

Practical-6

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 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 the input from Channel file.
2. Apply hamming code on the binary data to check for errors.
3. If there is an error, display the position of the error.
4. Else remove the redundant bits and convert the binary data to ascii and display the output.

Student observation:-

Write the code here:

Sender:

```
def text_to_binary(text):  
    return ''.join(format(ord(c), '08b') for c in text)  
  
def calculate_parity_bits_length(data_length):  
    r = 0  
    while (2**r) < (data_length + r + 1):  
        r += 1  
    return r  
  
def insert_parity_bits(data):  
    n = len(data)
```

Practical-6

```
r = calculate_parity_bits_length(n)
result = []
j = 0
k = 0
total_len = n + r
for i in range(1, total_len + 1):
    if i == 2**k:
        result.append('0') # placeholder for parity bit
        k += 1
    else:
        result.append(data[j])
        j += 1
# Calculate parity bits
for i in range(r):
    pos = 2**i
    val = 0
    for j in range(1, total_len + 1):
        if j & pos == pos:
            val ^= int(result[j-1])
    result[pos-1] = str(val)
return ''.join(result)

def save_to_channel(data, filename="channel.txt"):
    with open(filename, "w") as f:
        f.write(data)

text = input("Enter text to send: ")
binary_data = text_to_binary(text)
hamming_code = insert_parity_bits(binary_data)
save_to_channel(hamming_code)
```

CS19541-COMPUTER NETWORKS-LAB MANUAL

Practical-6

```
print("Data with Hamming code saved to channel.txt")
```

Receiver:

```
def read_from_channel(filename="channel.txt"):
```

```
    with open(filename, "r") as f:
```

```
        return f.read().strip()
```

```
def detect_and_correct_error(data):
```

```
    n = len(data)
```

```
    r = calculate_parity_bits_length(n - calculate_parity_bits_length(n))
```

```
    error_pos = 0
```

```
    for i in range(r):
```

```
        pos = 2**i
```

```
        val = 0
```

```
        for j in range(1, n + 1):
```

```
            if j & pos == pos:
```

```
                val ^= int(data[j-1])
```

```
        if val != 0:
```

```
            error_pos += pos
```

```
    corrected_data = list(data)
```

```
    if error_pos != 0:
```

```
        print(f"Error detected at position: {error_pos}")
```

```
        corrected_data[error_pos-1] = '1' if data[error_pos-1] == '0' else '0'
```

```
        print("Error corrected.")
```

```
    else:
```

```
        print("No error detected.")
```

```
    return "".join(corrected_data)
```

```
def remove_parity_bits(data):
```

```
    result = []
```

Practical-6

```
k = 0
n = len(data)
for i in range(1, n+1):
    if i != 2**k:
        result.append(data[i-1])
    else:
        k += 1
return ".join(result)

def binary_to_text(binary):
    text = ""
    for i in range(0, len(binary), 8):
        byte = binary[i:i+8]
        text += chr(int(byte, 2))
    return text

data_from_channel = read_from_channel()
corrected_data = detect_and_correct_error(data_from_channel)
original_data = remove_parity_bits(corrected_data)
output_text = binary_to_text(original_data)
print("Original transmitted text is:")
print(output_text)
```

Input:-

Enter text to send: hello

CS19541-COMPUTER NETWORKS-LAB MANUAL

Practical-6

Output:

No error detected.
Original transmitted text is:
Hello

Error detected at position: 5
Error corrected.
Original transmitted text is:
hello