Data Link Layer :-

Error Control in Data Link layer :-

During the communication in the network there is possibility of a noise getting added in the network. This noise is known as an "error".

Types of error :-

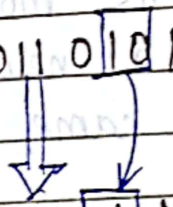
1) Single bit error :-

Eg. S^x Data - 10101101

R^x Data - 10001101

2) Multi-bit error :-

Eg. S^x Data : 10110101



R^x Data : 10110011

Error detection in data link layer :-

1) Parity check

In this a extra parity bit will be added to make the data a even parity data, the data & parity bit will be send to the

receiver through the network & receiver would perform the parity check & would accept the data & there is no error.

Eg. S^x Data : 1011 0101 [1]

R^x Data : 1011 0101 [1]

- This will probably work well for single bit error it may fail in case of multi-bit error

Eg. S^x Data : 1011 0101 [1]

1010 0101 [1]

- To solve the above problem used two level parity check.

Eg.

1110 1110	0
0101 0101	0
1101 1101	0
1110 1111	1
1001 0001	

Cyclic Redundancy Code.

Q:1] Consider a message represented by polynomial $M(x) = x^5 + x^4 + x$. Consider a generating polynomial $G(x) = x^3 + x^2 + 1$. Generate a CRC & show what would be transmitted & also specify how CRC detects an error.

Soln:- Here,

Sender side calculation :-

$M(x) = 110010$

$G(x) = 1101$

```
1101 | 110010 0000
       1101
       -----
        0011 00
          1101
          ----
          000 100   ← remainder
```

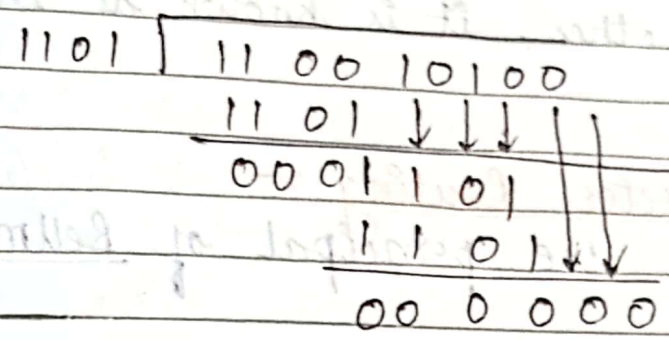
∴ Data to be transmitted = 11 0010 100

- At receiving side same operation would be performed with same divisor, if result generated is zero then no error, accept the data, if result is non-zero, error generated discard.

CN NOTES BY PROF. AKN

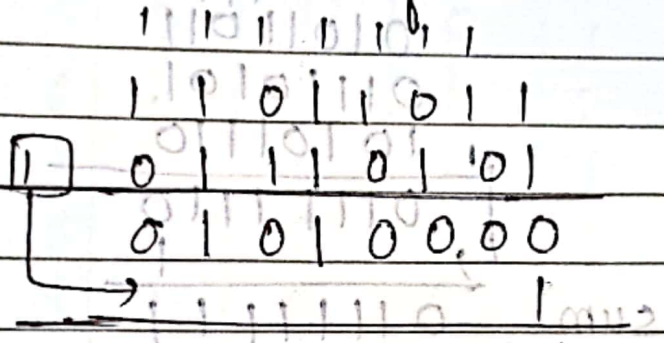
Page No.
Date

Receiver side calculation :—



Unit 3) Checksum Calculation

All the sender side divide the data which Need to be transmitted into k section each of n bits.
For eg. Data to be transmitted is 11011011 01110101
∴ 16 bits of data is divided into K=2 sections each of n=8 bits.
Now, perform binary addition calculate the sum take 1's complement of the sum and attach the checksum at end of data.



1's complement 1 0 1 0 1 1 1 0 ← checksum.

Data to be transmitted.
11011011 01110101 10101110.

At the receiving side, either the data is transmitted error free or it has error.
- Perform the same operation at the receiving side, if final result is zero there is No error accept the data else rejected.

Page No.
Date

Case 1 :- No error

Data R^x = 11011011 01110101 10101110

```
  1 1 1 1 1 1 1 1
  1 1 0 1 1 0 1 1
  0 1 1 1 0 1 0 1
+ 1 0 1 0 1 1 1 0
-----------------
  1 1 1 1 1 1 1 0
      ↳       1
-----------------
sum 1 1 1 1 1 1 1 1
```

1's complement 00000000 ← Result = 0 Data is accepted.

Case 2 : Error

Data R^x = 01011011 01110101 10101110

```
  0 1 0 1 1 0 1 1
  0 1 1 1 0 1 0 1
+ 1 0 1 0 1 1 1 0
-----------------
  0 1 1 1 1 1 1 0
      ↳       1
-----------------
sum 0 1 1 1 1 1 1 1
```

1's complement 10000000 ← Result ≠ 0 Data is rejected.

Error detection and correction.

Hamming code :-

It is an error correction and detection mechanism which uses redundant bits are to be added in the data using the following criteria which should be fulfilled.

$$2^k \geq m + k + 1$$

k = No. of redundant bit.

m = No. of bit in data.

So to transmit seven bits of data [m=7] 4 redundant bits can be used!

2^4 [16] $\geq$ 7+4+1 [12]

So a 4 bit hamming code can be represented in following manner.

0	0	0	r_8	0	0	0	r_4	0	r_2	r_1
11	10	9	8	7	6	5	4	3	2	1

	r_8	r_4	r_2	r_1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1

r_1 = 1, 3, 5, 7, 9, 11

r_2 = 2, 3, 6, 7, 10, 11

r_4 = 4, 5, 6, 7

r_8 = 8, 9, 10, 11

Transmitting 7 bits of data [1001100]

Sender side calculation :-

1	0	0	r_8	1	1	0	r_4	0	r_2	r_1
11	10	9	8	7	6	5	4	3	2	1

r_1 11 9 7 5 3 1

1 0 1 0 0 $r_1 = 0$

r_2 11 10 7 6 3 2

1 0 1 1 0 $r_2 = 1$

r_4	7	6	5	4
	1	1	0	$r_4 = 0$

r_8	11	10	9	8
	1	0	0	$r_8 = 1$

Data to be transmitted :

1	0	0	1	1	1	0	0	0	1	0

Receiver side calculation :→

Case 1 : No error

∴ Data R^X = 1001 1100010

n	11	9	7	5	3	1
0	1	0	1	0	0	0

r_2	11	10	7	6	3	2
0	1	0	1	1	0	1

r_4	7	6	5	4
0	1	1	0	0

r_8	11	10	9	8
0	1	0	0	1

r_8	r_4	r_2	n
0	0	0	0

Case 2 :- Error

Data R^X = 1001110101 0

r_1	11	9	7	5	3	1	
0	1	0	1	0	0	0	
r_2	11	10	7	6	3	2	
0	1	0	1	1	0	1	
r_4	7	6	5	4			
1	1	1	0	1			
r_8	11	10	9	8			
0	1	0	0	1			

r_8	r_4	r_2	r_1
0	1	0	0

Error is at r_4.

So change the bit at r_4.

Framing

- Framing is a unit of transmission using data link layer.
- It is a digital data transmitted in a link layer.
- Data link layer accepts the packet from network layer. converts that into collection of bits known as frames & finally transmits this to a physical layer which further forwards the data through network in the form of Raw bits.

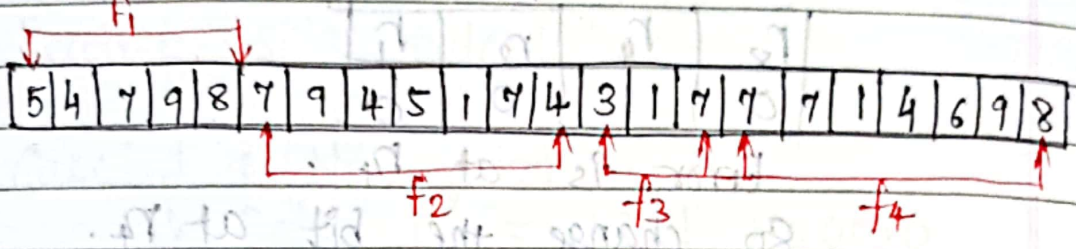
Types of framing:-

1) Character or byte count.
2) Bit stuffing.

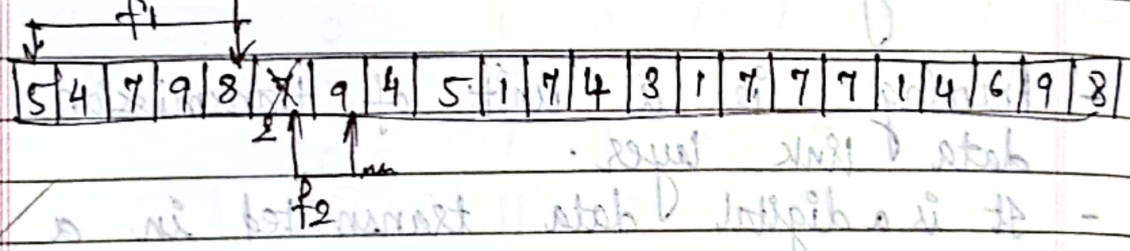
3) Byte stuffing.
4) Physical layer coding. (Manchester encoding)

① Character or byte count.
- The data consist of a counting variable and data variable.
- The first bit is counting variable.
- No. of bits = Counting variable.

Eg.



- If during data transfer the counting variable is changed then all the data variable is misinterpreted.



② Bit Stuffing:
- In bit stuffing flag is inserted at both the ends of data (flag = 01111110)
- If there are 5 consecutive 1's in the data add extra 0 after 5 consecutive 1's.

Eg. Data 0101101111111101111110111.

Data transmitted 01111110 0101101111101 10 11111010111111001

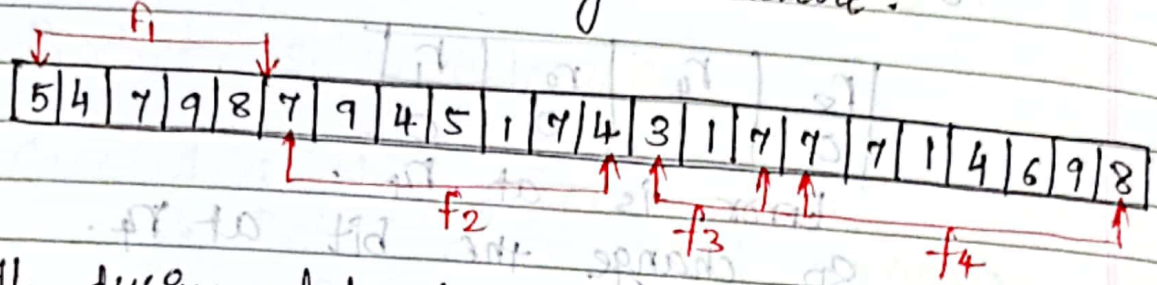
Data received 01111110 01011011111010111111010111111001

[Discard the inserted 0 & flag -]

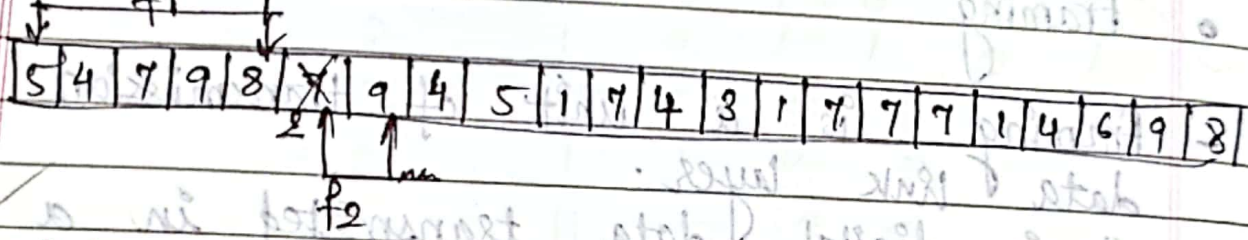
3) Byte stuffing.
4) Physical layer coding. (Manchester encoding)

① Character or byte count.
- The data consist of a counting variable and data variable.
- The first bit is counting variable.
- No. of bits = Counting variable.

Eg.



- If during data transfer the counting variable is changed then all the data variable is misinterpreted.



② Bit Stuffing:
- In bit stuffing flag is inserted at both the ends of data (flag = 01111110)
- If there are 5 consecutive 1's in the data add extra 0 after 5 consecutive 1's.

Eg. Data 0101101111111011111101111.

After stuffing

Data transmitted 01111110 01011011111011011111010111110 01111100

After destuffing

Data received 01111110 01011011111011011111010111110 01111110

[Discard the inserted 0 & flag]

CN NOTES BY PROF. AKN

Error

1) If the added extra 0 is changed during data transfer.
2) If one of the 5 consecutive 1's is changed to 0 during data transfer.

3) Byte stuffing

- Byte stuffing is also known as character stuffing.
- In this the flag is stuffed at the start & end of data.

for Eg.

S^X M N O

↓

After stuffing STX DLE M N O DLE ETX

↓

After Destuffing ~~STX~~ ~~DLE~~ M N O ~~DLE~~ ~~ETX~~

↓

R^X M N O

- If a data contains any component of the flag, an escape sequence (DLE) is stuffed before that flag component.

S^X M DLE N DLE O DLE

After stuffing STX DLE M DLE DLE DLE N DLE DLE O DLE DLE DLE ETX

↓

After destuffing ~~STX~~ ~~DLE~~ M ~~DLE~~ DLE N ~~DLE~~ DLE O ~~DLE~~ DLE ~~DLE~~ ~~ETX~~

R^X M DLE N DLE O DLE

- So the character stuffing for the following data:

CN NOTES BY PROF. AKN