Design and Analysis of Computer Networks
LAB ASSIGNMENT 1 - REPORT

Group Members :

1. Name : Sai Saneep Nagalla
   Email : szn0080@auburn.edu

2. Name : Bhanu Lalith Aakash Medury
   Email : bzm0092@auburn.edu

Lab 1.1: Introduction to Socket Programming :

As we have used C language, we used GCC compiler to compile the code:

a) We have created a concurrent UDP echo server which is listening to the port 10010. This server sends back any message that it receives which is given by us.

The code submitted is working as expected.

To compile the c code, we need to enter the following command in terminal which will create an executable file:

```
gcc server11.c -o server11
```

To run the executable file , we need to give the following command:

```
./server11 &
```

We are running both server and client on the same machine, so we have to run server in the background in order to communicate with the server as server takes the requests, and we give a "&" symbol after ./server11 .
So the process runs in the background
Server receives the message from the client and sends the Echo back to the client again.

b) We have created a UDP Client which is also running on the same machine where the server is running so they can communicate with each other.

The code submitted is working as expected.

To compile the code, we need to give the following command:

```
gcc client11b.c -o client11b
```

To run the executable file , we need to give the following command:

```
./client11b <Host IP address >
```

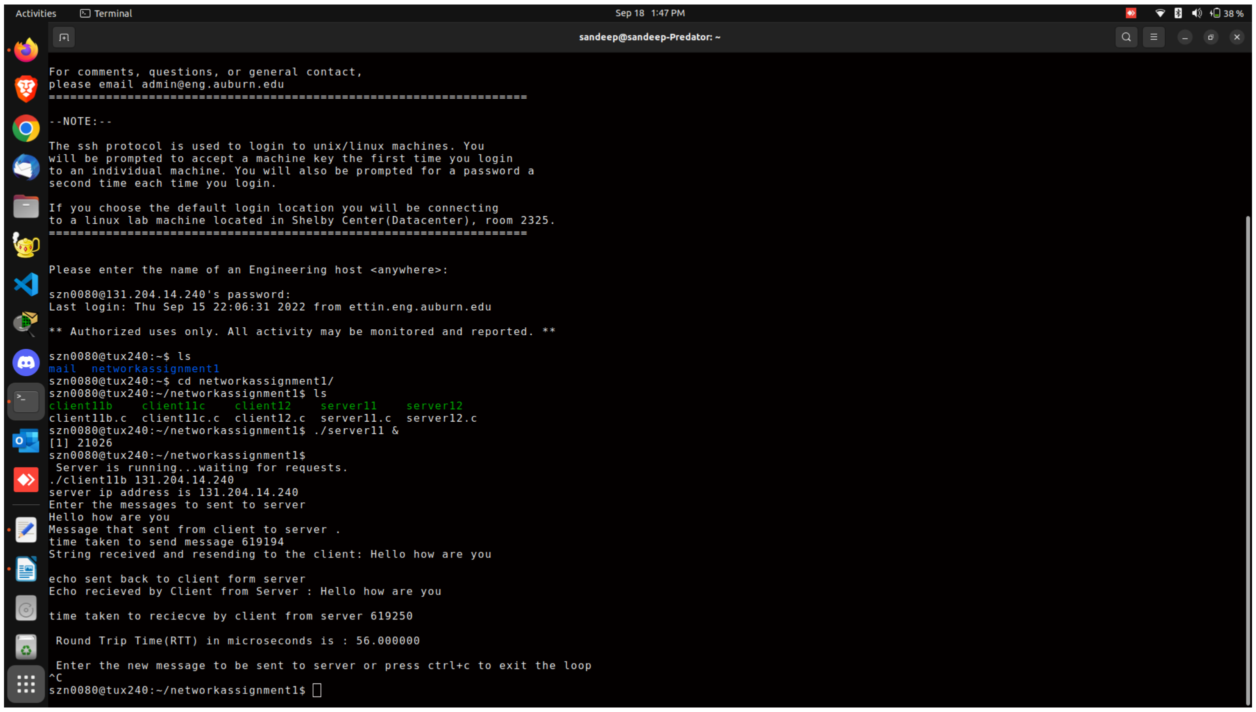
where Host IP address is the ip address of the TUX machine that the code is running or Therefore, the command will be:

```
./client11b 131.204.14.240
```

We have to run the client after server is running in the background

After running the client, a successful connection will be established between server and client11b.As a result, a message will be popped up on the console prompting user input, which will be forwarded to the server expecting an echo back from the server
. Further, we will also get additional information such as:
1. Time at which the message is sent and received in microseconds
2. Echoed message from the server
3. Round Trip Time (RTT) in microseconds is
.
After that, it prompts you to type a new message to be sent to the server or to terminate the process by pressing ctrl+c since it is a loop which will only terminate by ctrl+c.

Output:



```
For comments, questions, or general contact,
please email admin@eng.auburn.edu
==================================================================

--NOTE:--

The ssh protocol is used to login to unix/linux machines. You
will be prompted to accept a machine key the first time you login
to an individual machine. You will also be prompted for a password a
second time each time you login.

If you choose the default login location you will be connecting
to a linux lab machine located in Shelby Center(Datacenter), room 2325.
==================================================================


Please enter the name of an Engineering host <anywhere>:

szn0080@131.204.14.240's password:
Last login: Thu Sep 15 22:06:31 2022 from ettin.eng.auburn.edu

** Authorized uses only. All activity may be monitored and reported. **

szn0080@tux240:~$ ls
mail  networkassignment1
szn0080@tux240:~$ cd networkassignment1/
szn0080@tux240:~/networkassignment1$ ls
client11b    client11c    client12    server11    server12
client11b.c  client11c.c  client12.c  server11.c  server12.c
szn0080@tux240:~/networkassignment1$ ./server11 &
[1] 21026
szn0080@tux240:~/networkassignment1$
 Server is running...waiting for requests.
./client11b 131.204.14.240
server ip address is 131.204.14.240
Enter the messages to sent to server
Hello how are you
Message that sent from client to server .
time taken to send message 619194
String received and resending to the client: Hello how are you

echo sent back to client form server
Echo recieved by Client from Server : Hello how are you

time taken to reciecve by client from server 619250

 Round Trip Time(RTT) in microseconds is : 56.000000

 Enter the new message to be sent to server or press ctrl+c to exit the loop
^C
szn0080@tux240:~/networkassignment1$ 
```

c)
We have created a client which performs two processes:
1. it will pass number from 1 to 10000 as a string to the server
2. It will receive the 10000 echoes back from the server.

The code submitted is working as expected.
To compile the code, we need to enter the following command:

```
gcc client11c.c -o client11c
```

To run the executable file , we need to give the following command:

```
./client11c <Host IP adress >
```

where Host IP adress is the ip adress of the TUX machine that the code is running or
Therefore, the command will be:

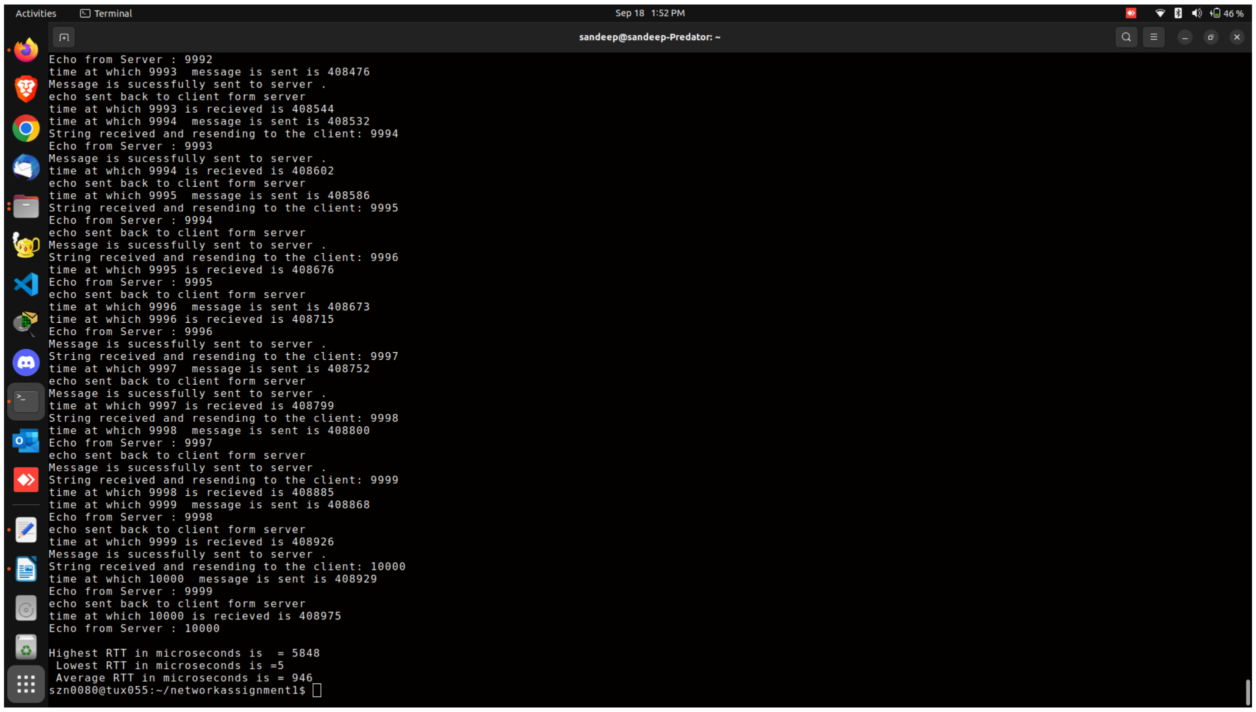
```
./client11c 131.204.14.240
```

Once you run the code, you can see that string 1 to 10000 are sent and received concurrently.
We will have a message being printed on the console which includes the following:

The Message sent to the server and the time at which it is sent to the server.
The Echo that is received from the server and the time at which it is received.
If there are any missing echoes they will be printed, if there are no missing echoes the missing echoes will not be printed.global variable MSIZE can be changed in the code as per requirements to any number currently i have given 10000

At the end we will be having the Round Trip Time (RTT) being printed we have the Highest RTT , Lowest RTT , Average RTT being printed .

Output:



```
Echo from Server : 9992
time at which 9993  message is sent is 408476
Message is sucessfully sent to server .
echo sent back to client form server
time at which 9993 is recieved is 408544
time at which 9994  message is sent is 408532
String received and resending to the client: 9994
Echo from Server : 9993
Message is sucessfully sent to server .
time at which 9994 is recieved is 408602
echo sent back to client form server
time at which 9995  message is sent is 408586
String received and resending to the client: 9995
Echo from Server : 9994
echo sent back to client form server
Message is sucessfully sent to server .
String received and resending to the client: 9996
time at which 9995 is recieved is 408676
Echo from Server : 9995
echo sent back to client form server
time at which 9996  message is sent is 408673
time at which 9996 is recieved is 408715
Echo from Server : 9996
Message is sucessfully sent to server .
String received and resending to the client: 9997
time at which 9997  message is sent is 408752
echo sent back to client form server
Message is sucessfully sent to server .
time at which 9997 is recieved is 408799
String received and resending to the client: 9998
time at which 9998  message is sent is 408800
Echo from Server : 9997
echo sent back to client form server
Message is sucessfully sent to server .
String received and resending to the client: 9999
time at which 9998 is recieved is 408885
time at which 9999  message is sent is 408868
Echo from Server : 9998
echo sent back to client form server
time at which 9999 is recieved is 408926
Message is sucessfully sent to server .
String received and resending to the client: 10000
time at which 10000  message is sent is 408929
Echo from Server : 9999
echo sent back to client form server
time at which 10000 is recieved is 408975
Echo from Server : 10000

Highest RTT in microseconds is  = 5848
 Lowest RTT in microseconds is =5
 Average RTT in microseconds is = 946
szn0080@tux055:~/networkassignment1$ 
```

Lab 1.2: TCP Calculator :

We have created a TCP calculator server which listens to the port 10020. This server performs basic mathematical operations such as +,-,*,/ given by the user between two integers
The code submitted is working as expected.

To Compile the code, we need to give the following command:

```
gcc client12.c -o client12
gcc server12.c -o server12
```

To run the executable file , we need to give the following command:

```
./server12 &
```

Here we are running both server and client on the same machine, so we have to run server in the background in order to communicate with the server as server takes the requests, and we give a “&” symbol after ./server12 . So the process runs in the background
Then, we input the following to run the client :

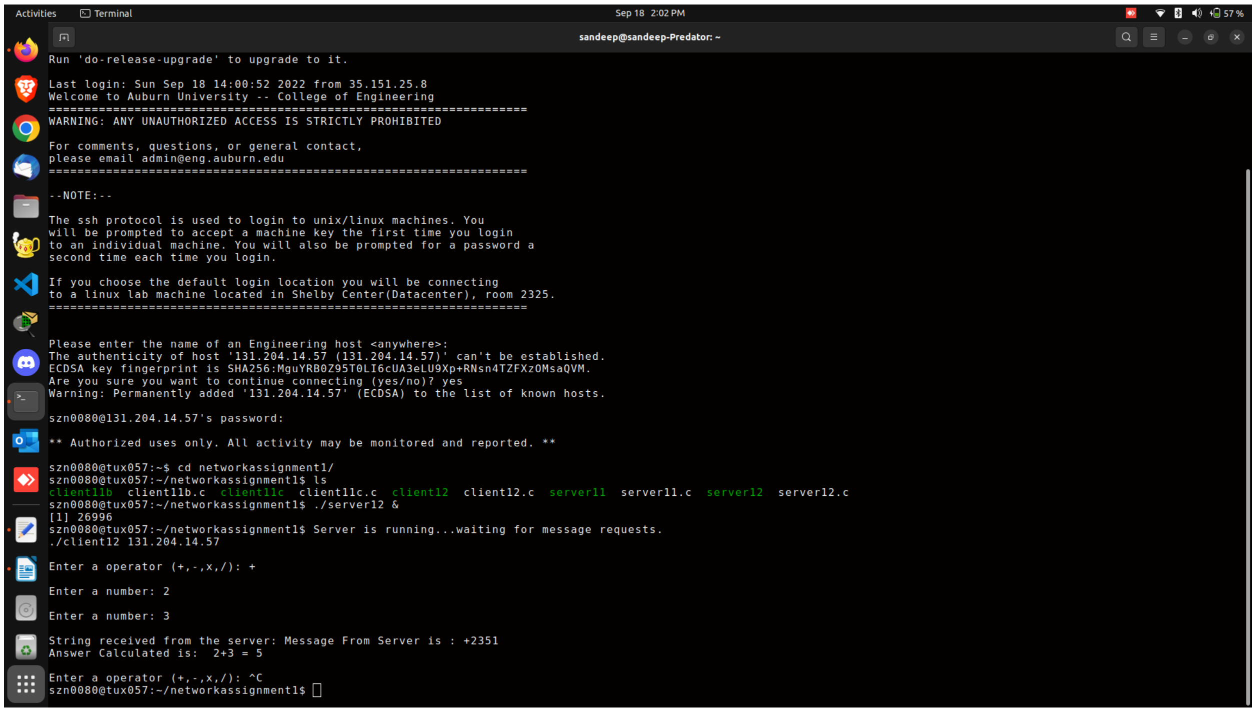
```
./client12 <Host ip address>
```

Here, we follow the run client command with the IP address of the device that is running the server. For example, if we are running a server on tux machine062, the ip address is 131.204.14.57.

After running the client it will prompt you to enter the operation like +,-,*,/ and two integer operands are provided as input.
Inputs are then bundled and forwarded to the server. Server decodes them and calculates the result as per operation and operands given.
If the operation is valid, the loop prompts the user to enter the new operator and operands to perform another calculation until ctrl+c was given to terminate the program.

When we are sending the message from client to server, we are storing data in a byte array of size 9, so the size of message sent is always 9 bytes. Similarly, when we send a message from server to client, we are sending it in a byte array of size 14, so the client always receives message of size 14 bytes .

output:



```
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Sep 18 14:00:52 2022 from 35.151.25.8
Welcome to Auburn University -- College of Engineering
==================================================================
WARNING: ANY UNAUTHORIZED ACCESS IS STRICTLY PROHIBITED

For comments, questions, or general contact,
please email admin@eng.auburn.edu
==================================================================

--NOTE:--

The ssh protocol is used to login to unix/linux machines. You
will be prompted to accept a machine key the first time you login
to an individual machine. You will also be prompted for a password a
second time each time you login.

If you choose the default login location you will be connecting
to a linux lab machine located in Shelby Center(Datacenter), room 2325.
==================================================================


Please enter the name of an Engineering host <anywhere>:
The authenticity of host '131.204.14.57 (131.204.14.57)' can't be established.
ECDSA key fingerprint is SHA256:MguYRB0Z95T0LI6cUA3eLU9Xp+RNsn4TZFXzOMsaQVM.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '131.204.14.57' (ECDSA) to the list of known hosts.

szn0080@131.204.14.57's password:

** Authorized uses only. All activity may be monitored and reported. **

szn0080@tux057:~$ cd networkassignment1/
szn0080@tux057:~/networkassignment1$ ls
client11b  client11b.c  client11c  client11c.c  client12  client12.c  server11  server11.c  server12  server12.c
szn0080@tux057:~/networkassignment1$ ./server12 &
[1] 26996
szn0080@tux057:~/networkassignment1$ Server is running...waiting for message requests.
./client12 131.204.14.57

Enter a operator (+,-,x,/): +

Enter a number: 2

Enter a number: 3

String received from the server: Message From Server is : +2351
Answer Calculated is:  2+3 = 5

Enter a operator (+,-,x,/): ^C
szn0080@tux057:~/networkassignment1$ 
```