

II Datalink layer design issue:- The purpose of introducing datalink layer.

- Network layer services

- framing

- Error control

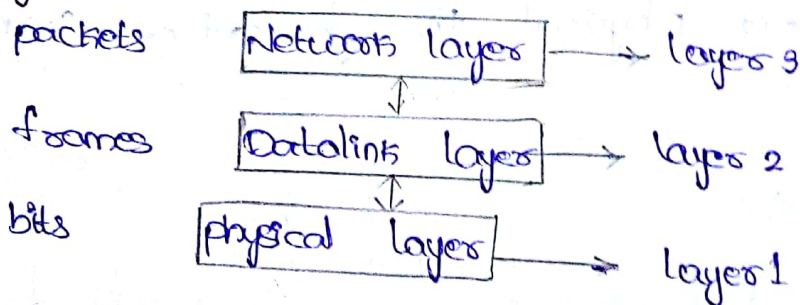
- flow control

Network layer services implementation:-

* Data link layer data format is frame. Data link layer receives either packets (or) binary's form n/w layer at the side of sender and from physical layer at the side of receiver.

* Framing a process of converting packets and binary's into frames.

Framing:-

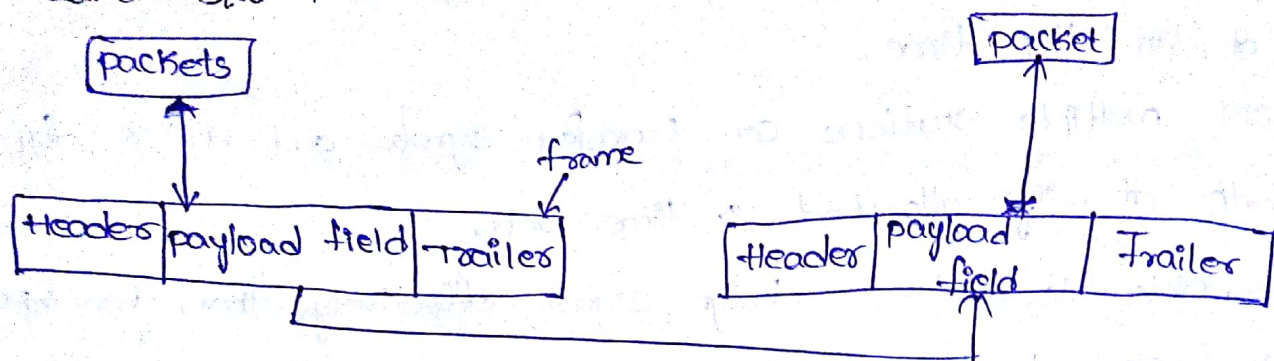


* Datalink layer is design with the primary purpose of achieving reliable and efficient communication b/w source & destination.

* Datalink layer accepting as well as offering the services to the network layer.

* Datalink layer frame is consisting frame header, payload field as well as trailer

Relation b/w packets and frame:-



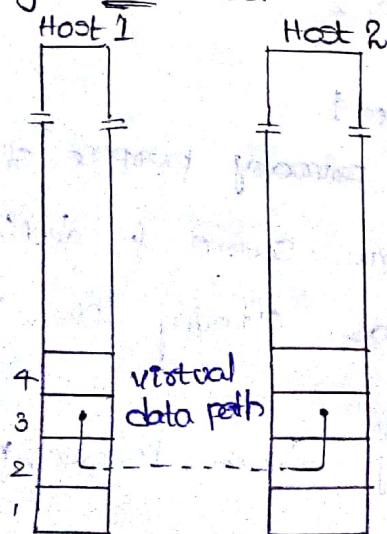
Header & trailer parts of a frame is representing the starting and ending sequences of a message which is to be transported.

A message is contained in a payload field.

The network layer service is forwarded to the datalink layer and the datalink layer principal service function is to transfer the data from the network layer on the source machine to network layer of destination machine logically.

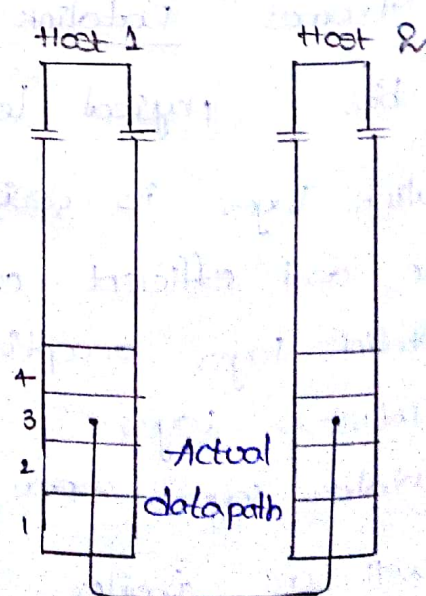
Datalink layer logically connected from source to datalink layer destination machine is through virtual communication layer the datalink layer accepts the packets converts into the frame data forward to physical layer. from physical layer of source machine to connect physical through actual communication path.

Network layer services:-



(a)

Virtual communication



(b)

Actual communication.

Data link layer implement possible services offered.

1. unacknowledged connectionless service

2. Acknowledged connectionless service

3. Acknowledged connection-oriented service.

Data link layer implements even acknowledgement policy in connectionless services.

Unacknowledged connectionless service:-

* It consists of having the source machine send independent frames to the destination machine without having the destination machine acknowledge them.

Eg:-

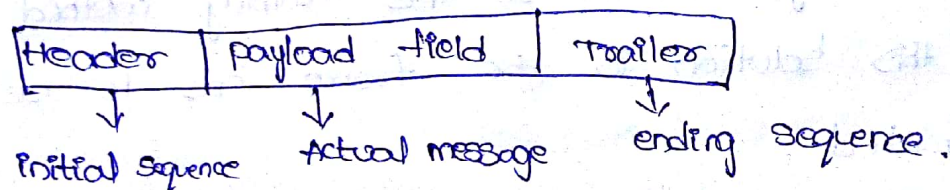
Ethernet, voice over IP, etc. in all the communication channel where real-time operation is more important than quality of transmission.

Framing:- The process of converting packets at the sender side binaries at the receiver side into frames is known as framing.

* Actually framing process is too difficult to implement. Framing

* Framing is a specific functionality implemented by data link layer.

* The general frame structure is consisting three parts.



* Depending on the framing process the trailer sequence is optional.

* They have been proposed four various trailing methods by

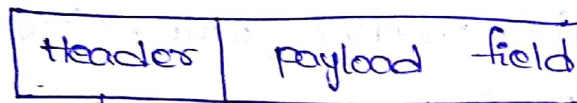
the datalink layer.

1. Byte count
2. Flag bytes with byte stuffing
3. Flag bits with bit stuffing
4. physical layer coding violations.

Byte count Framing Method:-

* Byte count framing method is a specific type of framing method which is implemented by data link layer.

* The frame structure in this framing method is generally two parts like



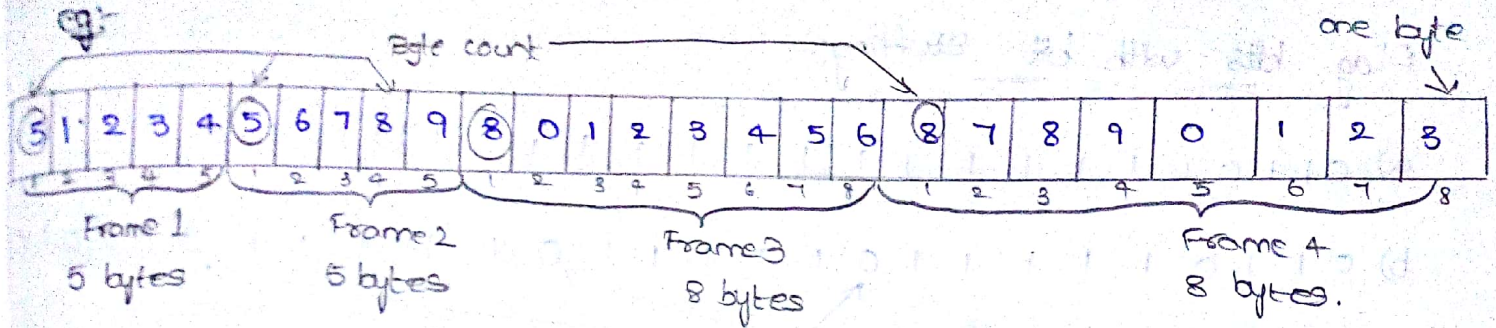
↓
Initial sequence

↓
Actual message.

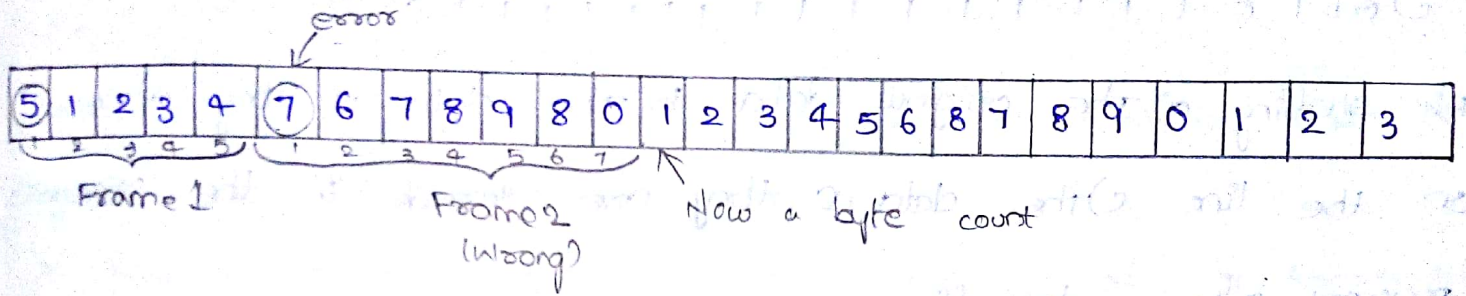
* In this framing method the header part of a frame structure is specifying byte count which gives the total amount of bytes that are transported from source to destination.

* Once the header information is being received by receiver it will be used to determine the end of the frame.

* Problem with this framing method is when the header information (Byte count) is incorrect then it will be effected to lead wrong messages to the current frame as well as to the other frames also. So, that the total message transmission get out of synchronization. In this framing method to avoid this problem this solution is the frame has to be entirely re-transmitted.



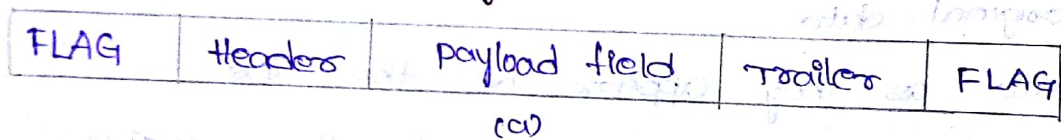
(a)



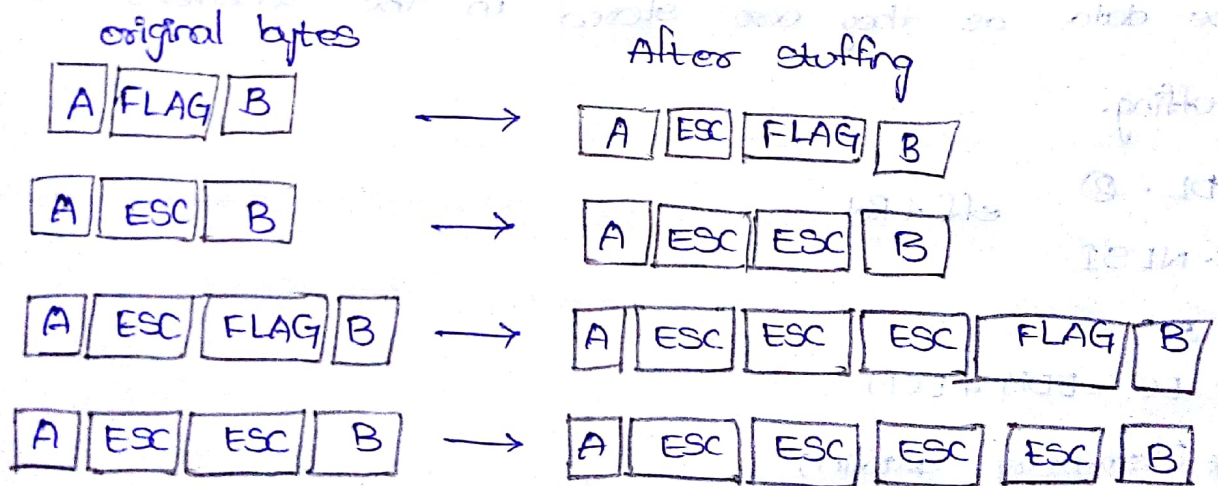
(b)

A byte stream a) without errors b) with one error.

2. Flag bytes with byte stuffing:-



(a)



(b)

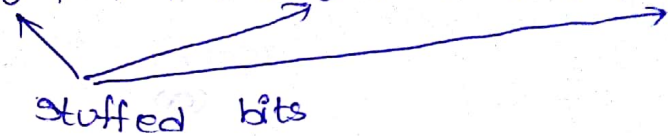
a) A frame delimited by flag bytes

b) two examples of byte sequences before and after byte stuffing.

Flag bits with bit stuffing:-

a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

b) 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0



stuffed bits

c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

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

Physical layer coding violations:-

Bit stuffing (a)

a) the original data

b) the data as they appear on the line

c) the data as they are stored in the receiver's memory after destuffing.

DL - ②

eff + Rel

1. NLSI

2.

EC = EDM + ECM

* Elementary datalinks

* Sliding window

Elementary data link protocols:-

- Utopian simplex protocol
- Simplex stop-and-wait protocol for Error-free channel.
- Simplex stop-and-wait protocol for Noisy channel.

Utopian simplex protocol:-

* This protocol is also known as protocol 1 of datalink layer.
In this protocol it provides the data transmission (both sender and receiver) in one direction only.

* The communication channel is assumed to be error free channel and can quickly process infinite data frame.

problem with protocol (1):-

* Data loss because of simplex data transmission b/w sender and receiver.

* channel capacity will be wastage because of re-transmission

typedef enum {frame arrival} event type;

#include "protocol.h"

void Sender1(void)

{

frame s;

packet buffer;

while (true){

frame network layer (&buffer);

s.info = buffer;

to physical layer (&s);

}

}

....

void receive1(void)

{

frame r;

event type event;

while(true){

wait for event(&event);

from physical layer (48);
to network layer (48.info);

}

}

Simplex stop-and-wait protocol for Error Free channel:

- * This protocol is also known as protocol 2 of data link layer
- * In this stop and wait protocol of data link layer to achieve error free data transmission the sender transmits data frames and stops the data frame transmission until as Receiver waits until the complete data frame transmission arrives to the receiver after that only a receiver can utilize the channel for data frames transmission.

- * In this protocol also there is one direction data flow between sender and Receiver.

- * Even though in this protocol also there is a possibility to occur errors because of simplex direction of data-frame transmission b/w sender and Receiver.

- * The protocol 2 assumptions are 1) finite buffer capacity at receiver side to process finite no. of data frame.

- 2) still in protocol 2 it is assumed an error free channel b/w sender and receiver.

- * problem with protocol 2 is data loss because of sender prolongs channel utilization time of data-frame transmission from sender to receiver.

protocol 2 is better in performance when compared with protocol 1 but still there is a possibility to miss the

data-frame because of practical implementation problem with stop-and-wait protocol.

```
#include "protocol.h"
```

```
void sender2(void)
```

```
{
```

```
    frame s;
```

```
    packet buffer;
```

```
    event type event;
```

```
    while (true) {
```

```
        from network layer (&buffer);
```

```
        s.info = buffer;
```

```
        to physical layer (&s);
```

```
        wait for event (&event);
```

```
    }
```

```
} ...
```

```
void receiver2(void)
```

```
{
```

```
    frame r, s;
```

```
    event type event;
```

```
    while (true) {
```

```
        wait for event (&event);
```

```
        from physical layer (&r);
```

```
        to network layer (&r.info);
```

```
        to physical layer (&s);
```

```
    }
```

```
}
```

Simplex stop-and-wait protocol for noisy channel :-

* protocol 3 uses simplex direction data transfer b/w sender and receiver, still in protocol 3 sender sends a data frame and waits for an acknowledgement from the receiver machine so that even though in physical layer frames coded into signals with errors still protocol 3 can attempt to minimize the rate of errors generated through the noisy channel.

* protocol 1 is the least priority comparatively to implement with protocol 2 and protocol 3.

* protocol 3 is highest priority for practical implementations between sender and receiver for efficient frames transmission in Simplex data flow.

Sliding Window Protocols :-

* Data Link layer implements sliding window protocol for data frame transmission b/w sender and receiver by employing windows at the side of sender and receiver and this protocol introduced to implement bidirectional data frame transmission b/w sender and receiver.

* Because of uni-direction (simplex protocol) protocols of data link layer will results maximum data loss to reduce the data frames to the sliding window protocols are introduced

* Basically sliding window protocols are classified into 3 types.

- 1) 1-Bit sliding window protocol
- 2) Go-back-N sliding window protocol
- 3) selective repeat sliding window protocol

1-bit sliding window:-

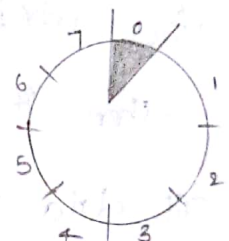
Sender



Receiver



(a)

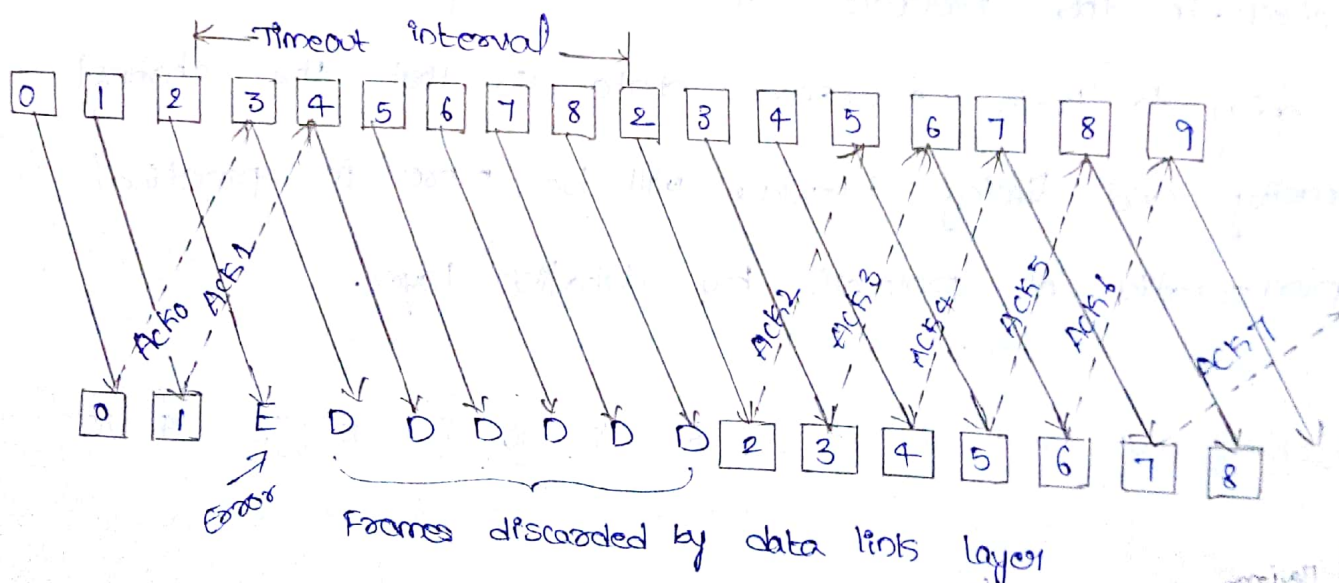


(b)

A sliding window of size 1, with a 3-bit sequence numbers.

a) Initially b) After the first frame has been sent.

Go-Back-N



time →

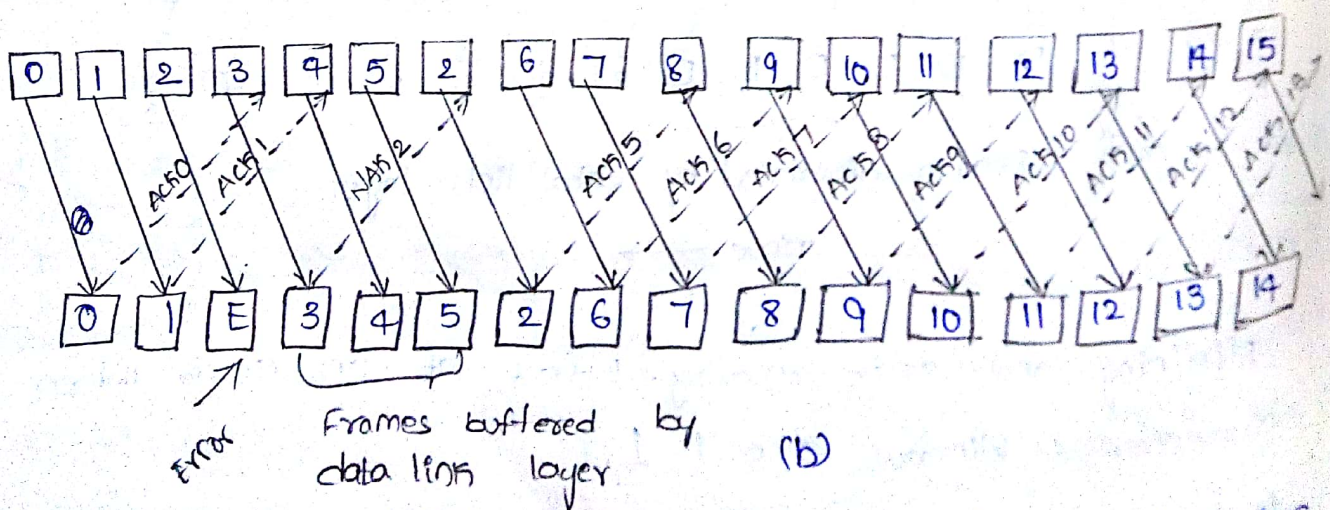
(a)

pipelining and error recovery. Effect of an error when

as receiver's window size is 1.

Go-Back-N sliding window protocol is known as protocols of data link layer. In this protocol the sender sends the list of data frames to the receiver upto the specified

timing interval where as the receiver has to be report its
 Acknowledgement back to the sender. The sender continuously
 transporting data frames till the time limit expires once the
 time limit is expired and verifies the receiver status with
 all set of acknowledgements for the sent data frames &
 the acknowledgement for which data frame is not receive by
 the senders window then it repeats retransmissions from
 sender to receiver all data frames successfully use go-back-N
 protocol will consider the error position at the receiver side
 and from that error position the remaining data frames are also
 discarded eventhough the sender transported data frames to the
 receiver. In this protocol the data transportation is repeated
 by going to Back error recovery state so that the channel
 capacity and timing intervals will be more in practical
 implementation of protocol 5 by datalink layer.



pipelining and error recovery. Effect of an error when

b) receiver's window size is large

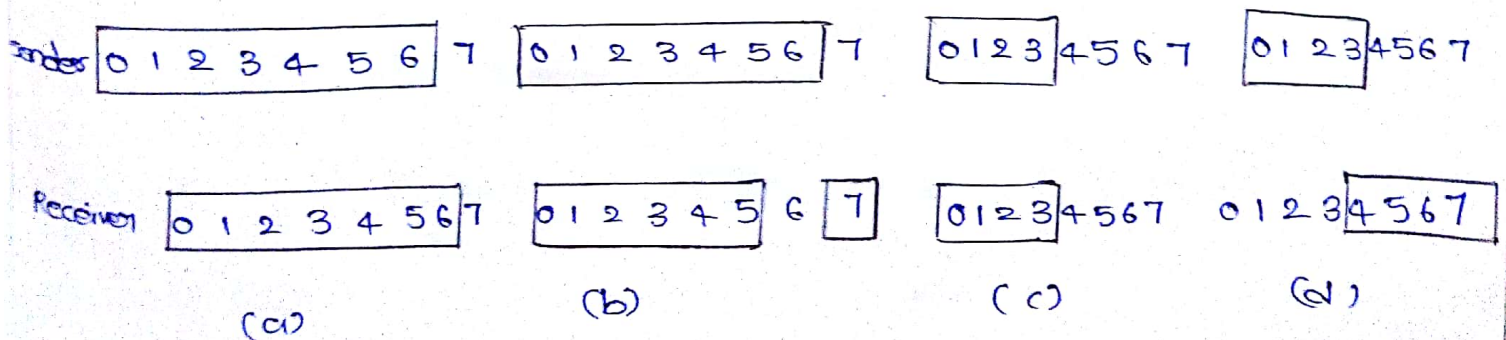
* In the above figure protocol using Go-Back-N in this case the sender is sending data frame to the receiver, and the receiver will respond with ACK if receiver receives data frame without any errors; * If the receiver is not received the data frame then it is an error (E) then the receiver responds NAK (Negative Acknowledgement) back to the sender.

* The error followed by remaining frames are generally buffer by datalink layer for error recovery. Once the errored data frame is retransmitted by the sender then the buffer frames are sequentially attach to form an actual message and the remaining frames are transmitted in a regular fashion from sender to receiver.

* Comparatively in the above case (fig a) in this fig(b) the go-Back-N is revised to minimize the number of data frames retransmission and also supports, only to re-transmit the missed (Error) data frame.

* protocols allows to transmit multiple outstanding frames upto Maximum Sequence frames without waiting for an Acknowledgement.

protocol using Selective Repeat (10)



a) Initial situation with a window of size 7

- b) After 7 frames sent and received but not acknowledged
- c) Initial situation with a window size of 4.
- d) After 4 Frames sent and received but not acknowledged.

~~Go - Back - N~~

Selective Repeat protocol (protocol 6) is selectively retransmit the data frame to the sender and accordingly the window size is resize.

Comparatively protocol 4 & 5 ; protocol 6 is able to transport maximum no. of data frames and minimizes no. of re-transmissions of data frames.

5/1/18

Error control :- Data link layer is implements error control

Error control service by using EDC (Error Detection codes) and ECC (Error correction codes).

* EDC & ECC are commonly implemented by data link layer to detect and correct the collected bit in the original message to implement reliability

* Datalink layer using error correcting codes is also referred as forward error correction (FEC) to correct single bit errors as well as multiple bit errors

* Datalink layer to implement error control service there has been defined four different error correcting codes.

*1. Hamming code

*2. Binary convolutional code

3. Reed - Solomon code

4. Low - Density parity check codes.

Hamming code:-

* Data link layer to implement error handling, it has to look at closely what the error has been a really occurred along with the message.

* The senders message and receivers message binary representation is consider to be a codeword,

* For example sender sending a message - codeword is 10001001

* The senders message transmitted as it is but the receiver receives the message - codeword is 10110001

* To implement hamming code by the data link layer has to be consider by the above given two codewords.

* Codeword 1 - 10001001

Codeword 2 - 10110001

Hamming Distance $\rightarrow$ 0011000
 $\rightarrow$ 3

(How many bits are differ from these two codewords then it is called as hamming distance.)

"XOR"	
0	0 $\rightarrow$ 0
0	1 $\rightarrow$ 1
1	0 $\rightarrow$ 1
1	1 $\rightarrow$ 0

* Hamming code applies "XOR" gate to handle the errors by finding out the hamming distance between the two given codewords.

* According to the hamming distance value the datalink layer identifies the error positions and applies its complement at the receivers codeword 2. so that the message which is wrongly delivered will be get a codeword 1 10001001 is transmitted by the datalink layer to the network layer on receivers machine.

X (and get corrected the receivers message)

* Senders message is 11110011 and receiver received the message as 00010001 apply

codeword1 : 11110011

codeword2 : 00010001

1^{st} complement $\overline{11100010}$
00011101

XOR

→ In this output where we are getting 1 that is error in codeword

* Hamming code is used to identify single bit as well as multiple bit errors and also get correct the receive wrong message into actual message. So that hamming code is also known as error detection and correction code.

* The senders message is consisting 'm' no. of bits, the receivers message may be consisting all 'm' no. of error bits (or) n no. of error bits but still the hamming code is able to detect the error positions as well as able to correct receive wrong message.

* The implementation of hamming code on circuit by the datalink layer is easy and inexpensive. The problem with hamming code is the complex linear code when sent by the senders message and received wrongly by the receiver; in this case hamming code implementation on circuit of datalink layer is difficult and expensive.

eg:- codeword 1 : 1 1 1 1

codeword 2 : $\begin{array}{cccc} 1 & 0 & 1 & 1 \\ \hline 0 & \boxed{1} & 0 & 0 \end{array}$

* A single bit error at second position of receiver's message at codeword 2 and the hamming distance is 1.

* The codeword 2 wrongly received message bit position is 2 and the bit is '0'. Now the hamming bit 0 has to be applied with 1's complement now the value is '1'.

* The corrected codeword 2 is 1 1 1 1

NOTE:-

* Data link layer uses error correcting code to achieve reliability by avoiding errors but these error correcting codes add redundancy to the information which is sent by the sender.

* Data link layer uses block code, system code and linear code to implement error control service.

* Most of the networks are using hamming codes for error corrections which is valuable for understanding block codes.

* Still to avoid the errors of the data many of the networks are using stronger codes than hamming code.

Binary convolutional Code (BCC):-

* The BCC code is stronger than hamming code and this code is widely used in deployed networks like GSM (Global System for Mobile Communication). BCC code is not a block code.

* In BCC an encoder processes a sequence of input bits and generates a sequence of output bits.

* There is no natural message size or encoding boundary like in block code.

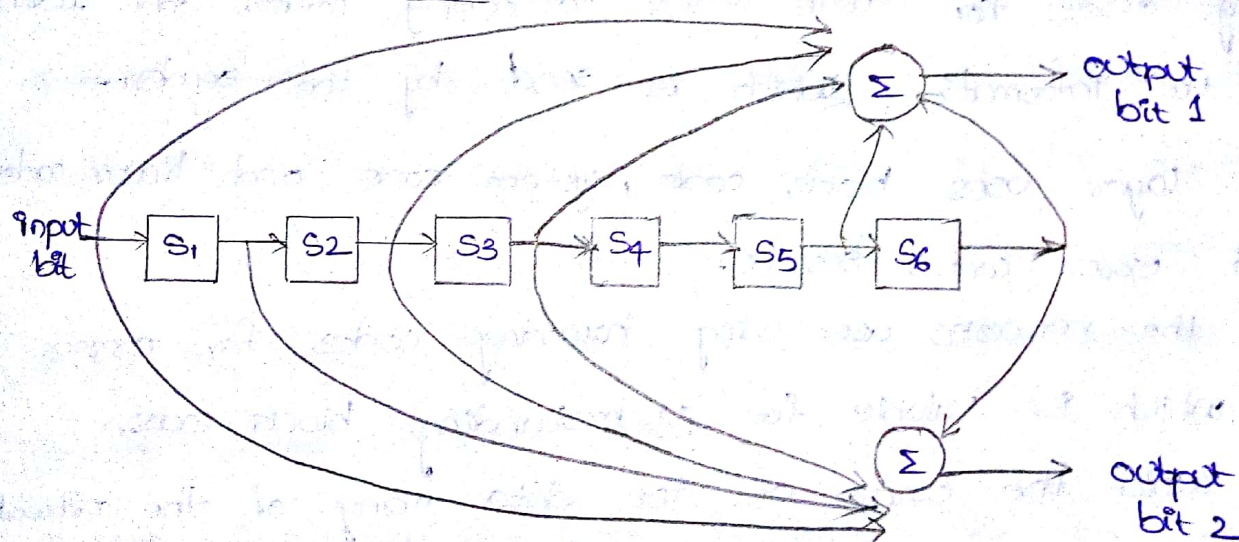
* In BCC the output depends on current and previous input bits i.e. the encoder has enough memory space to store previous bits.

* The no. of previous bits on which the output depends is called the "constraint length" of the code.

* Convolutional codes are specified in terms of their rate and constraint length and these are most widely used in satellite communications, GSM and 802.11.

→ ~~Serial~~ This code is also known as NASA binary convolutional code.

Error Detection code:-



* The NASA binary convolutional code used in 802.11

* In the above figure each input will bit on left side produces two output bits on the right side that is "XOR" of input and internal state.

* Deals with bits and linear operations produce proper output at the side of datalinks layer receiver.

* In the above figure each input produces two output bits and code rate is $\frac{1}{2}$.

* The above circuit is implemented by the systems data link layer along with gates and flipflops.

Reed-Solomon code

* The third kind of error correction code is Reed-Solomon code. Just like Hamming code, this code is linear block code and are used as systematic circuit codes.

* These codes are implemented by data link layer and involve lot of mathematical operations and are based on polynomial equations.

* These codes are actually defined as polynomials which are operate over finite F and widely used in DSL, Satellite communication, blue ray disks. and these codes are popularly known as strong error correction codes.

* For example a line having the form $ax+b$ is determined by two points. Extra points on the same line are redundant which is helpful for error correction.

* The two data points of the line and the two extra check points on the line are used for error correction.

Low density parity check - (LDPC)

* LDPC is a linear block code invented by Robert in 1962 for computing errors to achieve reliability by data link layer.

* In LDPC each output bit is form from only a fraction of $1/p$ bits. This leads a sparse representation of the code that has a low density of 1's hence the name given for that code.

* The receiver codes are decoded with an approximate algorithm which iteratively improve the best fit data and corrects the errors.

* The LDPC are practically implemented for large block size

and have an excellent error correction abilities in practice.
* These are used in ethernet power line networks and later version of 802.11

5/11/19
Error Detecting codes (EDC):-

- 1) parity bit Error Detecting code
- 2) checksum Error Detecting code
- 3) cyclic Redundancy Check (CRC) Error Detecting code.

* The above three EDC are implemented by datalink layer to detect the errors on actual message to implement error control service.

* Error detecting codes are used to find out single bit as well as multiple bit errors to achieve reliability.

parity Bit Error Detecting codes (EDC):-

* parity Bit EDC is used on wireless links when the data is transported using co-axial cable from source to destination. The parity bit error detecting code is used to detect single data bit error in sender's codeword.

* The parity bit is an extra appended bit in the given codeword and the system datalink layer will decide to implement either even parity error detecting mechanism (or) odd parity error detecting mechanism.

* An even parity of code word should be carries the total no. of 1 bits evenly.

* The odd parity of codeword will carries the total no. of 1 bits is odd number.

* For example the given codeword is consisting total no. of 8 bits and apply even parity all the given codeword is

1 1 1 1 1 0 0 0

* Even parity 1 1 1 1 1 0 0 0 1 ← extra bit

* If a datalink layer implements odd parity error detecting mechanism then the given codeword is 1 1 1 1 1 0 0 0

odd parity 1 1 1 1 1 0 0 0 0 ← extra bit

eg:- 1 1 1 1 1 1 1 1

eg:-

1 1 1 1 1 1 1 1

even parity 1 1 1 1 1 1 1 1 0 ← extra bit

odd parity 1 1 1 1 1 1 1 1 1 ← extra bit

Check Sum Error Detecting Code (EDC):-

* check sum EDC is implemented by data link layer to detect ~~single~~ ^{Multiple} bit errors. Whenever a sender sends a message which is consisting a group of bits, this block is received by receiver if the block is carrying more than a single bit error then the data link layer is implementing check sum error detecting mechanism to achieve reliability.

* The error control service of datalink layer by using check sum EDC is "By adding check bits" either side of sender as well as receiver.

* The check sum error detecting code with check total no. of check bits of the senders message as well as receivers message if both are same then there is no error. If a group of check bits calculated check sum is not similar either side of datalink layer then it seems to be there has been error errors in the codeword.

* The 16-bit check sum is commonly used in the internet packets of networks communications.

* The check sum is the sum of message bits will be divided individually and calculated the sum which is transferred to the data link layer of receiver.

* The check sum error detection code is usually placed at the end of the message, as the complement of the sum function.

* This way of detecting errors by summing the entire received codeword b with both data bits and check sum.

eg:-

N	-	0 0 1
E	-	1 0 0
T	-	1 0 1
W	-	1 1 1
O	-	1 1 0
R	-	1 1 1
S	-	0 1 0
		1 0 0

even parity

Cyclic Redundancy Check (CRC)

* CRC is an error detection Mechanism

* CRC is a stronger errors detecting Mechanism and also known as "polynomial code".

* CRC is implemented with help of check sum calculated by "XOR" gate as well as polynomial equations.

* The polynomial generator circuit is generates the polynomial equation which is use to division the frames data.

$$F(x) = x^4 + x + 1$$

P.E $\downarrow$

1 0 0 1 $\downarrow$

$G(x) = \text{Frames data} \}$ Receiver's

Input $\rightarrow$ CRC

CRC = $F(x)$ the polynomial equation has to be divide $G(x)$

* To implement CRC the $F(x)$ and $G(x)$ division has to be done along with "XOR" gate.

* consider $F(x)$ highest order value and add that many no. of 0's to the $G(x)$ value.

$F(x) = 10011$
 $2^4 2^3 2^2 2^1 2^0$
 19

110000110									
1101011110000									
10011									
010011									
10011									
000001									
00000									
000011									
00000									
000111									
00000									
001111									
00000									
011111									
00000									
011110									
10011									
011010									
10011									
010010									
00000									
000100									
00000									
00100									

$$f(x) = x^3 + x + 1$$

* * * $g(x) = 1110011.$

*** $g(x) = 1110011$.
*** Illustrate cyclic Redundancy check (CRC) to find out the errors in the senders Message.

$$F(x) = x^3 + x + 1$$

1011

$$G(x) = 1110011 \boxed{000}$$

[illegible]

$G(x)$ is received by the receiver as it is there is no error from the sender's message.

$$F(x) = 10011$$

$$G(x) = 11010111116010$$

