LAB ASSIGNMENT 1 REPORT

Group Members :

Name : Guttikonda Sri Ram Pavan Kumar

Email : [szg0148@auburn.edu](mailto:szg0148@auburn.edu)

Name : Carter Pierce

Email : [cwp0023@auburn.edu](mailto:cwp0023@auburn.edu)

Lab 1.1: Introduction to Socket Programming :

1. We have created a concurrent UDP server which is listening on port 10010. This server sends back any message you send to it. The code submitted is working as expected.

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

gcc server11.c -o server11

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

./server11 &

Here we are running both server and client on the same machine, so we run server in the background, and we give a & symbol after ./server11 .

Server receives the message from the client and sends the Echo back to the client again.

b) We have created a UDP Client running on the same machine the server is running.

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

gcc client11b.c -o client11b

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

./client11b hostname

where hostname is the name of the machine where we are running the server. For Example, if we are running the server on tux machine 062, the hostname of it will be tux062.eng.auburn.edu. Therefore, the command will be:

./client11b tux062.eng.auburn.edu

After we run it, we can see the ip address of the server being displayed as 131.204.14.062 . Next, we will be prompted to type a message to send to the server. After you type a message and press enter, the message will be sent, and echo will be received from the server. We will have info such as time at which the message is sent and received in microseconds, and we get this using the gettimeofday() function in C. Echo message which is received from the server is also printed .Round Trip Time (RTT) is also printed in microseconds .

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 .

c) We have created a client which does two processes:

1. It will continuously send numbers from 1 to 10000 as string.
2. It will receive the 10000 echoes from the server.

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

gcc client11c.c -o client11c

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

./client11c hostname

where hostname is the name of the machine where we are running the server. For example, if we are running the server on tux machine 062, the hostname of it will be tux062.eng.auburn.edu. Therefore, the command will be:

./client11b tux062.eng.auburn.edu

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.

We get the time using the gettimeofday() function in c .

If there are any missing echoes they will be printed , if there are no missing echoes the missing echoes will not be printed .

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 .

Note : I have defined a global variable called MSIZE in my code currently, and its value is 10000 . If we want to send and receive a string of different numbers like 100 or 100000, we need to change the MSIZE values as 100 or 100000 .

I have taken care of the blocking Issue of the recvfrom() function using setsockopt(). If some of the echo is missed, then recvfrom() acts as a blocking state which is fixed after 3 seconds, and it will pass the control.

Lab 1.2: TCP Calculator :

To compile the code, give the following commands in terminal:

gcc client12.c -o client12

gcc server12.c -o server12

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

./server12 &

Here we are running both server and client on the same machine, so we run server in the background and give a & symbol after ./server11 .

Then, we input the following to run the client :

./client12 ipaddress

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 server on tux machine062, the ipaddress is 131.204.14.62.

The Client then prompts you to input a valid operation (+,-,\*,/), and then input your 2 unsigned integers. Then, the client packages the 3 inputs and sends it to the server (in the provided format of cab) which decodes the message and calculates the result based on the inputted operation and unsigned integers. The server then sends the 14-byte array back (in the provided format of cabrv where r is an unsigned int and v is a 1 or 2 based on if it is valid (1) or invalid (2)). The client prints out the received string and an interpreted one after based on what mathematical operation format is (4+2=6) . If a 1 was at the end, it is valid and if a 2 was at the end, it is an invalid result statement. This program loops for another calculation until Ctrl+C or the program is ended. There was one issue where an input of ‘\’ (instead of division symbol ‘/’) would infinitely loop the code, but it was fixed. 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 .

Note : If inputted values of a and b are not an integer, the input value and if there's a following input value will convert to 0 which will correctly give an operation with both 0 inputs. Please Input a valid operation out of + , - , \* , / only. Please enter unsigned integers only. If you enter input like -2, it will be converted to its equivalent unsigned int value and operation is performed . If you enter a number out of the range of unsigned int such as 567657678968677900449, it will be converted to its equivalent unsigned int value . If we perform an operation like

4294967296 +1 , we will get the answer as = 1 , since 4294967296 equivalent unsigned value is 0 , we have 0+1 = 0.