CS339 Lab Report - Socket Programming

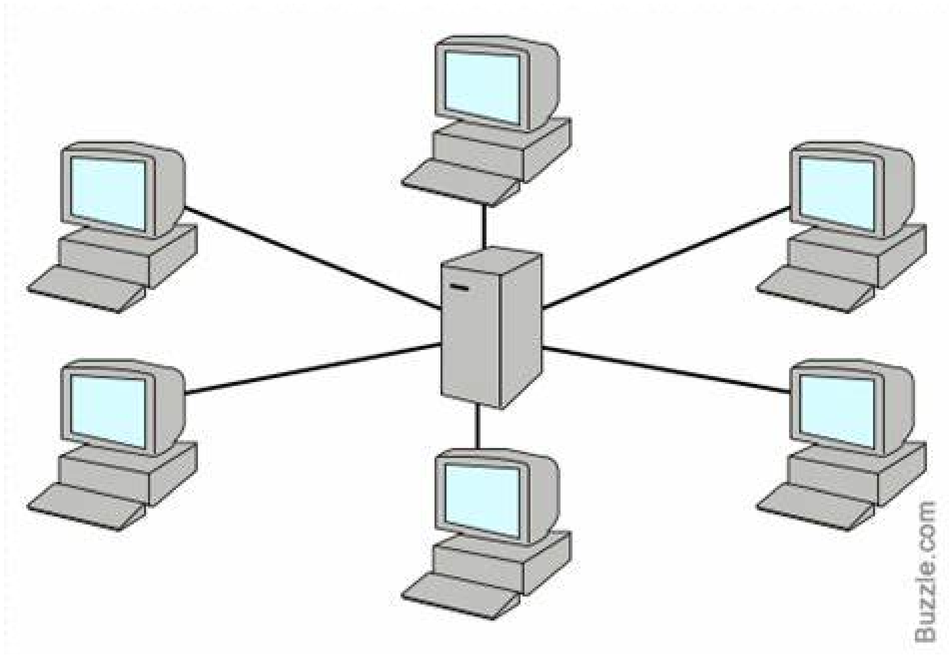
Student Name: Zhou Haoquan

Student ID: 519370910059

Client-Server Model

Topology Construction

The topology of this lab is a star-shaped network, as shown in the following figure



The switch is at the center while six hosts are connected to the switch. Among the hosts, one of them is a server while the others are clients. The detailed code for the topology is shown as follows:

```python
class simpleTopo(Topo):
    def build(self, **_opts):
        s1 = self.addSwitch('s1')
        h1 = self.addHost('h1', ip = '10.0.0.1')
        h2 = self.addHost('h2', ip = '10.0.0.2')
        h3 = self.addHost('h3', ip = '10.0.0.3')
        h4 = self.addHost('h4', ip = '10.0.0.4')
        h5 = self.addHost('h5', ip = '10.0.0.5')
        h6 = self.addHost('h6', ip = '10.0.0.6')
        self.addLink(h1,s1, bw = 100)
        self.addLink(h2,s1, bw = 100)
        self.addLink(h3,s1, bw = 100)
        self.addLink(h4,s1, bw = 100)
        self.addLink(h5,s1, bw = 100)
        self.addLink(h6,s1, bw = 100)
```

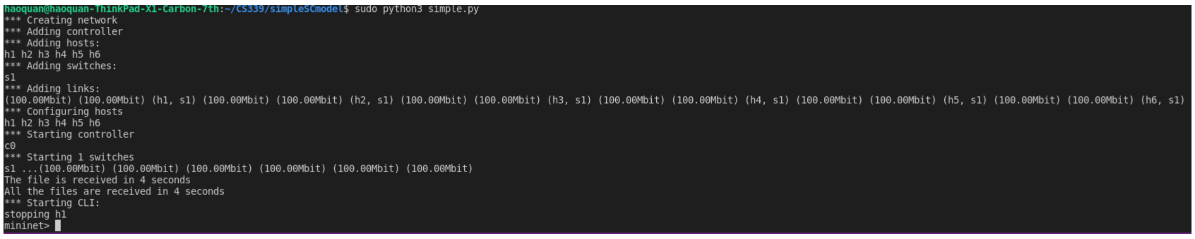
Simple Test

In this test, we want to test a simple case that only one client and one server is involved in the communication. In other word, the server will only provide service to that client. This should give us the transmission rate for a single user. The `server.py` and `client.py` are shown as follows, respectively:

```
from socket import *

#define the server port, we do not need to define the server name
serverPort = 12000

#define the buffersize
buffersize = 1024

#create the server socket and then bind it to this port
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))

#make this socket listen to 1 client
serverSocket.listen(1)

#initialze the number of clinets to be zero
num_of_clients = 0

#now we have done the preparation and waiting for the clients
print('The server is ready to receive')

while True:
    #when the serversocket accept some connection request, it create a new socket
together with the address of the client
    connectSocket, addr = serverSocket.accept()

    #then it read in the sentence and check what sentence it is
    #the first sentence should be the request for the client number
    sentence = connectSocket.recv(buffersize).decode()
    if sentence == 'Please give me a client number':
        num_of_clients = num_of_clients + 1
        send_back_message = '%d'%num_of_clients
        connectSocket.sendall(send_back_message.encode())
    else:
        print('Receive a wrong sentence')

    #the second sentence should be the request for file
    sentence = connectSocket.recv(buffersize).decode()
    if sentence == 'Please give me the file':
        fs = open('test.txt')
        while True:
            buffer = fs.read(buffersize)
            if not buffer:
                break
            connectSocket.sendall(buffer.encode())
        connectSocket.sendall(''.encode())
        fs.close()

    print("The file has already been sent to ", addr)
    #after down the connection, it shut down this connection and ready for next
connection
    connectSocket.close()
```

```
from socket import *
import time

#define the name and port of server of this client
serverName = '10.0.0.1'
serverPort = 12000
buffersize = 1024

#create a TCP socket and then connect this socket to the server
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))

#after build this socket, we make it sent message to the server, request for a
client number
clientSocket.sendall('Please give me a client number'.encode())
client_number = clientSocket.recv(buffersize).decode()

#now we create a file for this host according to the client number it received
#and then it sends a message for transmitting the file
filename = "file%c"%client_number
clientSocket.sendall('Please give me the file'.encode())

#it then begins to receive the file
start = time.time()
fs = open(filename,'wb')
while True:
    buffer = clientSocket.recv(buffersize)
    if not len(buffer):
        break
    fs.write(buffer)
fs.flush()
fs.close()

finish = time.time()
print("The file is received in %d seconds"%((finish - start)))

#close the clientsocket
clientSocket.close()
```

The file being sent from the `server` to the `client` is `test.txt`, which is about `40 MB` big, the terminals show the result shown in the following pictures:



```
haoquan@haoquan-ThinkPad-X1-Carbon-7th:~/CS339/simpleSCmodel$ sudo python3 simple.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6
*** Adding switches:
s1
*** Adding links:
(100.00Mbit) (100.00Mbit) (h1, s1) (100.00Mbit) (100.00Mbit) (h2, s1) (100.00Mbit) (100.00Mbit) (h3, s1) (100.00Mbit) (100.00Mbit) (h4, s1) (100.00Mbit) (100.00Mbit) (h5, s1) (100.00Mbit) (100.00Mbit) (h6, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6
*** Starting controller
c0
*** Starting 1 switches
s1 ...(100.00Mbit) (100.00Mbit) (100.00Mbit) (100.00Mbit) (100.00Mbit) (100.00Mbit)
The file is received in 4 seconds
All the files are received in 4 seconds
*** Starting CLI:
stopping h1
mininet> 
```

As we see that the file is sent and received in `4` seconds. While the theoretical value for the transmission is

$$t = \frac{8 \times 50}{100} = 4\,\mathrm{s}$$

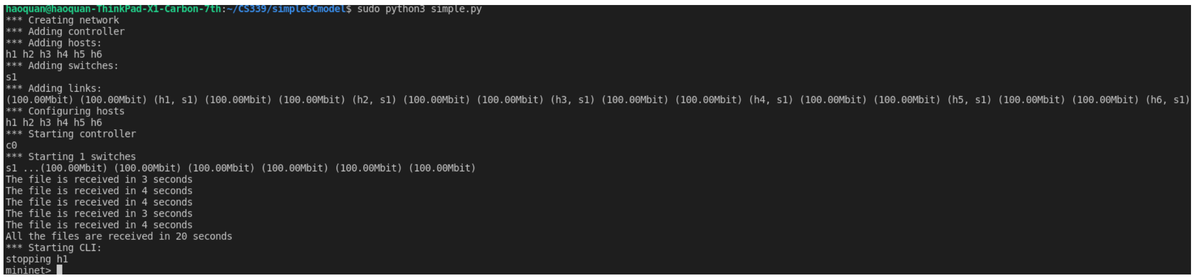
The result fits well.

Burden Test

In this test, all the five clients will request the file from the server roughly at the same time. However, there is only one `TCP` socket, which allows only one communication happens at one time, the expected time used for transmission will increase. The driving program is shown as follows:

```python
from mininet.net import Mininet
from mininet.topo import Topo
from mininet.cli import CLI
from mininet.log import setLogLevel, info
import time

class simpleTopo(Topo):
    def build(self, **_opts):
        s1 = self.addSwitch('s1')
        h1 = self.addHost('h1', ip = '10.0.0.1')
        h2 = self.addHost('h2', ip = '10.0.0.2')
        h3 = self.addHost('h3', ip = '10.0.0.3')
        h4 = self.addHost('h4', ip = '10.0.0.4')
        h5 = self.addHost('h5', ip = '10.0.0.5')
        h6 = self.addHost('h6', ip = '10.0.0.6')
        self.addLink(h1,s1, bw = 100)
        self.addLink(h2,s1, bw = 100)
        self.addLink(h3,s1, bw = 100)
        self.addLink(h4,s1, bw = 100)
        self.addLink(h5,s1, bw = 100)
        self.addLink(h6,s1, bw = 100)

def run():
    topo = simpleTopo()
    net = Mininet( topo = topo )
    net.start()
    start = time.time()
    h1,h2,h3,h4,h5,h6 = [net.get(h) for h in ('h1','h2','h3','h4','h5','h6')]
    h1.sendCmd('python3 compxserver.py')
    #time.sleep(1)
    h2.sendCmd('python3 compxclient.py')
    finish = time.time()
    print("All the files are received in %d micro-seconds"%((finish - start))
    CLI( net )
    net.stop()


if __name__ == '__main__':
    setLogLevel( 'info' )
    run()
```

The terminal shows that the time consumed by transmitting the file is `20` seconds, as shown in the following figure



```
haoquan@haoquan-ThinkPad-X1-Carbon-7th:~/CS339/simpleSCmodel$ sudo python3 simple.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6
*** Adding switches:
s1
*** Adding links:
(100.00Mbit) (100.00Mbit) (h1, s1) (100.00Mbit) (100.00Mbit) (h2, s1) (100.00Mbit) (100.00Mbit) (h3, s1) (100.00Mbit) (100.00Mbit) (h4, s1) (100.00Mbit) (100.00Mbit) (h5, s1) (100.00Mbit) (100.00Mbit) (h6, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6
*** Starting controller
c0
*** Starting 1 switches
s1 ...(100.00Mbit) (100.00Mbit) (100.00Mbit) (100.00Mbit) (100.00Mbit) (100.00Mbit)
The file is received in 3 seconds
The file is received in 4 seconds
The file is received in 4 seconds
The file is received in 3 seconds
The file is received in 4 seconds
All the files are received in 20 seconds
*** Starting CLI:
stopping h1
mininet> 
```

We see that the growth of the time is linear with respect to the number of clients requesting for the file. This is consistent to the expectation that `TCP` socket will finish its service for a client and then continue to open its service to next one.