Name : Sahil Dattatray Mohite

Div : SY-IT-A   Batch : B1

Roll No. 30   PRN : 12010501

Computer Network – Lab Assignment 4

Unit : Hamming Code

Question:

Implement Hamming Code Error Correction Algorithm using TCP Socket.

Server.c

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<stdbool.h>
#include<math.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>

bool isPowerOfTwo(int n)
{
   if(n==0)
   return false;

   return (ceil(log2(n)) == floor(log2(n)));
}

int main()
{

int socket_server , clientsocketfd, bindstatus;

socket_server = socket(AF_INET , SOCK_STREAM , 0);

struct sockaddr_in serveraddress , clientaddress;
serveraddress.sin_family = AF_INET;
serveraddress.sin_port = htons(9000);
serveraddress.sin_addr.s_addr = INADDR_ANY;

bindstatus = bind( socket_server ,
                   (struct sockaddr *)&serveraddress ,
                    sizeof(serveraddress)
                  );
```

```c
if (bindstatus<0)
{
    printf("Binding Failed\n");
}
else
{
    printf("Binding is successful\n");
}

listen(socket_server , 10);
printf("Send reply to the client\n");

int cliaddlen = sizeof(clientaddress);
clientsocketfd = accept(socket_server ,
                        (struct sockaddr *)&clientaddress,
                        &cliaddlen );


    char Data[100];
    char p1,p2,p4,p8;
    int n,r;
    int i,j,k;
    int count;

    read(clientsocketfd , Data , 100);
    read(clientsocketfd , &n , sizeof(n) );
    read(clientsocketfd , &r , sizeof(r) );
    read(clientsocketfd , p1 , 1);
    read(clientsocketfd , p2 , 1);
    read(clientsocketfd , p4 , 1);
    read(clientsocketfd , p8 , 1);


    printf("\n\tData received from the client:");
    printf("\n");
    for (i=0; i<=n+r ; i++)
    {
        printf("%c\t" , Data[i]);
    }

    char p1_ ='0';
    char p2_ ='0';
    char p4_ ='0';
    char p8_ ='0';

    for (i=1; i<n+r ; i++)
    {
        if(i==1)
        {
            count=0;
            for(j=1;j<=n+r;j=j+2) //1,3,5,7,9,...
            {
                if (Data[j] == '1')
                {
                    count++;
                }
            }

            if (p1=='1')
            {
                count++;
            }

            Data[1] = count%2==0 ? '0' : '1';
            p1_ = Data[1];
        }

        if(i==2)
        {
            count=0;
            for(j=2;j<=n+r;j=j+4)//2,3,6,7,10,11,...
            {
                if (Data[j] == '1')
                {
                    count++;
                }
                if(Data[j+1] == '1')
                {
                    count++;
                }
            }
            if (p2=='1')
            {
                count++;
            }
            Data[2] = count%2==0 ? '0' : '1';
            p2_ = Data[2];
        }

        if(i==4)
        {
            count=0;
            for(j=4;j<=n+r;j=j+8)//4,5,6,7,12,13,14,15,20,21,22,23,...
            {
                if (Data[j] == '1')
                {
                    count++;
                }
                for(k=j+1;k<j+8;k++)
                {

                    if(Data[k] == '1')
                    {
                        count++;
                    }
                }

            }
            if (p4=='1')
            {
                count++;
            }

            Data[4] = count%2==0 ? '0' : '1';
            p4_ = Data[4];
        }

        if(i==8)
        {
            count=0;
            for(j=8;j<n+r;j=j+16)
            {
                if (Data[j] == '1')
                {
                    count++;
                }
                for(k=i+1;k<j+16;k++)
                {
                    if(Data[k] == '1')
                    {
                        count++;
                    }
                }

            }
            if (p8=='1')
            {
                count++;
            }

            Data[8] = count%2==0 ? '0' : '1';
            p8_ = Data[8];
```

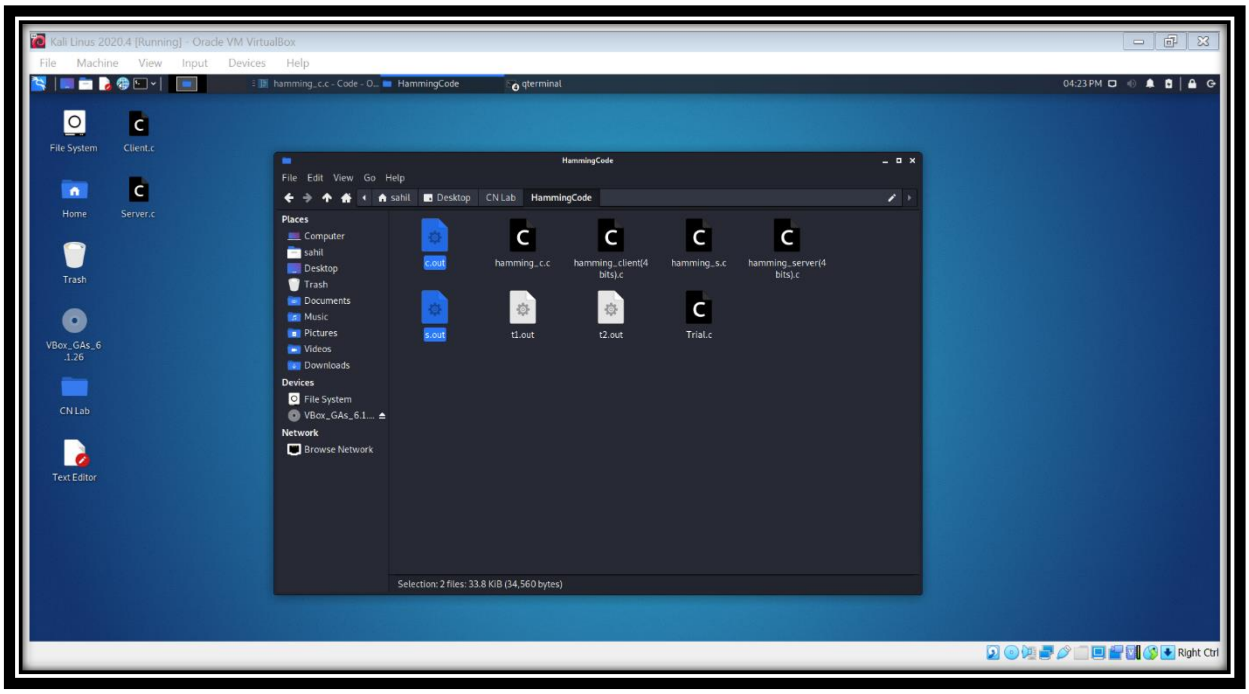
```
            //Data[i] = (Data[8]==Data[9] ? '0' : '1') == Data[10] ? '0' :
'1';
        }

    }

    int error_position;
    error_position = (p8_ -'0')*8 + (p4_ -'0')*4 + (p2_ -'0')*2 + (p1_-
'0')*1 ;

    printf("\n\n\tError Occur At : %d " , error_position+1);
    //error_position = (error_position +1) ;

    Data[error_position] = Data[error_position] =='1' ? '0' : '1' ;


    printf("\n");
    printf("\n\tFinal bits after correction:");
    printf("\n");
    for (i=0; i<=n+r ; i++)
    {
        printf("%c\t" , Data[i]);
    }

    char destuff[100];

    printf("\n\tData bits at receiver are:");
    printf("\n");
    for (i=2,j=0; i<=n+r ; i++,j++)
    {
        if(isPowerOfTwo(i))
        {
            i++;
        }

        destuff[j] = Data[i];

    }
    destuff[i] = '\0';
    for (i=strlen(destuff) - 1 ; i>=0 ; i--)
    {
        printf("%c\t" , destuff[i]);
    }

close(socket_server);
return 0;
}
```

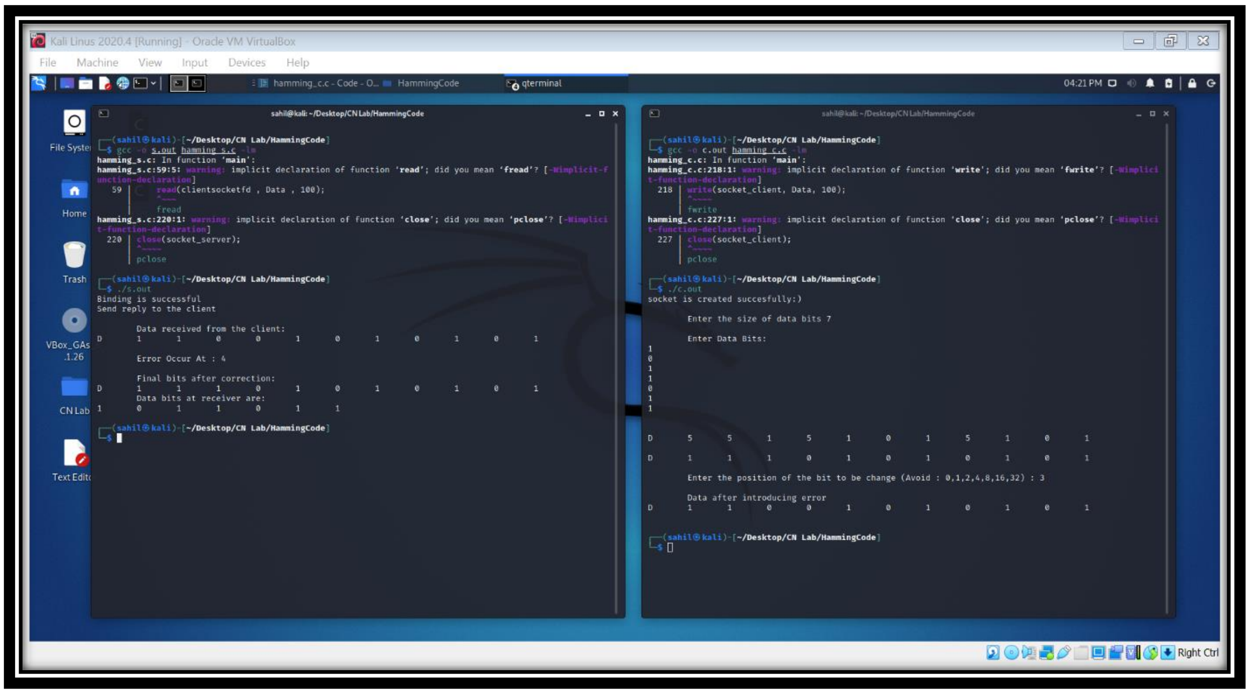
Client.c

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<stdbool.h>
#include<math.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>


bool isPowerOfTwo(int n)
{
   if(n==0)
   return false;

   return (ceil(log2(n)) == floor(log2(n)));
}


int main()
{

int socket_client , serversocketfd;
struct sockaddr_in serveraddress;
struct hostent *server;

socket_client = socket(AF_INET , SOCK_STREAM , 0);

if(socket_client<0)
printf("Socket is NOT created:(\n");
else
printf("socket is created succesfully:)\n");

serveraddress.sin_family = AF_INET;
serveraddress.sin_port = htons(9000);
serveraddress.sin_addr.s_addr= INADDR_ANY;


int connectionstatus = connect(socket_client,
                              (struct sockaddr *) &serveraddress,
                               sizeof(serveraddress));

if(connectionstatus == -1)
{
    printf("There was an error in the connection with server:( Try
again!\n");
}
```

```
    int n;
    printf("\n\tEnter the size of data bits ");
    scanf("%d",&n);

    char Data[100] , Data_Bits[100] , Parity_Bits[100];

    int i,j,k;
    int r=1;
    int count1;

    while(n+r+1 > pow(2,r))//counts the number of parity bits required
    {
        r++;
    }

    for (i=0; i<r; i++)//Putting 5 at the position of parity bits  (2^0 ,
2^1 , 2^2... likewise)
    {
        k=pow(2,i);
        Parity_Bits[i] = '5';//Just to understnd positions of parity
    }

    Parity_Bits[i] = '\0';

    printf("\n\tEnter Data Bits:\n");
    for(i=0; i<n; i++)
    {
        scanf("%s" , &Data_Bits[i]);
    }
    Data_Bits[i] = '\0';

    Data[0] = 'D';

    Data[1] = Parity_Bits[0];

    for (i=2,j=1,k=n-1; i<=n+r; i++)
    {

        if (isPowerOfTwo(i))
        {
            Data[i] = Parity_Bits[j];
            j++;
        }

        else
        {
            Data[i] = Data_Bits[k];
            k--;
        }
    }
    Data_Bits[i] = '\0';
    printf("\n\n");
    for (i=0; i<=n+r ; i++)
    {
        printf("%c\t" , Data[i]);
    }

    int count;
    char p1,p2,p4,p8;

    for (i=1; i<n+r ; i++)
    {
        if(Data[i] == '5')
        {
            if(i==1)
            {
                count=0;
                for(j=1;j<=n+r;j=j+2) //1,3,5,7,9,...
                {
                    if (Data[j] == '1')
                    {
                        count++;
                    }
                }
                Data[1] = count%2==0 ? '0' : '1';
                p1 = Data[1];

            }
            if(i==2)
            {
                count=0;
                for(j=2;j<=n+r;j=j+4)//2,3,6,7,10,11,...
                {
                    if (Data[j] == '1')
                    {
                        count++;
                    }
                    if(Data[j+1] == '1')
                    {
                        count++;
                    }
                }
                Data[2] = count%2==0 ? '0' : '1';
                p2 = Data[2];
            }
            if(i==4)
            {
                count=0;
                for(j=4;j<=n+r;j=j+8)//4,5,6,7,12,13,14,15,20,21,22,23,...
                {
                    if (Data[j] == '1')
                    {
                        count++;
                    }
                    for(k=j+1;k<j+8;k++)
                    {

                        if(Data[k] == '1')
                        {
                            count++;
                        }
                    }

                }
                Data[4] = count%2==0 ? '0' : '1';
                p4 = Data[4];
            }
            if(i==8)
            {
                count=0;
                for(j=8;j<n+r;j=j+16)
                {
                    if (Data[j] == '1')
                    {
                        count++;
                    }
                    for(k=i+1;k<j+16;k++)
                    {
                        if(Data[k] == '1')
                        {
                            count++;
                        }
                    }

                }
                Data[8] = count%2==0 ? '0' : '1';
                p8 = Data[8];
                //Data[i] = (Data[8]==Data[9] ? '0' : '1') == Data[10] ? '0'
: '1';
            }
```

```
        }
    }

    printf("\n\n");
    for (i=0; i<=n+r ; i++)
    {
        printf("%c\t" , Data[i]);
    }

    int x;
    printf("\n\n\tEnter the position of the bit to be change (Avoid :
0,1,2,4,8,16,32) : ");
    scanf("%d" , &x);

    if(x<n+r)
    {
        Data[x] = Data[x] == '1' ? '0' : '1' ;

        printf("\n\tData after introducing error");
        printf("\n");
        for (i=0; i<=n+r ; i++)
        {
            printf("%c\t" , Data[i]);
        }

    }


printf("\n");
write(socket_client, Data, 100);
write(socket_client, &n   , sizeof(n));
write(socket_client, &r   , sizeof(r));
write(socket_client, p1   , 1);
write(socket_client, p2   , 1);
write(socket_client, p4   , 1);
write(socket_client, p8   , 1);
printf("\n");

close(socket_client);

return 0;

}
```




Kali Linus 2020.4 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
hamming_c.c - Code - O... HammingCode qterminal
04:21 PM
sahil@kali: ~/Desktop/CN Lab/HammingCode
Right Ctrl

```
┌──(sahil㉿kali)-[~/Desktop/CN Lab/HammingCode]
└─$ gcc -o s.out hamming_s.c -lm
hamming_s.c: In function ‘main’:
hamming_s.c:59:5: warning: implicit declaration of function ‘read’; did you mean ‘fread’? [-Wimplicit-f
unction-declaration]
   59 |     read(clientsocketfd , Data , 100);
      |     ^~~~
      |     fread
hamming_s.c:220:1: warning: implicit declaration of function ‘close’; did you mean ‘pclose’? [-Wimplici
t-function-declaration]
  220 | close(socket_server);
      | ^~~~~
      | pclose

┌──(sahil㉿kali)-[~/Desktop/CN Lab/HammingCode]
└─$ ./s.out
Binding is successful
Send reply to the client

        Data received from the client:
D       1       1       0       0       1       0       1       0       1       0       1

        Error Occur At : 4

        Final bits after correction:
D       1       1       1       0       1       0       1       0       1       0       1
        Data bits at receiver are:
1       0       1       1       0       1       1

┌──(sahil㉿kali)-[~/Desktop/CN Lab/HammingCode]
└─$ 
```

```
┌──(sahil㉿kali)-[~/Desktop/CN Lab/HammingCode]
└─$ gcc -o c.out hamming_c.c -lm
hamming_c.c: In function ‘main’:
hamming_c.c:218:1: warning: implicit declaration of function ‘write’; did you mean ‘fwrite’? [-Wimplici
t-function-declaration]
  218 | write(socket_client, Data, 100);
      | ^~~~~
      | fwrite
hamming_c.c:227:1: warning: implicit declaration of function ‘close’; did you mean ‘pclose’? [-Wimplici
t-function-declaration]
  227 | close(socket_client);
      | ^~~~~
      | pclose

┌──(sahil㉿kali)-[~/Desktop/CN Lab/HammingCode]
└─$ ./c.out
socket is created succesfully:)

        Enter the size of data bits 7

        Enter Data Bits:
1
0
1
1
0
1
1


D       5       5       1       5       1       0       1       5       1       0       1

D       1       1       1       0       1       0       1       0       1       0       1

        Enter the position of the bit to be change (Avoid : 0,1,2,4,8,16,32) : 3

        Data after introducing error
D       1       1       0       0       1       0       1       0       1       0       1


┌──(sahil㉿kali)-[~/Desktop/CN Lab/HammingCode]
└─$ 
```