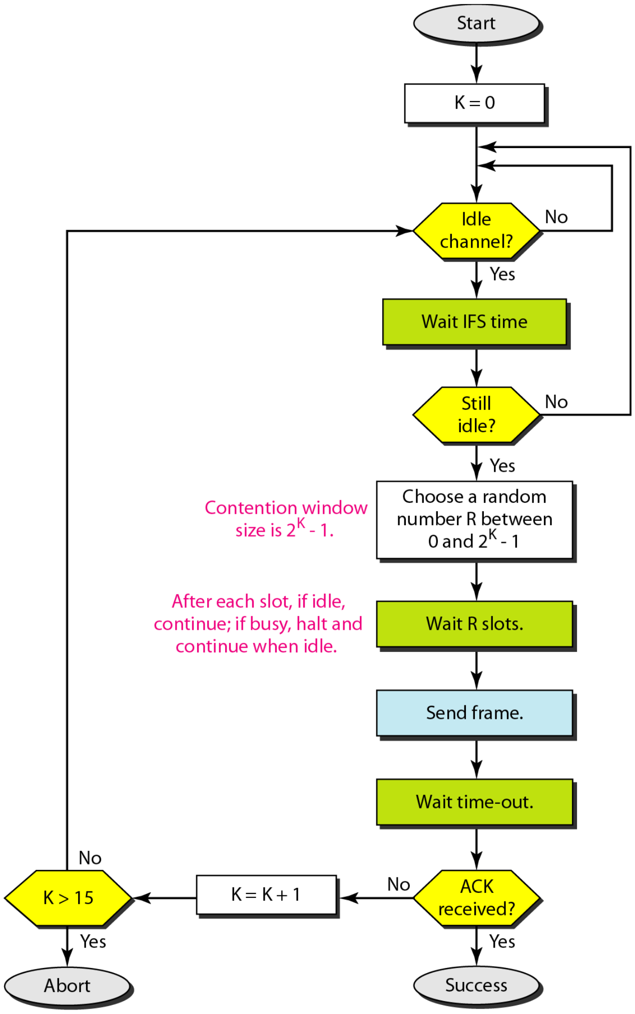
The contention window is an amount of time divided into slots. A station that is ready to send chooses a random number of slots as its wait time. The number of slots in the window changes according to the binary exponential back-off strategy. This means that it is set to one slot the first time and then 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.



Contention window size is $2^K - 1$.

After each slot, if idle, continue; if busy, halt and continue when idle.

This is the CSMA protocol with collision avoidance.

- The station ready to transmit, senses the line by using one of the persistent strategies.
- As soon as it finds the line to be idle, the station waits for an IFS (Inter frame space) amount of time.
- If then waits for some random time and sends the frame.
- After sending the frame, it sets a timer and waits for the acknowledgement from the receiver.
- If the acknowledgement is received before expiry of the timer, then the transmission is successful.
- But if the transmitting station does not receive the expected acknowledgement before the timer expiry then it increments the back off

parameter, waits for the back off time and re senses the line

Controlled Access Protocols

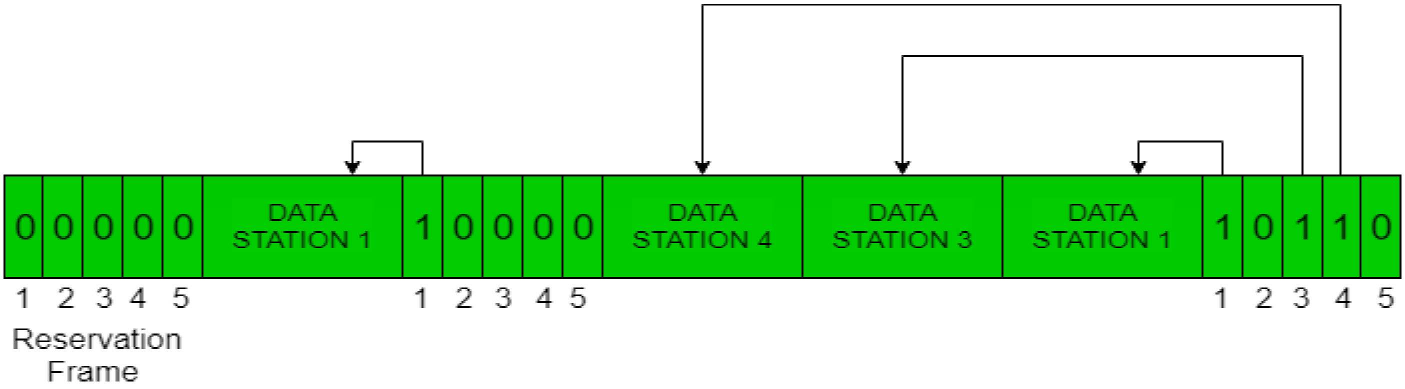
In controlled access, the stations seek information from one another to find which station has the right to send. It allows only one node to send at a time, to avoid collision of messages on shared medium.
The three controlled-access methods are:

1 Reservation 2 Polling 3 Token Passing

Reservation

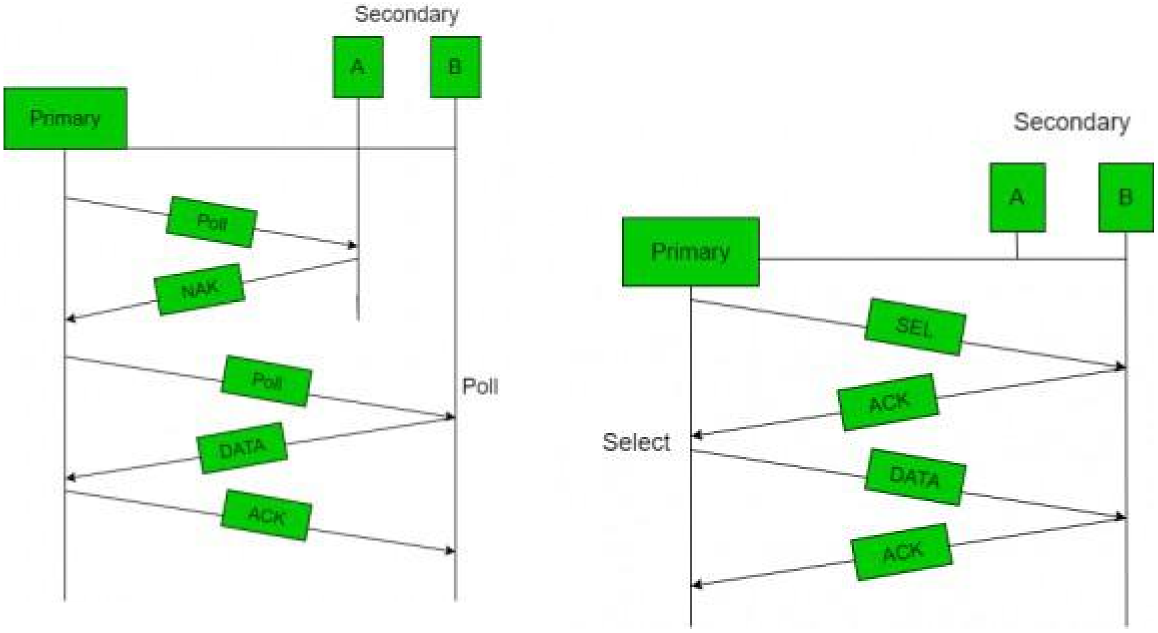
- In the reservation method, a station needs to make a reservation before sending data.
- The time line has two kinds of periods:
  1. Reservation interval of fixed time length
  2. Data transmission period of variable frames.
- If there are M stations, the reservation interval is divided into M slots, and each station has one slot.
- Suppose if station 1 has a frame to send, it transmits 1 bit during the slot 1. No other station is allowed to transmit during this slot.
- In general, i th station may announce that it has a frame to send by inserting a 1 bit into i th slot. After all N slots have been checked, each station knows which stations wish to transmit.
- The stations which have reserved their slots transfer their frames in that order.
- After data transmission period, next reservation interval begins.
- Since everyone agrees on who goes next, there will never be any collisions.

The following figure shows a situation with five stations and a five slot 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.



Polling

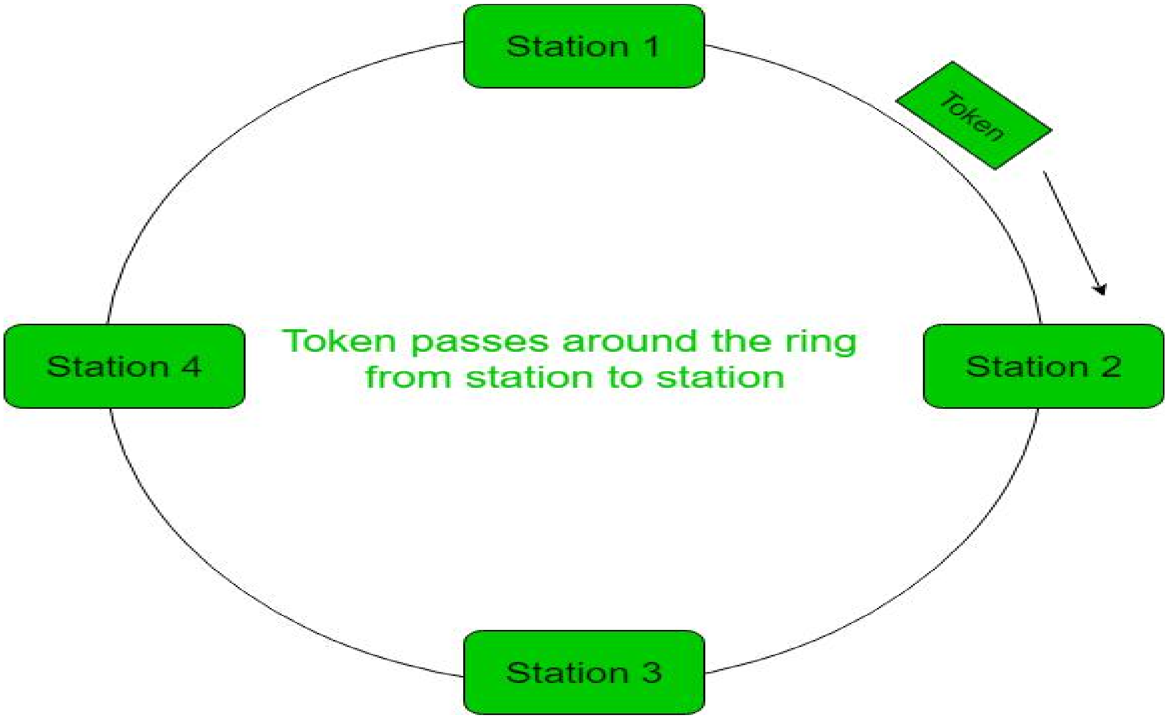
- Polling process is similar to the roll-call performed in class. Just like the teacher, a controller sends a message to each node in turn.
- In this, one acts as a primary station(controller) and the others are secondary stations. All data exchanges must be made through the controller.
- The message sent by the controller contains the address of the node being selected for granting access.
- Although all nodes receive the message but the addressed one responds to it and sends data, if any. If there is no data, usually a “poll reject”(NAK) message is sent back.
- Problems include high overhead of the polling messages and high dependence on the reliability of the controller.



Token Passing

- In token passing scheme, the stations are connected logically to each other in form of ring and access of stations is governed by tokens.
- A token is a special bit pattern or a small message, which circulate from one station to the next in the some predefined order.

- In Token ring, token is passed from one station to another adjacent station in the ring whereas incase of Token bus, each station uses the bus to send the token to the next station in some predefined order.
- In both cases, token represents permission to send. If a station has a frame queued for transmission when it receives the token, it can send that frame before it passes the token to the next station. If it has no queued frame, it passes the token simply.
- After sending a frame, each station must wait for all N stations (including itself) to send the token to their neighbors and the other N – 1 stations to send a frame, if they have one.
- There exists problems like duplication of token or token is lost or insertion of new station, removal of a station, which need be tackled for correct and reliable operation of this scheme.



Error Detection

Error

A condition when the receiver's information does not matches with the sender's information. During transmission, digital signals suffer from noise that can introduce errors in the binary bits travelling from sender to receiver. That means a 0 bit may change to 1 or a 1 bit may change to 0.

Error Detecting Codes (Implemented either at Data link layer or Transport Layer of OSI Model)

Whenever a message is transmitted, it may get scrambled by noise or data may get corrupted. To avoid this, we use error-detecting codes which are additional data added to a given digital message to help us detect if any error has occurred during transmission of the message.

Basic approach used for error detection is the use of redundancy bits, where additional bits are added to facilitate detection of errors. Some popular techniques for error detection are:

1. Simple Parity check

2. Two-dimensional Parity check

3. Checksum

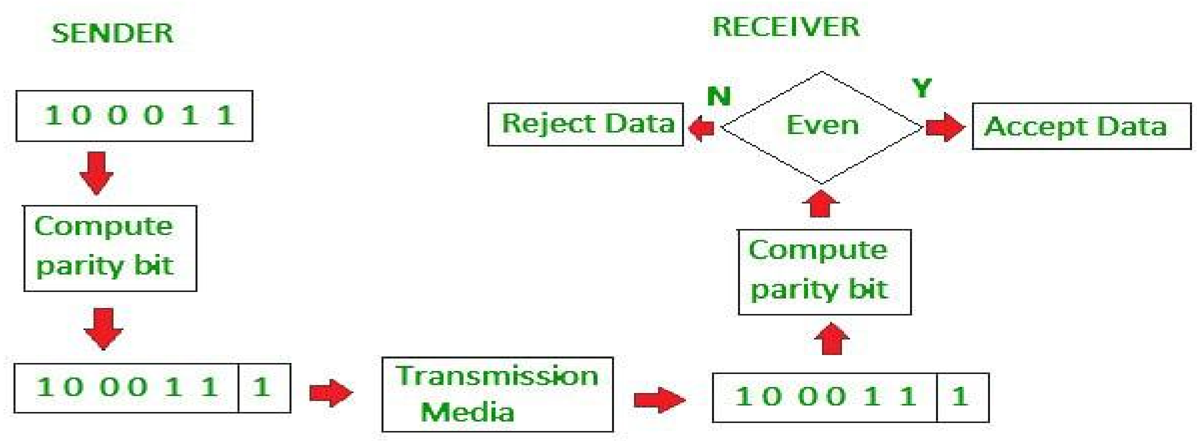
4. Cyclic redundancy check

Simple Parity check

Blocks of data from the source are subjected to a check bit or parity bit generator form, where a parity of : 1 is added to the block if it contains odd number of 1’s, and

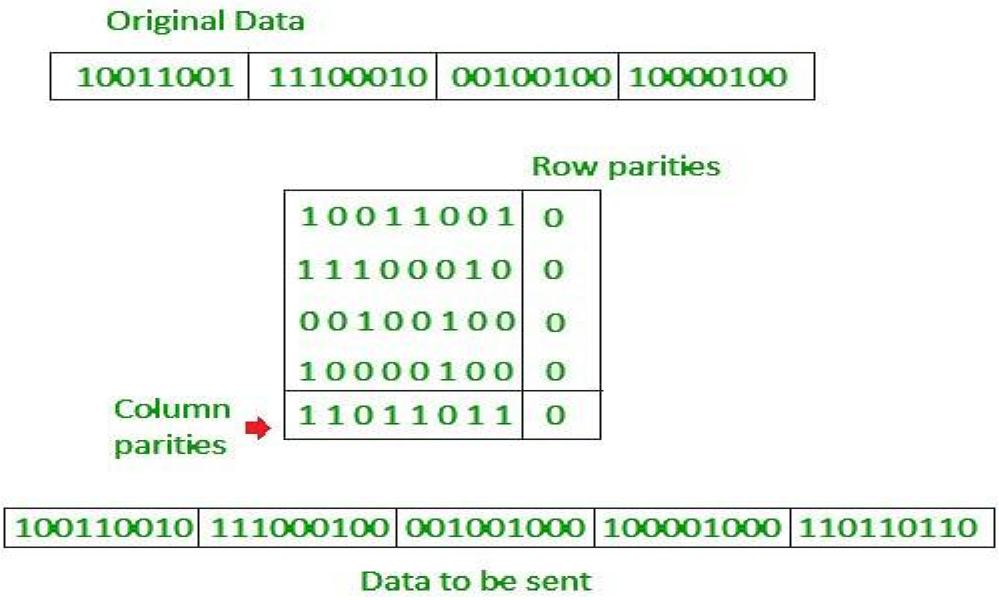
0 is added if it contains even number of 1’s

This scheme makes the total number of 1’s even, that is why it is called even parity checking.



Two-dimensional Parity check

Parity check bits are calculated for each row, which is equivalent to a simple parity check bit. Parity check bits are also calculated for all columns, then both are sent along with the data. At the receiving end these are compared with the parity bits calculated on the received data.


Original Data

10011001	11100010	00100100	10000100

Row parities

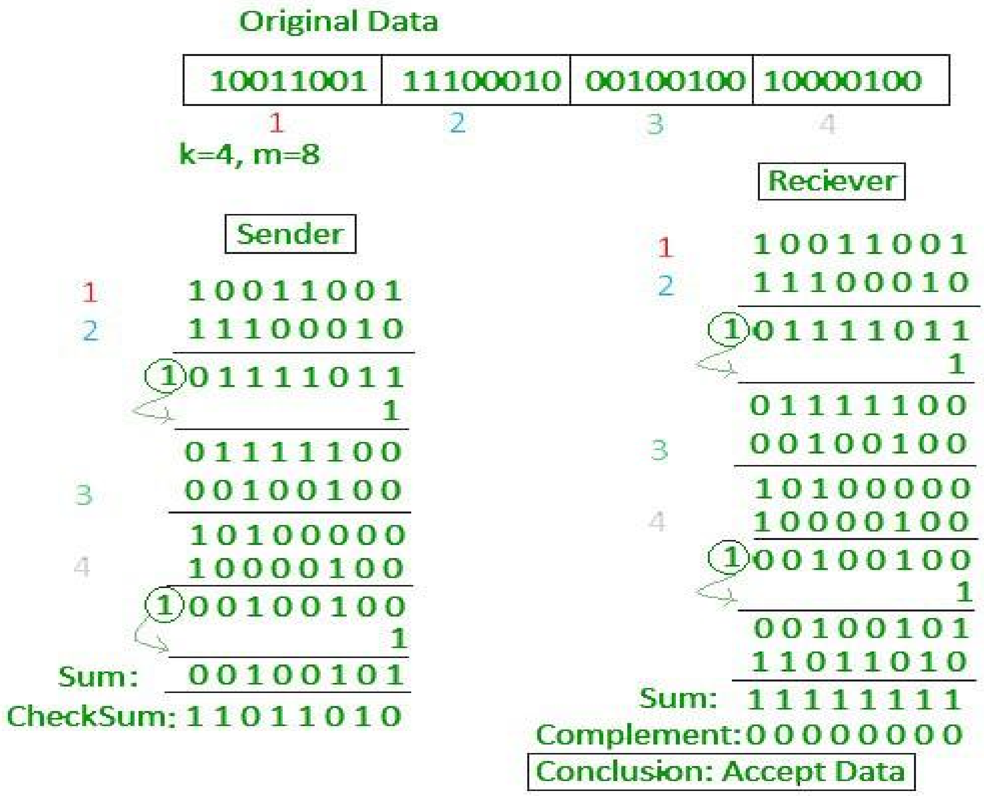
10011001	0
11100010	0
00100100	0
10000100	0
11011011	0

100110010	111000100	001001000	100001000	110110110

Data to be sent

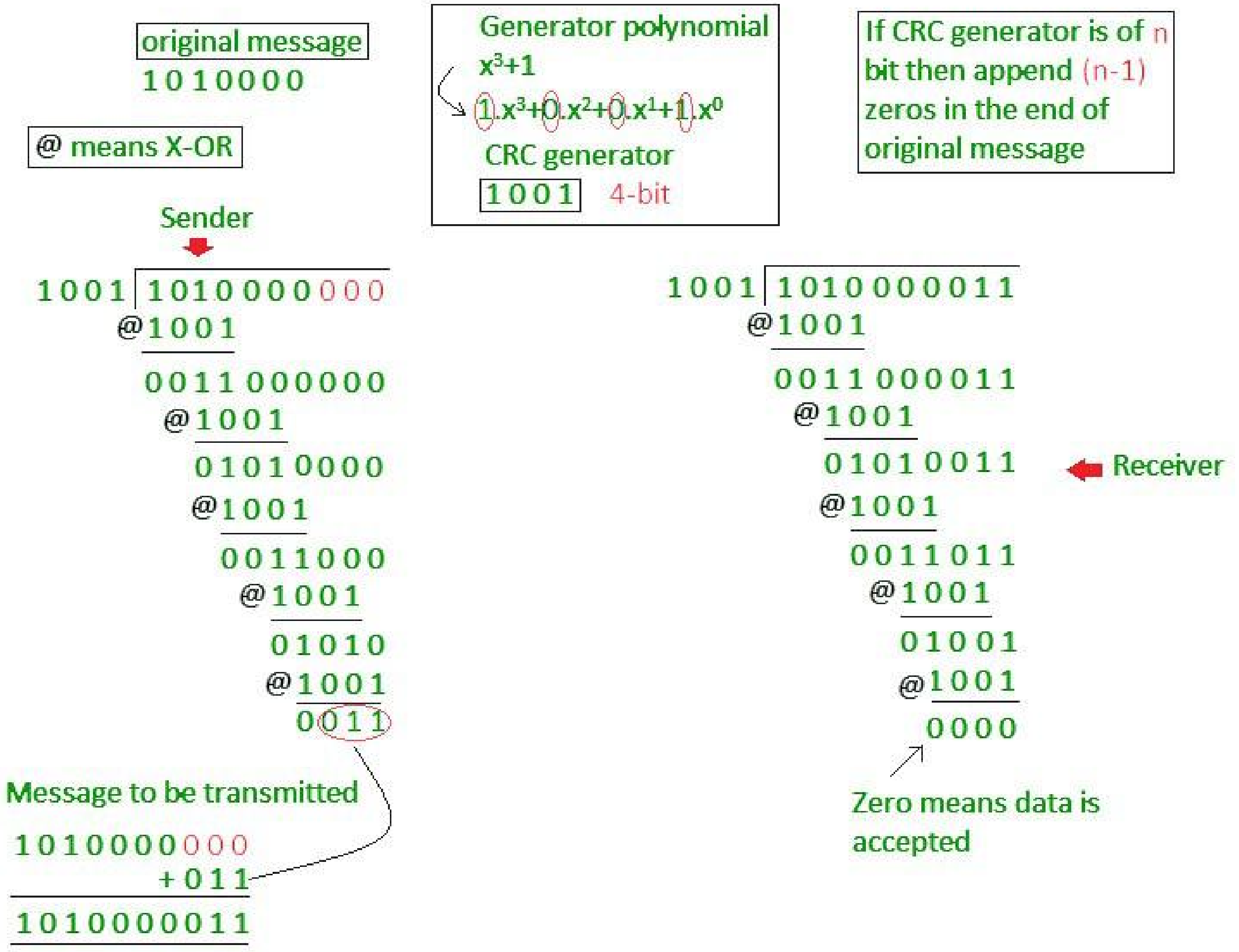
Checksum

- In checksum error detection scheme, the data is divided into k segments each of m bits.
- In the sender’s end the segments are added using 1’s complement arithmetic to get the sum. The sum is complemented to get the checksum.
- The checksum segment is sent along with the data segments.
- At the receiver’s end, all received segments are added using 1’s complement arithmetic to get the sum. The sum is complemented.
- If the result is zero, the received data is accepted; otherwise discarded.


Original Data

10011001	11100010	00100100	10000100
1	2	3	4

Cyclic redundancy check (CRC)



- Unlike checksum scheme, which is based on addition, CRC is based on binary division.
- In CRC, a sequence of redundant bits, called cyclic redundancy check bits, are appended to the end of data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number.
- At the destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is assumed to be correct and is therefore accepted.
- A remainder indicates that the data unit has been damaged in transit and therefore must be rejected.

Error Correction

Error Correction codes are used to detect and correct the errors when data is transmitted from the sender to the receiver.

Error Correction can be handled in two ways:
Backward error correction: Once the error is discovered, the receiver requests the sender to retransmit the entire data unit.
Forward error correction: In this case, the receiver uses the error-correcting code which automatically corrects the errors.
A single additional bit can detect the error, but cannot correct it.

For correcting the errors, one has to know the exact position of the error. For example, If we want to calculate a single-bit error, the error correction code will determine which one of seven bits is in error. To achieve this, we have to add some additional redundant bits.

Suppose r is the number of redundant bits and d is the total number of the data bits. The number of redundant bits r can be calculated by using the formula:
$2^r >= d+r+1$
The value of r is calculated by using the above formula. For example, if the value of d is 4, then the possible smallest value that satisfies the above relation would be 3.

To determine the position of the bit which is in error, a technique developed by R.W Hamming is Hamming code which can be applied to any length of the data unit and uses the relationship between data units and redundant units.

Hamming Code

Parity bits: The bit which is appended to the original data of binary bits so that the total number of 1s is even or odd.

Even parity: To check for even parity, if the total number of 1s is even, then the value of the parity bit is 0. If the total number of 1s occurrences is odd, then the value of the parity bit is 1.

Odd Parity: To check for odd parity, if the total number of 1s is even, then the value of parity bit is 1. If the total number of 1s is odd, then the value of parity bit is 0.

Algorithm of Hamming code:

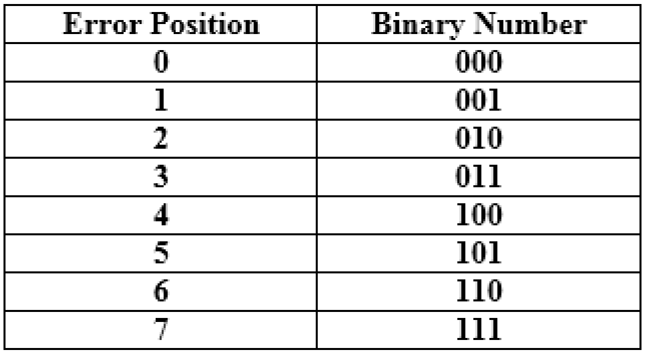
An information of 'd' bits are added to the redundant bits 'r' to form d+r.

The location of each of the (d+r) digits is assigned a decimal value.

The 'r' bits are placed in the positions 1,2,.....2^{k-1}

At the receiving end, the parity bits are recalculated. The decimal value of the parity bits determines the position of an error.

Relationship b/w Error position & binary number.



Error Position	Binary Number
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Let's understand the concept of Hamming code through an example:

Suppose the original data is 1010 which is to be sent.

Total number of data bits 'd' = 4

Number of redundant bits r : $2^r >= d+r+1$

$2^r >= 4+r+1$

Therefore, the value of r is 3 that satisfies the above relation.
Total number of bits = d+r = 4+3 = 7;

Determining the position of the redundant bits

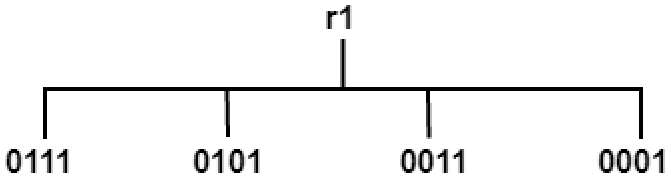
The number of redundant bits is 3. The three bits are represented by r1, r2, r4. The position of the redundant bits is calculated with corresponds to the raised power of 2. Therefore, their corresponding positions are 1, 2^1, 2^2.
The position of r1 = 1, The position of r2 = 2 , The position of r4 = 4

Representation of Data on the addition of parity bits:

7	6	5	4	3	2	1
1	0	1	r4	0	r2	r1

Determining the Parity bits

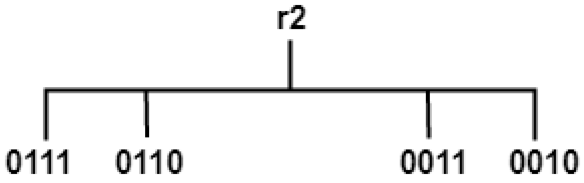
Determining the r1 bit: The r1 bit is calculated by performing a parity check on the bit positions whose binary representation includes 1 in the first position.



7	6	5	4	3	2	1
1	0	1	r4	0	r2	r1

We observe from the above figure that the bit position that includes 1 in the first position are 1, 3, 5, 7. Now, we perform the even-parity check at these bit positions. The total number of 1 at these bit positions corresponding to r1 is even, therefore, the value of the r1 bit is 0.

Determining r2 bit: The r2 bit is calculated by performing a parity check on the bit positions whose binary representation includes 1 in the second position

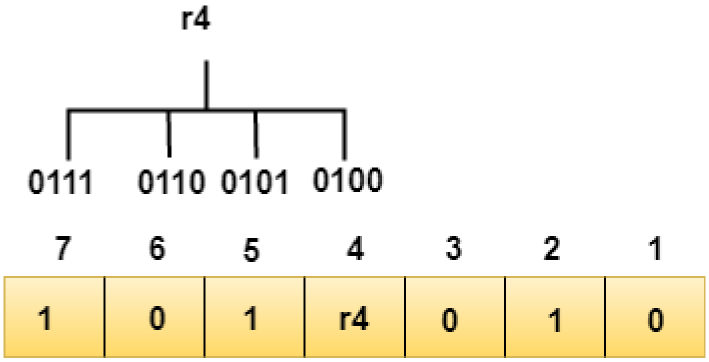


7	6	5	4	3	2	1
1	0	1	r4	0	r2	0

.

We observe from the above figure that the bit positions that includes 1 in the second position are 2, 3, 6, 7. Now, we perform the even-parity check at these bit positions. The total number of 1 at these bit positions corresponding to r2 is odd, therefore, the value of the r2 bit is 1.

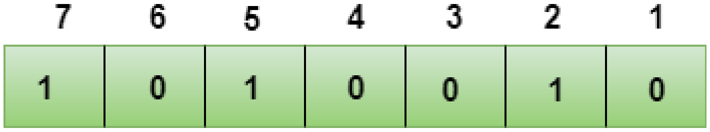
Determining r4 bit: The r4 bit is calculated by performing a parity check on the bit positions whose binary representation includes 1 in the third position.



7	6	5	4	3	2	1
1	0	1	r4	0	1	0

We observe from the above figure that the bit positions that includes 1 in the third position are 4, 5, 6, 7. Now, we perform the even-parity check at these bit positions. The total number of 1 at these bit positions corresponding to r4 is even, therefore, the value of the r4 bit is 0.

Data transferred is given below:

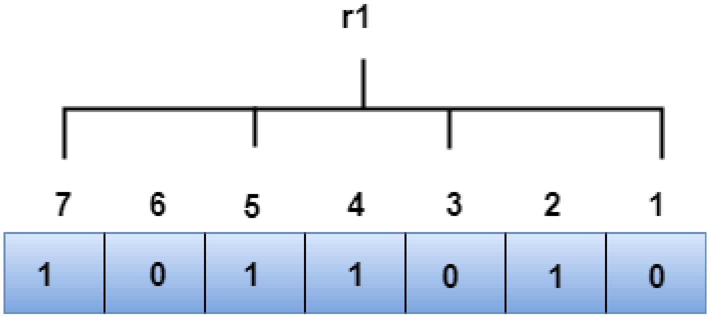


7	6	5	4	3	2	1
1	0	1	0	0	1	0

Suppose the 4th bit is changed from 0 to 1 at the receiving end, then parity bits are recalculated.

R1 bit

The bit positions of the r1 bit are 1,3,5,7

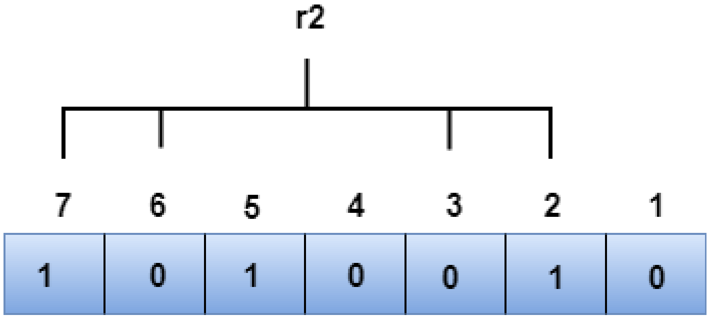


7	6	5	4	3	2	1
1	0	1	1	0	1	0

We observe from the above figure that the binary representation of r1 is 1100. Now, we perform the even-parity check, the total number of 1s appearing in the r1 bit is an even number. Therefore, the value of r1 is 0.

R2 bit

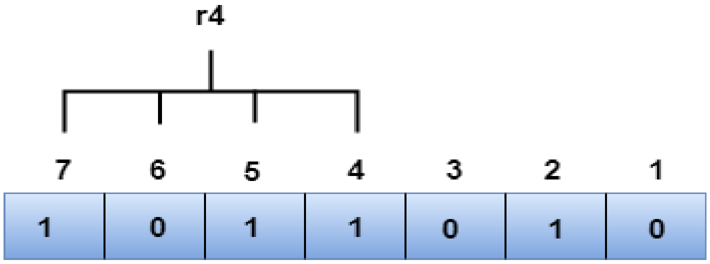
The bit positions of r2 bit are 2,3,6,7.



We observe from the above figure that the binary representation of r2 is 1001. Now, we perform the even-parity check, the total number of 1s appearing in the r2 bit is an even number. Therefore, the value of r2 is 0.

R4 bit

The bit positions of r4 bit are 4,5,6,7.



We observe from the above figure that the binary representation of r4 is 1011. Now, we perform the even-parity check, the total number of 1s appearing in the r4 bit is an odd number. Therefore, the value of r4 is 1.

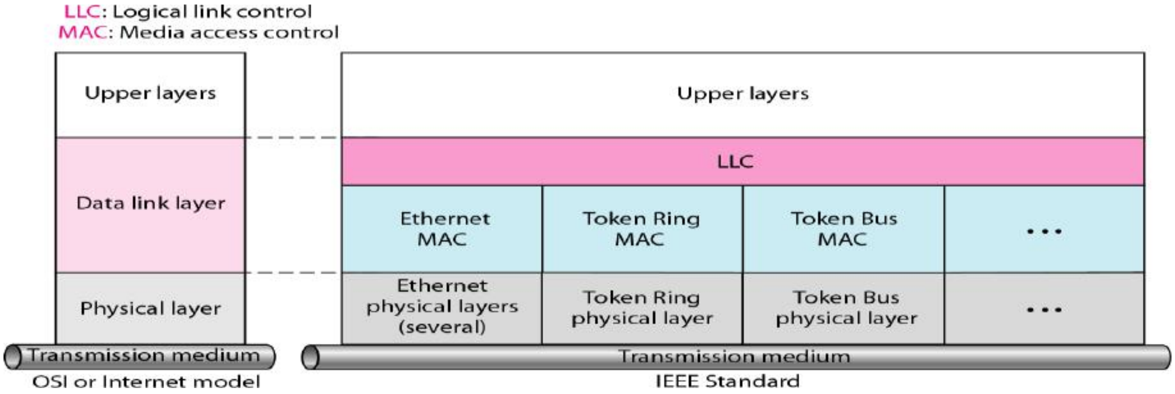
The binary representation of redundant bits, i.e., r4r2r1 is 100, and its corresponding decimal value is 4. Therefore, the error occurs in a 4th bit position. The bit value must be changed from 1 to 0 to correct the error.

Wired LANs: Ethernet

In 1985, the Computer Society of the IEEE started a project, called Project 802, to set standards to enable intercommunication among equipment from a variety of manufacturers. Project 802 is a way of specifying functions of the physical layer and the data link layer of major LAN protocols.

The relationship of the 802 Standard to the traditional OSI model is shown in below Figure. The IEEE has subdivided the data link layer into two sub layers: logical link control (LLC) and media access control).

IEEE has also created several physical layer standards for different LAN protocols



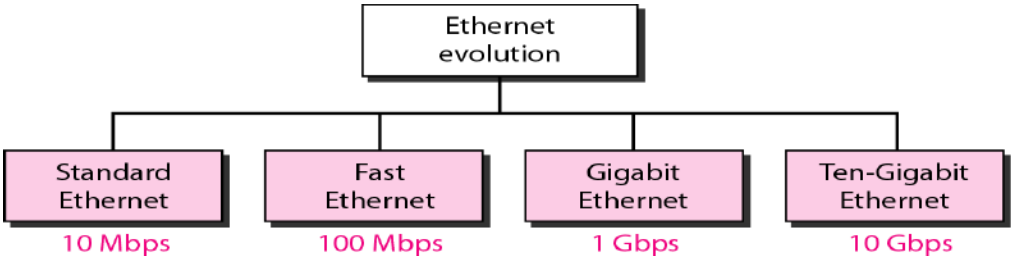
IEEE standard for LANs

STANDARD ETHERNET

The original Ethernet was created in 1976 at Xerox's Palo Alto Research Center (PARC). Since then, it has gone through four generations.

Standard Ethernet (l0 Mbps), Fast Ethernet (100 Mbps), Gigabit Ethernet (l Gbps), and Ten-Gigabit Ethernet (l0 Gbps),

We briefly discuss the Standard (or traditional) Ethernet in this section



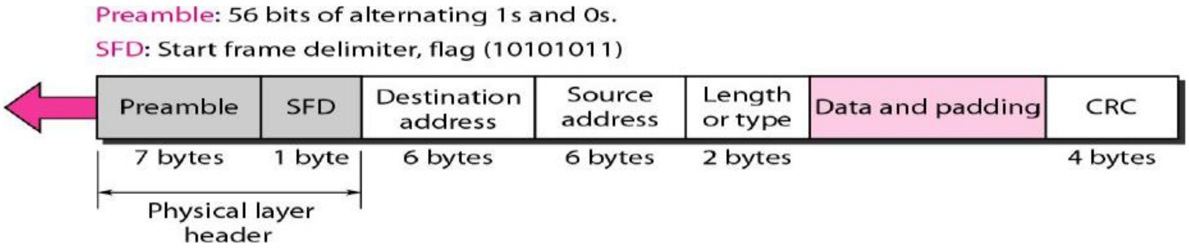
Ethernet evolution through four generations

MAC Sublayer

In Standard Ethernet, the MAC sublayer governs the operation of the access method. It also frames data received from the upper layer and passes them to the physical layer.

Frame Format

The Ethernet frame contains seven fields: preamble, SFD, DA, SA, length or type of protocol data unit (PDU), upper-layer data, and the CRC. Ethernet does not provide any mechanism for acknowledging received frames, making it what is known as an unreliable medium. Acknowledgments must be implemented at the higher layers. The format of the MAC frame is shown in below figure



802.3 MAC frame

Preamble. The first field of the 802.3 frame contains 7 bytes (56 bits) of alternating 0s and 1s that alerts the receiving system to the coming frame and enables it to synchronize its input timing. The pattern provides only an alert and a timing pulse. The 56-bit pattern allows the stations to miss some bits at the beginning of the frame. The preamble is actually added at the physical layer and is not (formally) part of the frame.

Start frame delimiter (SFD). The second field (l byte: 10101011) signals the beginning of the frame. The SFD warns the station or stations that this is the last chance for synchronization. The last 2 bits is 11 and alerts the receiver that the next field is the destination address.

Destination address (DA). The DA field is 6 bytes and contains the physical address of the destination station or stations to receive the packet.

Source address (SA). The SA field is also 6 bytes and contains the physical address of the sender of the packet.

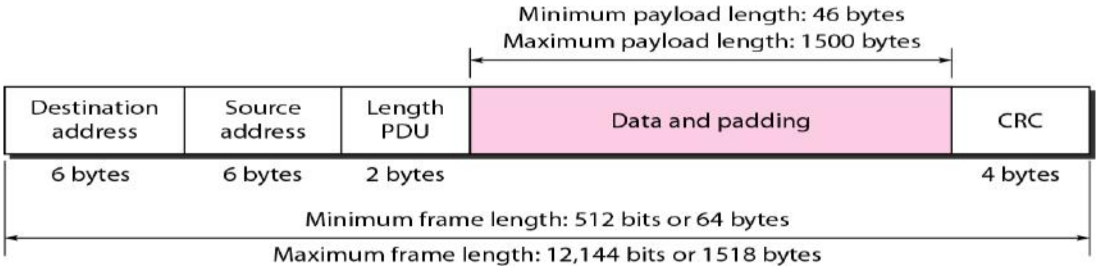
Length or type. This field is defined as a type field or length field. The original Ethernet used this field as the type field to define the upper-layer protocol using the MAC frame. The IEEE standard used it as the length field to define the number of bytes in the data field. Both uses are common today.

Data. This field carries data encapsulated from the upper-layer protocols. It is a minimum of 46 and a maximum of 1500 bytes.

CRC. The last field contains error detection information, in this case a CRC-32

Frame Length

Ethernet has imposed restrictions on both the minimum and maximum lengths of a frame, as shown in below Figure


Minimum payload length: 46 bytes
Maximum payload length: 1500 bytes
Destination address
Source address
Length PDU
Data and padding
CRC
6 bytes
6 bytes
2 bytes
4 bytes
Minimum frame length: 512 bits or 64 bytes
Maximum frame length: 12,144 bits or 1518 bytes

Minimum and maximum lengths

An Ethernet frame needs to have a minimum length of 512 bits or 64 bytes. Part of this length is the header and the trailer. If we count 18 bytes of header and trailer (6 bytes of source address, 6 bytes of destination address, 2 bytes of length or type, and 4 bytes of CRC), then the minimum length of data from the upper layer is 64 - 18 = 46 bytes. If the upper-layer packet is less than 46 bytes, padding is added to make up the difference

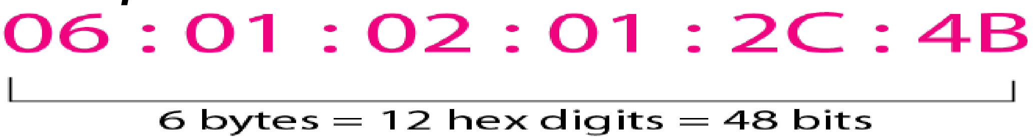
The standard defines the maximum length of a frame (without preamble and SFD field) as 1518 bytes. If we subtract the 18 bytes of header and trailer,

the maximum length of the payload is 1500 bytes.
The maximum length restriction has two historical reasons.
First, memory was very expensive when Ethernet was designed: a maximum length restriction helped to reduce the size of the buffer.
Second, the maximum length restriction prevents one station from monopolizing the shared medium, blocking other stations that have data to send.

Addressing

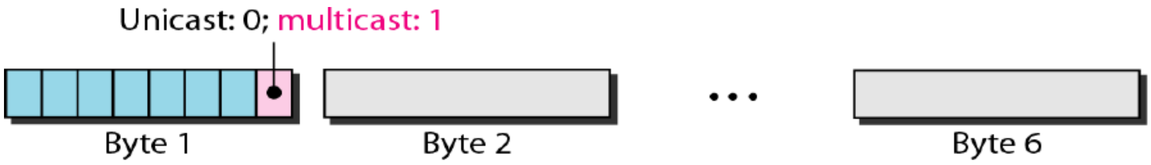
The Ethernet address is 6 bytes (48 bits), normally written in hexadecimal notation, with a colon between the bytes.

Example of an Ethernet address in hexadecimal notation


06 : 01 : 02 : 01 : 2C : 4B
6 bytes = 12 hex digits = 48 bits

Unicast, Multicast, and Broadcast Addresses A source address is always a unicast address-the frame comes from only one station. The destination address, however, can be **unicast, multicast, or broadcast**. Below Figure shows how to distinguish a unicast address from a multicast address.

If the least significant bit of the first byte in a destination address is 0, the address is unicast; otherwise, it is multicast.



Unicast and multicast addresses

A unicast destination address defines only one recipient; the relationship between the sender and the receiver is one-to-one.
A multicast destination address defines a group of addresses; the relationship between the sender and the receivers is one-to-many.
The broadcast address is a special case of the multicast address; the recipients are all the stations on the LAN. A broadcast destination address is forty-eight 1s.
Access Method: CSMA/CD
Standard Ethernet uses I-persistent CSMA/CD
Slot Time In an Ethernet network.
Slot time =round-trip time + time required to send the jam sequence

The slot time in Ethernet is defined in bits. It is the time required for a station to send 512 bits. This means that the actual slot time depends on the data rate; for traditional 10-Mbps Ethernet it is 51.2 micro sec.

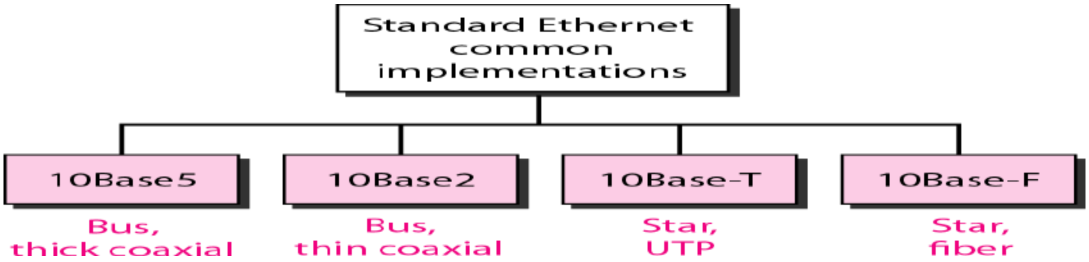
Slot Time and Maximum Network Length There is a relationship between the slot time and the maximum length of the network (collision domain). It is dependent on the propagation speed of the signal in the particular medium.

In most transmission media, the signal propagates at 2 x 10^8 m/s (two-thirds of the rate for propagation in air).
For traditional Ethernet, we calculate
MaxLength =PropagationSpeedx (SlotTime/2)
MaxLength= (2 x 10^8) X(51.2 X10^{-6})/2= 5120m

Of course, we need to consider the delay times in repeaters and interfaces, and the time required to send the jam sequence. These reduce the maximum-length of a traditional Ethernet network to 2500 m, just 48 percent of the theoretical calculation. MaxLength=2500 m

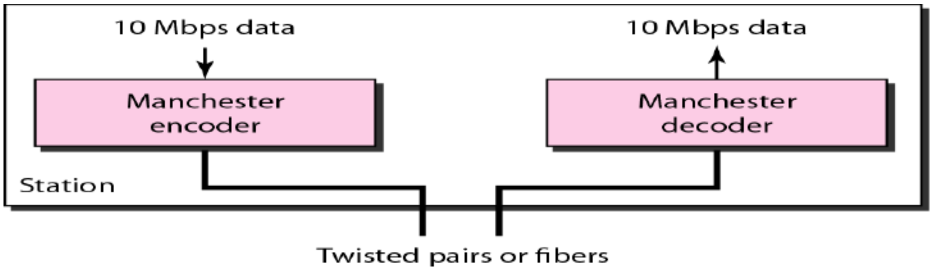
Physical Layer

The Standard Ethernet defines several physical layer implementations; four of the most common, are shown in Figure

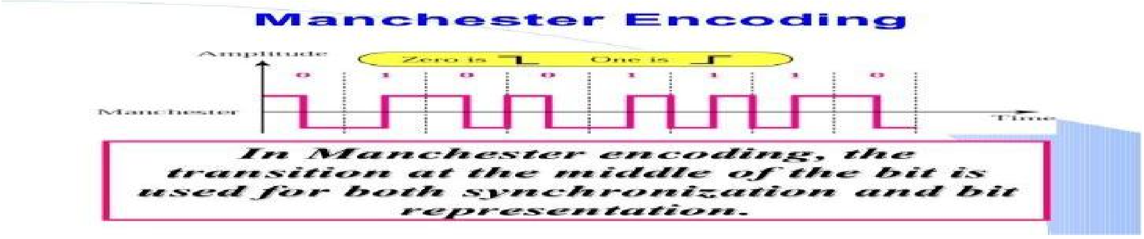


Encoding and Decoding

All standard implementations use digital signaling (baseband) at 10 Mbps. At the sender, data are converted to a digital signal using the Manchester scheme; at the receiver, the received signal is interpreted as Manchester and decoded into data. Manchester encoding is self-synchronous, providing a transition at each bit interval. Figure shows the encoding scheme for Standard Ethernet



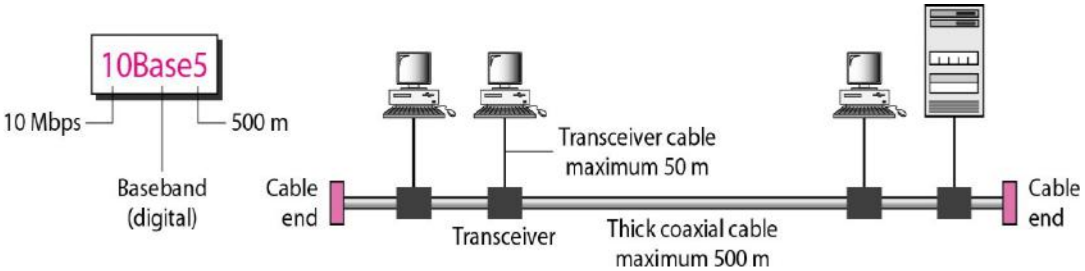
In Manchester encoding, the transition at the middle of the bit is used for synchronization



In Manchester encoding, the transition at the middle of the bit is used for both synchronization and bit representation.

lOBase5: Thick Ethernet

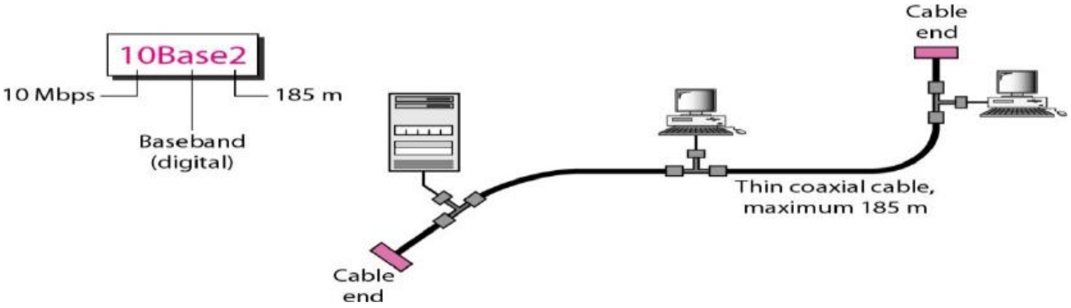
The first implementation is called **10Base5, thick Ethernet, or Thicknet.** lOBase5 was the first Ethernet specification to use a bus topology with an external **transceiver** (transmitter/receiver) connected via a tap to a thick coaxial cable. Figure shows a schematic diagram of a lOBase5 implementation



10Base5 implementation

10Base2: Thin Ethernet

The second implementation is called 10 Base2, **thin** Ethernet, or Cheapernet. 10Base2 also uses a bus topology, but the cable is much thinner and more flexible. Figure shows the schematic diagram of a 10Base2 implementation.



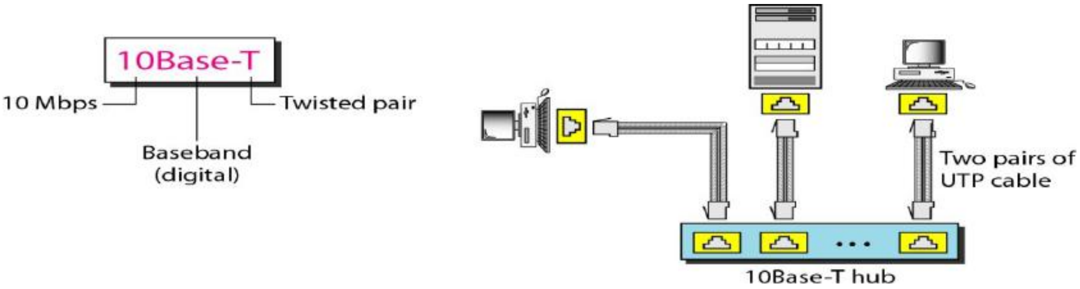
10Base2 implementation

thin coaxial cable is less expensive than thick coaxial.
Installation is simpler because the thin coaxial cable is very flexible.
However, the length of each segment cannot exceed *185* m (close to 200 m) due to the high level of attenuation in thin coaxial cable.

1OBase-T: Twisted-Pair Ethernet

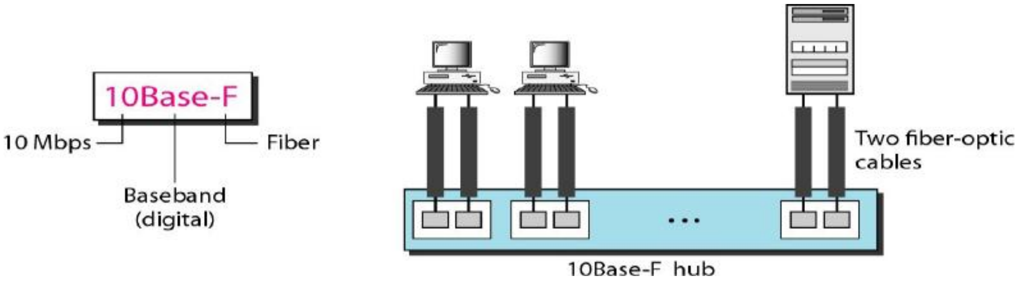
The third implementation is called 10Base-T or twisted-pair Ethernet. It uses a physical star topology. The stations are connected to a hub via two pairs of twisted cable, as shown in Figure

The maximum length of the twisted cable here is defined as 100 m, to minimize the effect of attenuation in the twisted cable



10Base-T implementation

Although there are several types of optical fiber 10-Mbps Ethernet, the most common is called 10Base-F.10Base-F uses a star topology to connect stations to a hub. The stations are connected to the hub using two fiber-optic cables, as shown in Figure



10Base-F implementation