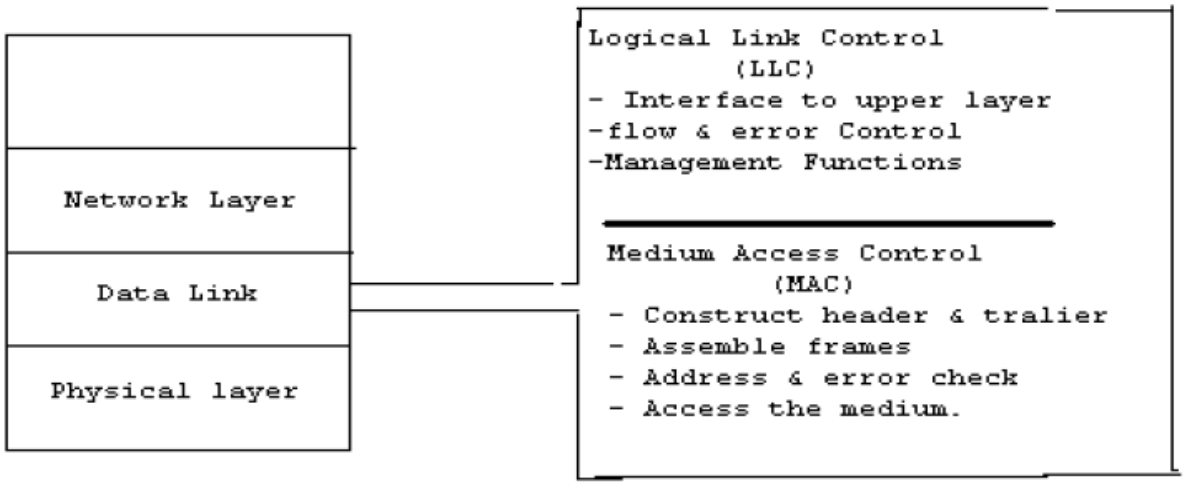
Chapter 4

Data Link layers

The architecture of a LAN can be considered as a set of layered protocols.
In OSI terms, the higher layer protocols are totally independent of the LAN architecture. Hence, only lower order layers are considered for the design of LAN architecture.
The datalink layer of LAN is split into two sub layers.

I. Medium Access Control (MAC),
II. Logical Link Control Layer (LLC)



LLC Frame Format

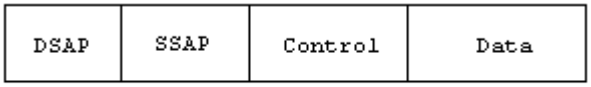


Fig :- LLC Frame Format

Destination Service Access Point (DSAP): IEEE 802.2 header begins with a 1 byte field, which identifies the receiving upper-layer process.

Source Service Access Point (SSAP): Following the DSAP address is the 1-byte address, which identifies the sending upper-layer process.

Control: The Control field employs three different formats, depending on the type of LLC frame used.

- Information (I) frame -- Carries upper-layer information and some control information.
- Supervisory (S) frame -- Provides control information. An S frame can request and suspend transmission, reports on status, and acknowledge receipt of I frames. S frames do not have an Information field.
- Unnumbered (U) frame -- Used for control purposes and is not sequenced. A U frame can be used to initialize secondaries. Depending on the function of the U frame, its Control field is 1 or 2 bytes. Some U frames have an Information field.

Data: Variable-length field bounded by the MAC format implemented. Usually contains IEEE 802.2 Subnetwork Access Protocol (SNAP) header information, as well as application-specific data.

MAC Frame Format

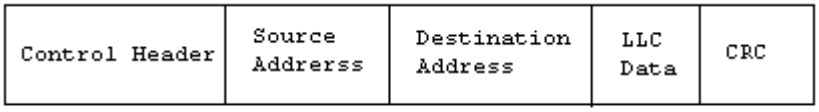


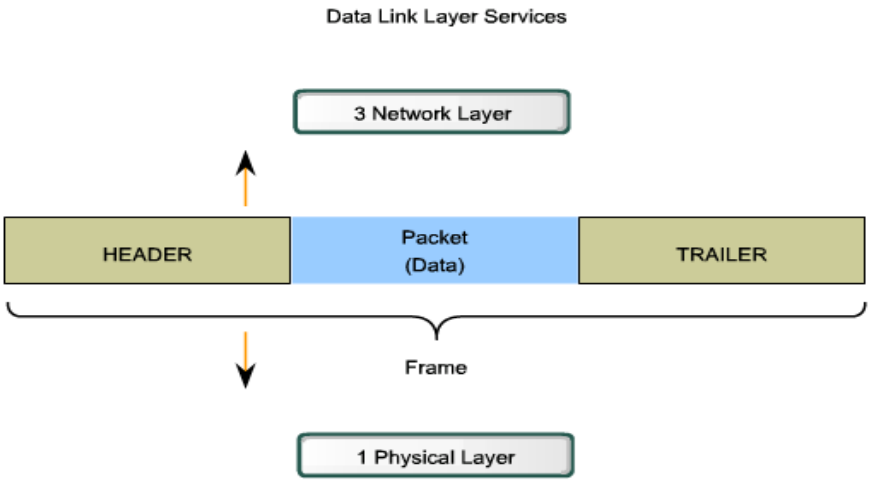
Fig : General MAC frame Format

Framing:

The data link layer needs to pack bits into frames, so that each frame is distinguishable from another. The Data Link layer prepares a packet for transport across the local media by encapsulating it with a header and a trailer to create a frame.

The Data Link layer frame includes:

- **Data** - The packet from the Network layer
- **Header** - Contains control information, such as addressing, and is located at the beginning of the PDU
- **Trailer** - Contains control information added to the end of the PDU



Our postal system practices a type of framing. The simple act of inserting a letter into an envelope separates one piece of information from another; the envelope serves as the delimiter. In addition, each envelope defines the sender and receiver addresses since the postal system is a many-to-many carrier facility. Framing in the data link layer separates a message from one source to a destination, or from other messages to other destinations, by adding a sender address and a destination address. The destination address defines where the packet is to go; the sender address helps the recipient acknowledge the receipt.

Fixed-Size Framing

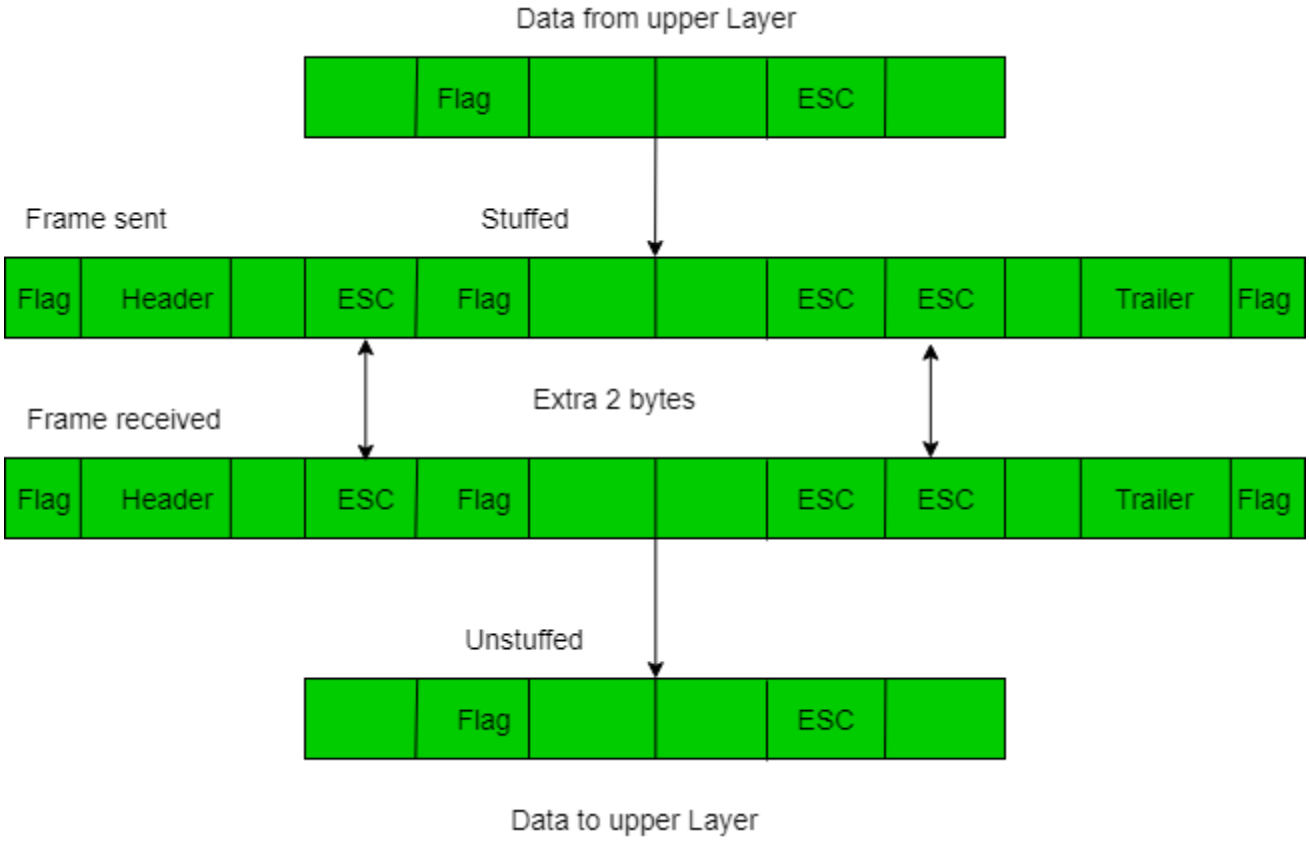
Frames can be of fixed or variable size. In fixed-size framing, there is no need for defining the boundaries of the frames; the size itself can be used as a delimiter. An example of this type of framing is the ATM wide-area network, which uses frames of fixed size called cells.

Variable-Size Framing

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

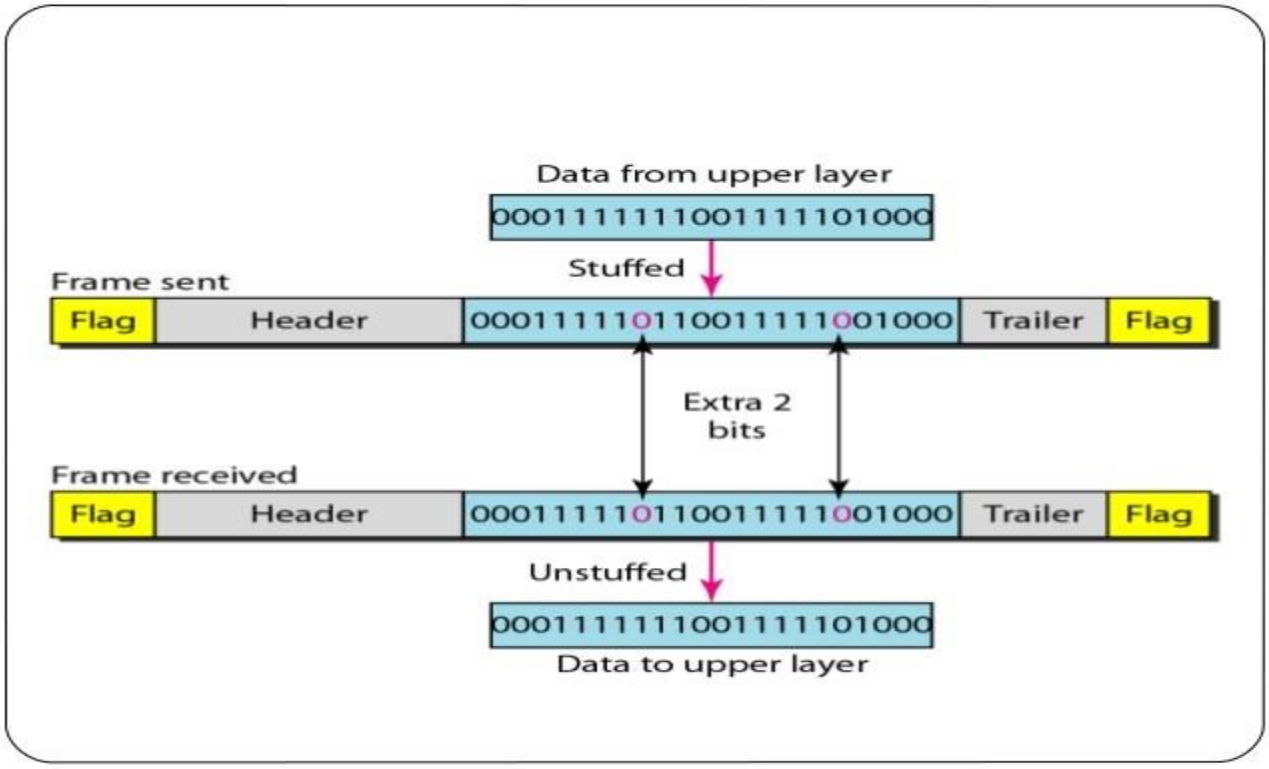
Character-Oriented Protocols

In a character-oriented protocol, data to be carried are 8-bit characters from a coding system such as ASCII. The header, which normally carries the source and destination addresses and other control information, and the trailer, which carries error detection or error correction redundant bits, are also multiples of 8 bits. To separate one frame from the next, an 8-bit (I-byte) flag is added at the beginning and the end of a frame. The flag, composed of protocol-dependent special characters, signals the start or end of a frame. Any pattern used for the flag could also be part of the information. To fix this problem, a byte-stuffing strategy was added to character-oriented framing. In byte stuffing (or character stuffing), a special byte is added to the data section of the frame when there is a character with the same pattern as the flag. The data section is stuffed with an extra byte. This byte is usually called the escape character (ESC), which has a predefined bit pattern. Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not a delimiting flag. Character-oriented protocols present a problem in data communications. The universal coding systems in use today, such as Unicode, have 16-bit and 32-bit characters that conflict with 8-bit characters. We can say that in general, the tendency is moving toward the bit-oriented protocols that we discuss next.



Bit-Oriented Protocols

In a bit-oriented protocol, the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, and so on. However, in addition to headers (and possible trailers), we still need a delimiter to separate one frame from the other. Most protocols use a special 8-bit pattern flag 01111110 as the delimiter to define the beginning and the end of the frame.



This flag can create the same type of problem we saw in the byte-oriented protocols. That is, if the flag pattern appears in the data, we need to somehow inform the receiver that this is not the end of the frame. We do this by stuffing 1 single bit (instead of 1 byte) to prevent the pattern from looking like a flag. The strategy is called bit stuffing. In bit stuffing, if a 0 and five consecutive 1 bits are encountered, an extra 0 is added. This extra stuffed bit is eventually removed from the data by the receiver. Note that the extra bit is added after one 0 followed by five 1s regardless of the value of the next bit. This guarantees that the flag field sequence does not inadvertently appear in the frame. This means that if the flag like pattern 01111110 appears in the data, it will change to 011111010 (stuffed) and is not mistaken as a flag by the receiver. The real flag 01111110 is not stuffed by the sender and is recognized by the receiver.

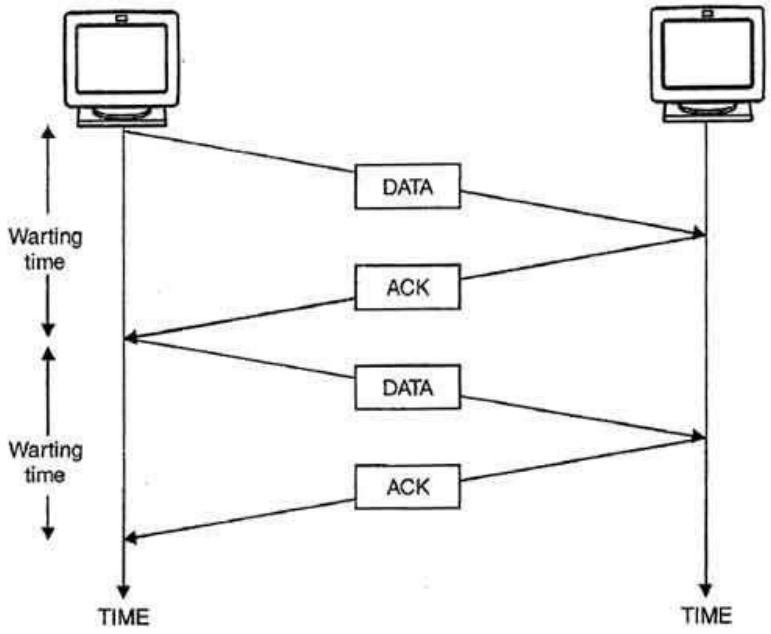
Flow Control

Flow Control is a set of procedures that tells the sender how much data it can transmit before it must wait for an acknowledgment from the receiver. The flow of data should not be allowed to overwhelm the receiver. Receiver should also be able to inform the transmitter before its limits (this limit may be amount of memory used to store the incoming data or the processing power at the receiver end) are reached and the sender must send fewer frames. Hence, Flow control refers to the set of procedures used to restrict the amount of data the transmitter can send before waiting for acknowledgment.

There are two methods developed for flow control namely Stop-and-wait and Sliding-window.

Stop & Wait Protocol

- In this method of flow control, the sender sends a single frame to receiver & waits for an acknowledgment.
- The next frame is sent by sender only when acknowledgment of previous frame is received.
- This process of sending a frame & waiting for an acknowledgment continues as long as the sender has data to send.
- To end up the transmission sender transmits end of transmission (EOT) frame.
- The main advantage of stop & wait protocols is its accuracy. Next frame is transmitted only when the first frame is acknowledged. So there is no chance of frame being lost.
- The main disadvantage of this method is that it is inefficient. It makes the transmission process slow. In this method single frame travels from source to destination and single acknowledgment travels from destination to source. As a result each frame sent and received uses the entire time needed to traverse the link. Moreover, if two devices are distance apart, a lot of time is wasted waiting for ACKs that leads to increase in total transmission time.



Stop & Wait Method.

Sliding Window Protocol

- In sliding window method, multiple frames are sent by sender at a time before needing an acknowledgment.

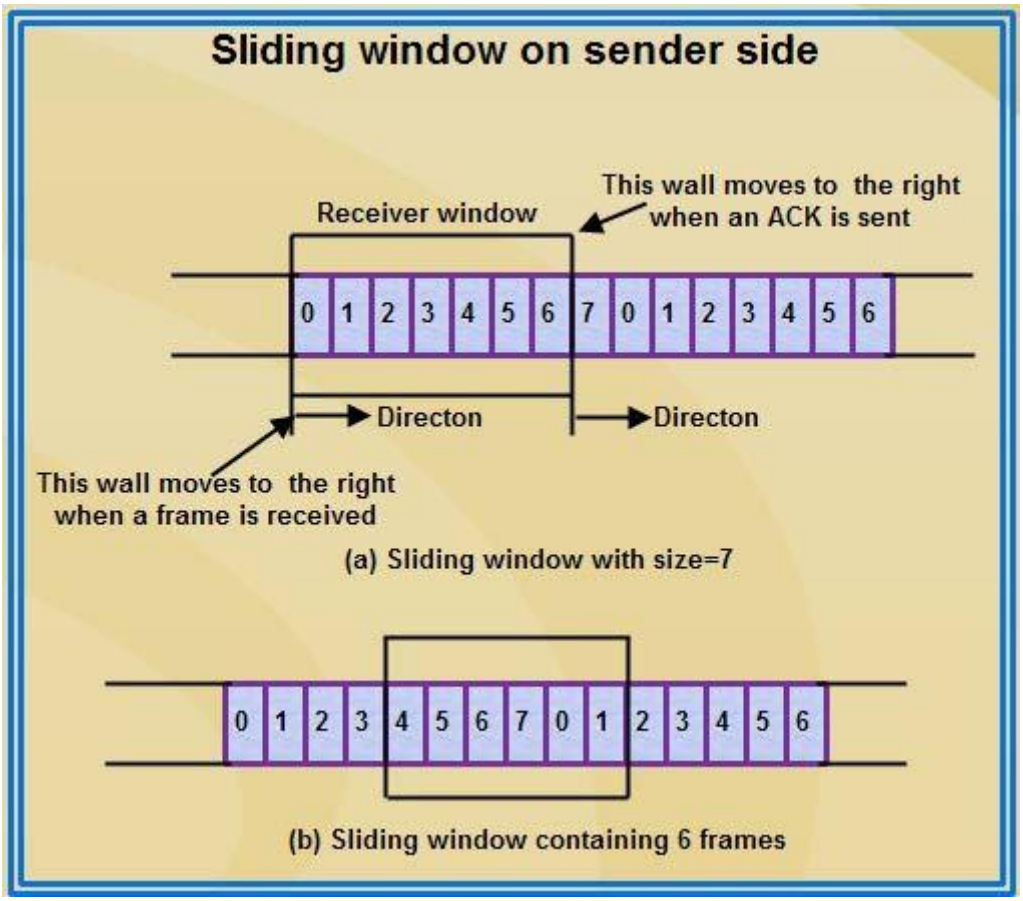
- Multiple frames sent by source are acknowledged by receiver using a single ACK frame.

Sliding Window

- Sliding window refers to an imaginary boxes that hold the frames on both sender and receiver side.
- It provides the upper limit on the number of frames that can be transmitted before requiring an acknowledgment.
- Frames may be acknowledged by receiver at any point even when window is not full on receiver side.
- Frames may be transmitted by source even when window is not yet full on sender side.
- The windows have a specific size in which the frames are numbered modulo- n, which means they are numbered from 0 to n-l. For e.g. if n = 8, the frames are numbered 0, 1,2,3,4,5,6, 7, 0, 1,2,3,4,5,6, 7, 0, 1, ...
- The size of window is n-1. For e.g. In this case it is 7. Therefore, a maximum of n-l frames may be sent before an acknowledgment.
- When the receiver sends an ACK, it includes the number of next frame it expects to receive. For example in order to acknowledge the group of frames ending in frame 4, the receiver sends an ACK containing the number 5. When sender sees an ACK with number 5, it comes to know that all the frames up to number 4 have been received.

Sliding Window on Sender Side

- At the beginning of a transmission, the sender's window contains n-l frames.
- As the frames are sent by source, the left boundary of the window moves inward, shrinking the size of window. This means if window size is w, if four frames are sent by source after the last acknowledgment, then the number of frames left in window is w-4.
- When the receiver sends an ACK, the source's window expand i.e. (right boundary moves outward) to allow in a number of new frames equal to the number of frames acknowledged by that ACK.
- For example, Let the window size is 7 (see diagram (a)), if frames 0 through 3 have been sent and no acknowledgment has been received, then the sender's window contains three frames - 4,5,6.
- Now, if an ACK numbered 3 is received by source, it means three frames (0, 1, 2) have been received by receiver and are undamaged.
- The sender's window will now expand to include the next three frames in its buffer. At this point the sender's window will contain six frames (4, 5, 6, 7, 0, 1). (See diagram (b)).

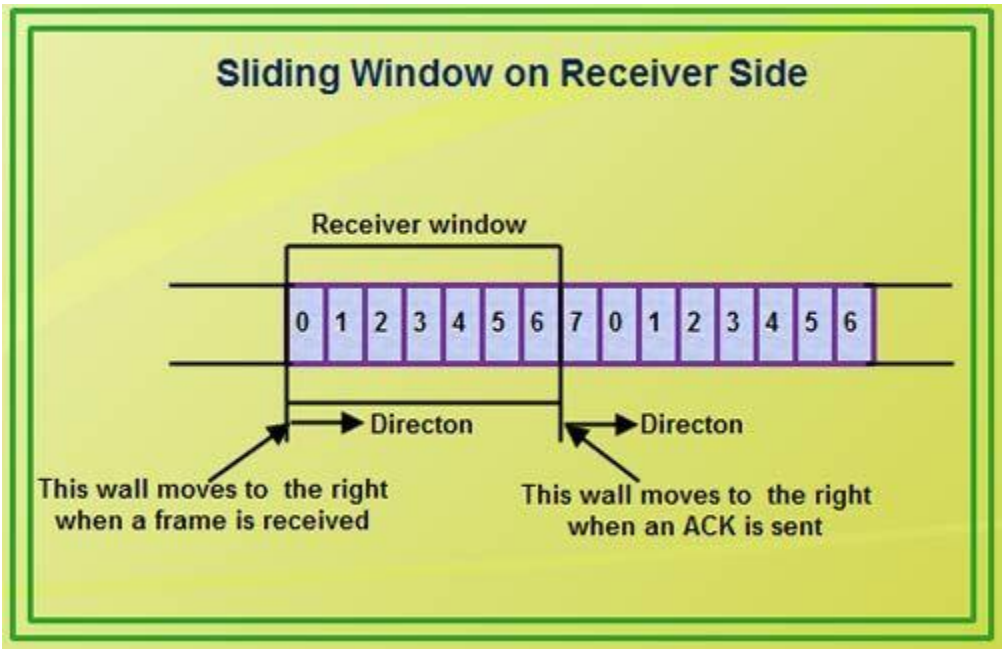


Sliding Window on Receiver Side

- At the beginning of transmission, the receiver's window contains n-1 spaces for frame but not the frames.
- As the new frames come in, the size of window shrinks.
- Therefore the receiver window represents not the number of frames received but the number of frames that may still be received without an acknowledgment ACK must

be sent.

- Given a window of size w, if three frames are received without an ACK being returned, the number of spaces in a window is w-3.
- As soon as acknowledgment is sent, window expands to include the number of frames equal to the number of frames acknowledged.
- For example, let the size of receiver's window is 7 as shown in diagram. It means window contains spaces for 7 frames.
- With the arrival of the first frame, the receiving window shrinks, moving the boundary from space 0 to 1. Now, window has shrunk by one, so the receiver may accept six more frame before it is required to send an ACK.
- If frames 0 through 3 have arrived but have DOC been acknowledged, the window will contain three frame spaces.
- As receiver sends an ACK, the window of the receiver expands to include as many new placeholders as newly acknowledged frames.
- The window expands to include a number of new frame spaces equal to the number of the most recently acknowledged frame minus the number of previously acknowledged frame. For e.g., If window size is 7 and if prior ACK was for frame 2 & the current ACK is for frame 5 the window expands by three (5-2).

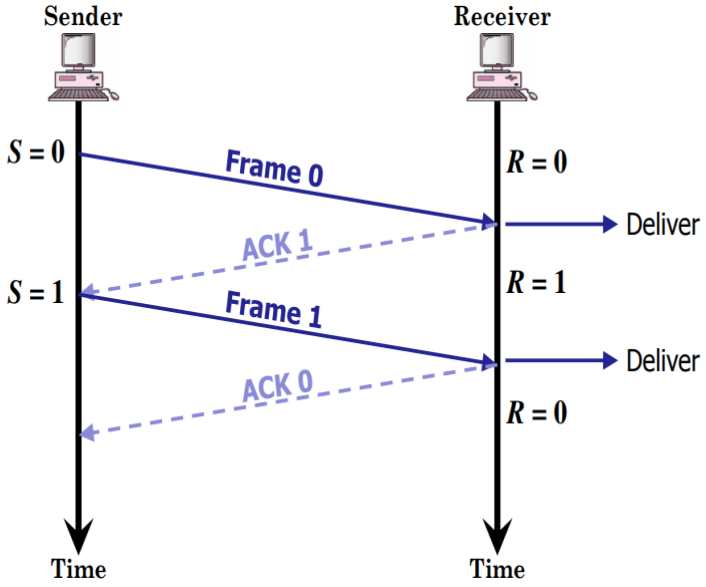


- Therefore, the sliding window of sender shrinks from left when frames of data are sending. The sliding window of the sender expands to right when acknowledgments are received.
- The sliding window of the receiver shrinks from left when frames of data are received. The sliding window of the receiver expands to the right when acknowledgement is sent.

There are three types of techniques available which Data-link layer may deploy to control the errors by Automatic Repeat Requests (ARQ):

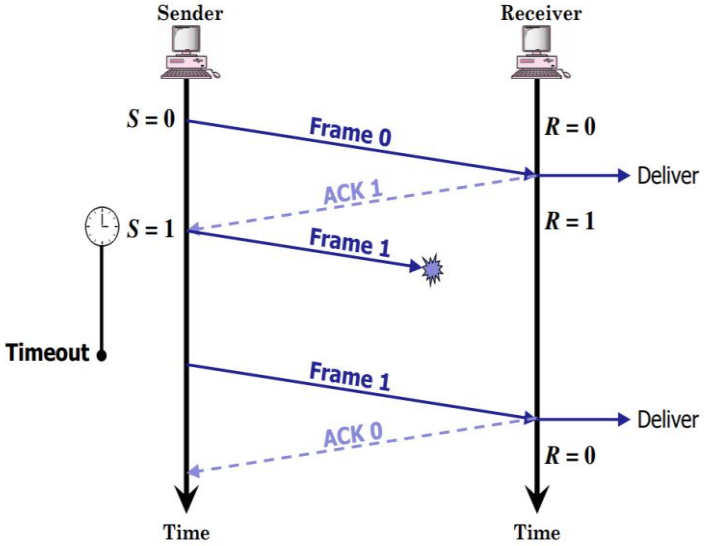
Stop-and-Wait ARQ

In Stop-and-Wait ARQ, which is simplest among all protocols, the sender (say station A) transmits a frame and then waits till it receives positive acknowledgement (ACK) or negative acknowledgement (NACK) from the receiver (say station B). Station B sends an ACK if the frame is received correctly, otherwise it sends NACK. Station A sends a new frame after receiving ACK; otherwise it retransmits the old frame, if it receives a NACK.

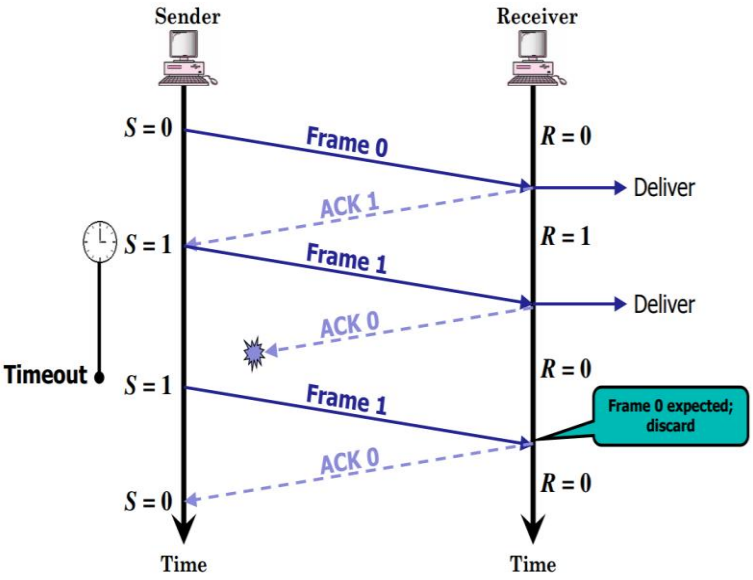


Normal operation

To tackle the problem of a lost or damaged frame, the sender is equipped with a timer. In case of a lost ACK, the sender transmits the old frame. The sender is unaware of this loss, but starts a timer after sending each PDU. Normally an ACK PDU is received before the timer expires. In this case no ACK is received, and the timer counts down to zero and triggers retransmission of the same PDU by the sender.



Lost Frame



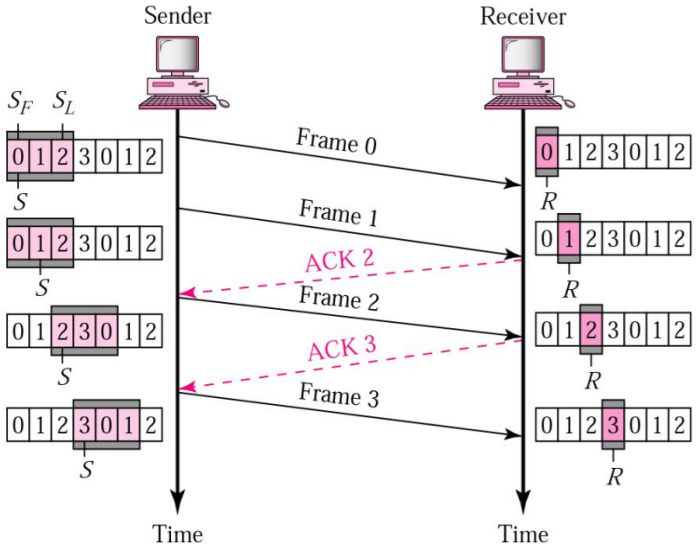
Lost ACK

The receiver now can identify that it has received a duplicate frame from the label of the frame and it is discarded.

The main advantage of stop-and-wait ARQ is its simplicity. It also requires minimum buffer size. However, it makes highly inefficient use of communication links.

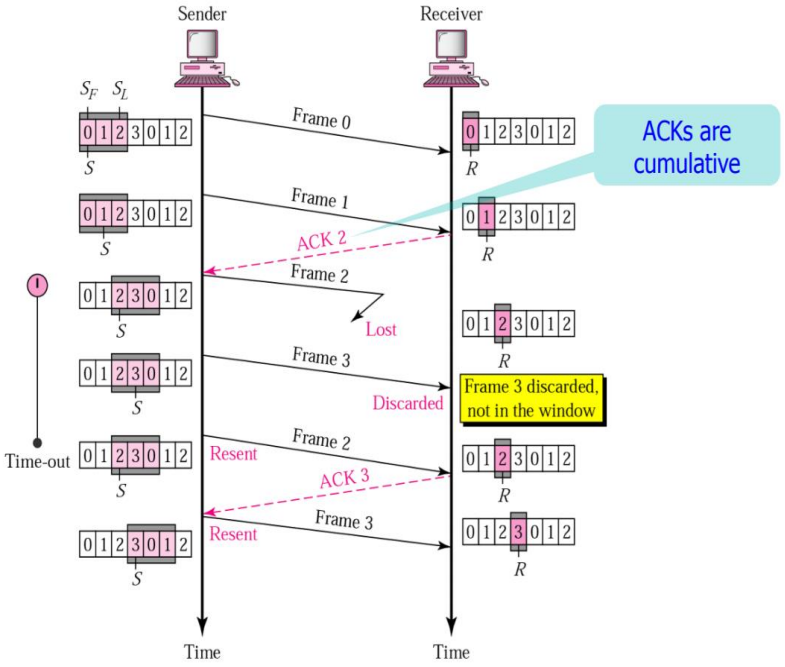
Go-back-N ARQ

The most popular ARQ protocol is the go-back-N ARQ, where the sender sends the frames continuously without waiting for acknowledgement. That is why it is also called as continuous ARQ. As the receiver receives the frames, it keeps on sending ACKs or a NACK, in case a frame is incorrectly received. When the sender receives a NACK, it retransmits the frame in error plus all the succeeding frames. Hence, the name of the protocol is go-back-N ARQ



Normal Operation

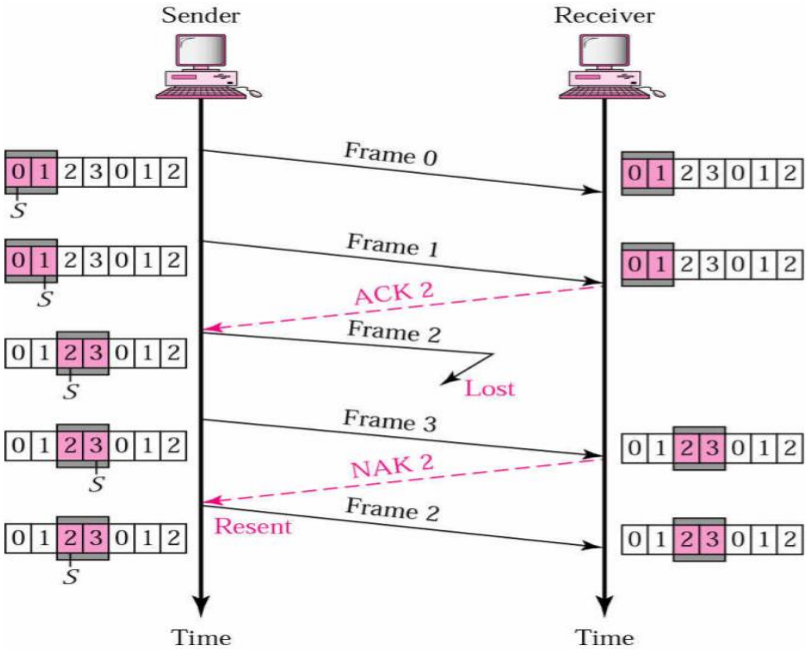
If a frame is lost, the receiver sends NAK after receiving the next frame. In case there is long delay before sending the NAK, the sender will resend the lost frame after its timer times out. If the ACK frame sent by the receiver is lost, the sender resends the frames after its timer times out



Lot Frame

Selective-Repeat ARQ

The selective-repetitive ARQ scheme retransmits only those for which NAKs are received or for which timer has expired. This is the most efficient among the ARQ schemes, but the sender must be more complex so that it can send out-of-order frames. The receiver also must have storage space to store the post NAK frames and processing power to reinsert frames in proper sequence.



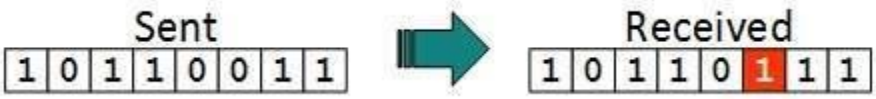
Error Control Mechanisms

Data-link layer uses some error control mechanism to ensure that frames (data bit streams) are transmitted with certain level of accuracy. But to understand how errors is controlled, it is essential to know what types of errors may occur.

Types of Errors

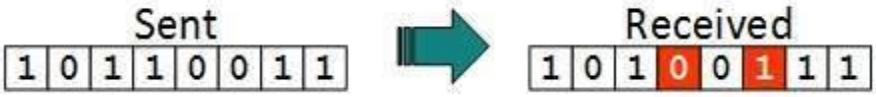
There may be three types of errors:

- **Single bit error**



In a frame, there is only one bit, anywhere though, which is corrupt.

- **Multiple bits error**



Frame is received with more than one bit in corrupted state.

- **Burst error**

Frame contains more than one consecutive bits corrupted.

Error control mechanism may involve two possible ways:

- Error detection
- Error correction

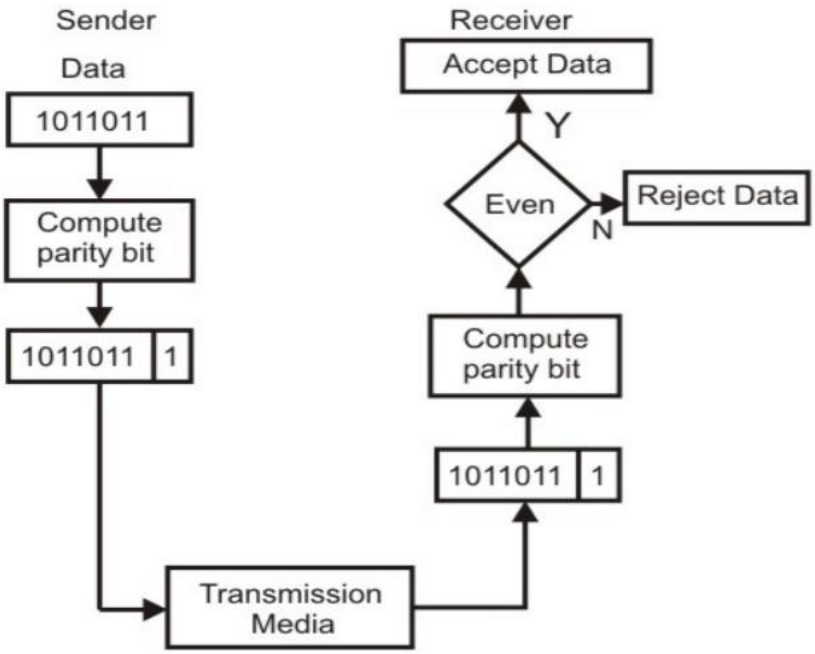
Error Detecting Codes
Basic approach used for error detection is the use of redundancy, where additional bits are added to facilitate detection and correction of errors. Popular techniques are:

- Simple Parity check
- Two-dimensional Parity check
- Cyclic redundancy check

Simple Parity Checking or One-dimension Parity Check

The most common and least expensive mechanism for error- detection is the simple parity check. In this technique, a redundant bit called parity bit, is appended to every data unit so that the number of 1s in the unit (including the parity becomes even).

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 an odd number of 1's (ON bits) and 0 is added if it contains an even number of 1's. At the receiving end the parity bit is computed from the received data bits and compared with the received parity bit, as shown in figure. This scheme makes the total number of 1's even, that is why it is called even parity checking.



Even-parity checking scheme

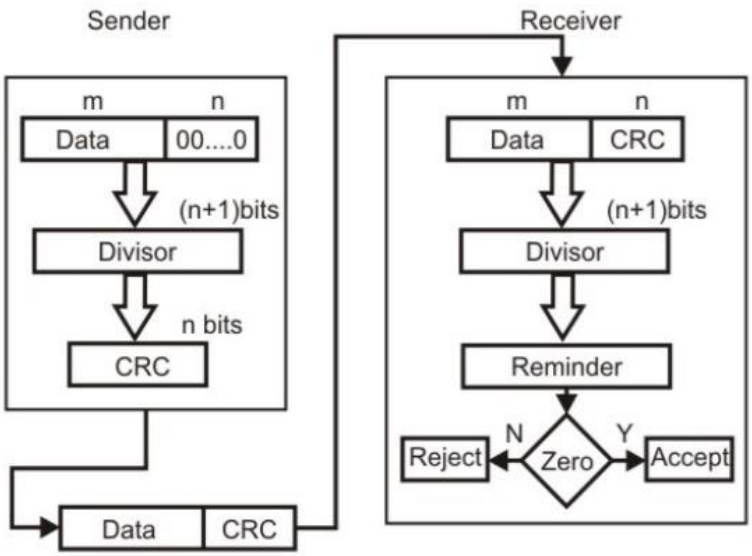
Performance

An observation of the table reveals that to move from one code word to another, at least two data bits should be changed. Hence these set of code words are said to have a minimum distance (hamming distance) of 2, which means that a receiver that has knowledge of the code word set can detect all single bit errors in each code word. However, if two errors occur in the code word, it becomes another valid member of the set and the decoder will see only another valid code word and know nothing of the error. Thus errors in more than one bit cannot be detected. In fact it can be shown that a single parity check code can detect only odd number of errors in a code word.

Cyclic Redundancy Checks (CRC)

This Cyclic Redundancy Check is the most powerful and easy to implement technique. 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.

The generalized technique can be explained as follows. If a k bit message is to be transmitted, the transmitter generates an r-bit sequence, known as Frame Check Sequence (FCS) so that the (k+r) bits are actually being transmitted. Now this r-bit FCS is generated by dividing the original number, appended by r zeros, by a predetermined number. This number, which is (r+1) bit in length, can also be considered as the coefficients of a polynomial, called Generator Polynomial. The remainder of this division process generates the r-bit FCS. On receiving the packet, the receiver divides the (k+r) bit frame by the same predetermined number and if it produces no remainder, it can be assumed that no error has occurred during the transmission. Operations at both the sender and receiver end are shown in figure.



Basic scheme for Cyclic Redundancy Checking

This mathematical operation performed is illustrated in figure by dividing a sample 4- bit number by the coefficient of the generator polynomial x3 +x+1, which is 1011, using the modulo-2 arithmetic. Modulo-2 arithmetic is a binary addition process without any carry over, which is just the Exclusive-OR operation. Consider the case where k=1101. Hence we have to divide 1101000 (i.e. k appended by 3 zeros) by 1011, which produces the remainder r=001, so that the bit frame (k+r) =1101001 is actually being transmitted through the communication channel. At the receiving end, if the received number, i.e., 1101001 is divided by the same generator polynomial 1011 to get the remainder as 000, it can be assumed that the data is free of errors.

```
          1 1 1 1   <-- k
      ____________
1011 | 1 1 0 1 0 0 0
       1 0 1 1
       -------
         1 1 0 0
         1 0 1 1
         -------
           1 1 1 0
           1 0 1 1
           -------
             1 0 1 0
             1 0 1 1
             -------
               0 0 1   <-- r
```

Performance

CRC is a very effective error detection technique. If the divisor is chosen according to the previously mentioned rules, its performance can be summarized as follows:

- CRC can detect all single-bit errors
- CRC can detect all double-bit errors (three 1's)
- CRC can detect any odd number of errors (X+1)
- CRC can detect all burst errors of less than the degree of the polynomial.
- CRC detects most of the larger burst errors with a high probability.
- For example CRC-12 detects 99.97% of errors with a length 12 or more.

Error Correcting Codes

Error Correction can be handled in two ways.

- One is when an error is discovered; the receiver can have the sender retransmit the entire data unit. This is known as backward error correction.
- In the other, receiver can use an error-correcting code, which automatically corrects certain errors. This is known as forward error correction.

Hamming Codes

The most common types of error-correcting codes used in RAM are based on the codes devised by R. W. Hamming. In the Hamming code, k parity bits are added to an n-bit data word, forming a new word of n+k bits. The bit positions are numbered in sequence from 1 to n+k. Those positions numbered with powers of two are reserved for the parity bits. The remaining bits are the data bits. The code can be used with words of any length.

General Algorithm of Hamming code

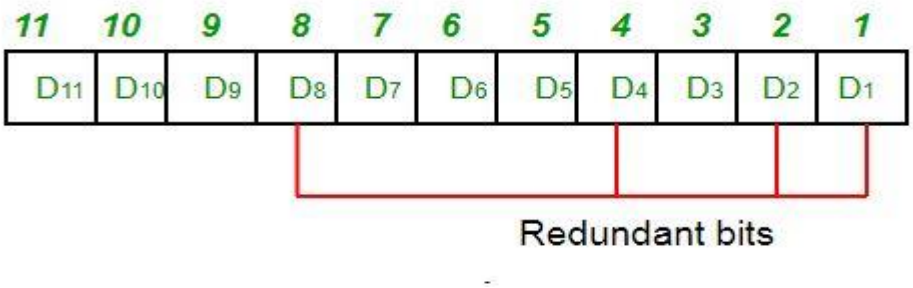
The Hamming Code is simply the use of extra parity bits to allow the identification of an error.

I. Write the bit positions starting from 1 in binary form (1, 10, 11, 100, etc).
II. All the bit positions that are a power of 2 are marked as parity bits (1, 2, 4, 8, etc).
III. All the other bit positions are marked as data bits.
IV. Each data bit is included in a unique set of parity bits, as determined its bit position in binary form.
    a Parity bit 1 covers all the bits positions whose binary representation includes a 1 in the least significant position (1, 3, 5, 7, 9, 11, etc).
    b Parity bit 2 covers all the bits positions whose binary representation includes a 1 in the second position from the least significant bit (2, 3, 6, 7, 10, 11, etc).
    c Parity bit 4 covers all the bits positions whose binary representation includes a 1 in the third position from the least significant bit (4–7, 12–15, 20–23, etc).
    d Parity bit 8 covers all the bits positions whose binary representation includes a 1 in the fourth position from the least significant bit bits (8–15, 24–31, 40–47, etc).
    e In general each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero.
V. Since we check for even parity set a parity bit to 1 if the total number of ones in the positions it checks is odd.
VI. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

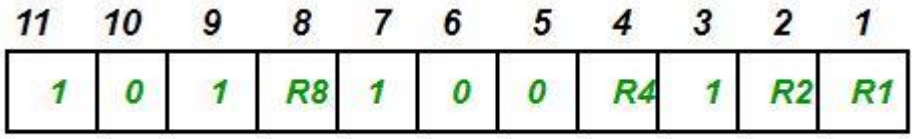
Determining the position of redundant bits

These redundancy bits are placed at the positions which correspond to the power of 2. As in the above example:

1. The number of data bits = 7
2. The number of redundant bits = 4
3. The total number of bits = 11
4. The redundant bits are placed at positions corresponding to power of 2: 1, 2, 4, and 8

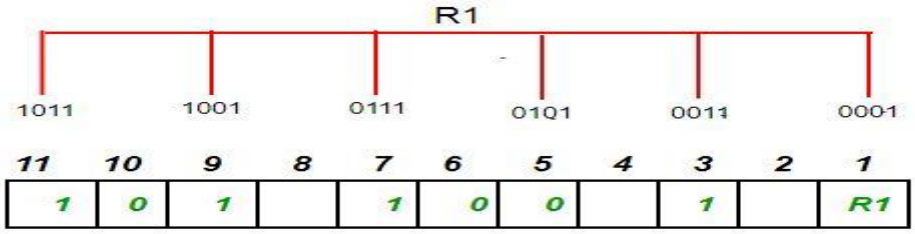


Suppose the data to be transmitted is 1011001, the bits will be placed as follows:



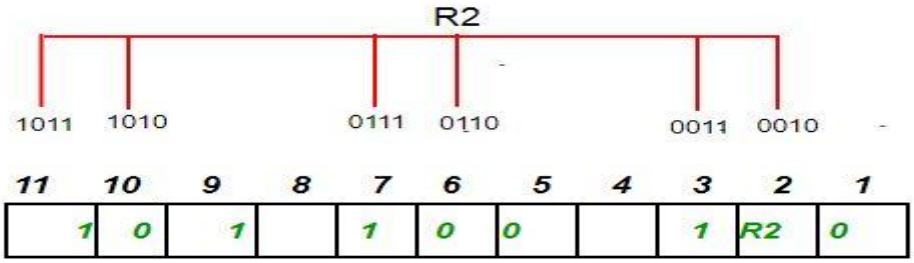
Determining the Parity bits –

1. R1 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the least significant position.
   R1: bits 1, 3, 5, 7, 9, 11



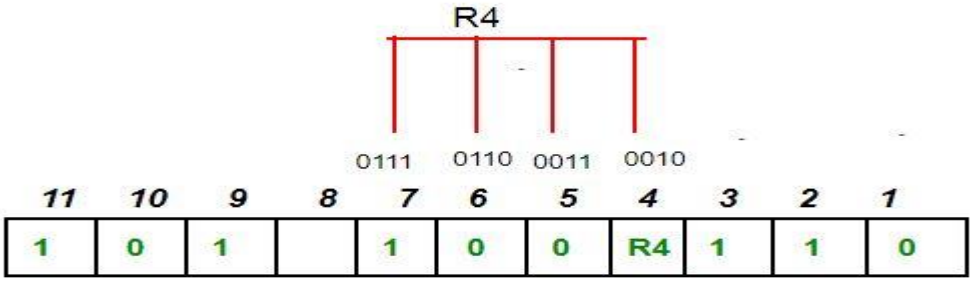
   To find the redundant bit R1, we check for even parity. Since the total number of 1's in all the bit positions corresponding to R1 is an even number the value of R1 (parity bit's value) = 0

2. R2 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the second position from the least significant bit.
   R2: bits 2,3,6,7,10,11



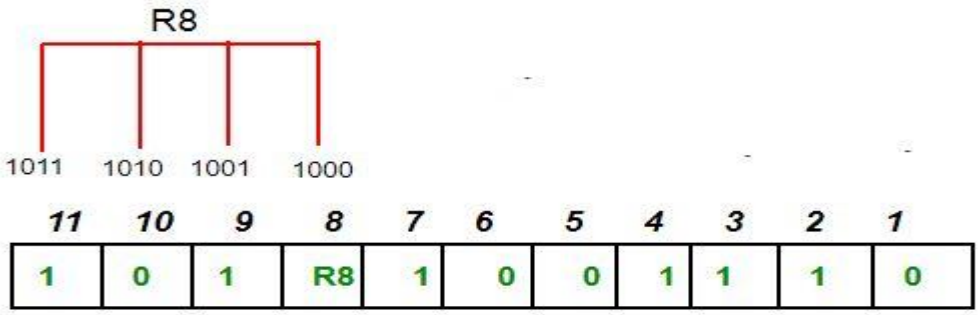
   To find the redundant bit R2, we check for even parity. Since the total number of 1's in all the bit positions corresponding to R2 is an odd number the value of R2(parity bit's value)=1

3. R4 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the third position from the least significant bit.
   R4: bits 4, 5, 6, 7



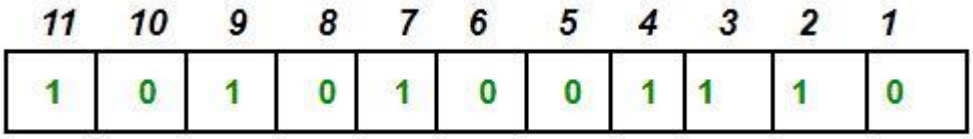
To find the redundant bit R4, we check for even parity. Since the total number of 1's in all the bit positions corresponding to R4 is an odd number the value of R4(parity bit's value) = 1

4. R8 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the fourth position from the least significant bit.
   R8: bit 8,9,10,11



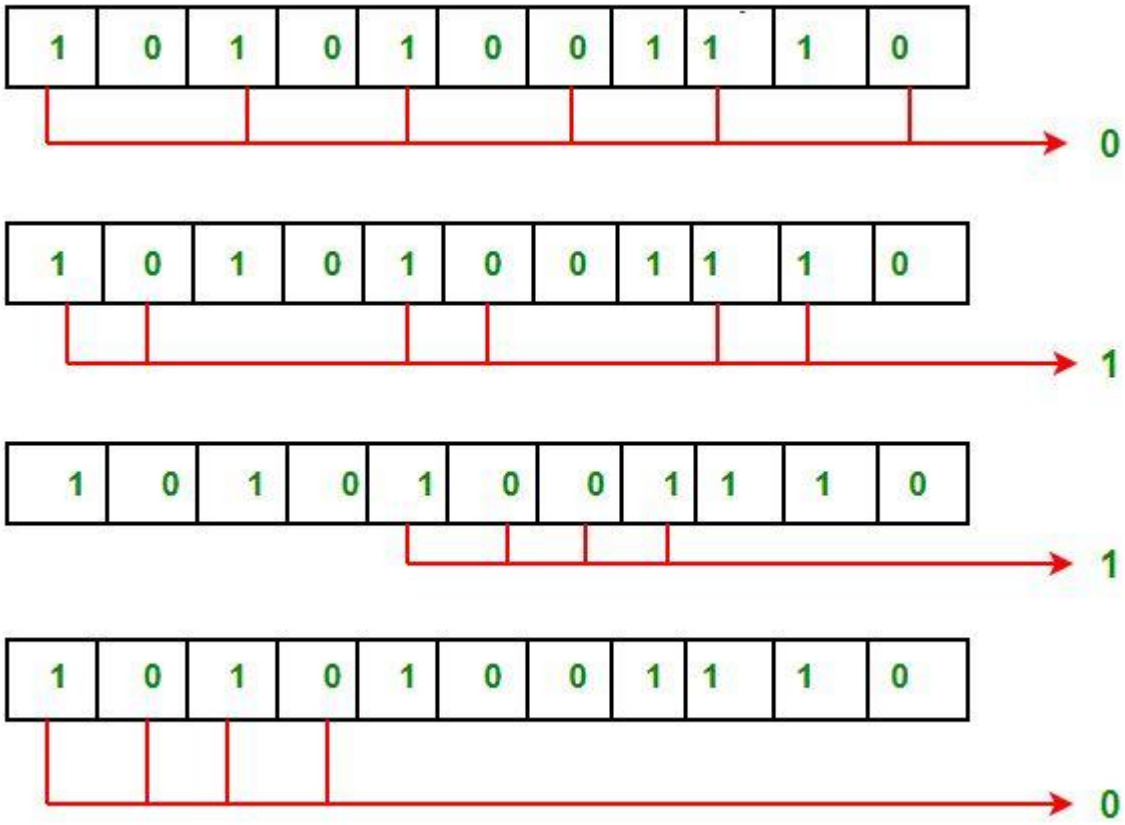
To find the redundant bit R8, we check for even parity. Since the total number of 1's in all the bit positions corresponding to R8 is an even number the value of R8(parity bit's value)=0.

Thus, the data transferred is:



Error detection and correction

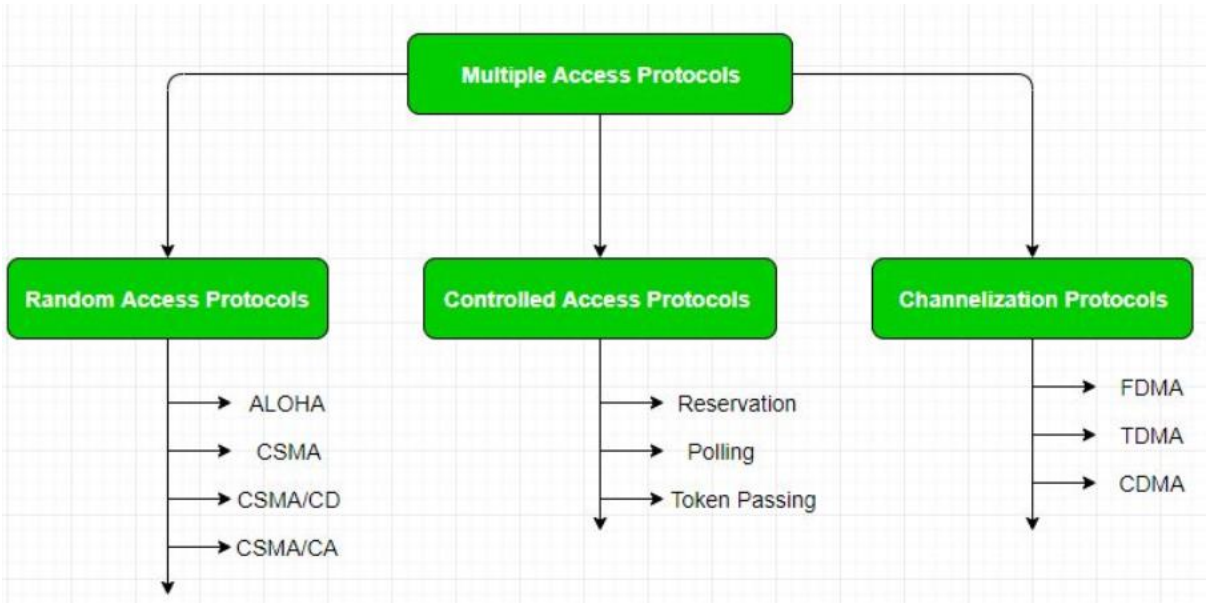
Suppose in the above example the 6th bit is changed from 0 to 1 during data transmission, then it gives new parity values in the binary number:



The bits give the binary number as 0110 whose decimal representation is 6. Thus, the bit 6 contains an error. To correct the error the 6th bit is changed from 1 to 0.

Multiple Access Control

If there is a dedicated link between the sender and the receiver then data link control layer is sufficient, however if there is no dedicated link present then multiple stations can access the channel simultaneously. Hence multiple access protocols are required to decrease collision and avoid crosstalk. For example, in a classroom full of students, when a teacher asks a question and all the students (or stations) start answering simultaneously (send data at same time) then a lot of chaos is created (data overlap or data lost) then it is the job of the teacher (multiple access protocols) to manage the students and make them answer one at a time.
Thus, protocols are required for sharing data on non-dedicated channels. Multiple access protocols can be subdivided further as –



Random Access Protocol
In this, all stations have same superiority that is no station has more priority than another station. Any station can send data depending on medium's state (idle or busy). It has two features:

1. There is no fixed time for sending data
2. There is no fixed sequence of stations sending data

The Random access protocols are further subdivided as:

1. ALOHA
2. CSMA
    - CSMA/CD
    - CSMA/CA

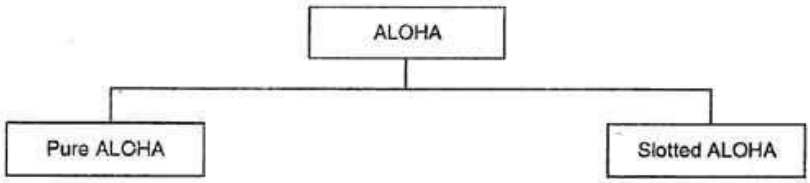
ALOHA

ALOHA is a system for coordinating and arbitrating access to a shared communication Networks channel. It was developed in the 1970s by Norman Abramson and his colleagues at the University of Hawaii. The original system used for ground based radio broadcasting, but the system has been implemented in satellite communication systems.

A shared communication system like ALOHA requires a method of handling collisions that occur when two or more systems attempt to transmit on the channel at the same time. In the ALOHA system, a node transmits whenever data is available to send. If another node

transmits at the same time, a collision occurs, and the frames that were transmitted are lost. However, a node can listen to broadcasts on the medium, even its own, and determine whether the frames were transmitted. Aloha is a multiple access protocol at the datalink layer and proposes how multiple terminals access the medium without interference or collision.

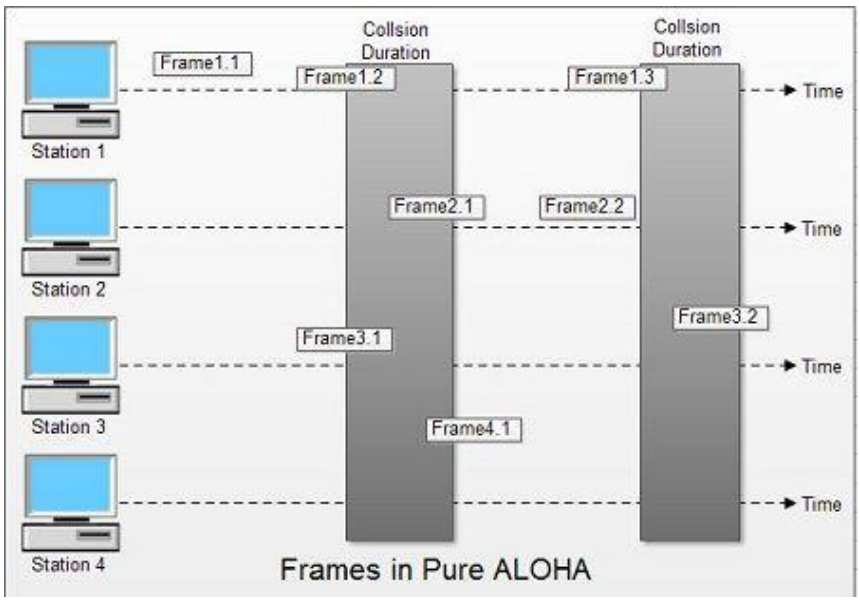
There are two different versions of ALOHA



Types of ALOHA

Pure ALOHA

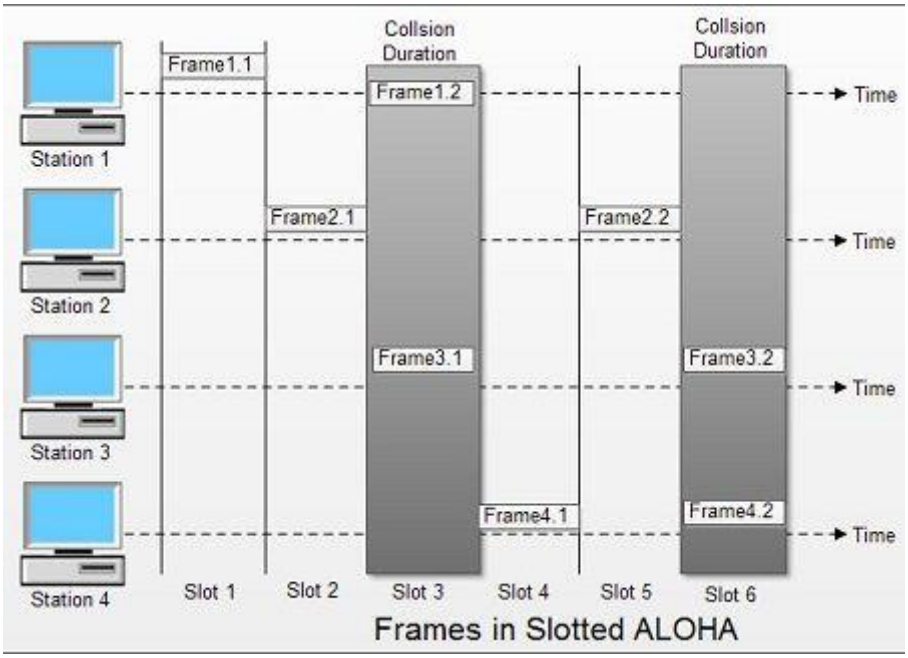
- In pure ALOHA, the stations transmit frames whenever they have data to send.
- When two or more stations transmit simultaneously, there is collision and the frames are destroyed.
- In pure ALOHA, whenever any station transmits a frame, it expects the acknowledgement from the receiver.
- If acknowledgement 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.
- Figure shows an example of frame collisions in pure ALOHA.



- In fig there are four stations that .contended with one another for access to shared channel. All these stations are transmitting frames. Some of these frames collide because multiple frames are in contention for the shared channel. Only two frames, frame 1.1 and frame 2.2 survive. All other frames are destroyed.
- Whenever two frames try to occupy the channel at the same time, there will be a collision and both will be damaged. If first bit of a new frame overlaps with just the last bit of a frame almost finished, both frames will be totally destroyed and both will have to be retransmitted.

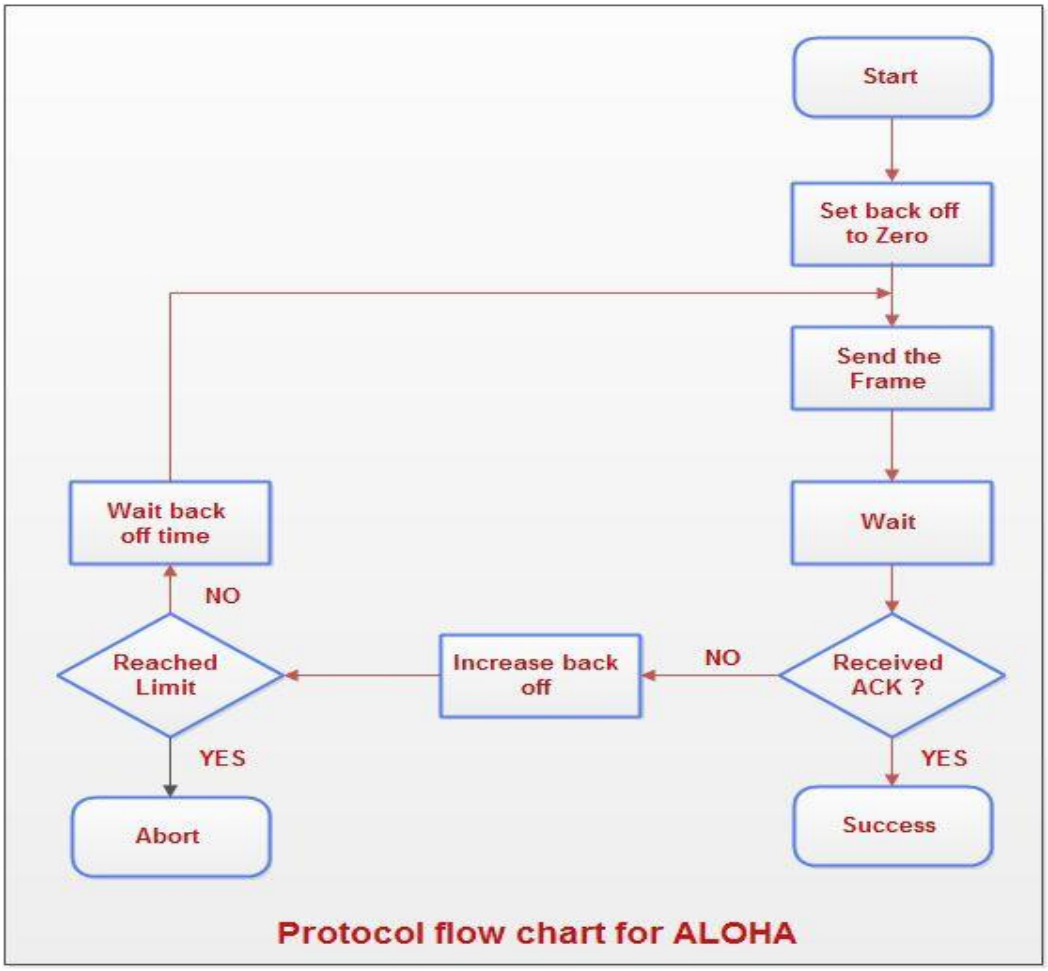
Slotted ALOHA

- Slotted ALOHA was invented to improve the efficiency of pure ALOHA as chances of collision in pure ALOHA are very high
- In slotted ALOHA, the time of the shared channel is divided into discrete intervals called slots.
- The stations can send a frame only at the beginning of the slot and only one frame is sent in each slot.



- In slotted ALOHA, if any station is not able to place the frame onto the channel at the beginning of the slot *i.e.* it misses the time slot then the station has to wait until the beginning of the next time slot.
- In slotted ALOHA, there is still a possibility of collision if two stations try to send at the beginning of the same time slot as shown in fig.
- Slotted ALOHA still has an edge over pure ALOHA as chances of collision are reduced to one-half

Protocol Flow Chart for ALOHA



Explanation:

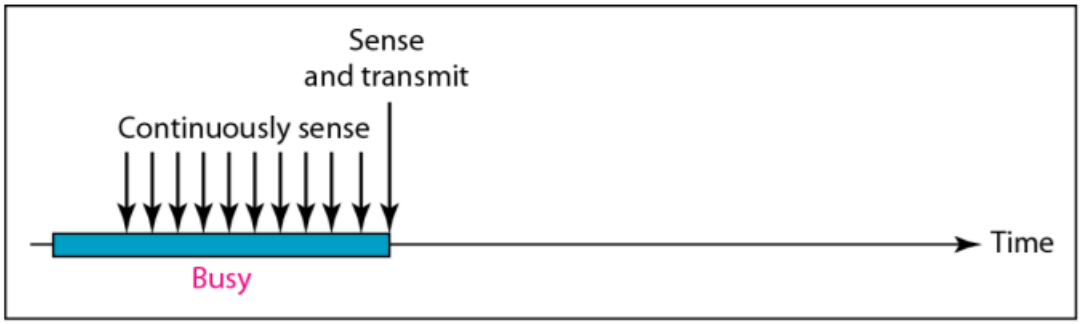
- A station which has a frame ready will send it.
- Then it waits for some time.
- If it receives the acknowledgement then the transmission is successful.
- Otherwise the station uses a backoff strategy, and sends the packet again.
- After many times if there is no acknowledgement then the station aborts the idea of transmission.

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. CSMA requires that each station first listen to the medium (or check the state of the medium) before sending. it is based on the principle "sense before transmit". CSMA can reduce the possibility of collision, but it cannot eliminate it. The possibility of collision still exists because of propagation delay.

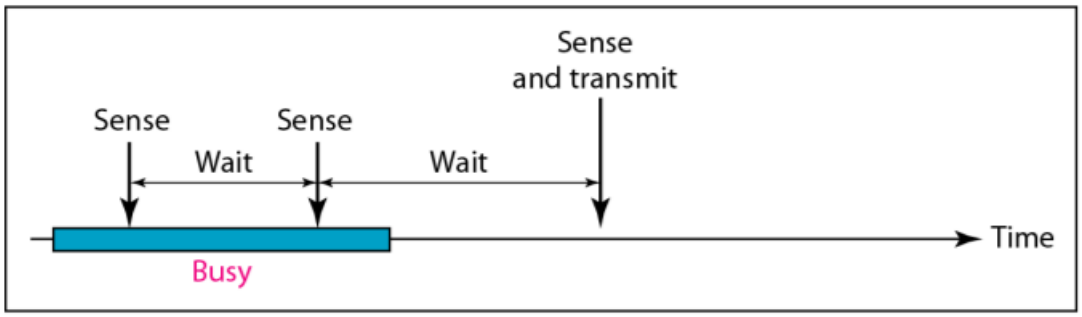
What should a station do if the channel is busy or idle? Three methods have been devised are:

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



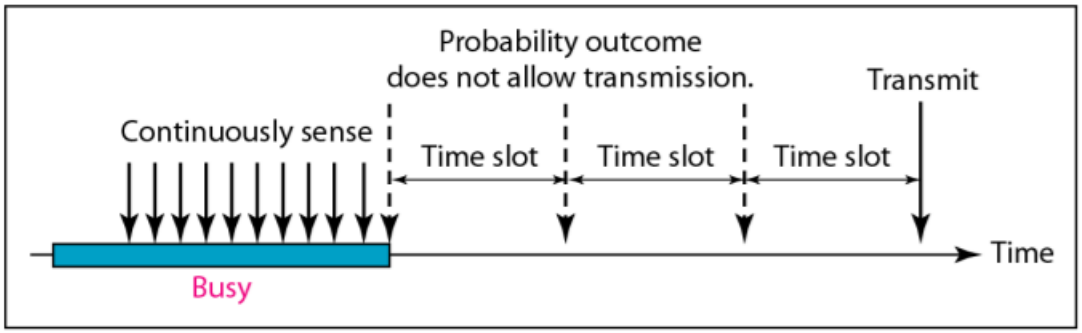
a. 1-persistent

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



b. Nonpersistent

3. **P-Persistent**: The p-persistent method is used if the channel has time slots with a slot duration equal to or greater than the maximum propagation time. ν The p-persistent approach combines the advantages of the other two strategies. It reduces the chance of collision and improves efficiency.



c. p-persistent

CSMA/CD

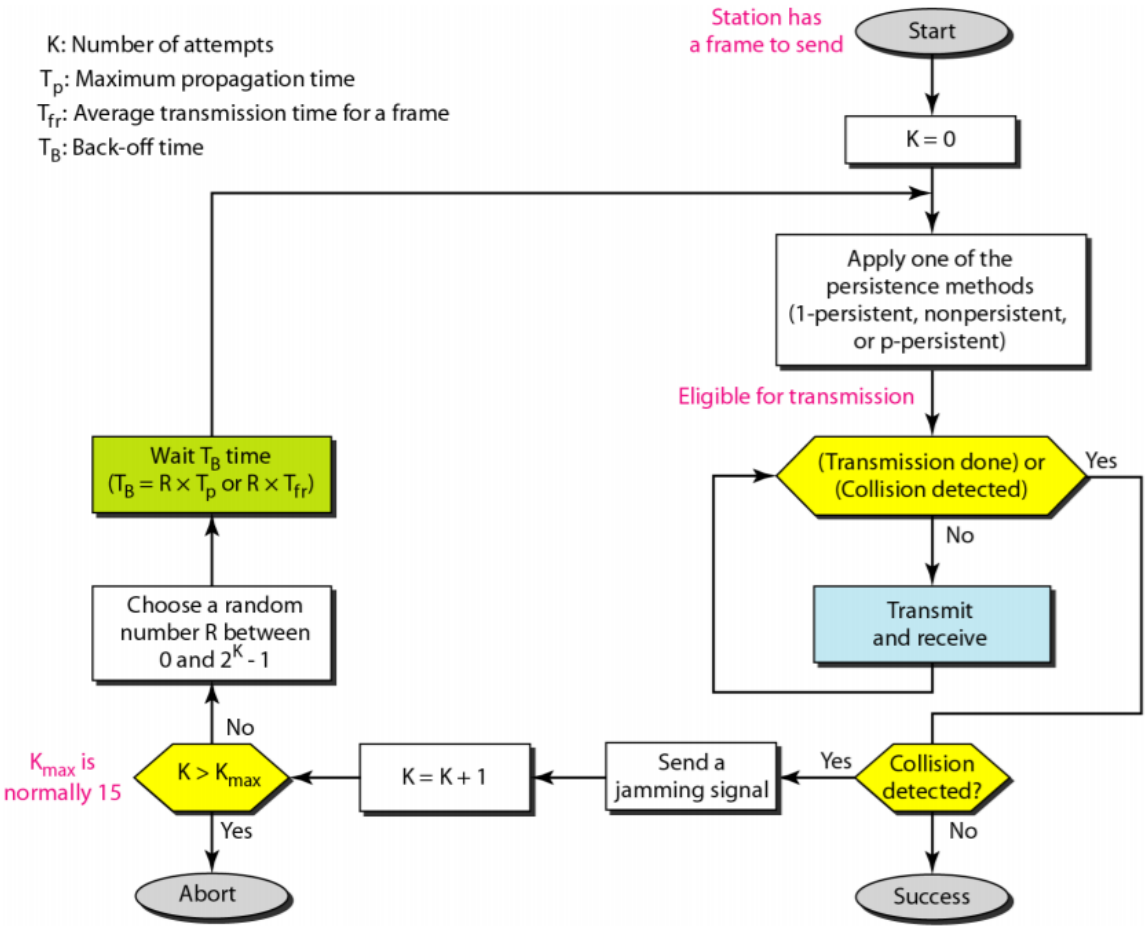
CSMA/CD protocol can be considered as a refinement over the CSMA scheme. It has evolved to overcome one glaring inefficiency of CSMA. In CSMA scheme, when two packets collide the channel remains unutilized for the entire duration of transmission time of both the packets. If the propagation time is small (which is usually the case) compared to the packet transmission time, wasted channel capacity can be considerable. This wastage of channel capacity can be reduced if the nodes continue to monitor the channel while transmitting a packet and immediately cease transmission when collision is detected. This refined scheme is known as Carrier Sensed Multiple Access with Collision Detection (CSMA/CD) or Listen-While-Talk.

On top of the CSMA, the following rules are added to convert it into CSMA/CD:

I. If a collision is detected during transmission of a packet, the node immediately ceases transmission and it transmits jamming signal for a brief duration to ensure that all stations know that collision has occurred.

II. After transmitting the jamming signal, the node waits for a random amount of time and then transmission is resumed.

Flow diagram for the CSMA/CD



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

CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) is a protocol for carrier transmission in networks. Unlike CSMA/CD (Carrier Sense Multiple Access/Collision Detect) which deals with transmissions after a collision has occurred, CSMA/CA acts to prevent collisions before they happen. In CSMA/CA, as soon as a node receives a packet that is to be sent, it checks to be sure the channel is clear (no other node is transmitting at the time). If the channel is clear, then the packet is sent. If the channel is not clear, the node waits for a randomly chosen period of time, and then checks again to see if the channel is clear. This period of time is called the backoff factor, and is counted down by a backoff counter. If the channel is clear when the backoff counter reaches zero, the node transmits the packet. If the channel is not clear when the backoff counter reaches zero, the backoff factor is set again, and the process is repeated.

Flow diagram for the CSMA/CD