Liu Yongxiang
519021910029
sam.liu@sjtu.edu.cn

Computer Network Lab3 Report

2021-10-16

1 Introduction

This lab requires us to implement a client server model under TCP protocal and run the model on mininet under a star-like topology. Based on this, we may also implement a p2p model optionally.

2 Process of C/S Model under TCP

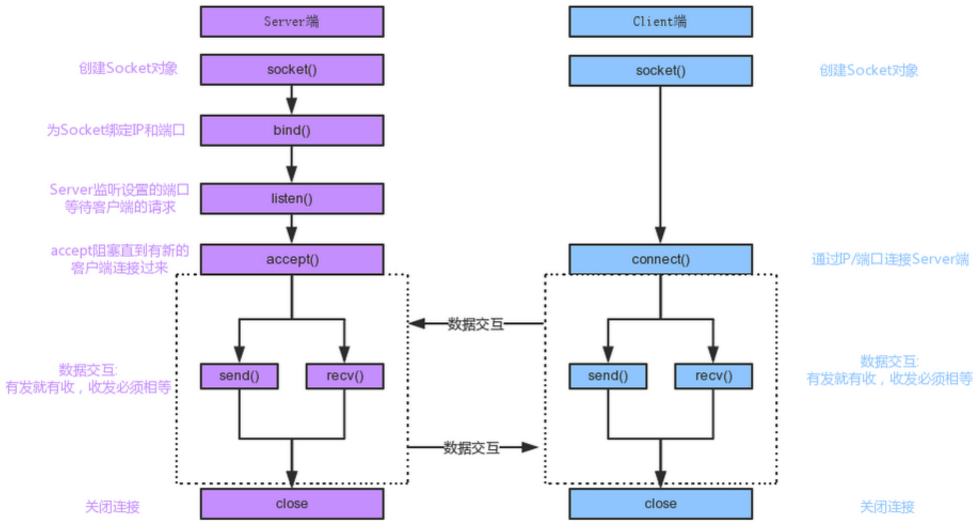


Figure 1: Process of TCP

Figure 1 shows the main process of client server model under TCP protocol.

From the aspect of server, it needs to firstly create a socket, bind( ) the socket to a certain IP address and Port. Then the socket needs to listen( ) for the request from client. Then the server runs accept( ) to block the program until some clients try to connect to the server.

From the aspect of client, it is much more simple. It only needs to create a socket for itself to connect to the server. Then it runs connect( ) to try to connect to the server. If the server is not ready, the connection will not succeed.

When the client and server are connected successfully, they can interact with each other by using send( ) and recv( ) and transmit the data.

3 C/S Model in C++

Referring to the documents provided, I tried to implement a client server model in C++ first. The codes given have already set the connection between client and server so we only need to add the function of send( ) and recv( ) and then implement it on mininet.

3.1 Send and Receive Data

Before I start to deal with the data, I need to create a data firstly. According to the document provided, I used the command /dev/urandom to create a 10MB file and purpose to send it.

The data is saved in server and sent to client. Therefore, the server uses send( ) and the client uses recv( ). Notice that the server cannot use the socket created for connection to send the data, but to use the socket return from accept( ) to send the data. Besides, the client needs to set a buffer size to limit the data received once.

3.2 Run on Mininet

To run the client and server model on mininet, we need to decide which host to be the server. It is important since the client needs the host's IP address to connect to the server. In the code used above, the server's IP is '127.0.0.1' and client can only connect to itself but not others.
To simplify the question, I suppose that the model always uses the host firstly created to be the server. I use ifconfig command to check its IP and get the result of '10.0.0.1'.

3.3 Problem I met

When I tried to send a file filled with NULL, I succeeded. However, when I tried to send the file created by /dev/urandom, I failed since it cannot decode utf-8 to bytes even if I used 'rb' and 'wb' as the arguments in open. Therefore, I decided to restart the program by python.

4 C/S Model in Python

The main process is almost the same with C++. The problem I met in C++ did not happen this time.
Furthermore, I need to implement a multi-thread program for server.
Searching for some information on the Internet, I have learned that there is a library called threading that can provide multi-thread programming. Therefore, I create a thread for each client as soon as the server has accepted a request. The thread's function is to send data to the clients.

5 P2P Model in Python

Having implemented C/S model in python, I further implement the P2P model in python as well. The key difference between C/S model and P2P model is that in C/S model, clients can only receive data sent from the server. However, in P2P model, clients can also send data to each other. For this reason, P2P model can reach a better performance in data transferring compared to C/S model.
In P2P model, server segments the data into many pieces and sends each piece to a client. Later on, the clients will send their own pieces to others and combine all the pieces into a complete data.

5.1 Data Segmentation

At first, I ask the server to read all the data and count the number of lines' in the txt data. Then I tell the server how many clients are waiting for connection and the server will cut the data into the exact number of chunks.
Then the server will send each chunk to each of the client and the client could save the chunk locally and temporally.

5.2 Interaction between Clients

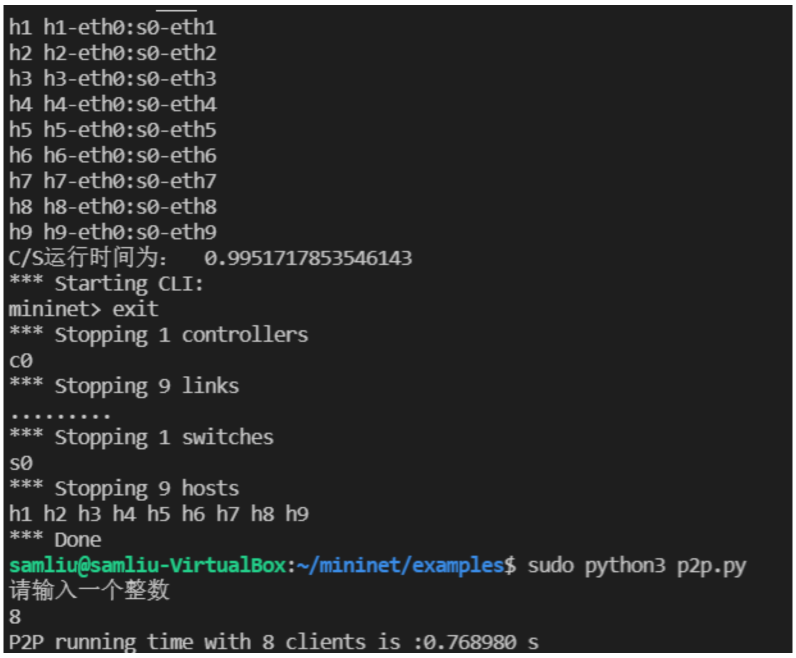
This is the most significant part in implementing P2P model !
Before we start to code, we need to think of how it works clearly. In P2P model, each client can also be the server for other clients. Therefore, I need to tell the clients that receive data the IP address of other clients that provide data. Besides, I need to ask the clients to be client and server at the same time. Otherwise, the clients cannot provide data for each other.
Hence, I wrote two different thread for the clients to run, one for being server while the other for being client. The server thread will finish when it successfully send data to all clients and the client thread will end when it receive all data from other clients. I used an integer to count the number of clients server thread send and the number of chunks that client thread receive. Then I start the two threads at the same time and implement the P2P model successfully.
Since the IP address of the hosts created by mininet is continuous, it is easy for each client thread to find other clients' IP address. The client thread needs to connect with other clients in a certain order to ensure the correctness of data. While the server thread can send the chunk of data to whatever clients connects.

6 Run My Code

You can run C/S model by typing "sudo python3 client_server.py" and run P2P model by typing "sudo python3 p2p.py"

To run my code, you only need to give an integer to set the number of clients. Then the code will do all the work automatically and return the running time.
Since the time is limited, my P2P model is not that complete and robust and may not always finish the job. Maybe need to try for several times.

7 Result and Evaluation



```
h1 h1-eth0:s0-eth1
h2 h2-eth0:s0-eth2
h3 h3-eth0:s0-eth3
h4 h4-eth0:s0-eth4
h5 h5-eth0:s0-eth5
h6 h6-eth0:s0-eth6
h7 h7-eth0:s0-eth7
h8 h8-eth0:s0-eth8
h9 h9-eth0:s0-eth9
C/S运行时间为:   0.9951717853546143
*** Starting CLI:
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 9 links
.........
*** Stopping 1 switches
s0
*** Stopping 9 hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Done
samliu@samliu-VirtualBox:~/mininet/examples$ sudo python3 p2p.py
请输入一个整数
8
P2P running time with 8 clients is :0.768980 s
```

Figure 2: Running Result Example

Figure 2 is an example of running C/S model and P2P model when the number of clients is 8. I run the C/S model and P2P model with clients 2,4,6 8,10,12 and record the running time. I run each number three times and select the least running time of them. Figure3 is the least running time graph.

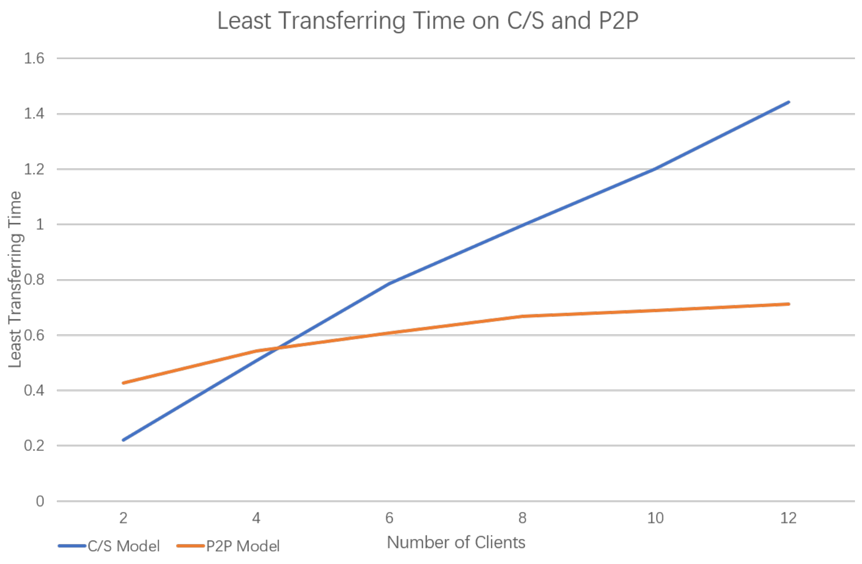


Figure 3: Running result of C/S and P2P

When the client number is two, the running time of P2P is longer than C/S. That is because in my program, P2P need to accept the chunk first and send the chunk to the client. In that case, when the number of client is two, the data would be sent three times.
When the number of clients become bigger, the least transferring time of P2P is much shorter than that of C/S model.

8 Conclusion

After completing this lab, I have got a further understanding in C/S model and P2P model. Besides, I have learnt how to create TCP connection both in C++ and python. To sum up, it is a quite fruitful experience for me.