# Chapter 3

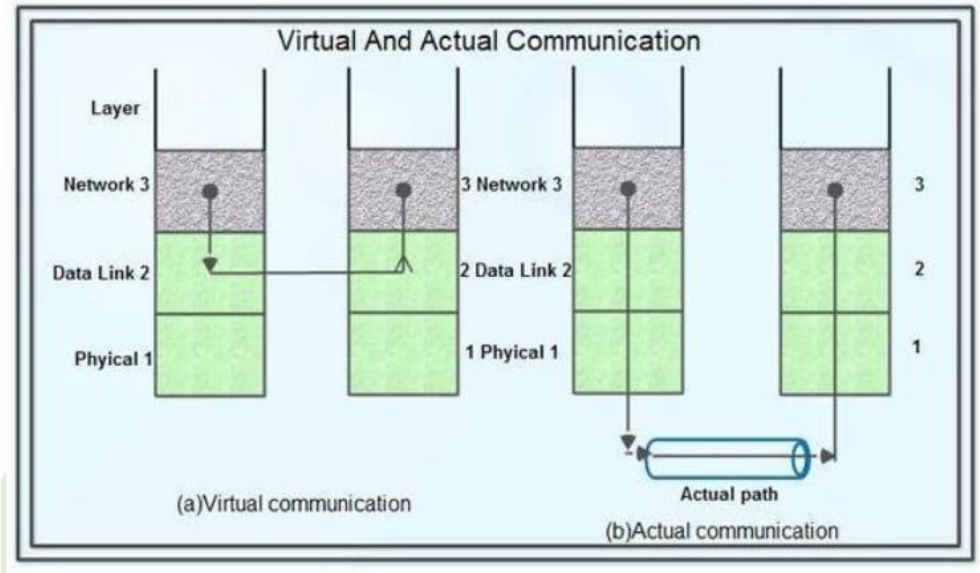
# Data Link Layer

## 1. Introduction

The Data Link Layer is the second layer in the OSI model, above the Physical Layer, which ensures that the error free data is transferred between the adjacent nodes in the network. It breaks the datagram passed down by above layers and converts them into frames ready for transfer.

This layer is responsible for achieving reliable, efficient communication between two adjacent machines which are directly connected by a communication channel.

Hence, it is a very important layer and takes the responsibility of transmitting data from one node in the network to the next.




## 2. Functions of Data Link Layer

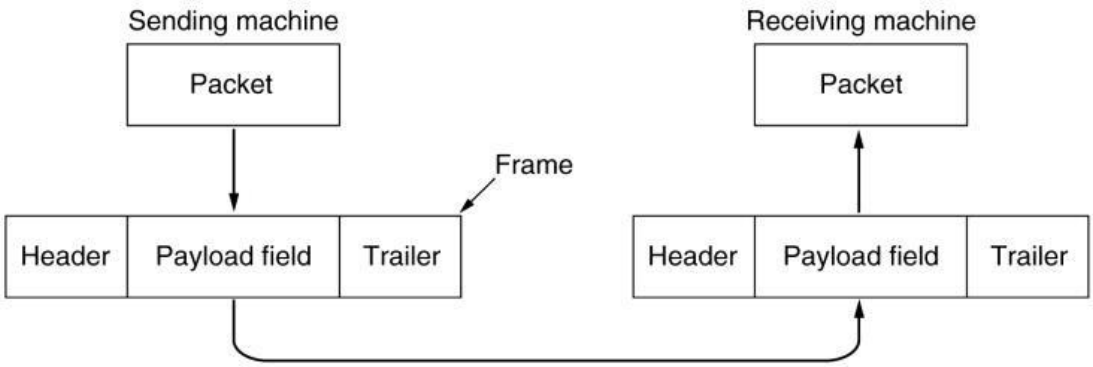
Specific responsibilities of the data link layer include framing, addressing, flow control, error control, and media access control:

- The data link layer divides the stream of bits received from the network layer into manageable data units called frames. The data link layer adds a header to the frame to define the addresses of the sender and receiver of the frame.

- If the rate at which the data are absorbed by the receiver is less than the rate at which data are produced in the sender, the data link layer imposes a flow control mechanism to avoid overwhelming the receiver.
- The data link layer also adds reliability to the physical layer by adding mechanisms to detect and retransmit damaged, duplicate, or lost frames.
- When two or more devices are connected to the same link, data link layer protocols are necessary to determine which device has control over the link at any given time.

## 3. Framing

One of the important functions performed by the data link layer is framing. The sending data link layer takes a packet from the network layer and puts it into frames for transmission. The frame contains a frame header, a payload field for storing the packet and a frame trailer.




**Packets and Frames**

In order to identify the start and end of a frame, it must have delimiters. Four commonly use methods for framing are:

- Character Count
- Flag bytes with byte stuffing
- Starting and Ending flags with bit stuffing
- Physical Layer Coding Violations

### 3.1. Character Count

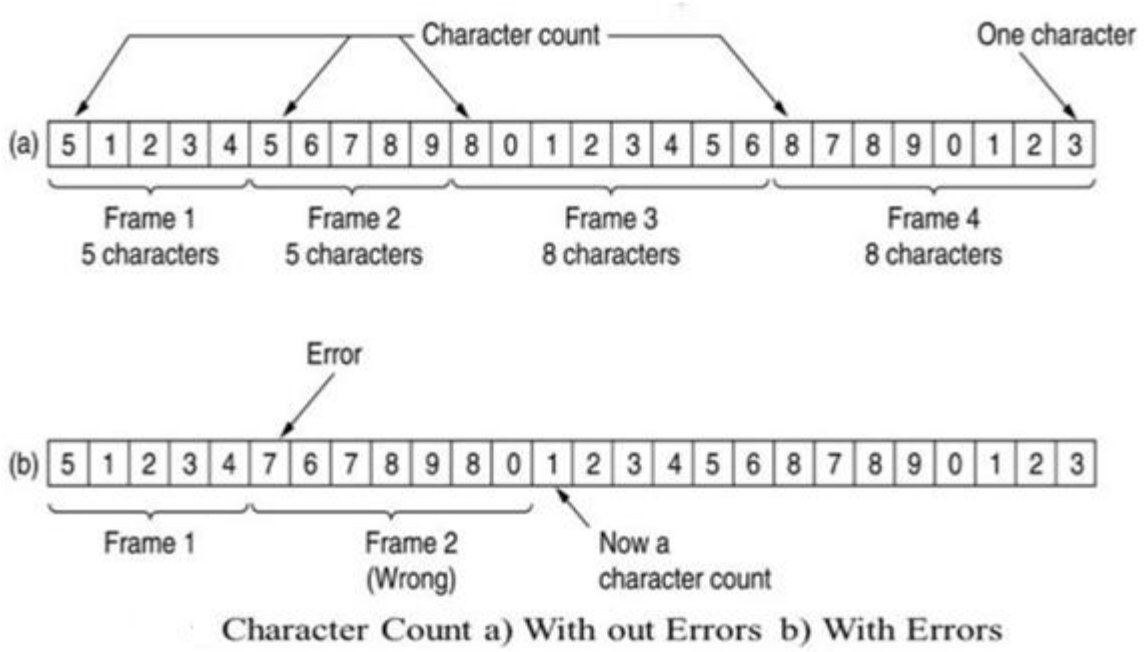
In this method of framing, a field in the frame header is used to store the number of characters in the frame.

When the receiving data link layer receives the frame, it reads the character count and knows how many characters are in the frame. Hence, it can detect the end of the frame. The following example shows this technique for four frames of sizes 5, 5, 8, 8 respectively.

One problem with this method is that a transmission error can modify the count value. If this happens, there is no way of telling where the next frame starts or ends.

Even if the receiver detects a transmission error, the receiver cannot ask for retransmission of a specific frame since it does not know which frame had the error. Thus, the entire data need to be retransmitted.

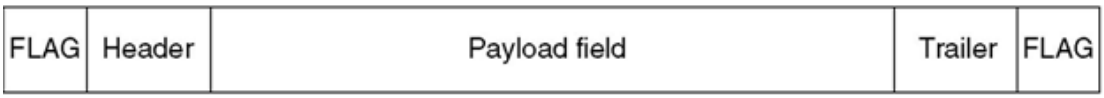
For example, if the count in the second frame gets modified from 8 to 7, the length of the next frame will be interpreted as 1.




Character Count a) With out Errors b) With Errors

## 3.2. Flag Bytes with Byte Stuffing

Some protocols use special bytes as frame delimiter. This byte is called flag byte and is used as the starting as well as the ending delimiter for a frame. This flag byte is used to identify start and end of the frame.

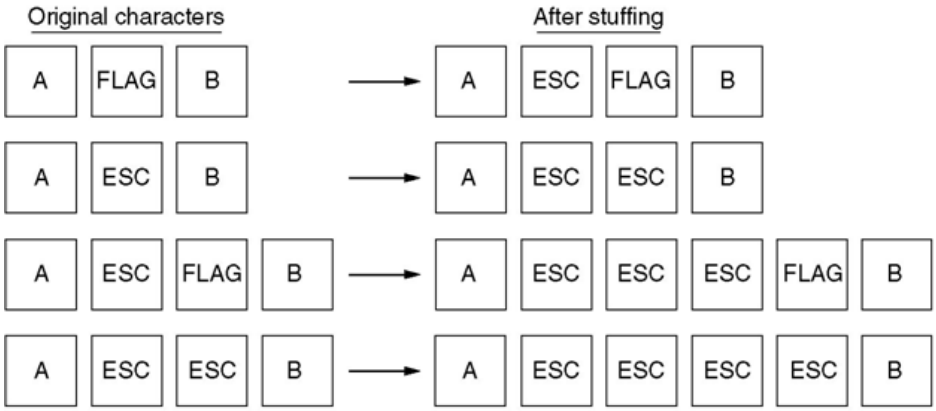



**Flag bytes**

However, a problem occurs when the flag byte's binary code occurs as a part of the data itself. The receiver will mistakenly interpret it as a frame delimiter, this problem can be solved using byte stuffing or character stuffing.

Here, a special byte called **Escape byte** is inserted or stuffed before each accidental flag byte in the data. The receiving data link layer removes the escape byte from the frame. In case an escape byte occurs in the data, it is also preceded by an inserted escape byte.

The problem with this method is that it allows only 8 bit characters and cannot be used for arbitrary sized characters.




**Character stuffing**

### 3.3. Starting and Ending Flags with Bit Stuffing

In this method, the data frames can contain an arbitrary number of bits and allows character codes with arbitrary number of bits per character. Each frame begins and ends with a special bit pattern called the flag byte which is **01111110**.

Here again, this bit pattern could occur in the data. To distinguish it from the flag byte, bit stuffing is used. When the sending data link layer encounters five consecutive l's in the data, it stuffs a **'0'** into the bit stream. The receiving data link layer de-stuffs the **'0'** following five consecutive 1's.

- **Example:**

**(a) Original data**
0 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0

**(b)After bit stuffing**
0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 0 0 1 0

stuffed bits

**(c) After destuffing**
0 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0

### 3.4. Physical Layer Coding Violations

This method of framing is used for networks in which the encoding scheme used for data contains some redundancy, i.e. there are some signal levels which are not used for transmitting data bits. Normally, bit 1 is encoded as a high-low pair and 0 is a low-high pair. Thus, every data bit has a transition in the middle and so, the receiver can easily identify bit boundaries. The high-high and low-low pairs are not used for data. These pairs can be used to represent frame boundaries.

# 4. Error Control

- Because of Attenuation, distortion, noise and interferences, errors during transmission are inevitable, leading to corruption transmitted bits.
- The data link layer has to ensure that the frames are sent and received properly at the receiving end. It has to take into consideration transmission errors.
- One way to deal with errors is to include enough redundant information along with the data so that receiver can deduce what the transmitted data must have been (Error correcting codes).
- The other strategy is to include only enough redundant information so that the receiver can deduce that an error has occurred but not which error, and request a retransmission (Error-detecting codes).

## 4.1. Error detection techniques

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
- Checksum
- Cyclic redundancy check

### *4.1.1. Simple parity check*

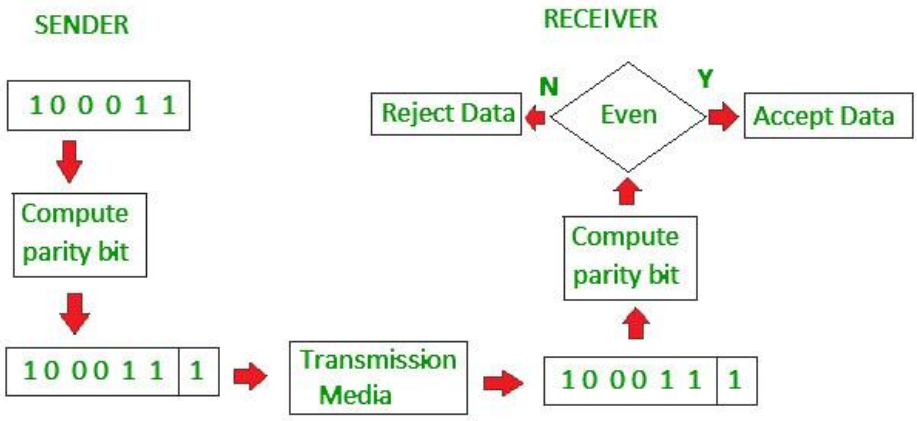
The simplest and most popular error detection scheme. Appends a Parity bit to the end of the data.

- **Even Parity**: **01000001 –** Number of ones in the group of bits is even
- **Odd Parity**: **11000001 –** Number of ones in the group of bits is odd

A parity of 1 is added to the block if it contains an odd number of 1's 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.

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

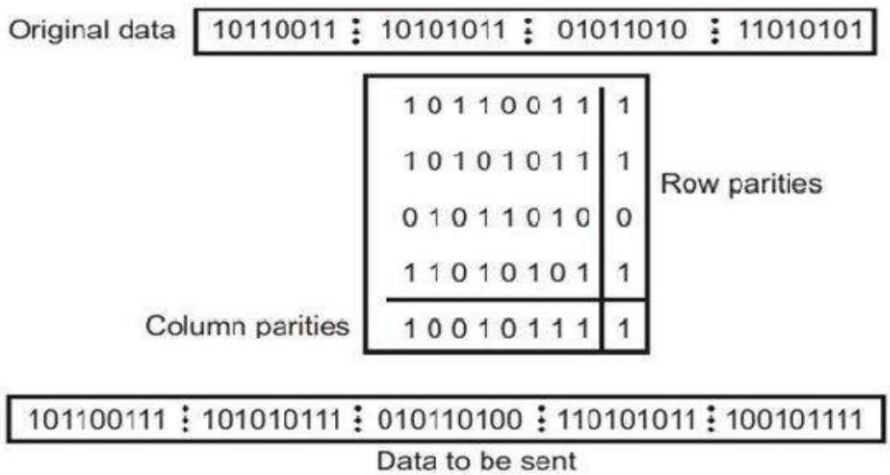
- **Example:**




- **Performance of simple parity check:**
  - Simple parity check can detect all **single-bit error** (only one bit of given data unit is changed from 1 to 0 or from 0 to 1).
  - It can also detect burst error (more than one bit gets corrupted)**.**
  - The technique is not foolproof against burst errors that **invert more than one bit**. If an even number of bits is inverted due to error, the **error is not detected.**

### *4.1.2. Two-dimension parity checking*

- Performance can be improved by using two dimensional parity check, which organizes the block of bits in the form of table.
- Parity check bits are calculated from each row, which is equivalent to a simple parity check.
- Parity check bits are also calculated for all columns.
- Both are sent along with the data.
- At the receiving end these are compared with the parity bits calculated on the received data.
  - **Example:**



- **Performance:**
  - If two bits in one data unit are damaged and two bits in exactly same position in another data unit are also damaged, The 2-D Parity check checker will not detect an error.
  - For example, if two data units: 11001100 and 10101100. If first and second from last bits in each of them is changed, making the data units as 01001110 and 00101110, the error cannot be detected by 2-D Parity check.

### *4.1.3. 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.

### *4.1.4. Cyclic Redundancy Code (CRC)*

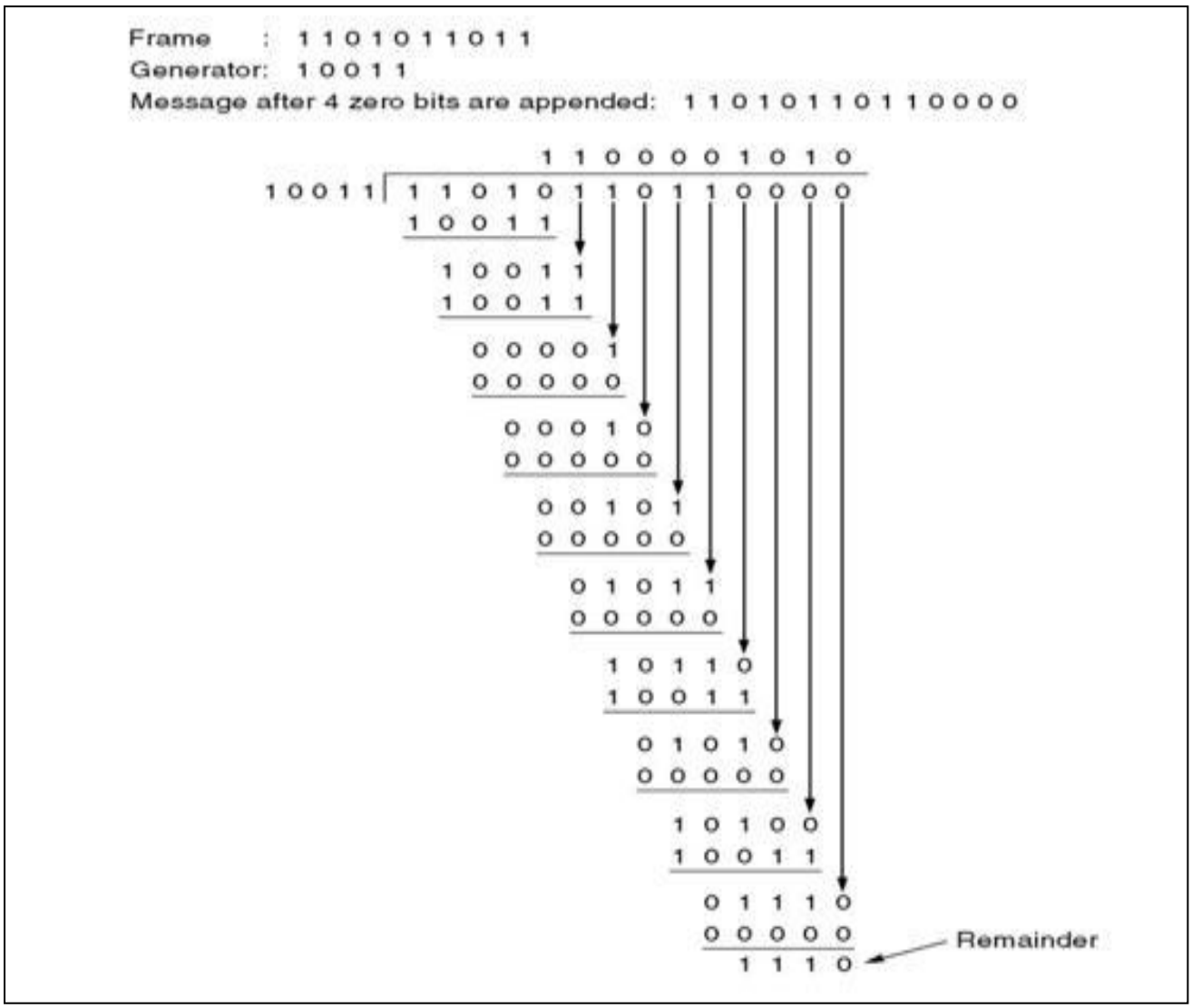
- This is one of the most widely used error detection method. In this method, the 'm' bit frame is regarded as a coefficient list for a polynomial with **'m'** terms from $\mathbf{x^{m-l}}$ to $\mathbf{x^0}$.
- Both the sender and the receiver have to agree upon a generator polynomial **G (x)** in advance.
- A check-sum is appended to the m bit frame such that the check-summed frame is divisible by G(x).
- When the receiver gets the check-summed frame, it divides all received bits by G(x).
- If the remainder is 0, there were no transmission errors. If there is a remainder, there is an error.
- **The checksum is calculated as follows:**
  1. Let **r** be the degree of **G(x).** Append 'r' zero bits to the end of 'm' message bits. This represents polynomial $\mathbf{x^r M(x).}$
  2. Divide the bit stream corresponding to $\mathbf{x^r M(x)}$ by **G(x)** using **modulo 2 division**.
  3. Subtract the remainder having 'r' bits from the bit stream of $\mathbf{x^r M(x)}$ using **modulo 2 subtraction**. The result is the check summed frame to be transmitted: **T(x).**
     - **Example:** Consider message 1101011011 and $G(x)=x^4+x+1$

       Generator = 10011.

       Since the degree of G(x) = r = 4, append four zeros to the message.

       Message after appending = 11010110110000.

       Now, we divide this stream by the generator.




Now subtract the remainder from **m+r**:

1 1 0 1 0 1 1 0 1 1 0 0 0 0

\- 1 1 1 0

= 1 1 0 1 0 1 1 0 1 1 1 1 1 0

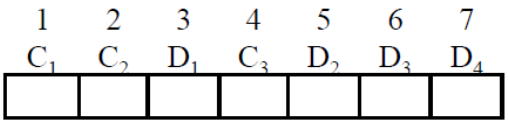
Thus, the transmitted frame is:

**11010110111110**

- When the receiver receives this frame, it divides the frame (using **modulo 2 division**) by the same polynomial **G(x)**. If the remainder is 0, there were no transmission errors.
- **Performance of CRC:**
  - 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 of less than the degree of the polynomial.
  - CRC detects most of the larger burst errors with a high probability.

## 4.2. Error correcting codes (Hamming code)

- **Hamming Distance:**
  - Hamming distance is used for error detection as well as correction.
  - Hamming distance is the number of bits that need to change to convert one valid codeword into another.
  - For example, Consider two valid codewords 100 and 111. Here, two bits need to change for one codeword to be converted to another during transmission.
  - But if one bit changes, the receiver will know that it's an error. Hence, 1 bit error can be detected.
  - To detect ‘n’ bit errors in transmission, a hamming code with distance n+1 is needed and to correct 'n' bit errors, hamming distance of 2n + 1 is needed.
  - For example, to detect 1 bit errors, the hamming distance should be 2 and to correct 1 bit errors, the hamming distance should be at least 3.
  - To calculate the hamming distance, XOR the two codewords and count the number of 1's in the result. For example, Hamming distance between (110011, 101010) is 3 since XOR(110011, 101010) = 011001.
- **To generate the Hamming code:**
  - For a message with ‘m' bits, 'r' additional hamming bits or check bits are inserted such that $(m + r + 1) <= 2^r$.
  - Therefore for a 4-bit message, 3 check bits are required.
  - These check bits are placed at positions that are powers of 2 in the codeword (i.e. 1, 2, 4 etc.) The remaining are data bits. For n = 7, (m = 4 and r = 3), the positions are:




where $C_1$, $C_2$, and $C_3$ are check bits and $D_1$ to $D_4$ are data bits.

The check bits are calculated such that the check bit along with some data bits has even (or odd) parity.

  - $C_1$ is calculated from bit position 1, 3, 5, 7, 9, i.e., position numbers having bit l in their LSB (Least Significant Bit).
  - $C_2$ is calculated from 2, 3, 6, 7, 10, i.e., having bit 1 in the second place from LSB.
  - $C_3$ is calculated from 4, 5, 6, 7, ... , i.e., having bit l in the third place from LSB.
    - **Example:** Calculate the Hamming bits for data message 1001.
      - Here, m= 4. Thus, r = 3, since $(m + r + 1) <= 2^r$ i.e. $4 + 3+1 <=2^3$.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| $C_1$ | $C_2$ | $D_1$ | $C_3$ | $D_2$ | $D_3$ | $D_4$ |
| | | 1 | | 0 | 0 | 1 |

- We will now calculate the check bits $C_1$, $C_2$ and $C_3$
- $C_1$ = even parity on bits (1, 3, 5, 7) = even parity on bits ($C_1$, 1, 0, 1) ∴ $C_1$ = 0
- $C_2$ = even parity on bits (2, 3, 6, 7) = even parity on bits ($C_2$, 1, 0, 1) ∴ $C_2$ = 0
- $C_3$ = even parity on bits (4, 5, 6, 7) = even parity on bits ($C_3$, 0, 0, 1) ∴ $C_3$ = 1
- Thus, the transmitted message will be :

| $C_1$ | $C_2$ | $D_1$ | $C_3$ | $D_2$ | $D_3$ | $D_4$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 1 |

- When the codeword arrives at the receiver, it calculates each check bit.
- If the parity is correct, it accepts the codeword.
- If it is incorrect, it adds the positions of the wrong checkbits to give the position of the erroneous bit.
- For example, if the message arrives as 0011011:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| $C_1$ | $C_2$ | $D_1$ | $C_3$ | $D_2$ | $D_3$ | $D_4$ |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |

- $D_3$ has changed during transmission. Now, $C_2$ and $C_3$ will be incorrect. Their sum is 6 (position 2 and 4). This means that the 6th bit, i.e., $D_3$ has changed.

## 5. Elementary Data Link Protocols

The services provided by the data link layer are implemented by protocols. Certain assumptions are made by these protocols:

1. We assume that the physical layer, data link layer and network layer are independent processes and communicate by passing messages back and forth.
2. Machine A wants to send a long stream of data to machine B using a reliable, connection oriented service.
3. A is assumed to have infinite supply of data and never has to wait for data to be produced.
4. For the data link layer, every bit coming from the network layer has to be delivered to the destinations network layer. The fact that the packet also contains a header is of no concern to the data link layer.
5. We assume that there exist suitable procedures to get data and send data to and from the network and physical layer.

6. The transmitting hardware computes the checksum and adds it to the data. The data link layer need not worry about it.
7. Initially the receiver has nothing to do. It is just waiting for something to happen. Events could be Frame Arrival, Error or Time-Out.
8. When a frame arrives, the hardware computes the checksum. If there is no error, the data link layer is informed. It then takes the frame, inspects the header and if it is correct, passes only the packet portion to the network layer but not the header.

These protocols are classified as follows:

- For noiseless channel:
  1. Simplex
  2. Stop-and-wait
- For noisy channel:
  1. Stop-and-wait ARQ
  2. Go back n ARQ
  3. Selective repeat ARQ

## 5.1. Simplex Protocol

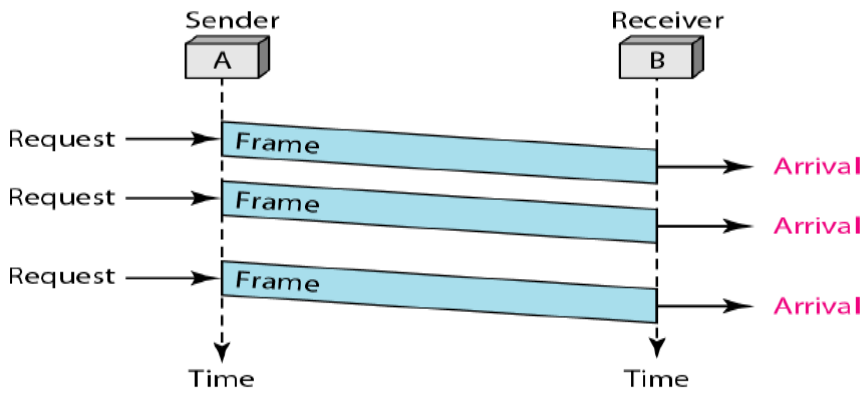
This is the simplest protocol, thoroughly unrealistic, here the assumptions made are:

1. Data transmission is simplex, i.e., in one direction only.
2. Both the transmitting and receiving network layers are always ready.
3. Processing time can be ignored.
4. Infinite buffer space is available.
5. The communication channel is ideal, i.e., it never loses or damages frames.

In this protocol, there are two distinct procedures:

1. A sender, which runs in the data link layer of the source machine.
2. A receiver which runs in the data link layer of the destination machine.

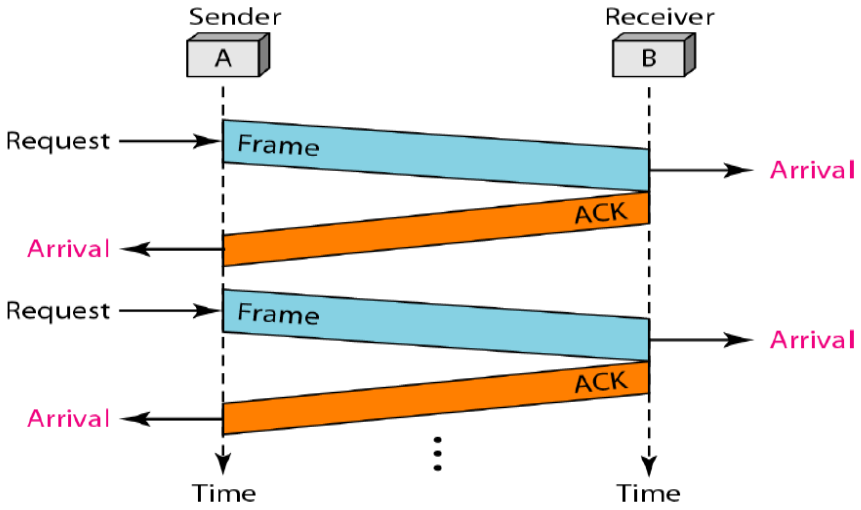
The following diagram shows the communication between the sender and receiver.




**Simplex Protocol Operation**

### 5.2.Stop-and-Wait Protocol

- In this protocol, we drop the unrealistic assumption that the receiving network layer can process incoming data infinitely fast.
- In such a case, the receiver must have some buffer space to store the frames. However, this space is limited. If more frames are sent, they will be discarded.
- The sender should be prevented from transmitting the next frame before the receiver sends an acknowledgement for the received frame.
- The sender transmits the next frame only after it has received an acknowledgement. Hence it is called Stop and Wait protocol.
- In this case, the communication channel has to be capable of bi-directional information transfer. But since the data frame and acknowledgement frame do not have to travel simultaneously, a half-duplex physical channel would be sufficient. The operation of this protocol is shown below:




**Stop and Wait Protocol operation**

The problem with this method is that the sender has to wait till the receiver sends back an acknowledgement. Hence, the bandwidth utilization is below optimum.

- **Note:** The above two protocols are for a noiseless channel which is unrealistic. Hence, these protocols are not implemented. The protocols defined below are for a Noisy channel.

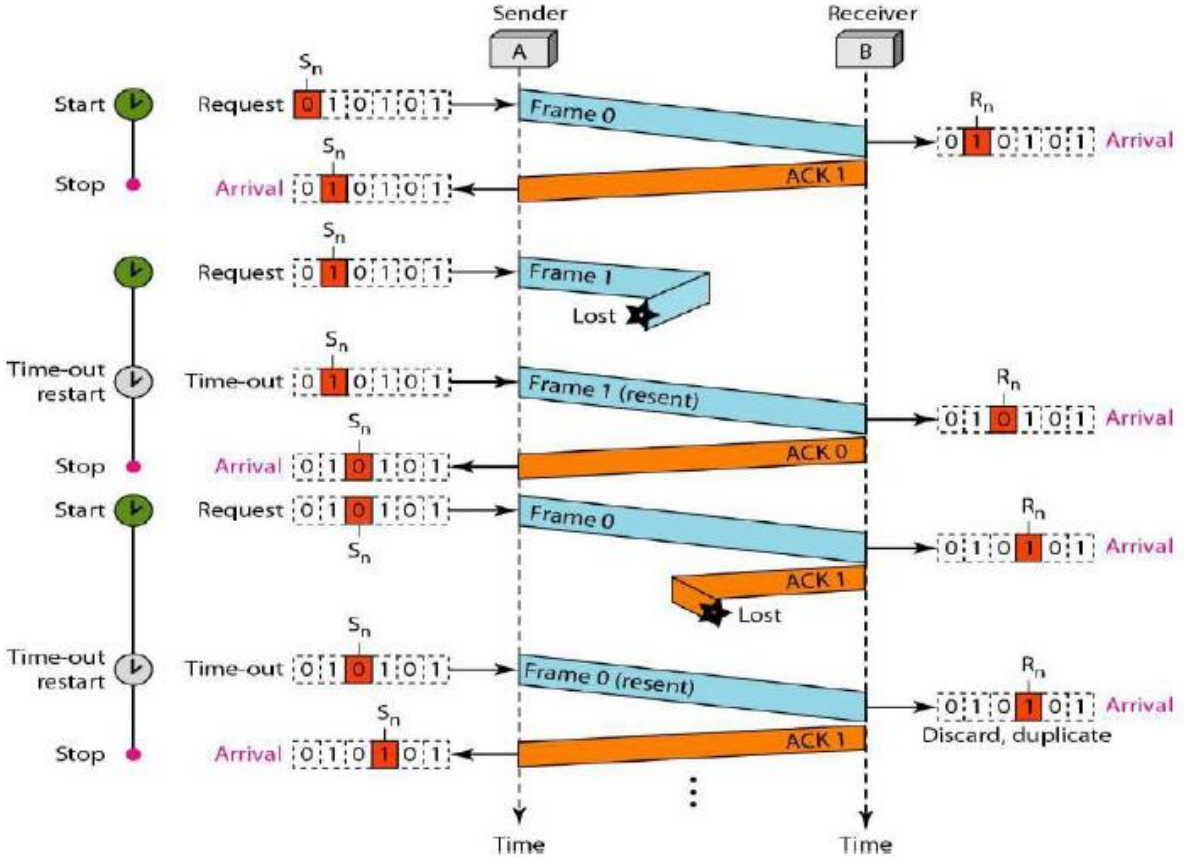
### 5.3.Simple Protocol For Noisy Channel

This protocol is also called PAR (Positive Acknowledgement with Retransmission) or Stop and Wait ARQ (Automatic Repeat reQuest).

In this protocol, we will drop the assumption that the communication channel is ideal. We will consider the situation of a communication channel that makes errors. Frames may either get damaged or lost.

- A simple scheme would be that a sender sends a frame.
- If it arrives properly at the receiver, it will send an acknowledgement, otherwise not.
- If the sender does not receive the acknowledgement, it will time-out and retransmit the frame.
- However, this scheme could pass duplicate frames to the receiver.
- This will happen if the receiver receives a frame, sends an acknowledgement and it gets lost.
- The sender retransmits the frame and the receiver accepts it because it does not know it is a duplicate frame.
- This problem can be avoided by using sequence numbers for every frame.
- A one-bit sequence number (0 or 1) is enough since at any given instant of time, we will be dealing with only two frames: the frame sent and the next frame to send.
- Initially, the sender sends frame 0.
- The receiver expects frame 0.
- If it arrives correctly, the receiver increments its frame expected number to 1.
- When the sender receives acknowledgement for frame 0, the sender increments next frame to send to 1.
- In modulo2 arithmetic, the sequence number will be incremented from 0101 and so on.
- **Algorithm Sender:**
    1. The sender remembers the sequence number of next frame to be sent in **Sn**.
    2. After transmitting a frame, the sender starts the timer. (Time interval is chosen to allow enough time for the frame to travel, the receiver to process it and the acknowledgement to come back).
    3. If the timer times out, the sender sends a duplicate frame.
    4. If a valid acknowledgement comes in, the sender clears the buffer, advances the sequence number and sends another frame if there is data from the network layer.
- **Algorithm Receiver:**
    1. The receiver remembers the number of the expected Frame in **Rn**.
    2. The receiver waits for a frame to arrive.
    3. If an undamaged frame arrives and it is the expected frame, the receiver receives the frame, sends an acknowledgement and increments **Rn**.
    4. If it is a duplicate frame, i.e., **Rn** is not equal to the sequence number, the receiver simply sends an acknowledgement of the previous received frame.
    5. If the frame arrives with an error, the receiver goes into the sleep state waiting for an event to occur.

- **Example:** The figure bellow show an example of Stop-and-Wait ARQ exchange :




**An example of Stop-and-Wait ARQ**

➢ **Event:**

- Frame 0 is sent and acknowledged.
- Frame 1 is lost and resent after the time-out.
- The resent frame 1 is acknowledged and the timer stops.
- Frame 0 is sent and acknowledged, but the acknowledgment is lost.
- The sender has no idea if the frame or the acknowledgment is lost.
- So after the time-out, it resends frame 0, which is acknowledged.

## 5.4. Sliding Window Protocols

In the protocols studied so far, data transfer was simplex, i.e. in one direction only. However, in practical situation full duplex transmission is required, i.e., the two communicating machines send data simultaneously.

### *5.4.1. Concept of SlidingWindow Protocol*

- In a sliding window protocol, the data link layer of the sender and receiver maintains a set of sequence numbers corresponding to frames it is permitted to send and receive respectively.
- These lists are called 'sending window' and 'receiving window' respectively.

- The sizes and limits of the windows need not be the same in the sender and receiver.
- All protocols belonging to this category have common features listed below:
  - Each outbound frame contains a sequence number ranging from 0 to some maximum (usually $2^n$-1, so n bit field can be used).
  - At any instant of time, the sender and receiver maintain a sending and receiving window respectively.
  - The communication channel must deliver frames in the order that they were sent.
  - The sender's window represents frames sent but not acknowledged. A new frame is given the next highest sequence number and the upper edge of the window is advanced by one. When an acknowledgement arrives, the lower edge is advanced by one.
  - Since the sent frames may be lost or damaged, the sender has to store them in buffers. If the maximum window size is 'n', 'n' buffers are required.
  - The receiver's window corresponds to the frames it may accept. Any frame falling outside the window is discarded. When a frame whose number is equal to the low edge arrives, an acknowledgement is generated and the window is rotated by one. If the window size is one, the receiver only accepts frames in order. For larger size window, this is not true but the network layer is always given data packets in order.

### *5.4.2. One Bit Sliding Window Protocol*

This is also called stop-and-wait one bit sliding window protocol. In this protocol:

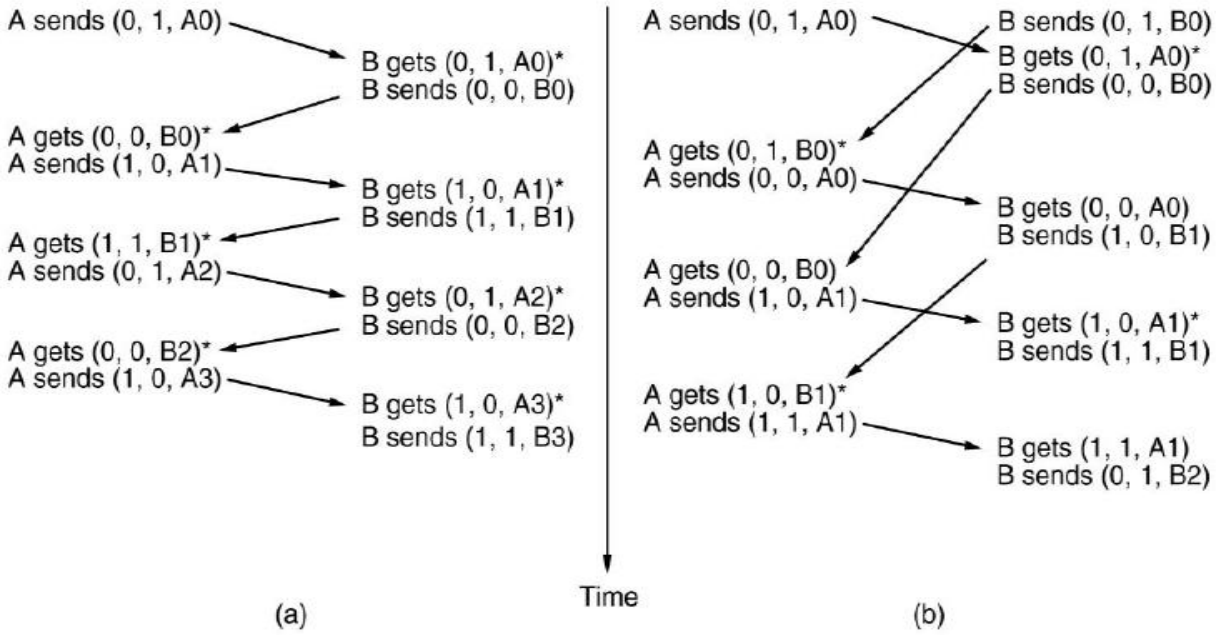
1. The size of the sending and receiving window is 1, i.e., the sender can send only one frame at a time and the receiver can receive only one frame at a time.
2. The sender transmits a frame and waits for an acknowledgement before sending the next one.
3. Since only one frame is sent or received at a time, a one-bit sequence number is enough. We can use only two sequence numbers 0 and 1.

- **Mechanism:**
  - The sender initializes sequence-number to 0. The sender builds a frame with sequence number 0 and sends it. The receiver expects frame 0.
  - When the receiver receives a frame, it checks to see if it is a duplicate one. If it is the expected frame, it is accepted, an acknowledgement is sent and the receiver window is slide up, i.e., it expects frame 1.

- The acknowledgement field contains the number of the last frame received without error. If the sender receives an acknowledgement with correct sequence number, it slides up its sequence number to 1 and sends a new frame.
- If they do not match, it has to resend the same frame.
- The notation is (Sequence, acknowledgement, packet).

This protocol ensures that no duplicate frames will be delivered to the network layer if one transmits before the other.

- **Peculiar Situation:**
  - If both, A and B send simultaneously, a peculiar situation arises. There will be a synchronization problem if both A and B transmit simultaneously.
  - This causes duplicate frames to be delivered even though there is no transmission error.
  - Figure (b) shows that duplicate frames will be delivered since A and B sends at the same time.
  - The * indicates where a frame is accepted.




**Two scenarios for one bit sliding window protocol**

**(a) Normal Operation (b) Duplicate frames delivered**

### *5.4.3. Pipeline Technique*

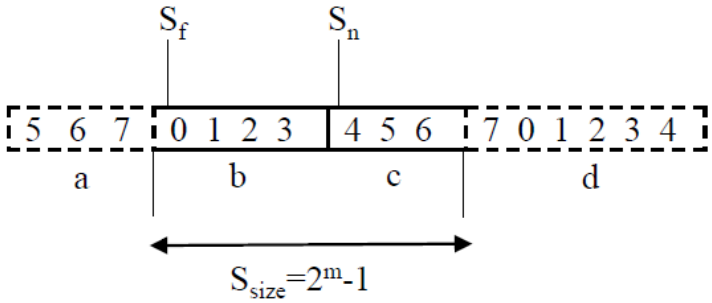
- In the previous protocol, the sender had to wait till it received an acknowledgement before sending the next frame.
- In order to achieve better efficiency, the sender can transmit 'w' frames instead of 1 such that by the time it has transmitted the w frame, the acknowledgement for frame 1 arrives, thus eliminating the sender's waiting period.
- Thus, 'w' unacknowledged frames are outstanding at all times. The sender's window size is thus 'w'.
- This technique is used when the propagation delay is large and the line cannot be kept idle during this time. Thus, more frames can be sent at a time to utilize the channel bandwidth efficiently.
- Since multiple frames are transmitted at the same time, two strategies are used to deal with errors in the pipelining technique:
    1. Go back n ARQ
    2. Selective repeat ARQ

### *5.2.5. Go back N Automatic Repeat Request*

- In this strategy, pipelining technique is used. If a frame gets lost or corrupted in transit, the receiver simply discards the damaged frame and all subsequent ones, sending no acknowledgement.
- It accepts frames only in order.
- The sender will eventually time out and retransmits all unacknowledged frames.
- **Sequence Numbers**: Since each frame must be identified, it is assigned a sequence number. If ‘m' bits are allocated in the frame header for sequence number, sequence numbers are from 0 to $2^m$-1. For example, if m = 3, the sequence numbers are:
  0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 ....
- **Sender Sliding Window:** The sender and receiver maintain a sliding window of frame numbers. The sender sliding window is an abstract concept or box having size $2^m$-1. It has 4 parts:
    a. Frames numbers which have been sent and acknowledged.
    b. Frames which have been sent and not yet acknowledged.
    c. Frames within the window size which can be sent.
    d. Frames falling out of the window.

  For example, consider m = 3. Frame numbers can be 0 - 7.

  The window size = 7. The four parts of the sender window are shown below.




**Sender sliding window**

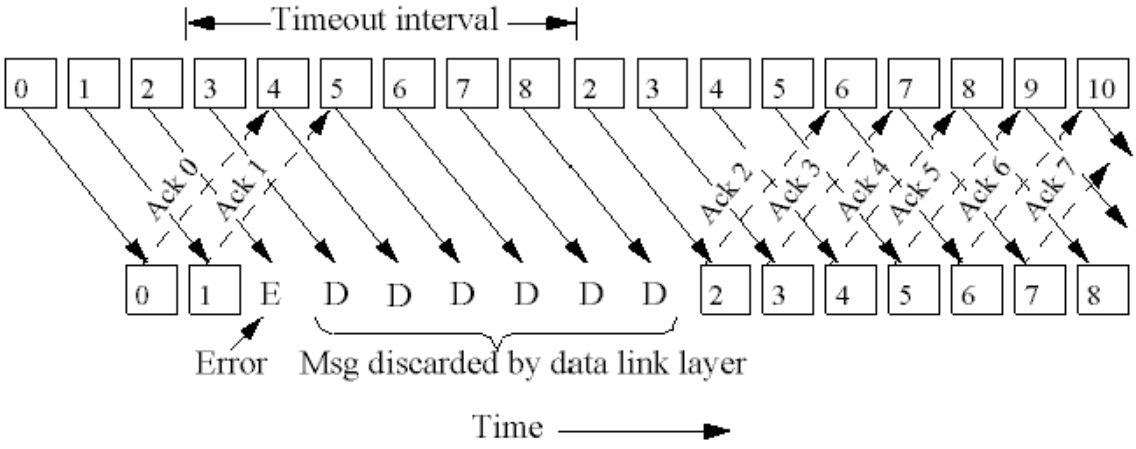
Here,

$S_f$ = first outstanding frame

$S_n$ = next outstanding frame

$S_{size}$ = size of the sending window

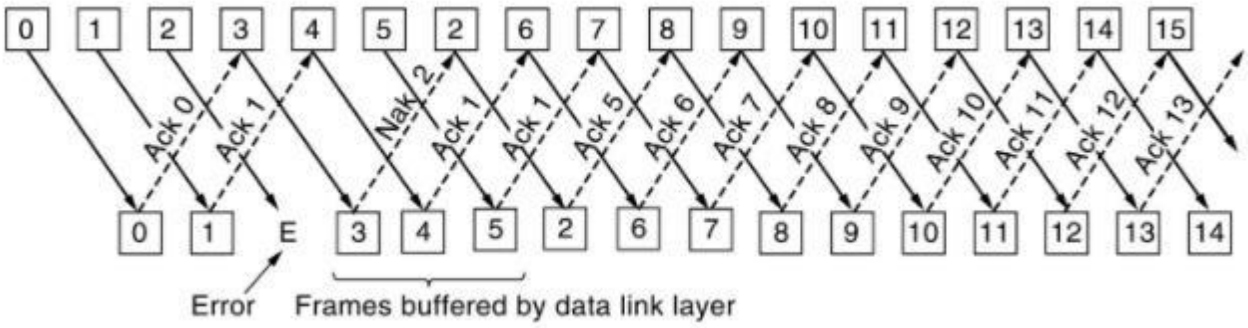
- **Receiver Sliding Window:** The receiver only accepts frames in order. Hence, the receiver window size is 1. It stores the expected frame number in Rn. When the expected frame arrives, Rn is incremented.
- **Method:**
  - The sender sends multiple frames and maintains a timer for each sent frame.
  - The receiver sends a separate acknowledgement for each correctly received frame. If it receives a frame with error, the receiver discards the damaged frame and all subsequent frames.
  - When the sender does not receive an acknowledgement for a frame, the timer times out. The sender then retransmits all the outstanding frames.
  - The Go-back-N strategy is illustrated in the next figure. Here, frame 2 is arrived with an error, hence it is discarded. The receiver also discards all subsequent frames 3-8. The sender retransmits frames 2-8.




**Go back n strategy**

### *5.2.6. Selective Repeat Automatic Repeat Request*

- In Go-back-N, the receiver window size is 1 and it only accepts frame in order. This causes a lot of retransmissions at the sender.
- In selective repeat ARO, the receiver does not discard but buffers all frames following a damaged or lost frame.
- The sender only resends the damaged or lost frame. This is more efficient that Go-back-n.
- **Window Sizes:**
  - The sender and receiver maintain a sliding window of frame numbers. The sender and receiver window sizes are the same.
  - The receiver window corresponds to the frame numbers which the receiver can accept and buffer.
- **Operation:**
  - The sender sends multiple frames corresponding to its sending window.
  - The receiver window corresponds to frame numbers which it can accept.
  - If a frame arrives damaged or is lost, the receiver sends a **NAK** (negative acknowledgement) for that frame and stores all subsequent frames.
  - When the sender receives the NAK, it just **resends that frame**. The following diagram shows the operation of Selective Repeat ARQ.




**Selective Repeat ARQ protocol**

- **Restriction on Sender and Receiver window size:**
  - The **window sizes** in Selective Repeat ARQ is maximum $\mathbf{2^m/2}$ **(or** $\mathbf{2^{m-1}}$**).**
  - This restriction is **to prevent the receiver from accepting duplicate frames**.
  - When the receiver slides its window after accepting frames, the new frame numbers must not be identical to the frame numbers before sliding the window.

# 6. Wired LANs: Ethernet

Although a LAN can be used as an isolated network to connect computers in an organization for the sole purpose of sharing resources, most LANs today are also linked to a wide area network (WAN) or the Internet.

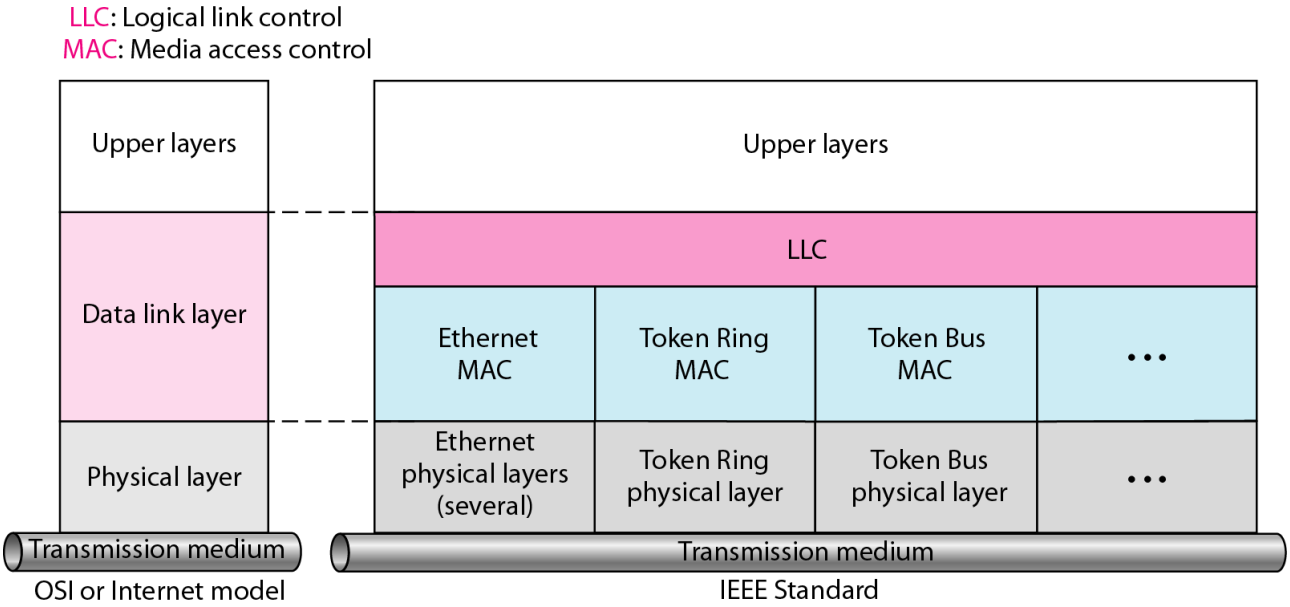
The LAN market has several technologies such as Ethernet, Token Ring, Token Bus, FDDI, and ATM LAN. Some of these technologies survived for a while, but Ethernet is by far the dominant technology.

## 6.1.IEEE Standards

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 the figure bellow.

The IEEE has subdivided the data link layer into two sublayers: logical link control (LLC) and media access control (MAC). IEEE has also created several physical layer standards for different LAN protocols.




**IEEE standard for LANs**

- **Data Link Layer:** As we mentioned before, the data link layer in the IEEE standard is divided into two sublayers: LLC and MAC.
  - **Logical Link Control (LLC):** In IEEE Project 802, flow control, error control, and part of the framing duties are collected into one sublayer called the logical link control. Framing is handled in both the LLC sublayer and the MAC sublayer.

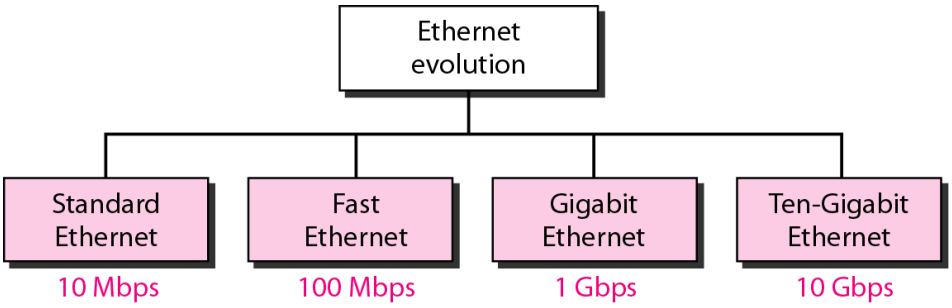
    The LLC provides one single data link control protocol for all IEEE LANs. In this way, the LLC is different from the media access control sublayer MAC, which provides different protocols for different LANs.

    The purpose of the LLC is to provide flow and error control for the upper-layer protocols that actually demand these services.
  - **Media Access Control (MAC):** IEEE Project 802 has created a sublayer called media access control that defines the specific access method for each LAN. For example, it defines CSMA/CD as the media access method for Ethernet LANs and the token passing method for Token Ring and Token Bus LANs. Part of the framing function is also handled by the MAC layer.

    In contrast to the LLC sublayer, the MAC sublayer contains a number of distinct modules; each defines the access method and the framing format specific to the corresponding LAN protocol.
- **Physical Layer:** The physical layer is dependent on the implementation and type of physical media used. IEEE defines detailed specifications for each LAN implementation. For example, although there is only one MAC sublayer for Standard Ethernet, there is a different physical layer specifications for each Ethernet implementations.

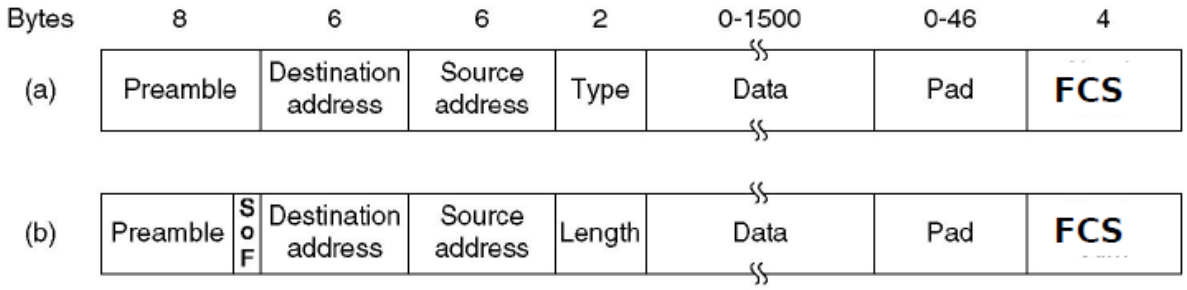
## 6.2.Standard ETHERNET

The original Ethernet was created in 1976 at Xerox's Palo Alto Research Center (PARC). Since then, it has gone through four generations. We briefly discuss the Standard (or traditional) Ethernet in this section.




**Ethernet evolution through four generations**

### *6.2.1. Ethernet Frame format*

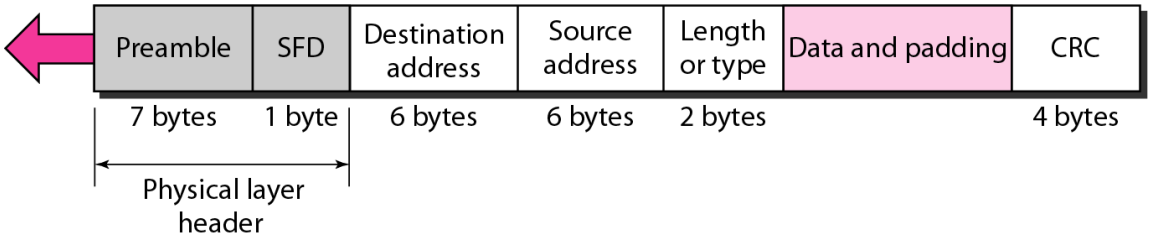



**Frame formats. (a) DIX Ethernet , (b) IEEE 802.3**

- **802.3 MAC frame format:**

Preamble: 56 bits of alternating 1s and 0s.

SFD: Start frame delimiter, flag (10101011)




**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 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

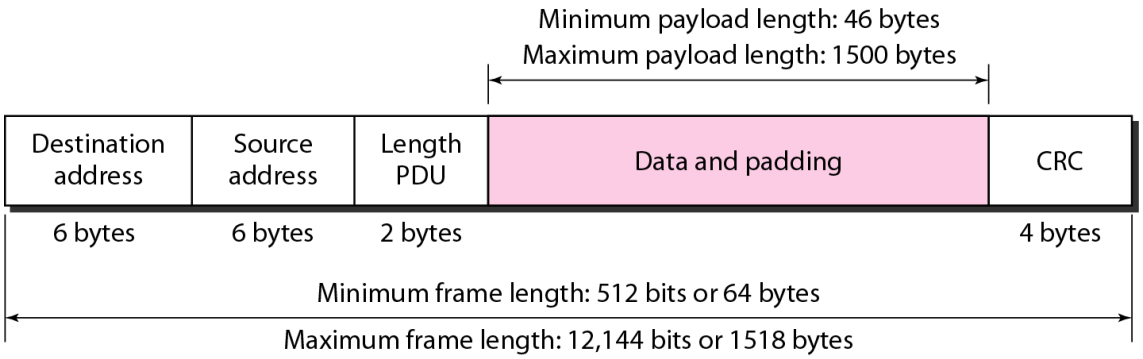
MAC frame. The IEEE standard used it as the length field to define the number of bytes in the data field. Both uses are common today.

- **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.

- **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.

  - Minimum Frame length:: 64 bytes (512 bits)
  - Maximum Frame length:: 1518 bytes (12,144 bits)




### *6.2.2. Addressing*

Each station on an Ethernet network (such as a PC, workstation, or printer) has its own network interface card (NIC). The NIC fits inside the station and provides the station with a 6-byte physical address (MAC address). Primarily specified as a unique identifier during device manufacturing, the MAC address is often found on a device's NIC. The Ethernet address (MAC address) is 6 bytes (48 bits), nonnally written in hexadecimal notation, with a colon between the bytes.

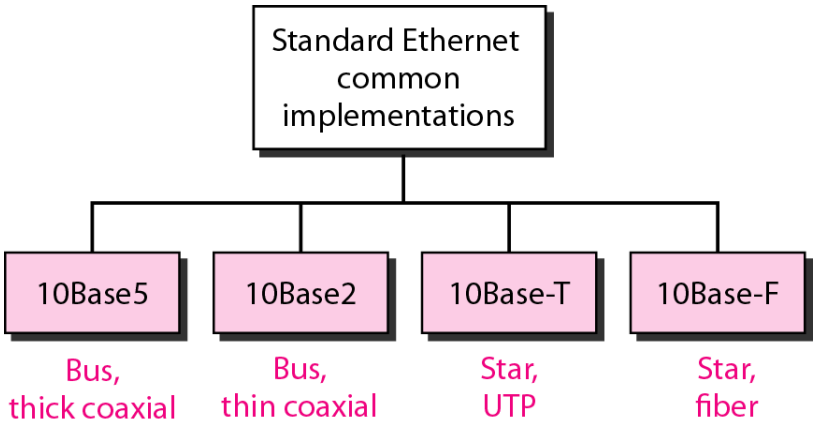

**Example of an Ethernet address in hexadecimal notation**

- **Unicast, Multicast, and Broadcast 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 source address is always a unicast address-the frame comes from only one station. The destination address, however, can be unicast, multicast, or broadcast. If the least significant bit of the first byte in a destination address is 0, the address is unicast; otherwise, it is multicast. The broadcast destination address is a special case of the multicast address in which all bits are 1s.

### *6.2.3. Physical Layer*

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




**Categories of Standard Ethernet**

The next table shows a summary of Standard Ethernet implementations:

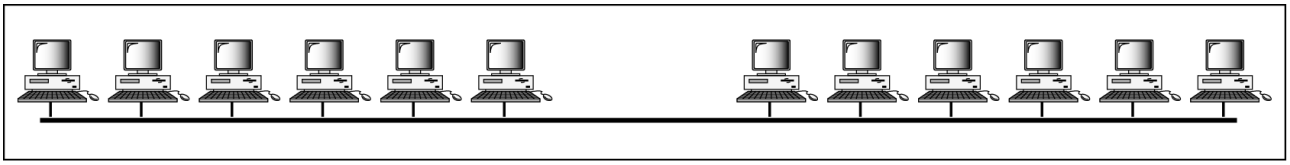
| *Characteristics* | *10Base5* | *10Base2* | *10Base-T* | *10Base-F* |
|---|---|---|---|---|
| Media | Thick coaxial cable | Thin coaxial cable | 2 UTP | 2 Fiber |
| Maximum length | 500 m | 185 m | 100 m | 2000 m |
| Line encoding | Manchester | Manchester | Manchester | Manchester |

## 6.3. Changes in the standard

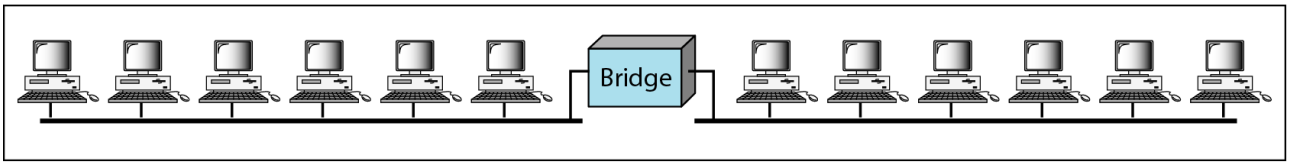
The 10-Mbps Standard Ethernet has gone through several changes before moving to the higher data rates. These changes actually opened the road to the evolution of the Ethernet to become compatible with other high-data-rate LANs.

### *6.3.1. Bridged Ethernet*

The first step in the Ethernet evolution was the division of a LAN by bridges. Bridges have two effects on an Ethernet LAN: They raise the bandwidth and they separate collision domains.



a. Without bridging

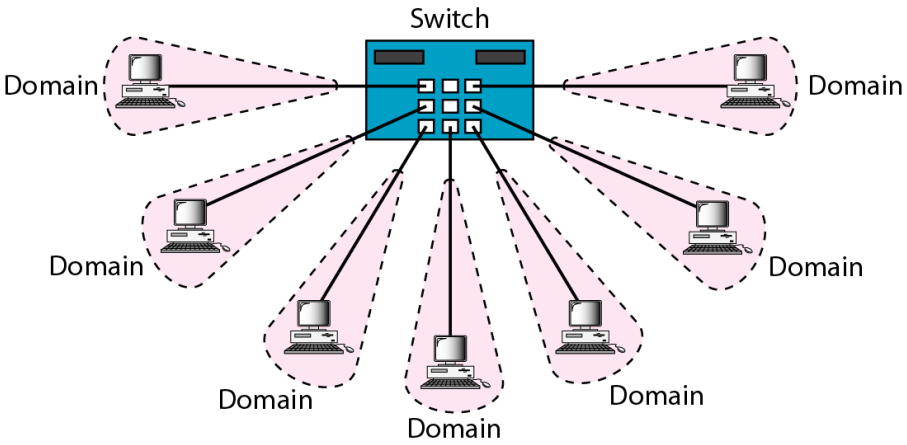



b. With bridging

**A network with and without a bridge**

### *6.3.2. Switched Ethernet*

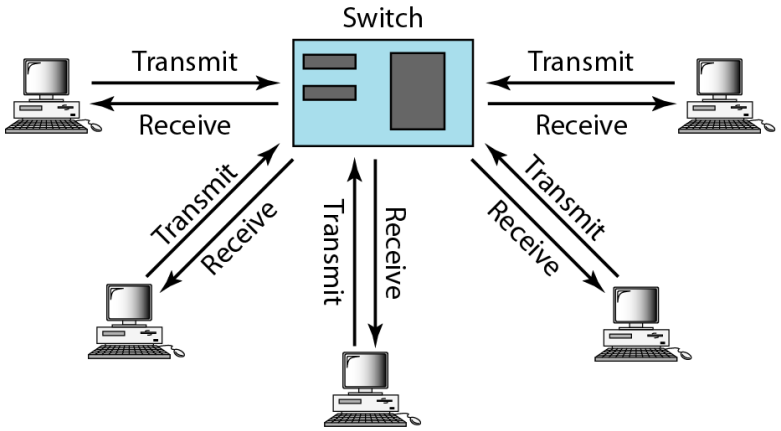
The idea of a bridged LAN can be extended to a switched LAN. Instead of having two to four networks, why not have N networks, where N is the number of stations on the LAN? In other words, if we can have a multiple-port bridge, why not have an N-port switch? In this way, the bandwidth is shared only between the station and the switch. In addition, the collision domain is divided into N domains.




**Switched Ethernet**

### *6.3.3. Full-Duplex Ethernet*

One of the limitations of 10Base5 and lOBase2 is that communication is half-duplex; a station can either send or receive, but may not do both at the same time. The next step in the evolution was to move from switched Ethernet to full-duplex switched Ethernet. Note that instead of using one link between the station and the switch, the configuration uses two links: one to transmit and one to receive.




**Full-duplex switched Ethernet**

## 7. The channel allocation problem

The channel allocation problem is how to allocate a single broadcast channel among competing users.

### 7.1.Collision domain

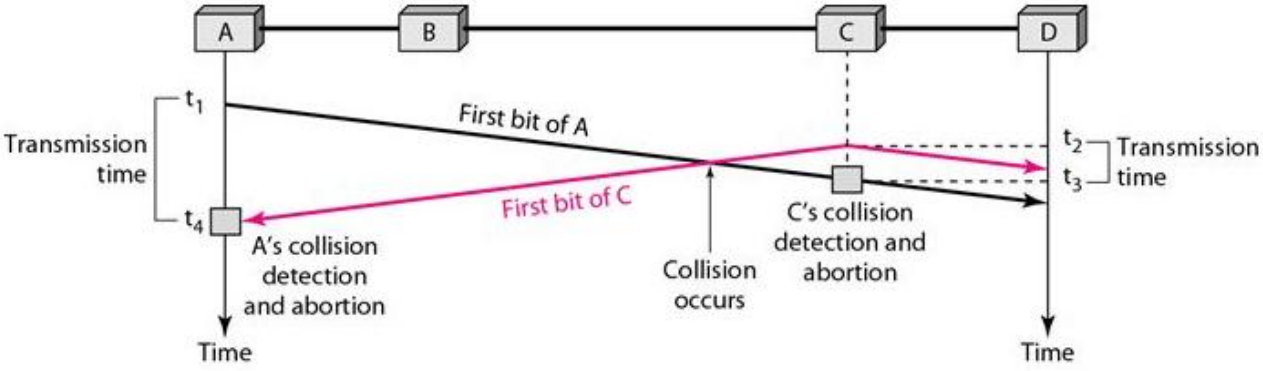
A collision domain is a network segment (connected by a shared medium or through repeaters) where simultaneous data transmissions collide with one another as a result of more than one device attempting to send a packet on the network segment at the same time. The collision domain applies particularly in wireless networks, but also affected early versions of Ethernet.

A channel access method dictates that only one device in the collision domain may transmit at any one time, and the other devices in the domain listen to the network and refrain from transmitting while others are already transmitting in order to avoid collisions.

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

- Carrier Sense Multiple Access (CSMA) is a probabilistic Media Access Control (MAC) protocol in which a node verifies the absence of other traffic before transmitting on a shared transmission medium, such as an electrical bus, or a band of the electromagnetic spectrum.
- "Carrier Sense" describes the fact that a transmitter uses feedback from a receiver that detects a carrier wave before trying to send. That is, it tries to detect the presence of an encoded signal from another station before attempting to transmit. If a carrier is sensed, the station waits for the transmission in progress to finish before initiating its own transmission.
- "Multiple Access" describes the fact that multiple stations send and receive on the medium. Transmissions by one node are generally received by all other stations using the medium.

- Carrier sense multiple access with collision detection (CSMA/CD) is a Media Access Control method in which:
  - A carrier sensing scheme is used.
  - A transmitting data station that detects another signal while transmitting a frame, stops transmitting that frame, transmits a jam signal, and then waits for a random time interval before trying to resend the frame.
  - CSMA/CD is a modification of pure carrier sense multiple access (CSMA). CSMA/CD is used to improve CSMA performance by terminating transmission as soon as a collision is detected, thus shortening the time required before a retry can be attempted.
  - The **jam signal** is a signal that carries a 32-bit binary pattern sent by a data station to inform the other stations that they must not transmit.




- **ALGORITHM:** The following procedure is used to initiate a transmission. The procedure is complete when the frame is transmitted successfully or a collision is detected during transmission.

  When a station wants to send some information, it uses the following algorithm.
  - **Main procedure:**

    1. Is a frame ready for transmission? If not, wait for a frame.
    2. Is medium idle? If not, wait until it becomes ready.
    3. Start transmitting and monitor for collision during transmission.
    4. Did a collision occur? If so, go to collision detected procedure.
    5. Reset retransmission counters and end frame transmission.
  - **Collision detected procedure:** The following procedure is used to resolve a detected collision. The procedure is complete when retransmission is initiated or the retransmission is aborted due to numerous collisions.

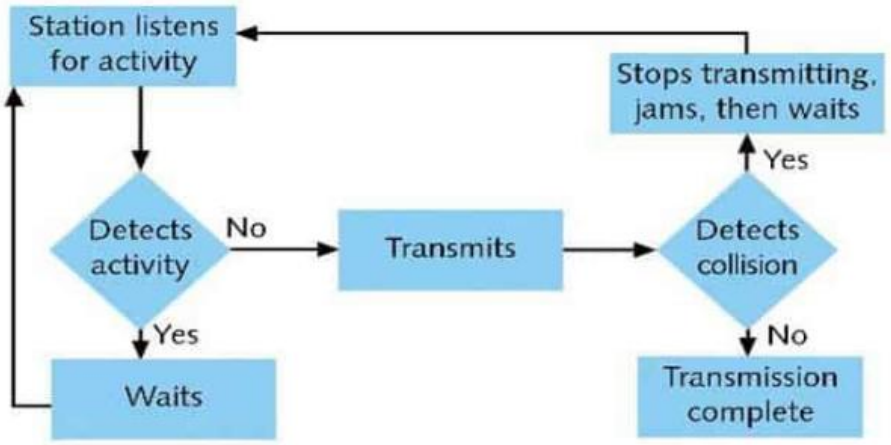
    1. Continue transmission until minimum packet time is reached to ensure that all receivers detect the collision.

2. Increment retransmission counter.

3. Was the maximum number of transmission attempts reached? If so, abort transmission.

4. Calculate and wait random backoff period based on number of collisions.

5. Re-enter main procedure at stage 1.




**CSMA/CD process**