Page No. 01
Date

NAME: PRERNA SUNIL JADHAV
SAP ID: 60004220127

CLASS: S.Y. B Tech (COMPUTER ENGG.)
COURSE: COMPUTER NETWORK (DJ19CEL405)

DATE OF PERFORMANCE:
DATE OF SUBMISSION:

EXPERIMENT 4

AIM: Write a program to implement Error Detection and correction mechanism
i) Hamming code ii) CRC

THEORY:

ERROR DETECTION

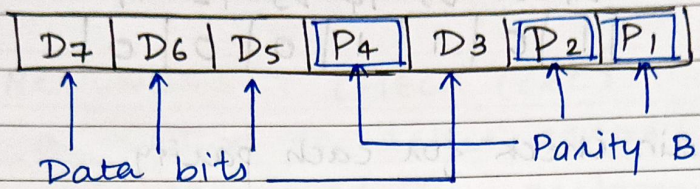
- Error is a condition when the receiver's information doesnt match with senders info.
- whenever a message is transmitted, it may get scrambled by noise or data may get corrupted.
- To avoid this, error detecting codes are added to help us detect if any error occured
- Basic approach used is the use of the redundancy bits, where additional bits are added to detect errors. Some popular ones are 1. CRC, 2. Checksum, 3. simple Parity check etc.

ERROR CORRECTION

- Error connection codes are used to detect and connect the errors when data is transmitted
- For connecting the error, one has to know the exact position of the error. To achieve this, we have to add some additional bits
- Eg: Hamming code

HAMMING CODE

- This technique is capable of detecting and connecting errors that occur in the data
- In this technique, parity bits are used the number of redundant bits can be calculated using: $2^r \geq m+r+1$ where, r is the redundant bit, m = data bit
- Suppose the no. of data bits is 7 (m=7), then r = 4 i.e., Parity bits = 4, since $2^4 \geq 7+4+1$
- There are 2 types of parity bits
  1. Even parity bits: In this case, if the count of 1's is odd the parity value is set to 1, making the total count of 1's even
  2. Odd parity bit: In this case, if the count of 1's is even, the parity value is set to 1, making odd no. of 1's.
- For Eg; if we have length of data = 7
  1) No. of databits = 7
  2) Redundant bits = 4
  3) Position of Redundant bits = 2^0, 2^1, 2^2, 2^3 = 1, 2, 4, 8



For Eg: Data given: 1110 , even parity

1	1	1	P4	0	P2	P1

To find P_1 :

P_1	D_3	D_5	D_7
0	0	1	1

← Because D_5, D_7 has 1 so we can put P_1 0

To find P_2:

P_2	D_3	D_6	D_7
0	0	1	1

← same as above reasoning

To find P_4:

P_4	D_5	D_6	D_7
1	1	1	1

← To make even no. of 1's
$P_4 \Rightarrow 1$

Now, our data to be transmitted is -

1	1	1	1	0	0	0

Suppose that D_6 is flipped, now the data to be transmitted will be -

D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	0	1	1	0	0	0

Now again check for each parity,

For P_1,

P_1	D_3	D_5	D_7
0	0	1	1

→ All good, even no. of 1's

For P_2,

P_2	D_3	D_6	D_7
0	0	0	1

$P_2 = 1$ ← If any error put parity 1

→ Here there should be even 1's but there are odd 1's present

For P_4,

P_4	D_5	D_6	D_7
1	1	0	1

$P_4 = 1$ → same error as above

No change for P_1, but $P_2 \to 1$ and $P_4 \to 0$

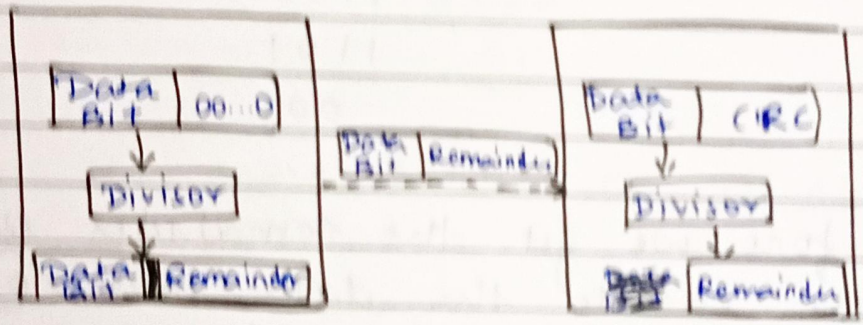
$\therefore (P_4, P_2, P_1) = (1, 1, 0)$ i.e., 6th bits has error and we need to correct it

$\therefore$ The corrected data is

D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	1	1	1	0	0	0

CYCLIC REDUNDANCY CHECK (CRC)

- It is a method to detect errors in the data.
- This is performed by performing a binary solution on the transmitted data at the sender's side and verifying the same at the received side.
- For eg:
  Given: $x^2 + x + 1$ (polynomial eqn)
  Convert it into binary format
  [1 1 1]
- Working

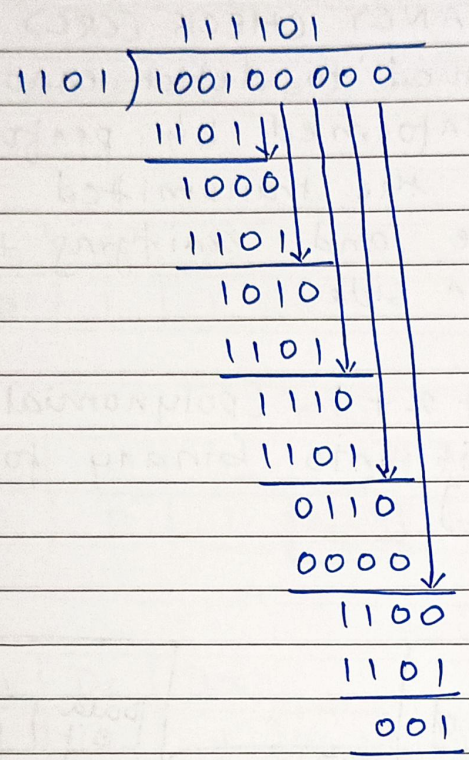


Sender side Receiver side

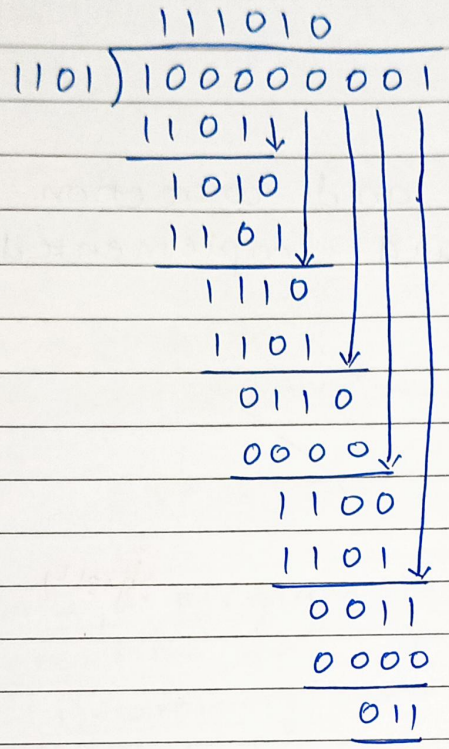
- If data to be sent = 100100 / key:- 1101 then we would append (n-1) zeros to the data where n is the length of key
  ⇒ 100100 000

- Now we would perform binary division operation of 100100000 using 1101 as divisor.

we get,



- Therefore we get the remainder 001 and hence encode the data 100100001 and send to the receiver
- Assuming there was some error in the data transmission and the received data was 100000001.
- Now at receivers end we would again perform binary division. to check for error ~~and then correct it if any.~~



- Since the remainder is not all zero, the error is detected.

WHY CRC IS ADDED AT TRAILER OF THE FRAME?

- Having CRC in the trailer helps to compute the CRC while transmitting the packet and then splat the final CRC at the end
- If we put CRC in the header, it will handle each byte twice, once for checksum and once for transmitting
- Placing the CRC at the end of a frame reduces packet latency and reduces h/w buffering requirement.

CONCLUSION

Error detection and correction mechanism are studied and implemented.