**Computer Network project**

Group members

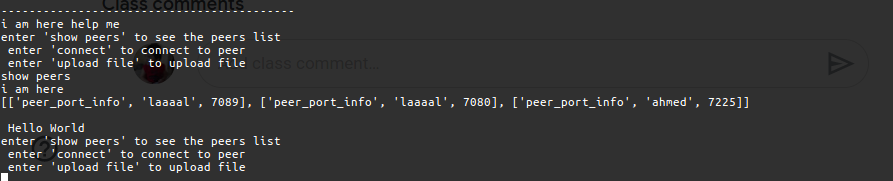
***F17 9433 Ahmed Waheed***

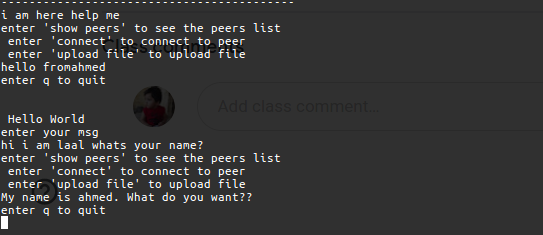
***F17 9428 Shaheer Rahi***

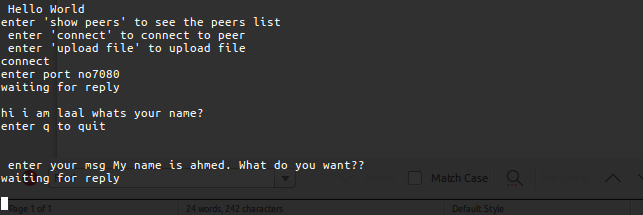
***F17 9436 Haroon Hamid***

***F16 9401 Awais Rajoka***

show peers.

 Talk between two peers





Server.py

*from* socket *import* \*

*import* threading

*import* random

*import* json

*import* pickle

class server:

    counter = 0

    serverPort = random.randint(7000,9000)

    print(serverPort)

    def \_\_init\_\_(self):

        self.peer\_list = []

        self.serverSocket = socket(AF\_INET, SOCK\_STREAM)

        self.serverSocket.bind(('',self.serverPort))

        self.serverSocket.listen(10)

        self.listen()

    def *listen*(self):

*while* True:

            self.client\_conn, self.addr = self.serverSocket.accept()

            threading.Thread(target = self.listenToclient).start()

            print(self.addr)

            # *self.listenToclient()*

    # *def add\_peers(self,client\_conn,addr):*

    #     *self.peer.append(addr)*

    def *listenToclient*(self):

        # *self.add\_peers(client\_conn,addr)*

*while* True:

            print("hello")

            data = pickle.loads(self.client\_conn.recv(4096))

            print(data)

*if*(data[0] == "peer\_port\_info"):

                self.peer\_list.append(data)

                self.client\_conn.sendall("done".encode())

*if*(data[0] == "get\_peers\_info"):

                print("line 39")

                data = pickle.dumps(self.peer\_list)

                self.counter+=1

                print(self.counter)

                s = self.client\_conn.sendall(data)

                print("status",s)

        # *print("message recv from client\_conn: ", fileName)*

        # *fileName = fileName.upper()*

        # *client\_conn.send(fileName.encode())*

        # *print(self.peer)*

        # *self.client\_conn.close()*

    # *def dict\_to\_binary(self,the\_dict):*

    #     *data\_string = pickle.dumps(the\_dict)*

    #     *return data\_string*

    def *binary\_to\_dict*(self,the\_binary):

        data\_loaded = pickle.loads(the\_binary) #*data loaded.*

*return* data\_loaded

server = server()

client.py

*from* socket *import* \*

*import* json

*import* \_pickle *as* pickle

*import* threading

*import* sys

class client:

    #*defining protocols*

    peer\_port\_info = "peer\_port\_info"

    get\_peers\_info = "get\_peers\_info"

    file\_upload = "file\_upload"

    def \_\_init\_\_(self,p2p\_server\_addr,p2p\_server\_port,our\_port,name):

        self.server\_addr = p2p\_server\_addr

        self.server\_Port = p2p\_server\_port

        self.peer\_port = our\_port

        self.name = name

        #*to connect with server*

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

        self.socket.connect((self.server\_addr,self.server\_Port))

        data = pickle.dumps([self.peer\_port\_info,self.name,self.peer\_port])

        self.socket.sendall(data)

        hello = self.socket.recv(5012)

        print(hello.decode())

        # *self.get\_peer\_info()*

        # *self.get\_peer\_info()*

        # *print(self.clientSocket)*

        #*to connect with this peer*

        self.clientSocket = socket(AF\_INET, SOCK\_STREAM)

        self.clientSocket.bind(('',self.peer\_port))

        t2 = threading.Thread(target = self.menu)

        t2.start()

        print("------------------------------------------")

        # *print(self.clientSocket.getsockname())*

        print("------------------------------------------")

        self.clientSocket.listen(4)

        print("i am here help me")

        client\_conn, addr = self.clientSocket.accept()

        t1 = threading.Thread(target = self.listenToclient,args = (client\_conn,addr))

        t1.start()

            # *t1.join()*

    def *listenToclient*(self,client\_conn,addr):

        # *self.add\_peers(client\_conn,addr)*

        fileName = client\_conn.recv(1024).decode()

        print(fileName)

        print("enter q to quit")

*if*(fileName != "q"):

            msg = input("\nenter your msg ")

            client\_conn.send(msg.encode())

            self.listenToclient(client\_conn,addr)

            # *client\_conn.send(self.dict\_to\_binary(self.peer))*

    def *display\_info*(self,info):

        print(info)

    def *get\_peer\_info*(self):

        data = pickle.dumps([self.get\_peers\_info])

        self.socket.sendall(data)

        print("i am here")

        peer\_info = self.socket.recv(5555012)

*if* peer\_info == b'':

            print("emptyy")

            peer\_info = self.socket.recv(5555012)

            print(peer\_info)

*else*:

            print(pickle.loads(peer\_info))

    def *connect*(self):

        self.new\_Sock = socket(AF\_INET, SOCK\_STREAM)

        port = int(input("enter port no"))

        self.new\_Sock.connect((self.server\_addr,port))

        self.talk()

    def *talk*(self):

        self.new\_Sock.send(("hello from"+self.name).encode())

*while* True:

            print("waiting for reply \n")

            fileName = self.new\_Sock.recv(1024).decode()

            print(fileName)

            print("enter q to quit \n")

*if*(fileName != "q"):

                msg = input(" \n enter your msg ")

                self.new\_Sock.send(msg.encode())

*else*:

*break*

    def *upload\_file*(self):

        fname = input("enter file name : ")

        data = pickle.dumps([self.upload\_file,fname,self.peer\_port])

        self.socket.sendall(data)

        hello = self.socket.recv(5012)

        print(hello.decode())

    def *menu*(self):

*while* True:

            print(" \n Hello World")

            user\_input = input("enter 'show peers' to see the peers list \n enter 'connect' to connect to peer \n enter 'upload file' to upload file  \n")

*if* user\_input == "show peers":

                self.get\_peer\_info()

*if* user\_input == "connect":

                self.connect()

*if* user\_input == "upload file":

                self.upload\_file()

    # *message = input("client:: enter your message ")*

    # *clientSocket.send(message.encode())*

    # *modifiedMessage = clientSocket.recv(1024)*

    # *print('from server: ', modifiedMessage.decode())*

    # *.clientSocket.close()*

name = sys.argv[1]

our\_port = int(sys.argv[2])

p2p\_server\_addr = sys.argv[3]

p2p\_server\_port = int(sys.argv[4])

c = client(p2p\_server\_addr,p2p\_server\_port,our\_port,name)