

**Experiment No. 6**

**Title:** Implementation of Error Detection and Correction

**Batch: A-4 Roll No.:16010422211 Experiment No.:6**

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

**Resources Used: Java/ C / C++ / Python**

**Theory:**

It is often the case that the data retrieved or received is different from the data stored or transmitted, either because the medium is susceptible to damage or because the channel used for transmission is noisy. For example, the data sequence 1101 could be transmitted, but 1001 could be received because the second bit got flipped by the channel. One of the simplest mechanisms to detect whether a single-bit error has occurred is by adding a parity bit to the data sequence. In this practical, we will look at a slightly more complex mechanism to detect and correct single bit errors. The mechanism takes as its input a data sequence of 4 bits and encodes it into a data sequence of 7 bits, which are then transmitted. There is structured redundancy in these 7 bits so that the receiver can detect up to two bit errors and can correct a single bit error.

The (7,4) Hamming code was introduced by Richard Hamming in 1950, who was working then at Bell Telephone Labs. Hamming code uses parity bits concept which is added when you prepare hamming code from the given data stream.

Let ‘k’ be the no of data bits

‘r’ be the no of parity bits and

‘n’ be the number of message bits

Then

2r > = k + r +1

No of message bits : n = k + r No of parity bits : r = n - k

Minimum number of Parity bits :-

Given below is a list of minimum no. of parity bits needed for various ranges of ‘k’ information bit.

Table 1: Parity bits

|  |  |
| --- | --- |
| No of ‘k’ | No of parity bits(r) |
| 2 to 4 | 3 |
| 5 to 11 | 4 |
| 12 to 26 | 5 |
| 27 to 57 | 6 |
| 58 to 120 | 7 |

Consider a data stream of 7 bits 1011101. It requires four redundant bits. These redundant bits are placed at positions1, 2 , 4 and 8 a shown below.

11 10 9 8 7 6 5 4 3 2 1

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| d | d | d | r | d | d | d | r | d | r | r |

These redundant bits are r1, r2, r4and r8 Calculate

r1 = d3 EXOR d5 EXOR d7 EXOR d9 EXOR d11=1 r2 = d3 EXOR d6 EXOR d7 EXOR d10 EXOR d11= 1 r4 = d5 EXOR d6 EXOR d7 = 1

r8 = d9 EXOR d10 EXOR d11 = 1

The message to be transmitted is 10111101111

This is the hamming code which is to be transferred to the destination.

The receiver takes the transmission and recalculates the new values of r . Again the same mechanism is implemented at the receiver end using the same set of data bits plus the relevant parity bits ( r )bit for each set .

If we get the values 0000 then no error, else there is error.

For Correction of the bit in error, find the decimal equivalent of the binary number obtained using parity bits ( r ) and invert the bit in the corresponding position.

**Activity:**

**Write a program for following:**

1. **The program should have 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 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).**

**Program:**

def hamming\_sender(dataword):

if len(dataword) != 7:

print("Enter 7 digits only")

return None

new = '00' + dataword[0] + '0' + dataword[1:4] + '0' + dataword[4:]

r1 = new[2:12:2]

r2 = new[2] + new[5:7] + new[9:11]

r3 = new[4:7]

r4 = new[8:11]

p1 = '1' if r1.count('1') % 2 == 0 else '0'

p2 = '1' if r2.count('1') % 2 == 0 else '0'

p3 = '1' if r3.count('1') % 2 == 0 else '0'

p4 = '1' if r4.count('1') % 2 == 0 else '0'

msg = p1 + p2 + dataword[0] + p3 + dataword[1:4] + p4 + dataword[4:]

return msg

def hamming\_receiver(received\_msg):

if len(received\_msg) != 11:

print("Received message should be 11 bits.")

return None

r1 = received\_msg[2:12:2]

r2 = received\_msg[2] + received\_msg[5:7] + received\_msg[9:11]

r3 = received\_msg[4:7]

r4 = received\_msg[8:11]

p1 = '1' if r1.count('1') % 2 == 0 else '0'

p2 = '1' if r2.count('1') % 2 == 0 else '0'

p3 = '1' if r3.count('1') % 2 == 0 else '0'

p4 = '1' if r4.count('1') % 2 == 0 else '0'

error\_position = int(p4 + p3 + p2 + p1, 2)

if error\_position == 0:

print("No error found.")

else:

print(f"Error found at bit {error\_position}. Correcting...")

received\_msg = list(received\_msg)

received\_msg[error\_position - 1] = '1' if received\_msg[error\_position - 1] == '0' else '0'

received\_msg = ''.join(received\_msg)

dataword = received\_msg[3] + received\_msg[5:8] + received\_msg[9:11]

return dataword

dataword = input("Enter your 7-bit dataword: ")

sender\_msg = hamming\_sender(dataword)

if sender\_msg:

print("Sender: The message is", sender\_msg)

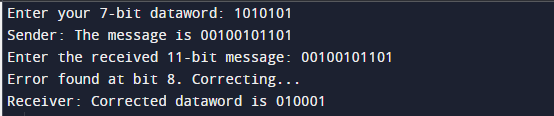
received\_msg = input("Enter the received 11-bit message: ")

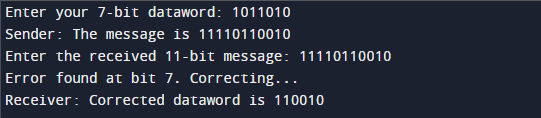
dataword = hamming\_receiver(received\_msg)

if dataword:

print("Receiver: Corrected dataword is", dataword)

**Output:**

****

****

**Questions:**

1. What are the different methods used for error detection Ans.

***Detection:*** ***1. 2d parity check***

***2. Checksum***

***3. Cyclic redundancy Check***

***4. Simple parity***

***Correction:***

***1. Hamming code***

1. If the data unit is 111111 and the divisor is 1010,wht is the dividend at the Transmitter? Ans.

***the dividend at the transmitter is 1111101***.

1. Which layer of the OSI model usually does the function of error detection? Ans

***Data link layer does the function of error detection.***

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

Ans

***Hamming distance measures the difference between two strings.***

***Minimum Hamming distance is the smallest Hamming distance among all pairs of distinct code words in a set, crucial for error detection and correction in coding theory.***

**Outcomes:**CO4: Execute their knowledge of computer communication principles, including error detection and correction, multiplexing, flow control and error control

**Conclusion:**

Conclusion: We understood the implementation of receiver and transmitter of hamming code

**Grade: AA / AB / BB / BC / CC / CD /DD**



**Signature of faculty in-charge with date**

**References:**

**Books/ Journals/ Websites:**

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