SSN COLLEGE OF ENGINEERING, KALAVAKKAM
(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NETWORKS LAB
EXERCISE 6

Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 205001049*

Computing Hamming code for Error correction

Aim:

To implement Hamming Code for Single Error Correction using C socket program

Algorithm:

SERVER

Establish TCP/IP connection from server to client

1. Read the input from a user (zero's and one's)
2. Encoding a message by Hamming Code
    a. Calculate the number of redundant bits.
    b. Position the redundant bits.
    c. Calculate the values of each redundant bit.
3. Introduce error (single bit error or no error)
4. Send the data to receiver

CLIENT

Establish TCP/IP connection from server to client

1. Receive the data from the sender and.
2. Check for any error by performing the following operations
    a. Calculation of the number of redundant bits.
    b. Positioning the redundant bits.
    c. Parity checking.
    d. If any error, correct the error and display the original message.

Code:
Server

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<arpa/inet.h>
#include<fcntl.h>
#include<stdbool.h>
#include<math.h>
#define MAXSIZE 1024

int binary(int num)
{
    int bin = 0, r;
    int i = 0;
    while(num > 0)
    {
        r = num % 2;
        bin += r * pow(10, i);
        num /= 2;
        i++;
    }
    return bin;
}
int ispresent(int num,int pos)
{
    int rem;
    for(int i = 0; i < pos; i++)
    {
        rem = num % 10;
        num = num / 10;
    }
    if(rem == 1)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
int isapower2(int n)
{
    if(ceil(log2(n)) == floor(log2(n)))
```

```
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
int main(int argc, char ** argv)
{
    srand(time(NULL));

    if (argc < 2){
        fprintf(stderr, "Enter port number as argument!\n");
        exit(EXIT_FAILURE);
    }

    int PORT = atoi(argv[1]);

    int sockfd, newfd, n, arr[30];
    char buff[MAXSIZE], buffer[MAXSIZE], data_t[40];
    long data;
    struct sockaddr_in servaddr,clientaddr;

    if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        perror("Socket creation failed!");
        exit(1);
    }

    bzero(&servaddr,sizeof(servaddr));
    servaddr.sin_family = AF_INET; // IPv4
    servaddr.sin_addr.s_addr = INADDR_ANY;
    servaddr.sin_port = htons(PORT);

    if(bind(sockfd, (const struct sockaddr *)&servaddr,sizeof(servaddr)) < 0)
    {
        perror("Bind failed!");
        exit(1);
    }

    listen(sockfd, 2);
    printf("Enter the data to send: ");
    scanf("%lu", &data);
    long num =data,count = 0;
    while(num > 0)
    {
        num = num / 10;
        count++;
    }
```

```
    n=count;
    int r=1;
    //r = log2(n);
    //r = floor(r);

    // Finding number of redundant bits
    while(pow(2, r) < n + r + 1)
    {
        r += 1;
    }


    int total = n + r;

    printf("\nNo. of redundant bits : %d\nTotal no. of bits:%d\n", r,total);
    //nob = floor(log2(total));

    //creating spaces for redundant bits
    for(int i = 1; i <= total; i++)
    {
        int digit = data % 10;
        if(isapower2(i) == 0)
        {
            arr[total - i] = digit;
            data /= 10;
        }
        else
        {
            arr[total-i]=0;
        }
    }

    //assigning redundant bits
    for(int i = 0; i < r; i++)
    {
        for(int j = 1; j <= total; j++)
        {
            if((int)(pow(2, i)) != j)
            {
                int bin = binary(j);
                if(ispresent(bin, i + 1))
                    count += arr[total - j];
            }
        }
        if(count % 2 == 0)
            arr[total - (int)(pow(2, i))] = 0;
        else
            arr[total - (int)(pow(2, i))] = 1;
        count = 0;
    }
```

```
    printf("\nData with redundant bits: ");
    for(int i = 0; i < total; i++)
    {
        printf("%d", arr[i]);
    }
    // printf("\nEnter error position: ");
    // scanf("%d", &pos);

    int pos = rand() % total + 1;
    printf("\nIntroducing error randomly at bit: %d\n", pos);

    if(arr[total - pos] == 0)
    {
        arr[total - pos] = 1;
    }
    else
    {
        arr[total - pos] = 0;
    }

    int k = 0;
    num = 0;
    for(int i = total - 1; i >= 0; i--)
    {
        num += pow(10, k) * arr[i];
        k++;
    }
    sprintf(data_t, "%lu", num);
    printf("Data transmitted is %s\n", data_t);
    int len = sizeof(clientaddr);
    newfd = accept(sockfd, (struct sockaddr*)&clientaddr, &len);
    int m = write(newfd, data_t, sizeof(data_t));
}
```

Client

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>
#include<unistd.h>
#include<arpa/inet.h>
#include<stdlib.h>
#include<math.h>
#define MAXLINE 1024
```

```
int countbits(long num)
{
    int r, count = 0;
    while(num > 0)
    {
        num = num / 10;
        count++;
    }
    return count;
}

int binary(int num)
{
    int bin = 0, r;
    int i = 0;
    while(num > 0)
    {
        r = num % 2;
        bin += r * pow(10, i);
        num /= 2;
        i++;
    }
    return bin;
}

int ispresent(int num,int pos)
{
    int rem;
    for(int i = 0; i < pos; i++)
    {
        rem = num % 10;
        num = num / 10;
    }
    if(rem == 1)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

int decimal(int num)
{
    int rem, i = 0, result;
    while(num > 0)
    {
        rem = num % 10;
```

```
        result += pow(2, i) * rem;
        num /= 10;
        i++;
    }
    return result;
}


int main(int argc, char **argv)
{
    if (argc < 2){
        fprintf(stderr, "Please enter port number as second argument!\n");
        exit(EXIT_FAILURE);
    }

    int PORT = atoi(argv[1]);

    long num;
    int sockfd, arr[20], newarr[20], finalarr[20];
    char buffer1[40];int binary(int num)
{
    int bin = 0, r;
    int i = 0;
    while(num > 0)
    {
        r = num % 2;
        bin += r * pow(10, i);
        num /= 2;
        i++;
    }
    return bin;
}
    struct sockaddr_in servaddr;
    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("Socket creation failed!");
        exit(1);
    }
    bzero(&servaddr,sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_port = htons(PORT);
    servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    int n, len;
    connect(sockfd, (struct sockaddr*)&servaddr, sizeof(servaddr));
    n = read(sockfd, buffer1, sizeof(buffer1));
    num = atol(buffer1);


    int count = 0;
    long num1=num;
    while(num1 > 0)
```

```
{
    num1 = num1 / 10;
    count++;
}
int total = count;


printf("Received data: %lu\nTotal bits:%d\n", num,total);


int i = 1;
while(num > 0)
{
    int rem = num % 10;
    arr[total - i] = rem;
    num /= 10;
    i++;
}


//calculationg number of redundant bits
int r=0;
for(i = 1; i <= total; i++)
{
    if(ceil(log2(i)) == floor(log2(i)))
    {
        r++;
    }
}

//checking parity of redundant bits
int result=0,k = 0;
for(i = 0; i < 4; i++)
{
    for(int j = 1; j <= total; j++)
    {
        int bin = binary(j);
        if(ispresent(bin, i + 1))
        {
            count += arr[total - j];
        }
    }
    if(count % 2 == 0)
    {
        result += pow(10, k) * 0;
    }
    else
    {
        result += pow(10, k) * 1;
    }
```

```
        k++;
        count=0;
    }


    int error = decimal(result);
    printf("\nError bit in binary: %d\n", result);
    printf("\nError in bit in decimal %d\n", error);

    //error correction at error bit
    if(arr[total - error] == 0)
    {
        arr[total - error] = 1;
    }
    else
    {
        arr[total - error] = 0;
    }


    printf("\nMessage after error correction: ");
    for(i = 0; i < total; i++)
    {
        printf("%d", arr[i]);
    }
    printf("\n");


    //shifting bits to its position
    k = 0;
    for(i = total - 1; i >= 0; i--)
    {
        newarr[k] = arr[i];
        k++;

    }

    //removing redundant bits
    int x = 0;
    for(i = 0;i < k; i++)
    {
        if(ceil(log2(i + 1)) != floor(log2(i + 1)))
        {
            finalarr[x] = newarr[i];
            x++;
        }
    }

    printf("\nData after error correction and removing redundant bits: ");
    for(i = x - 1; i >= 0; i--)
```

```
        printf("%d", finalarr[i]);
    printf("\n");
    return 0;
}
```

Output:

Server :

```
Enter the data to send: 1010101

No. of redundant bits : 4
Total no. of bits:11

Data with redundant bits: 10100101110
Introducing error randomly at bit: 5
Data transmitted is 10100111110
```

```
Enter the data to send: 1010101

No. of redundant bits : 4
Total no. of bits:11

Data with redundant bits: 10100101110
Introducing error randomly at bit: 3
Data transmitted is 10100101010
```

```
root@spl13:~/Desktop/Jayannthan/MB/NetworksLab-master/Assignment-07# ./s 8080
Enter the data to send: 1010101

No. of redundant bits : 4
Total no. of bits:11

Data with redundant bits: 10100101110
Introducing error randomly at bit: 8
Data transmitted is 10110101110
```

Client:

```
Received data: 10100111110
Total bits:11

Error bit in binary: 101

Error in bit in decimal 5

Message after error correction: 10100101110

Data after error correction and removing redundant bits: 1010101
```

```
Received data: 10100101010
Total bits:11

Error bit in binary: 11

Error in bit in decimal 3

Message after error correction: 10100101110

Data after error correction and removing redundant bits: 1010101
```

```
root@spl13:~/Desktop/Jayannthan/MB/NetworksLab-master/Assignment-07# ./c 8080
Received data: 10110101110
Total bits:11

Error bit in binary: 1000

Error in bit in decimal 8

Message after error correction: 10100101110

Data after error correction and removing redundant bits: 1010101
```

Learning outcome:

Learnt the working of hamming code error correction