UCS1511 - COMPUTER NETWORKS

Simulation of Error Correction using Hamming code

REG NO : 205001085 **EX.NO : 9**
NAME : SABARIVASAN
DATE : 10.10.22

OBJECTIVE :

To be proficient in developing an application to simulate error correction using hamming code technique using socket programming in C

CODE :

CLIENT :

```
#include <netinet/in.h> //structure for storing address information
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h> //for socket APIs
#include <sys/types.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#define PORT 6666
int find_r(char *code)
{
```

```
        int r = 0, m = strlen(code);
        int cur = 1;
        while (cur < m + r + 1)
        {
        cur *= 2;
        r++;
        };
        return r;
}
char *encode(char *code, int r, int err)
{
        int c = 1, n = strlen(code) + r;
        char enc_code[n + 1];
        int i = 0;
        while (i < r)
        {
        enc_code[n - c] = 'r';
        c *= 2;
        i++;
        }
        int k = 0;
        for (i = 0; i < n; i++)
        {
        if (enc_code[i] != 'r')
        {
        enc_code[i] = code[k++];
        }
        }
        enc_code[i] = '\0';
        int chk = 1;
        while (r--)
        {
```

```
        int o = 0;
        for (int i = n - chk; i >= 0; i -= 2 * chk)
        {
        for (int j = i; j > i - chk; j--)
        {
                if (j < 0)
                break;
                if (enc_code[j] == '1')
                o++;
        }
        }
        enc_code[n - chk] = o % 2 ? '1' : '0';
        chk *= 2;
        }
        if (err != -1)
        enc_code[err] = (enc_code[err] == '1') ? '0' : '1';
        char *enc = enc_code;
        return enc;
}
int main(int argc, char **argv)
{
        int client_socket = socket(AF_INET, SOCK_STREAM, 0);
        if (client_socket < 0)
        {
        perror("[-]Error in socket");
        exit(1);
        }
        printf("[+]Server socket created. \n");

        struct sockaddr_in client_address;
        client_address.sin_family = AF_INET;
        client_address.sin_port = htons(PORT);
```

```
        client_address.sin_addr.s_addr = INADDR_ANY;

        int connectStatus = connect(client_socket, (struct sockaddr
*)&client_address, sizeof(client_address));
        if (connectStatus == -1)
        {
        perror("[-]Error in Connecting");
        exit(1);
        }

        char *enc = encode(argv[1], find_r(argv[1]), atoi(argv[2]));
        send(client_socket, enc, strlen(argv[1]) + find_r(argv[1]), 0);
        char msg[255];
        recv(client_socket, msg, sizeof(msg), 0);
        printf("Message:%s\n", msg);
        close(client_socket);
        return 0;
}
```

SERVER :

```
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#define PORT 6666
```

```
int find_r(char *code)
{
	int r = 0, m = strlen(code);
	int cur = 1;
	while (cur < m + r + 1)
	{
	cur *= 2;
	r++;
	};
	return r;
}
void decode(char *code, int r)
{
	int c = 1, n = strlen(code);
	int parity[r];
	int x = 0;
	int chk = 1;
	while (r > x)
	{
	int o = 0;
	for (int i = n - chk; i >= 0; i -= 2 * chk)
	{
	for (int j = i; j > i - chk; j--)
	{
		if (j < 0)
		break;
		if (code[j] == '1')
		o++;
	}
	}
	parity[x++] = o % 2 ? 1 : 0;
	chk *= 2;
```

```
        }
        int val = 0, v = 1;
        for (int j = 0; j < r; j++)
        {
        val += v * parity[j];
        v *= 2;
        }
        printf("\nError in bit %d", n - val);
        printf("\nCorrected Code : ");
        for (int i = 0; i < n; i++)
        {
        if (i == n - val)
        {
        if (code[i] == '1')
                printf("0");
        else
                printf("1");
        }
        else
        printf("%c", code[i]);
        }
        printf("\n");
}
int main(int argc, char **argv)
{
        int server_socket = socket(AF_INET, SOCK_STREAM, 0);
        if (server_socket < 0)
        {
        perror("[-]Error in socket");
        exit(1);
        }
        printf("[+]Server socket created. \n");
```

```
	struct sockaddr_in server_address;
	server_address.sin_family = AF_INET;
	server_address.sin_port = htons(PORT);
	server_address.sin_addr.s_addr = INADDR_ANY;

	int e = bind(server_socket, (struct sockaddr *)&server_address,
sizeof(server_address));
	if (e < 0)
	{
	perror("[-]Error in Binding");
	exit(1);
	}
	printf("[+]Binding Successfull.\n");

	e = listen(server_socket, 1);
	if (e == 0)
	{
	printf("[+]Listening...\n");
	}
	else
	{
	perror("[-]Error in Binding");
	exit(1);
	}

	int client_socket = accept(server_socket, NULL, NULL);
	char code[255];
	recv(client_socket, code, sizeof(code), 0);
	int m = strlen(code);
	printf("\nRecieved : %s\n", code);
	char *msg = "msg received";
```

```
    decode(code, find_r(code));
    send(client_socket, msg, strlen(msg), 0);
    close(server_socket);
    return 0;
}
```

OUTPUT :