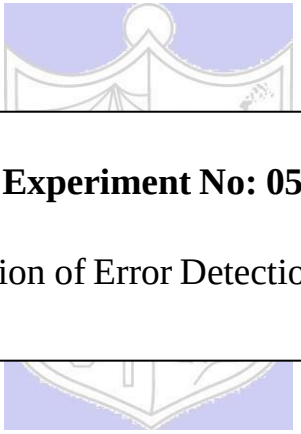

**Experiment No: 05**

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

**Batch:B1** **Roll No.:16010420133** **Experiment No. 05**

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

**Resources Used:** Java/ Turbo 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

$2^r >= k + r + 1$

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.

| 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 $r_1$, $r_2$, $r_4$ and $r_8$.
Consider even parity and Calculate

$r_1$ = d3 EXOR d5 EXOR d7 EXOR d9 EXOR d11=0

$r_2$ = d3 EXOR d6 EXOR d7 EXOR d10 EXOR d11=0

$r_4$ = d5 EXOR d6 EXOR d7=0

$r_8$ = d9 EXOR d10 EXOR d11 =0

The message to be transmitted is 10101100100. 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 one bit in error, find the decimal equivalent of the binary number obtained using parity bits ( r ) and invert the bit in the corresponding position.

---

**Program :**

```
trans = input("Enter transmission data: ")

rlis = [0]*4
if(4<=len(trans)<=11):
    new_str = trans[-len(trans):-4]+"0"+trans[3:6]+"0"+trans[6]+"00"
    rlis[-1] = int(new_str[-11]) ^ int(new_str[-9]) ^ int(new_str[-7]) ^
int(new_str[-5]) ^ int(new_str[-3])
    rlis[-2] = int(new_str[-3]) ^ int(new_str[-6]) ^ int(new_str[-7]) ^
int(new_str[-10]) ^ int(new_str[-11])
    rlis[-3] = int(new_str[-5]) ^ int(new_str[-6]) ^ int(new_str[-7])
    rlis[-4] = int(new_str[-9]) ^ int(new_str[-10]) ^ int(new_str[-11])
```

```
    print(rlis)
    print("The Hamming code is: ",new_str)

# ----------------------------------------------
rec = input("Enter received data: ")
rlis2 = [0]*4
if(8<=len(rec)<=15):
    rlis2[-1] = int(rec[-11]) ^ int(rec[-9]) ^ int(rec[-7]) ^ int(rec[-5]) ^
int(rec[-3]) ^ int(rec[-1])
    rlis2[-2] = int(rec[-3]) ^ int(rec[-6]) ^ int(rec[-7]) ^ int(rec[-10]) ^
int(rec[-11])^int(rec[-2])
    rlis2[-3] = int(rec[-5]) ^ int(rec[-6]) ^ int(rec[-7]) ^ int(rec[-4])
    rlis2[-4] = int(rec[-9]) ^ int(rec[-10]) ^ int(rec[-11]) ^ int(rec[-8])

    err = int("".join( [str(x) for x in rlis2]),2)
    print(err)

    correctBit = str((int(rec[-err])+1)%2)
    resolved = rec[:-err] + correctBit + rec[-err+1:-1]
    print("After solving the error string is: ",resolved)
```

**Output:**

```
Enter transmission data: 1011101
[0, 0, 0, 0]
The Hamming code is:  10101100100
Enter received data: 10101000100
6
After solving the error string is:  1010110010
```

**Questions:**

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

**Ans:** Whenever a message is transmitted, it may get scrambled by noise or data may get corrupted. To avoid this, we use error-detecting codes which are additional data added to a given digital message to help us detect if any error has occurred during transmission of the message. Basic approach used for error detection is the use of redundancy bits, where additional bits are added to facilitate detection of errors. Some popular techniques for error detection are:

1. Simple Parity check
2. Two-dimensional Parity check
3. Checksum
4. Cyclic redundancy check

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

**Ans:** 11111110000

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

**Ans:** Error detection is done both by Data- link and transport layers, but error correction is done only at the transport layer. Many communication channels are at the mercy of channel noise, and thus errors may be found during transmissions from the source to a receiver. Error observation techniques allow detecting such errors, while errors corrections enable reconstruction from the original data in many cases. Error observation is the detection of errors caused by noise or other impairments during transmission from the transmitter to the receiver.

4) **What arithmetic is used to add data items in checksum calculation?**

**Ans:** One's complement arithmetic is used to add data items in checksum calculation. In this arithmetic, when a number needs more than n bits, the extra bits are wrapped and added to the number.

**5) What is Hamming distance? What is minimum Hamming distance?**

**Ans:** The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. In other words, it measures the minimum number of substitutions required to change one string into the other, or the minimum numberof errors that could have transformed one string into the other. In a more general context, the Hamming distance is one of several string metrics for measuring the edit distance between two sequences. It is named after the American mathematician Richard Hamming. The minimum Hamming distance is used to define some essential notions in coding theory, such as error detecting and error correcting codes. In particular, a code C is said to be k error detecting if, and only if, the minimum Hamming distance between any two of its codewords is at least k.

---

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

---

**Conclusion:** The concept of redundancy in data message for error detection and Correction using Hamming Code was understood successfully and program based on Hamming Codewas implemented successfully.

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

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

**References:**

**Books/ Journals/ Websites:**


- B. A. Forouzan and Firouz Mosharraf, ”Computer Networks ”, A Top-Down Approach, Special Indian Edition 2012, Tata McGraw Hill.
- Behrouz A Forouzan, Data Communication and Networking, Tata Mc Graw Hill, India, 4th Edition