**HAMMING CODE IMPLEMENTATION USING SOCKETS:**

<<SOURCE\_CODE>>

>>hammingServer.c

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#include<unistd.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<string.h>

#include<arpa/inet.h>

#define SIZE 1000

#define MAX\_CLIENT 1

int sum=0;

void errorPosition(int );

void ErrorCorrection(char \*str)

{

        int i, j, flag=-1, count=0;

        for(i=0;i<12;i++)

        {

                if(i==0 || i==1 || i==3 || i==7)

                {

                        j=i;

                        while(j<12)

                        {

                                if(flag!=i)

                                {

                                        if(str[j]=='1')

                                        {

                                                count++;

                                        }

                                        j++;

                                        flag++;

                                }

                                else

                                {

                                        j=j+i+1;

                                        flag=-1;

                                }

                        }

                        if((count%2)!=0)

            {

                                errorPosition(1);

                                count=0;

                        }

                        else

                        {

                                errorPosition(0);

                                count=0;

                        }

                }

        }

}

void errorPosition(int value)

{

        static int arr[4]={-1,-1,-1,-1}, count=0;

        int i;

        for(i=0; i<4; i++)

        {

                if(arr[i]==-1)

                {

                        arr[i]=value;

                        break;

                }

        }

        count++;

        int e=3;

        if(count==4)

        {

                for(i=3; i>=0; i--)

                {

                        sum+=(arr[i] \* pow(2,e));

                        e--;

                }

        count=0;

        }

}

int main(int argc, char \*argv[])

{

        if(argc!=3)

        {

                printf("Some arguments are missing.\n");

                exit(1);

        }

        int sockfd, newsockfd, clilen;

        char buffer[SIZE];

        struct sockaddr\_in serv\_addr, cli\_addr;

        sockfd=socket(AF\_INET, SOCK\_STREAM, 0);

        if(sockfd<0)

        {

                printf("Error in creating the socket.\n");

                exit(1);

        }

        bzero((char \*)&serv\_addr, sizeof(serv\_addr));

        serv\_addr.sin\_family=AF\_INET;

        serv\_addr.sin\_addr.s\_addr=inet\_addr(argv[1]);

        serv\_addr.sin\_port=htons(atoi(argv[2]));

        if(bind(sockfd, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr))<0)

        {

                printf("Couldn't bind to the Socket.\n");

                exit(1);

        }

        listen(sockfd, MAX\_CLIENT);

     while(1){

                clilen=sizeof(cli\_addr);

                newsockfd= accept(sockfd, (struct sockaddr \*)&cli\_addr, &clilen);

                if(newsockfd<0) {

                        printf("Error in accepting the Client's request.\n");

                        exit(1);

                }

                memset((char \*)&buffer, 0x0, sizeof(buffer));

                recv(newsockfd, buffer, sizeof(buffer), 0);

                printf("Received data from client is %s\n", buffer);

                ErrorCorrection(buffer);

                printf("Bit Error Position is %d\n", sum);

                if(buffer[sum-1]=='1')

                        buffer[sum-1]='0';

                else

                        buffer[sum-1]='1';

                printf("Data after error correction is %s\n", buffer);

                close(newsockfd);

        }

                close(sockfd);

        return 0;

}

hammingClient.c

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<math.h>

#include<unistd.h>

#include<time.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

void myMessage(char \*new, char \*str)

{

        int i, j=0;

        for(i=0;i<12;i++)

        {

                if(i==0 || i==1 || i==3 || i==7)

                {

                        str[i]='0';

                }

                else

                {

                        str[i]=new[j++];

                }

        }

        str[i]='\0';

}

void myHammingCode(char \*new, char \*str)

{

        int i, j, flag=-1, count=0;

        for(i=0;i<12;i++)

        {

                if(i==0 || i==1 || i==3 || i==7)

                {

                        j=i;

                        while(j<12)

                        {

                                if(flag!=i){

                                        if(new[j]=='1'){

                                                count++;

                                        }

                                        j++;

                                        flag++;

                }

                                else{

                                        j=j+i+1;

                                        flag=-1;

                                }

                        }

                        if((count%2)!=0){

                                str[i]='1';

                                count=0;

                        }

                        else{

                                str[i]='0';

                                count=0;

                        }

                }

                else

                {

                        str[i]=new[i];

                }

        }

        str[i]='\0';

}

void myTransmittingCode(float p,char \*str)

{

        int rno, rpos;

        srand(time(0));

        rno=rand() % 10;

        printf("The generated random no is %d\n", rno);

        p=p\*10;

        if(rno>= 0 && rno<= ((int)p))

        {

                rpos=rand() % 12;

                printf("Bit position to be corrupted is %d.\n", (rpos+1));

                if(str[rpos]=='0')

                {

                        str[rpos]='1';

                }

        else if(str[rpos]=='1')

                {

                        str[rpos]='0';

                }

        }

        else

        {

                printf("NO BIT TO BE CORRUPTED.\n");

        }

}

void sendToServer(char \*data, char \*ip, char \*port)

{

        int sockfd, connectfd;

        struct sockaddr\_in serv\_addr, cli\_addr;

        sockfd=socket(AF\_INET, SOCK\_STREAM, 0);

        if(sockfd<0)

        {

                printf("Error in creating the socket.\n");

                exit(1);

        }

        bzero((char \*)&serv\_addr, sizeof(serv\_addr));

        serv\_addr.sin\_family=AF\_INET;

        serv\_addr.sin\_addr.s\_addr=INADDR\_ANY;

        serv\_addr.sin\_port=htons(atoi(port));

        connectfd=connect(sockfd, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr));

        if(connectfd<0)

        {

                printf("Couldn't connect to the Server.\n");

                exit(1);

        }

        send(sockfd, data, strlen(data)+1, 0);

}

int main(int argc, char \*argv[])

{

        if(argc!=5)

        {

                printf("Some arguments aren't provided.\n");

                exit(1);

        }

        float p=strtof(argv[2],NULL); //OR USE atof()

        char message[100];

        char HammingCode[100];

        printf("User's data is %s\n", argv[1]);

        myMessage(argv[1], message);

        myHammingCode(message, HammingCode);

        printf("Hamming code is %s\n", HammingCode);

        myTransmittingCode(p, HammingCode);

        printf("Data to be sent to the server is %s\n", HammingCode);

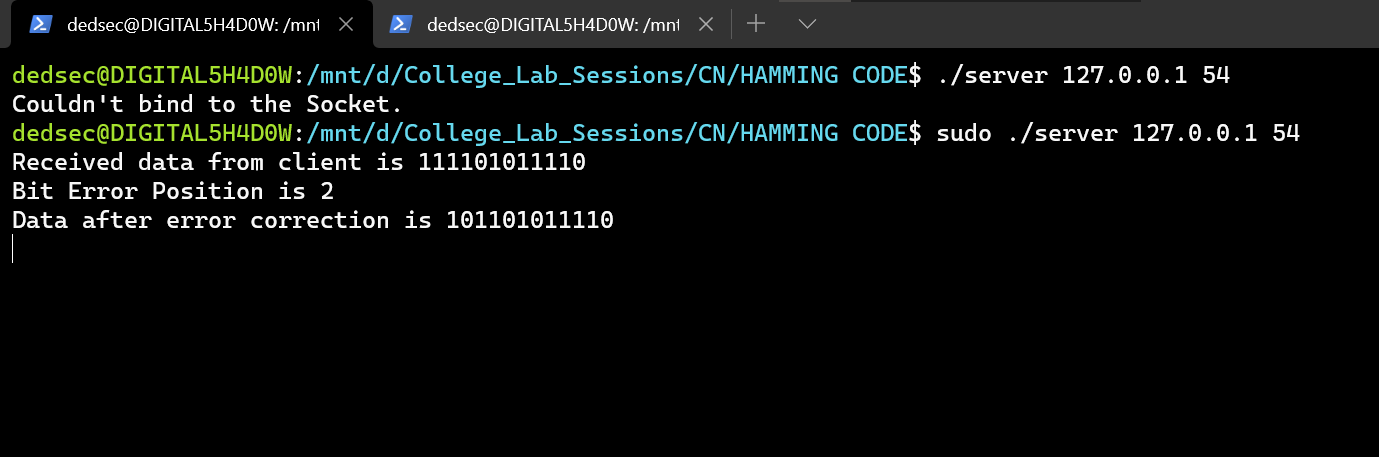
        sendToServer(HammingCode,argv[3],argv[4]);

        return 0;

}

**<<SCREENSHOTS>>**

SERVER SIDE :



CLIENT SIDE :

