UNIT 3- DATALINK LAYER

Responsible for reliable transmission of packets over a link

– Framing: Determine the start and end of packets

– Error Detection: Determine when a packet contains errors

– Error recovery: Retransmission of packets containing errors

Framing:

010100111010100100101010100111000100

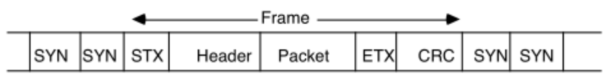
Where is data in the above?

• Three approaches to find frame and idle fill boundaries:

1)Character oriented framing 2) Length counts - fixed length 3) Bit oriented protocols (flags)

Three approaches to find frame and idle fill boundaries:
1) Character oriented framing
2) Length counts
- fixed length
3) Bit oriented protocols (flags)
----

1)Character Based Framing



SYN is synchronous idle
STX is start text
ETX is end text

**Standard character codes such as ASCII and EBCDIC contain special communication characters that cannot appear in data

** Entire transmission is based on a character code

Issues with Character Based Framing:

• Character code dependent

– How do you send binary data?

• Frames must be integer number of characters

• Errors in control characters are messy

2)Length field approach (DECNET):

• Use a header field to give the length of the frame (in bits or bytes)

– Receiver can count until the end of the frame to find the start of the next frame

– Receiver looks at the respective length field in the next packet header to find that packet's length

• Length field must be log2 (Max_Size_Packet) + 1 bits long

– This restricts the packet size to be used

• Issues with length counts

– Difficult to recover from errors

– Resynchronization is needed after an error in the length count

<u>Fixed Length Packets (e.g., ATM)</u>

• All packets are of the same size

– In ATM networks all packets are 53 Bytes

• Requires synchronization upon initialization

• Issues:

– Message lengths are not multiples of packet size

Last packet of a message must contain idle fill (efficiency)

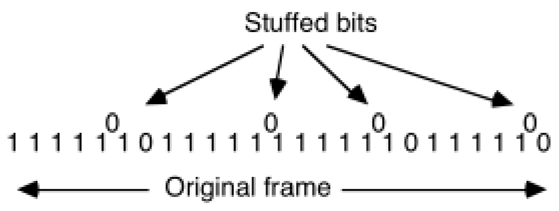
– Synchronization issues

– Fragmentation and re-assembly is complicated at high rates

<u>3)BIT STUFFING (Transmitter)</u>

• Used to remove flag from original data

• A 0 is stuffed after each consecutive five 1's in the original frame



• Why is it necessary to stuff a 0 in 0111110?

– If not, then

0111110111 -> 0111110111

011111111 -> 0111110111

– How do you differentiate at the receiver? Its very tough.

Error Detection Methods | CRC, VRC, LRC, Checksum error detection techniques

This page describes error detection methods or techniques.The methods of error detection in networking are VRC, LRC, CRC and Checksum. CRC error detection and correction example is also explained.

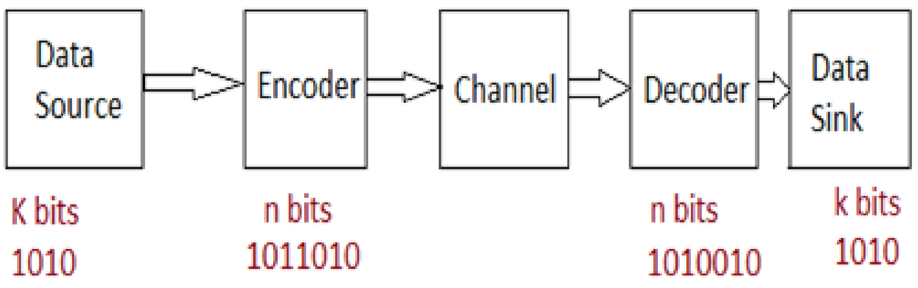
What is Error Detection and Correction?

As we know errors occur to the data during data transmission as well as data processing stages. The errors can occur due to various reasons such as electrostatic interference from nearby circuits, attenuation due to cable or path resistance, distortion due to inductance/capacitance, transmission loss due to leakages etc.

LAN and optical cables introduce less errors than wireless networks. The errors can be of two types viz. single bit error and burst error.

Single bit error : Only one bit in the data unit changes in single bit error. Single bit error can happen if we are sending data using parallel transmission.
Burst error : Multiple bits are changed in the burst error. Burst errors can be caused by impulse noise.

Principle of Error Detection



Principle of Error Detection

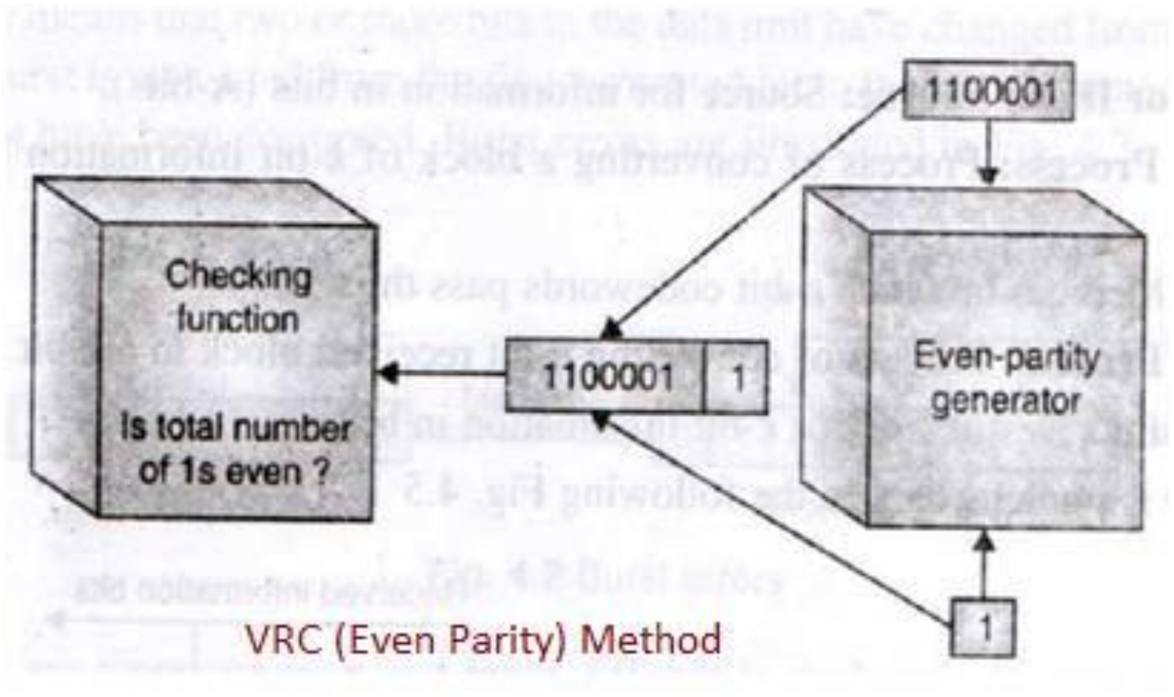
• When frame is transmitted from transmitter to the receiver. There are two possibilities viz. frame is received without error, frame is received in error (i.e. frame is bad).
• Error detection helps in detecting errors in a received block or frame by the receiver.
• Once the error is detected receiver informs the transmitter to re-transmit the same frame again.
• Error detection can be made possible by adding redundant bits in each frame during transmission. Based on all the bits in the frame (i.e. data + error check bits), receiver is capable of detecting errors in the frame.
Following are the explanation of error detection modules.
➤Message (or data) source : Source of information in bits (K bits)
➤Encoding process : Process of converting block of k-bit information to n-bit codeword. n=k .
➤Channel : Medium in which n-bit codewords pass through.
➤Decoding process : Process of converting n-bit received block to k-bit message.
➤Message sink : Destination for k-bit information in bits.

Error Detection Methods

Following are the **error detection methods** or techniques of error detection in networking.

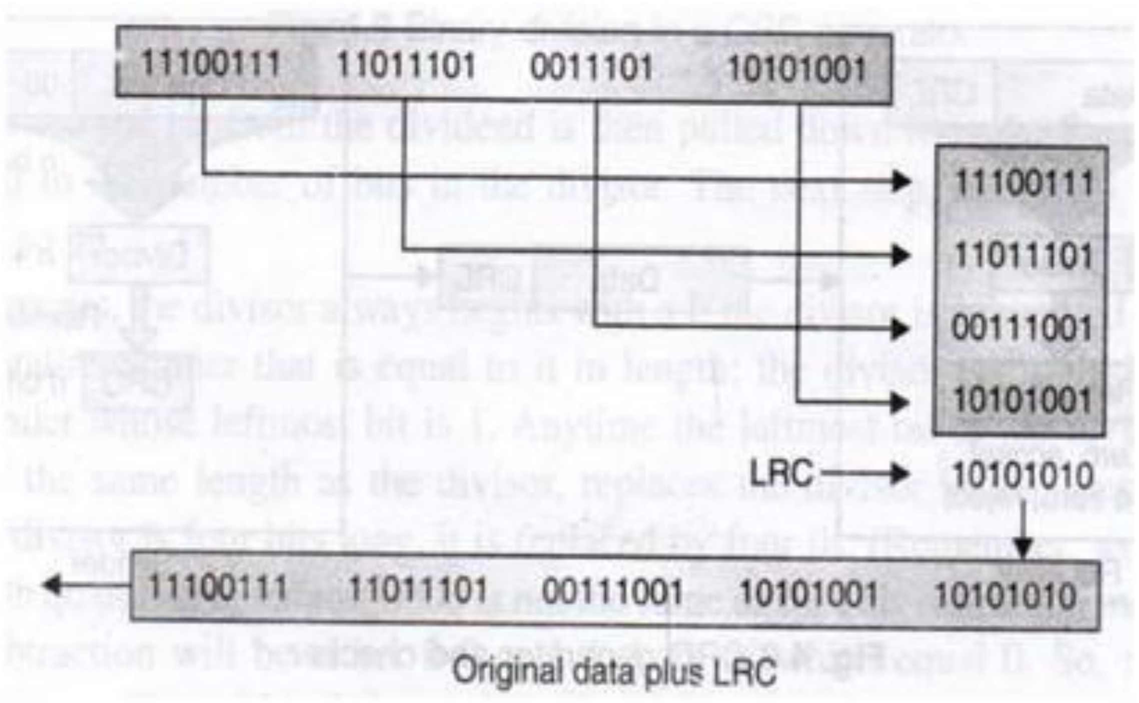
1. VRC Method 2. LRC method 3. CRC method 4. Checksum method

Parity check or vertical redundancy check (VRC) method



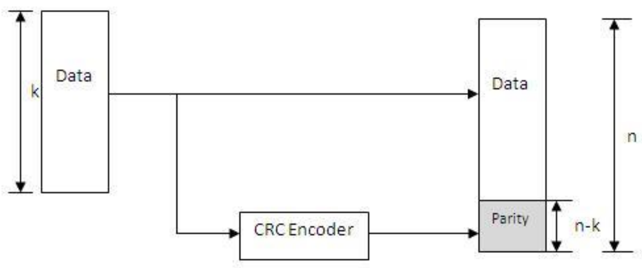
In this error detection technique, a redundant bit called parity bit is appended to every data unit so that total number of 1's in the unit (including parity bit) becomes even. The system now transmits entire extended unit across the network link. At the receiver, all eight received bits are checked through even parity checking function. If it counts even 1's data unit passes. If it counts odd number of 1's, it means error has been introduced in the data somewhere. Hence receiver rejects the whole data unit. Similar way odd parity VRC can also be implemented. In this method, total number of 1's in should be odd before transmission.

Longitudinal Redundancy Check (LRC) method

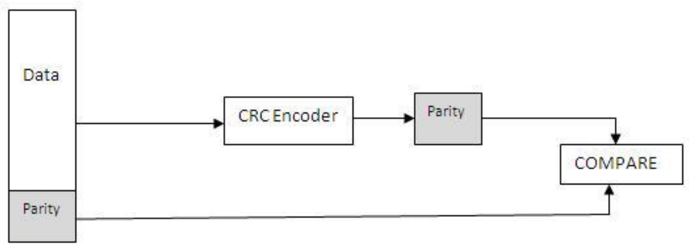


In this error detection method, a block of bits are organized in a table (of rows and columns). For example, instead of sending block of 32 bits, first it is organized into four rows and eight columns. Then parity bits for each column is calculated and new row of eight parity bits is formed. These eight parity bits are appended to original data before transmission.

Following figure-2 depicts CRC addition at transmitter end. CRC is calculated based on received block and compared with CRC appended by transmitter. When calculated CRC and original CRC is equal, frame is considered to be error free. When calculated CRC and original CRC is not equal, frame is said to be erroneous.



Error detection at transmitter end

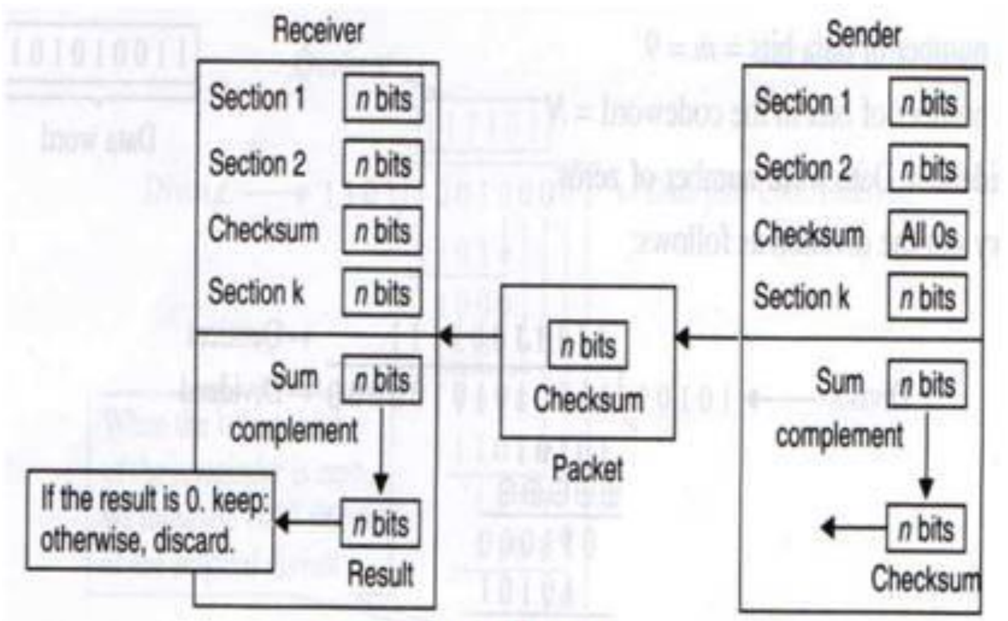


Error detection at receiver end

As shown in the figure, k bits are passed to the encoder block to generate parity bits. This parity bits are added to input data bits and are transmitted as n bits. Hence n-k are parity bits. This happens at the transmitter.

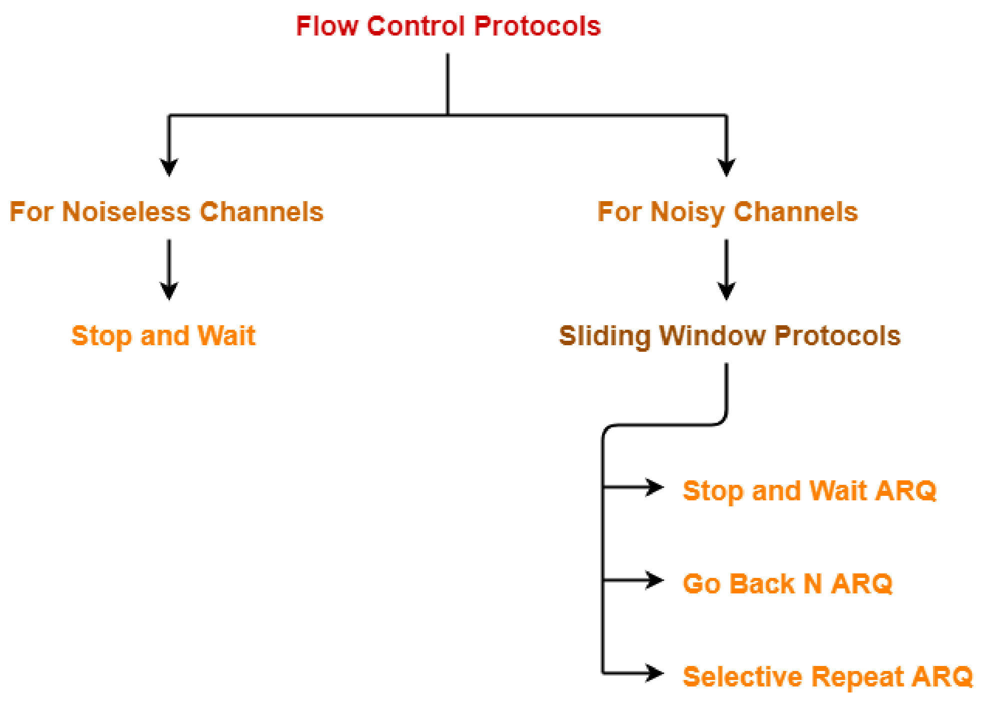
As shown in the figure at the receiver, parity bits along with data bits of total length n bits are passed to the encoder. From the data part CRC is again computed and will be compared with the received CRC bits and based on this data is corrupted or not is decided. This process is called error detection and correction.

Checksum method



Checksum Method

There are two modules in this error detection method viz. checksum generator and checksum checker. In the transmitter, checksum generator subdivides data unit into equal segments of n bits (usually 16). These segments are added together using one's complement arithmatic in such a way that total is also n bits long. The total (i.e. sum) is then complemented and appended to the end of the original data unit as redundancy bits, called checksum field. The extended data unit is transmitted across the network. So it sum of data segment is equal to T, checksum will be -T. The receiver subdivides the data unit as above and adds all segments together and complements the result. If the extended data unit is intact, the total value found by adding all data segments and checksum field should be zero. If the result is not zero, the packet contains error and receiver rejects the packet.



Stop and Wait Protocol-

Stop and Wait Protocol is the simplest flow control protocol.

It works under the following assumptions-

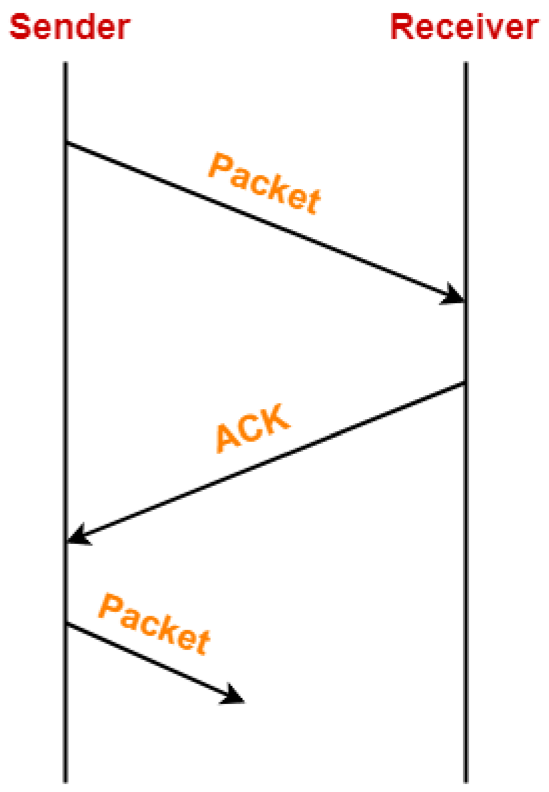
- Communication channel is perfect.
- No error occurs during transmission.

Working-

The working of a stop and wait protocol may be explained as-

- Sender sends a data packet to the receiver.
- Sender stops and waits for the acknowledgement for the sent packet from the receiver.
- Receiver receives and processes the data packet.
- Receiver sends an acknowledgement to the sender.
- After receiving the acknowledgement, sender sends the next data packet to the receiver.

These steps are illustrated below-



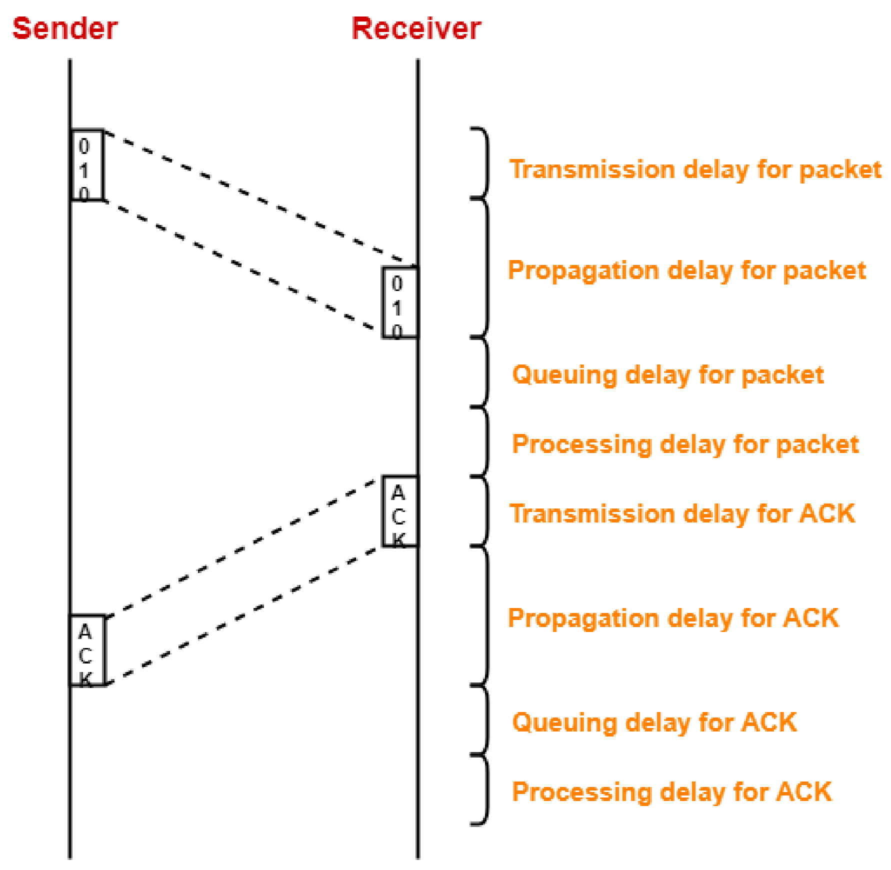
Stop and Wait Protocol

Analysis-

Now, let us analyse in depth how the transmission is actually carried out-

- Sender puts the data packet on the transmission link.
- Data packet propagates towards the receiver's end.
- Data packet reaches the receiver and waits in its buffer.
- Receiver processes the data packet.
- Receiver puts the acknowledgement on the transmission link.
- Acknowledgement propagates towards the sender's end.
- Acknowledgement reaches the sender and waits in its buffer.
- Sender processes the acknowledgement.

These steps are illustrated below-


Sender
Receiver
010
010
ACK
ACK
Transmission delay for packet
Propagation delay for packet
Queuing delay for packet
Processing delay for packet
Transmission delay for ACK
Propagation delay for ACK
Queuing delay for ACK
Processing delay for ACK

Stop and Wait Protocol

Total Time-

Total time taken in sending one data packet

= (Transmission delay + Propagation delay + Queuing delay + Processing delay)$_{\text{packet}}$

+

(Transmission delay + Propagation delay + Queuing delay + Processing delay)$_{\text{ACK}}$

Assume-

- Queuing delay and processing delay to be zero at both sender and receiver side.
- Transmission time for the acknowledgement to be zero since it’s size is very small.

Under the above assumptions.

Total time taken in sending one data packet

= (Transmission delay + Propagation delay)$_{\text{packet}}$ + (Propagation delay)$_{\text{ACK}}$

We know,

- Propagation delay depends on the distance and speed.
- So, it would be same for both data packet and acknowledgement.

So, we have-

Total time taken in sending one data packet

= (Transmission delay)$_{\text{packet}}$ + 2 x Propagation delay

Efficiency-

Efficiency of any flow control control protocol is given by-

Efficiency (η) = Useful Time / Total Time

where-

- Useful time = Transmission delay of data packet = (Transmission delay)$_{\text{packet}}$
- Useless time = Time for which sender is forced to wait and do nothing = 2 x Propagation delay
- Total time = Useful time + Useless time

Thus,

$$\boxed{\text{Efficiency } (\eta) = \frac{(\text{Transmission delay})_{\text{packet}}}{(\text{Transmission delay})_{\text{packet}} + 2 \text{ x Propagation delay}}}$$

OR

$$\boxed{\text{Efficiency } (\eta) = \frac{T_t}{T_t + 2T_p}}$$

OR

$$\boxed{\text{Efficiency } (\eta) = \frac{1}{1 + 2\left(\frac{T_p}{T_t}\right)}}$$

OR

$$\boxed{\text{Efficiency } (\eta) = \frac{1}{1 + 2a} \text{ , where } a = \left(\frac{T_p}{T_t}\right)}$$

Factors Affecting Efficiency-

We know,

Efficiency (η)

= (Transmission delay)$_{\text{packet}}$ / { (Transmission delay)$_{\text{packet}}$ + 2 x Propagation delay }

Dividing numerator and denominator by (Transmission delay)$_{\text{packet}}$, we get-

$$\text{Efficiency } (\eta) = \frac{1}{1 + 2 \times \left(\frac{\text{Propagation delay}}{(\text{Transmission delay})_{\text{packet}}} \right)}$$

$$\text{Efficiency } (\eta) = \frac{1}{1 + 2 \times \left(\frac{\text{Distance}}{\text{speed}} \right) \times \left(\frac{\text{Bandwidth}}{\text{Packet length}} \right)}$$

From here, we can observe-

- Efficiency (η) ∝ 1 / Distance between sender and receiver
- Efficiency (η) ∝ 1 / Bandwidth
- Efficiency (η) ∝ Transmission speed
- Efficiency (η) ∝ Length of data packet

Throughput-

- Number of bits that can be sent through the channel per second is called as its throughput.

$$\boxed{\text{Throughput} = \text{Efficiency } (\eta) \times \text{Bandwidth}}$$

Round Trip Time-

Round Trip Time = 2 x Propagation delay

Advantages-

The advantages of stop and wait protocol are-

- It is very simple to implement.

- The incoming packet from receiver is always an acknowledgement.

Limitations-

The limitations of stop and wait protocol are-

Point-01:

It is extremely inefficient because-

- It makes the transmission process extremely slow.
- It does not use the bandwidth entirely as each single packet and acknowledgement uses the entire time to traverse the link.

Point-02:

If the data packet sent by the sender gets lost, then-

- Sender will keep waiting for the acknowledgement for infinite time.
- Receiver will keep waiting for the data packet for infinite time.

Point-03:

If acknowledgement sent by the receiver gets lost, then-

- Sender will keep waiting for the acknowledgement for infinite time.
- Receiver will keep waiting for another data packet for infinite time.

Important Notes-

Note-01:

Efficiency may also be referred by the following names-

- Line Utilization
- Link Utilization
- Sender Utilization
- Utilization of Sender

Note-02:

Throughput may also be referred by the following names-

- Bandwidth Utilization
- Effective Bandwidth
- Maximum data rate possible
- Maximum achievable throughput

Note-03:

Stop and Wait protocol performs better for LANs than WANs.

This is because-

- Efficiency of the protocol is inversely proportional to the distance between sender and receiver.
- So, the protocol performs better where the distance between sender and receiver is less.
- The distance is less in LANs as compared to WANs.

Stop and Wait ARQ-

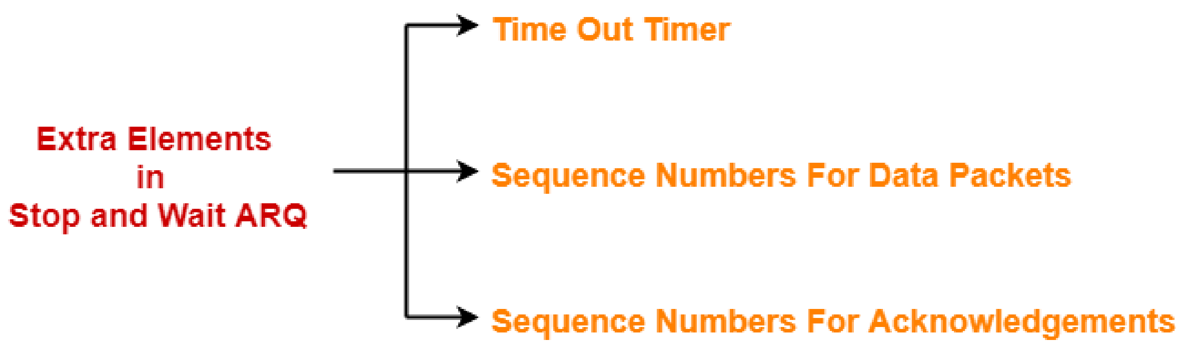
Stop and Wait ARQ is an improved and modified version of Stop and Wait protocol.

Stop and Wait ARQ assumes-

- The communication channel is noisy.
- Errors may get introduced in the data during the transmission.

Working-

- Stop and wait ARQ works similar to stop and wait protocol.
- It provides a solution to all the limitations of stop and wait protocol.
- Stop and wait ARQ includes the following three extra elements.


Extra Elements in Stop and Wait ARQ
Time Out Timer
Sequence Numbers For Data Packets
Sequence Numbers For Acknowledgements

Thus, we can say-

Stop and Wait ARQ

= Stop and Wait Protocol + Time Out Timer + Sequence Numbers for Data Packets and Acknowledgements

Number of Sequence Numbers Required-

NOTE

For any sliding window protocol to work without any problem,

the following condition must be satisfied-

Available Sequence Numbers >= Sender Window Size + Receiver Window Size

Stop and wait ARQ is a one bit sliding window protocol where-

- Sender window size = 1
- Receiver window size = 1

Thus, in stop and wait ARQ,

Minimum number of sequence numbers required

= Sender Window Size + Receiver Window Size

= 1 + 1

= 2

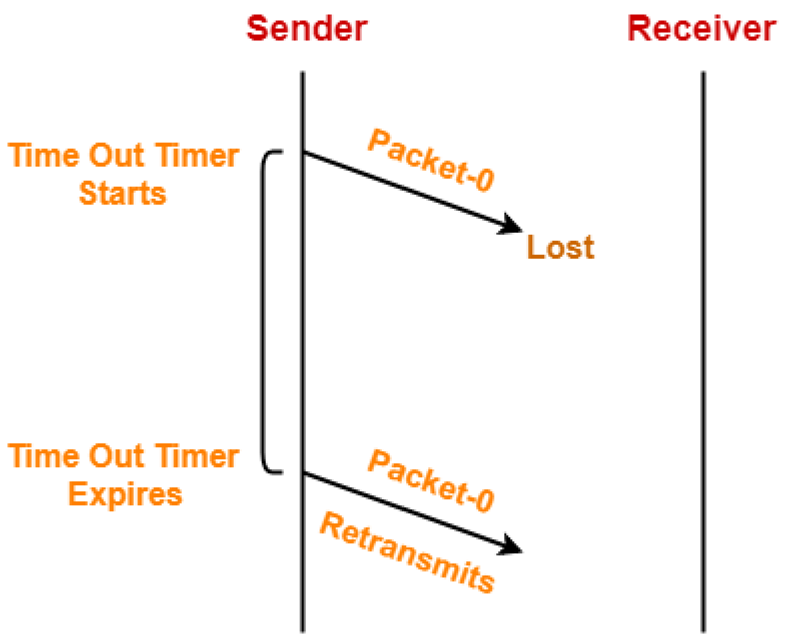
Thus,

- Minimum number of sequence numbers required in Stop and Wait ARQ = 2.
- The two sequence numbers used are 0 and 1.

How Stop and Wait ARQ Solves All Problems?

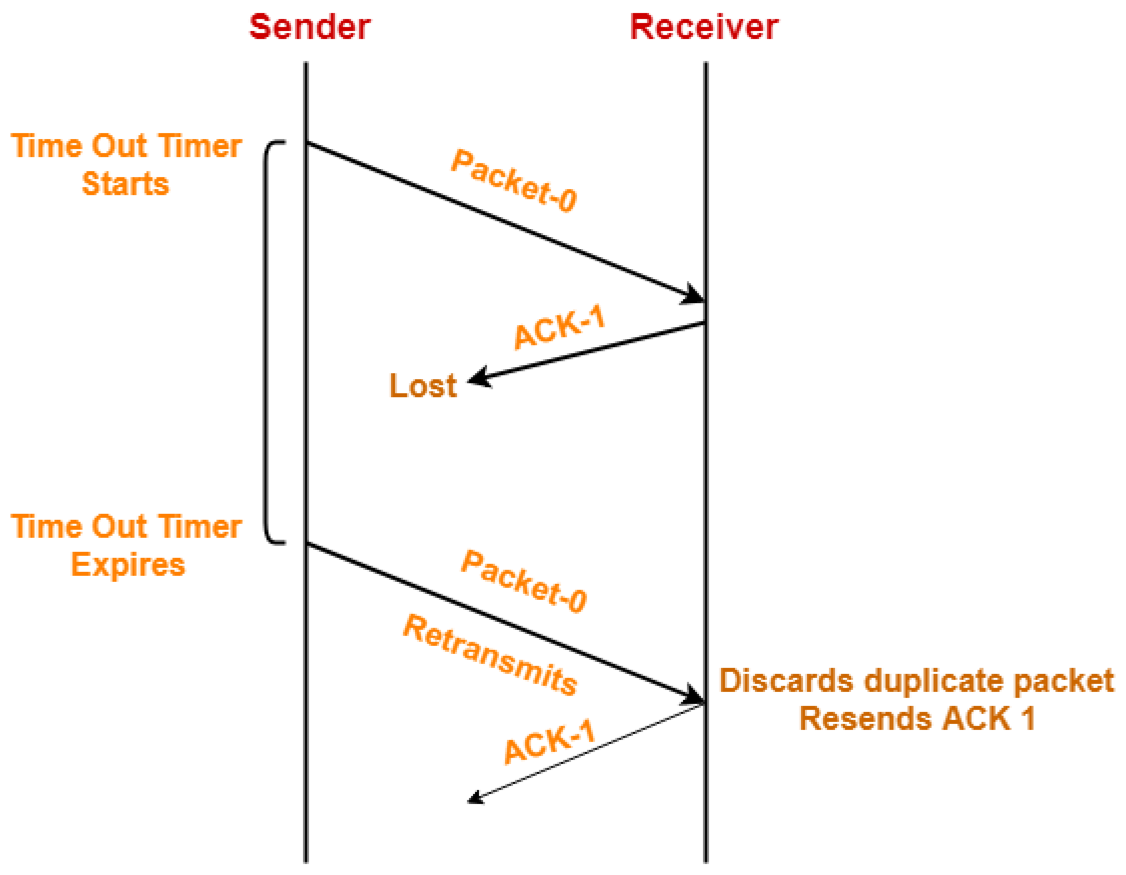
1. Problem of Lost Data Packet-

- Time out timer helps to solve the problem of lost data packet.
- After sending a data packet to the receiver, sender starts the time out timer.
- If the data packet gets acknowledged before the timer expires, sender stops the time out timer.
- If the timer goes off before receiving the acknowledgement, sender retransmits the same data packet.
- After retransmission, sender resets the timer.
- This prevents the occurrence of deadlock.



2. Problem of Lost Acknowledgement-

- Sequence number on data packets help to solve the problem of delayed acknowledgement.
- Consider the acknowledgement sent by the receiver gets lost.
- Then, sender retransmits the same data packet after its timer goes off.
- This prevents the occurrence of deadlock.
- The sequence number on the data packet helps the receiver to identify the duplicate data packet.
- Receiver discards the duplicate packet and re-sends the same acknowledgement.



Role of Sequence Number on Data Packets

Consider the above example-

Step-01:

- Sender sends a data packet with sequence number-0 to the receiver.

Step-02:

- Receiver receives the data packet correctly.
- Receiver now expects data packet with sequence number-1.
- Receiver sends the acknowledgement ACK-1.

Step-03:

- Acknowledgement ACK-1 sent by the receiver gets lost on the way.

Step-04:

- Sender receives no acknowledgement and time out occurs.
- Sender retransmits the same data packet with sequence number-0.
- This will be a duplicate packet for the receiver.

Step-05:

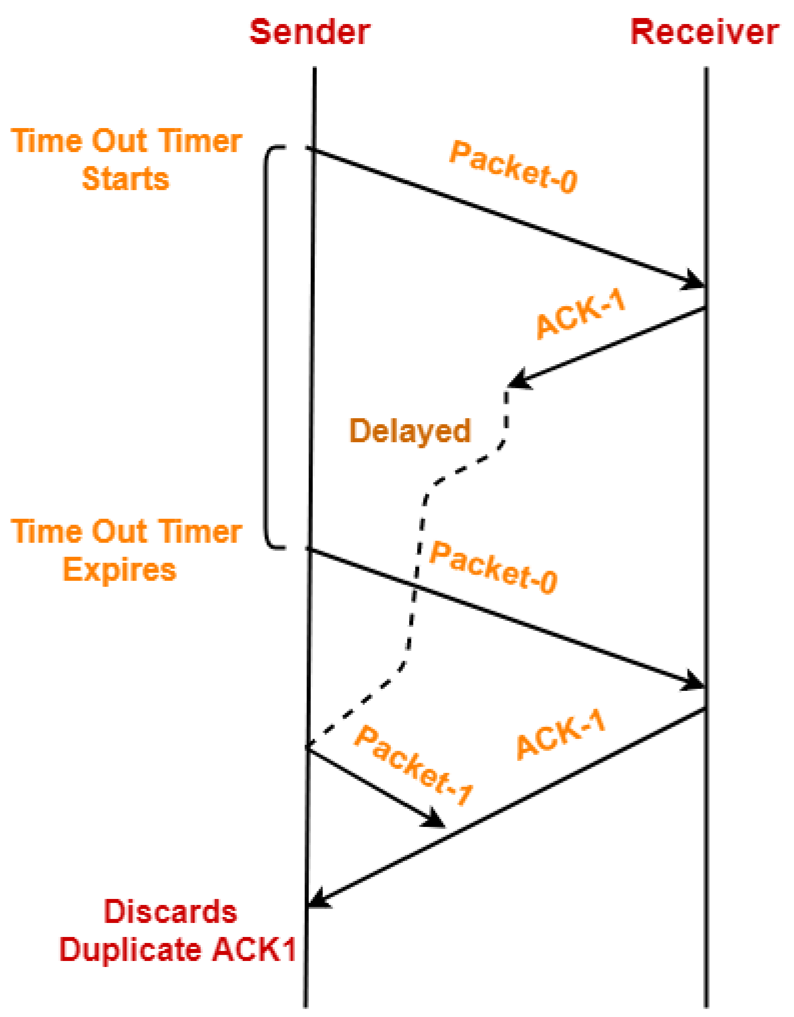
- Receiver receives the data packet and discovers it is the duplicate packet.
- It expects the data packet with sequence number-1 but receiving the data packet with sequence number-0.
- It discards the duplicate data packet and re-sends acknowledgement ACK-1.
- ACK-1 requests the sender to send a data packet with sequence number-1.
- This avoids the inconsistency of data.

Conclusion-

- Had the sequence numbers not been allotted to the data packets, receiver would have accepted the duplicate data packet thinking of it as the new data packet.
- This is how sequence numbers allotted to the data packets prove to be useful for identifying the duplicate data packets and discarding them.

3. Problem of Delayed Acknowledgement-

- Sequence number on acknowledgements help to solve the problem of delayed acknowledgement.



Role of Sequence Number on Acknowledgements

Consider the above example-

Step-01:

- Sender sends a data packet with sequence number-0 to the receiver.

Step-02:

- Receiver receives the data packet correctly.
- Receiver now expects data packet with sequence number-1.
- Receiver sends the acknowledgement ACK-1.

Step-03:

- Acknowledgement ACK-1 sent by the receiver gets delayed in reaching the sender.

Step-04:

- Sender receives no acknowledgement and time out occurs.
- Sender retransmits the same data packet with sequence number-0.
- This will be a duplicate packet for the receiver.

Step-05:

- Receiver receives the data packet and discovers it is the duplicate packet.
- It expects the data packet with sequence number-1 but receiving the data packet with sequence number-0.
- It discards the duplicate data packet and re-sends acknowledgement ACK-1.
- ACK-1 requests the sender to send a data packet with sequence number-1.

Step-06:

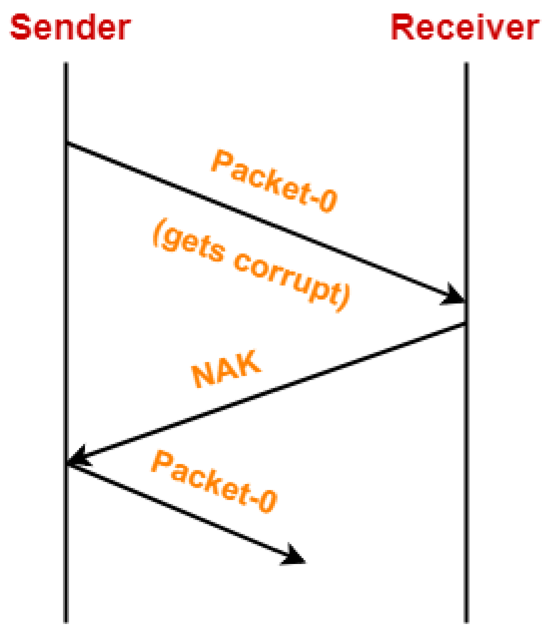
- Two acknowledgements ACK1 reaches the sender.
- When first acknowledgement ACK1 reaches the sender, sender sends the next data packet with sequence number 1.
- When second acknowledgement ACK1 reaches the sender, sender rejects the duplicate acknowledgement.
- This is because it has already sent the data packet with sequence number-1 and now sender expects the acknowledgement with sequence number 0 from the receiver.

Conclusion-

- Had the sequence numbers not been allotted to the acknowledgements, sender would have accepted the duplicate acknowledgement thinking of it as the new acknowledgement for the latest data packet sent by it.
- This is how sequence numbers allotted to the acknowledgements prove to be useful for identifying duplicate acknowledgements and discarding them.

4. Problem of Damaged Packet-

- If receiver receives a corrupted data packet from the sender, it sends a negative acknowledgement (NAK) to the sender.
- NAK requests the sender to send the data packet again.



Stop and Wait Protocol Vs Stop and Wait ARQ-

The following comparison table states the differences between the two protocols-

Stop and Wait Protocol	**Stop and Wait ARQ**
It assumes that the communication channel is perfect and noise free.	It assumes that the communication channel is imperfect and noisy.
Data packet sent by the sender can never get corrupt.	Data packet sent by the sender may get corrupt.
There is no concept of negative acknowledgements.	A negative acknowledgement is sent by the receiver if the data packet is found to be corrupt.

There is no concept of time out timer.	Sender starts the time out timer after sending the data packet.
There is no concept of sequence numbers.	Data packets and acknowledgements are numbered using sequence numbers.

Limitation of Stop and Wait ARQ-

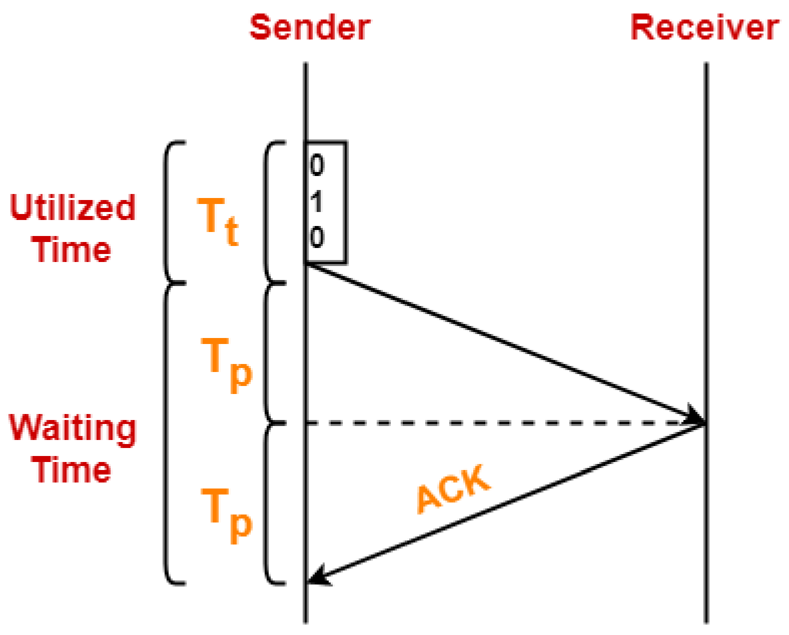
The major limitation of Stop and Wait ARQ is its very less efficiency.

To increase the efficiency, protocols like **Go back N** and **Selective Repeat** are used.

Explanation-

In stop and wait ARQ,

- Sender window size is 1.
- This allows the sender to keep only one frame unacknowledged.
- So, sender sends one frame and then waits until the sent frame gets acknowledged.
- After receiving the acknowledgement from the receiver, sender sends the next frame.



Here,

- Sender uses T_t time for transmitting the packet over the link.
- Then, sender waits for 2 x T_p time.
- After 2 x T_p time, sender receives the acknowledgement for the sent frame from the receiver.
- Then, sender sends the next frame.
- This 2 x T_p waiting time is the actual cause of less efficiency.

Efficiency Improvement-

- The efficiency of stop and wait ARQ can be improved by increasing the window size.
- This allows the sender to keep more than one unacknowledged frame in its window.
- Thus, sender can send frames in the waiting time too.

PRACTICE PROBLEMS BASED ON STOP AND WAIT PROTOCOL-

Problem-01:

If the bandwidth of the line is 1.5 Mbps, RTT is 45 msec and packet size is 1 KB, then find the link utilization in stop and wait.

Solution-

Given-

- Bandwidth = 1.5 Mbps
- RTT = 45 msec
- Packet size = 1 KB

Calculating Transmission Delay-

Transmission delay (T_t)

= Packet size / Bandwidth

= 1 KB / 1.5 Mbps

= (2^{10} x 8 bits) / (1.5 x 10^6 bits per sec)

= 5.461 msec

Calculating Propagation Delay-

Propagation delay (T_p)

= Round Trip Time / 2

= 45 msec / 2

= 22.5 msec

Calculating Value Of 'a'-

$a = T_p / T_t$

a = 22.5 msec / 5.461 msec

a = 4.12

Calculating Link Utilization-

Link Utilization or Efficiency (η)

= 1 / 1+2a

= 1 / (1 + 2 x 4.12)

= 1 / 9.24

= 0.108

= 10.8 %

Problem-02:

A channel has a bit rate of 4 Kbps and one way propagation delay of 20 msec. The channel uses stop and wait protocol. The transmission time of the acknowledgement frame is negligible. To get a channel efficiency of at least 50%, the minimum frame size should be-

1. 80 bytes
2. 80 bits
3. 160 bytes

4. 160 bits

Solution-

Given-

- Bandwidth = 4 Kbps
- Propagation delay (T_p) = 20 msec
- Efficiency >= 50%

Let the required frame size = L bits.

Calculating Transmission Delay-

Transmission delay (T_t)

= Packet size / Bandwidth

= L bits / 4 Kbps

Calculating Value Of 'a'-

a = T_p / T_t

a = 20 msec / (L bits / 4 Kbps)

a = (20 msec x 4 Kbps) / L bits

Condition For Efficiency To Be At least 50%-

For efficiency to be at least 50%, we must have-

1 / 1+2a >= 1/2

a <= 1/2

Substituting the value of 'a', we get-

(20 msec x 4 Kbps) / L bits <= 1/2

L bits >= (20 msec x 4 Kbps) x 2

L bits >= $(20 \times 10^{-3}$ sec $\times 4 \times 10^{3}$ bits per sec$) \times 2$

L bits >= 20×4 bits $\times 2$

L >= 160

From here, frame size must be at least 160 bits.

Thus, Correct Option is (D).

Problem-03:

What is the throughput achievable in stop and wait protocol by a maximum packet size of 1000 bytes and network span of 10 km.

Assume the speed of light in cable is 70% of the speed of light in vaccum.

Solution-

We have-

$$\text{Throughput} = \text{Efficiency} \times \text{Bandwidth}$$

$$\text{Throughput} = \frac{T_t}{T_t + 2 \times T_p} \times \text{Bandwidth}$$

$$\text{Throughput} = \frac{L / B}{T_t + 2 \times d / v} \times B$$

$$\text{Throughput} = \frac{L}{2 \times d / v}$$

- In the given question, we are not provided with the network's bandwidth.
- So, in the above formula of throughput, we have ignored the term T_t from the denominator.
- Although it is incorrect, but we still ignore it for solving the question.

Now, Given-

- L = 1000 bytes
- d = 10 km = 10^4 m
- v = 70% of 3 x 10^8 m/sec = 2.1 x 10^8 m/sec

Substituting the values in the above relation, we get-

Throughput

= 1000 bytes / [2 x 10^4 m / (2.1 x 10^8 m/sec)]

= 1.05 x 10^7 bytes per sec

= 10.5 MBps

Problem-04:

If the packet size is 1 KB and propagation time is 15 msec, the channel capacity is 10^9 b/sec, then find the transmission time and utilization of sender in stop and wait protocol.

Solution-

Given-

- Packet size = 1 KB
- Propagation time (T_p) = 15 msec
- Channel capacity = Bandwidth (here) = 10^9 b/sec

NOTE-

- Generally, channel capacity is the total number of bits which a channel can hold. So, its unit is bits.
- But here, channel capacity is actually given as bandwidth because its unit is b/sec.

Calculating Transmission Delay-

Transmission delay (T_t)

= Packet size / Bandwidth

= 1 KB / 10^9 bits per sec

= 2^{10} bits / 10^9 bits per sec

= 1.024 µsec

Calculating Value Of ‘a’-

a = T_p / T_t

a = 15 msec / 1.024 µsec

a = 15000 µsec / 1.024 µsec

a = 14648.46

Calculating Sender Utilization-

Sender Utilization or Efficiency (η)

= 1 / 1+2a

= 1 / (1 + 2 x 1468.46)

= 1 / 29297.92

= 0.0000341

= 0.00341 %

Problem-05:

Consider a MAN with average source and destination 20 Km apart and one way delay of 100 µsec. At what data rate does the round trip delay equals the transmission delay for a 1 KB packet?

Solution-

Given-

- Distance = 20 Km
- Propagation delay (T_p) = 100 µsec
- Packet size = 1 KB

We need to have-

Round Trip Time = Transmission delay

2 x Propagation delay = Transmission delay

Substituting the values in the above relation, we get-

2 x 100 µsec = 1 KB / Bandwidth

Bandwidth = 1 KB / 200 µsec

Bandwidth = (2^{10} x 10^6 / 200) bytes per sec

Bandwidth = 5.12 MBps or 40.96 Mbps

Problem-06:

Consider two hosts X and Y connected by a single direct link of rate 10^6 bits/sec. The distance between the two hosts is 10,000 km and the propagation speed along the link is 2 x 10^8 m/sec. Host X sends a file of 50,000 bytes as one large message to host Y continuously. Let the transmission and propagation delays be p milliseconds and q milliseconds respectively.

Then the value of p and q are-

1. p = 50 and q = 100
2. p = 50 and q = 400
3. p = 100 and q = 50
4. p = 400 and q = 50

Solution-

Given-

- Bandwidth = 10^6 bits/sec
- Distance = 10,000 km
- Propagation speed = 2 x 10^8 m/sec
- Packet size = 50,000 bytes

Calculating Transmission Delay-

Transmission delay (T_t)

= Packet size / Bandwidth

= 50000 bytes / 10^6 bits per sec

= (5 x 10^4 x 8 bits) / 10^6 bits per sec

= (4 x 10^5 bits) / 10^6 bits per sec

= 0.4 sec

= 400 msec

Calculating Propagation Delay-

Propagation delay (T_p)

= Distance / Propagation speed

= 10000 km / (2 x 10^8 m/sec)

= 10^7 m / (2 x 10^8 m/sec)

= 50 msec

Thus, Option (D) is correct.

Problem-07:

The values of parameters for the stop and wait ARQ protocol are as given below-

- Bit rate of the transmission channel = 1 Mbps
- Propagation delay from sender to receiver = 0.75 ms
- Time to process a frame = 0.25 ms
- Number of bytes in the information frame = 1980
- Number of bytes in the acknowledge frame = 20
- Number of overhead bytes in the information frame = 20

Assume that there are no transmission errors. Then the transmission efficiency (in %) of the stop and wait ARQ protocol for the above parameters is ____________ . (correct to 2 decimal places)

Solution-

Given-

- Bandwidth = 1 Mbps

- Propagation delay (T_p) = 0.75 ms
- Processing time ($T_{process}$) = 0.25 ms
- Data frame size = 1980 bytes
- Acknowledgement frame size = 20 bytes
- Overhead in data frame = 20 bytes

Calculating Useful Time-

Useful data sent

= Transmission delay of useful data bytes sent

= Useful data bytes sent / Bandwidth

= (1980 bytes – 20 bytes) / 1 Mbps

= 1960 bytes / 1 Mbps

= (1960 x 8 bits) / (10^6 bits per sec)

= 15680 μsec

= 15.680 msec

Calculating Total Time-

Total time

= Transmission delay of data frame + Propagation delay of data frame + Processing delay of data frame + Transmission delay of acknowledgement + Propagation delay of acknowledgement

= (1980 bytes / 1 Mbps) + 0.75 msec + 0.25 msec + (20 bytes / 1 Mbps) + 0.75 msec

= 15.840 msec + 0.75 msec + 0.25 msec + 0.160 msec + 0.75 msec

= 17.75 msec

Calculating Efficiency-

Efficiency (η)

= Useful time / Total time

= 15.680 msec / 17.75 msec

= 0.8833

= 88.33%

Problem-08:

A sender uses the stop and wait ARQ protocol for reliable transmission of frames. Frames are of size 1000 bytes and the transmission rate at the sender is 80 Kbps. Size of an acknowledgement is 100 bytes and the transmission rate at the receiver is 8 Kbps. The one way propagation delay is 100 msec.

Assuming no frame is lost, the sender throughput is __________ bytes/sec.

Solution-

Given-

- Frame size = 1000 bytes
- Sender bandwidth = 80 Kbps
- Acknowledgement size = 100 bytes
- Receiver bandwidth = 8 Kbps
- Propagation delay (T_p) = 100 msec

Calculating Transmission Delay Of Data Frame-

Transmission delay (T_t)

= Frame size / Sender bandwidth

= 1000 bytes / 80 Kbps

= (1000 x 8 bits) / (80 x 10^3 bits per sec)

= 0.1 sec

= 100 msec

Calculating Transmission Delay Of Acknowledgement-

Transmission delay (T_t)

= Acknowledgement size / Receiver bandwidth

= 100 bytes / 8 Kbps

= (100 x 8 bits) / (8 x 10^3 bits per sec)

= 100 msec

Calculating Useful Time-

Useful Time

= Transmission delay of data frame

= 100 msec

Calculating Total Time-

Total Time

= Transmission delay of data frame + Propagation delay of data frame + Transmission delay of acknowledgement + Propagation delay of acknowledgement

= 100 msec + 100 msec + 100 msec + 100 msec

= 400 msec

Calculating Efficiency-

Efficiency (η)

= Useful time / Total time

= 100 msec / 400 msec

= 1 / 4

= 25%

Calculating Sender Throughput-

Sender throughput

= Efficiency (η) x Sender bandwidth

= 0.25 x 80 Kbps

= 20 Kbps

= (20 x 1000 / 8) bytes per sec

= 2500 bytes/sec

Sliding Window Protocol-

- Sliding window protocol is a flow control protocol.
- It allows the sender to send multiple frames before needing the acknowledgements.
- Sender slides its window on receiving the acknowledgements for the sent frames.
- This allows the sender to send more frames.
- It is called so because it involves sliding of sender's window.

Maximum number of frames that sender can send without acknowledgement

= Sender window size

Optimal Window Size-

In a sliding window protocol, optimal sender window size = 1 + 2a

Derivation-

We know,

$$\text{Efficiency } (\eta) = \frac{T_t}{T_t + 2 \times T_p}$$

To get 100% efficiency, we must have-

$\eta = 1$

$T_t / (T_t + 2T_p) = 1$

$T_t = T_t + 2T_p$

Thus,

- To get 100% efficiency, transmission time must be $T_t + 2T_p$ instead of T_t.
- This means sender must send the frames in waiting time too.
- Now, let us find the maximum number of frames that can be sent in time $T_t + 2T_p$.

We have-

- In time T_t, sender sends one frame.
- Thus, In time $T_t + 2T_p$, sender can send $(T_t + 2T_p) / T_t$ frames i.e. 1+2a frames.

Thus, to achieve 100% efficiency, window size of the sender must be 1+2a.

Required Sequence Numbers-

- Each sending frame has to be given a unique sequence number.
- Maximum number of frames that can be sent in a window = 1+2a.
- So, minimum number of sequence numbers required = 1+2a.

o have 1+2a sequence numbers,

Minimum number of bits required in sequence number field = $\lceil \log_2(1+2a) \rceil$

NOTE-

- When minimum number of bits is asked, we take the ceil.
- When maximum number of bits is asked, we take the floor.

Choosing a Window Size-

The size of the sender's window is bounded by-

1. Receiver's Ability-

- Receiver's ability to process the data bounds the sender window size.

- If receiver can not process the data fast, sender has to slow down and not transmit the frames too fast.

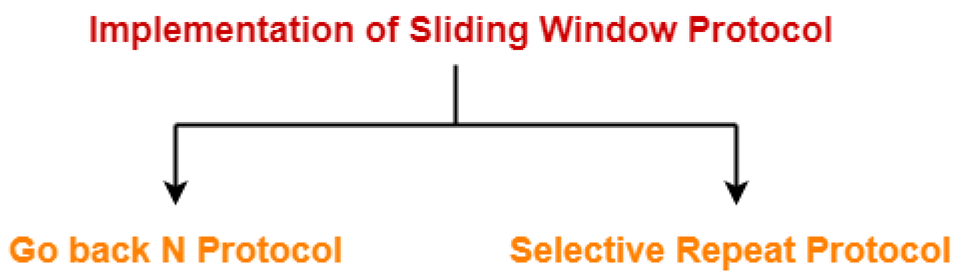
2. Sequence Number Field-

- Number of bits available in the sequence number field also bounds the sender window size.
- If sequence number field contains n bits, then 2^n sequence numbers are possible.
- Thus, maximum number of frames that can be sent in one window = 2^n.

For n bits in sequence number field, Sender Window Size = min $(1+2a , 2^n)$

Implementations of Sliding Window Protocol-

The two well known implementations of sliding window protocol are-


Implementation of Sliding Window Protocol
Go back N Protocol
Selective Repeat Protocol

1. **Go back N Protocol**
2. **Selective Repeat Protocol**

Efficiency-

Efficiency of any flow control protocol may be expressed as-

$$\boxed{\text{Efficiency } (\eta) = \frac{\text{Number of frames sent in one window}}{\text{Total number of frames that can be sent in one window}}}$$

OR

$$\boxed{\text{Efficiency } (\eta) = \frac{\text{Sender Window Size in the Protocol}}{\text{Optimal Sender Window Size}}}$$

OR

$$\boxed{\text{Efficiency } (\eta) = \frac{\text{Sender Window Size in the Protocol}}{1 + 2a}}$$

Example-

In **Stop and Wait ARQ**, sender window size = 1.

Thus,

Efficiency of Stop and Wait ARQ = 1 / 1+2a

PRACTICE PROBLEMS BASED ON SLIDING WINDOW PROTOCOL-

Problem-01:

If transmission delay and propagation delay in a sliding window protocol are 1 msec and 49.5 msec respectively, then-

1. What should be the sender window size to get the maximum efficiency?
2. What is the minimum number of bits required in the sequence number field?
3. If only 6 bits are reserved for sequence numbers, then what will be the efficiency?

Solution-

Given-

- Transmission delay = 1 msec
- Propagation delay = 49.5 msec

Part-01:

To get the maximum efficiency, sender window size

= 1 + 2a

= 1 + 2 x (T_p / T_t)

= 1 + 2 x (49.5 msec / 1 msec)

= 1 + 2 x 49.5

= 100

Thus,

For maximum efficiency, sender window size = 100

Part-02:

Minimum number of bits required in the sequence number field

= $\lceil \log_2(1+2a) \rceil$

= $\lceil \log_2(100) \rceil$

= $\lceil 6.8 \rceil$

= 7

Thus,

Minimum number of bits required in the sequence number field = 7

Part-03:

If only 6 bits are reserved in the sequence number field, then-

Maximum sequence numbers possible = 2^6 = 64

Now,

Efficiency

= Sender window size in the protocol / Optimal sender window size

= 64 / 100

= 0.64

= 64%

Problem-02:

If transmission delay and propagation delay in a sliding window protocol are 1 msec and 99.5 msec respectively, then-

1. What should be the sender window size to get the maximum efficiency?
2. What is the minimum number of bits required in the sequence number field?
3. If only 7 bits are reserved for sequence numbers, then what will be the efficiency?

Solution-

Given-

- Transmission delay = 1 msec
- Propagation delay = 99.5 msec

Part-01:

To get the maximum efficiency, sender window size

= 1 + 2a

= 1 + 2 x (T_p / T_t)

= 1 + 2 x (99.5 msec / 1 msec)

= 1 + 2 x 99.5

= 200

Thus,

For maximum efficiency, sender window size = 200

Part-02:

Minimum number of bits required in the sequence number field

= $\lceil \log_2(1+2a) \rceil$

= $\lceil \log_2(200) \rceil$

= ⌈7.64⌉

= 8

Thus,

Minimum number of bits required in the sequence number field = 8

Part-03:

If only 6 bits are reserved in the sequence number field, then-

Maximum sequence numbers possible = 2^7 = 128

Now,

Efficiency

= Sender window size in the protocol / Optimal sender window size

= 128 / 200

= 0.64

= 64%

PRACTICE PROBLEMS BASED ON SLIDING WINDOW PROTOCOL-

Problem-01:

A 3000 km long trunk operates at 1.536 Mbps and is used to transmit 64 byte frames and uses sliding window protocol. If the propagation speed is 6 μsec / km, how many bits should the sequence number field be?

Solution-

Given-

- Distance = 3000 km
- Bandwidth = 1.536 Mbps
- Packet size = 64 bytes
- Propagation speed = 6 μsec / km

Calculating Transmission Delay-

Transmission delay (T_t)

= Packet size / Bandwidth

= 64 bytes / 1.536 Mbps

= (64 x 8 bits) / (1.536 x 10^6 bits per sec)

= 333.33 μsec

Calculating Propagation Delay-

For 1 km, propagation delay = 6 μsec

For 3000 km, propagation delay = 3000 x 6 μsec = 18000 μsec

Calculating Value Of ‘a’-

a = T_p / T_t

a = 18000 μsec / 333.33 μsec

a = 54

Calculating Bits Required in Sequence Number Field-

Bits required in sequence number field

= $\lceil \log_2(1+2a) \rceil$

= $\lceil \log_2(1 + 2 \times 54) \rceil$

= $\lceil \log_2(109) \rceil$

= $\lceil 6.76 \rceil$

= 7 bits

Thus,

- Minimum number of bits required in sequence number field = 7
- With 7 bits, number of sequence numbers possible = 128
- We use only (1+2a) = 109 sequence numbers and rest remains unused.

<u>**Problem-02:**</u>

Compute approximate optimal window size when packet size is 53 bytes, RTT is 60 msec and bottleneck bandwidth is 155 Mbps.

<u>**Solution-**</u>

Given-

- Packet size = 53 bytes
- RTT = 60 msec
- Bandwidth = 155 Mbps

<u>Calculating Transmission Delay-</u>

Transmission delay (T_t)

= Packet size / Bandwidth

= 53 bytes / 155 Mbps

= (53 x 8 bits) / (155 x 10^6 bits per sec)

= 2.735 μsec

<u>Calculating Propagation Delay-</u>

Propagation delay (T_p)

= Round Trip Time / 2

= 60 msec / 2

= 30 msec

<u>Calculating Value of ‘a’-</u>

$a = T_p / T_t$

a = 30 msec / 2.735 μsec

a = 10968.921

Calculating Optimal Window Size-

Optimal window size

= 1 + 2a

= 1 + 2 x 10968.921

= 21938.84

Thus, approximate optimal window size = 21938 frames.

Problem-03:

A sliding window protocol is designed for a 1 Mbps point to point link to the moon which has a one way latency (delay) of 1.25 sec. Assuming that each frame carries 1 KB of data, what is the minimum number of bits needed for the sequence number?

Solution-

Given-

- Bandwidth = 1 Mbps
- Propagation delay (T_p) = 1.25 sec
- Packet size = 1 KB

Calculating Transmission Delay-

Transmission delay (T_t)

= Packet size / Bandwidth

= 1 KB / 1 Mbps

= (2^{10} x 8 bits) / (10^6 bits per sec)

= 8.192 msec

Calculating Value of ‘a’-

a = T_p / T_t

a = 1.25 sec / 8.192 msec

a = 152.59

Calculating Bits Required in Sequence Number Field-

Bits required in sequence number field

$= \lceil \log_2(1+2a) \rceil$

$= \lceil \log_2(1 + 2 \times 152.59) \rceil$

$= \lceil \log_2(306.176) \rceil$

$= \lceil 8.25 \rceil$

= 9 bits

Thus,

- Minimum number of bits required in sequence number field = 9
- With 9 bits, number of sequence numbers possible = 512.
- We use only (1+2a) sequence numbers and rest remains unused.

Problem-04:

Host A is sending data to host B over a full duplex link. A and B are using the sliding window protocol for flow control. The send and receive window sizes are 5 packets each. Data packets (sent only from A to B) are all 1000 bytes long and the transmission time for such a packet is 50 µs. Acknowledgement packets (sent only from B to A) are very small and require negligible transmission time. The propagation delay over the link is 200 µs. What is the maximum achievable throughput in this communication?

1. 7.69×10^6 Bps
2. 11.11×10^6 Bps
3. 12.33×10^6 Bps
4. 15.00×10^6 Bps

Solution-

Given-

- Sender window size = Receiver window size = 5
- Packet size = 1000 bytes
- Transmission delay (T_t) = 50 µs

- Propagation delay (T_p) = 200 μs

<u>Calculating Bandwidth-</u>

We know,

Transmission delay = Packet size / Bandwidth

So, Bandwidth

= Packet Size / Transmission delay (T_t)

= 1000 bytes / 50 μs

= (1000 x 8 bits) / (50 x 10^{-6} sec)

= 160 Mbps

<u>Calculating Value of ‘a’-</u>

a = T_p / T_t

a = 200 μsec / 50 μsec

a = 4

<u>Calculating Optimal Window Size-</u>

Optimal window size

= 1 + 2a

= 1 + 2 x 4

= 9

<u>Calculating Efficiency-</u>

Efficiency (η)

= Sender window size / Optimal window size

= 5 / 9

= 0.5555

= 55.55%

<u>Calculating Maximum Achievable Throughput-</u>

Maximum achievable throughput

= Efficiency (η) x Bandwidth

= 0.5555 x 160 Mbps

= 88.88 Mbps

= 88.88 x 10^6 bps or 11.11 x 10^6 Bps

Thus, Option (B) is correct.

Problem-05:

Station A uses 32 byte packets to transmit messages to station B using a sliding window protocol. The round trip delay between A and B is 80 msec and the bottleneck bandwidth on the path between A and B is 128 Kbps. What is the optimal window size that A should use?

1. 20
2. 40
3. 160
4. 320

Solution-

Given-

- Packet size = 32 bytes
- Round Trip Time = 80 msec
- Bandwidth = 128 Kbps

<u>Calculating Transmission Delay-</u>

Transmission delay (T_t)

= Packet size / Bandwidth

= 32 bytes / 128 Kbps

= (32 x 8 bits) / (128 x 10^3 bits per sec)

= 2 msec

Calculating Propagation Delay-

Propagation delay (T_p)

= Round Trip Time / 2

= 80 msec / 2

= 40 msec

Calculating Value of ‘a’-

a = T_p / T_t

a = 40 msec / 2 msec

a = 20

Calculating Optimal Window Size-

Optimal window size

= 1 + 2a

= 1 + 2 x 20

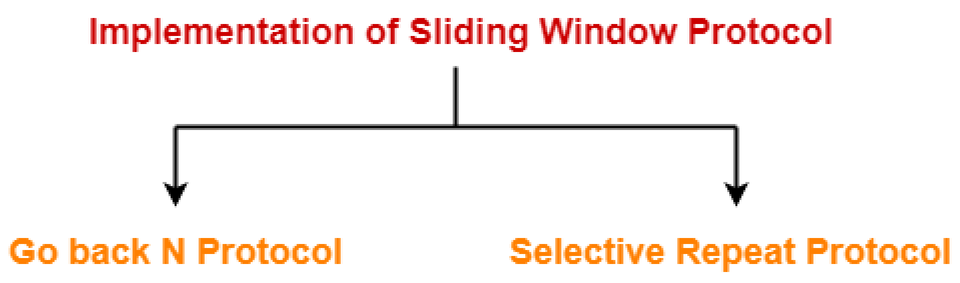
= 41 which is close to option (B)

Thus, Option (B) is correct.

Sliding Window Protocol-

Before you go through this article, make sure that you have gone through the previous article on **Sliding Window Protocol**.

The two well known implementations of sliding window protocol are-


Implementation of Sliding Window Protocol
Go back N Protocol
Selective Repeat Protocol

1. Go back N Protocol
2. Selective Repeat Protocol

Go back N Protocol-

Go back N protocol is an implementation of a sliding window protocol.

The features and working of this protocol are explained in the following points-

Point-01:

In Go back N, sender window size is N and receiver window size is always 1.

In Go back N,

- Sender window size = N. Example in Go back 10, sender window size will be 10.
- Receiver window size is always 1 for any value of N.

Point-02:

Go back N uses cumulative acknowledgements.

In Go back N,

- Receiver maintains an acknowledgement timer.
- Each time the receiver receives a new frame, it starts a new acknowledgement timer.

- After the timer expires, receiver sends the cumulative acknowledgement for all the frames that are unacknowledged at that moment.

NOTE-

- A new acknowledgement timer does not start after the expiry of old acknowledgement timer.
- It starts after a new frame is received.

Point-03:

Go back N may use independent acknowledgements too.

- The above point does not mean that Go back N can not use independent acknowledgements.
- Go back N may use independent acknowledgements too if required.
- The kind of acknowledgement used depends on the expiry of acknowledgement timer.

Example-

- Consider after the expiry of acknowledgement timer, there is only one frame left to be acknowledged.
- Then, Go back N sends the independent acknowledgement for that frame.

Point-04:

Go back N does not accept the corrupted frames and silently discards them.

In Go back N,

- If receiver receives a frame that is corrupted, then it silently discards that frame.
- The correct frame is retransmitted by the sender after the time out timer expires.
- Silently discarding a frame means-

"Simply rejecting the frame and not taking any action"

(like not sending a NACK to the sender to send the correct frame)

Point-05:

Go back N does not accept out of order frames and silently discards them.

In Go back N,

- If receiver receives a frame whose sequence number is not what the receiver expects, then it silently discards that frame.
- All the following frames are also discarded.
- This is because receiver window size is 1 and therefore receiver can not accept out of order frames.

Point-06:

Go back N leads to retransmission of entire window if for any frame, no ACK is received by the sender.

In Go back N,

- Receiver silently discards the frame if it founds the frame to be either corrupted or out of order.
- It does not send any acknowledgement for such frame.
- It silently discards the following frames too.

Thus,

- If for any particular frame, sender does not receive any acknowledgement, then it understands that along with that frame, all the following frames must also have been discarded by the receiver.
- So, sender has to retransmit all the following frames too along with that particular frame.
- Thus, it leads to the retransmission of entire window.
- That is why, the protocol has been named as "**Go back N**".

Point-07:

Go back N leads to retransmission of lost frames after expiry of time out timer.

In Go back N,

- Consider a frame being sent to the receiver is lost on the way.
- Then, it is retransmitted only after time out timer expires for that frame at sender's side.

Efficiency of Go back N-

Efficiency of any flow control protocol is given by-

Efficiency = Sender Window Size in Protocol / (1 + 2a)

In Go back N protocol, sender window size = N.

Thus,

Efficiency of Go back N = N / (1 + 2a)

Go Back N Protocol-

Before you go through this article, make sure that you have gone through the previous article on **Go back N Protocol**.

We have discussed-

- **Sliding window protocols** allow the sender to send multiple frames before needing acknowledgements.
- Go back N is an implementation of a sliding window protocol.

In this article, we will discuss practice problems based on Go back N protocol.

PRACTICE PROBLEMS BASED ON GO BACK N PROTOCOL-

Problem-01:

A 20 Kbps satellite link has a propagation delay of 400 ms. The transmitter employs the “go back n ARQ” scheme with n set to 10.

Assuming that each frame is 100 bytes long, what is the maximum data rate possible?

1. 5 Kbps
2. 10 Kbps
3. 15 Kbps
4. 20 Kbps

Solution-

Given-

- Bandwidth = 20 Kbps
- Propagation delay (T_p) = 400 ms
- Frame size = 100 bytes
- Go back N is used where N = 10

Calculating Transmission Delay-

Transmission delay (T_t)

= Frame size / Bandwidth

= 100 bytes / 20 Kbps

= (100 x 8 bits) / (20 x 10^3 bits per sec)

= 0.04 sec

= 40 msec

Calculating Value Of ‘a’-

a = T_p / T_t

a = 400 msec / 40 msec

a = 10

Calculating Efficiency-

Efficiency (η)

= N / (1+2a)

= 10 / (1 + 2 x 10)

= 10 / 21

= 0.476

= 47.6 %

Calculating Maximum Data Rate Possible-

Maximum data rate possible or Throughput

= Efficiency x Bandwidth

= 0.476 x 20 Kbps

= 9.52 Kbps

≅ 10 Kbps

Thus, Correct Option is (B)

Problem-02:

Consider the Go back N protocol with a sender's window size of 'n'. Suppose that at time 't', the next inorder packet the receiver is expecting has a sequence number of 'K'. Assume that the medium does not reorder messages.

Answer the following questions-

Part-01:

What are the possible sets of sequence numbers inside the sender's window at time 't'. Assume the sender has already received the ACKs.

1. [K-1, K+n-1]
2. [K, K+n-1]
3. [K, K+n]
4. [K+n, K-1]

Part-02:

If acknowledgements are still on their way to sender, what are all possible values of the ACK field in the messages currently propagating back to the sender at a time 't'?

1. [K-n, K-1]
2. [K-1, K-n]
3. [K, K-n]
4. [K-n, K+1]

Solution

Part-01:

- In Go back N protocol, the receiver window size is 1.
- It is given that receiver expects the packet having sequence number 'K'.
- It means it has processed all the packets ranging from 0 to K-1.
- It is given that sender has received the acknowledgement for all these packets.
- So, outstanding packets in sender's window waiting for the acknowledgement starts from K.
- Sender window size = n.
- Therefore, last packet in sender's window will have sequence number K+n-1.

Thus, Option (B) is correct.

Part-02:

- Acknowledgement number is the next expected sequence number by the receiver.
- Receiver expects the packet having sequence number 'K' at time 't'.
- It means it has received the packets ranging from 0 to K-1 whose acknowledgements are are on the way.
- For the (K-1)th packet, acknowledgement number would be 'K'.
- For the (K-2)th packet, acknowledgement number would be 'K-1' and so on.

Now,

- At any time, maximum number of outstanding packets can be 'n'.
- This is because sender's window size is 'n'.
- Therefore, the possible values of acknowledgement number ranges from [K-n+1, ……, K-3, K-2, K-1, K] (total n values)

- Here, we have assumed that the acknowledgement for all the packets are sent independently.

Thus, Option (C) is correct.

Problem-03:

Station A needs to send a message consisting of 9 packets to station B using a sliding window (window size 3) and go back n error control strategy. All packets are ready and immediately available for transmission.

If every 5th packet that A transmits gets lost (but no ACKs from B ever get lost), then what is the number of packets that A will transmit for sending the message to B?

1. 12
2. 14
3. 16
4. 18

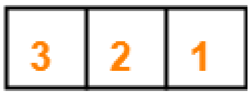
Solution-

Given-

- Total number of packets to be sent = 9
- Go back N is used where N = 3
- Every 5th packet gets lost

Step-01:

Since sender window size is 3, so sender sends 3 packets (1, 2, 3)-



Total packets sent till now from sender side = 3

Step-02:

After receiving the acknowledgement for packet-1, sender slides its window and sends packet-4.

4	3	2

Total packets sent till now from sender side = 4

<u>Step-03:</u>

After receiving the acknowledgement for packet-2, sender slides its window and sends packet-5.

5	4	3

Total packets sent till now from sender side = 5

<u>Step-04:</u>

After receiving the acknowledgement for packet-3, sender slides its window and sends packet-6.

6	5	4

Total packets sent till now from sender side = 6

<u>Step-05:</u>

After receiving the acknowledgement for packet-4, sender slides its window and sends packet-7.

7	6	5

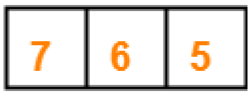
Total packets sent till now from sender side = 7

<u>Step-06:</u>

- According to question, every 5th packet gets lost.
- So, packet-5 gets lost and when time out occurs, sender retransmits packet-5.
- In Go back N, all the following packets are also discarded by the receiver.

- So, packet-6 and packet-7 are discarded by the receiver and they are also retransmitted.
- Thus, the entire window is retransmitted.

So, we have-

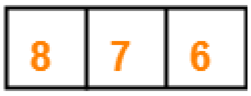


Total packets sent till now from sender side = 10

Now, the next 5th packet that will be lost will be packet-7. (6, 7, 5, 6, 7)

<u>Step-07:</u>

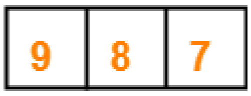
After receiving the acknowledgement for packet-5, sender slides its window and sends packet-8.



Total packets sent till now from sender side = 11

<u>Step-08:</u>

After receiving the acknowledgement for packet-6, sender slides its window and sends packet-9.

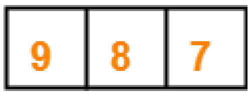


Total packets sent till now from sender side = 12

<u>Step-09:</u>

- According to question, every 5th packet gets lost.
- So, packet-7 gets lost and when time out occurs, sender retransmits packet-7 and the following packets.
- Thus, the entire window is retransmitted.

So, we have-

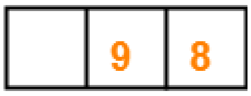


Total packets sent till now from sender side = 15

Now, the next 5th packet that will be lost will be packet-9. (8, 9, 7, 8, 9)

<u>Step-10:</u>

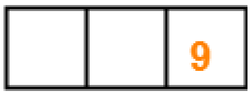
After receiving the acknowledgement for packet-7, sender slides its window.



Total packets sent till now from sender side = 15

<u>Step-11:</u>

After receiving the acknowledgement for packet-8, sender slides its window.

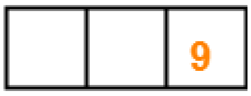


Total packets sent till now from sender side = 15

<u>Step-12:</u>

- According to question, every 5th packet gets lost.
- So, packet-9 gets lost and when time out occurs, sender retransmits packet-9.

So, we have-



Total packets sent till now from sender side = 16

Finally, all the 9 packets got transmitted which took total 16 number of transmissions.

Thus, Correct Option is (C).

<u>Problem-04:</u>

In Go back 4, if every 6th packet that is being transmitted is lost and if total number of packets to be sent is 10, then how many transmissions will be required?

<u>Solution-</u>

- Try yourself!
- We have to solve in exactly the same way as we have solved Problem-03.
- Total number of transmissions required will be 17.

<u>Problem-05:</u>

A 1 Mbps satellite link connects two ground stations. The altitude of the satellite is 36504 km and speed of the signal is 3 x 10^8 m/sec. What should be the packet size for a channel utilization of 25% for a satellite link using go back 127 sliding window protocol?

1. 120 bytes
2. 60 bytes
3. 240 bytes
4. 90 bytes

<u>Solution-</u>

Given-

- Bandwidth = 1 Mbps
- Distance = 2 x 36504 km = 73008 km
- Propagation speed = 3 x 10^8 m/sec
- Efficiency = 25% = 1/4
- Go back N is used where N = 127

Let the packet size be L bits.

<u>Calculating Transmission Delay-</u>

Transmission delay (T_t)

= Packet size / Bandwidth

= L bits / 1 Mbps

= L μsec

<u>Calculating Propagation Delay-</u>

Propagation delay (T_p)

= Distance / Speed

= $(73008 \times 10^3$ m$) / (3 \times 10^8$ m/sec$)$

= 24336×10^{-5} sec

= 243360 μsec

<u>Calculating Value of 'a'-</u>

$a = T_p / T_t$

a = 243360 μsec / L μsec

a = 243360 / L

<u>Calculating Packet Size-</u>

Efficiency (η) = N / (1+2a)

Substituting the values, we get-

$1/4 = 127 / (1 + 2 \times 243360 / L)$

$1/4 = 127 \times L / (L + 486720)$

$L + 486720 = 508 \times L$

$507 \times L = 486720$

$L = 960$

From here, packet size = 960 bits or 120 bytes.

Thus, Correct Option is (A).

Problem-06:

Consider a network connecting two systems located 8000 km apart. The bandwidth of the network is 500 x 10^6 bits per second. The propagation speed of the media is 4 x 10^6 meters per second. It is needed to design a Go back N sliding window protocol for this network. The average packet size is 10^7 bits. The network is to be used to its full capacity.

Assume that processing delays at nodes are negligible. Then, the minimum size in bits of the sequence number field has to be ______ ?

Solution-

Given-

- Distance = 8000 km
- Bandwidth = 500 x 10^6 bps
- Propagation speed = 4 x 10^6 m/sec
- Packet size = 10^7 bits

Now,

- For using the network to its full capacity, Efficiency (η) = 1
- Efficiency (η) = 1 when sender window size = 1+2a

Calculating Transmission Delay-

Transmission delay (T_t)

= Packet size / Bandwidth

= 10^7 bits / (500 x 10^6 bits per sec)

= 1 / 50 sec

= 0.02 sec

Calculating Propagation Delay-

Propagation delay (T_p)

= Distance / Speed

= 8000 km / (4 x 10^6 m/sec)

= 2 sec

Calculating Value of ‘a’-

$a = T_p / T_t$

a = 2 sec / 0.02 sec

a = 100

Calculating Sender Window Size-

Sender window size

= 1 + 2a

= 1 + 2 x 100

= 201

Calculating Minimum Size of Sequence Number Field-

Minimum number of bits required in the sequence number field

$= \lceil \log_2(1+2a) \rceil$

$= \lceil \log_2(201) \rceil$

$= \lceil 7.65 \rceil$

= 8

Thus, Minimum size of sequence number field = 8 bits.

Selective Repeat Protocol-

Before you go through this article, make sure that you have gone through the previous article on **Selective Repeat Protocol**.

We have discussed-

- **Sliding Window Protocols** allow the sender to send multiple frames before needing acknowledgements.
- Selective Repeat is an implementation of a sliding window protocol.

In this article, we will discuss practice problems based on selective repeat protocol.

PRACTICE PROBLEMS BASED ON SELECTIVE REPEAT PROTOCOL-

Problem-01:

The maximum window size for data transmission using the selective repeat protocol with n bit frame sequence numbers is-

1. 2^n
2. 2^{n-1}
3. 2^n-1
4. 2^{n-2}

Solution-

We know-

- With n bits, total number of sequence numbers possible = 2^n.
- In SR Protocol, sender window size = receiver window size = W (say)

For any sliding window protocol to work without any problems,

Min Available Sequence Numbers = Sender window size + Receiver window size

So, we have-

$2^n = W + W$

$2^n = 2W$

$W = 2^{n-1}$

Therefore, maximum window size possible of sender and receiver = 2^{n-1}

Thus, Option (B) is correct.

Problem-02:

In SR protocol, suppose frames through 0 to 4 have been transmitted. Now, imagine that 0 times out, 5 (a new frame) is transmitted, 1 times out, 2 times out and 6 (another new frame) is transmitted.

At this point, what will be the outstanding packets in sender's window?

1. 341526
2. 3405126
3. 0123456
4. 654321

Solution-

In SR Protocol, only the required frame is retransmitted and not the entire window.

Step-01:

Frames through 0 to 4 have been transmitted-

4 , 3 , 2 , 1 , 0

Step-02:

0 times out. So, sender retransmits it-

0 , 4 , 3 , 2 , 1

Step-03:

5 (a new frame) is transmitted-

5 , 0 , 4 , 3 , 2 , 1

Step-04:

1 times out. So, sender retransmits it-

1 , 5 , 0 , 4 , 3 , 2

Step-05:

2 times out. So, sender retransmits it-

2 , 1 , 5 , 0 , 4 , 3

Step-06:

6 (another new frame) is transmitted-

6 , 2 , 1 , 5 , 0 , 4 , 3

Thus, Option (B) is correct.

Problem-03:

The selective repeat protocol is similar to Go back N except in the following way-

1. Frame Formats are similar in both the protocols
2. The sender has a window defining maximum number of outstanding frames in both the protocols
3. Both uses piggybacked acknowledgements where possible and does not acknowledge every frame explicitly.
4. Both uses piggyback approach that acknowledges the most recently received frame

Solution-

Option (A)-

- Both the protocols use the same frame formats because both are sliding window protocols.
- The variation occurs only in the coding and implementation.

Option (B)-

- In both the protocols, sender has a window which defines the maximum number of outstanding frames.

Option (C)-

- Both the protocols use piggybacked acknowledgements wherever possible.
- Sending acknowledgements along with the data are called as **piggybacked acknowledgements**.
- But Go back N protocol uses cumulative acknowledgements and does not acknowledge every frame explicitly.
- On the other hand, Selective repeat protocol acknowledges each frame independently.

Option (D)-

- Both the protocols use piggyback approach.
- Go back N acknowledges the most recently received frame by sending a cumulative acknowledgement which includes the acknowledgement for previous packets too if any.
- On the other hand, Selective Repeat protocol acknowledges all the frames independently and not only the recently received frame.

Thus, Options (C) and (D) are correct.

Problem-04:

Consider a 128 x 10^3 bits/sec satellited communication link with one way propagation delay of 150 msec. Selective Retransmission (repeat) protocol is used on this link to send data with a frame size of 1 KB. Neglect the transmission time of acknowledgement. The minimum number of bits required for the sequence number field to achieve 100% utilization is ________ .

Solution-

Given-

- Bandwidth = 128 x 10^3 bits/sec
- Propagation delay (T_p) = 150 msec
- Frame size = 1 KB

Now,

- To achieve 100% utilization, efficiency must be 100%.
- Efficiency is 100% when sender window size is optimal i.e. 1+2a

Calculating Transmission Delay-

Transmission delay (T_t)

= Frame size / Bandwidth

= 1 KB / (128 x 10^3 bits per sec)

= (1 x 2^{10} x 8 bits) / (128 x 10^3 bits per sec)

= 64 msec

Calculating Value of 'a'-

a = T_p / T_t

a = 150 msec / 64 msec

a = 2.34

Calculating Optimal Sender Window Size-

Optimal sender window size

= 1 + 2a

= 1 + 2 x 2.34

= $\lceil 5.68 \rceil$

= 6

Calculating Number Of Sequence Numbers Required-

In SR Protocol, sender window size and receiver window size are same.

So, sender window size = receiver window size = 6

Now,

For any sliding window protocol, minimum number of sequence numbers required

= Sender window size + Receiver window size

= 6 + 6

= 12

Calculating Bits Required in Sequence Number Field-

To have 12 sequence numbers,

Minimum number of bits required in sequence number field

= $\lceil \log_2(12) \rceil$

= 4

Thus,

- Minimum number of bits required in sequence number field = 4
- With 4 bits, number of sequence numbers possible = 16
- We use only 12 sequence numbers and rest 4 remains unused.

Comparison Table of Sliding window protocol-

	Stop and Wait ARQ	**Go back N**	**Selective Repeat**	**Remarks**
Efficiency	1 / (1+2a)	N / (1+2a)	N / (1+2a)	Go back N and Selective Repeat gives better efficiency than Stop and Wait ARQ.
Window Size	Sender Window Size = 1 Receiver Window Size = 1	Sender Window Size = N Receiver Window Size = 1	Sender Window Size = N Receiver Window Size = N	Buffer requirement in Selective Repeat is very large. If the system does not have lots of memory, then it is better to choose Go back N.

Minimum number of sequence numbers required	2	N+1	2 x N	Selective Repeat requires large number of bits in sequence number field.
Retransmissions required if a packet is lost	Only the lost packet is retransmitted	The entire window is retransmitted	Only the lost packet is retransmitted	Selective Repeat is far better than Go back N in terms of retransmissions required.
Bandwidth Requirement	Bandwidth requirement is Low	Bandwidth requirement is high because even if a single packet is lost, entire window has to be retransmitted. Thus, if error rate is high, it wastes a lot of bandwidth.	Bandwidth requirement is moderate	Selective Repeat is better than Go back N in terms of bandwidth requirement.
CPU usage	Low	Moderate	High due to searching and sorting required at sender and receiver side	Go back N is better than Selective Repeat in terms of CPU usage.
Level of difficulty in Implementation	Low	Moderate	Complex as it requires extra logic and sorting and searching	Go back N is better than Selective Repeat in terms of implementation difficulty.

Acknowledgements	Uses independent acknowledgement for each packet	Uses cumulative acknowledgements (but may use independent acknowledgements as well)	Uses independent acknowledgement for each packet	Sending cumulative acknowledgements reduces the traffic in the network but if it is lost, then the ACKs for all the corresponding packets are lost.
Type of Transmission	Half duplex	Full duplex	Full duplex	Go back N and Selective Repeat are better in terms of channel usage.

Conclusions-

- Go back N is more often used than other protocols.
- SR protocol is less used because of its complexity.
- Stop and Wait ARQ is less used because of its low efficiency.
- Depending on the context and resources availability, Go back N or Selective Repeat is employed.
- Selective Repeat and Stop and Wait ARQ are similar in terms of retransmissions.
- Go back N and Selective Repeat are similar in terms of efficiency if sender window sizes are same.
- SR protocol may be considered as a combination of advantages of Stop and Wait ARQ and Go back N.
- SR protocol is superior to other protocols but because of its complexity, it is less used.

Important Notes-

Note-01:

Protocols at data link layer like HDLC (Low level protocols) use Go back N.

This is because-

1. Bandwidth is high
2. CPU is very busy doing routing job

3. Error rate is low since out of order packets are not possible in wired medium

Note-02:

Protocols at transport layer like TCP (High level protocols) use selective repeat.

PRACTICE PROBLEMS BASED ON SLIDING WINDOW PROTOCOLS-

Problem-01:

If the bandwidth between the sender and receiver is sufficient, CPU and buffers are moderate, then which flow control protocol would you suggest to use?

Solution-

The suggested protocol would be Go back N.

Problem-02:

If the bandwidth between the sender and receiver is moderate, CPU and buffers are sufficient, then which flow control protocol would you suggest to use?

Solution-

The suggested protocol would be Selective Repeat.

PRACTICE PROBLEMS BASED ON FLOW CONTROL PROTOCOLS-

Problem-01:

In what protocols is it possible for the sender to receive an acknowledgement for a packet that falls outside its current window?

1. Stop and Wait
2. Selective Repeat
3. Go back N
4. All of the above

Solution-

- Delayed Acknowledgements fall outside the current window.
- They may occur in any of the flow control protocols and received by the sender.

Thus, correct option is (D).

Problem-02:

On a wireless link, the probability of packet error is 0.2. A stop and wait protocol is used to transfer data across the link. The channel condition is assumed to be independent from transmission to transmission. What is the average number of transmission attempts required to transfer 100 packets?

1. 100
2. 125
3. 150
4. 200

Solution-

Method-01:

Given-

- Probability of packet error = 0.2
- We have to transfer 100 packets

Now,

- When we transfer 100 packets, number of packets in which error will occur = 0.2 x 100 = 20.
- Then, these 20 packets will have to be retransmitted.
- When we retransmit 20 packets, number of packets in which error will occur = 0.2 x 20 = 4.

- Then, these 4 packets will have to be retransmitted.
- When we retransmit 4 packets, number of packets in which error will occur = 0.2 x 4 = 0.8 ≅ 1.
- Then, this 1 packet will have to be retransmitted.

From here, average number of transmission attempts required = 100 + 20 + 4 + 1 = 125.

Thus, Option (B) is correct.

Method-02:

REMEMBER

If there are n packets to be transmitted and p is the probability of packet error, then-

Number of transmission attempts required

$= n + np + np^2 + np^3 + \ldots\ldots + \infty$

$= n / (1-p)$

Substituting the given values, we get-

Average number of transmission attempts required = 100 / (1-0.2) = 125.

Thus, Option (B) is correct.

Problem-03:

Compute the fraction of the bandwidth that is wasted on overhead (headers and retransmissions) for a protocol on a heavily loaded 50 Kbps satellite channel with data frames consisting of 40 bits header and 3960 data bits. Assume that the signal propagation time from the earth to the satellite is 270 msec. ACK frames never occur. NAK frames are 40 bits. The error rate for data frames is 1% and the error rate for NAK frames is negligible.

1. 1.21 %
2. 2.12 %
3. 1.99 %
4. 1.71 %

Solution-

Consider 100 frames are being sent. Then, we have-

Useful Data Sent-

Since each frame contains 3960 data bits, so while sending 100 frames,

Useful data sent

= 100 x 3960 bits

= 396000 bits

Useless Data Sent / Overhead-

In general, overhead is due to headers, retransmissions and negative acknowledgements.

Now,

- The error rate for data frames is 1%, therefore out of 100 sent frames, error occurs in one frame.
- This causes the negative acknowledgement to follow which causes the retransmission.

So, we have-

- Overhead due to headers = 100 x 40 bits = 400 bits.
- Overhead due to negative acknowledgement = 40 bits.
- Overhead due to retransmission = 40 bits header + 3960 data bits = 4000 bits.

From here,

Total overhead

= 400 bits + 40 bits + 4000 bits

= 8040 bits

Calculating Efficiency-

Efficiency (η) = Useful data sent / Total data sent

Here,

- Useful data sent = 396000 bits
- Total data sent = Useful data sent + Overhead = 396000 bits + 8040 bits = 404040 bits

Substituting the values, we get-

Efficiency (η)

= 396000 bits / 404040 bits

= 0.9801

Calculating Bandwidth Utilization-

Bandwidth Utilization

= Efficiency x Bandwidth

= 0.9801 x 50 Kbps

= 49.005 Kbps

Calculating Bandwidth Wasted-

Bandwidth wasted

= Bandwidth – Bandwidth Utilization

= 50 Kbps – 49.005 Kbps

= 0.995 Kbps

Calculating Fraction of Bandwidth Wasted-

Fraction of bandwidth wasted

= Wasted Bandwidth / Total Available Bandwidth

= 0.995 Kbps / 50 Kbps

= 0.0199

= 1.99 %

Thus, Option (C) is correct.

Problem-04:

Consider 1 Mbps error free line. The maximum frame size is 1000 bits. New packets are generated about 1 sec apart. The time out interval is 10 msec. If the ack timer is eliminated. How many times the average message be transmitted?

1. Only once
2. Twice
3. Thrice
4. Can't say

Solution-

- Transmission delay (T_t) = L / B = 1000 bits / 10^6 bits per sec = 1 msec.
- After packet is put on the link, the time out timer is started which is 10 msec long.
- The next packet is transmitted after 1 sec = 1000 msec.
- If no acknowledgement is received within 10 msec, the packet will be retransmitted.
- We have been asked how many times the average message be transmitted i.e. how many retransmissions are possible.
- Retransmission occurs or not depends on the propagation delay (Tp).
- If T_p is more, time out will occur and retransmission will take place but if T_p is less, then there will be no time out.
- Since propagation delay (Tp) is not given in the question, therefore we can not say anything.

Thus, Option (D) is correct.

Problem-05:

What is the effect on line utilization if we increase the number of frames for a constant message size?

1. Lower line efficiency
2. Higher line efficiency
3. No change in line efficiency
4. No relation between line efficiency and frame size

Solution-

In both the following cases, line utilization remains the same-

- Whether the entire message is sent as a single entity
- Or the entire message is divided into frames and then frames are sent.

This is because line contains the same amount of data in both cases.

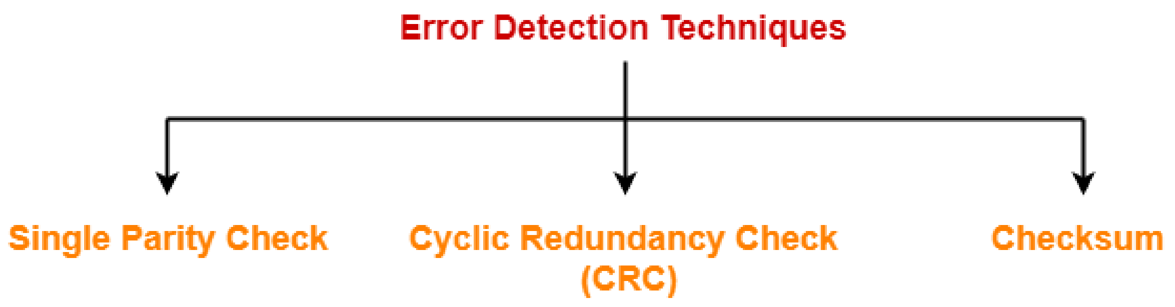
So,

- If the number of frames are increased by dividing the message, there is no change in line efficiency.
- The line efficiency remains the same.

Thus, Option (C) is correct.

Error Detection Methods-

Some popular error detection methods are-



1. Single Parity Check
2. Cyclic Redundancy Check (CRC)
3. Checksum

In this article, we will discuss about Single Parity Check.

Single Parity Check-

In this technique,

- One extra bit called as **parity bit** is sent along with the original data bits.
- Parity bit helps to check if any error occurred in the data during the transmission.

Steps Involved-

Error detection using single parity check involves the following steps-

Step-01:

At sender side,

- Total number of 1's in the data unit to be transmitted is counted.
- The total number of 1's in the data unit is made even in case of even parity.
- The total number of 1's in the data unit is made odd in case of odd parity.
- This is done by adding an extra bit called as **parity bit**.

Step-02:

- The newly formed code word (Original data + parity bit) is transmitted to the receiver.

Step-03:

At receiver side,

- Receiver receives the transmitted code word.
- The total number of 1's in the received code word is counted.

Then, following cases are possible-

- If total number of 1's is even and even parity is used, then receiver assumes that no error occurred.
- If total number of 1's is even and odd parity is used, then receiver assumes that error occurred.
- If total number of 1's is odd and odd parity is used, then receiver assumes that no error occurred.
- If total number of 1's is odd and even parity is used, then receiver assumes that error occurred.

Parity Check Example-

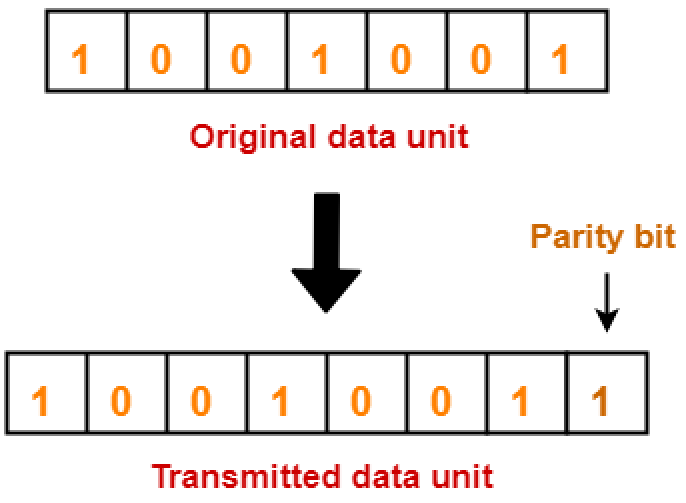
Consider the data unit to be transmitted is 1001001 and even parity is used.

Then,

At Sender Side-

- Total number of 1's in the data unit is counted.

- Total number of 1's in the data unit = 3.
- Clearly, even parity is used and total number of 1's is odd.
- So, parity bit = 1 is added to the data unit to make total number of 1's even.
- Then, the code word 10010011 is transmitted to the receiver.



At Receiver Side-

- After receiving the code word, total number of 1's in the code word is counted.
- Consider receiver receives the correct code word = 10010011.
- Even parity is used and total number of 1's is even.
- So, receiver assumes that no error occurred in the data during the transmission.

Advantage-

- This technique is guaranteed to detect an odd number of bit errors (one, three, five and so on).
- If odd number of bits flip during transmission, then receiver can detect by counting the number of 1's.

Limitation-

- This technique can not detect an even number of bit errors (two, four, six and so on).
- If even number of bits flip during transmission, then receiver can not catch the error.

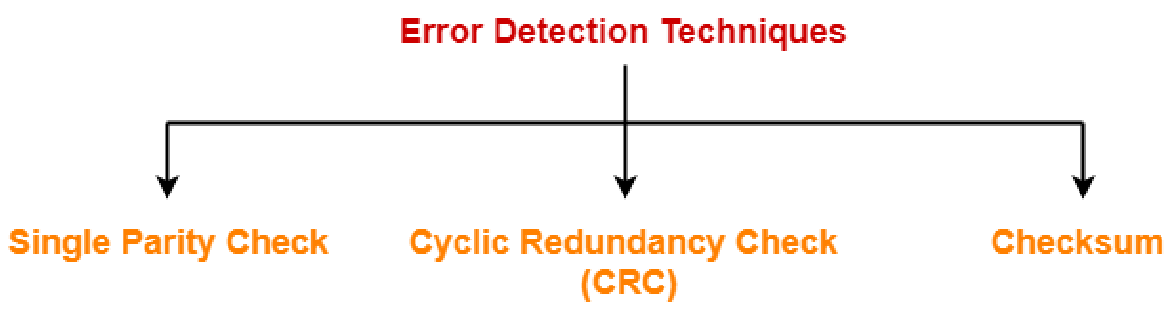
EXAMPLE

- Consider the data unit to be transmitted is 10010001 and even parity is used.
- Then, code word transmitted to the receiver = 100100011
- Consider during transmission, code word modifies as 101100111. (2 bits flip)
- On receiving the modified code word, receiver finds the number of 1's is even and even parity is used.
- So, receiver assumes that no error occurred in the data during transmission though the data is corrupted.

Error Detection in Computer Networks-

Error detection is a technique that is used to check if any error occurred in the data during the transmission.

Some popular error detection methods are-



1. Single Parity Check
2. Cyclic Redundancy Check (CRC)
3. Checksum

In this article, we will discuss about Cyclic Redundancy Check (CRC).

Cyclic Redundancy Check-

- Cyclic Redundancy Check (CRC) is an error detection method.

- It is based on binary division.

CRC Generator-

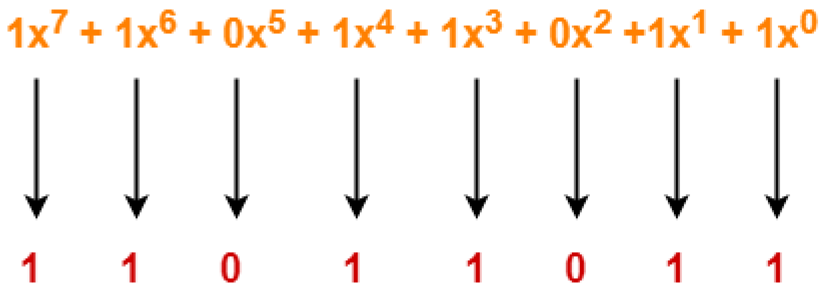
- CRC generator is an algebraic polynomial represented as a bit pattern.
- Bit pattern is obtained from the CRC generator using the following rule-

> The power of each term gives the position of the bit and the coefficient gives the value of the bit.

Example-

Consider the CRC generator is $x^7 + x^6 + x^4 + x^3 + x + 1$.

The corresponding binary pattern is obtained as-



Thus, for the given CRC generator, the corresponding binary pattern is 11011011.

Properties Of CRC Generator-

The algebraic polynomial chosen as a CRC generator should have at least the following properties-

Rule-01:

- It should not be divisible by x.
- This condition guarantees that all the burst errors of length equal to the length of polynomial are detected.

Rule-02:

- It should be divisible by x+1.
- This condition guarantees that all the burst errors affecting an odd number of bits are detected.

Important Notes-

If the CRC generator is chosen according to the above rules, then-

- CRC can detect all single-bit errors
- CRC can detect all double-bit errors provided the divisor contains at least three logic 1's.
- CRC can detect any odd number of errors provided the divisor is a factor of x+1.
- CRC can detect all burst error of length less than the degree of the polynomial.
- CRC can detect most of the larger burst errors with a high probability.

Steps Involved-

Error detection using CRC technique involves the following steps-

Step-01: Calculation Of CRC At Sender Side-

At sender side,

- A string of n 0's is appended to the data unit to be transmitted.
- Here, n is one less than the number of bits in CRC generator.
- Binary division is performed of the resultant string with the CRC generator.
- After division, the remainder so obtained is called as **CRC**.
- It may be noted that CRC also consists of n bits.

Step-02: Appending CRC To Data Unit-

At sender side,

- The CRC is obtained after the binary division.
- The string of n 0's appended to the data unit earlier is replaced by the CRC remainder.

Step-03: Transmission To Receiver-

- The newly formed code word (Original data + CRC) is transmitted to the receiver.

Step-04: Checking at Receiver Side-

At receiver side,

- The transmitted code word is received.
- The received code word is divided with the same CRC generator.
- On division, the remainder so obtained is checked.

The following two cases are possible-

Case-01: Remainder = 0

If the remainder is zero,

- Receiver assumes that no error occurred in the data during the transmission.
- Receiver accepts the data.

Case-02: Remainder ≠ 0

If the remainder is non-zero,

- Receiver assumes that some error occurred in the data during the transmission.
- Receiver rejects the data and asks the sender for retransmission.

PRACTICE PROBLEMS BASED ON CYCLIC REDUNDANCY CHECK (CRC)-

Problem-01:

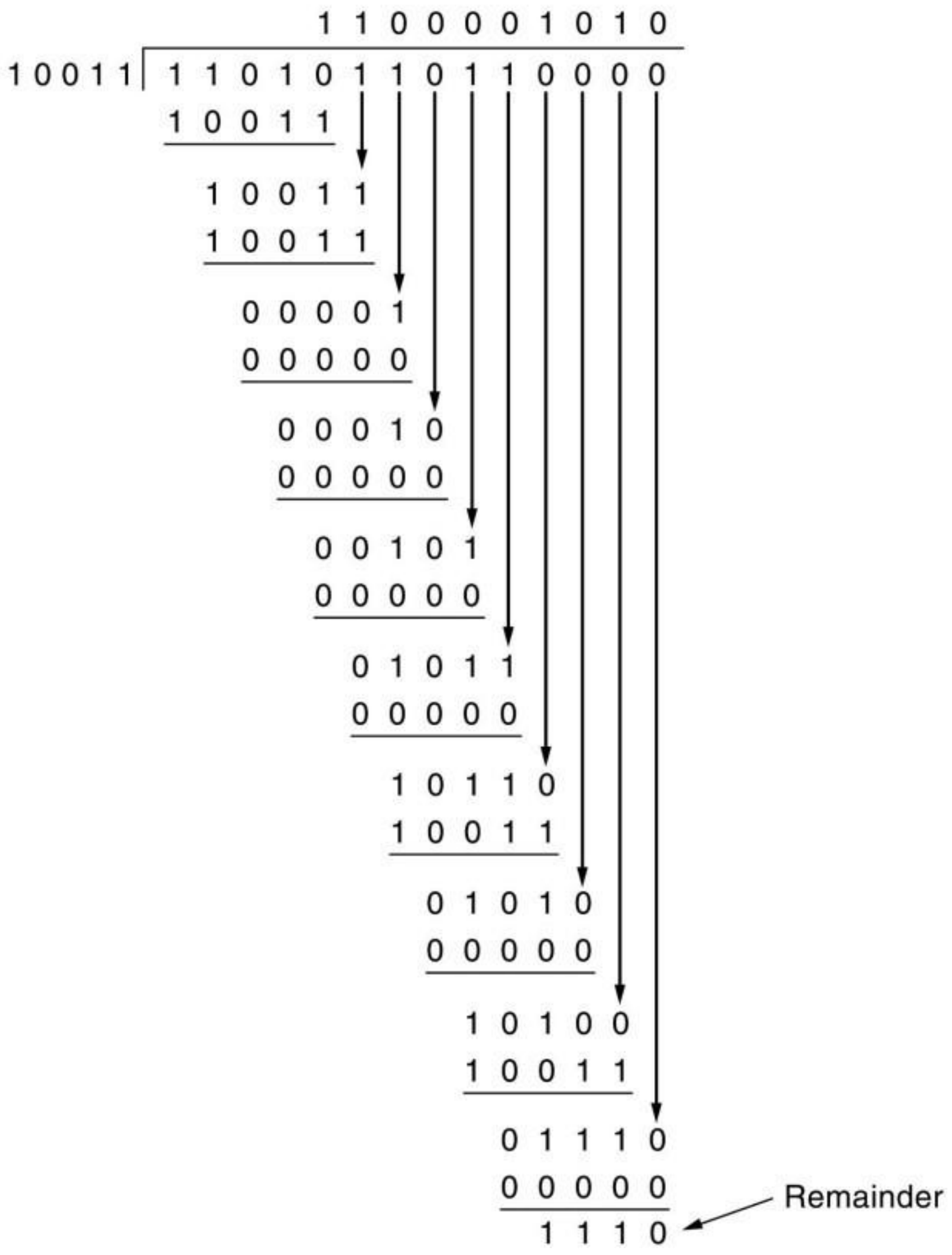
A bit stream 1101011011 is transmitted using the standard CRC method. The generator polynomial is x^4+x+1. What is the actual bit string transmitted?

Solution-

- The generator polynomial $G(x) = x^4 + x + 1$ is encoded as 10011.
- Clearly, the generator polynomial consists of 5 bits.
- So, a string of 4 zeroes is appended to the bit stream to be transmitted.

- The resulting bit stream is 1101011011**0000**.

Now, the binary division is performed as-



From here, CRC = 1110.

Now,

- The code word to be transmitted is obtained by replacing the last 4 zeroes of 1101011011**0000** with the CRC.

- Thus, the code word transmitted to the receiver = 11010110111**110**.

Problem-02:

A bit stream 10011101 is transmitted using the standard CRC method. The generator polynomial is x^3+1.

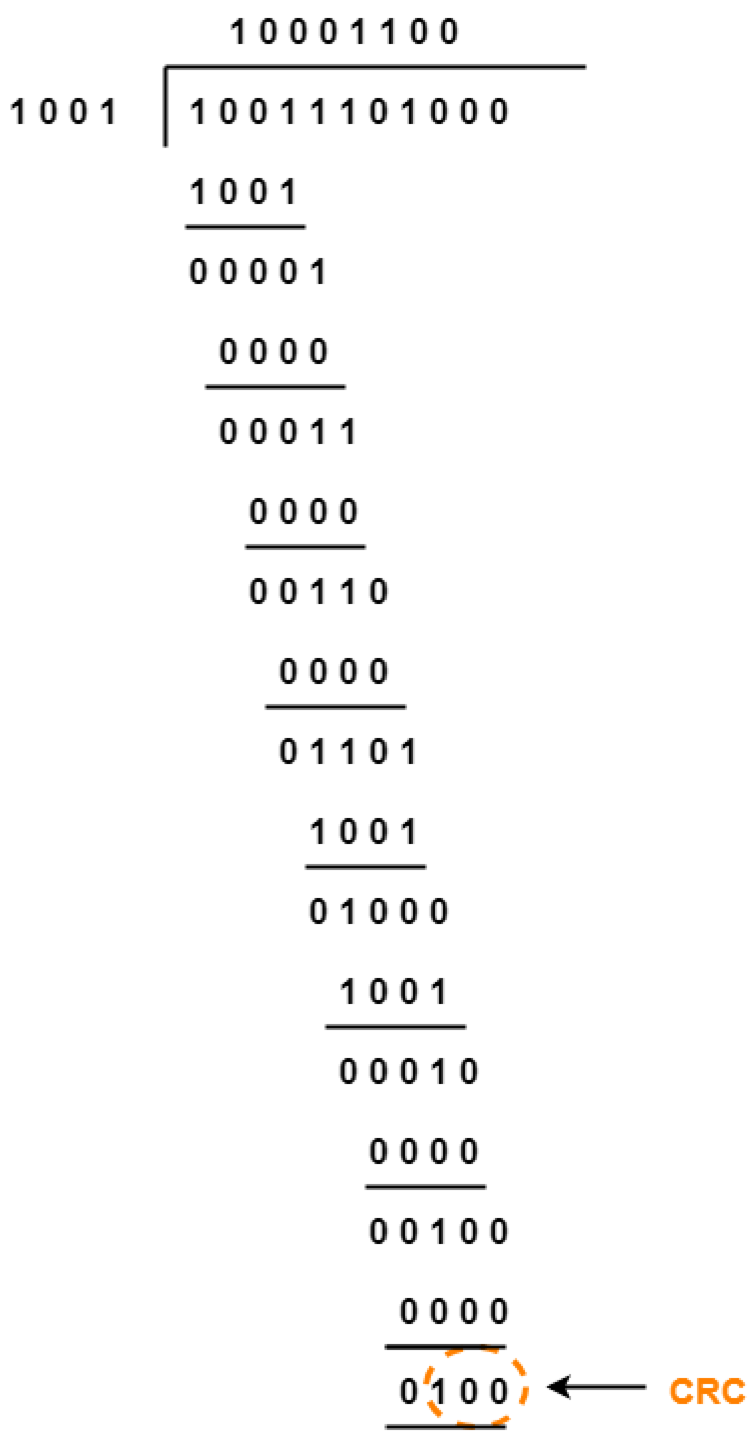
1. What is the actual bit string transmitted?
2. Suppose the third bit from the left is inverted during transmission. How will receiver detect this error?

Solution-

Part-01:

- The generator polynomial $G(x) = x^3 + 1$ is encoded as 1001.
- Clearly, the generator polynomial consists of 4 bits.
- So, a string of 3 zeroes is appended to the bit stream to be transmitted.
- The resulting bit stream is 10011101**000**.

Now, the binary division is performed as-



From here, CRC = 100.

Now,

- The code word to be transmitted is obtained by replacing the last 3 zeroes of 10011101**000** with the CRC.
- Thus, the code word transmitted to the receiver = 10011101**100**.

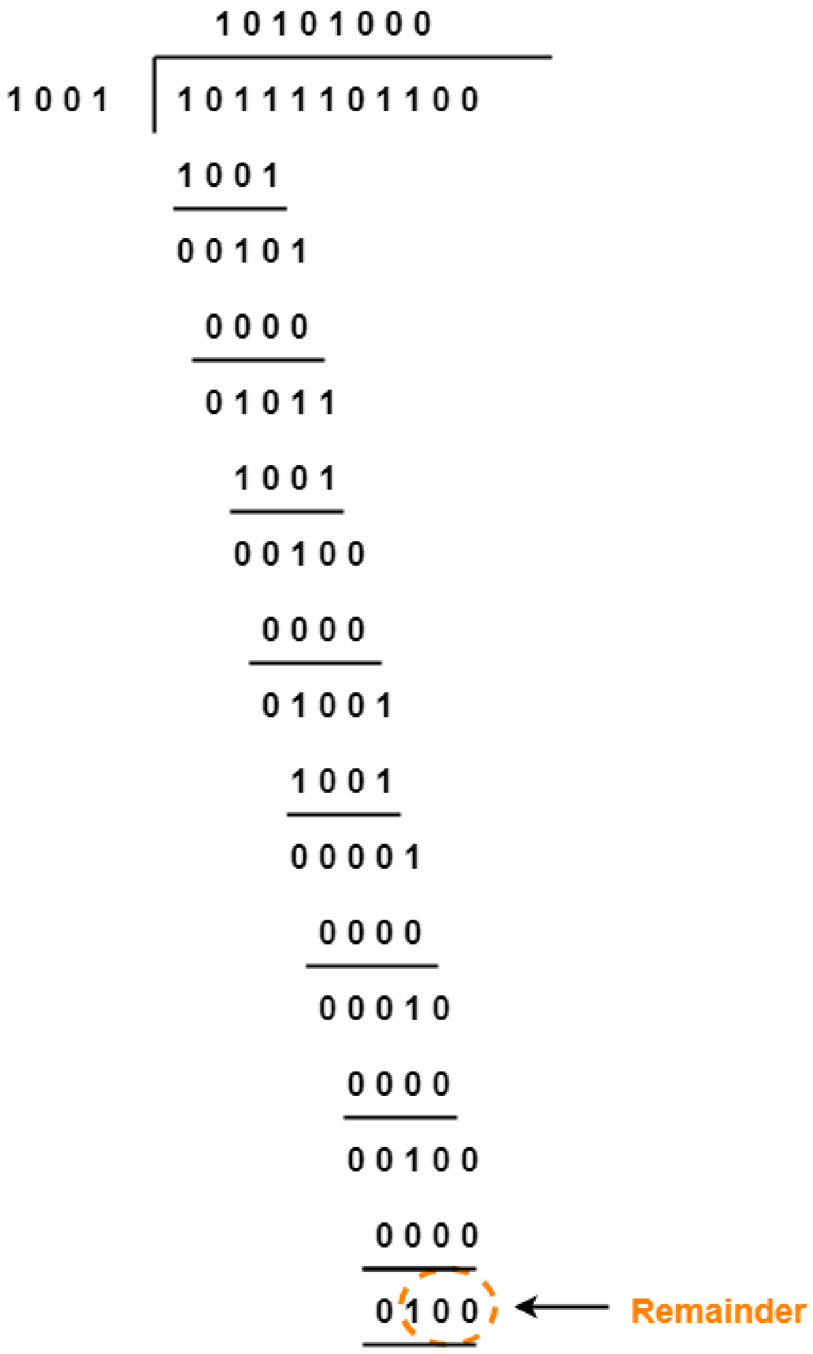
Part-02:

According to the question,

- Third bit from the left gets inverted during transmission.
- So, the bit stream received by the receiver = 10111101100.

Now,

- Receiver receives the bit stream = 10111101100.
- Receiver performs the binary division with the same generator polynomial as-



From here,

- The remainder obtained on division is a non-zero value.
- This indicates to the receiver that an error occurred in the data during the transmission.
- Therefore, receiver rejects the data and asks the sender for retransmission.

Checksum-

Checksum is an error detection method.

Error detection using checksum method involves the 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:

At receiver side,

- If m bit checksum is being used, the received data unit is divided into segments of m bits.
- All the m bit segments are added along with the checksum value.
- The value so obtained is complemented and the result is checked.

Then, following two cases are possible-

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.

Checksum Example-

Consider the data unit to be transmitted is-

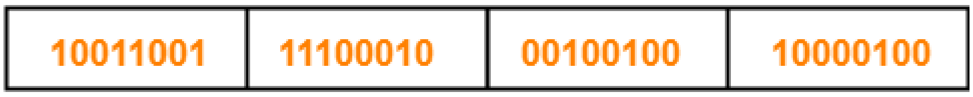
10011001111000100010010010000100

Consider 8 bit checksum is used.

Step-01:

At sender side,

The given data unit is divided into segments of 8 bits as-



10011001	11100010	00100100	10000100

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.

Important Notes-

Note-01:

- Consider while adding the m bit segments, the result obtained consists of more than m bits.
- Then, wrap around the extra bits and add to the result so that checksum value consists of m bits.

Note-02:

- While calculating the checksum, if checksum value is needed, then assume it to be zero.
- After calculating the checksum value, substitute the checksum value in the checksum field.
- This will be required during checksum calculation of **IP Header**, **TCP Header** and **UDP Header**.

Note-03:

- The checksum is used in the internet by several protocols although not at the data link layer.

PRACTICE PROBLEM BASED ON CHECKSUM ERROR DETECTION METHOD-

Problem-

Checksum value of 1001001110010011 and 1001100001001101 of 16 bit segment is-

1. 1010101000011111
2. 1011111000100101

3. 1101010000011110
4. 1101010000111111

Solution-

We apply the above discussed algorithm to calculate the checksum.

- 1001001110010011 + 1001100001001101 = 10010101111100000
- Since, the result consists of 17 bits, so 1 bit is wrapped around and added to the result.
- 0010101111100000 + 1 = 0010101111100001
- Now, result consists of 16 bits.
- Now, 1’s complement is taken which is 1101010000011110
- Thus, checksum value = 1101010000011110

Thus, Option (C) is correct.

Types of Data Link Protocol:

 Data Link Protocols are divided into two categories:

 Asynchronous Protocols **Synchronous Protocols**

Asynchronous Protocols

 Asynchronous protocols treat each character in a bit stream independently.

 These protocols are used in modems.

 They use start and stop bits, and variable gaps between characters.

 They are slower than synchronous protocols in transmitting data.

different asynchronous protocols are:

- XMODEM
- YMODEM
- ZMODEM
- Block Asynchronous Transmission (BLAST)
- Kermit

Synchronous Protocols

 Synchronous Protocols take the whole bit stream and divide it into characters of equal size.

 These protocols have high speed and are used for LAN, WAN and MAN.

 Synchronous protocols are categorized into two groups:

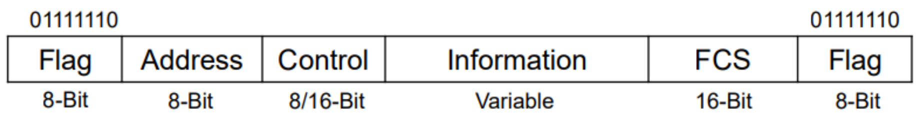
- Character-Oriented Protocol
- Bit-Oriented Protocol

The examples of bit-oriented protocol are:

- Synchronous Data Link Control (SDLC)
- High Level Data Link Control (HDLC)

Frame Structure in HDLC

- Frame in HDLC can have six fields:



- Flag Field: It is the 8-bit field that contains 01111110. It marks the beginning and end of a frame.
- Address Field: This field contains the address of the receiver. It is 8-bit long.
- Control Field: It carries the sequence number, acknowledgements, requests and responses. It can be of 8-bit or 16-bit.
- Information Field: It contains user data. Its length is different for different networks.
- FCS Field: FCS stands for Frame Check Sequence. It is the error detection field and is 16-bit long. It contains either 16-bit CRC or 32-bit CRC.
- HDLC defines three types of frames:
- Information Frame (I-Frame):
- I-Frames carry user data, and control information about user's data.
- Supervisory Frame (S-Frame):
- S-Frames carry flow & error control information.
- Unnumbered Frame (U-Frame):
- U-Frames are reserved for system management.
- They are used to exchange session management & control information between the two connected devices.

===