Name: Arka Das

MCA 2nd Year 3rd Sem

Roll number: 002210503046

2022-2024

networking assignment – 1,2,3

Jadavpur university

**Assignment- 1**

**Question: 1**

**Problem Statement:**

Write a TCP Day-Time server program that returns the current time and date. Also write a TCP client program that sends requests to the server to get the current time and date. Choose your own formats for the request/reply messages.

**Design for request and reply:**

Here we have a single server and a single client who communicates among themselves. After the server is successfully created and has been bound it waits for the client to send requests.

The client after connecting with the server can send a message to the server. But if the client sends the message “GET\_TIME” only then the server will reply with the current date and time.

The server formats the current date and time as Date: dd/mm/yy, Time: hh:mm:ss and sends this as reply to the client. The server keeps running and multiple clients can connect with the server at any given time.

**Source Code:**

The code for both the server and the client has been written using Python language and the socket library of python is used.

**Code for Server:**

import socket

import threading

from datetime import datetime

ipAddr = "127.0.0.1" #address for localhost

port = 5555

#thread for new client

def onNewClient(con, addr):

data = con.recv(1024)

if(data.decode() == "GET\_TIME"):

print("Client: ", addr , " requested for date and time")

now = datetime.now()

curr = now.strftime("Date: %d/%m/%Y, Time: %H:%M:%S")

con.send(curr.encode()) #send this string to client

print("Response sent to client")

else:

print("Recieved: ", data.decode(), ", from: ", addr)

msg = "Different request given"

con.send(msg.encode())

con.close()

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

print("Server socker object created")

serverSocket.bind((ipAddr, port))

print("Socket bind successfull")

print("Server is listining for clients")

while True:

serverSocket.listen(5) #at most 5 client connection

con, clientAddr = serverSocket.accept()

print("New connection")

print("Connected to client: ", clientAddr)

t = threading.Thread(target = onNewClient, args = (con, clientAddr, ))

t.start()

serverSocket.close()

**Code for Client:**

import socket

ipAddr = "127.0.0.1"

port = 5555

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

print("Client socker object created")

clientSocket.connect((ipAddr, port)) #connect with server with particular port

print("Connected with server")

#ask for date-time to server

message = input("Enter message: ") #GET\_TIME for time

clientSocket.send(message.encode())

#recieve date time from server

if(message == "GET\_TIME"):

print("Request for current date and time sent to server")

currTime = clientSocket.recv(1024)

print("Current date and time recieved from server")

print(currTime.decode())

else:

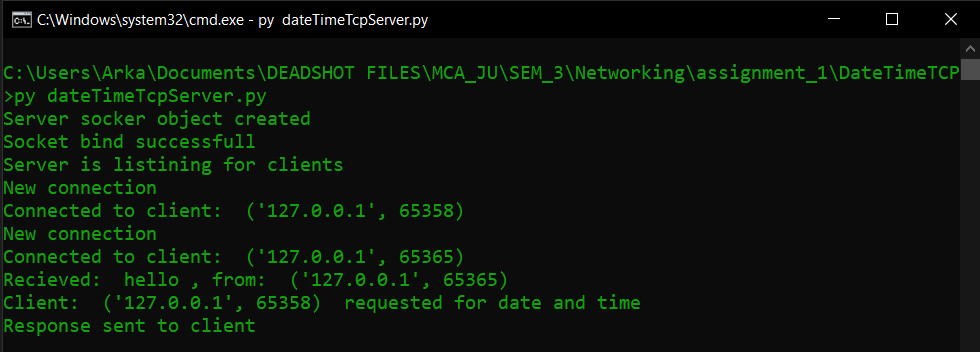
msg = clientSocket.recv(1024)

print(msg.decode())

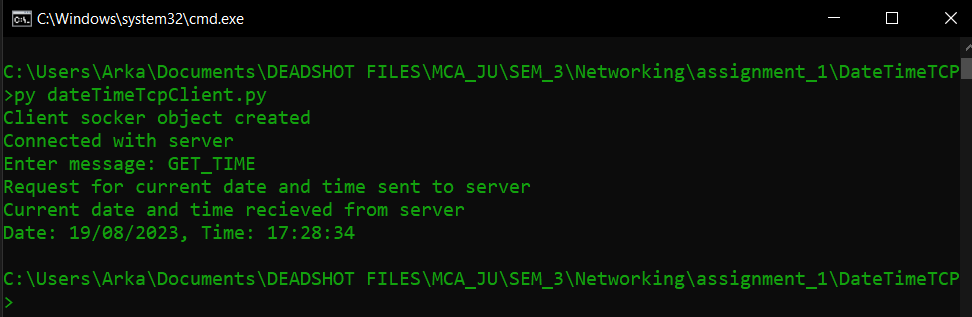
clientSocket.close() #close connections

**Output:**

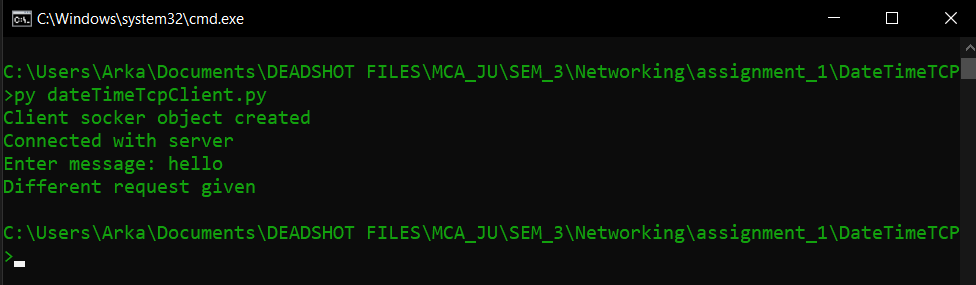
Server



First client



Second client



**Question: 2**

**Problem Statement:**

Write a TCP Math server program that accepts any valid integer arithmetic expression, evaluates it and returns the value of the expression. Also write a TCP client program that accepts an integer arithmetic expression from the user and sends it to the server to get the result of evaluation. Choose your own formats for the request/reply messages.

**Design for request and reply:**

Here we have a single server and a single client who communicates among themselves. After the server is successfully created and has been bound it waits for the client to send requests.

The client after connecting with the server can send a message to the server. Here the client can send any arithmetic expression either valid or invalid.

The server will receive the expression and will try to evaluate that expression. Now if the given expression is a valid one then the server will send the result back to the client.

If the expression is invalid then the server will catch any exception during evaluation and will send appropriate messages to the client.

The program is written in such way that server will keep running and multiple clients can connect with the server. Clients can connect and disconnect and all client operations are in isolation from other clients

**Source Code:**

The code for both the server and the client has been written using Python language and the socket library of python is used.

**Code for Server:**

import socket

import threading

ipAddr = "127.0.0.1" #address of localhost

port = 5555

def onNewThread(con, addr):

expression = con.recv(1024).decode() #recieve expression from client

print("From client: ", addr)

print("Expression recieved: ", expression, "\n")

try:

result = eval(expression) #evaluate the expression

except: #if any exception occurs

msg = "Invalid expression given"

print(msg, "\n")

con.send(msg.encode())

else: #if no exception occurs

print("Sending result back to client: ", addr, "\n")

con.send(str(result).encode())

con.close()

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

print("Server socker object created")

serverSocket.bind((ipAddr, port)) #bind socket with localhost and port

print("Socket bind successfull")

print("Server is listining for clients")

while True:

serverSocket.listen(5)

con, clientAddr = serverSocket.accept() #accept incoming connection

print("New connection")

print("Connected to client: ", clientAddr)

t = threading.Thread(target=(onNewThread), args=(con, clientAddr, ))

t.start()

serverSocket.close()

**Code for Client:**

import socket

ipAddr = "127.0.0.1"

port = 5555

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

print("Client socker object created")

clientSocket.connect((ipAddr, port)) #connect with server

print("Connected with server")

#ask for expression

print("Enter and integer expression to evaluate")

message = input()

clientSocket.send(message.encode()) #send the expression to server

print("Request for expression evaluation sent to server")

result = clientSocket.recv(1024) #recieve result of expression

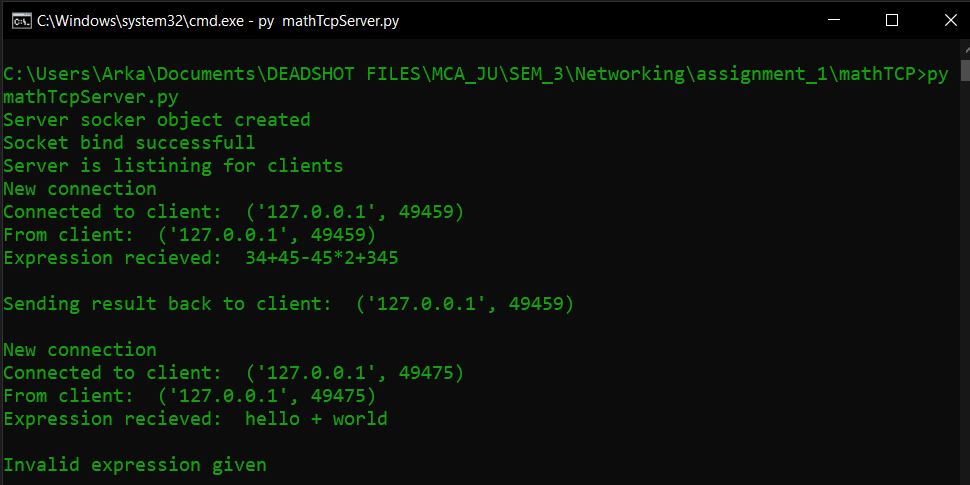
print("Result recieved from server")

print("Result of expression is: ", result.decode())

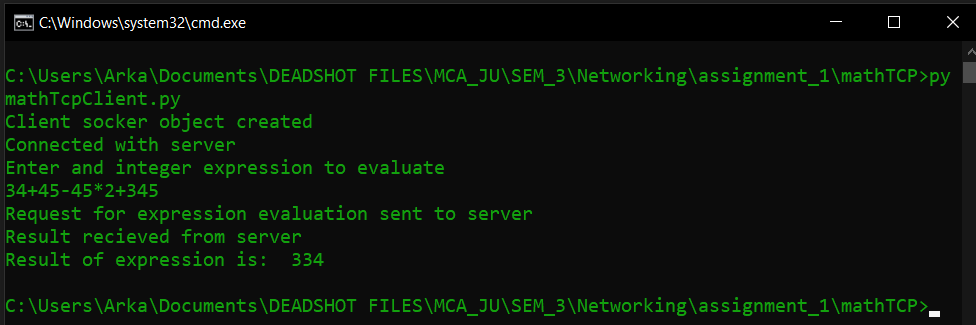
clientSocket.close() #close all connection

**Output:**

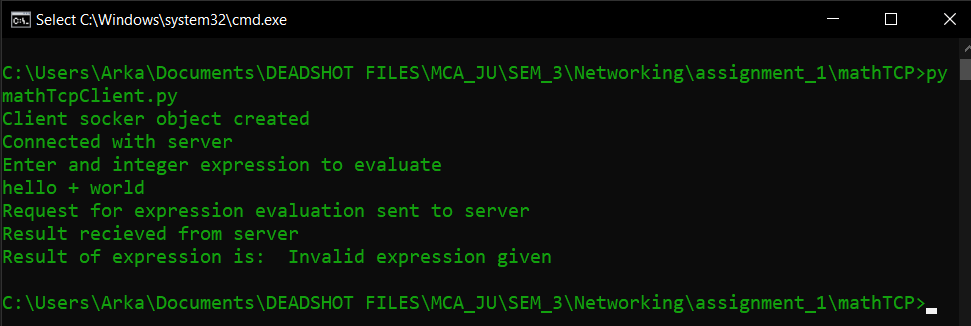
Server



Frist client



Second client



**Question: 3**

**Problem Statement:**

Implement a UDP server program that returns the permanent address of a student upon receiving a request from a client. Assume that a text file that stores the names of students and their permanent addresses is available locally to the server. Choose your own formats for the request/reply messages.

**Design for request and reply:**

Here we have a single server and a single client who communicates among themselves. After the server is successfully created and has been bound it waits for the client to send requests. Here the protocol will be used is UDP.

The client will now send a message using the server's address and port number. There is no connection object here like TCP. Both server and client will communicate using each other’s address and port.

Server will have a CSV file containing the data about students and their addresses. Before creating the socket, it will load the data of the CSV file and will create a dictionary with the student name being the key and corresponding addresses being the value.

If a student name (key) doesn’t exist in the dictionary then it will send an appropriate message to the client.

**Source Code:**

The code for both the server and the client has been written using Python language and the socket library of python is used.

**Code for Server:**

import socketimport csv

ipAddr = "127.0.0.1"

port = 5555

#creating map of addresses

data = {}

with open('data.csv', mode = 'r') as file:

csvFile = csv.reader(file) #load data from a CSV file

for lines in csvFile:

data[lines[0]] = lines[1] #populate the dictonary

udpServerSocket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM) #udp socket object

#if connection in port already exists then reuse that connection

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

print("Socket object created")

udpServerSocket.bind((ipAddr, port))

print("Socket bind done")

#recieve student name from client

while True:

conn = udpServerSocket.recvfrom(1024) #recieve data from client

#conn[0] contains the message send by client

#conn[1] contains the address of client who has sent the data

studentName = conn[0].decode()

print("Name = \"", studentName, "\" revieved from: ", conn[1])

add = data.get(studentName) #get data from dictionary

if add == None: #if address not found

msg = "Student record not found"

udpServerSocket.sendto(msg.encode(), conn[1])

else:

udpServerSocket.sendto(add.encode(), conn[1])

udpServerSocket.close() #close connection

**Code for Client:**

import socket

ipAddr = "127.0.0.1"

port = 5555

udpClientSocket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

print("Socket object created")

while True:

print("Enter a student name: ")

msg = input()

if(msg == "exit"):

break

udpClientSocket.sendto(msg.encode(), (ipAddr, port)) #send message to server

data = udpClientSocket.recvfrom(1024) #recieve response

#data[0] contains the message from server

#data[1] contains the address of server

add = data[0].decode()

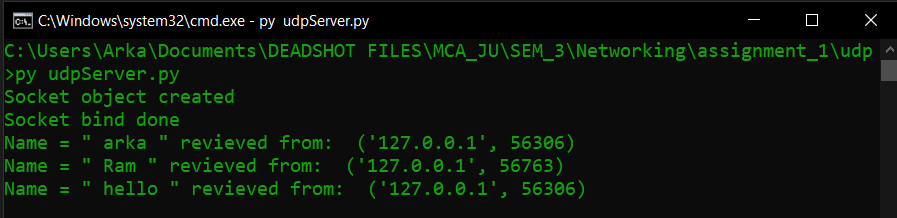
print("Address for student: ", end="")

print(add, "\n")

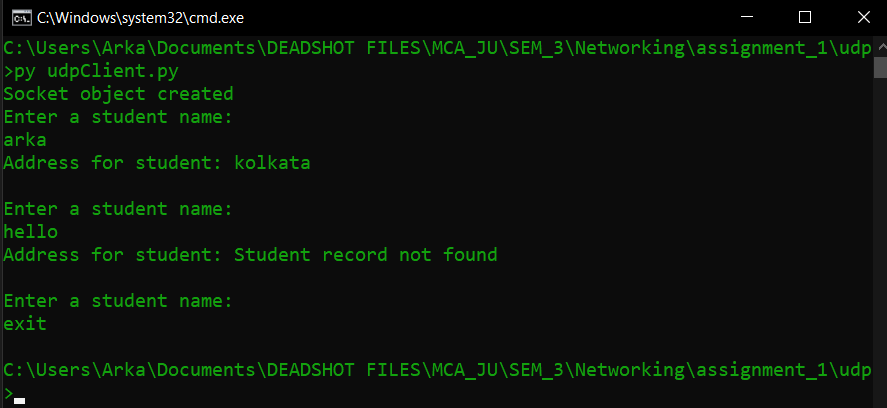
udpClientSocket.close()

**Output:**

Server



First client



Second Client

