ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

Laboratory Report of Computer Networks

Assignment II. Socket Programming

Name: TIAN Peilin

No.: 517261910018

Date: 2020/10/17

Score: _____

SHANGHAI JIAOTONG UNIVERSITY

SJTU-ParisTech Elite Institute of Technology

Contents

1 Introduction

This experiment mainly aims at the implementation of a simple file share application using TCP Socket APIs. In this experiment, **Python** is used to realize a C/S model and P2P model. Besides, **Mininet** is used to compare the overall file downloading time of the two models, and we will study how the number of downloading time changes with respect to the number of peers. In this experiment, we focus on a network of star topology with one host as a server, and the other hosts as peers requesting files, as is shown in Figure 1.

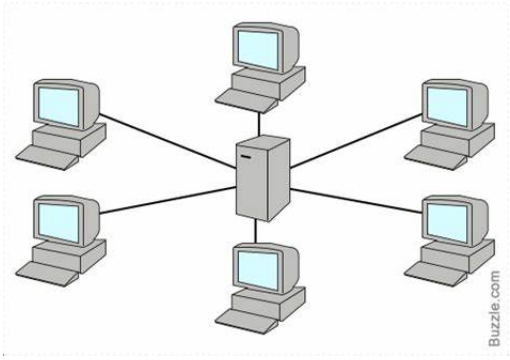

Figure 1 Star topology of network studied in this experiment

2 Study of the C/S Model

2.1 Description of the implementation

For the client/server model, we focus on the transmission of a file from a server host to several client hosts. For this, we create a socket listening in the server that waits for connections from different clients. Once a client is connected, we send the file from the server to the client. We notice that multiple clients could request the same file from the server, thus we will create a thread of sending the file for each request of a client. For each client, after being connected to the server, we let it receive the file and save the file to its local directory. Finally, we use `time.time()` in Python to calculate the running time of downloading the file.

2.2 Experiments and results

The C/S model is realized in the folder **c_s**. Files in this folder are explained as follows:

- **server.py:** the server that sends files to clients is implemented
- **client.py:** the client that downloads files from the server is implemented
- **topo.py:** a network of star topology is implemented with Mininet for the experiments
- **utils.py:** some functions and variables used by both the server and the client are defined

- **src:** containing a file **test** of the server needed to be sent to the clients
- **dst:** containing the local directories of clients to save the file **test** from the server
- **log:** outputs of server and clients will be saved in this folder while the program is executed

For this experiment, an environment **Linux** with **Python3** and **Mininet** is needed.

To run the program, we can go to the folder **c_s** and enter [`sudo python topo.py`] in the terminal. The execution of program will last 120 seconds. Then, in the folder **dst**, we can enter [`ls -R`] in the terminal to check if all clients have downloaded the file **test** from the server. The result of the downloaded files is shown as follows in Figure 2.

```
andy@andy-virtual-machine:~/文档/tpl/socket/assign/c_s/dst$ ls -R
.:
data1  data2  data3  data4  data5

./data1:
test

./data2:
test

./data3:
test

./data4:
test

./data5:
test
```

Figure 2 Files downloaded in the local directory of each client

We notice that all files have been well saved in the corresponding directories. Now we compare the downloaded files with the original file of the server. For this, we can go to the local directory of each client, for example, the folder **data1**, and enter [`md5sum test`] to calculate the MD5 code of the file **test**. The result is shown in Figure 3. We can continue to check other downloaded files in the same way.

```
andy@andy-virtual-machine:~/文档/tpl/socket/assign/c_s/dst$ cd data1
andy@andy-virtual-machine:~/文档/tpl/socket/assign/c_s/dst/data1$ md5sum test
8bfe3258508dc04b6228998c11790893  test
```

Figure 3 MD5 code of the downloaded file

Then we calculate the MD5 code of the original **test** file of the server. In the folder **src**, we enter [`md5sum test`] again to get the result, which is shown in Figure 4.

```
andy@andy-virtual-machine:~/文档/tpl/socket/assign/c_s$ cd src
andy@andy-virtual-machine:~/文档/tpl/socket/assign/c_s/src$ md5sum test
8bfe3258508dc04b6228998c11790893  test
```

Figure 4 MD5 code of the original file

We notice that the MD5 code of the downloaded file is the same as the original file, so we can conclude that the downloading of file is successful.

Then we move to the folder **log** to check the outputs of the program. For the server, the outputs have been saved in the file **server**. We can see which file is sent to the clients, the information of the connected clients, and the information indicating the ends of transmissions. The result is shown in Figure 5.

```
./src/test
* Server started, waiting for client to connect...
Got connection from ('10.0.0.3', 52742)
Got connection from ('10.0.0.5', 43930)
Got connection from ('10.0.0.4', 50426)
Got connection from ('10.0.0.2', 40408)
Got connection from ('10.0.0.6', 33636)
Transmission completed!

Transmission completed!

Transmission completed!

Transmission completed!

Transmission completed!
```

Figure 5 Output of the server

For each client, for example, the first one, has its output saved in the file **h1**. In this file, we can see the download path of the file, the length of the file, the indication for the end of downloading and the time of the downloading process. The result is shown in Figure 6.

```
Download file as: ./dst/data1/test
* Downloading file from the server:
Length of the file: 10816961
Downloading finished!

* Running time: 45.156553506851196
```

Figure 6 Output of the client

Now we calculate the average downloading time of the clients with the time of each client shown in Table 1, and we get the average time in the C/S model which is around 43s. Considering the length of file downloaded which is 10.8MB, we can get the average downloading speed 251KB/s.

Client	**1**	**2**	**3**	**4**	**5**	**Average time (s)**
Time (s)	45.16	42.04	42.49	45.34	40.16	43.038

Table 1 Downloading time of clients in C/S model

3 Study of the P2P Model

3.1 Description of the implementation

For the P2P model, we focus on the same task of the transmission of file as the C/S model. In this case, the server will send a different part of the file to different peers, then each peer will distribute its part of file to all the other peers. The implementation of server in this model is similar to the C/S model, but this time, we let it send part of the file to each peer according to the order in which the peers are connected. For each peer, we create firstly a thread to start a socket waiting for connections from other peers, then we let it receive the corresponding part of file from the server in the same way as the C/S model. Once the part of file is downloaded, we create a new thread with several subthreads to connect to each other peer. When the current peer is connected to another peer, we let it receive the part of file from the peer; when the current peer receives the connection from another peer, we create a new thread to send its part of file to the peer. Finally, after a peer has received all parts of the file, it will combine them to get the original file.

3.2 Experiments and results

The P2P model is realized in the folder **p2p**. Files in this folder are explained as follows:

- **server.py:** the server that sends parts of the file to peers is implemented
- **peer.py:** the peer that downloads the file from the server is implemented
- **topo.py:** a network of star topology is implemented with Mininet for the experiments
- **utils.py:** some functions and variables used by both the server and the peer are defined
- **src:** containing a file **test** of the server needed to be sent to the peers
- **dst:** containing the local directories of peers to save the file **test** from the server
- **log:** outputs of server and peers will be saved in this folder while the program is executed

The environment needed for this experiment is the same as the C/S model. To run the program, we can go to the folder **p2p** and enter [`sudo python topo.py`] in the terminal. The execution of program will last 120 seconds. Then, in the folder **dst**, we can enter [`ls -R`] in the terminal to check if all peers have downloaded the file **test** from the server. The result of the downloaded files is shown as follows in Figure 7.

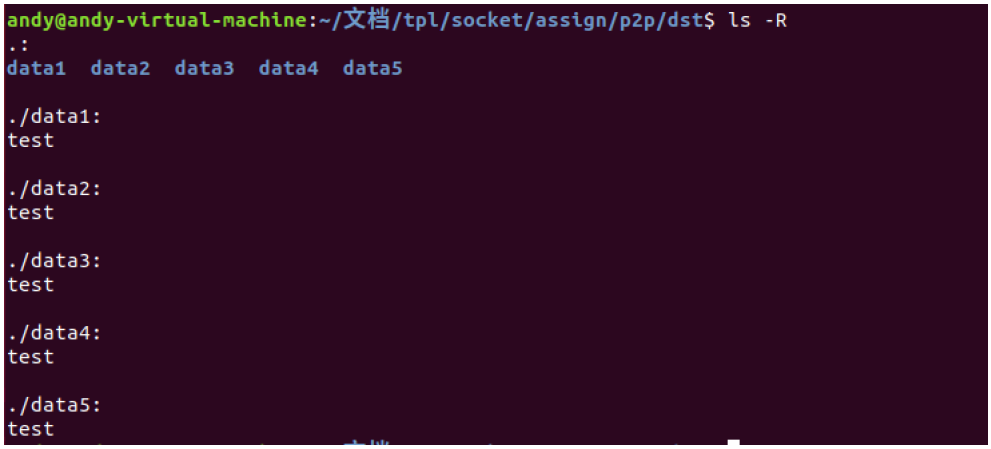

Figure 7 Files downloaded in the local directory of each peer

We notice that all files have been well saved in the corresponding directories. Now we compare the downloaded files with the original file of the server. For this, we can go to the local directory of each peer, for example, the folder **data1**, and enter [`md5sum test`] to calculate the MD5 code of the file **test**. The result is shown in Figure 8. We can continue to check other downloaded files in the same way.

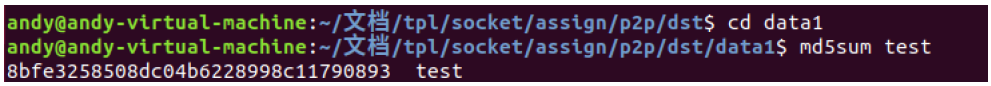

Figure 8 MD5 code of the downloaded file

Then we calculate the MD5 code of the original **test** file of the server. In the folder **src**, we enter [`md5sum test`] again to get the result, which is shown in Figure 9.

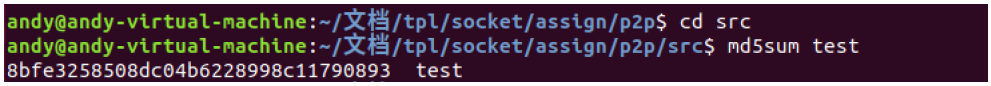

Figure 9 MD5 code of the original file

We notice that the MD5 code of the downloaded file is the same as the original file, so we can conclude that the downloading of file is successful.

Then we move to the folder **log** to check the outputs of the program. For the server, the outputs have been saved in the file **server**. We can see which file is sent to the peers, the information of the connected peers, and the information indicating the ends of transmissions. The result is shown in Figure 10.

```
./src/test 5
Server started, waiting for client to connect...
Got connection from (10.0.0.3, 53690) at index 0
Got connection from (10.0.0.5, 44878) at index 1
Got connection from (10.0.0.6, 34580) at index 2
Got connection from (10.0.0.2, 41356) at index 3
Got connection from (10.0.0.4, 51378) at index 4
Transmission of block 3 completed!

Transmission of block 4 completed!

Transmission of block 1 completed!

Transmission of block 0 completed!

Transmission of block 2 completed!
```

Figure 10 Output of the server

For each peer, for example, the first one, has its output saved in the file **h1**. In this file, we can see the download path of the file, the information of the peers, the processes of receiving a part of file from the server, of sending the part to other peers and of downloading other parts of file from other peers. The result is shown in Figure 11.

```
Download file as: ./dst/data1/test
Peer ip: 10.0.0.2
Peer port: 2681

* Getting a block from the server:
* Peer started, waiting for other peers to connect...
Index of current peer: 3
Number of peers: 5
Information of peers: [('10.0.0.3', 2682), ('10.0.0.5', 2684), ('10.0.0.6', 2685),
('10.0.0.2', 2681), ('10.0.0.4', 2683)]
Length of the file: 10816961
Name of the block: ./dst/data1/3
Length of the block: 2163393
Got connection from ('10.0.0.3', 41540)
Downloading of block finished!

* Asking other peers for blocks...
Asking peer at index 0 for a block
Asking peer at index 1 for a block
Asking peer at index 2 for a block
Asking peer at index 4 for a block
Got connection from ('10.0.0.6', 58324)
Got connection from ('10.0.0.5', 53748)
Got connection from ('10.0.0.4', 49244)
Transmission of block 3 to another peer completed!

Downloading of block 4 finished!

Transmission of block 3 to another peer completed!

Transmission of block 3 to another peer completed!

Downloading of block 0 finished!

Transmission of block 3 to another peer completed!

Downloading of block 2 finished!

Downloading of block 1 finished!

* Combining the blocks to get the file
* Run time: 17.42712116241455
```

Figure 11 Output of the peer

Now we calculate the average downloading time of the peers with the time of each peer shown in Table 2, and we get the average time in the P2P model which is around 18s. Considering the length of file downloaded which is 10.8MB, we can get the average downloading speed 616KB/s.

Client	**1**	**2**	**3**	**4**	**5**	**Average time (s)**
Time (s)	17.43	17.46	17.83	17.35	17.73	17.56

Table 2 Downloading time of peers in P2P model

4 Study of the downloading speed

In this section, we will study the influence of the number of clients/peers on the speed of downloading the file from the server. For this, we execute 4 experiments while setting the number of clients/peers as 3, 5, 8 and 12. For each experiment, we calculate the average downloading time according to the outputs of clients/peers, then we calculate the average downloading speed of a client/peer. The results of the C/S model are shown in Table 3, and the results of the P2P model are shown in Table 4.

Peer	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**Average time (s)**	**Average speed (KB/s)**
Time (s)	27.22	26.63	26.79										26.88	402.42
	45.16	42.04	42.49	45.34	40.16								43.04	251.34
	72.51	71.78	59.12	71.79	69.89	65.39	67.47	62.3					67.53	160.18
	99.6	107.61	90.24	96.39	97.27	97.43	104.38	104.03	95.47	100.24	104.12	108.82	100.47	107.67

Table 3 Downloading time and speed of clients in C/S model

Peer	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**Average time (s)**	**Average speed (KB/s)**
Time (s)	16.73	16.71	16.27										16.57	652.81
	17.43	17.46	17.83	17.35	17.73								17.56	616.00
	18.76	18.44	18.44	19.07	18.82	18.72	18.27	19.41					18.74	577.18
	25.39	19.23	25.93	24.42	25.88	24.31	25.12	23.09	25.93	25.51	25.5	25.15	24.62	439.33

Table 4 Downloading time and speed of peers in P2P model

Then we draw a figure based on the experimental data in Figure 12.

For the C/S model, we notice that the downloading speed is inversely proportional to the number of clients, which is reasonable because of the limitation of bandwidth of links in the network. For the P2P model, the downloading speed will be theoretically faster with the increasing number of peers. However, in this environment of experiments, we also notice a decrease of the speed. Several factors could lead to this result. For example, the performance of CPU may become the bottleneck in a simulated network with a relatively large bandwidth. Besides, the synchronization of a large number of peers may also decrease the overall process of downloading the file.

With the comparison between these two models, we can notice that the P2P model is markedly faster than the C/S model, and the proportion of speed between these two models becomes

larger as the number of clients/peers increases. This result has well indicated the advantage of P2P for a larger speed.

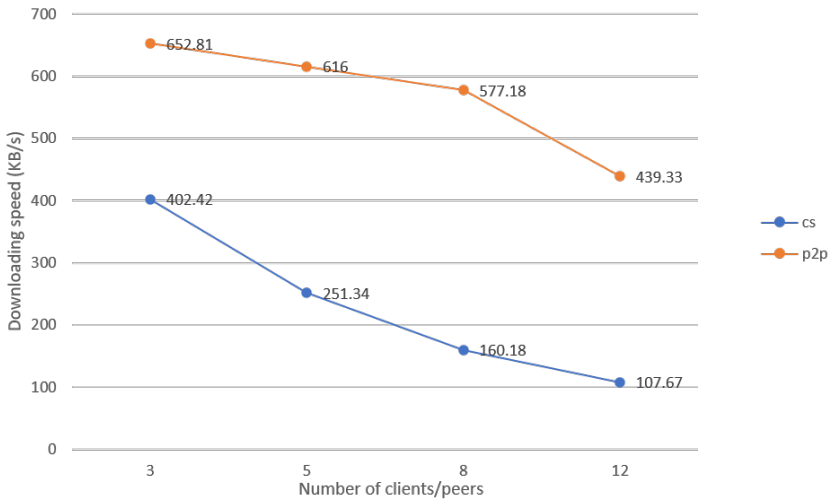

Figure 12 Downloading speed influenced by the number of clients/peers

5 Summary

In this experiment, the Client/Server model and P2P model of file transmission have been realized and tested in a simulated environment. We have verified the feasibility of these two models, and their performances have been tested and compared with different experiments.

Generally speaking, The P2P model has a better performance than the C/S model with regard to the speed of transmission. As the number of clients or peers becomes larger, he decrease of speed in the C/S model has been verified, but the increase of speed in the P2P model is not presented because of some possible factors of the simulated environment, which requires more study and time in the future to locate the problem and to optimize the algorithms.