Exp No: 6

Date: 21.08.24

Hamming Code.

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.

Student observation:

Write the code here:

Sender.py (filename)

```
import os
def text_to_binary(text):
    return ''.join(format(ord(char), '08b')
                   for char in text)
def calculate_redundant_bits(m):
    r = 0
    while (2**r) < (m + r + 1):
        r += 1
    return r
```

```
def position_redundant_bits(data, r):
    j = 0
    k = 1
    m = len(data)
    res = ''
    for i in range(1, m+r+1):
        if i == 2**j:
            res += '0'
            j += 1
        else:
            res += data[-k]
            k += 1
    return res[::-1]
def calculate_parity_bits(arr, r):
    n = len(arr)
    for i in range(r):
        val = 0
        for j in range(1, n+1):
            if j & (2**i) == (2**i):
                val = val ^ int(arr[-j])
        arr = arr[:n-(2**i)] + str(val) +
              arr[n-(2**i)+1:]
    return arr
def apply_hamming_code(data):
    m = len(data)
    r = calculate_redundant_bits(m)
    arranged_data = position_redundant_bits
    hamming_code = calculate_parity_bits(
                   (data, r)
                   arranged_data, r)
    return hamming_code
```

```
def save_to_channel(hamming_code):
    with open('channel', 'w') as file:
        file.write(hamming_code)
if __name__ == "__main__":
    text = input("Enter the text: ")
    binary_data = text_to_binary(text)
    hamming_code = apply_hamming_code(binary_data)
    save_to_channel(hamming_code)
```

```
# receiver.py
def read_from_channel():
    with open('channel', 'r') as file:
        return file.read()
def calculate_redundant_bits
    r=0
    while (2**r) < (m+r+1):
        r+=1
    return r
def detect_error(arr, nr):
    n = len(arr)
    res = 0
    for i in range(nr):
        val = 0
        for j in range(1, n+1):
            if j & (2**i) == (2**i):
                val = val ^ int(arr[-j])
        res = res + val * (10**i)
    return int(str(res), 2)
```

```
def correct_error(arr, pos):
    if pos >=1:
        arr = arr[:len(arr) - pos] + str(1-
                int(arr[len(arr) - pos]))
    return arr

def remove_redundant_bits(arr, nr):
    n = len(arr)
    j = 0
    res = ''
    for i in range(1, n+1):
        if i != 2**j:
            res += arr[-i];
        else:
            j += 1
    return res[::-1]
def binary_to_text(binary_data):
    text = ''
    for i in range(0, len(binary_data), 8):
        byte = binary_data[i:i+8]
        text += chr(int(byte, 2))
    return text.
```

O/P:

for compiling python sender.py

Enter the text: data.

Data has been encoded and saved to 'channel file.

⇒ python receiverpy

Error detected at position: 8

Error is corrected!

14/9/24

Result:

The program was successfully executed and the O/P is verified.