**EE533 NETWORK PROCESSOR DESIGN & PROGRAMMING  
LAB#1: Familiarity with VM & Sockets**

**Instructor: Prof. Young Cho, PhD**

CREATED AND COMPILED BY:  
SARTHAK JAIN

(MS EE, UNIVERSITY OF SOUTHERN CALIFORNIA)

***GITHUB LINK FOR MY REPOSITORY:***  
You’ll find all the codes and the executables on this repository.   
 <https://github.com/SARTHAK-JAIN-ASIC/EE533/tree/main/LAB1>

**PART 1:**  
In the first part, we first had to make changes to the original code provided.   
 1. We added a few header files (the lack of which was causing compilation errors on my end).  
 2. We changed the error function’s arguments to be const instead of variable since we cannot convert a constant string to char\*   
 3. Changed data type of clilen from int to socklen\_t.

A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated  
A screen shot of a computer

Description automatically generated  
  
A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

***All these changes enabled us to compile the code correctly as seen above.***

**PART 2:**   
Next task was to get the client and the server to ping each other.

For this, I created a NAT network, that would “connect” server and client VMs (as below)  
A screenshot of a computer

Description automatically generated  
  
Then, I created one adapter for server and client (and attached it to the NAT network as above).

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Next step was to make sure the server and the client are communicating. For this, I figured out the IP addresses of the machines and made them ping using the ID addresses. The bottom screenshots show them communicating.

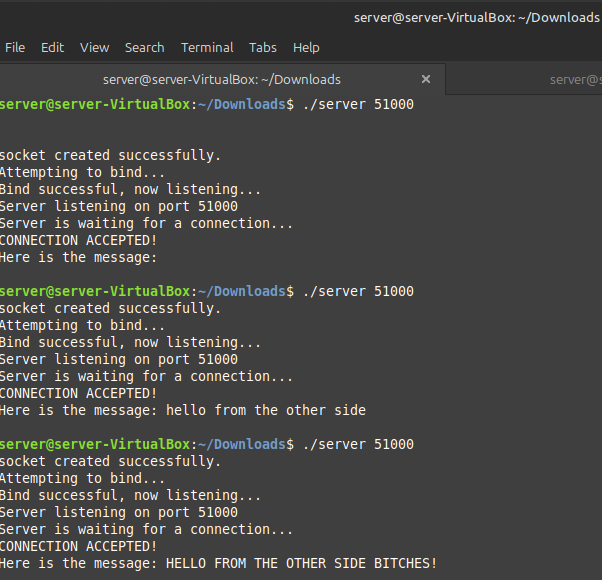
A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

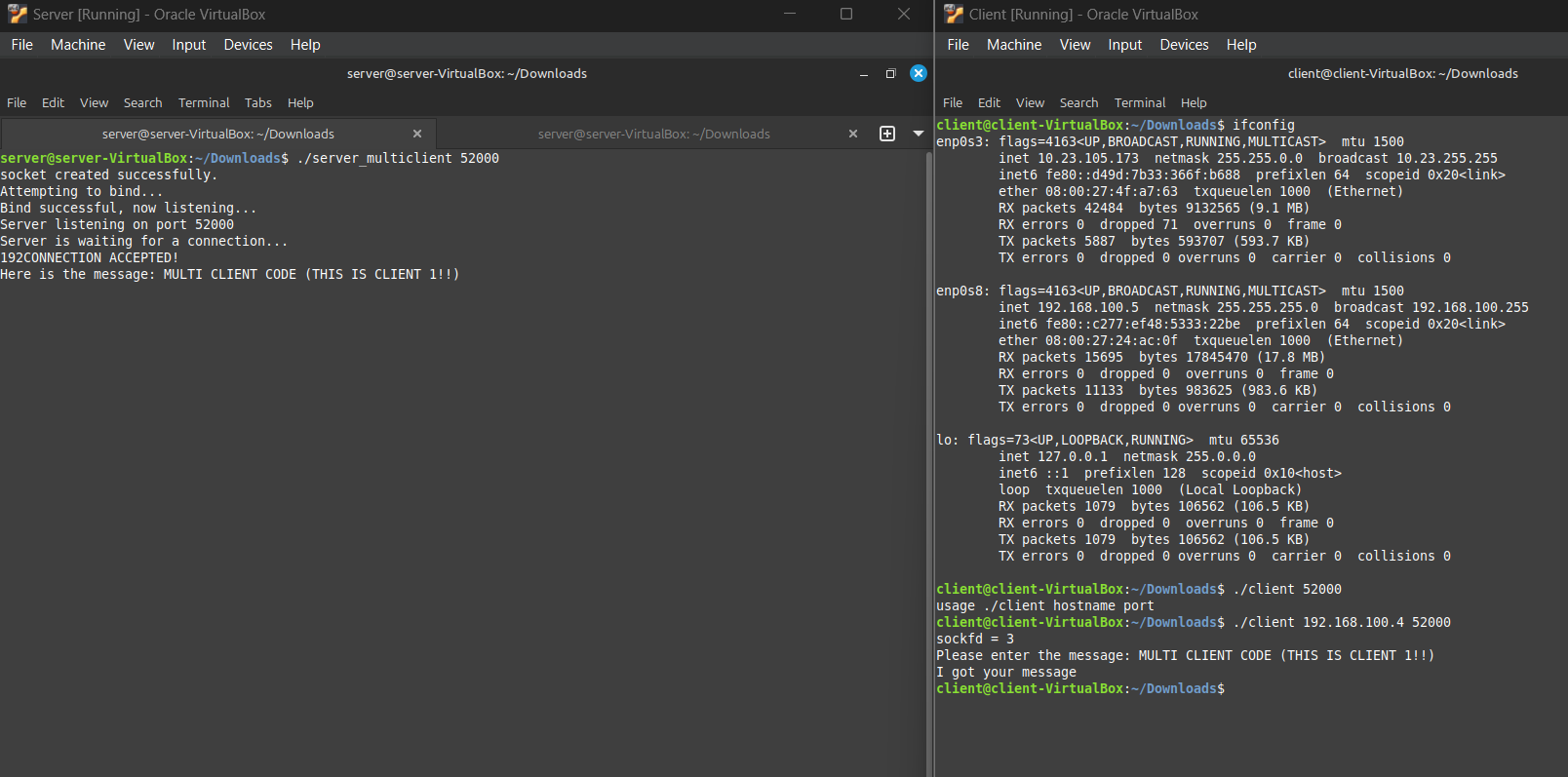
**PART 3:**  
Next step was to actually run the executables, run the server and wait for it to listen after the binding to the address gets done.  
Then we run the client, which basically makes the server go, “CONNECTION ACCEPTED!”. Finally, we send a message to the server by typing it in the client VM and see it live on the server (as shown in the images below).



A screen shot of a computer

Description automatically generated

**PART 4**  
For the multi-client connections, I made changes to the code as asked by the professor in the document and some of my own. Ran the executables as above and see the output. Only this time, the server didn’t drop the connection after receiving message from the client. It was waiting for some other connection to take place (evident by the blinking cursor)

I did not make any changes to the client-side code for this part.  


**PART 5**  
Finally, we needed to run and play around with datagram sockets and UDP connections.

For this we made wholesale changes to the code. After compiling, we ran the code and were FINALLY able to communicate using UDP sockets. See the images below.

A screenshot of a computer

Description automatically generated

**A screenshot of a computer screen

Description automatically generated**

**PART 6**I tried working on the UNIX based server-client system, and I opened two different tabs of the same VM. Ran the server code and then the client code in a different tab(SAME VM) and got the socket to accept connection and respond **A screenshot of a computer

Description automatically generated  
A screenshot of a computer

Description automatically generated  
A screenshot of a computer

Description automatically generated**

**GITHUB COMMIT HISTORY** -  
  
A screenshot of a computer

Description automatically generated