# Chapter No: 4

# ISO OSI Reference Model

**Who made:**

- International Standards Organization (ISO)
- A Model of How Protocols and Networking Components Could be Made
- "*Open*" means the concepts are nonproprietary; can be used by anyone.
- OSI is not a protocol. It is a model for understanding and designing a network architecture that is flexible and robust

## Open Systems Interconnect (OSI) Model: -

- The OSI model describes how data flows from one computer, through a network to another computer
- The OSI model divides the tasks involved with moving information between networked computers into 7 smaller, more manageable sub-task.
- A task is then assigned to each of the seven OSI layers.
- Each layer is reasonably self-contained so that the tasks assigned to each layer can be implemented independently.

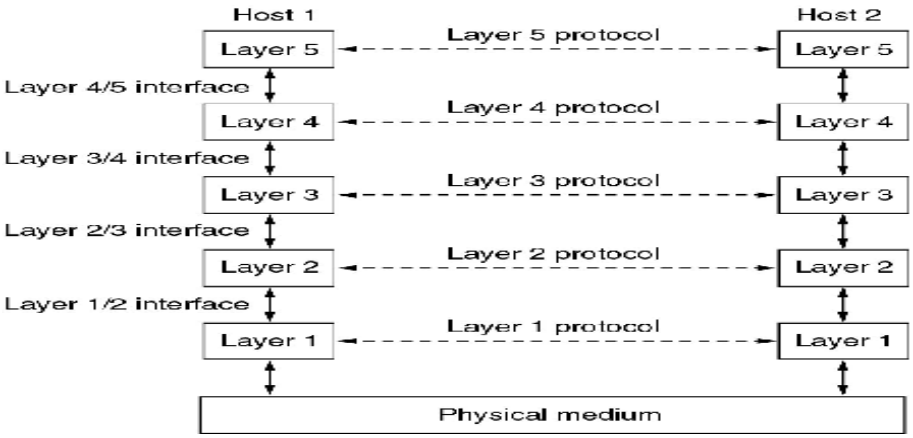
### Network Architecture

- A set of layers and protocols is called a network architecture
- It refers to the physical and logical design of a network

### Why Layered Architecture?

- Layer architecture simplifies the network design.
- It is easy to debug network applications in a layered architecture network.
- The network management is easier due to the layered architecture.
- Network layers follow a set of rules, called protocol.
- The protocol defines the format of the data being exchanged, and the control and timing for the handshake between layers.

### Layered Approach




The entities comprising the corresponding layers on different machines are called peers

- It is the peers that communicate by using the protocols
- Actually, data is not transferred from layer *n* on one machine to layer *n* on another machine
- Each layer passes data and control information to the layer immediately below it, until the lowest layer is reached
- Actual data communication takes place through the lowest layer – the physical layer

**Design Issues for the Layers**

- Addressing
- Error control
- Order of messages must be preserved
- Flow control – fast sender and slow receiver!
- Disassembling, transmitting, and reassembling large messages
- Multiplexing / de-multiplexing
- Routing

**Concept of Services and Protocols:-**

- A service is a set of operations that a layer provides to the layer above it
- Service defines what operations the layer is prepared to perform
- A service relates to the interface between two layers – the lower layer is service provider and the upper layer is service user

**Concept of Services and Protocols**

- A protocol is a set of rules governing the format and meaning of the packets
- Protocols relate to packets sent between peer entities on different machines
- Entities use protocols
- Protocols can be changed provided the services visible to the user do not change. Thus services and protocols are completely decoupled

**Services and Protocols**

- Analogy with programming languages
- A service is like an object in an object oriented language
- What operations can be performed on this object is defined
- How these operations are to be performed is not defined
- Protocol relates to the *implementation* of the service – how it is done

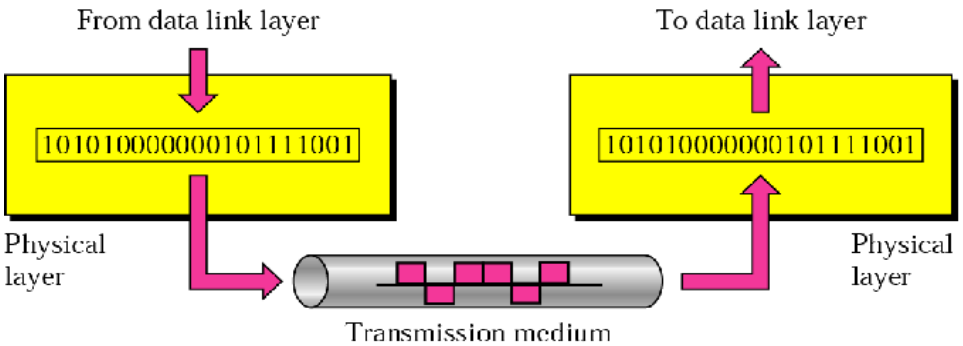
**The Layers of the OSI Model**


**Physical layer:-**

- Specifications for the physical components of the network.

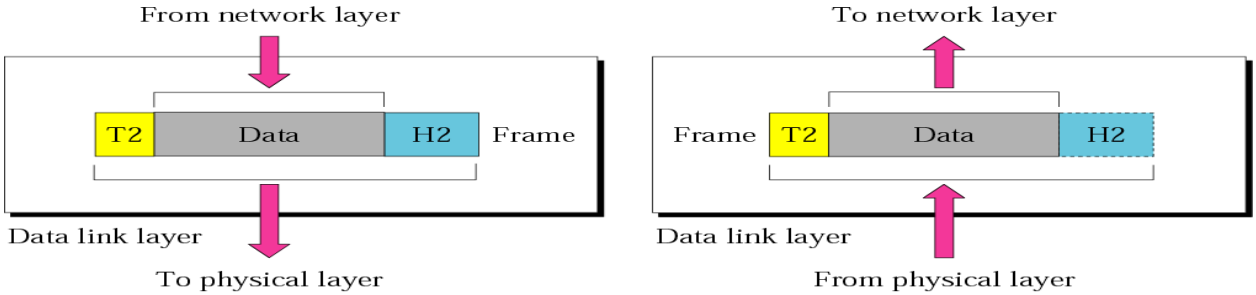
## Functions of Physical Layer:

- **Bit representation** – encode bits into electrical or optical signals
- **Transmission rate** – The number of bits sent each second
- Physical characteristics of transmission media
- **Synchronizing** the sender and receiver clocks
- **Transmission mode** – simplex, half-duplex, full duplex
- **Physical Topology** – how devices are connected – ring, star, mesh, bus topology



# Data Link Layer

- Data link layer attempts to provide reliable communication over the physical layer interface.
- **Breaks the outgoing data into frames** and **re-assemble the received frames**.
- Create and detect frame boundaries.
- **Handle errors** by implementing an acknowledgement and retransmission scheme.
- **Implement flow control**.



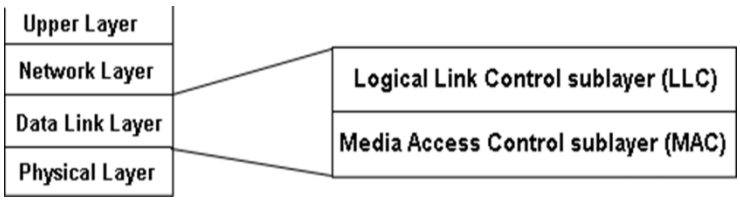

***The data link layer is responsible for moving frames from one hop (node) to the next.***

**Functions of Data Link Layer**

- **Framing-**
  - Divides the stream of bits into manageable data units called frames.
- **Physical addressing-**
  - Adds a header to the frame to define the sender and/or receiver of the frame.
- **Flow control-**
  - Imposes a flow control mechanism to avoid overwhelming the receiver. Synchronization between fast sender and slow receiver.
- **Error control-**
  - Adds mechanisms to detect and retransmit damaged or lost frames (CRC).
- **Access control-**
  - Determine which device has control over the link at any given time.
- **Link establishment and termination**:
  - Establishes and terminates the logical link between two nodes.
- **Frame sequencing**:
  - Transmits/receives frames sequentially.
- **Frame acknowledgment**:
  - Provides/expects frame acknowledgments.

**DLL is divided into two Sub-Layers**

- **LLC Sub Layer**
- **MAC Sub Layer**



**Logical Link Control Sub Layer**

- It is upper portion of the Data Link layer.
- Performs **Flow control** and **management of connection errors**.
- LLC supports three types of connections:

1. **Unacknowledged connectionless service**:
   - does not perform reliability checks or maintain a connection, very fast, most commonly used
2. **Connection oriented service**:
   - once the connection is established, blocks of data can be transferred between nodes until one of the node terminates the connection.
3. **Acknowledged connectionless service**:
   - provides a mechanism through which individual frames can be acknowledged.

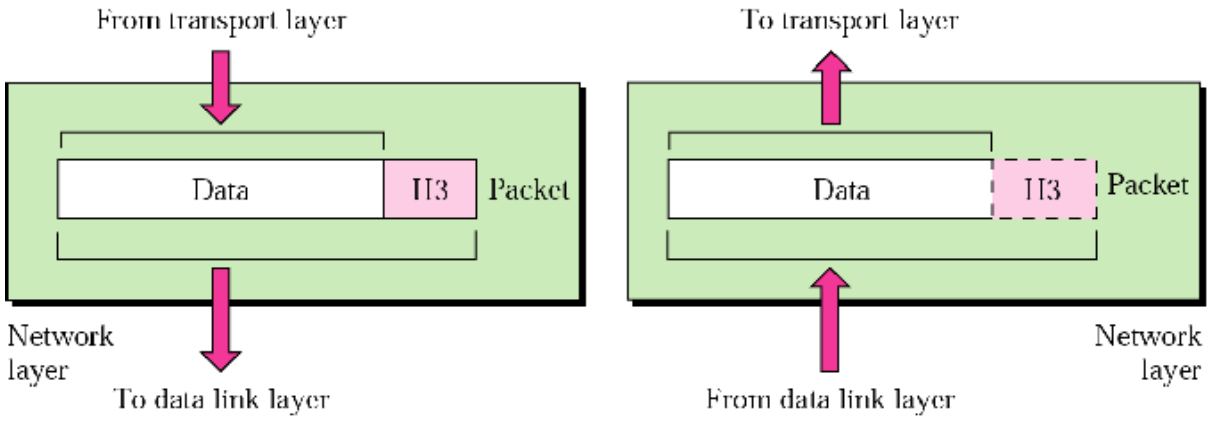
**Media Access Control Sub Layer**

- This sub layer contains methods to **regulate the timing** of data signals and **eliminate collisions**.
- The MAC sub layer determines where one frame of data ends and the next one starts - **frame synchronization.**
- There are four means of frame synchronization:
  - Time based,
  - Character counting,
  - Byte stuffing and
  - Bit stuffing.

**Network Layer:-**

**Main functions of this layer are:**

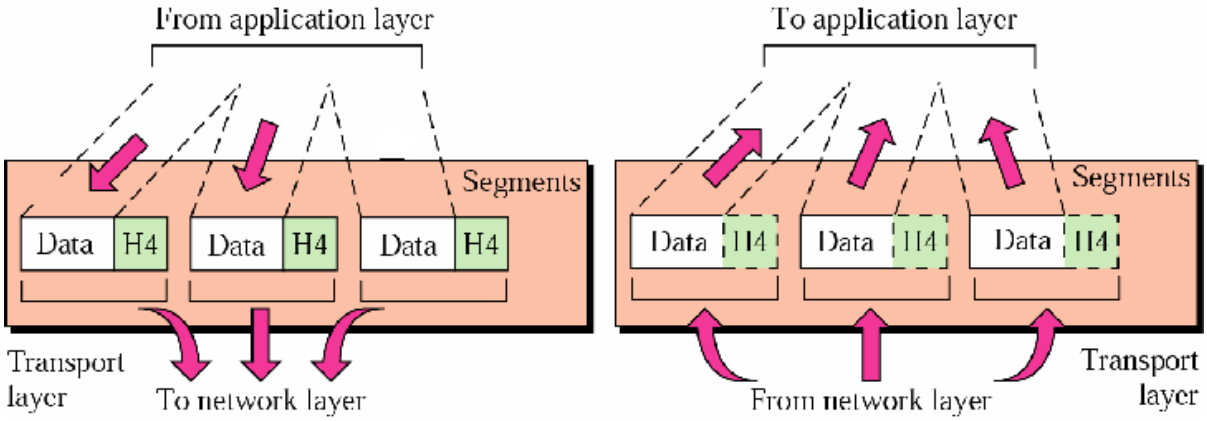
- Responsible for delivery of packets across multiple networks
- Routing – Provide mechanisms to transmit data over independent networks that are linked together.
- Network layer is responsible only for delivery of individual packets and it does not recognize any relationship between those packets



## Transport Layer:-

**Main functions of this layer are:**

- Responsible for source-to destination delivery of the entire message
- Segmentation and reassembly – divide message into smaller segments, number them and transmit. Reassemble these messages at the receiving end.
- Error control – make sure that the entire message arrives without errors – else retransmit.

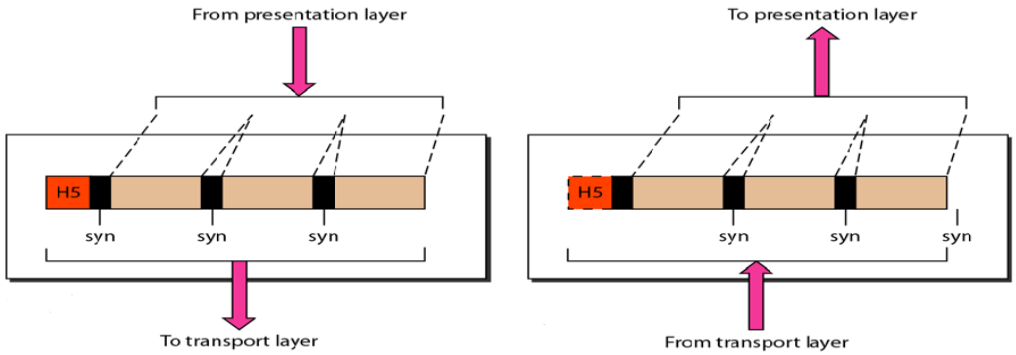



## Session Layer:-

**Main functions of this layer are:**

• **Dialog control** – allows two systems to enter into a dialog, keep a track of whose turn it is to transmit

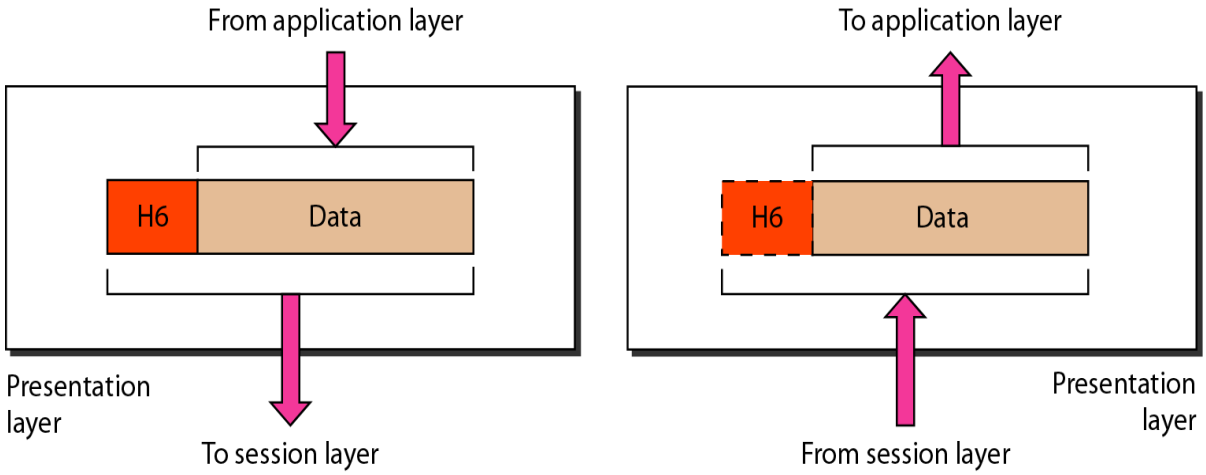
• **Synchronization** – adds check points (synchronization points) into stream of data.




## Presentation Layer:-

**Responsibilities of this layer are:**

- **Translation**
  - Different computers use different encoding systems (bit order translation)
  - Convert data into a common format before transmitting.
  - Syntax represents info such as character codes - how many bits to represent data – 8 or 7 bits
- **Compression** – reduce number of bits to be transmitted Encryption – transform data into an unintelligible format at the sending end for data security
- **Decryption** – at the receiving end

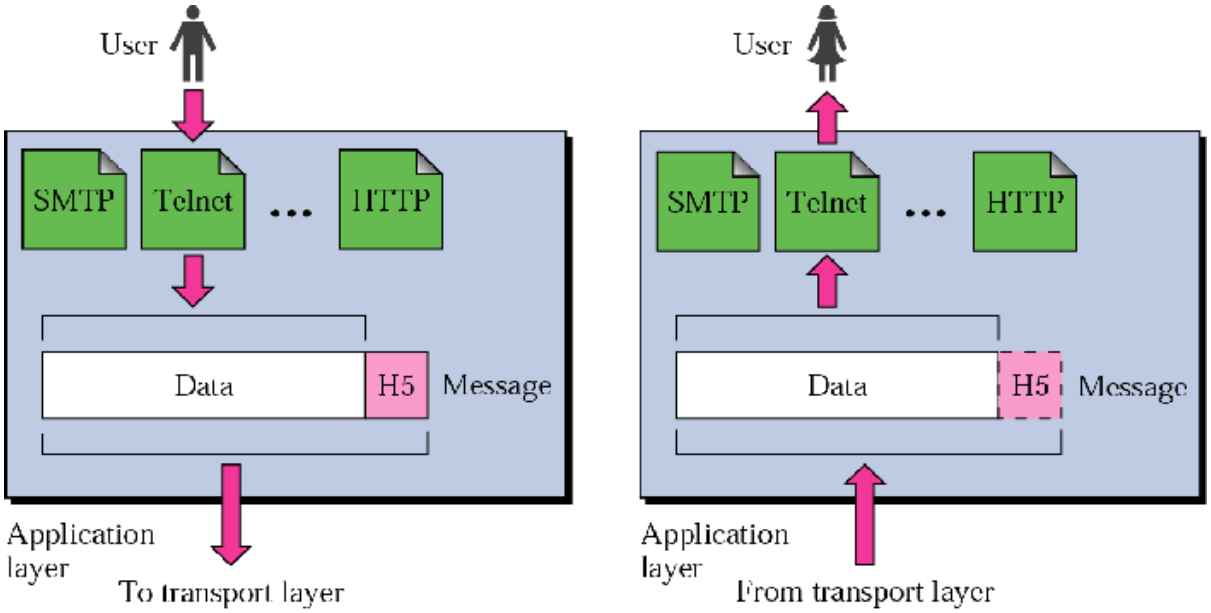



**Application Layer:-**

•Contains protocols that allow the users to access the network (FTP, HTTP, SMTP, etc)

• Does not include application programs such as email, browsers, word processing applications, etc.

• Protocols contain utilities and network-based services that support email via SMTP, Internet access via HTTP, file transfer via FTP, etc




## Data Encapsulation

- The outgoing information will travel down through the layers to the lowest layer.
- While moving down on the source machine, it acquires all the control information which is required to reach the destination machine.
- The control information is in the form of Headers and Trailer which surrounds the data received from the layer above.
- This process of adding headers and trailers to the data is called as **data encapsulation.**

- The information added by each layer is in the form of **headers or trailers**.
- At layer 1 the entire package is converted to a form that can be transferred to the receiving machine.
- At the **receiving machine**, **the message is unwrapped layer by layer**, with each process receiving and removing the data meant for it.
- For example, layer 2 removes the data meant for it, then passes the rest to layer 3.
- Layer 3 then removes the data meant for it and passes the rest to layer 4, and so on.
- The headers and trailers contain control information. The headers and trailers form **the envelope** which carries the message to the desired destination.

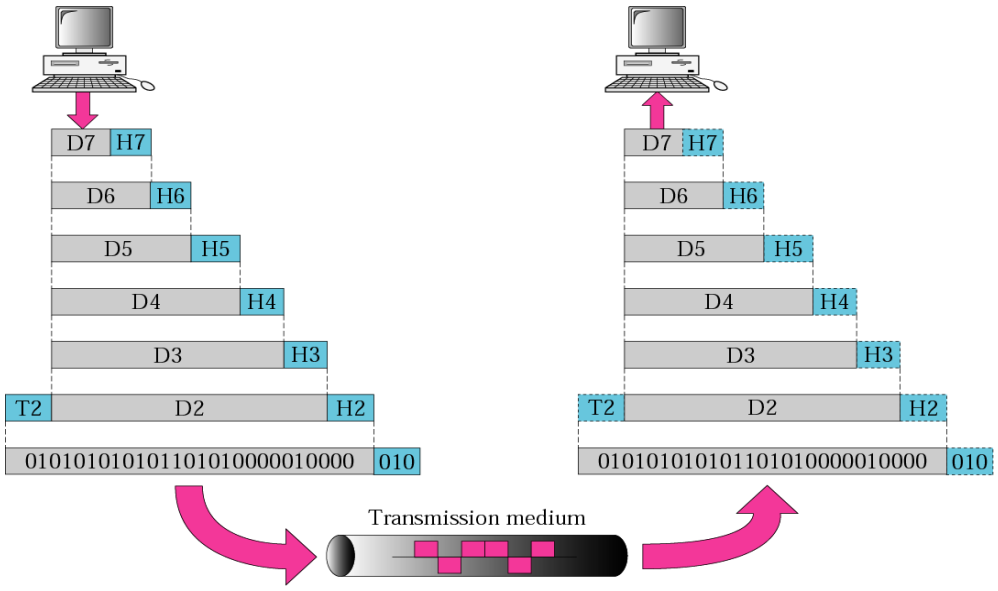



**Figure: Data Encapsulation**

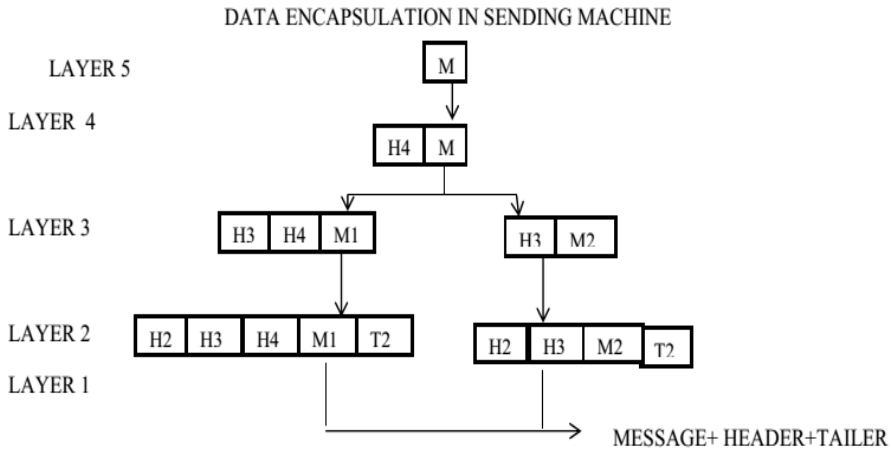
- D7 means the data unit at layer 7, D6 means the data unit at layer 6, and so on.
- The process starts at layer 7 (the application layer), then moves from layer to layer in descending, sequential order.
- At each layer, a **header,** or possibly a **trailer,** can be added to the data unit.
- Commonly, the trailer is added only at layer 2.
- When the formatted data unit passes through the physical layer (layer 1), it is changed into an electromagnetic signal and transported along a physical link.

**Example of Data Encapsulation**

The figure shows the example of five layer stack for data encapsulation.

- The fifth layer of sending machine wants to send a message M to the fifth layer of destination machine.
- The message M is produced by layer 5 of machine 1 and given to layer 4 for transmission. Layer 4 adds header H4 in front of the message and pass it to layer 3.

- Layer 3 breaks up the incoming message into small units as M1 and M2 and pass these packets to layer 2.
- Layer 2 adds the header as well as footer to each packet obtained from layer 3 and pass it to layer 1 for physical transmission.




## Horizontal communication

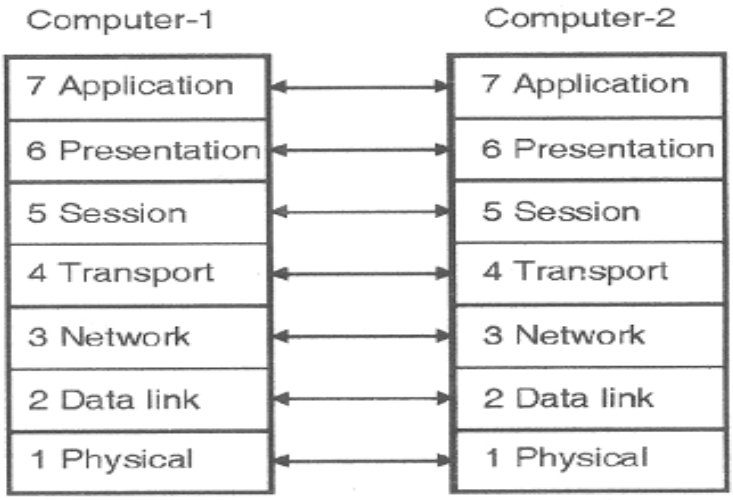



**Fig: Horizontal Communication in OSI Model.**

1. The horizontal communication is the logical connection between the layers, there is no direct communication between them.
2. Information included in each protocol header by the transmitting system is a message that will be carried to the same protocol in the destination system.
3. For two computers to communicate over a n/w, the protocol used at each layer of the OSI model in the transmitting system must be duplicated at the receiving system.
4. The packet travels up through the protocol stack and each successive header is stripped of by the appropriate protocol & processed.
5. When the packet arrived at its destination, the process by which the headers are applied at the source is repeated in server.

## Vertical communication:

1. In addition to communicating horizontally with the same protocol in the other system, the header information also enables each layer to communicate with the layer above & below it.
   Eg. The n/w layer will communicate with the data link layer & transport layer.
2. This interlayer communication is called communication vertical.
3. When a system receives a packet & passes it up through various layers the data link layer protocol header includes a field which specifies the name of n/w layer protocol to be used to process the packet.
4. The n/w layer protocol header will specify the name of transport layer protocol to be used to process the packet.
5. Due to vertical communication, it becomes protocol at each layer simultaneously.

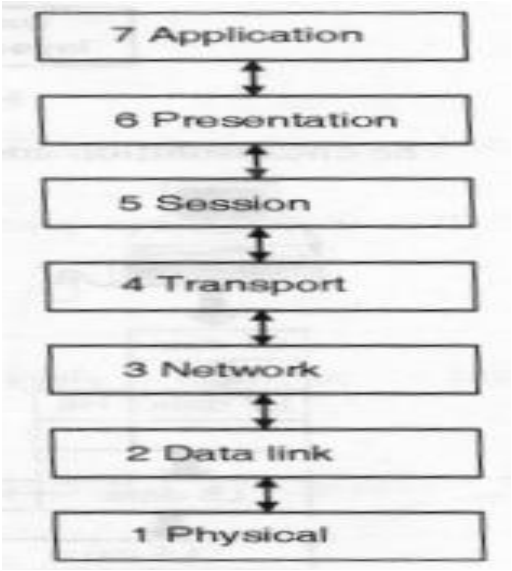



Fig: Vertical communication

## Error Detection and Correction

★ Networks must be able to transfer data from one device to another with complete accuracy.

★ Data can be corrupted during transmission.

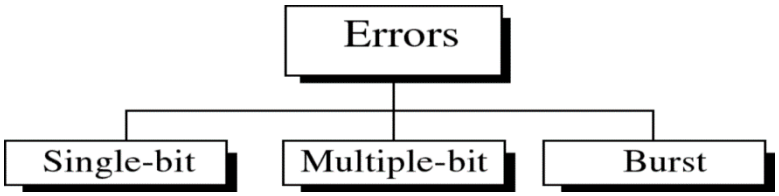
★ For reliable communication, errors must be detected and corrected.

★ Error detection and correction are implemented either at the data link layer or the transport layer of the OSI model.

### Definition of Error

Networks must be able to transform data from once device to another with complete accuracy. While the transmission data can be corrupted, for reliable communication errors must be detected and corrected.

### Types of Errors

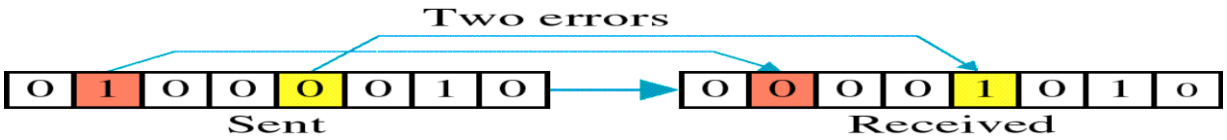



## Single-bit errors

- In a single-bit error, only 1 bit in the data unit has changed from either 0 to 1 or 1 to 0.
- Single bit errors are the least likely type of errors in serial data transmission because the noise must have a very short duration which is very rare. However this kind of errors can happen in parallel transmission.
- *Example:*
  - If data is sent at 1Mbps then each bit lasts only 1/1,000,000 sec. or 1 µs.
  - For a single-bit error to occur, the noise must have a duration of only 1 µs, which is very rare.

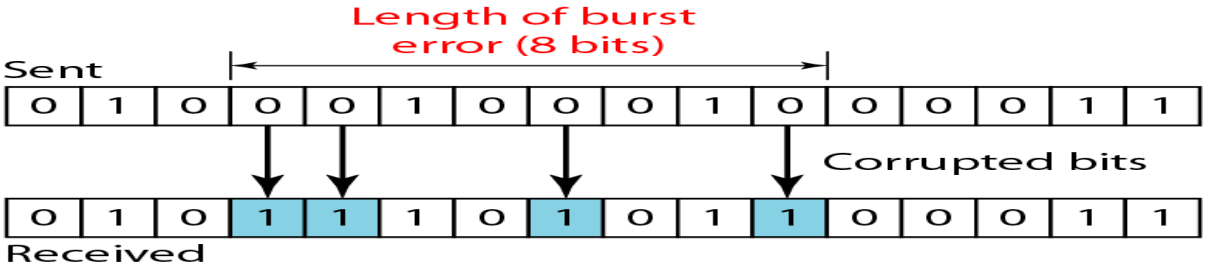

## Multiple-bit errors

- A multi bit error means that 2 or more bits in the data unit have changed




## Burst errors

- A burst error means that 2 or more **consecutive** bits in the data unit have changed.
- The term **burst error** means that two or more bits in the data unit have changed from 1 to 0 or from 0 to 1.
- **Burst errors does not** necessarily **mean that the errors occur in consecutive bits**, the length of the burst is measured from the first corrupted bit to the last corrupted bit. Some bits in between may not have been corrupted.



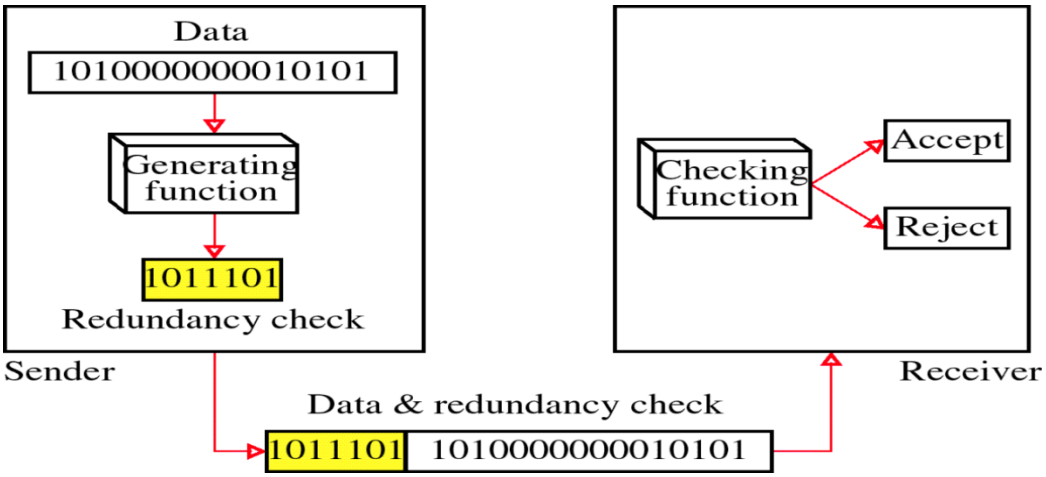
- ★ Burst error is most likely to happen in serial transmission since the duration of noise is normally longer than the duration of a bit.
- ★ The number of bits affected depends on the data rate and duration of noise.

*Example:*

- ➔ If data is sent at rate = 1Kbps then a noise of 1/100 sec can affect 10 bits.(1/100*1000)
- ➔ If same data is sent at rate = 1Mbps then a noise of 1/100 sec can affect 10,000 bits.(1/100*$10^6$)

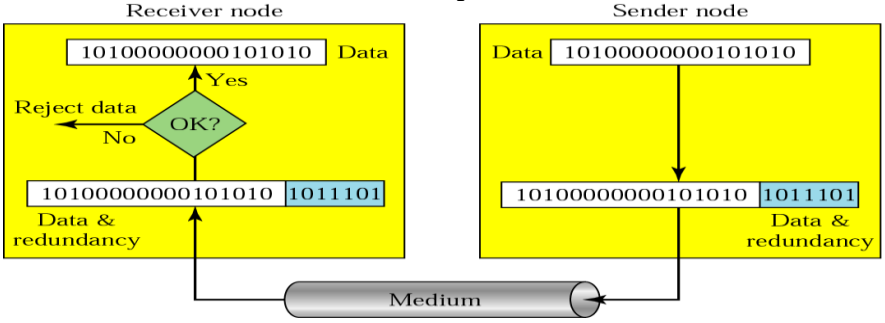
## Error detection

- Error detection means to decide whether the received data is correct or not without having a copy of the original message.
- Error detection uses the concept of **redundancy**, which means adding extra bits for detecting errors at the destination.
- Enough redundancy is added to detect an error.
- The receiver knows an error occurred but does not know which bit(s) is(are) in error.
- Has less overhead than error correction.




## Redundancy:

- The central concept in detecting or correcting errors is redundancy.
- To be able to detect or correct errors, we need to send some extra bits with our data.
- These redundant bits are added by the sender and removed by the receiver. Their presence allows the receiver to detect or correct corrupted bits.



## Detection versus Correction:

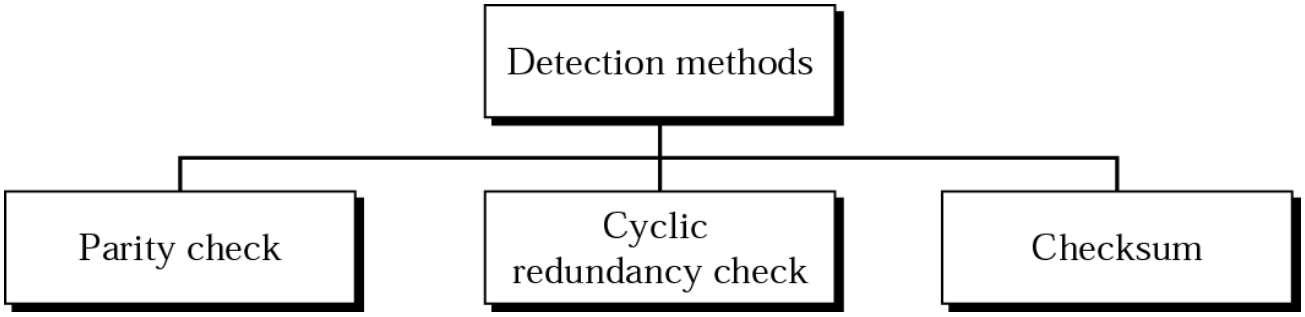
- The correction of errors is more difficult than detection.
- In error detection, we are looking only to see if any error has occurred. The answer is a simple yes or no.
- In error correction, we need to know the exact number of bits that are corrupted and more importantly, their location in the message. The number of errors and the size of message are important.

**Forward Error Correction Versus Retransmission:**
There are **two** main methods of **error correction**.

- **Forward error correction** is the process in which the receiver tries to guess the message by using redundant bits. This is possible if the number of errors is small.
- **Correction by retransmission** is a technique in which the receiver detects the occurrence of an error and asks the sender to resend the message. Resending is repeated until a message arrives that the receiver believes to be error-free.

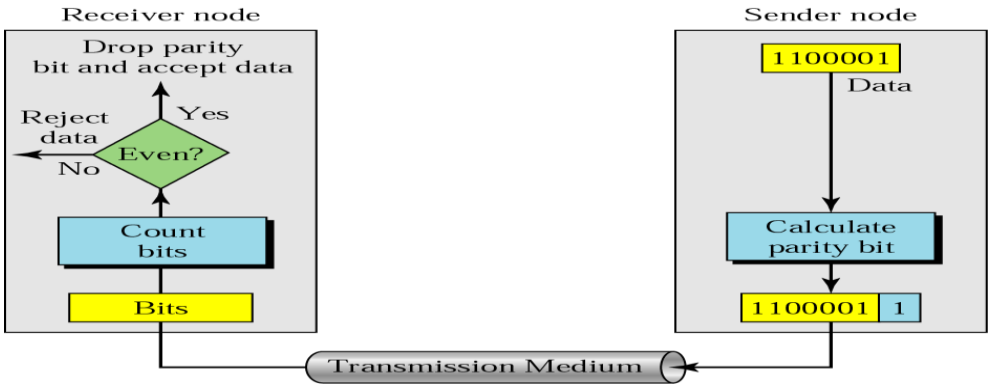
## Detection Techniques

- Parity Checks
- Checksum methods
- Cyclic redundancy checks (CRC)




## • Parity Check

- ❑ A parity bit is added to every data unit so that the total number of 1s(including the parity bit) becomes even for even-parity check or odd for odd-parity check
- ❑ Fig: Simple parity check



**Example-1: Suppose the sender wants to send the word *world*. In ASCII the five characters are coded as**

**1110111 1101111 1110010 1101100 1100100**

**The following shows the actual bits sent**

**11101110 11011110 11100100 11011000 11001001**

**Example-2: Now suppose the word world in Example 1 is received by the receiver without being corrupted in transmission.**

**11101110 11011110 11100100 11011000 11001001**

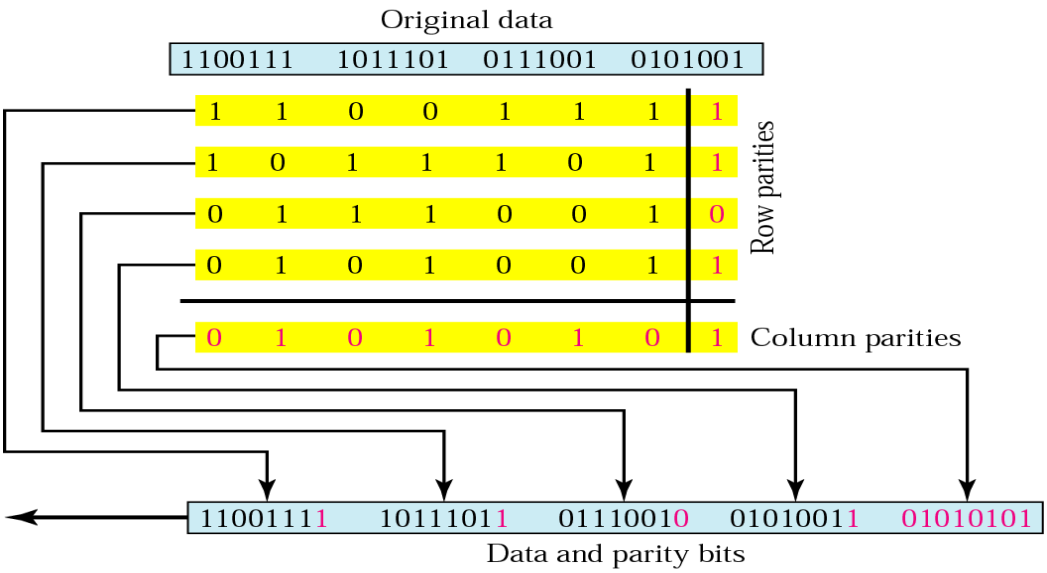
**The receiver counts the 1s in each character and comes up with even numbers (6, 6, 4, 4, 4). The data are accepted.**

**Example-3: Now suppose the word world in Example 1 is corrupted during transmission.**

**11111110 11011110 11101100 11011000 11001001**

The receiver counts the 1s in each character and comes up with even and odd numbers (7, 6, 5, 4, 4). **The receiver knows that the data are corrupted, discards them, and asks for retransmission.**

## Two –Dimensional Parity Check




**Example: Suppose the following block is sent:**

**10101001 00111001 11011101 11100111 10101010**

However, it is hit by a burst noise of length 8, and some bits are corrupted.

**10100011 10001001 11011101 11100111 10101010**

When the receiver checks the parity bits, some of the bits do not follow the even-parity rule and the whole block is discarded.

**10100011 10001001 11011101 11100111 10101010**

**Question: State any two drawbacks of parity checking for error detection.**

**Answer: Drawbacks of parity checking for error detection:**

1. Can be used to detect single bit errors
2. Cannot detect location of errors.
3. Overheads are more.

**Question: Assuming even parity technique find the parity bit for following frames:**
**i) 0000010 ii) 1111000 iii) 1010101 iv) 1011011**

**Answer:**

| Sr. No | Data | Parity bit |
|---|---|---|
| 1 | 0000010 | 1 |
| 2 | 1111000 | 0 |
| 3 | 1010101 | 0 |
| 4 | 1011011 | 1 |

**Question: Assuming odd parity, find the parity bit for each of the following data unit:**
**(i) 1011010 (ii) 0010110 (iii) 1001111 (iv) 1100000**

**Answer:** Odd parity refers to number of „1" present in a byte to be transmitted should be odd.

**(i) 1011010:**

Step 1: Count the number of "1"s in the byte
Answer: 4
Step 2: compute the parity bit
Answer: 1011010 1
Since the total number of 1's is 4, the odd parity will have a value of "1".

**(ii) 0010110:**

Step 1: Count the number of "1"s in the byte
Answer: 3
Step 2: compute the parity bit
Answer: 0010110 0
Since the total number of 1's is 3, the odd parity will have a value of "0".

**(iii) 1001111:**

Step 1: Count the number of 1's in the byte
Answer: 5
Step 2: compute the parity bit
Answer: 1001111 0

Since the total number of 1's is 5, the odd parity will have a value of "0".

**(iv) 1100000:**

Step 1: Count the number of 1's in the byte

Answer: 2

Step 2: compute the parity bit

Answer: 1100000 1

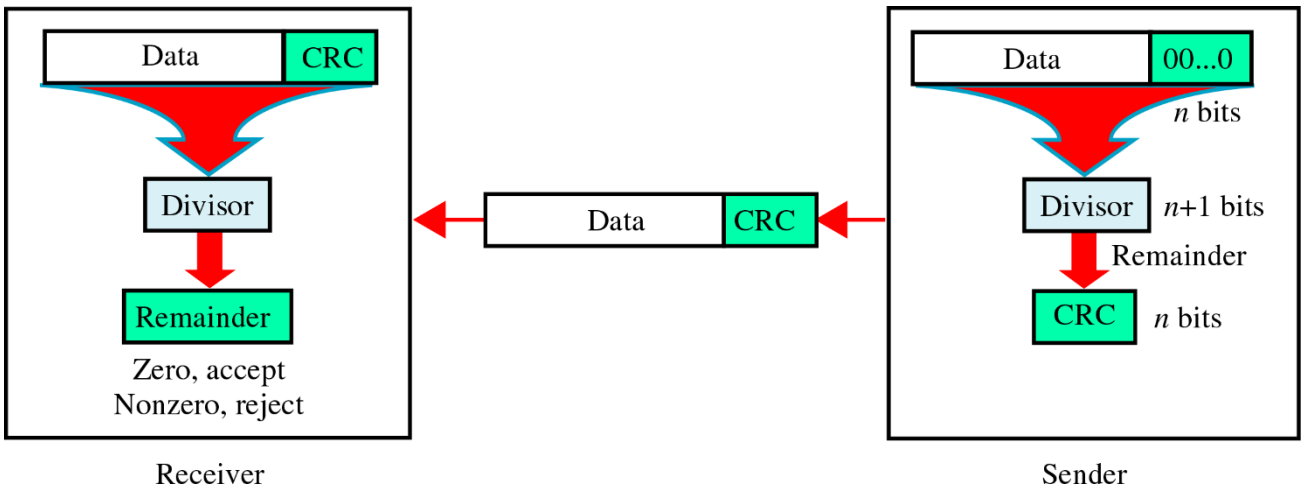
Since the total number of 1's is 2, the odd parity will have a value of "1".

- **CRC(Cyclic Redundancy Check)**

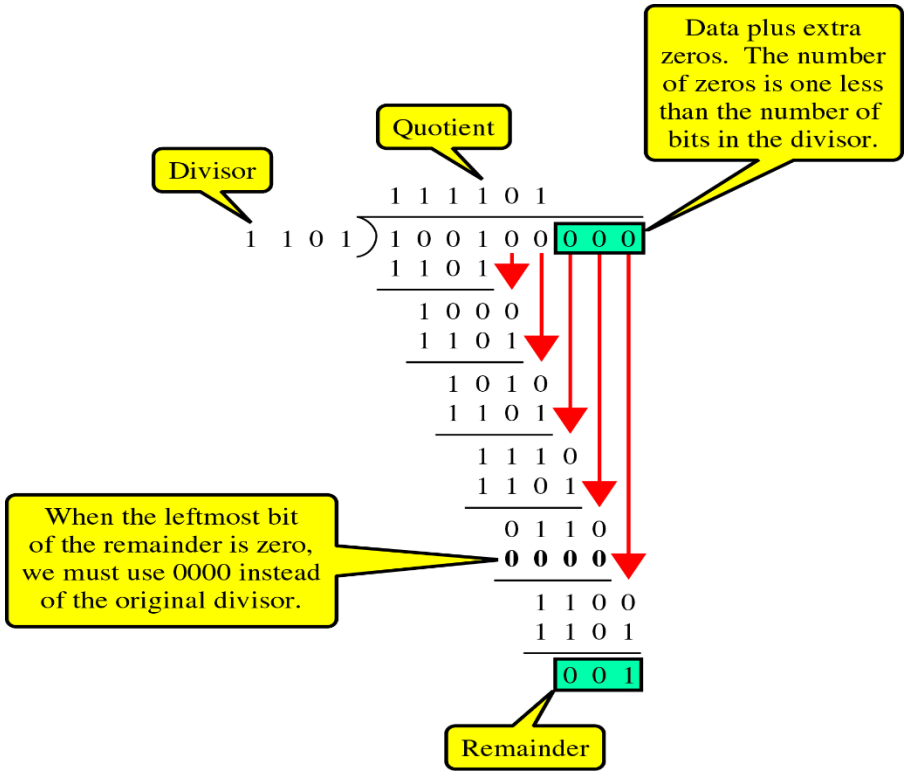
- **CRC Encoder:**
  - In the encoder, the dataword has $k$ bits; the codeword has n bits.
  - The size of the dataword is augmented by adding $n-k$ 0s to the right-hand side of the word. The $n$-bit result is fed into the generator.
  - The generator uses a divisor of size $n - k + 1$, predefined and agreed upon. The generator divides the augmented dataword by the divisor (modulo-2 division).
  - The quotient of the division is discarded; the remainder $r2\ r1\ r0$ is appended to the dataword to create the codeword.

- **CRC Decoder:**
  - The codeword can change during transmission.
  - The decoder does the same division process as the encoder. The remainder of the division is the syndrome.
  - If the syndrome is all 0s, there is no error; the dataword is separated from the received codeword and accepted.
  - Otherwise, everything is discarded.

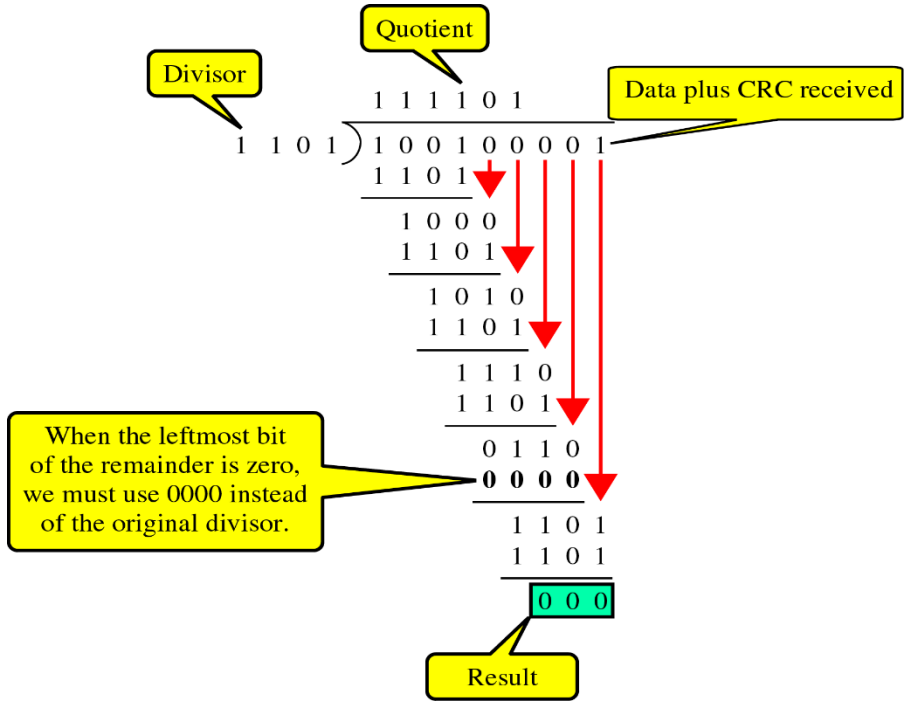
- **CRC(Cyclic Redundancy Check) is based on binary division.**




- **CRC generator uses modular-2 division.**

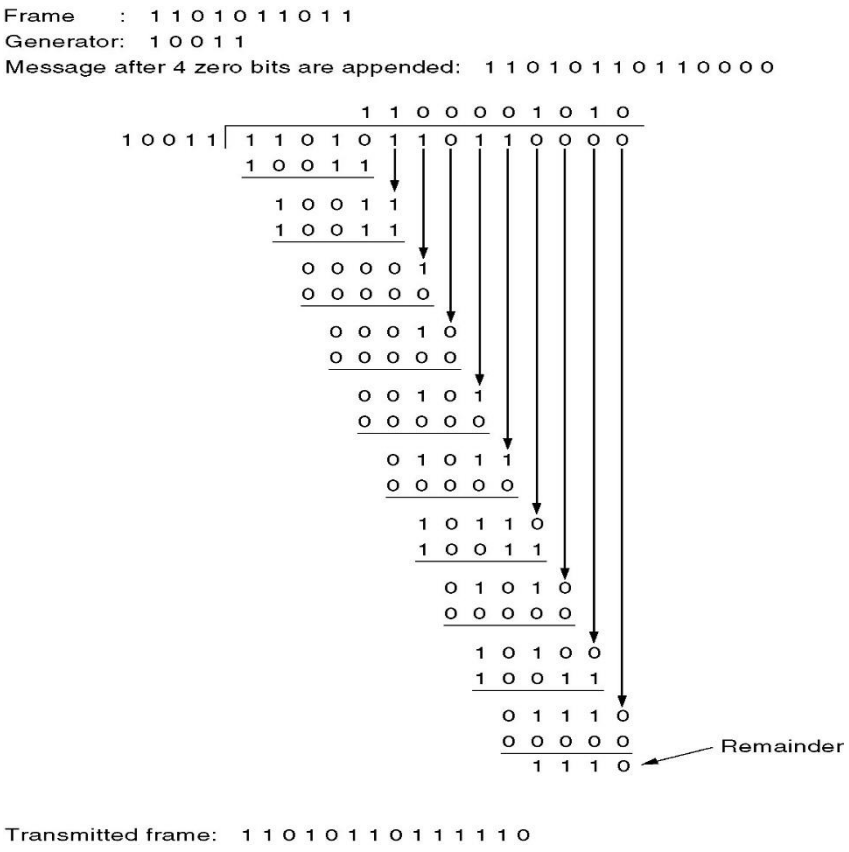



## CRC Decoder:




Another Example: **Original Message:1 1 0 1 0 1 1 0 1 1**

- Devisor: 1 0 0 1 1
- Remainder:1 1 1 0
- Transmitted msg: 1 1 0 1 0 1 1 0 1 1 1 1 1 0




Question: **Explain the process of CRC with respect to following example. If G (X) = 110010 and M (X) = 101 then calculate CRC for above stream.**

## Answer:

**Procedure:**- data bits= G(X)=110010 divisor=M(X)=101

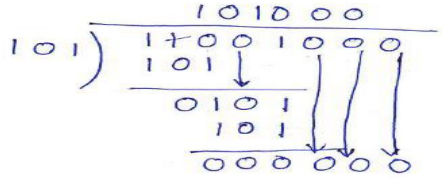
Here divisor is 3 bits so we need to append 2 zeroes (2 bit) to the data bits for division. Division carried is the normal binary division.

Result is calculated by the following condition:

1. If the remainder after division process is zero, it indicates that the data bits has no errors and the data bit is acceptable
2. If the remainder after division is non-zero , it indicates that the data bits has errors and we have to append the remainder bits to the original data bits and then send the data again. This remainder bits are called as the CRC. So the data bits transmitted will be DATA + CRC.

**Consider the given example, lets perform division process for CRC.**

Here the divisor is 3 bits hence we append 2 zeroes to the data bits, so the data bits will be 11001000 this will be divided by 101. **Since remainder is 0 there is no error in the data.**

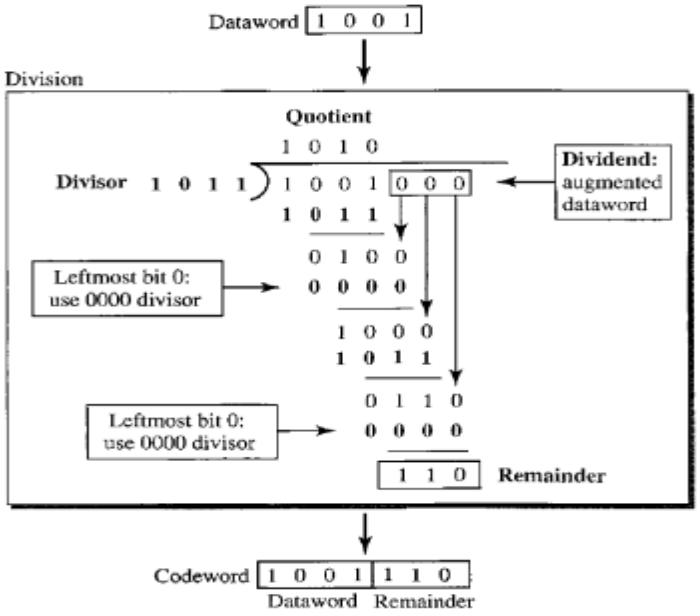



Reminder is 00 = CRC

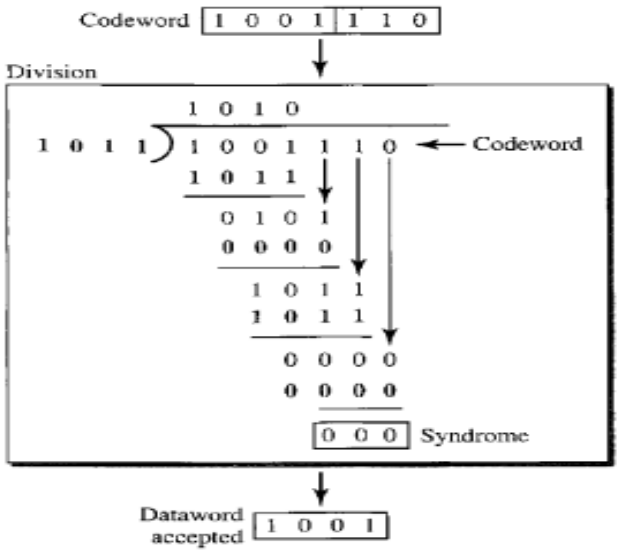
∴ Data + CRC = 11001 0 + 00

= 1100 1000

**Question: Explain process of CRC (Cyclic Redundancy Check) with example.**
**Answer:**
**CRC Encoder:**
In the encoder, the dataword has $k$ bits (4 here); the codeword has n bits (7 here). The size of the dataword is augmented by adding $n - k$ (3 here) 0s to the right-hand side of the word. The $n$-bit result is fed into the generator. The generator uses a divisor of size $n - k + 1$ (4 here), predefined and agreed upon. The generator divides the augmented dataword by the divisor (modulo-2 division). The quotient of the division is discarded; the remainder $r_2 r_1 r_0$ is appended to the dataword to create the codeword.
**Example:**
Let us take a closer look at the encoder. The encoder takes the dataword and augments it with $n - k$ number of 0s. It then divides the augmented dataword by th divisor, as shown in Figure.




**CRC Decoder:**
The codeword can change during transmission. The decoder does the same division process as the encoder. The remainder of the division is the syndrome. If the syndrome is all 0s, there is no error; the dataword is separated from the received codeword and accepted. Otherwise, everything is discarded.



## Checksum:

Checksum is an error detection method. It can be done by following steps

**Step-01:**

- At sender side, If m bit checksum is used, the data unit to be transmitted is divided into segments of m bits.
- All the m bit segments are added.
- The result of the sum is then complemented using 1"s complement arithmetic.
- The value so obtained is called as checksum**.**

**Step-02:**

- The data along with the checksum value is transmitted to the receiver.

**Step-03:**

- All the m bit segments are added along with the checksum value. The value so obtained is complemented and the result is checked.
- Case-01: Result = 0 If the result is zero
  - Receiver assumes that no error occurred in the data during the transmission.
  - Receiver accepts the data.
- Case-02: Result ≠ 0 If the result is non-zero,
  - Receiver assumes that error occurred in the data during the transmission.
  - Receiver discards the data and asks the sender for retransmission.

**Example:**

- At the sender Original data : 10101001 00111001

```
10101001
00111001
--------------
11100010      Sum
00011101      Checksum(1's complement)
```

Data to be transmitted:10101001 00111001 00011101

- At the receiver

Received data:10101001 00111001 00011101

```
10101001
00111001
00011101
---------------
11111111 ← Sum
00000000 ← Complement is zero so the data is accepted.
```

**Example-2**: Consider the data unit to be transmitted is- 10011001111000100010010010000100
Consider 8 bit checksum is used.

- Step-01:
  - At sender side,
  - The given data unit is divided into segments of 8 bits as-
- Now, all the segments are added and the result is obtained as- 10011001 + 11100010 + 00100100 + 10000100 = 1000100011
  - Since the result consists of 10 bits, so extra 2 bits are wrapped around.
    00100011 + 10 = 00100101 (8 bits)
  - Now, 1"s complement is taken which is 11011010.
  - Thus, checksum value = 11011010
- Step-02:
  - The data along with the checksum value is transmitted to the receiver.

- Step-03:
  - At receiver side, The received data unit is divided into segments of 8 bits.
  - All the segments along with the checksum value are added.
  - Sum of all segments + Checksum value =
  - 00100101 + 11011010= 11111111
  - Complemented value = 00000000
  - Since the result is 0, receiver assumes no error occurred in the data and therefore accepts it.

**Error Correction:** It can be handled in two ways

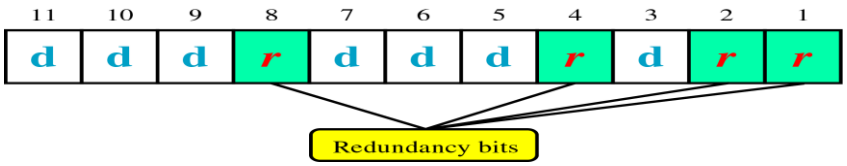
- When an error is discovered, the receiver can have the sender retransmit the entire data unit.
  - Automatic Repeat Request (ARQ)
- A receiver can use an error-correcting code, which automatically corrects certain errors.
  - Hamming Code

**Hamming Code**: **developed by R.W.Hamming**

- Positions of redundancy bits in 7 bit Hamming code is shown in fig.

**Each r bit is the VRC bit for one combination of data bits, ie even parity bit is generated using the following bits and code is generated.**

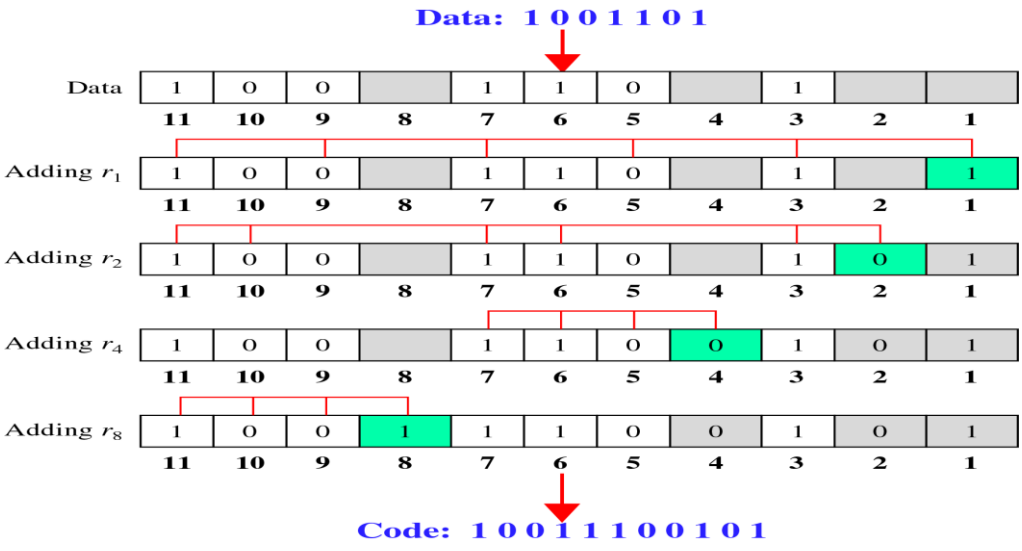
- $r_1$ = bits 1, 3, 5, 7, 9, 11
- $r_2$ = bits 2, 3, 6, 7, 10, 11
- $r_4$ = bits 4, 5, 6, 7
- $r_8$ = bits 8, 9, 10, 11



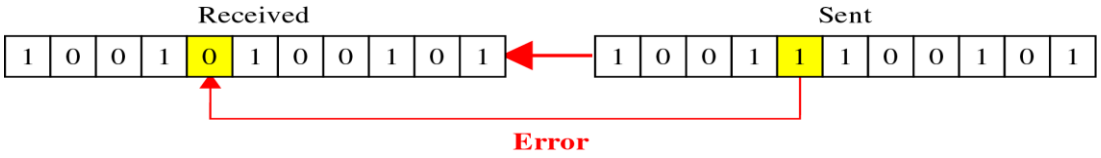

- The redundant bits are inserted at each $2^n$ bit where n=0,1,2,3,……

Example:

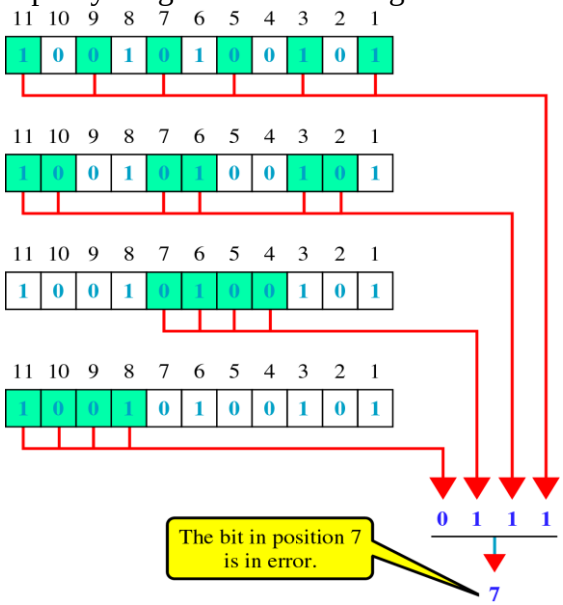
- Calculating the r values (Sender Side)



1. Now Consider that receiver receives data with one bit error as shown in the fig.




2. **Error detection using Hamming Code (Receiver Side)**
3. At the receiver side the even parity is again checked along with code.




4. The parity generated shows the bit position in error.

## Framing:

- The data link layer needs to pack bits into frames, so that each frame is distinguishable from another.
- *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.*
- Framing in the data link layer separates a message from one source to a destination, or from other messages to other destinations, by adding addresses.
- The destination address defines where the packet is to go; the sender address helps the recipient acknowledge the receipt.
- Frames can be of fixed or variable size.

**Fixed-Size Framing**

- 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 used 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.

**Question: Two channels one with a bit rate of 100 Kbps and another with bit rate of 200 Kbps are to be multiplexed.**

**Answer the following questions:**

**i) Calculate size of frames in bits**

**ii) Calculate the frame rate**

**iii) Calculate the duration of frame**

**Answer:**

Channel 1 has a bit rate of 100Kbps. Channel 2 has a bit rate of 200Kbps Hence channel 2 is demultiplexed into 2 channels of 100Kbps each. Hence 3 channels of 100 Kbps are multiplexed effectively.

Let us consider that one slot of the channel 1 is allocated and two slots of the channel 2 is allocated in the frame .

**i) Calculate size of frames in bits:** Thus **e**ach frame carries 3 bits.

**ii) Calculate the frame rate:** The total bit rate of the multiplexed link is 300kbps. Each frame has 3 bits. The frame rate is100,000 frames per second (Any other assumption may also be considered).

**iii) Calculate the duration of frame:** Thus the frame duration is

- 1/100,000s or 1μs.

## Flow and Error Control

- Data communication requires at least two devices working together, one to send and the other to receive.
- Even such a basic arrangement requires a great deal of coordination for an intelligible exchange to occur.
- The most important responsibilities of the data link layer are flow control and error control.
- Collectively, these functions are known as **data link control**.

## Flow Control:

- Flow control coordinates the amount of data that can be sent before receiving an acknowledgment and is one of the most important duties of the data link layer.
- The flow of data must not be allowed to overwhelm the receiver.
- Any receiving device has a limited speed at which it can process incoming data and a limited amount of memory in which to store incoming data.
- The receiving device must be able to inform the sending device before those limits are reached and to request that the transmitting device send fewer frames or stop temporarily.
- Incoming data must be checked and processed before they can be used.
- The rate of such processing is often slower than the rate of transmission.
- For this reason, each receiving device has a block of memory, called a **buffer**, reserved for storing incoming data until they are processed.
- If the buffer begins to fill up, the receiver must be able to tell the sender to halt transmission until it is once again able to receive.

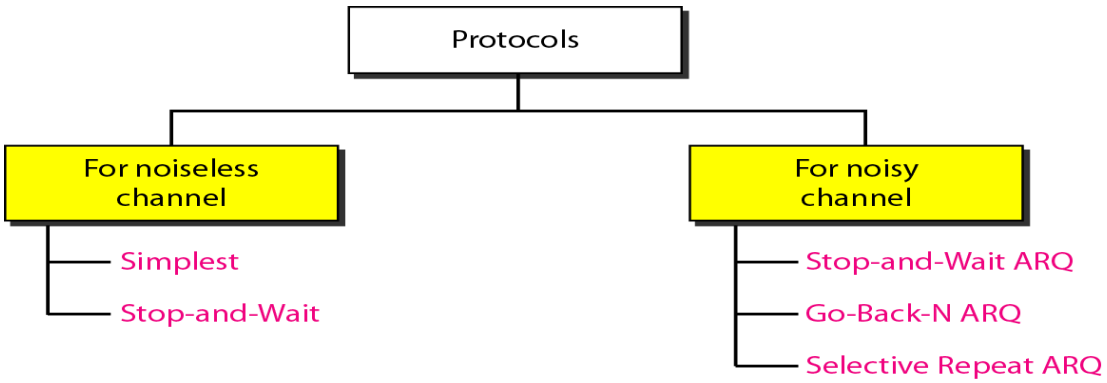
## Error Control:

- Error control is both error detection and error correction.

- It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender.

- In the data link layer, the term error control refers primarily to methods of error detection and retransmission.
- Error control in the data link layer is often implemented simply: Any time an error is detected in an exchange, specified frames are retransmitted. This process is called automatic repeat request (ARQ).

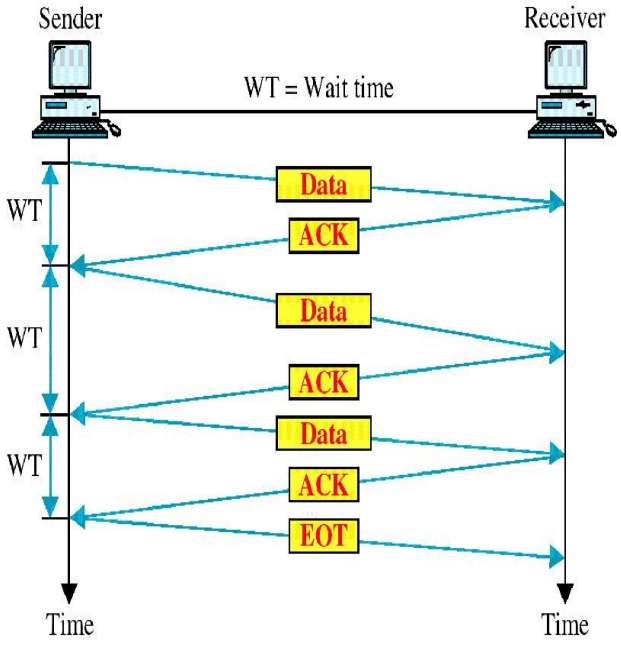
# Flow Control and Error Control Technics




**Question: Explain stop and wait ARQ with example.**

## Stop and Wait:

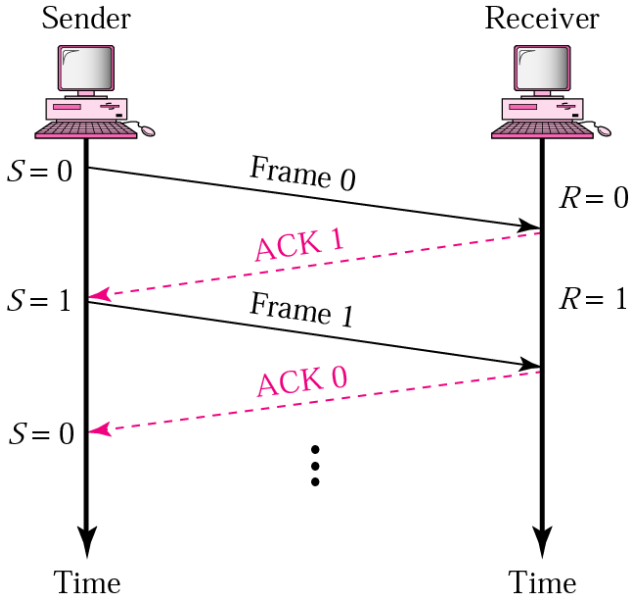
This is a very simple method where in the sender sends one frame of data and necessarily waits for an acknowledgement (ACK) from the receiver before sending the next frame. Only after the sender receives and acknowledgement for a frame does it send the next frame. Thus, the transmission always takes the form Data-ACK-Data-ACK….etc, where the Data frames are sent by the sender, and the ACK frames are sent by the receiver back to the sender. This is shown in figure. The stop-and wait- approach is pretty simple to implement. Every frame must be individually acknowledged before the next frame can be transmitted. However, therein also lies its drawback. Since the sender must receive each acknowledgement before it can transmit the next frame, it makes the transmission very slow.



5. ***Stop-and-Wait ARQ*** *has the following features:*
   - ✓ *The sending device keeps a copy of the sent frame transmitted until it receives an acknowledgment( ACK)*
   - ✓ *The sender starts a timer when it sends a frame. If an ACK is not received within an allocated time period, the sender resends it*
   - ✓ *Both frames and acknowledgment (ACK) are numbered alternately 0 and 1( two sequence number only)*
   - ✓ *This numbering allows for identification of frames in case of duplicate transmission.*
   - ✓ *The acknowledgment number defines the number of next expected frame. (frame 0 received ACK 1 is sent)*
   - ✓ *A damage or lost frame treated by the same manner by the receiver*
   - ✓ *If the receiver detects an error in the received frame, or receives a frame out of order it simply discards the frame*
   - ✓ *The receiver send only positive ACK for frames received safe; it is silent about the frames damage or lost.*
   - ✓ *The sender has a control variable Sthat holds the number of most recently sent frame (0 or 1). The receiver has control variable R, that holds the number of the next frame expected (0,or 1)*

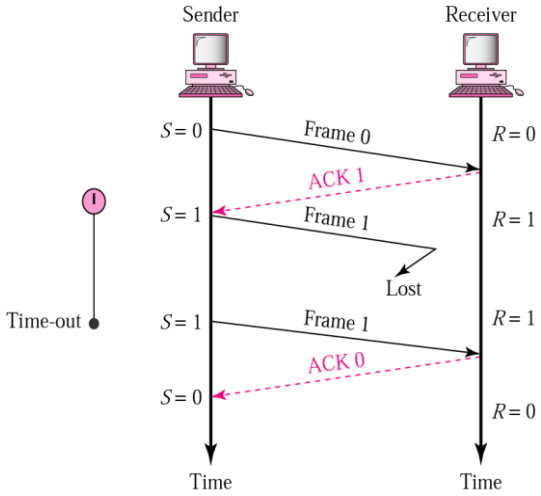
   ***Cases of Operations:***
   1. *Normal operation*
   2. *The frame is lost*
   3. *The Acknowledgment (ACK) is lost*
   4. *The Ack is delayed*

**1. Normal Operation:**
- ➢ The sender will not send the next frame until it is sure that the current one is correctly receive
- ➢ sequence number is necessary to check for duplicated frames

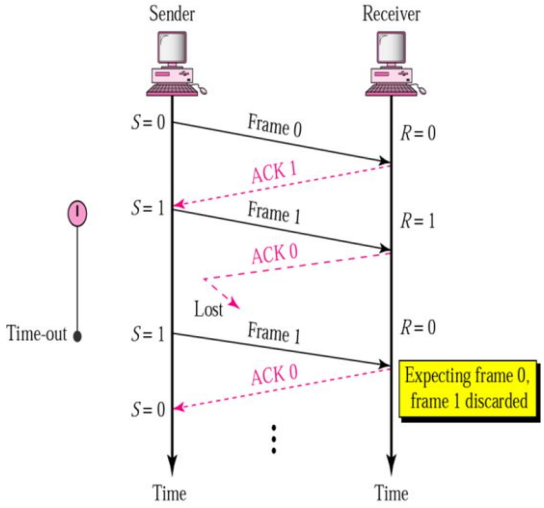


**2. Lost or damaged frame**

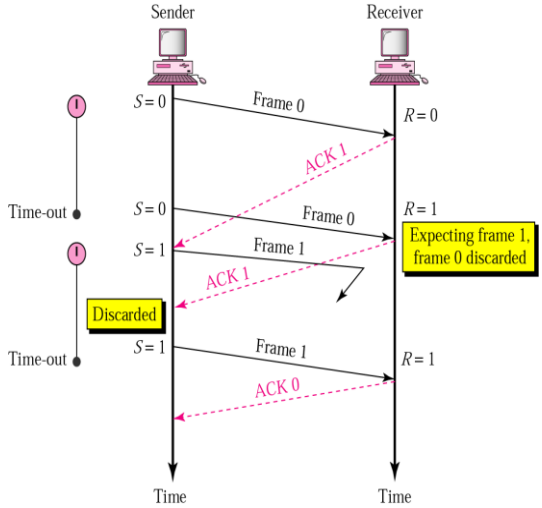
- A damage or lost frame treated by the same manner by the receiver.
- No NACK when frame is corrupted / duplicate




3. **Lost ACK frame**



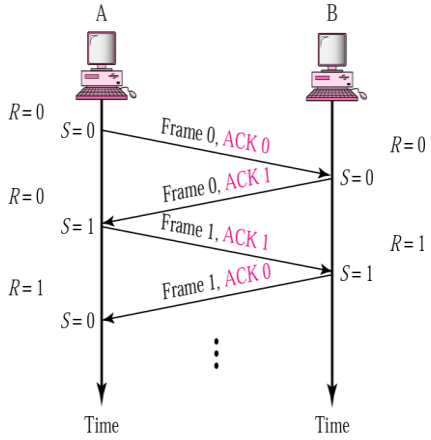

4. **Delayed ACK and lost frame**




5.

## *Piggybacking ( Bidirectional transmission)*

- Is a method to combine a data frame with an acknowledgment.
- It can save bandwidth because data frame and an ACK frame can combined into just one frame.




## Sliding window protocol

*Sliding window protocols apply Pipelining:A task is begun before the previous task has ended.*

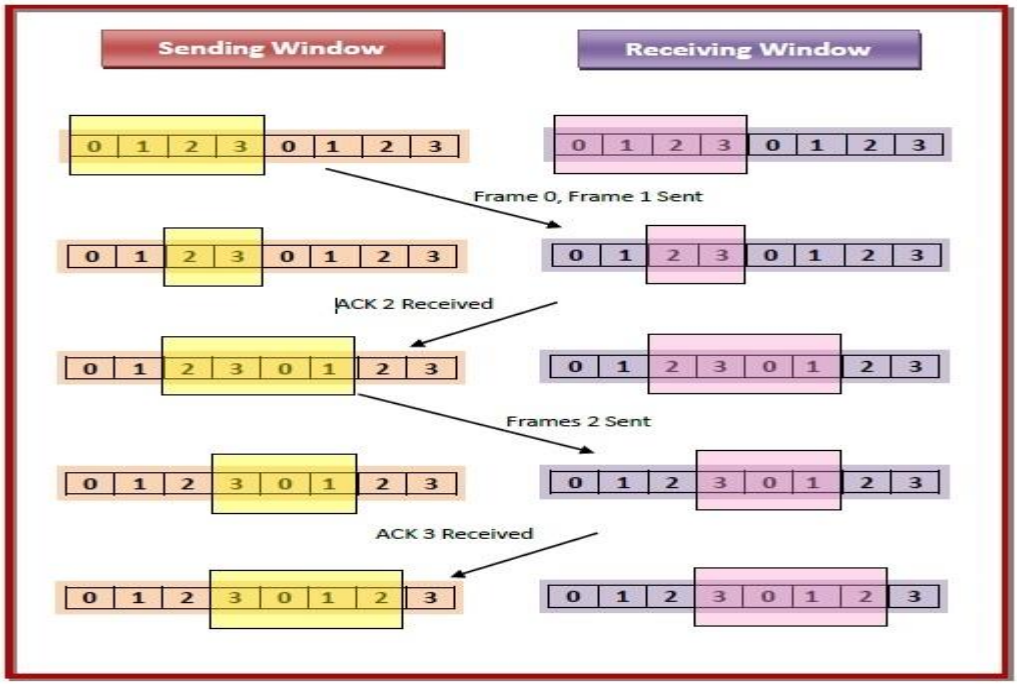
- The Stop and Wait ARQ offers error and flow control, but may cause big performance issues as sender always waits for acknowledgement even if it has next packet ready to send. Consider a situation where you have a high bandwidth connection and propagation delay is also high (you are connected to some server in some other country though a high speed connection), you can't use this full speed due to limitations of stop and wait.
- Sliding Window protocol handles this efficiency issue by sending more than one packet at a time with a larger sequence numbers. The idea is same as pipelining in architectures.

### Working Principle

- In these protocols, the sender has a buffer called the sending window and the receiver has buffer called the receiving window.
- The size of the sending window determines the sequence number of the outbound frames. If the sequence number of the frames is an n-bit field, then the range of sequence numbers that can be assigned is 0 to $2^n-1$. Consequently, the size of the sending window is $2^n-1$. Thus in order to accommodate a sending window size of $2^n-1$, a n-bit sequence number is chosen.
- The sequence numbers are numbered as modulo-n. For example, if the sending window size is 4, then the sequence numbers will be 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, and so on. The number of bits in the sequence number is 2 to generate the binary sequence 00, 01, 10, 11.
- The size of the receiving window is the maximum number of frames that the receiver can accept at a time. It determines the maximum number of frames that the sender can send before receiving acknowledgment.

## Example

- Suppose that we have sender window and receiver window each of size 4. So the sequence numbering of both the windows will be 0,1,2,3,0,1,2 and so on. The following diagram shows the positions of the windows after sending the frames and receiving acknowledgments.

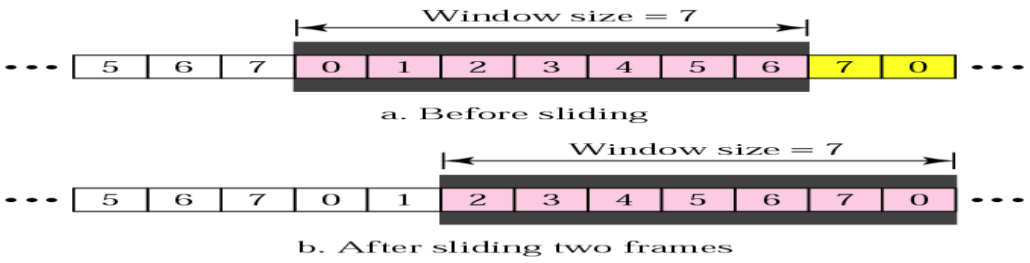



## Types of Sliding Window Protocols

The Sliding Window ARQ (Automatic Repeat reQuest) protocols are of two categories –


- **Go – Back – *N* ARQ**

Go – Back – *N* ARQ provides for sending multiple frames before receiving the acknowledgment for the first frame. It uses the concept of sliding window, and so is also called sliding window protocol. The frames are sequentially numbered and a finite number of frames are sent. If the acknowledgment of a frame is not received within the time period, all frames starting from that frame are retransmitted.



### Working Principle

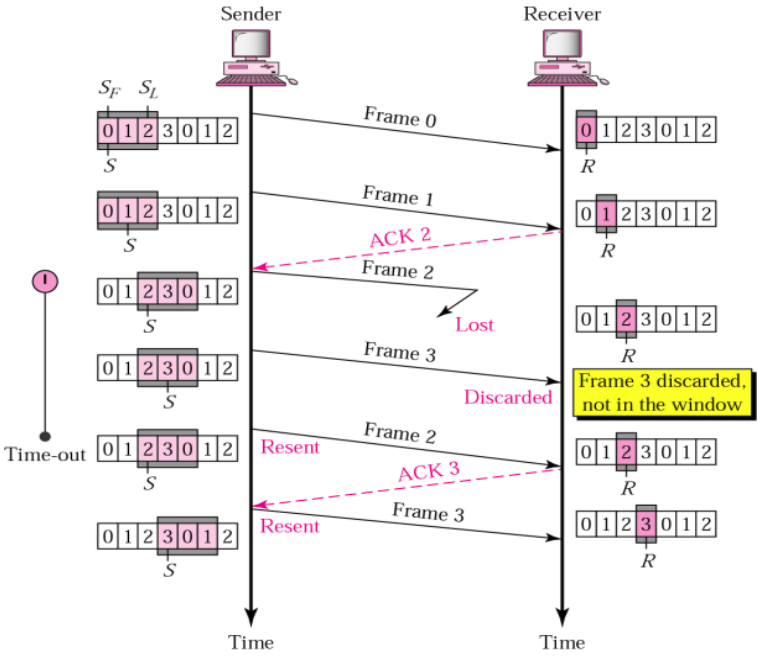
Go – Back – N ARQ provides for sending multiple frames before receiving the acknowledgment for the first frame. The frames are sequentially numbered and a finite number of frames. The maximum number of frames that can be sent depends upon the size of the sending window. If the acknowledgment of a frame is not received within an agreed upon time period, all frames starting from that frame are retransmitted.

The size of the sending window determines the sequence number of the outbound frames. If the sequence number of the frames is an n-bit field, then the range of sequence numbers that can be assigned is 0 to $2^n-1$. Consequently, the size of the sending window is $2^n-1$. Thus in order to accommodate a sending window size of $2^n-1$, a n-bit sequence number is chosen.

The sequence numbers are numbered as modulo-n. For example, if the sending window size is 4, then the sequence numbers will be 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, and so on. The number of bits in the sequence number is 2 to generate the binary sequence 00, 01, 10, 11.

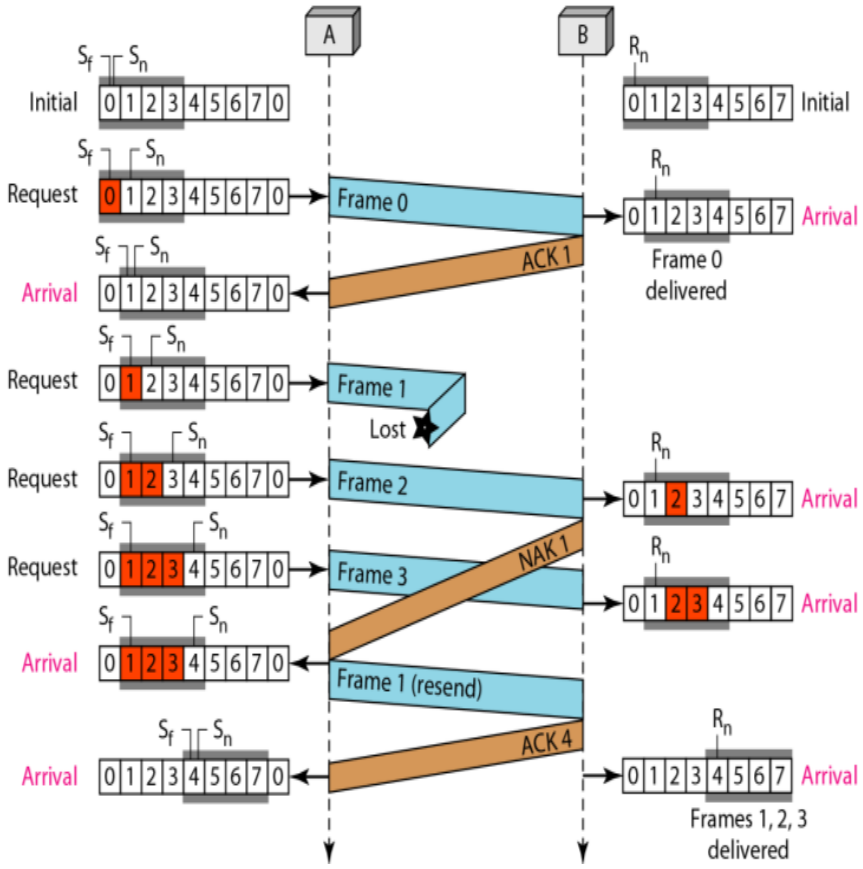
The size of the receiving window is 1.

*Example: The sender has sent frame 6, and timer expires for frame 3(frame 3 has not been acknowledge); the sender goes back and resends frames 3, 4,5 and 6*




- **Selective Repeat ARQ**

This protocol also provides for sending multiple frames before receiving the acknowledgment for the first frame. However, here only the erroneous or lost frames are retransmitted, while the good frames are received and buffered.




**Question: Explain the following flow and error control techniques: Go back N ARQ.**
**Answer: Go-Back-N ARQ:**
In Go-Back-N ARQ method, both sender and receiver maintain a window.

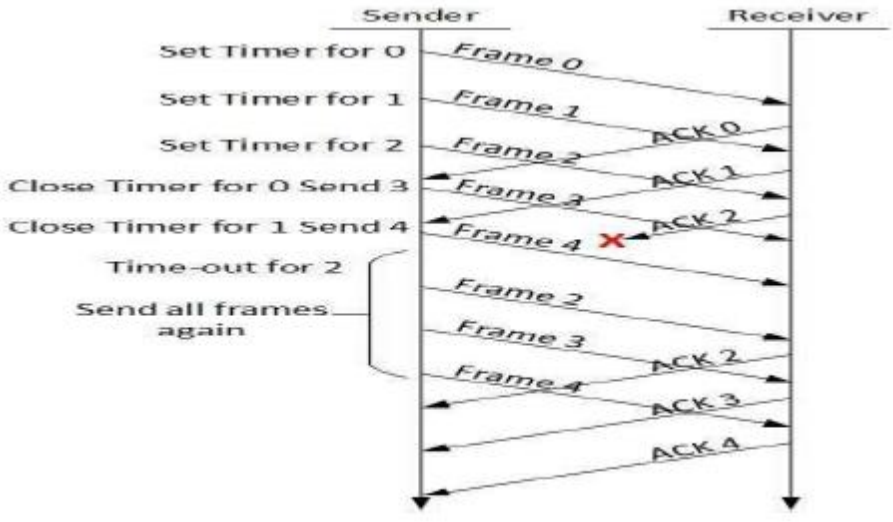



**Fig: Go-Back-N ARQ**

- The sending-window size enables the sender to send multiple frames without receiving the acknowledgement of the previous ones.
- The receiving-window enables the receiver to receive multiple frames and acknowledge them. The receiver keeps track of incoming frame's sequence number.
- When the sender sends all the frames in window, it checks up to what sequence number it has received positive acknowledgement.

- If all frames are positively acknowledged, the sender sends next set of frames.
- If sender finds that it has received NACK (negative acknowledgement)or has not receive any ACK for a particular frame, it retransmits all the frames after which it does not receive any positive ACK.