Computer Networks Lab Report

Md Talha Yaseen Khan 111701019

Lab-7 (Socket Programming and Hamming Codes)

1. Client

◦ Code Overview

- Function `void stringToBinary(char *S, int *bin, int l)` is used to convert the string S into the binary bits (8 bits/char) and storing binary bits in bin.
- Function `void generatedCode(int *B, int *A, int r)` is used to generate (m+r) bits code from m bits code in B with all the redundant bits value 0 and storing into A.
- Function `void checkBit(int *A, int m, int r)` is is used to find the value redundant bits.
- Function `int findr(int m)` is finding the value of number of minimum redundant bits possible for a given m.
- In the main() function, first I am finding the number packet possible to send to port number 8567 of h2 from h1. And making the number of bits divisible by m if not, by padding minimum zeores to the end of bin. After converting string into bits, I am connecting to UDP port number 8567 of h2. Then I am sending (m+r) length frames of char bits to the h2. And at last sending an extra rough string of char '0' sending to find the moment where previous string is completely sent.

2. Server

◦ Code Overview

- Function **void correct_err(char* frame, int fSize)** is used to correct the error occured by flipping one bit of frame by h2.
- Function **int deframe(char *frame, int fSize, int binSize)** is used to deframe (m+r) bits frame to m bits data bit after error correcting.
- In the main() function, Firstly connecting to port 9567 of h3. Then going into infinite loop and receiving the frames one by one which is sent by h2 at port number 9567 of h3. And correcting the frames error and deframming into m bits and storing these m bits to bin which is storing same as the bits of string sent by client. Then taking each 8-bits of bin and converting into character and storing these character into result array. After that if I found that the current length of frames which I received is not same as previous frame length means whole string is sent and then printing all character of result with newline.

Commands:

- **Command to copy client program from local machine to h1:**
  - *scp -P 14501 client.c tc@localhost:client.c*
  - Password is *user@123*
- **Command to copy server program from local machine to h3:**
  - *scp -P 14503 server.c tc@localhost:server.c*
  - Password is *user@123*

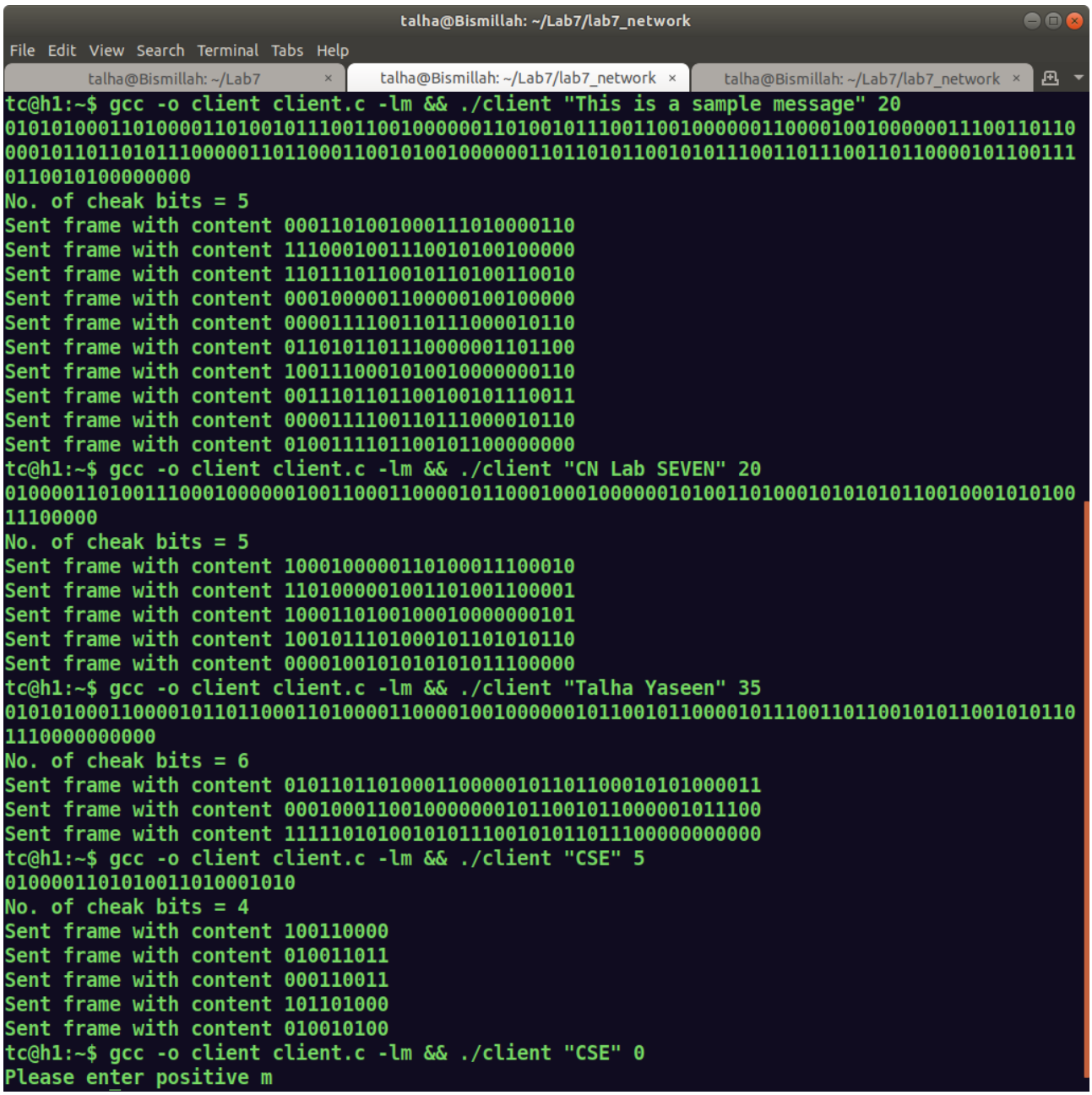
1. **Command to run client program**
   - *gcc -o client client.c -lm*
     ./client "String" m
2. **Command to run server program**
   - *gcc -o server server.c -lm*

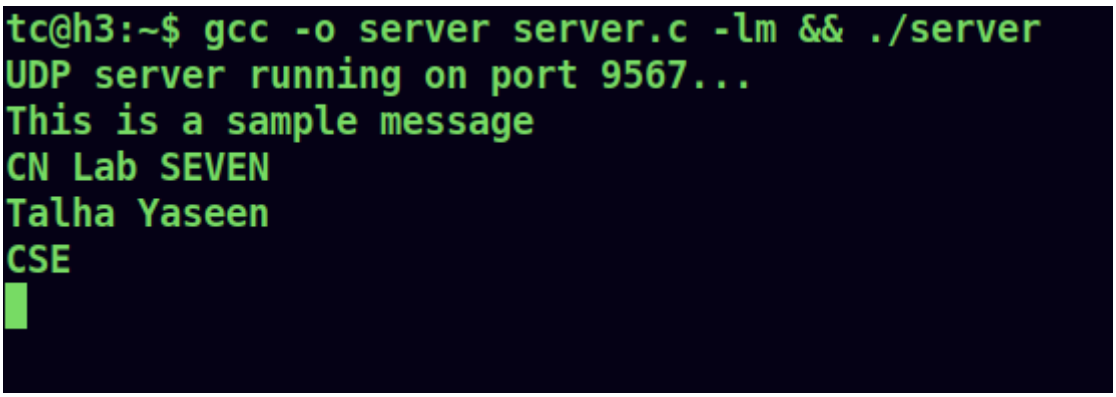
./server

Below attaching some snapshots:

```
tc@h3:~$ sudo tcpdump -i eth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes
05:44:33.130021 IP 192.168.1.2.8567 > 192.168.1.3.9567: UDP, length 13
05:44:33.130052 IP 192.168.1.3 > 192.168.1.2: ICMP 192.168.1.3 udp port 9567 unreachable, length 49
05:44:33.130463 IP 192.168.1.2.8567 > 192.168.1.3.9567: UDP, length 13
05:44:33.130473 IP 192.168.1.3 > 192.168.1.2: ICMP 192.168.1.3 udp port 9567 unreachable, length 49
05:44:33.130898 IP 192.168.1.2.8567 > 192.168.1.3.9567: UDP, length 13
05:44:33.130906 IP 192.168.1.3 > 192.168.1.2: ICMP 192.168.1.3 udp port 9567 unreachable, length 49
05:44:33.131346 IP 192.168.1.2.8567 > 192.168.1.3.9567: UDP, length 13
05:44:33.131354 IP 192.168.1.3 > 192.168.1.2: ICMP 192.168.1.3 udp port 9567 unreachable, length 49
05:44:33.131800 IP 192.168.1.2.8567 > 192.168.1.3.9567: UDP, length 13
05:44:33.131808 IP 192.168.1.3 > 192.168.1.2: ICMP 192.168.1.3 udp port 9567 unreachable, length 49
05:44:33.132321 IP 192.168.1.2.8567 > 192.168.1.3.9567: UDP, length 13
05:44:33.132330 IP 192.168.1.3 > 192.168.1.2: ICMP 192.168.1.3 udp port 9567 unreachable, length 49
05:44:33.132879 IP 192.168.1.2.8567 > 192.168.1.3.9567: UDP, length 13
05:44:33.133405 IP 192.168.1.2.8567 > 192.168.1.3.9567: UDP, length 13
05:44:33.133809 IP 192.168.1.2.8567 > 192.168.1.3.9567: UDP, length 13
05:44:33.134270 IP 192.168.1.2.8567 > 192.168.1.3.9567: UDP, length 13
05:44:33.134708 IP 192.168.1.2.8567 > 192.168.1.3.9567: UDP, length 13
05:44:33.135140 IP 192.168.1.2.8567 > 192.168.1.3.9567: UDP, length 13
```

```
talha@Bismillah:~/Lab7$ scp -P 14501 client.c tc@localhost:client.c
tc@localhost's password:
client.c                                      100% 2987     5.6MB/s   00:00
talha@Bismillah:~/Lab7$ scp -P 14503 server.c tc@localhost:server.c
tc@localhost's password:
server.c                                      100% 2503     3.2MB/s   00:00
talha@Bismillah:~/Lab7$
```


talha@Bismillah: ~/Lab7/lab7_network
File Edit View Search Terminal Tabs Help
talha@Bismillah: ~/Lab7
talha@Bismillah: ~/Lab7/lab7_network
talha@Bismillah: ~/Lab7/lab7_network
tc@h1:~$ gcc -o client client.c -lm && ./client "This is a sample message" 20
0101010001101000011010010111001100100000011010010111001100100000011000010010000011100110110
0001011011010111000001101100011001010010000001101101100101011100110111001101100001011001111
0110010100000000
No. of cheak bits = 5
Sent frame with content 000110100100011101000110
Sent frame with content 111000100111001010010000
Sent frame with content 110111011001011010011001 0
Sent frame with content 000100000110000010010000 0
Sent frame with content 000011110011011100001011 0
Sent frame with content 011010110111000000110110 0
Sent frame with content 100111000101001000000011 0
Sent frame with content 001110110110010010111001 1
Sent frame with content 000011110011011100001011 0
Sent frame with content 010011110110010110000000 0
tc@h1:~$ gcc -o client client.c -lm && ./client "CN Lab SEVEN" 20
0100001101001110001000000100110001100001011000100010000001010011010001010101011001000101010011100000
No. of cheak bits = 5
Sent frame with content 100010000110100011100010
Sent frame with content 110100000100110100110000 1
Sent frame with content 100011010010001000000010 1
Sent frame with content 100101110100010110101011 0
Sent frame with content 000010010101010101110000 0
tc@h1:~$ gcc -o client client.c -lm && ./client "Talha Yaseen" 35
0101010001100001011011000110100001100001001000000101100101100001011100110110010101100101011
1110000000000
No. of cheak bits = 6
Sent frame with content 010110110100011000001011011000101010000 11
Sent frame with content 000100011001000000010110010110000001011100
Sent frame with content 111110101001010111001010110111000000000000
tc@h1:~$ gcc -o client client.c -lm && ./client "CSE" 5
0100001101010011010001010
No. of cheak bits = 4
Sent frame with content 100110000
Sent frame with content 010011011
Sent frame with content 000110011
Sent frame with content 101101000
Sent frame with content 010010100
tc@h1:~$ gcc -o client client.c -lm && ./client "CSE" 0
Please enter positive m


tc@h3:~$ gcc -o server server.c -lm && ./server
UDP server running on port 9567...
This is a sample message
CN Lab SEVEN
Talha Yaseen
CSE