Exp. 6
28/8/25

Error Correction of Data link layer

Aim :- Write a program to implement error detection and correction using HAMMING code concept. Make a test run to input data stream and Variety error correction feature

Error Correction at Data link layer :-

Hamming code is a set of error correction code that can be used to detect and correct that error that can occur when the data is transmitted of the sender to the receiver. It is a technique developed by R.W. Hamming for error correction

Create sender program with below features:

1) Input to sender file should be a text of any length. Program should convert the text to binary.

2) Apply hamming code concept on the binary data and add redundant bits to it.

3) save this output in a file called channel.

Create a receiver program with below features:

1) Receiver program should read input from channel file.

2) Apply hamming code on binary data to check for error.

3) If there is an error, display the position of the error.

4) Else remote redudant bits and convert the binary data to ascii and display the o/p.

Code:

Sender.py

```
def char_to_binary (ch):
    """Convert a character to 8-bit binary
                                      string"""
    return format (ord(ch), '08b')
def hamming_encode (data 4) :
    d1, d2, d3, d4 = [int [bit] for bit in data4]
    P1 = d1 ^ d2 ^ d4
    P2 = d1 ^ d3 ^ d4
    P4 = d2 ^ d3 ^ d4
    return f" {P1} {P2} {d1} {P4} {d2} {d3} {d4}
text = input ("Enter text :")
with open ("channel.txt", "w") as
    for ch in text :
            bin_ch = char_to_binary
            for i in range (0,8,0)
              code = hamming_encode (bin_ch[i:i
                      f.write(code)
            print ("data within the channel
                         with hamming code")
```

receiver.py

```
   def hamming_decode (code_7):
        b = [0] + [int (bit) for bit in code_7]
        P1 = b[1] ^ b[3] ^ b[5] ^ b[7]
        P2 = b[2] ^ b[3] ^ b[6] ^ b[7]
        P4 = b[4] ^ b[5] ^ b[6] ^ b[7]
        error_pos = p1 * 1 + p2*2 + p4 *4
   if error_pos != 0
        print (f"error detected at position
                         {error_pos}..'}
   b[error_pos] ^= 1
   d1, d2, d3, d4 = b[3] b[5], b[6], b[7]
```

```
return f"{d1}{d2}{d3}{d4}"

binary_results = ""
with open("channel.txt","r") as f:
    code = f.read()
for i in range(0, len(code, 5), 7):
    binary_result += hamming_decode
                     (code[i:i+7])
text = ""
for i in range (0, len(binary_result
    byte = binary_result[i:i+8]
    text += chr(int(byte,2))
    print("Receive text after error
                   correction", text).
```

Output:

```
Enter 4-bit data:1011
    Sender side :- 0010011
Enter bit position to introduce
                              error:0
    Receiver side :-0010011
    No error detected
Original data bit extracted :1011
```

Result:

Hence the code is successfully executed and Completed.