

Networking Lab Assignment 7

Socket Programming -TCP

Albin Antony

14 March 2019

1 Socket Programming-TCP

1.1 Aim

To Implement Client-Server communication using Socket Programming and TCP as transport layer protocol.

1.2 Theory

1.2.1 TCP

TCP (Transmission Control Protocol) is a standard that defines how to establish and maintain a network conversation via which application programs can exchange data. TCP works with the Internet Protocol (IP), which defines how computers send packets of data to each other. Together, TCP and IP are the basic rules defining the Internet. TCP is defined by the Internet Engineering Task Force (IETF) in the Request for Comment (RFC) standards document number 793.

1.2.2 Client, Server and Socket

- Server- A server is a software that waits for client requests and serves or processes them accordingly.
- Client- a client is requester of this service. A client program request for some resources to the server and server responds to that request.
- Socket- Socket is the endpoint of a bidirectional communications channel between server and client. Sockets may communicate within a process, between processes on the same machine, or between processes on different machines. For any communication with a remote program, we have to connect through a socket port.

1.3 Algorithm

1.3.1 Server

Algorithm 1 Algorithm for creating a tcp server

```
1 START
2 Create TCP SOCKET
3 Bind SOCKET to a PORT
4 Start listing at the binded PORT for connection from CLIENT
5 WHILE TRUE:
6     ACCEPT connection from CLIENT
7     RECEIVE message from CLIENT
8     SEND message to CLIENT
9 STOP
```

1.3.2 Client

Algorithm 2 Algorithm for creating a tcp client

```
1 START
2 CREATE TCP SOCKET
3 Connect to server using IP and PORT
4 SEND message to server
5 RECEIVE message from server
6 STOP
```

1.4 Program

1.4.1 Server

```
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

port=8080

s.bind((' ',port))

s.listen(5)

while True:
    c,addr=s.accept()
    message=c.recv(1024)
    print ( 'Got_Connection_from',addr)
    print message
    c.send( 'Thanks_for_connecting' )
    c.close()
```

1.4.2 Client

```
import socket

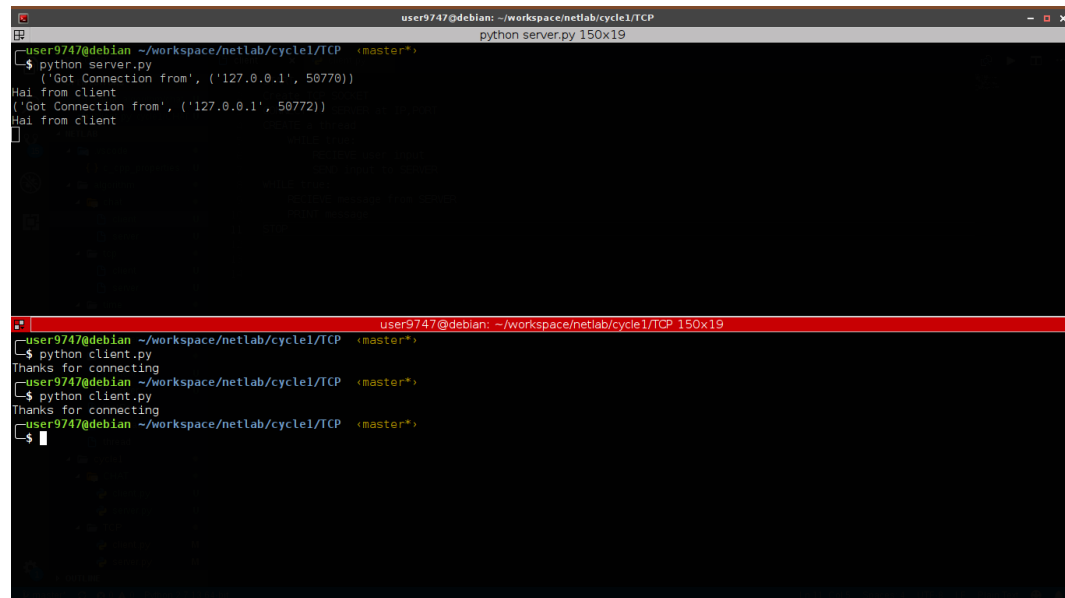
s= socket.socket(socket.AF_INET, socket.SOCK_STREAM)

port = 8080

s.connect(('127.0.0.1',port))
s.send( 'Hai_from_client' )
print s.recv(1024)

s.close()
```

1.5 Output



```
user9747@debian: ~/workspace/netlab/cycle1/TCP
python server.py 15019
user9747@debian ~/workspace/netlab/cycle1/TCP (master*)
$ python server.py
('Got Connection from', ('127.0.0.1', 50770))
Hai from client
('Got Connection from', ('127.0.0.1', 50772))
Hai from client
user9747@debian: ~/workspace/netlab/cycle1/TCP 150x19
user9747@debian: ~/workspace/netlab/cycle1/TCP 150x19
user9747@debian ~/workspace/netlab/cycle1/TCP (master*)
$ python client.py
Thanks for connecting
user9747@debian ~/workspace/netlab/cycle1/TCP (master*)
$ python client.py
Thanks for connecting
user9747@debian ~/workspace/netlab/cycle1/TCP (master*)
$
```

1.6 Result

Implemented TCP Socket Communication on Python 2.7.13 and executed on Debian 4.9 Kernel 4.9 and outputs were verified.

Server code creates a TCP socket using the socket library. Then binds the server to port 8080. The socket then listens for communications to this port. In an infinite while loop the server accepts the connection and address from connecting clients. Then it receives message from this connection and sends message on the same. After that the connection is closed.

Client code creates a TCP socket same as above. Then connects to the ip and port of the server. It then sends a message to the server. Then displays the message from server. Then closes the socket.