Batch: B-1 **Roll Number: 16010422234** **Name: Chandana Ramesh Galgali**

Experiment Number: 6 - Implementation of Error Detection and Correction

---

Aim of the Experiment: To interpret the concept of redundancy in data message for error detection and Correction using Hamming Code

---

Program/ Steps:

Write a program for following:
1. The program should have a transmitter and receiver function.
2. Accept input in the form of bits (7 bits) (dataword).
3. Generate the codeword at the transmitter using the algorithm.
4. At the receiver, accept the input as the code word (11 bits). Check if any error has occurred and determine the position of the error. Display the position.
5. Correct the error and display the original bits transferred (Dataword).

Output/Result:

Code:

```
def transmitter(databits):
    r1 = databits[6] ^ databits[5] ^ databits[3] ^ databits[2] ^
databits[0]
    r2 = databits[6] ^ databits[4] ^ databits[3] ^ databits[1] ^
databits[0]
    r4 = databits[5] ^ databits[4] ^ databits[3]
    r8 = databits[2] ^ databits[1] ^ databits[0]
    codeword = [databits[0], databits[1], databits[2], r8, databits[3],
databits[4], databits[5], r4, databits[6], r2, r1]
    return codeword
def receiver(codeword):
    r1 = codeword[8] ^ codeword[6] ^ codeword[4] ^ codeword[2] ^
codeword[0]
    r2 = codeword[8] ^ codeword[5] ^ codeword[4] ^ codeword[1] ^
codeword[0]
    r4 = codeword[6] ^ codeword[5] ^ codeword[4]
    r8 = codeword[2] ^ codeword[1] ^ codeword[0]
```

```
    error_position = r8 * 8 + r4 * 4 + r2 * 2 + r1 * 1
    if error_position == 0:
        print("No error occurred.")
    else:
        print("Error occurred at position: ", error_position)
        print("Rectifying the error!")
        codeword[error_position - 1] = 1 - codeword[error_position - 1]
    dataword = [codeword[0], codeword[1], codeword[2], codeword[4],
codeword[5], codeword[6], codeword[8]]
    return dataword
databits = input("Enter the databits to be sent (7 bits): ")
if len(databits) == 7 and all(bit in ['0', '1'] for bit in databits):
    databits = [int(bit) for bit in databits]
    codeword = transmitter(databits)
    print("Transmitted codeword:", ''.join(map(str, codeword)))
    dataword = receiver(codeword)
    print("Original dataword: ", ''.join(map(str, dataword)))
else:
    print("Invalid Input!")
```

Output:

```
PS C:\Users\daxay> & "C:/Python 3.11.0/python.exe" c:/Users/daxay/SY-DCN/EXP-6/Untitled.py
Enter the databits to be sent (7 bits): 1011101
Transmitted codeword: 10101100100
No error occurred.
Original dataword:  1011101
PS C:\Users\daxay> & "C:/Python 3.11.0/python.exe" c:/Users/daxay/SY-DCN/EXP-6/Untitled.py
Enter the databits to be sent (7 bits): 0000000
Transmitted codeword: 00000000000
No error occurred.
Original dataword:  0000000
PS C:\Users\daxay> & "C:/Python 3.11.0/python.exe" c:/Users/daxay/SY-DCN/EXP-6/Untitled.py
Enter the databits to be sent (7 bits): 1100001
Transmitted codeword: 11000000110
Error occurred at position:  2
Rectifying the error!
Original dataword:  1000001
```

Post Lab Question-Answers:

1) What are the different methods used for error detection?

Ans. There are several methods used for error detection in various fields. Here are some commonly used methods:

1. Parity Check: This method involves adding an extra bit to a group of bits to make the total number of 1s either even (even parity) or odd (odd parity). The receiver can then check if the received data has the correct parity.
2. Checksum: Checksum is a simple error detection method commonly used in network protocols. It involves adding up all the data bytes and storing the result as a checksum value. The receiver can then calculate the checksum of the received data and compare it with the transmitted checksum to detect errors.
3. Cyclic Redundancy Check (CRC): CRC is a widely used error detection technique in data communication. It involves dividing the data by a predetermined divisor using binary division. The remainder obtained is appended to the data and transmitted. The receiver performs the same division and checks if the remainder is zero to detect errors.
4. Hamming Code: Hamming codes are error-correcting codes that can detect and correct single-bit errors in data. They add extra parity bits to the data to create a code word with specific properties. The receiver can then use these parity bits to detect and correct errors.
5. Forward Error Correction (FEC): FEC is an error detection and correction technique commonly used in digital communication systems. It involves adding redundant bits to the transmitted data, which allows the receiver to detect and correct errors without the need for retransmission.

These are just a few examples of error detection methods. The choice of method depends on the specific application and the level of error detection and correction required.

2) If the data unit is 111111 and the divisor is 1010, what is the dividend at the Transmitter?

Ans. To calculate the dividend at the transmitter, we need to append zeros to the data unit until its length is equal to or greater than the length of the divisor. In this case, the data unit is "111111" and the divisor is "1010". To make the lengths equal, we need to append two zeros to the data unit. Therefore, the dividend at the transmitter would be "11111100".

3) Which layer of the OSI model usually does the function of error detection?

Ans. The function of error detection is typically performed at the Data Link Layer, which is the second layer of the OSI (Open Systems Interconnection) model. The Data Link Layer is responsible for the reliable transmission of data over a physical link between network nodes. It ensures error-free communication by detecting and correcting errors that may occur during transmission. This layer uses various error detection techniques such as checksums and cyclic redundancy checks (CRC) to verify the integrity of the data.

4) What is Hamming distance ? What is the minimum Hamming distance?

Ans. Hamming distance is a measure of the difference between two strings of equal length. It is defined as the number of positions at which the corresponding elements in the two strings are different. In other words, it calculates the minimum number of substitutions required to change one string into the other.

For example, consider two strings: "101010" and "111000". The Hamming distance between these two strings is 3 because there are three positions where the corresponding elements differ.

In the context of error detection and correction codes, the minimum Hamming distance refers to the smallest Hamming distance between any two valid codewords in a code. It is a crucial parameter that determines the error detection and correction capabilities of a code.

For instance, if a code has a minimum Hamming distance of 3, it means that the code can detect up to two errors and correct a single error. If more than two errors occur, the code may not be able to detect or correct them reliably.

In general, a larger minimum Hamming distance provides better error detection and correction capabilities. However, it also comes with the trade-off of increased redundancy and reduced data transmission efficiency.

Outcomes: Execute their knowledge of computer communication principles, including Error detection and correction, multiplexing, flow control, and error control.

Conclusion (based on the Results and outcomes achieved):

The experiment successfully interpreted the concept of redundancy in data messages for error detection and correction using Hamming Code. It demonstrated the effectiveness of Hamming Code in detecting and correcting errors, thereby improving the reliability of data transmission.

References:

Books/ Journals/ Websites:

1. Behrouz A Forouzan, Data Communication and Networking, Tata Mc Graw hill, India, 4th Edition
2. A. S. Tanenbaum, ”Computer Networks”, 4th edition, Prentice Hall