Name: Haobo Yang

USC E-mail: haoboyan@usc.edu

USC ID: 6295-9882-06

Date: January 16, 2025

GitHub Repository Link: https://github.com/HilbertYang/USC-EE533-projects

Youtube Link: EE533 lab1 HaoboYang

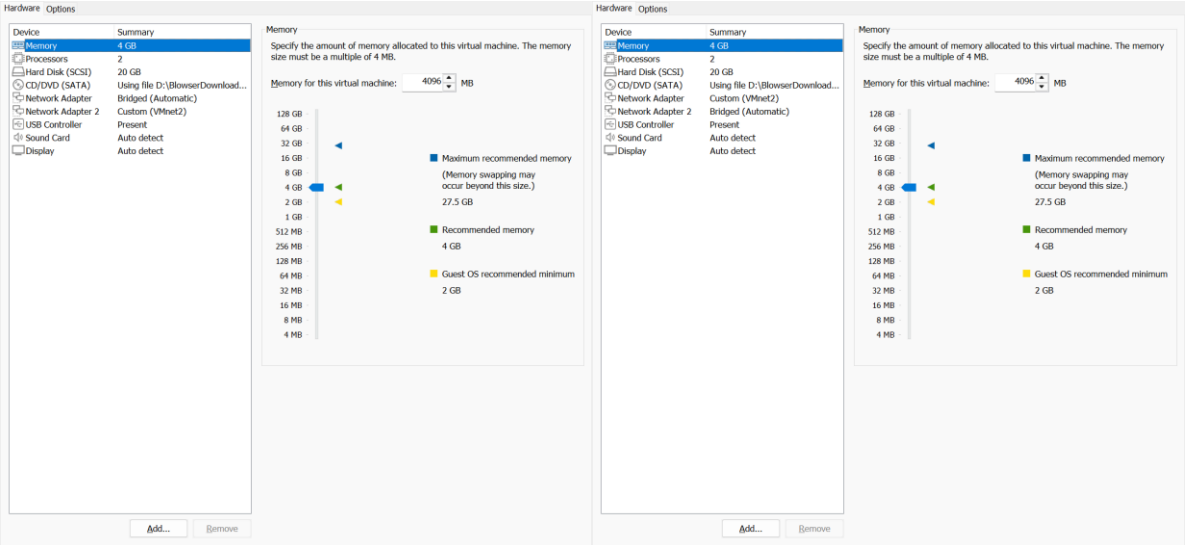
EE533 Lab1 Report

Introduction

In this lab, a client–server communication pattern was implemented using sockets in a Linux environment. Two virtual machines were configured on VMware to enable network communication between a client VM and a server VM. Sample code was used to test message sending, receiving, and replying over a TCP connection. The server was further modified to support multiple client connections, as well as UNIX domain sockets and single-process concurrent servers. This lab helped develop a practical understanding of basic socket programming and client–server communication.

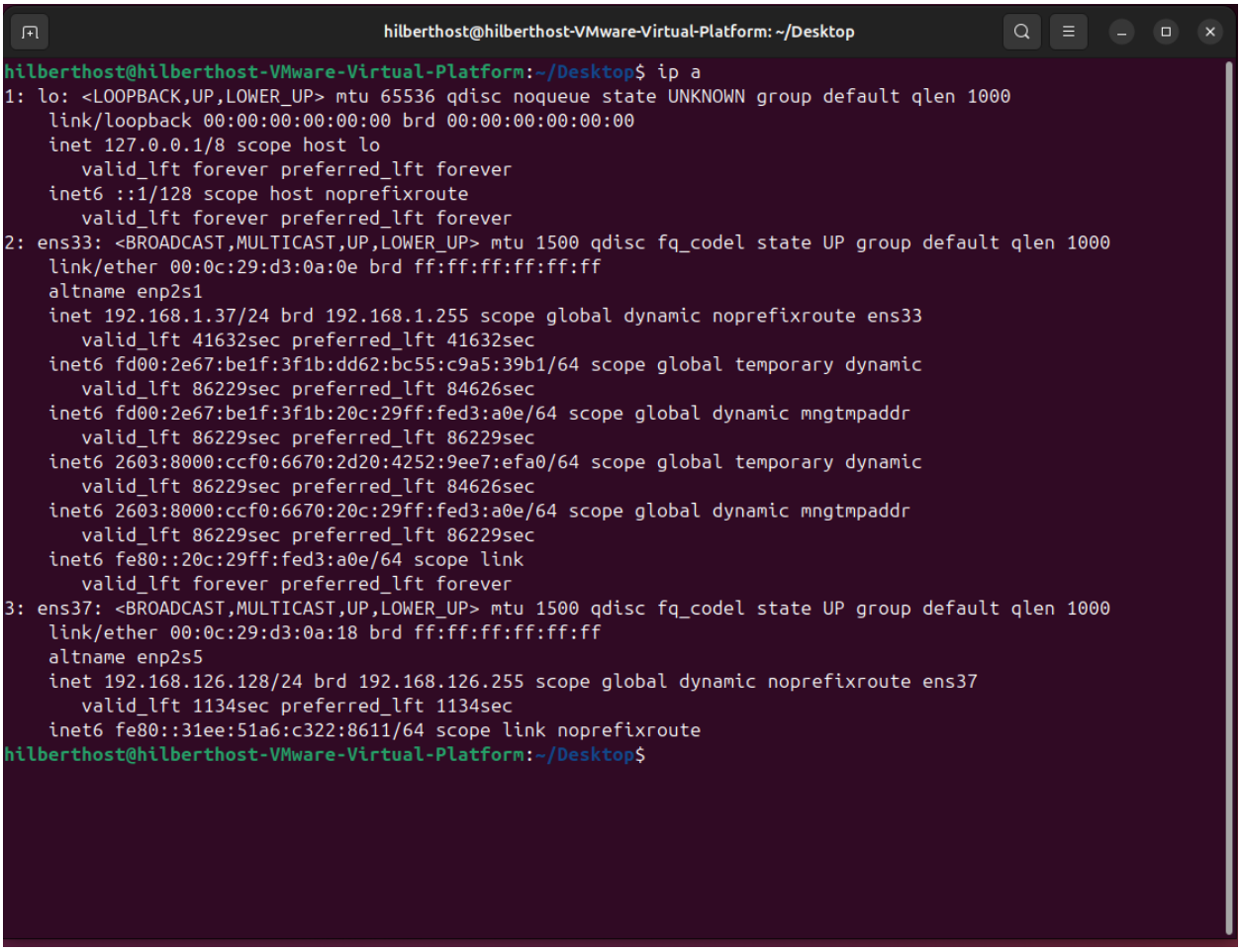
Environment establish

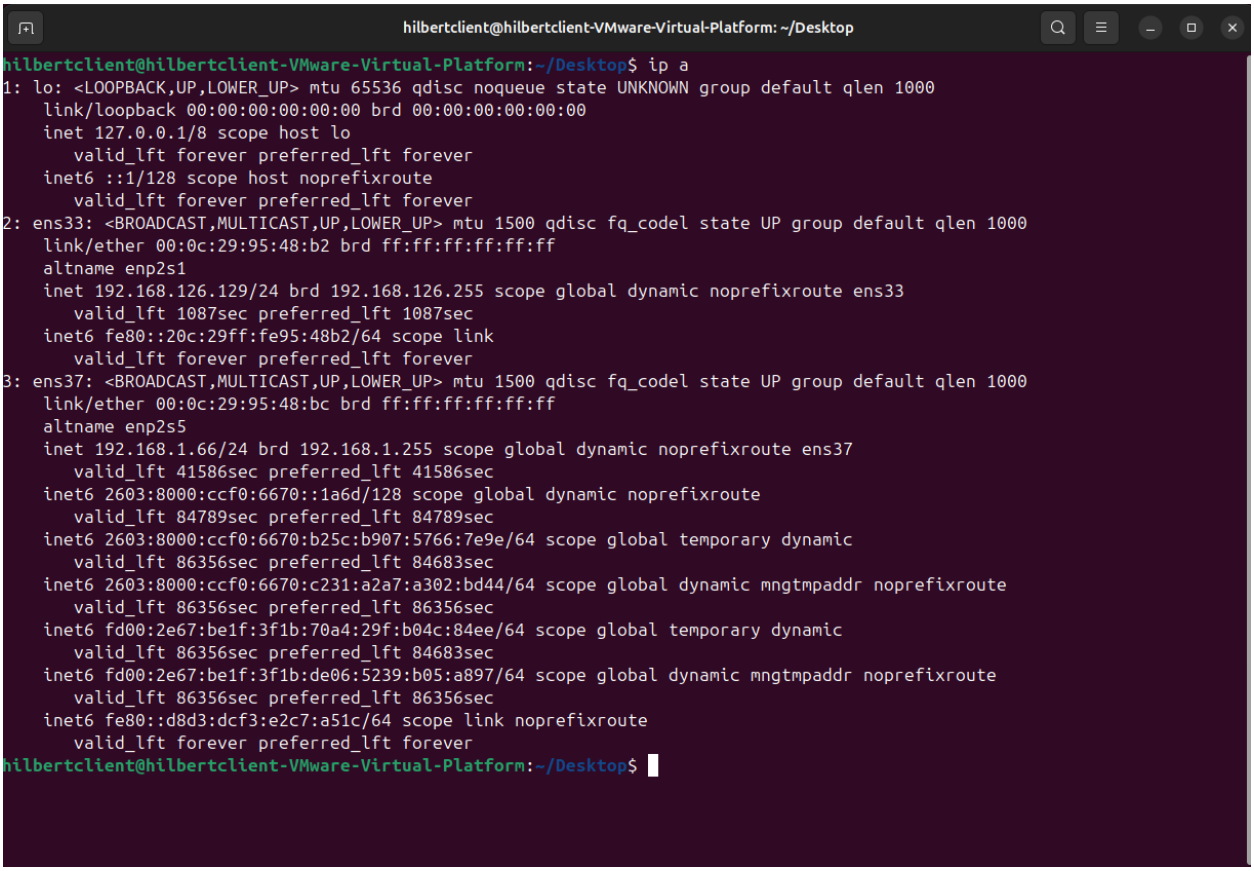
VM Network Setting



The VM1 and VM2 were set to connect bridge network adapter. By doing this, both of them can connect to the internet through the host computer.

Besides that, VM1 and VM2 were also set into a Vnet2, a local network between VMs.



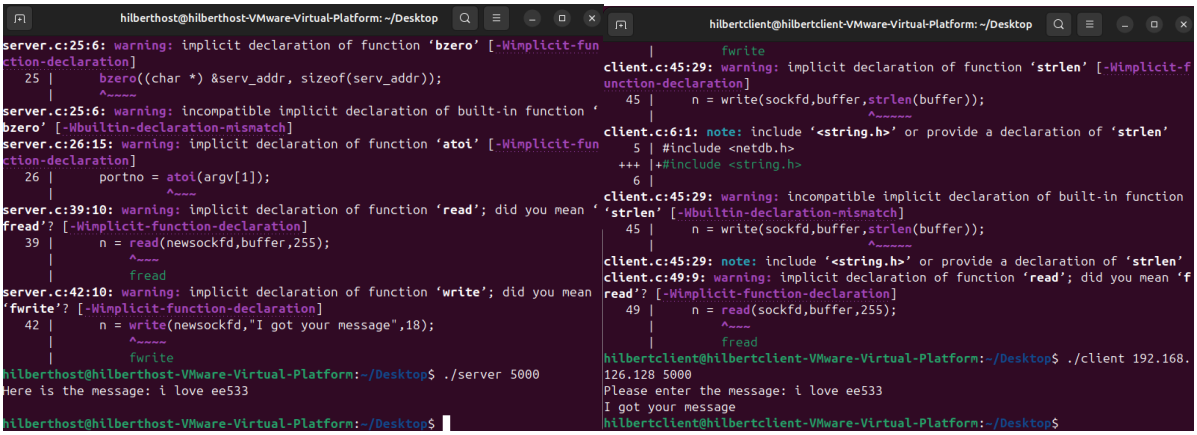


Client-server communication

Sample Test

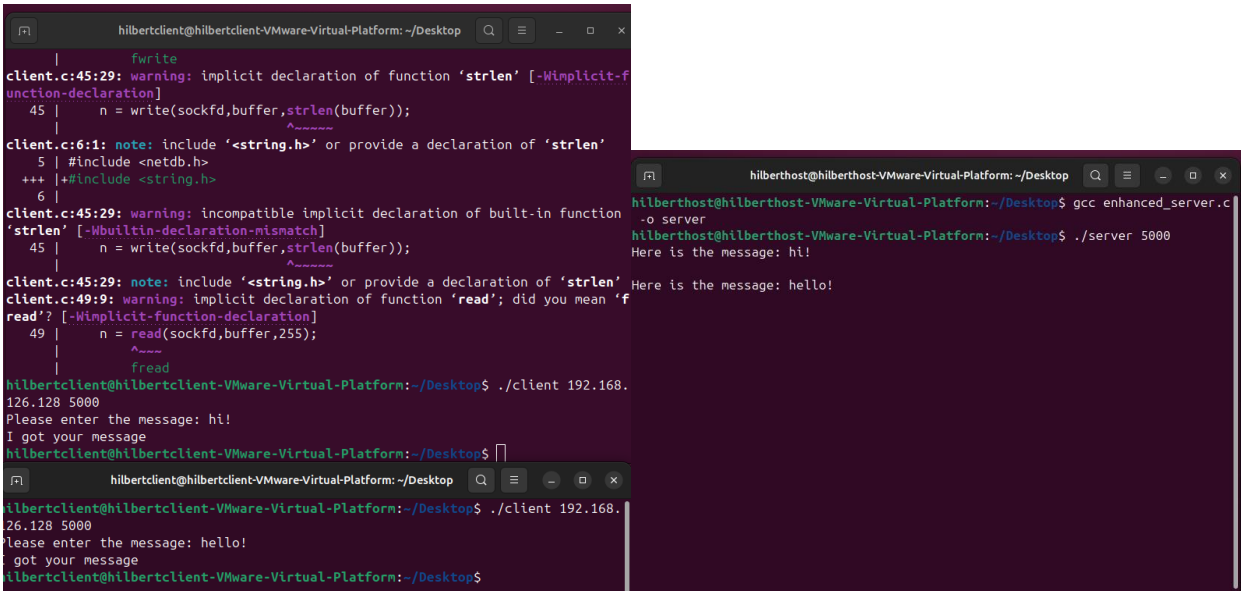
When doing TCP communication, the server use bind(), listen(), accept(), and read()/write().

The client uses the gethostbyname() function to retrieve the IP address from the DNS and uses bcopy to copy the information into the serv_addr. It then establishes the TCP connection using the connection function and performs read()/write() operations.



Adding Process-Based Concurrency Using fork()

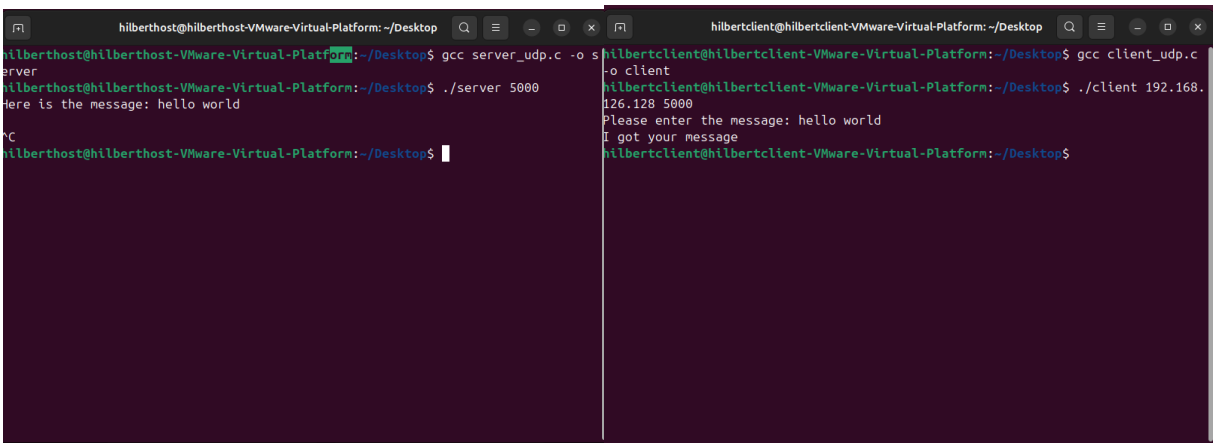
By using a child process, the server can just use the main function to listen to the channel and keep accepting new clients. When it accepts a new client, the host forks a new child process do stuff (), which has read()/write() inside.



UDP

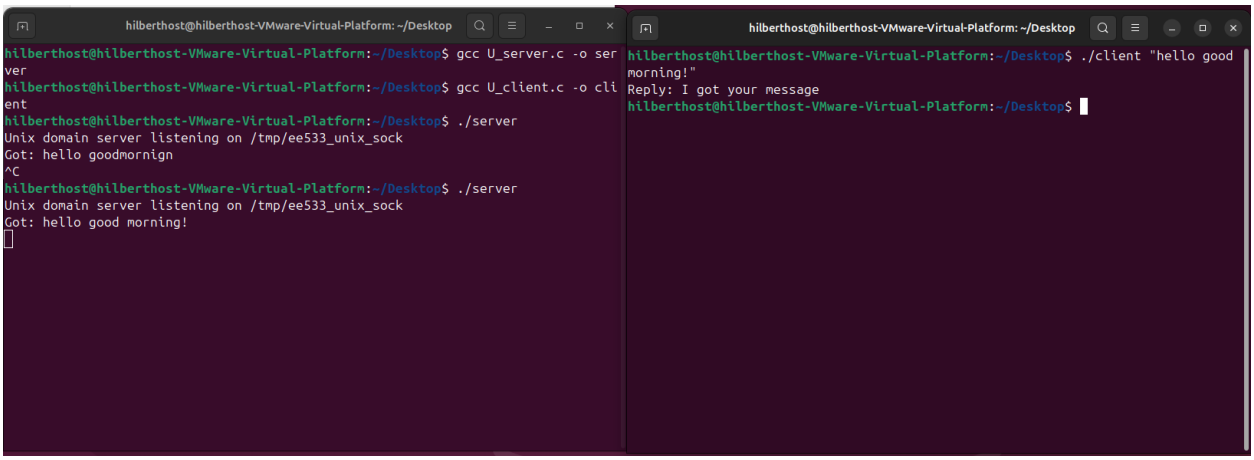
In the UDP protocol, the connections between the server and client are not needed anymore.

Instead of connction(), read(), and write, UDP protocal use sendto() and recvfrom()



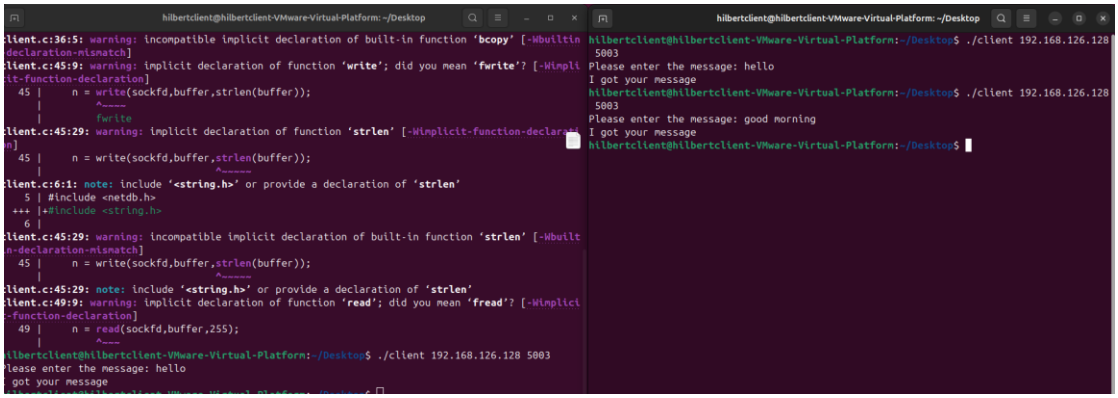
UNIX Domain Socket Implementation

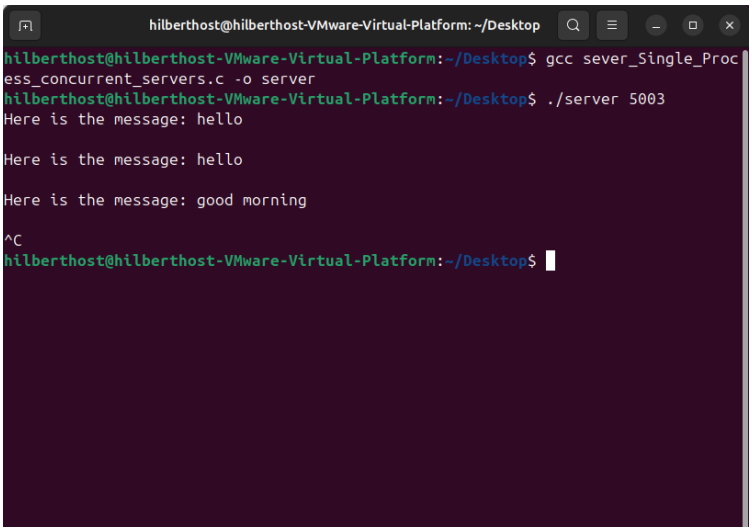
Using the UNIX Domain socket, we can directly use the file path as the address of the server



Single-process Concurrent Server

In a single-process concurrent server, there is no fork. Server use only one process to catch the connection. However, it does it in kind of selection.




hilberthost@hilberthost-VMware-Virtual-Platform: ~/Desktop
hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$ gcc sever_Single_Process_concurrent_servers.c -o server
hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$ ./server 5003
Here is the message: hello
Here is the message: hello
Here is the message: good morning
^C
hilberthost@hilberthost-VMware-Virtual-Platform:~/Desktop$