

EXPERIMENT : 6

4/8/25

## ERROR CORRECTION AT 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 verify error correction feature.

Error Correction at Data Link Layer:

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

Create sender program with the below features.

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

2) Apply Hamming code concept on the binary data and 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 the input from channel file.

2. Apply hamming code on the binary data to check errors.

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

A. Else, remove the redundant bits and convert the binary data to ASCII and display the output.

Student Observation: At output a string

Code:

After 300 minutes of the program is run

def calculate-parity(data):

C = [0, 0, 0, data[0], 0, data[1], data[2], data[3]]

C[1] = C[3] ^ C[5] ^ C[7]

C[2] = C[3] ^ C[6] ^ C[7]

C[4] = C[5] ^ C[6] ^ C[7]

return C[1]

def detect-and-correct(r):

is redundant = 0 if 10. minor set at reduce set

S1, S2, S4 = r[1] + r[3] ^ r[5] ^ r[7], r[2] ^ r[3] ^ r[5]

r[6] ^ r[7], r[4] ^ r[5] ^ r[6] ^ r[7]

err-pos = S4 \* 4 + S2 + r + S1

if err-pos:

Print ('Error at bit', err-pos)

if r[err-pos] == 0: minor flag 0

Print ('Corrected =', r[1:err-pos] + 1 + r[err-pos:9])

else:

Print ("No error detected")

return r[1:]

def correct-data(c):

return [c[2], c[4], c[5], c[6]]

def main():

data = input ("Enter 4-bit data")

if len(data) != 4 or any ('0' not in data or '1' not in data):

```

data = [int(b) for b in data]
if len(data) != 4:
    enc = calcParity(data)
    print("Encoded", enc)
if input("Introduced error? (y/n): ").lower() == 'y':
    p = int(input("Enter position 1-4:"))
    if 1 <= p <= 4:
        enc[p-1] = 1
        print("Received", enc)
        corr = detectCorrect(enc)
        print("Original data", extractData(corr))

```

Output:

Enter 4-bit data: 101,

Encoded: [0, 1, 1, 0, 0, 1, 1]

Introduce error? (y/n): y

Enter position 1-7: 5

Received: [0, 1, 0, 0, 0, 1, 1]

Error at bit 3

corrected [0, 1, 1, 0, 0, 1, 1]

Original data [1, 0, 1, 1] as sent: 63

Original received after error: 65

After forward error correction: 63

After backward error correction: 65

After error correction: 63

Result:

Hence the code for Hamming problem was successful executed.

Done  
Date: 20/10/2023