**CSE1004: Network and Communication, Winter 2018-19**

**Vellore Institute of Technology  
Instructor: Prof. JAYAKUMAR K - SCOPE**

**Lab report**

**Title of Lab: Multithreaded Socket Programming in Python**

**Experiment #: 5**

**Date: 13|01|2019**

**Author's name: Gagan Deep Singh**

**Registration ID: 17BCI0140**

**Lab section: FRIDAY L25 – L26**

**Aim:** To write a multithreaded client-server program (Socket program) in python 3 for multiple clients.

**Problem description**

Socket Programming helps us to connect a client to a server. Client is message sender and receiver and server are just a listener that works on data sent by client.

A thread is a light-weight process that does not require much memory overhead, they are cheaper than processes. Multithreading is a process of executing multiple threads simultaneously in a single process. A \_thread module & threading module is used for multi-threading in python, these modules help in synchronization and provide a lock to a thread in use.

**Algorithm description**

Server:

1. To write Internet servers, we use the socket function available in socket module to create a socket object. A socket object is then used to call other functions to setup a socket server.
2. Now call bind(hostname, port) function to specify a port for your service on the given host.
3. Next, call the accept method of the returned object. This method waits until a client connects to the port you specified, and then returns a connection object that represents the connection to that client.

Client:

1. Let us write a very simple client program which opens a connection to a given port 12345 and given host. This is very simple to create a socket client using Python's socket module function.
2. The socket.connect(hosname, port ) opens a TCP connection to hostname on the port. Once you have a socket open, you can read from it like any IO object.

**Program**

#Server:  
import socket

from threading import Thread

class ClientThread(Thread):

def \_\_init\_\_(self,ip,port):

Thread.\_\_init\_\_(self)

self.ip = ip

self.port = port

print("New server socket thread started for " + ip + ":" + str(port))

def run(self):

while 1:

print("Server received data:", conn.recv(2048))

MESSAGE = input("Server: Enter Response from Server or Enter 'exit' :")

if MESSAGE == 'exit':

break

conn.send(MESSAGE.encode('utf-8'))

print("Gagan Deep Singh: 17BCI0140")

TCP\_IP = '0.0.0.0'

TCP\_PORT = 2004

BUFFER\_SIZE = 20

tcpServer = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

tcpServer.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

tcpServer.bind((TCP\_IP, TCP\_PORT))

threads = []

while True:

tcpServer.listen(4)

print("Server: Waiting for connections...")

(conn, (ip,port)) = tcpServer.accept()

newthread = ClientThread(ip, port)

newthread.start()

threads.append(newthread)

for t in threads:

t.join()

#Client: Only one client code is written.

import socket

print("Gagan Deep Singh: 17BCI0140")

host = socket.gethostname()

port = 2004

BUFFER\_SIZE = 2000

mes = input("ClientA: Enter message or Enter 'exit':")

tcpClientA = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

tcpClientA.connect((host, port))

while mes != 'exit':

tcpClientA.send(mes.encode('utf-8'))

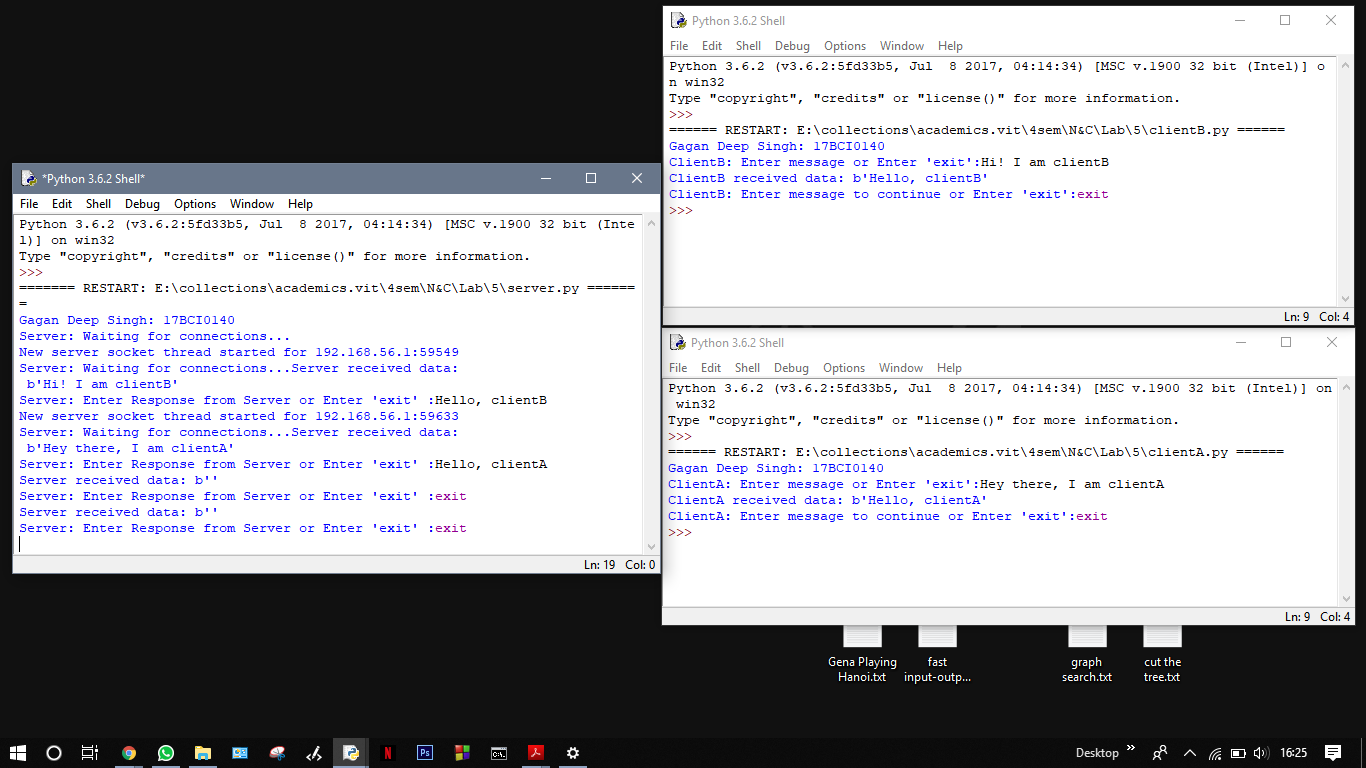
data = tcpClientA.recv(BUFFER\_SIZE)

print("ClientA received data:", data)

mes = input("ClientA: Enter message to continue or Enter 'exit':")

tcpClientA.close()

**Sample runs**



Note:

1. Only **two clients** are connected in this assignment.
2. Server got address and port of client connected.

**Analysis and conclusions**

It is shown that a simple client server code can simulate network connections and how connection between client and server is made and is used to transfer messages across network.