HAMMING CODE USING TCP

A MINIOR PROJECT REPORT

Submitted by

M. SAI KISHORE (171FA05305)

R. V. L. KARTHIK (171FA05329)

S. TUSHAR (171FA05362)

in

Electronics and Communication Engineering

Under the Esteemed Guidance of

Dr. B. Seetha Ramanjaneyulu

Professor

VIGNAN'S
Foundation for Science, Technology & Research
(Deemed to be University)
-Estd. u/s 3 of UGC Act 1956

(ACCREDITED BY **NAAC** WITH **'A'** GRADE)
MHRD **NIRF 88** RANK

DEPARTMENT OF
ELECTRONICS & COMMUNICATION ENGINEERING

VFSTR, VADLAMUDI.

GUNTUR-522213, ANDHRAPRADESH.

JANUARY 2021.

CERTIFICATE

This is to certify that the minor project report entitled **"HAMMING CODE USING TCP"** that is being submitted by **M. SAI KISHORE (171FA05305), R.V.L. KARTHIK (171FA05329), S. TUSHAR (171FA05362),** respectively in partial fulfilment for the award of IV year I semester B.Tech degree in Electronics and Communication Engineering to Vignan's Foundation for Science Technology and Research, is a record of work carried out by him/her under the guidance of **Dr. B. Seetha Ramanjaneyulu**, Professor of ECE Department.

Signature of faculty guide	Signature of Head of Department
Dr. B. Seetha Ramanjaneyulu	Mr.T.Pitchaiah,M.E,Ph.D,MIEEE
Professor	Professor, HOD

ABSTRACT

Thinking about a communication channel that doesn't add any kind of noise to the data being transmitted is not a real-world scenario. So, even the transmitted data has errors those errors must be detected and must be corrected by the receiver mostly. But always the receiver may not have the capability in order to correct the errors. So, it will request the sender for re-transmission of data. But always re-transmission of entire data is not an easy task. So, there arises a need of error correction mechanism at receiver. One of such mechanisms is hamming code. Hamming code is a one-bit error correction mechanism. Our main objective is to provide error correction capability to the receiver using hamming code.

CONTENTS

CHAPTER 1
INTRODUCTION

Hamming code is one of the error correction code which will detect and correct the errors when data is transmitted from sender to receiver. Generally, we can detect the single bit error using a parity bit whose value will be based upon no of one's present in the data. Whereas hamming code can detect up to two bits errors because of usage of more no of parity bits which will depend upon the no of bits present in the input data. Hamming code makes use of the concept of parity and parity bits, which are bits that are added to data so that the validity of the data can be checked when it is read or after it has been received in a data transmission. Using more than one parity bit, an error-correction code can not only identify a single bit error in the data unit, but also its location in the data unit. During data transmission the ability of receiver node to correct the errors in the received data is called as Forward Error Correction (FRC). Hamming code is an example of FRC.

SOFTWARE REQUIRMENTS

- Net Beans, any java compiler
- Windows 7 & above
- 64 Bit windows & amp 32 bits
- Processor i3 and above

HARDWARE REQUIRMENTS

- Intel core i3
- Floppy drive
- Ram (minimum 512 mb)
- 10 GB of free hard disk

CHAPTER 2

PROCEDURE

At Sender:

- ❖ First find out the no of bits present in the given message and store it in ‘m’.
- ❖ Then we need to find out the no of parity bits ‘p’ using formula

  2**p≥m+p+1
- ❖ Then we need to find out the positions where these parity bits need to be embedded within the message by substituting the p values in 2**p where p values have not satisfied the condition mentioned above.
- ❖ Then in those positions put the parity bits and in remaining positions place the original message bits in correct order.
- ❖ Then find the values of these parity bits using even parity.
- ❖ Then place these values at their specific positions in the message and send the obtained message.

1. Message: 1011011
2. Number of bits in Message m: 7
3. Find number of parity bits using 2^r >= m+r+1(until it becomes true):
4. r=0, 1, 2, 3, 4, 5, 6, etc.
5. m=7
6. 2^0>=7+0+1, 2^1>=7+1+1, 2^2>=7+2+1, 2^3>=7+3+1, 2^4>=7+4+1
7. 1>=8 (F)   2>=9(F)   4>=10(F)   8>=11(F)   16>=12(T)
8. So at positions (1, 2, 4, 8) we add parity bits (p0, p1, p2, p3) respectively.
9. Find parity bit values :(p0, p1, p2, p3).
10. P0 is at position1 so check 1 bit, skip 1 bit alternatively.
11. So p0 takes values at positions: 3, 5, 7, 9, 11.
12. Values at above positions are: 1, 0,1,0,1.
13. In above values number of 1’s are odd so p0 value is 1.
14. Perform same operations for p1, p2, p3.

15. Place these p0, p1, p2, p3 values in their respective positions.
Then send this message, parity bits position to receiver.

At Receiver:

- ❖ While transferring the message itself sender will intimate the receiver about the no.of parity bits added and also about the positions at where they are added.
- ❖ Then receiver will again find out the values for parity bits in the same procedure as sender does.
- ❖ After finding all the parity bits values receiver will arrange the parity bits as p3, p2, p1, p0 like that.
- ❖ Then receiver will convert that obtained binary number to decimal number and find that in that decimal position the bit has been corrupted.
- ❖ The in that position receiver will correct that bit by placing '0' if '1' is present and vice versa.
- ❖ Get parity bits (1, 2, 4, 8) and encoded message from sender.
- ❖ P0 is at position1 so check 1 bit, skip 1 bit alternatively.
- ❖ So p0 takes values at positions: 1,3,5,7,9,11.
- ❖ Values at above positions are: 1, 1, 0, 1, 0, 1.
- ❖ In above values number of 1's are even so p0 value is 0.
- ❖ Perform same operations for p1, p2, p3.
- ❖ If we get p0, p1, p2, p3 as 0's then message received from sender is correct.
- ❖ Else message received from sender is incorrect.
- ❖ If we get p3 as 1 then
- ❖ (p3,p2,p1,p0) =(1,0,0,0)
- ❖ Find decimal value for 1000
- ❖ Here decimal value of 1000 is 8 means 8th bit in encoded message is flipped.
- ❖ So we have to change the 8th bit value
- ❖ If 8th bit value is 0 place 1 else place

CHAPTER 3

ALGORITHM

At Sender:

- Start the program.
- Import java.net package and other packages.
- Create objects for ServerSocket, Socket, DataInputStream and PrintStream to transfer the data.
- Using PrintStream transfer the data in OutputStream via a port.
- Run an loop to send the data from server until an “end or exit” string is transferred.
- If “end or exit” is encountered, close the socket and exit the program.
- Stop the program.

At Receiver:

- Start the program.
- Import java.net package and other packages.
- Create objects for Socket and DataInputStream to receive the server data.
- Run a loop to receive the data from server and store it in a string using DataInputStream.
- Display the server data and exit when an “end or exit” message is encountered.
- Stop the program.

CHAPTER 4

CODE

SERVER:

```
import java.io.*;

import java.net.*;

import java.util.*;

public class MyServer {

public static void main(String[] args){

try{

ServerSocket ss=new ServerSocket(6666);

Socket a=ss.accept();//establishes connection

DataInputStream dis=new DataInputStream(a.getInputStream());

String  str1=(String)dis.readUTF();

System.out.println(str1);

DataInputStream dis1=new DataInputStream(a.getInputStream());

int d[]=new int[7];

for(int i=0;i<7;i++)

{

String  str=(String)dis.readUTF();

System.out.println(str);

d[i]=Integer.parseInt(str);

}

Scanner sc=new Scanner(System.in);
```

```
int c[]=new int[11];

System.out.println("complete code word is:");

for(int i=0;i<11;i++)

{

String  str2=(String)dis.readUTF();

System.out.print(" "+str2);

c[i]=Integer.parseInt(str2);

}

System.out.println();

System.out.println("Enter the Received codeword");

int r[]=new int[11];

for(int i=0;i<11;i++)

{

r[i]=sc.nextInt();

}

int pr[]=new int[4];

int rd[]=new int[7];

pr[0]=r[0];

pr[1]=r[1];

rd[0]=r[2];

pr[2]=r[3];

rd[1]=r[4];

rd[2]=r[5];
```

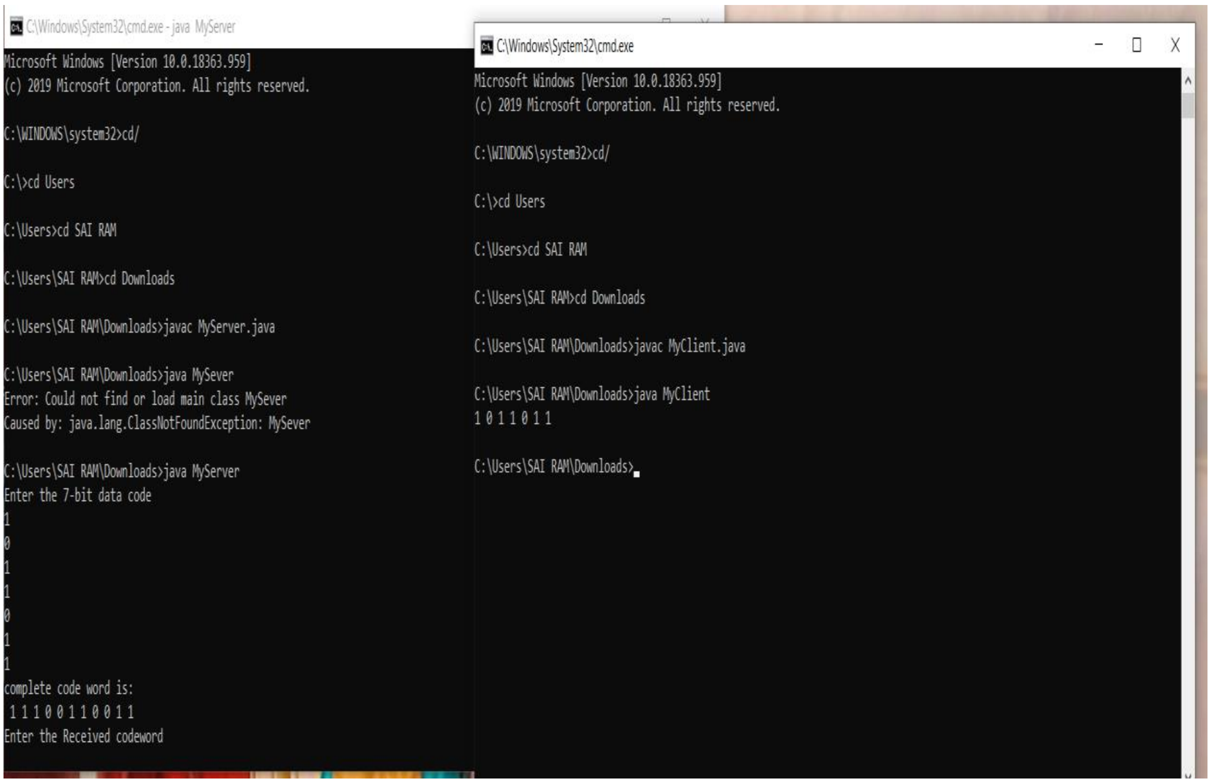
```
rd[3]=r[6];

pr[3]=r[7];

rd[4]=r[8];

rd[5]=r[9];

rd[6]=r[10];

int s[]=new int[4];

s[0]=pr[0]^rd[0]^rd[1]^rd[3]^rd[4]^rd[6];

s[1]=pr[1]^rd[0]^rd[2]^rd[3]^rd[5]^rd[6];

s[2]=pr[2]^rd[1]^rd[2]^rd[3];

s[3]=pr[3]^rd[4]^rd[5]^rd[6];

int dec=(s[0]*1)+(s[1]*2)+(s[2]*4)+(s[3]*8);

if(dec==0)

System.out.println("No error");

else

{

System.out.println("Error is at "+dec);

if(r[dec-1]==0)

r[dec-1]=1;

else

r[dec-1]=0;

System.out.println("Corrected code word is : ");

for(int i=0;i<11;i++)

System.out.print(r[i]+" ");
}ss.close();

}catch(Exception e){System.out.println(e);}

}  }
```

RECIEVER:

```
import java.io.*;

import java.net.*;

import java.util.*;

public class MyClient {

public static void main(String[] args) {

try{

Socket s=new Socket("localhost",6666);

Scanner sc=new Scanner(System.in);

DataOutputStream dout=new DataOutputStream(s.getOutputStream());

dout.writeUTF("Enter the 7-bit data code");

int d[]=new int[7];

DataOutputStream dout1=new DataOutputStream(s.getOutputStream());

for(int i=0;i<7;i++)

{

d[i]=sc.nextInt();

dout1.writeUTF(Integer.toString(d[i]));

}

int p[]=new int[4];

p[0]=d[0]^d[1]^d[3]^d[4]^d[6];
p[1]=d[0]^d[2]^d[3]^d[5]^d[6];

p[2]=d[1]^d[2]^d[3];

p[3]=d[4]^d[5]^d[6];

int c[]=new int[11];

c[0]=p[0];

c[1]=p[1];

c[2]=d[0];

c[3]=p[2];

c[4]=d[1];

c[5]=d[2];

c[6]=d[3];

c[7]=p[3];

c[8]=d[4];

c[9]=d[5];

c[10]=d[6];

DataOutputStream dout2=new DataOutputStream(s.getOutputStream());

for(int i=0; i<11;i++)

{

dout2.writeUTF(Integer.toString(c[i]));

}

dout.close();

s.close();

}catch(Exception e){System.out.println(e);}  } }
```

CHAPTER 5

OUTPUT



```
Microsoft Windows [Version 10.0.18363.959]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd/

C:\>cd Users

C:\Users>cd SAI RAM

C:\Users\SAI RAM>cd Downloads

C:\Users\SAI RAM\Downloads>javac MyServer.java

C:\Users\SAI RAM\Downloads>java MySever
Error: Could not find or load main class MySever
Caused by: java.lang.ClassNotFoundException: MySever

C:\Users\SAI RAM\Downloads>java MyServer
Enter the 7-bit data code
1
0
1
1
0
1
1
complete code word is:
 1 1 1 0 0 1 1 0 0 1 1
Enter the Received codeword
```

```
Microsoft Windows [Version 10.0.18363.959]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd/

C:\>cd Users

C:\Users>cd SAI RAM

C:\Users\SAI RAM>cd Downloads

C:\Users\SAI RAM\Downloads>javac MyClient.java

C:\Users\SAI RAM\Downloads>java MyClient
1 0 1 1 0 1 1

C:\Users\SAI RAM\Downloads>
```

CHAPTER 6

ADVANTAGES

- Most effective for bit error detection and correction.
- Easy and best to use in computer memory.

DISADVANTAGES

- Not applicable for multiple bit errors.
- Hamming code algorithm resolve only single bit errors.

APPLICATIONS

- Computing.
- Data transmission.
- Telecommunications.

CHAPTER 7

CONCLUSION

Hence by using this hamming code using tcp we detect and correct the error bits that can occur when computer data is moved or stored.

REFERENCES

- https://en.wikipedia.org/wiki/ECC_memory
- https://www.quora.com/Where-is-hamming-code-is-used-practically
- https://www.tutorialspoint.com/digital_circuits/digital_circuits_error_detection_correction_codes.htm