Unit - II Physical & Datalink Layer

OSI Model :-

→ OSI stands for open system Interconnection is a Reference model that describes how information from a software application in one computer moves through a physical medium to the software application in another computer.

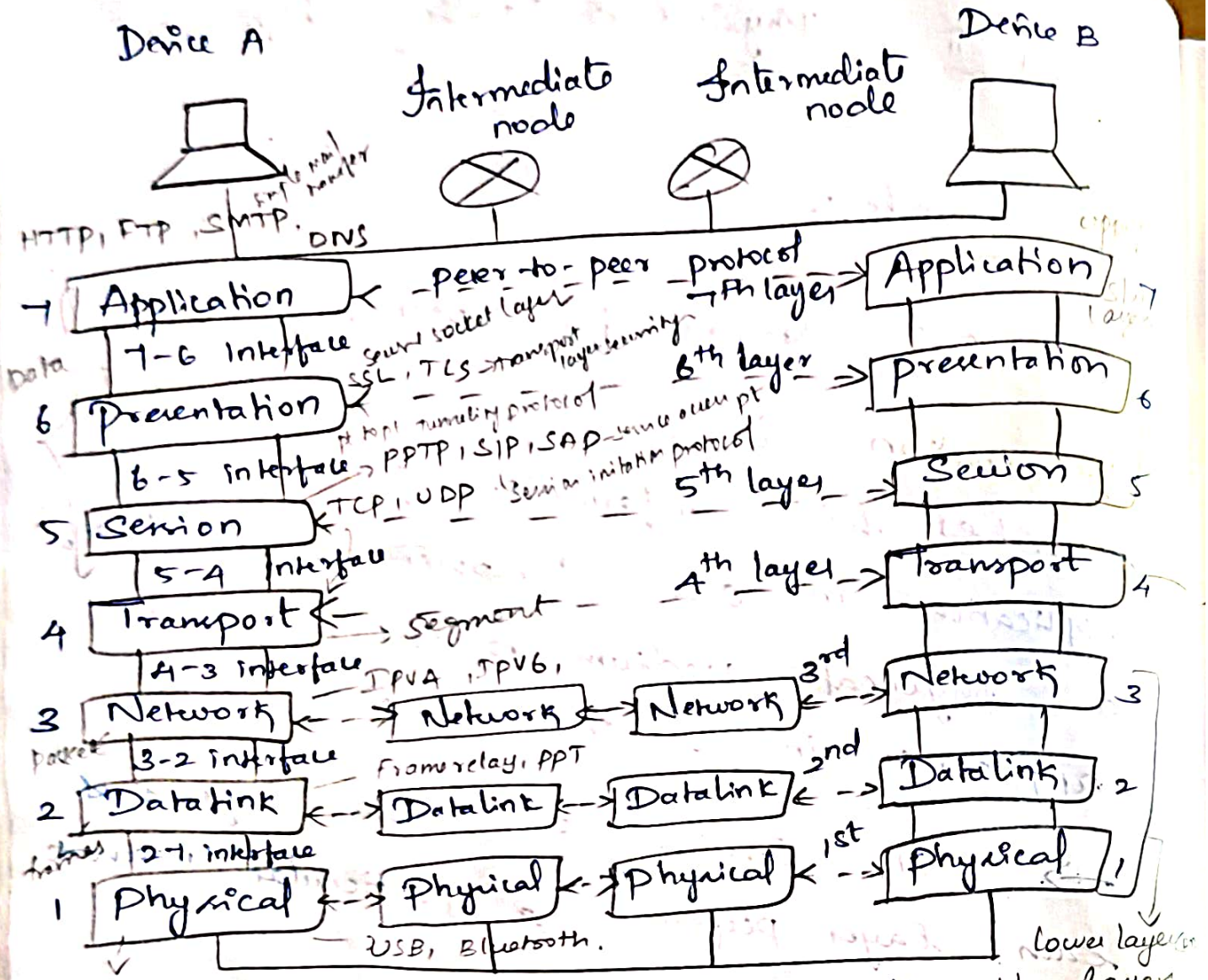
→ OSI consists of seven layers & each layer performs a particular network function.

→ OSI model was developed by the International organization for standardization (ISO) in 1984.

→ It is divided into a layers: upper layer & lower layers.

Application, presentation, Session, Transport → Responsibility of the Host

Network, Datalink, Physical → Responsibility of Network.



Physical communication. H/w layer

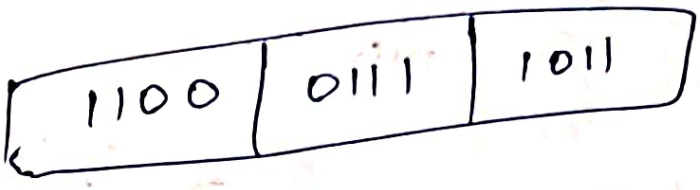
Physical layer:- (Hub, Repeater, modem & Cables are physical layer devices).

→ It is responsible for the physical cable or wireless connection between network nodes.

→ It contains the information in the form of bits. & responsible for transmitting individual bits from one node to the next.

→ When receiving data, this layer will get the signal received &

Convert it to 0's & 1's & send them to the Data link layer, which will put the frame back together.



Services provided by physical layer:-

→ Heart of OSI

→ physical Characteristics of the media

→ Representation of bits

→ Data Rate

→ synchronization of bits.

→ physical topology.

Data Link Layer:- will assign one address for each frames.

→ It is responsible for moving data (frames) from one node to another node.

→ The main function of this layer is to make sure data transfer is error-free from one node to another.

Services:-

→ Framing

→ flow Control

→ Error Control

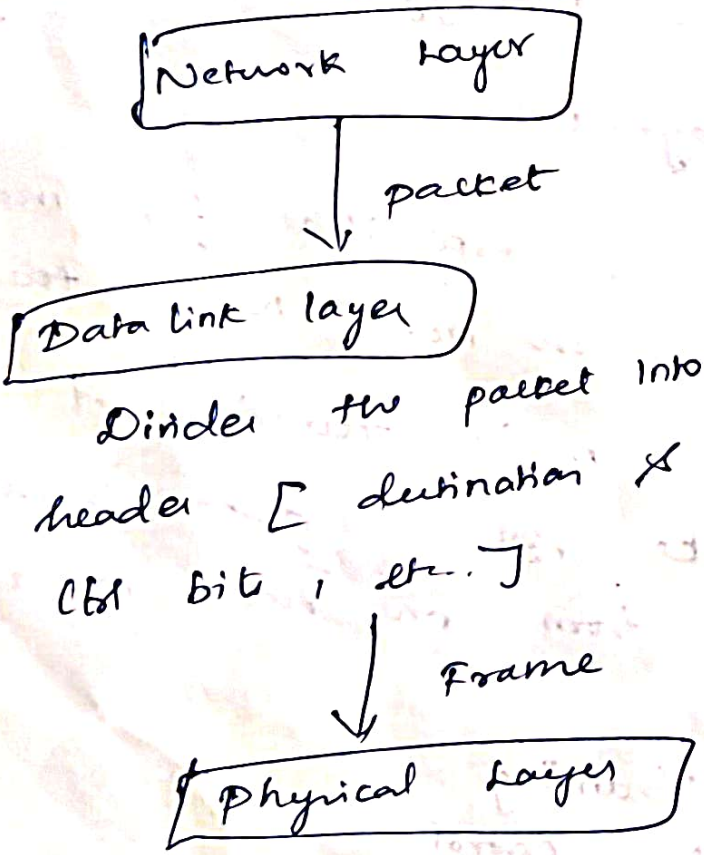
→ Access Control.

Network Layer:- addressing & routing.
Src & dst IP address are used.

→ It is responsible for delivery of data from the original source to the destination network.

→ It takes care of packet Routing i.e., selection of the shortest path to transmit the packet from source to destination.

Services :- → Logical addressing
→ Routing.



Transport layer:- (Heart of the OSI model)

→ It provides services to the application layer & take services from the network layer.

→ The data in the transport layer is referred to as segments.

segment segment no. port address

→ It provides the acknowledgment of the successful data transmission & re-transmits the data if an error is found.

Session layer :- → Dialog ctrl. Synchronization

→ This layer is responsible for the establishment of connection, maintenance of sessions & authentication & also ensures security.

Presentation layer :- → Translat encryption compression.

→ It is also called the translation layer.

→ The data from the application layer is extracted here & manipulated as per the required format to transmit over the network.

Application layer:-

→ It enables the user to access the network resources.

Eg:- Browsers, skype, messenger etc.

→ This layer is also called Desktop layer.

TCP/IP Model:-

→ TCP/IP model was developed prior to the OSI model. Therefore the layers in the TCP/IP protocol suite donot exactly match those with OSI model.

→ The functionalities of physical & datalink layer are combined into single layer called network access layer.

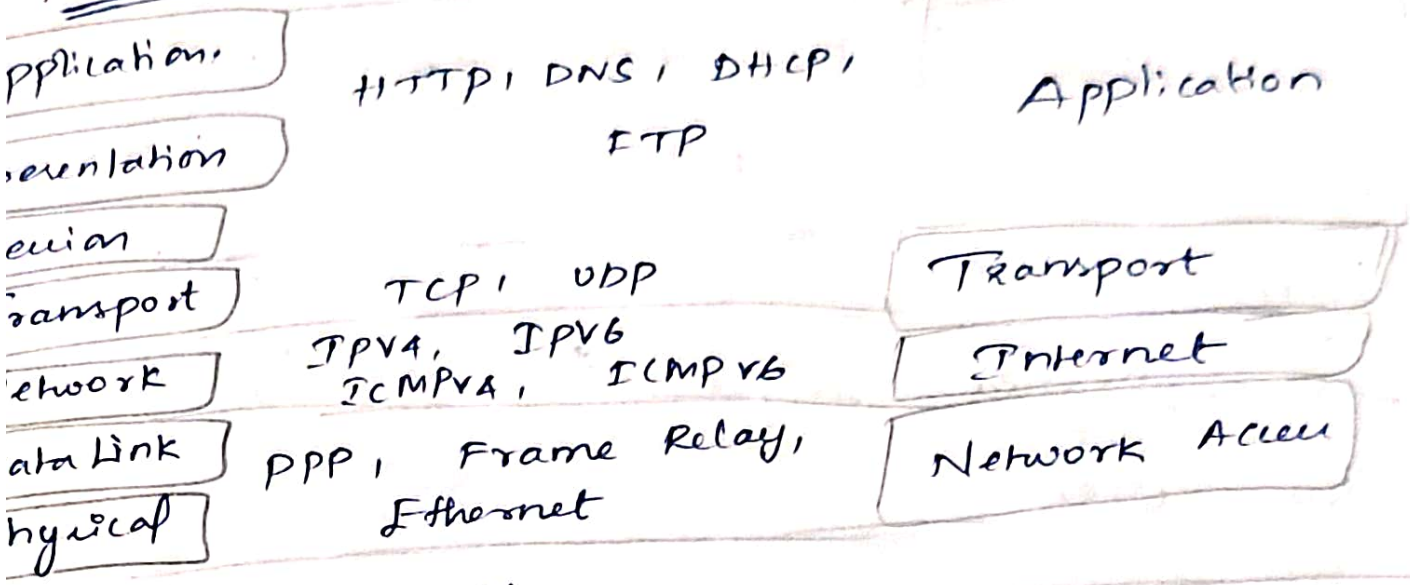
→ The network's layer is equivalent to Internet layer & the transport layer is same.

→ The functionalities of the session, Presentation & application layer is merged into single layer called application layer.

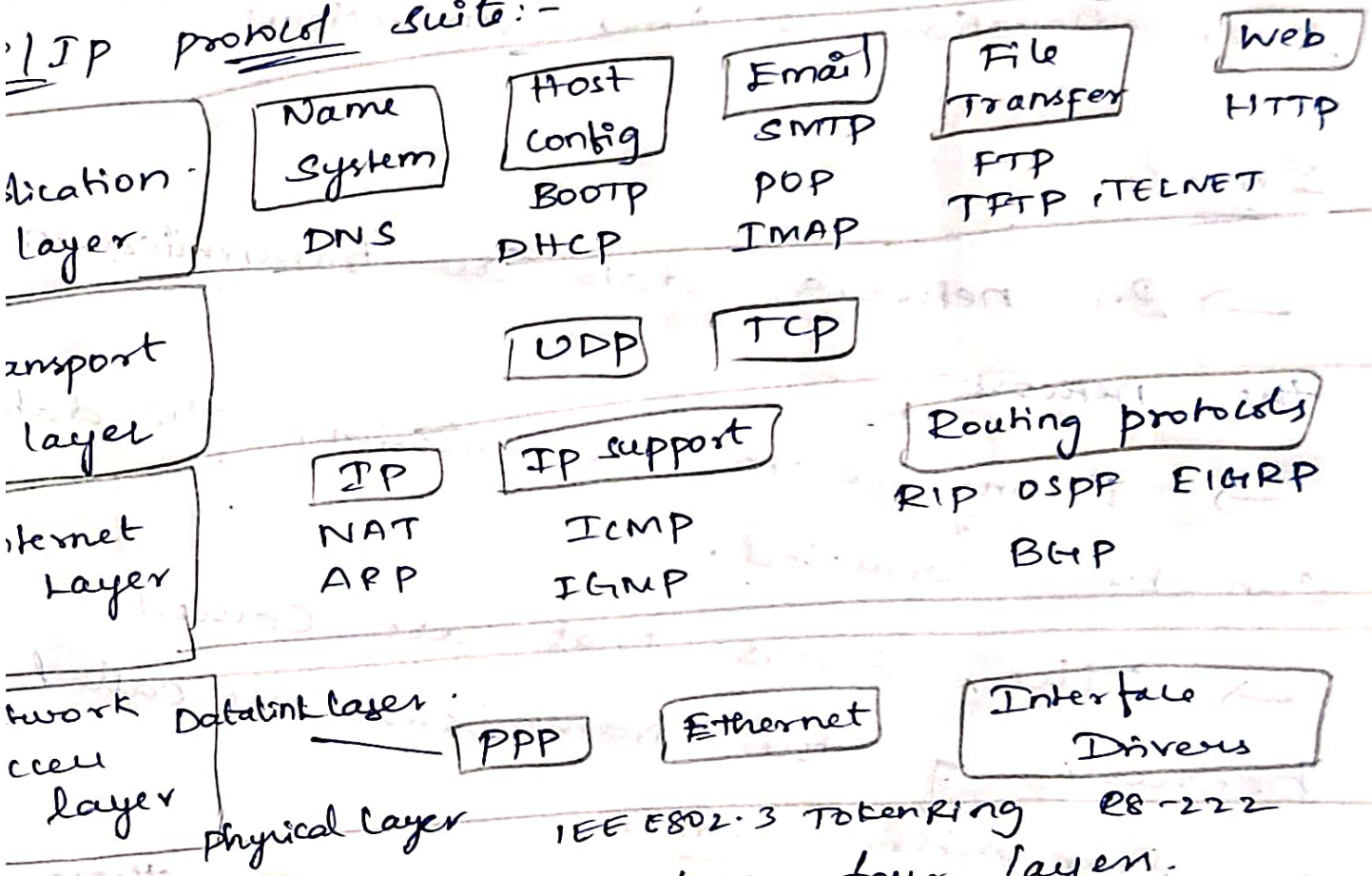
ICMP - Internet control Message protocol

IGMP - " Group management "

Ip model :-



IP protocol suite :-



→ TCP/IP model has four layers.

Network acces layer it controls the hardware because physical layer is a part of network acces layer, so it controls the hardware devices.

Internet layer is also known as network layer. , so it determines the best path through the network.

→ Transport layer, it supports communication between devices.

→ It deals with process to process communication.

→ Application layer is actually deals with the user.

Error Detection & Error correction:-

Error:-

→ In network, data are transmitted in the network.

→ During data transmission, the data can be corrupted.

→ The errors that are caused because of this transmission are called as transmission error.

→ For reliable communication, these errors must be detected & corrupted.

→ Error detection & correction are implemented either at the datalink layer or the transport layer in the OSI model.

Types of Error:-

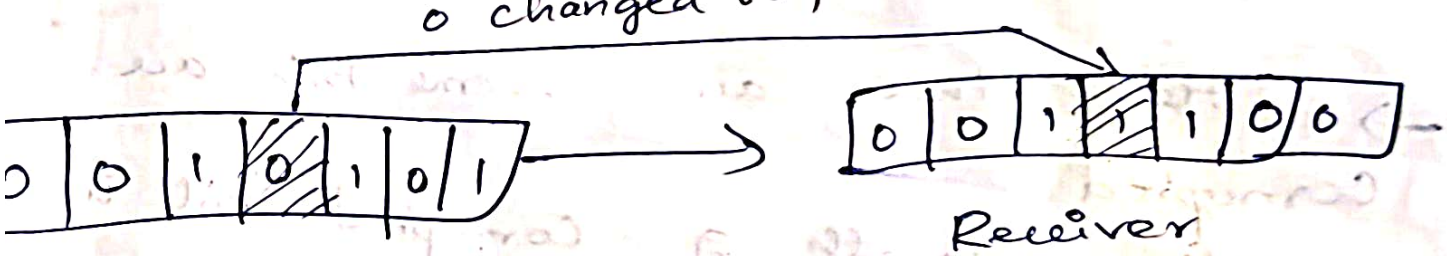
→ Bit Error, multiple bit error

→ Burst "

Bit Error:

→ In this bit error, only one bit is going to be changed.

→ It is also known as single bit error, i.e. only 1 bit in the data unit has been changed.



→ Eg:- From sender side to receiver sided 0 has been change to 1.

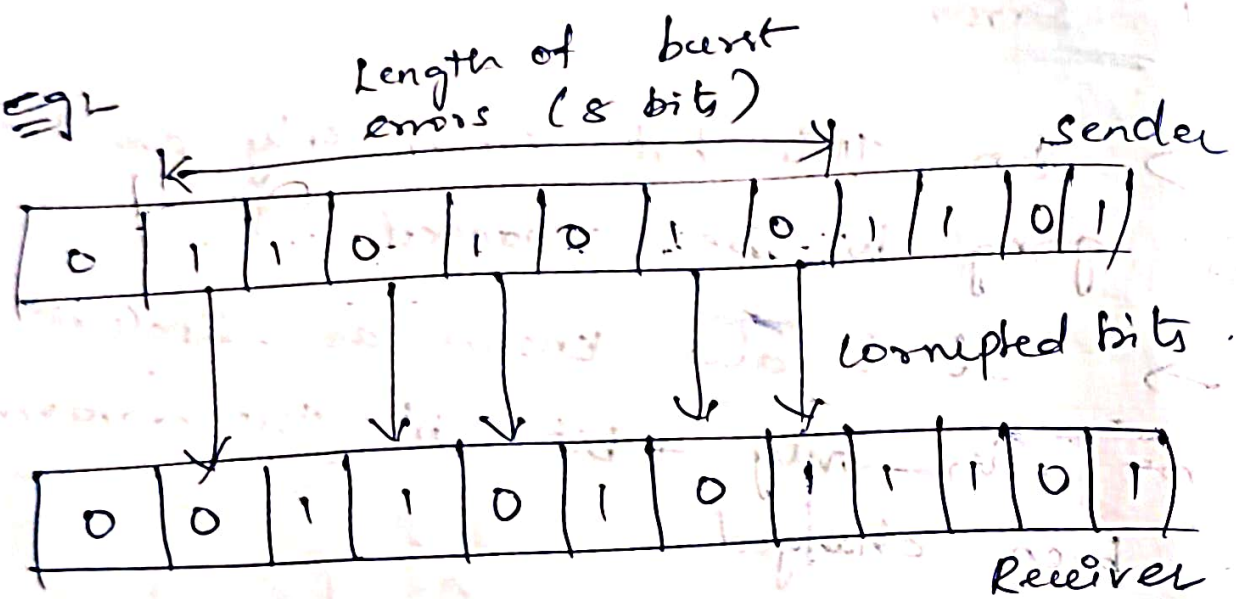
→ Sender is sending some information & receiver is receiving different information & there is a change in exactly one bit

→ When there is a only one bit change, it is known as bit error.

Burst errors:

→ In this, 2 or more bits in the data units have been changed

eg:-



→ Here, there are some bits are corrupted.

→ The length of corrupted bits is

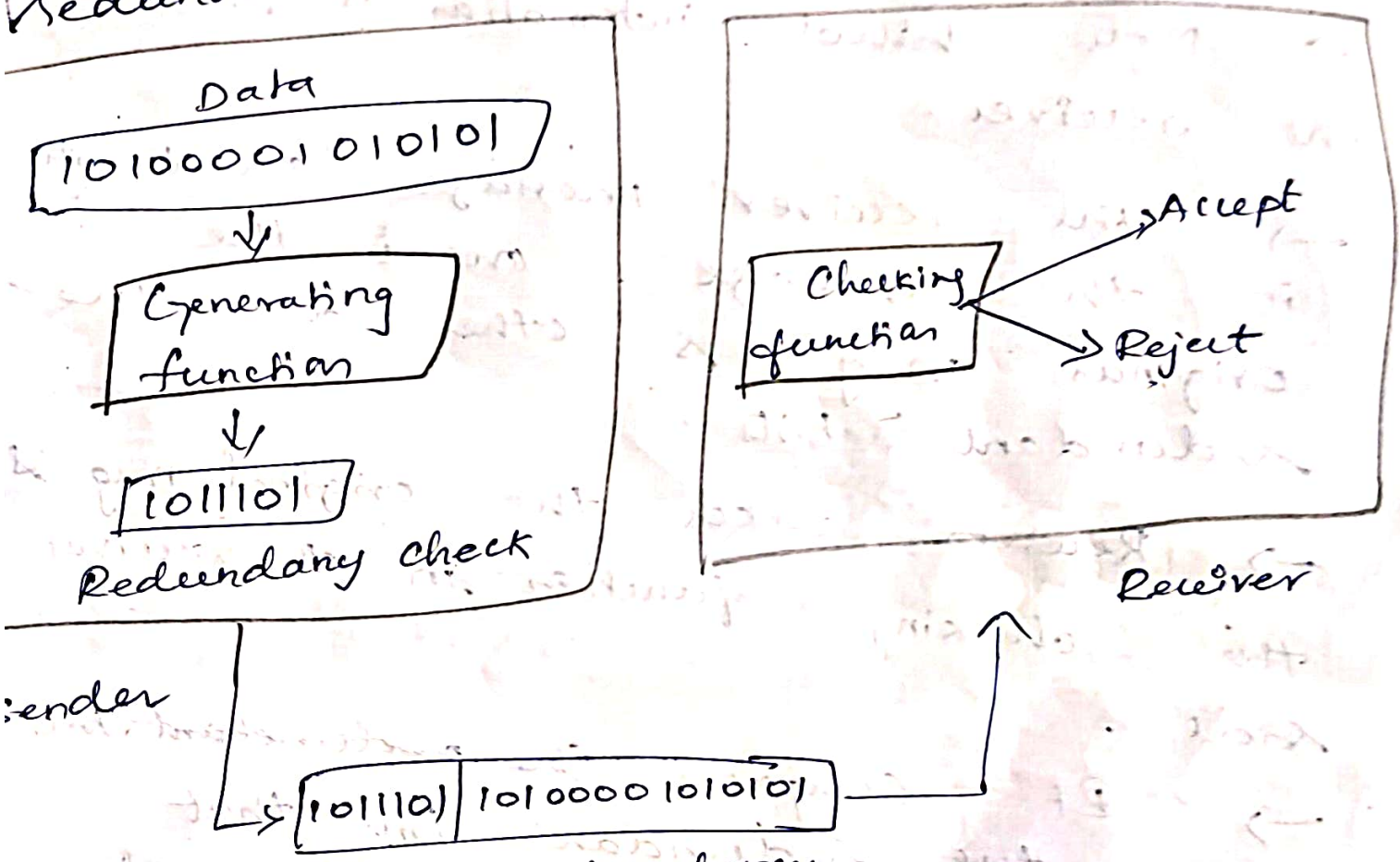
→ If there are 2 or more bits corrupted it is burst error.

How to detect the errors?

→ Error detection means to decide whether the received data is correct or not without having a copy of the original message.

→ To detect (or) correct errors, we need to send some extra bits with the data.

→ These extra bits are called as Redundant bits



→ Here, if the sender has this data to be sent, the sender gives the data to the generating function (or) an algorithm & this algorithm generates a code called as Redundant code (or) Redundancy bits (or) redundancy check.

→ This information is appended with the message that has to be sent

→ This redundancy check (or) redundant bits are added to the message.

→ Now, whole information is sent to the receiver.

→ The receiver message are 2 parts in the message. one is the original msg & other one is the redundant bits.

→ Receiver takes the original msg & the checking function in the receiver side., it calculate the redundant bits

→ & It compares the redundant bits with the redundant bits that was sent by the sender, if both are matching then the receiver understands that there are no errors in the transmission it accepts otherwise it rejects.

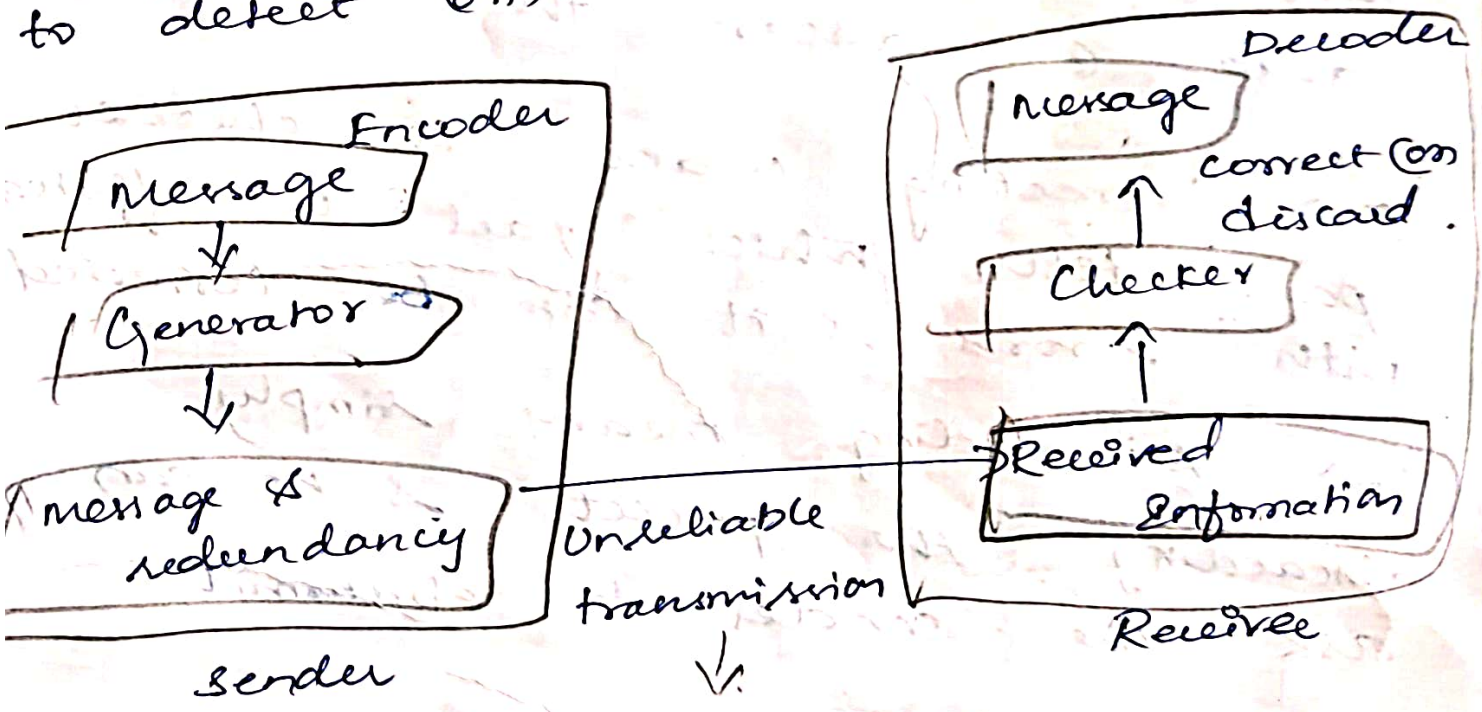
Error correction:-

→ It can be handled in 2 ways. - Forward - no retransmission
Backward → retransmission

↳ Receiver can have the sender retransmit the entire data unit.

↳ The receiver can use an error-correcting code, which automatically corrects the certain errors.

→ Both error correction & detection needs redundancy, because without additional information it's very difficult to detect (or) correct the errors.



→ The sender sends the data (or msg, there will be a generater (or algorithm to generate the redundany & this redundant information & the msg are generally sent over an unreliable transmission medium.

→ The receiver receiver this information whatever the receiver receives, it gives that information to the checker, this checker says, whether to accept the data (or reject the data. (or whether to correct the data (or discard the data.

→ correcting means, this checker part tells which part is affected with errors & it can be corrected.

→ Discarding means simply discarding the packet & we can ask the sender to retransmit.

Error detection techniques:-

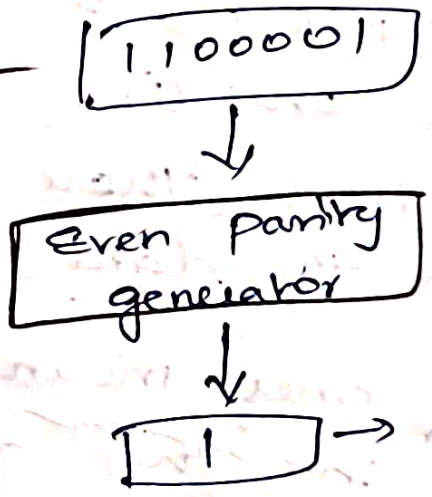
→ There are 4 types of error redundancy checks (or techniques that are used in data communications.

① Vertical Redundancy check (VRC).
② Longitudinal Redundancy check (LRC)
③ checksum. (or Internet (or Arithmetic Checksum)
④ Cyclic Redundancy check (CRC).

VRC :-

→ It is also known as parity check.

→ eg :- 1100001 Data generated by sender



→ Redundant bit (or VRC bit (or) parity bit.

VRC : 1 → if odd number of 1's
VRC : 0 → " even " " 1's

→ suppose, that data generated by the sender is 1100001 is of 7 bits.

→ Now, this data is given to the even parity generator & that accepts the data & generates a parity bit.

→ After accepting the data, the even parity generator will checks for the number of ones that are there in the data.

→ there are 3 ones (1100001) & 3 is an odd number, so, it creates the VRC, that is the redundant bit which is called as a VRC bit as one (1).

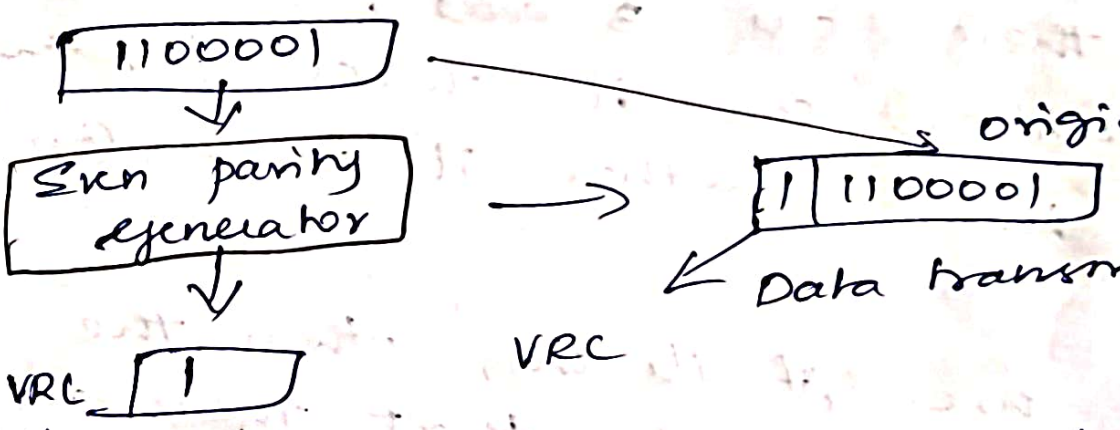
→ 1 → indicate that there are odd number of ones in the data.

→ odd num of ones means, there may be only one 1 in the data (or) 3 number of ones (or) seven numbers of ones in the data.

→ So, in this case there are 3 number of ones, so it generates one as the parity bit

→ This parity bit is also called redundant bit, VRC bit (or parity check.

→ Now, it appends this ~~data~~ one with data & sends to the receiver.



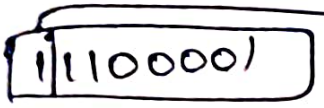
→ Now, this information is send to the receiver, if there are no errors during transmission, the receiver receives the information as 11100001.

→ Now, the receiver checks even number of ones in the received msg, if there are even number of ones, the receiver understands that there are no errors, if there are odd number of ones, the it understands

the msg was prone to errors & will rejects it.

→ How receiver checks the data:- ? (other way)

→ The Receiver knows that this bit will be parity bit,



so it places this bit aside & it checks how many number of ones in the actual msg.

→ If there are 3 ones, it means odd number, so it checks for the parity bit, whether it is one (or) zero.

→ For three, if it one, then the receiver understands that there are no transmission errors in parity bit.

Eg:-

① Append the parity bit after each block shown below using VRC.

1110110, 1101111, 1110010

LRC :-

→ In LRC, a block of data bits is organized in rows & columns.

→ LRC also known as 2-dimensional parity because of rows & columns.

→ The parity bit is calculated for each column & sent along with the data.

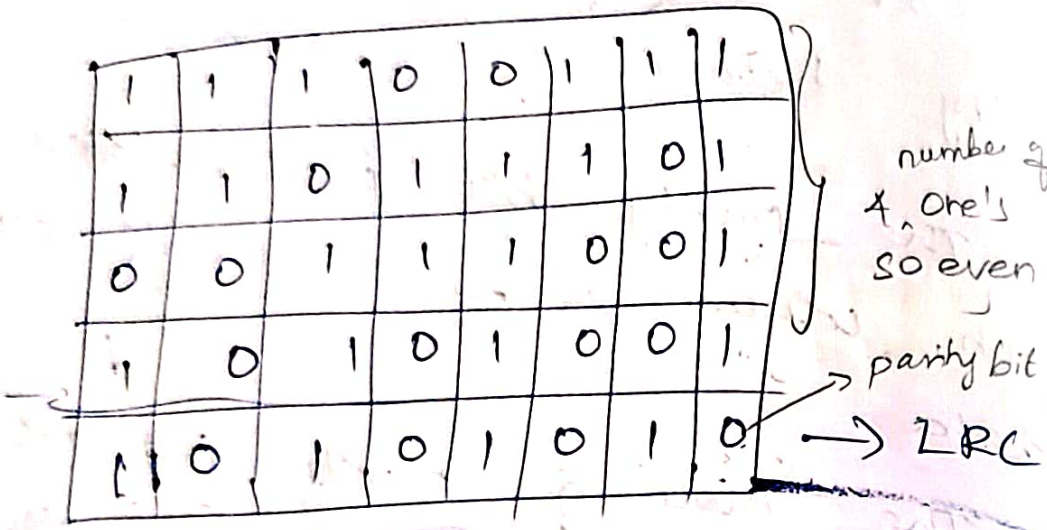
→ The block of parity bit acts as the redundant bits. ; because these redundant bit has to sent along with the data, so that the receiver can receive all the data block as well as the redundant bit.

→ These redundant bits are necessary for the receiver to check or detect whether there are errors in the transmission or not.

eg:- Find the LRC for the data blocks 11100111, 11011101, 00111001, 10101001 & determine the data that is transmitted.

Solution :-

Step 1 place the data blocks in every rows & columns.



Step 2 :- calculate the parity bit.

odd no of 1's 1

Even no of 1's 0

10101010 → This is the parity bit (or LRC bit. (or redundant bit.

→ so, the first 4 blocks are the original data & last block is the LRC.

Add => 0+0=0 0+1=1 1+0=1 1+1=0.

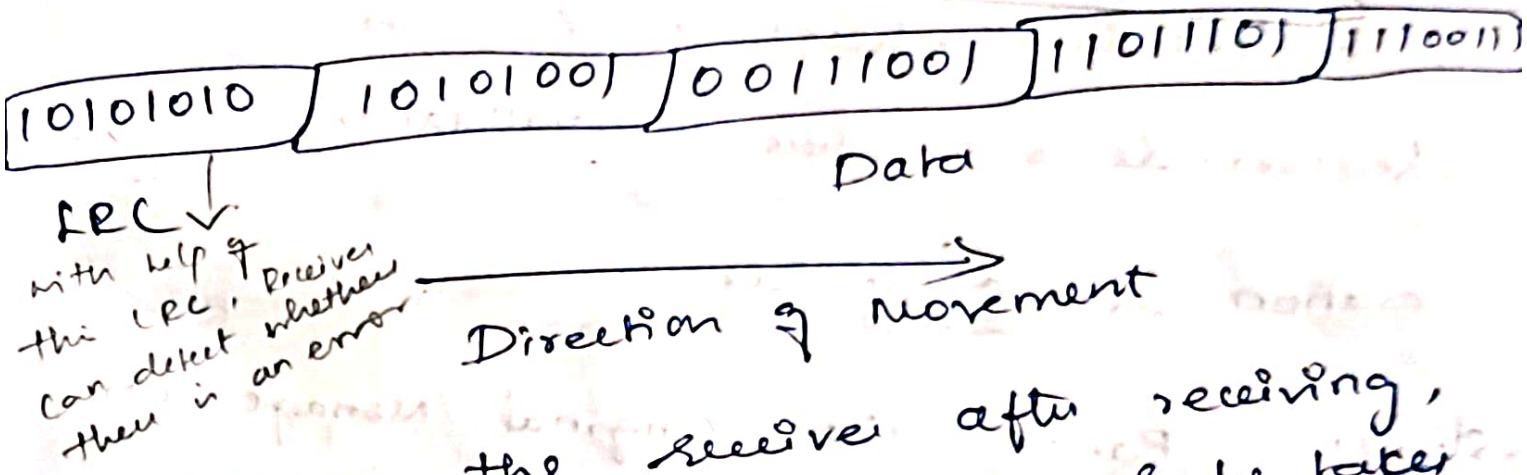
0 0 => 0+0=0

0 1 => 0+1=1

1 0 => 1+0=1

1 1 => 1+1=0.

→ The data that is transmitted to the receiver is



→ Now, the receiver after receiving, he places this LRC aside & he takes all these 4 data blocks, he computes LRC for these data blocks & the receiver just compares the LRC, if both are matching, the receiver understands that there no errors in the transmission, if LRC is mismatching, then he understands it is prone to errors.

Checksum :-

→ checksum = check + sum.

→ Senderside = checksum creation.

→ Receiverside = Checksum validation.

Operation at senderside :-

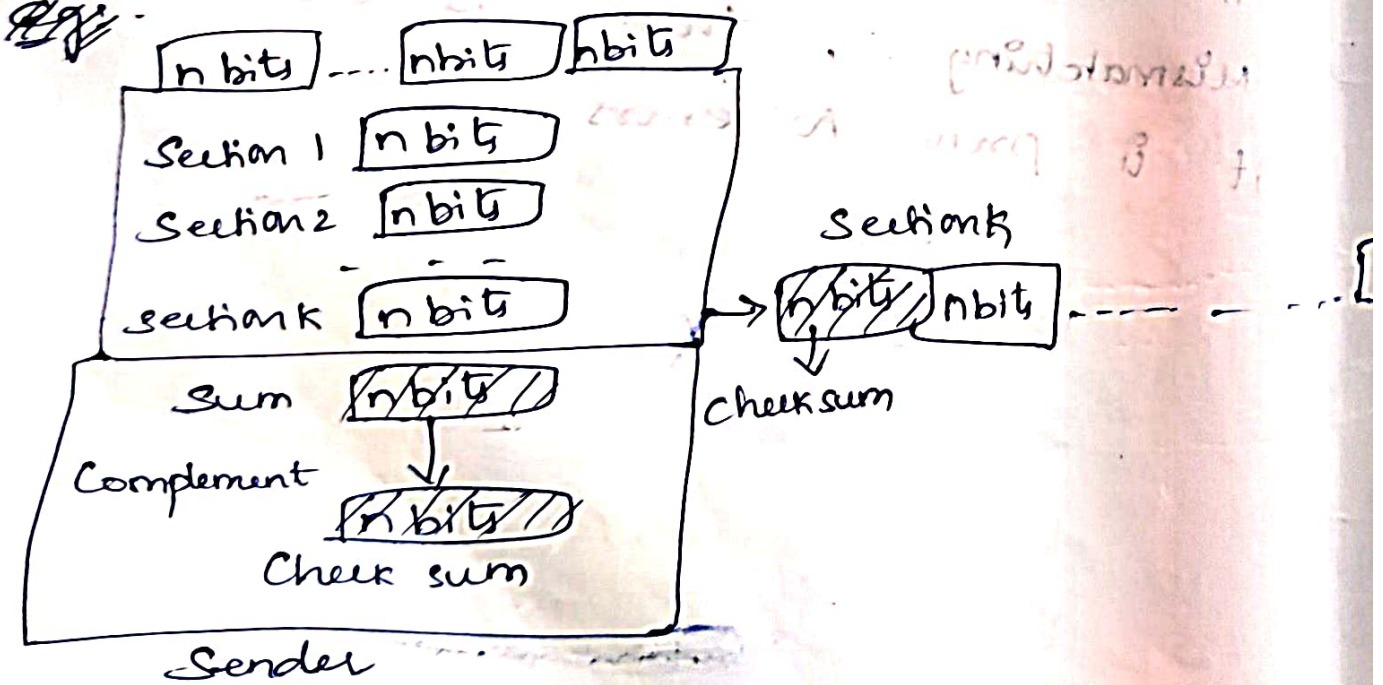
Step 1 :- Break the original message into 'k' number of blocks with 'n' bits in each block.

Step 2 :- Sum all the 'k' data blocks.

Step 3 :- Add the carry to the sum, if any.

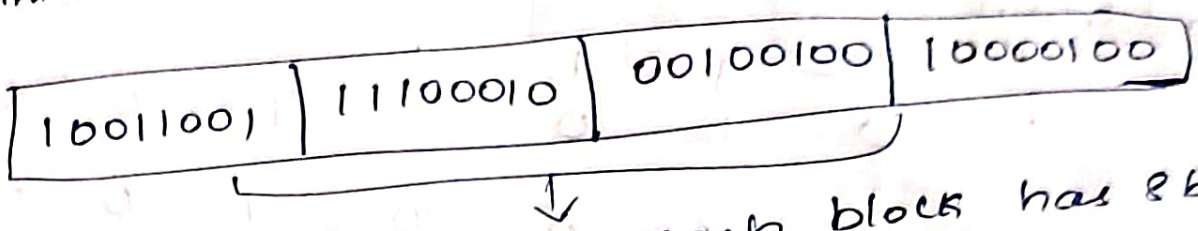
Step 4 :- DO 1's complement to the sum = Checksum.

Eg :-



3q):- Consider the data unit to be transmitted is: 10011001 11100010 00100100 10000100

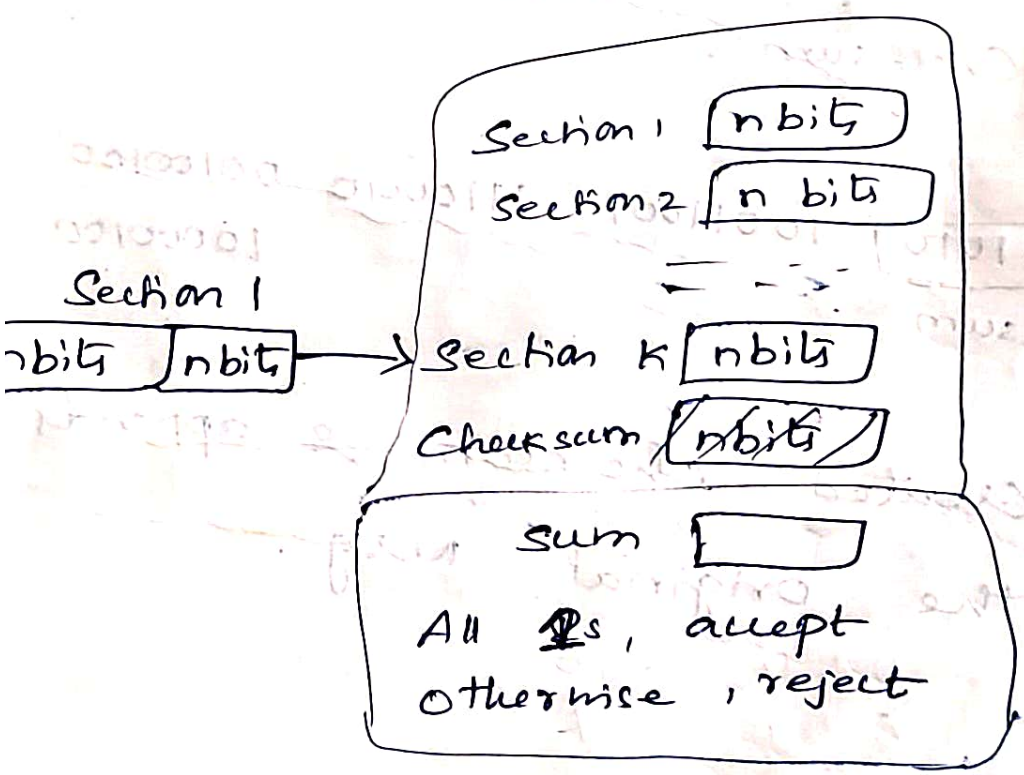
P1:-



Four blocks each block has 8 bits

P2:- Sender places all this blocks, to perform addition.

Q:- 10110011 | 10101011 | 01011010 | 11010101



→ The Sender now performs binary addition.

```
  1   1   1   1   1   0   0 → block 1
  1   0   0   0   0   1   0   0 → 2
  0   0   1   0   0   1   0   0 → 3
  1   1   1   0   0   0   1   0 → 4
  1   0   0   1   1   0   0   1 → 4
 ---------------------------------
 1 0 | 0   0   1   0   0   0   1   1
 ---------------------------------
 carry
```

Step 3 Add the carry to the sum

```
      0   0   1   0   0   0   1   1
              1   0
 ---------------------------------
      0   0   1   0   0   1   0   1
 ---------------------------------
perform 1's Complement
      1   1   0   1   1   0   1   0
 ---------------------------------
```

↓ Checksum.

Sender [11011010] 10011001 11100010 00100100 10000100

Checksum

→ Sender created checksum & appends that to the original msg.

0 + 0 => 0
0 + 1 => 1
1 + 0 => 1
1 + 1 => 0 with 1 carry

1+1+1 = 1 with 1 carry

Operation at Receiver side:-

→ Collect all the data blocks including the Checksum.

→ Sum all the data blocks & Checksum.

→ If the result is all 1's, Accept,
else Reject.

Receiver:-

```
  1  1  1  1  1  1  1  1  1
  1  0  0  0  0  1  0  0
  0  0  1  0  0  1  0  0
  1  1  1  0  0  0  1  0
  1  0  0  1  1  0  0  1
  1  1  0  1  1  0  1  0
  1  0  1  1  1  1  1  0  1
  1  0  1  1  1  1  1  1  0
```

check sum ←

Carry

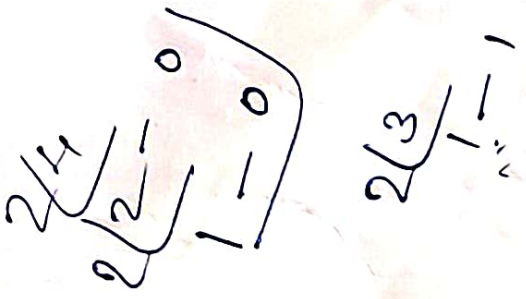
Result all 1's ←

→ Receiver accept it, because it all 1's, there is no transmission error.

---

0 => 0
1 => 1
0 => 1
1 => 0 with carry! [10]
1 => 1 with carry 1
1+1 => 100

2 - 10
3 - 11
4 - 100
5 - 101
6 - 110
7 - 111



2 | 4 - 0
2 | 2 - 0

CRC:-

Find the CRC for the data block 100100 with the divisor 1101?

CRC Generation at Sender side:-

1>. Find the length of the divisor 'L'.

2>. Append L-1 bits to the original message.

3>. Perform binary division operation.

4>. Remainder of the division = CRC.

Note:-

The CRC must be of L-1 bits.

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

XOR.

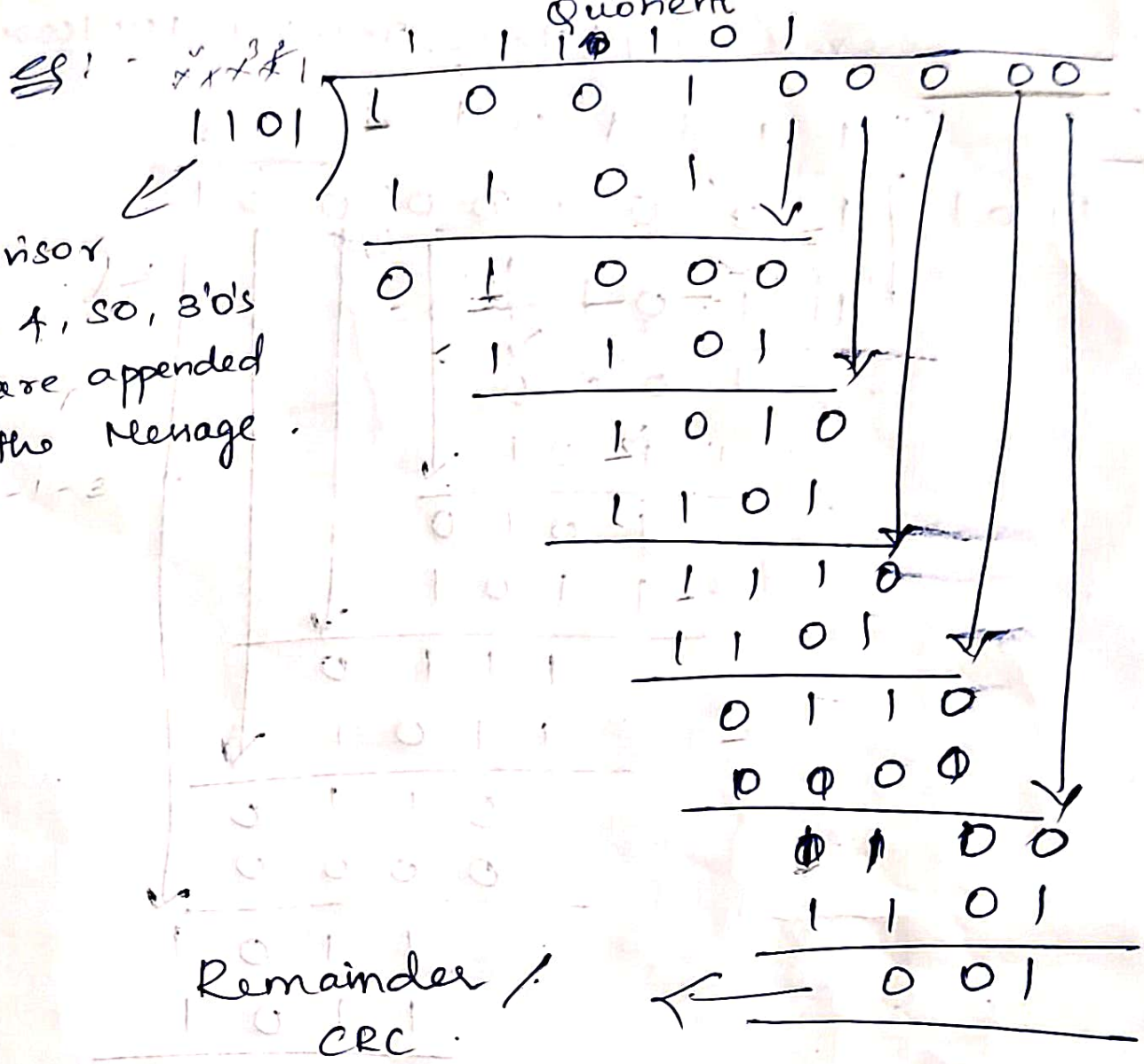
0+0 = 0

0+1 = 1

1+0 = 1

1+1 = 0

Eg:- x^3+x^2+1



Divisor,
L = 4, so, 3 0's are appended to the Message.

Remainder / CRC.

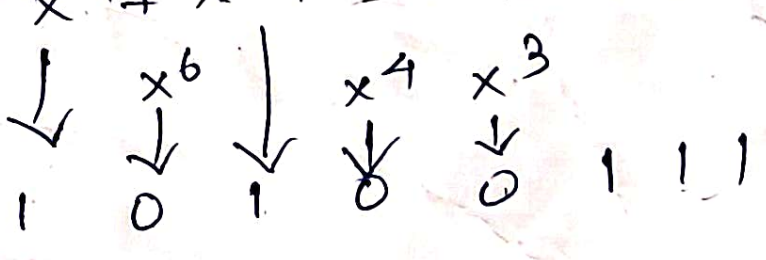
CRC => 001

Data transmitted 100100001

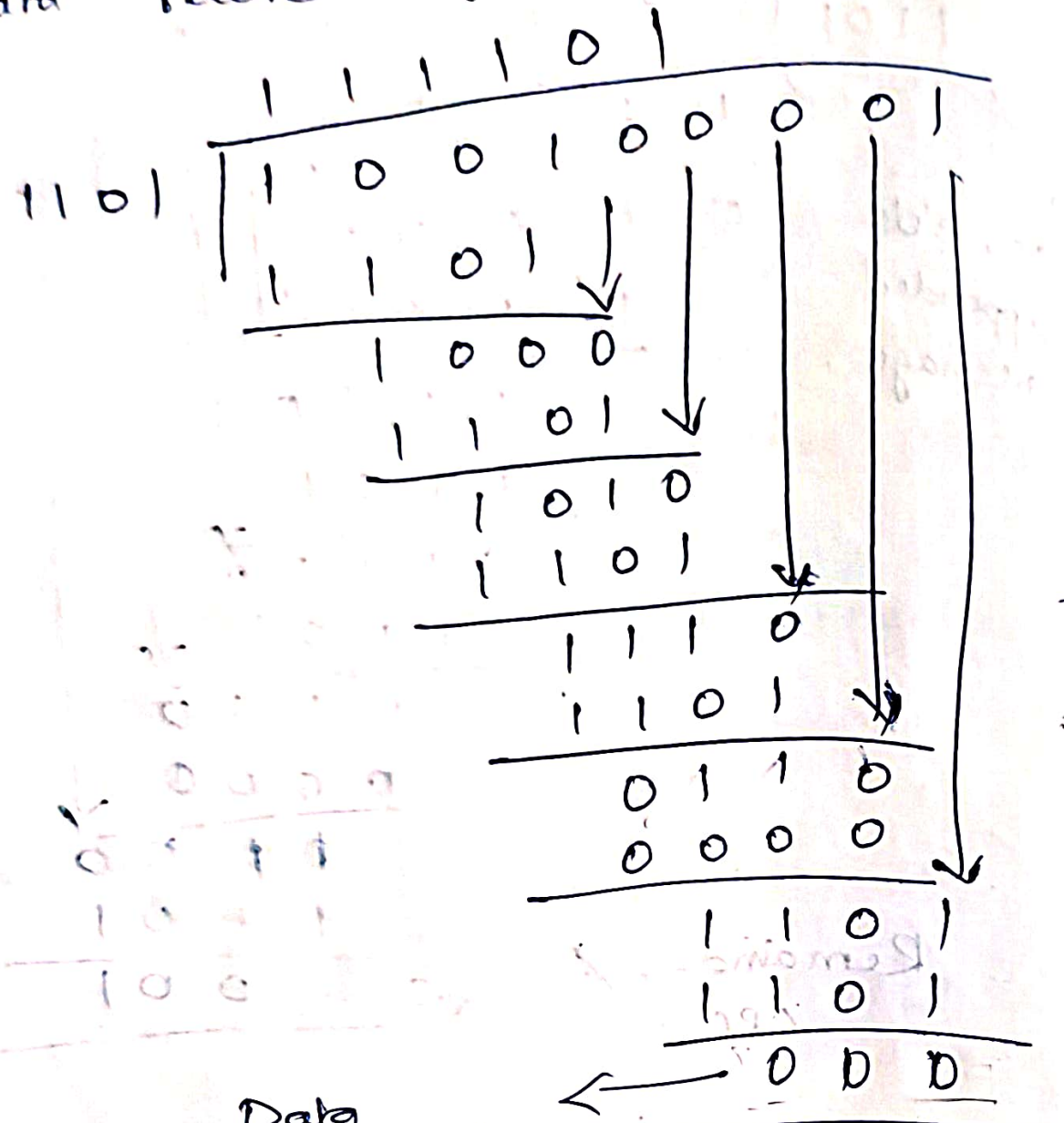
Data CRC

H/W. Find the CRC for 11100101 with the divisor x^3+x^2+1.

Hint:-



Data received by the



Data Accepted.

→ If receiver, receives all 0's then he understands there is no transmission error.

Eg. find CRC. 11010111111 for ~~$x^4+x^3+x^2+x+1$~~ x^4+x+1 CRC → 0010.

Eg find checksum for the given frames. Receiver → all 0's

11001100, 10101010, 11110000, 11000011.