**SWINBURNE UNIVERSITY OF TECHNOLGY**

**PROJECT REPORT**

**ON**

**INTRODUCTION TO NETWORK PROGRAMMING**

****Submitted to :

**Recep Ulusoy**

Submitted by

101979004 Mohmmed Sami Ahamed

101985810 Bikesh Nath Khanal

101989090 Bharadwaj Perala

101994230 Venkata Sai Raju Peetha

**CONTENTS OF PROJECT**

1. Introduction

2. Analysis & Implementation of design

a. Pseudocodes

b. Flowcharts

3. Coding in project

a. Server side

b. Client side

4. Testing of coding

5. Conclusion of program

6.References

Introduction

* In this project, the implementation of a client and server has been made on the Python. The implementation has been done by making the connection between the ports of the TCP socket that has been started on the same IP addresses.
* The IP addresses has been hardcoded in the python files so that error of any kind can be avoided in the manual entry. The implementation has been done by taking input from the user, and traversing the data on the client server sockets.
* The python implementation has been done on Python 3. The Thread library has been used for the purpose. The testing of the program has been done on the command line interface. The implementation and coding has been done on the Python official IDE.
* There has been de difficulties that has been observed while making the implementation. The difficulty in the compilation of the thread library has been sorted by searching for the threading library which is compatible with the Python 3 and not with Python 2. The same has been followed for the print statements of the code.
* The network analyser is for handling the connection on the front panel with the measurements that are set to connect with the other devices that are et under the introduction of the time delay and the phase shift. The factors are set to define the check on the different measurement elements with the systematic errors.
* In this project, a client server implementation has been done for the give scenario using Thread library of python. The socket programing has been used using TCP connections.
* The network analyser is the instrument for the measurement of the network parameters of the electrical networks. This is based on the measurement of the s parameters where the reflection and the transmission is for the electrical networks that are mainly found to be easy at the higher set of frequency.
* There is a check of the other parameters which are used to characterise the two port networks with the amplifiers and the filters that can be for the network through the arbitrary number of the ports. Here, the overview is based on the higher frequencies and the check on how the operating frequencies are able to range in between 5 to 1.05Thz.
* Here, there are analysers for handling the ranges which are down to 1Hz. There are networks for the check on the stability analysis of the open loops and for the measurement of the audio and the ultrasonic components.
* The network analyser is to analyse the scalar networks with the measurement of the amplitude properties with the vector network analyser to handle and measure the amplitude with the phase properties. For the network analyser, there is a need to check on the signal generator or the signal source which will be able to provide with the better network analysers that does not have their own signal generator.
* For this, it is possible to control all the modern network analysers which have been set for the building of the signal generators with the higher performance and the network analysers. (Puthukodan et al., 2014).
* There are two built in sources that are used for the applications like the mixer test where the sources are also providing the RF signals and the LO or the amplifier intermodulation testing the tests sets are set to take the signal generator output and then route it to the device which is under the tests.

Analysis & Implementation of design

A. Pseudocodes

Input Organization Name

Name Matched = Details Decoded and Displayed

Name Unmatched = Displayed error of User Authentication

Menu Option 1

Get Server Name and IP Address

Client:

Get Details from the users.

Send the detail of choice to Server.

Receive the Output.

Use of Exception Handling for Wrong Data.

Server:

Receive choice from Client.

Read choice by using define function.

Use of loop and Exception Handling to response the Client with

Right or Wrong Data.

**Menu Option 2**

Get Statistics (Mean, Minimum and Maximum)

Client:

Send Choice to Server.

Receive Statistics.

Output Displayed.

Server:

Receive Client Choice.

Define Statistics Value and Use of for loop and if / else loop.

Send Output to Client.

Menu Option 3

**Sort Data by Name or Minutes**

Input Correct: Complete the Action

Input Incorrect: Display the Error Message

Client:

Send the Choice to Server.

Receive the Statistics Value

Receive the Output and Display it.

Server:

Receive Input from the Client.

By using for loop and infile function get the output for the client

according to data.

Sending Output to the client.

**Menu Option 4**

**Add New Organization**

Client:

Send the Choice to the Server.

Send to Server the Name of Organization, IP Address, Domain

Name, Number of Minutes.

Server:

Receive the Input from Client.

Add more Name of Organization, IP Address, Domain Name,

Number of Minutes in the File.

By using Exception Handling matching names and allowed

maximum Attempts.

**Menu Option 5**

**Remove an Organization**

Delete Organization Name from file and Display Message to the

User after Maximum Attempts.

Client:

Send the Choice to the Server.

By using if-else loop Remove the Organization Name from the

List.

Receive the Organization Changes in the Output.

Display the Output.

Server:

Receive the Choice from the Users.

Use of if loop and Exception Handling remove an Organization

Name from the File.

**Menu Option 6**

**Quit Program**

Client:

Send the Choice and Close the Connection

Server:

Disconnect the Client

B. Flowcharts

**User login**:

Enter the user name

True

MENU OPTIONS

FALSE

INVALID USER NAME

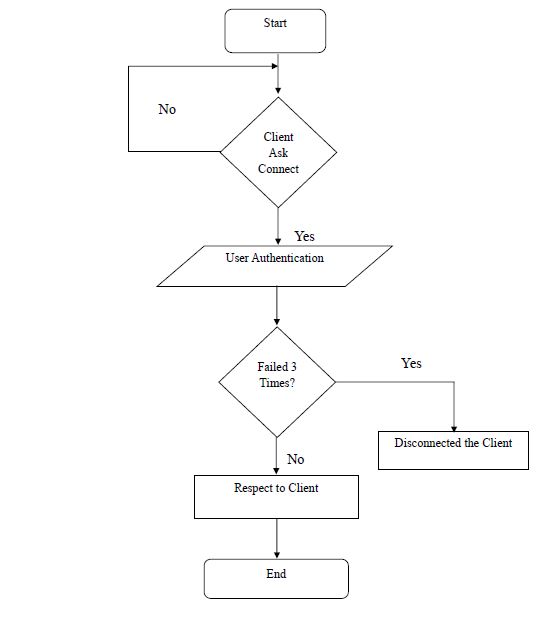
**If Failed Authentication:**

USER NAME

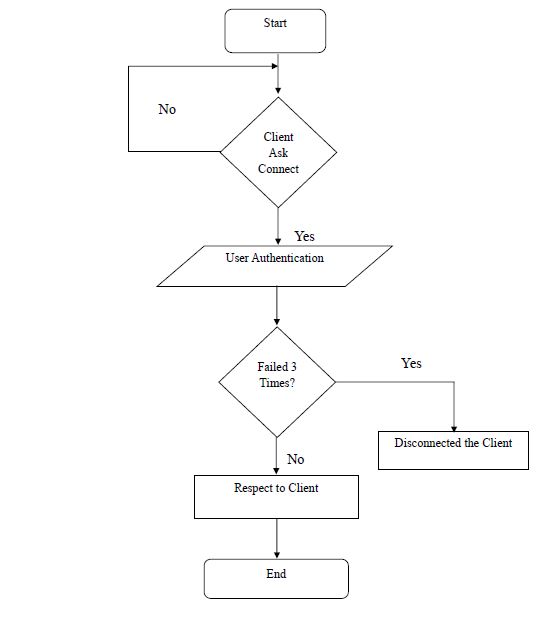
ENTER USER NAME 3 TIMES

DICONNECT THE SERVER

SERVER



CLIENT



Coding in project

Server side:

#server Program that handles multiple client

from socket import \*

from threading import Thread

import serverDetails, minMax, managing, adding, delete

#Socket Creation

host = gethostname()

port = 6666

address = (host, port)

bufsize = 1024

server = socket(AF\_INET, SOCK\_STREAM)

server.bind(address)

server.listen(5)

class ClientHandler(Thread):

def \_\_init\_\_(self, client):

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

self.\_\_client = client

def run(self):

welcomeMsg = "Welcome to the Server\n"

self.\_\_client.send(welcomeMsg.encode("utf-8")) #message to client

while True:

try:

userList = file\_contains("users.txt") #obtaining list of users from users.txt

userStatus, userName = serverAuthentication(self.\_\_client, userList)

try:

if userStatus == "Access Granted":

print ("Successfully authenticated")

print ("Waiting to choose an option...")

choice\_Recv = self.\_\_client.recv(bufsize)

choice = choice\_Recv.decode('utf-8')

orgList = file\_contains("organisations.txt")

status = "continue"

while status != "end":

if choice == '1':

status = serverDetails.server\_Details(self.\_\_client, orgList)

elif choice == '2':

status = minMax.statistics(self.\_\_client, orgList)

elif choice == '3':

status = managing.sorting(self.\_\_client,orgList)

elif choice == '4':

status = adding.add\_Organisation(self.\_\_client, orgList)

elif choice == '5':

status = delete.remove\_Organisation(self.\_\_client, orgList)

else:

status = "end"

if status == "continue":

choice\_Recv = self.\_\_client.recv(bufsize)

choice = choice\_Recv.decode('utf-8')

if status == "end":

print ("Closing Connection...")

print("logging out User")

self.\_\_client.close()#closing connections

print ('Connection Terminated')

break

#exceptional handling

except ConnectionResetError:

userStatus = "Access Denied"

usersList = []

usersOnline = open ("onlineUsers.txt", 'r')

userNames = usersOnline.read()

userNames = userNames.split()

for index in userNames:

if index != userName:

usersList.append(index)

usersOnline.close()

usersConnected = open ("onlineUsers.txt", 'w')

usersConnected.write(' '.join(usersList))

usersConnected.close()

return userStatus

except ConnectionResetError:

print ('\nClosing Connection...')

self.\_\_client.close()

print ('\nconnection Terminated')

break

# reading file from users.txt and organisations.txt

def file\_contains(fileName):

inputFile = open(fileName, 'r')

users = [] #create empty list

for line in inputFile:

line = line.rstrip()

users.append(line.split())

inputFile.close()

return users

#Authenticating Users

def serverAuthentication(client, userList):

flag = "Access Denied"

for index in [1, 2, 3]:

userNameRecv = client.recv(bufsize) #receiving Username

userName = userNameRecv.decode("utf-8")

passwordRecv = client.recv(bufsize) #receiving Password

password = passwordRecv.decode("utf-8")

print ("\nServer received Username and Password")

response = "Unsuccessful"

for index in userList: # userList contains all usernames and password

if index[0] == userName:

if index[1] == password:

response = "Connection Successful"

break

if response == "Connection Successful":

#checking already logged in user in onlineUsers.txt

usersOnline = open ("onlineUsers.txt", 'r')

users = usersOnline.read()

usersOnline.close()

if userName in users:

response = "User already in use"

client.send(response.encode('utf-8'))

else:

#writing username in onlineUsers file

usersOnline = open ("onlineUsers.txt", 'w')

users = users + " " + userName

usersOnline.write(users)

usersOnline.close()

#sending sucessful message to client

client.send(response.encode('utf-8'))

flag = "Access Granted"

return flag, userName

else:

response = "Invalid Username or Password"

client.send(response.encode('utf-8'))# send message to server

userName = "Invalid Username"

return flag, userName

while True:

print("\nWaiting for connection......")

client, address = server.accept()

print("connected from: ", address)

handler = ClientHandler(client)

handler.start()

**#Menu option 1 (server details)**

bufsize = 1024

def server\_Details(client, orgList):

flag = "Organisation Name not found"

for index in [1, 2, 3]:

print ("Waiting for the Organisation Name..")

orgNameRecv = client.recv(bufsize) #receive organisation name

orgName = orgNameRecv.decode('utf-8')

print ("Organisation Name received is ", orgName)

for index in orgList:

if index[0] == orgName:

flag = "Organisation Name found"

details = index

if flag == "Organisation Name not found":

client.send(flag.encode('utf-8')) #send status to server

elif flag == "Organisation Name found":

client.send(flag.encode('utf-8'))

print (flag)

reqServerName = client.recv(bufsize) #receive serverName from client

reqServerName = reqServerName.decode('utf-8')

client.send(details[1].encode('utf-8'))

reqIPAdd = client.recv(bufsize) #receive ipAddress from client

reqIPAdd = reqIPAdd.decode('utf-8')

client.send(details[2].encode('utf-8'))

return "continue"

else:

if index != 3:

print ("\nOrganisation Name not found\n")

return "end"

#Menu option 2(Minimum, Max, Mean)

bufsize = 1024

def statistics(client, orgList):

timeList = []

for index in orgList:

timeList.append(int(index[3]))

minimum = minVal(timeList)

req = client.recv(bufsize) #receive request from client

minRequest = req.decode('utf-8')

if minRequest == "Requesting Minimum":

client.send(str(minimum).encode('utf-8')) #send minRequest to server

maximum = maxVal(timeList)

request = client.recv(bufsize) #receive request from client

request = request.decode('utf-8')

if request == "Requesting Maximum":

client.send(str(maximum).encode('utf-8')) #send maxRequest to server

average = mean(timeList)

request = client.recv(bufsize) #receive request from client

request = request.decode('utf-8')

if request == "Requesting Mean":

client.send(str(average).encode('utf-8')) #send requesting mean to server

return "continue"

#send mean, min and max of connection time

def minVal(timeList):

minimum = timeList[0]

for index in timeList:

if index < minimum:

minimum = index

return minimum

def maxVal( timeList ):

maximum = timeList[0]

for index in timeList:

if index > maximum:

maximum = index

return maximum

def mean(timeList):

length=len(timeList)

add=0

for index in timeList:

add += index

average = add/length

return average

#menu option 3(Sorting)

# sorting alphabetically or numerically

bufsize = 1024

def sorting(client, orgList):

choiceRecv = client.recv(bufsize)

choiceReceived = choiceRecv.decode('utf-8')

if choiceReceived == '1':

message = sortAlphabetically(client, orgList)

elif choiceReceived == '2':

message = sortNumerically(client, orgList)

else:

message = "Please choose from the options given: "

client.send(message.encode('utf-8'))

return "continue"

#sort by alphabetically

def sortAlphabetically(client, orgList):

orgList.sort()

newOrgList = open ("organisations.txt", 'w')

for index in orgList:

newOrgList.write(" ".join(index))

newOrgList.write("\n")

newOrgList.close()

return "Sorting completed Alphabetically"

def fourthElement(elem):

return elem[3]

#sort by numerically

def sortNumerically(client, orgList):

orgList.sort(reverse = True, key = fourthElement)

newOrgList = open ("organisations.txt", 'w')

for index in orgList:

newOrgList.write(" ".join(index))

newOrgList.write("\n")

newOrgList.close()

return "Sorting Numerically"

#menu option 4 (Adding Organisations)

#adding a newOrganisation

bufsize=1024

def add\_Organisation(client, orgList):

for index in [1, 2, 3]:

flag = "Add Organisation"

#Get organisation details

orgName = client.recv(bufsize)

orgName = orgName.decode('utf-8')

svrName = client.recv(bufsize)

svrName = svrName.decode('utf-8')

ipAdd = client.recv(bufsize)

ipAdd = ipAdd.decode('utf-8')

noMinutes = client.recv(bufsize)

noMinutes = noMinutes.decode('utf-8')

print ("New Organisation Details Received")

#Check if Organisation name exists

for index in orgList:

if index[0] == orgName:

print ("\nOrganisation already exist")

flag = "Existing Organisation Name"

client.send(flag.encode('utf-8'))

break

#Add Organisation name to organisations.txt file

if flag == "Add Organisation":

currentOrg = open ("organisations.txt", 'a')

currentOrg.write("\n" + orgName + " " + svrName + " " + ipAdd + " " + noMinutes)

currentOrg.close()

client.send(flag.encode('utf-8'))

return "continue"

return "end"

#menu option 5(Removing Organisations)

#deleting organisation

bufsize = 1024

def remove\_Organisation(client, orgList):

for index in [1, 2, 3]:

print ("Waiting for the Organisation Name")

flag = "Organisation doesn't exist"

orgName = client.recv(bufsize) #receive orgName from client

orgName = orgName.decode('utf-8')

for index in orgList:

if index[0] == orgName: #check if Organisation name exists in File

print ("\nDeleting Organisation Name...\n")

flag = "Organisation exist"

client.send(flag.encode('utf-8')) #send organisation to server

orgList.remove(index) #deleting Organisation from the list

newOrgList = open ("organisations.txt", 'w')

for index in orgList:

newOrgList.write(" ".join(index))

newOrgList.write("\n")

newOrgList.close()

message = "Organisation Deleted"

print (message)

client.send(message.encode('utf-8')) #send message to server

return "continue"

client.send(flag.encode('utf-8'))

return "end"

Client side

#client side program

from socket import \*

bufsize = 1024

def main():

host = gethostname()

port = 6666

address = (host, port)

client = socket(AF\_INET, SOCK\_STREAM)

try:

#connecting to server

client.connect(address)

welcomeMsg = client.recv(bufsize)

status = ""

choice= ""

#displaying welcome message

print(welcomeMsg.decode('utf-8'))

authStatus = authentication(client)

if authStatus == "Access Denied":

print ("end")

else:

print (authStatus +"\nChoose an option")

choice = menu()

client.send(str(choice).encode('utf-8')) #send choice to server

while choice != 6:

choice = str(choice)

if choice == "1":

status = orgDetails(client)

elif choice == "2":

status = statistics(client)

elif choice == "3":

status = sorting(client)

elif choice == "4":

status = addNewOrganisation(client)

elif choice == "5":

status = removeOrganisation(client)

else:

print ("Please enter the choice from the given range")

#choice = int(choice)

choice = 6

if status == "continue":

choice = menu()

client.send(str(choice).encode('utf-8'))

except ConnectionRefusedError: #exception handling

print ("Server not Active")

status = "end"

choice = 6

print("Client is Terminated")

if status != "end":

choice = str(choice)

client.send (choice.encode('utf-8')) #send choice to server

client.close() #close the connection to server

def authentication(client):

flag = "Access Denied"

for index in [1, 2, 3]:

userName = input ("Username: ") #enter the userame

client.send(userName.encode('utf-8')) #send username to server

password = input ("Password: ") #Enter the password

client.send(password.encode('utf-8')) #send password to server

authMsg = client.recv(bufsize) #receive authentication from client

authMsg = authMsg.decode('utf-8')

if authMsg == "Connection Successful":

print ("Connected to the Server")

flag = "Access Granted"

return flag

elif authMsg == "User already in use":

print ("Username already in use. Please use different Authentication")

else:

if index != 3:

print ("\nInvalid Username or Password.\nTry Again\n")

print ("\n3 incorrect attempts!! \n Disconnecting from the Server...\n")

return flag

#check for orgDetails

#prompt for display orgDetails

def orgDetails(client):

status = "Organisation Name not found"

for index in [1, 2, 3]:

orgName = input ("Enter an Organisation Name: ")

client.send(orgName.encode('utf-8')) #send organisation name to server

status = client.recv(bufsize) #receive status from client

status = status.decode('utf-8')

if status == "Organisation Name found" :

status = "Thank you"

reqServerName = "Requesting servername:"

client.send(reqServerName.encode('utf-8'))

serverName = client.recv(bufsize)

name = serverName.decode('utf-8')

print("The ServerName is:",name)

reqIPAdd= "Requesting IPaddress:"

client.send(reqIPAdd.encode('utf-8')) #send ipAddress to server

ipAddress = client.recv(bufsize) #receive ipAddress from client

ipName = ipAddress.decode('utf-8')

print("The Ip address is:",ipName)

return "continue"

elif status == "Organisation Name not found":

print (status)

if index != 3:

print ("\nEnter an Organisation Name in Records\n")

print ("reached Max attempt\n")

print ("\nDisconnecting from the Server...\n")

return "end"

#prompt for display statistics

def statistics(client):

print("Requesting Mean, Minimum value and Maximum value of time from server")

reqMin = "Requesting Minimum"

client.send(reqMin.encode('utf-8')) #send minValue to server

minVal = client.recv(bufsize) #receive minValue from client

minVal = minVal.decode('utf-8')

print("The Minimum is:",int(minVal))

reqMax = "Requesting Maximum"

client.send(reqMax.encode('utf-8')) #send maxValue to server

maxVal = client.recv(bufsize) #receive maxValue from client

maxVal = maxVal.decode('utf-8')

print("The Maximum is:",int(maxVal))

reqMean = "Requesting Mean"

client.send(reqMean.encode('utf-8')) #send mean to server

mean = client.recv(bufsize)

mean = mean.decode('utf-8') #receive mean from client

print("The Mean is:",float(mean))

return "continue"

#Do sorting

def sorting(client):

choice = sortingMenu()

client.send(str(choice).encode('utf-8')) #send choice to server

message = client.recv(bufsize) #receive message from client

message = message.decode('utf-8')

print (message)

return "continue"

#create newOrganisation

def addNewOrganisation(client):

for index in [1, 2, 3]:

orgName = input ("Enter new Organisation name: ")

client.send(orgName.encode('utf-8'))

svrName = input ("Enter the Server Name: ")

client.send(svrName.encode('utf-8'))

ipAdd = input ("Enter the IP Address: ")

client.send(ipAdd.encode('utf-8'))

noMinutes = input ("Enter the number of Minutes: ")

client.send(noMinutes.encode('utf-8'))

regMessage = client.recv(bufsize)

regMessage = regMessage.decode('utf-8')

if regMessage == "Add Organisation":

print ("New Organisation Successfully created.\n")

return "continue"

else:

if index != 3:

print("\Organisation name already exists")

print ("reached Max attempt\n")

print ("Disconnecting from the server...\n")

return "end"

#Erasing the organisation

def removeOrganisation(client):

for index in [1, 2, 3]:

orgName = input ("Enter an Organisation name to be deleted: ")

client.send(orgName.encode('utf-8')) #send organisationName to server

message = client.recv(bufsize) #receive message from client

message = message.decode('utf-8')

if message == "Organisation exist":

print ("\nDeleting Organisation ...\n")

status = client.recv(bufsize)

status = status.decode('utf-8')

print (status)

return "continue"

else:

if index != 3:

print("Organisation name doesn't exists")

print ("reached Max attempt\n")

print ("\nDisconnecting from the server...\n")

return "end"

def menu():

print ("\n--------------------------------")

print (" Menu ")

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

print ("(1) Get sever name and IP address")

print ("(2) Get statistics (mean, minimum, maximum)")

print ("(3) Sort data by name or minutes")

print ("(4) Add new organisation")

print ("(5) Remove organisation")

print ("(6) Quit program")

try:

choice = int(input("Enter your choice (1, 2, 3, 4, 5, 6): "))

except ValueError:

print ("Please enter the choice from the given range")

choice = menu()

return choice

def sortingMenu():

print ("\n--------------------------------")

print (" Sorting Menu ")

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

print ("(1) Sort by Alphabetical Order")

print ("(2) Sort by Numerical Descending Order")

try:

choice = int(input("Enter your choice (1, 2): "))

except ValueError:

print ("Please enter the choice from the given range")

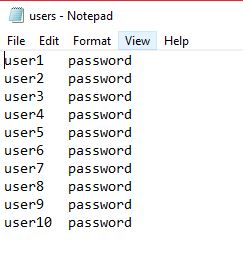
choice = menu()

return choice

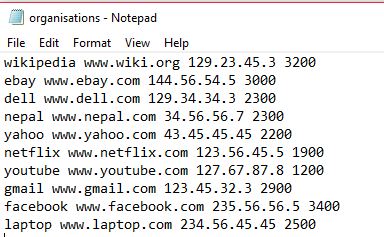
main()

Testing of coding

 **users.txt**



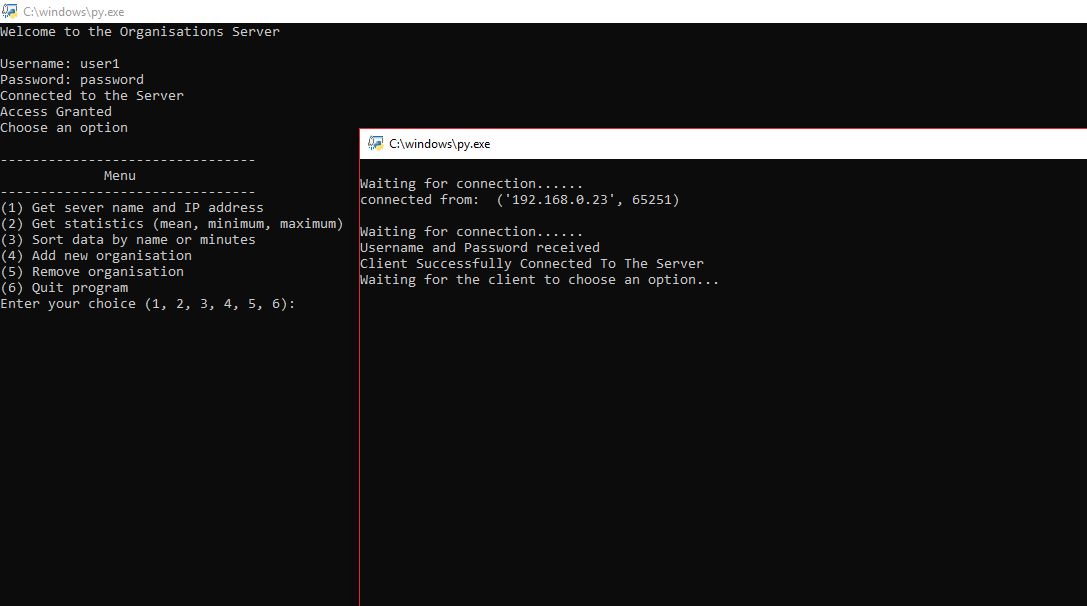
 **organizations.txt**



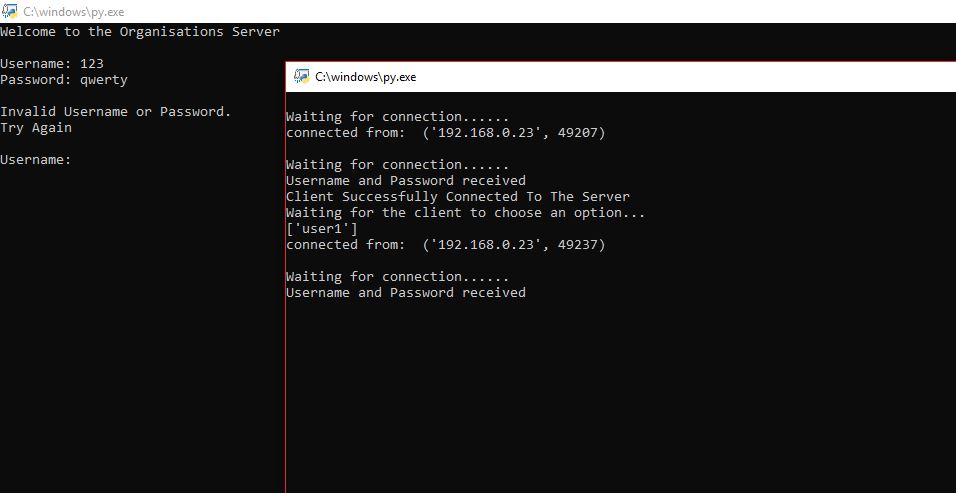
PART 1

client program starts running, it connects to the server and then prompts the user

for a username and a password.

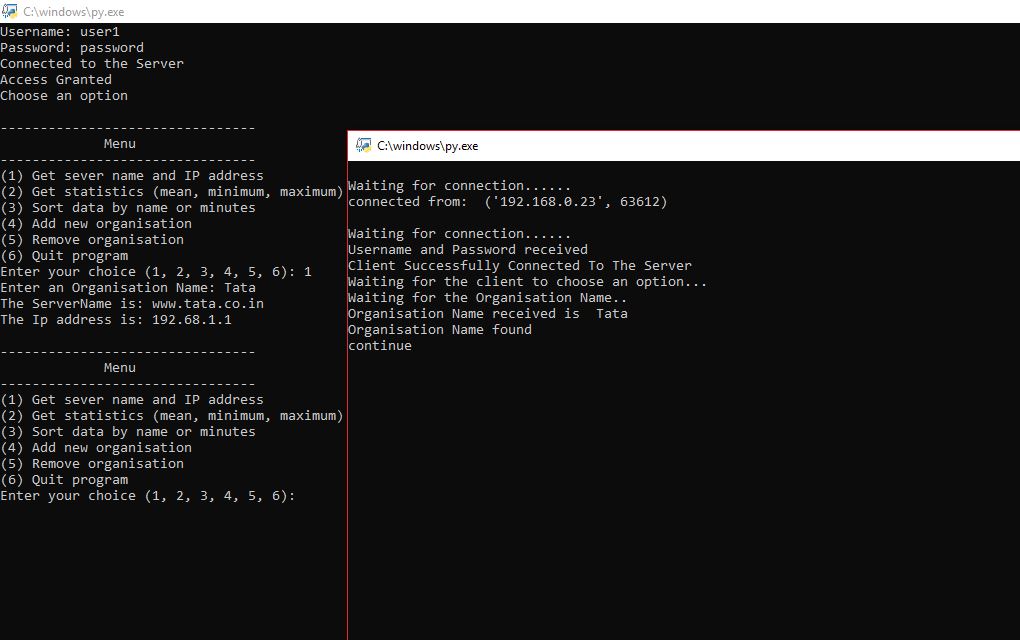


User Authentication

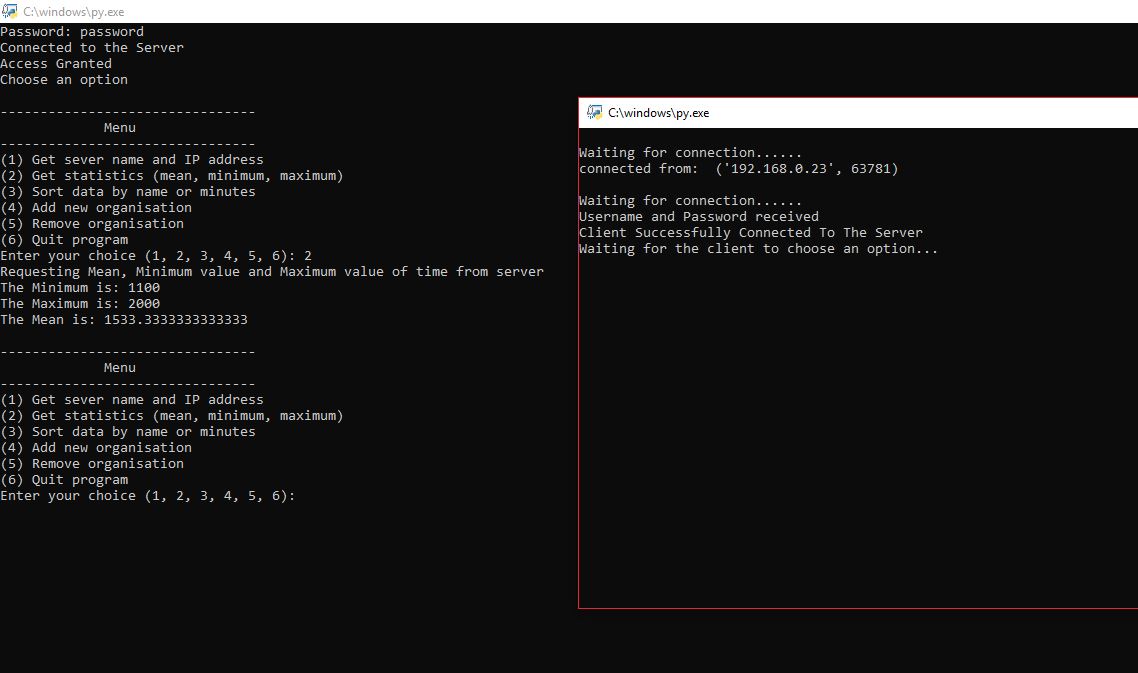


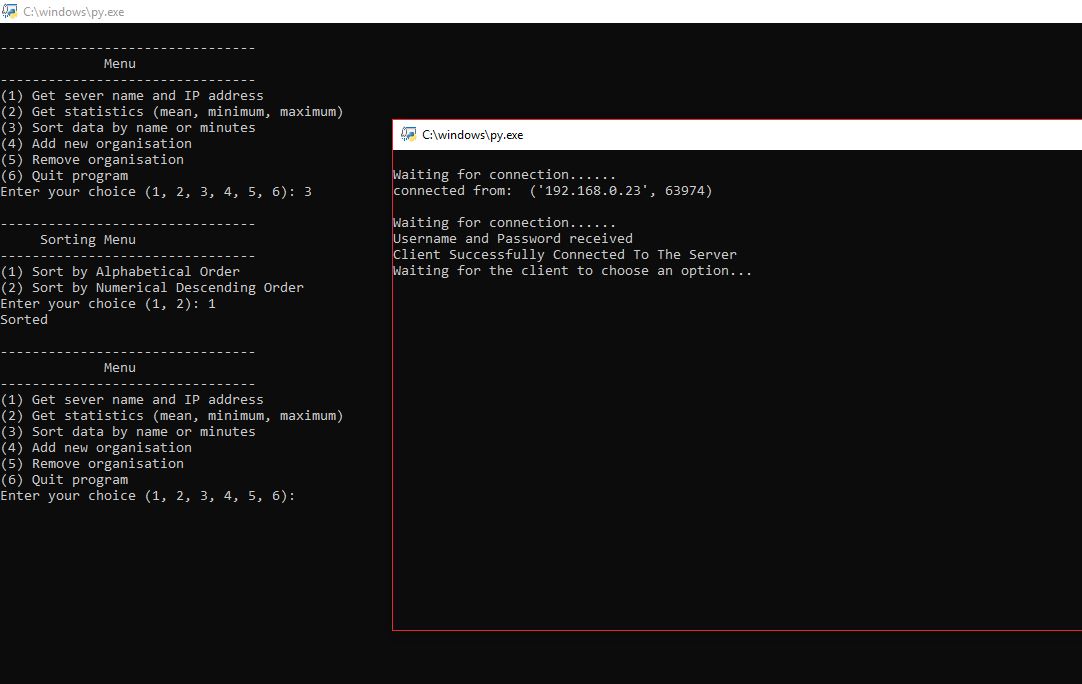
Trying Invalid Password

Part2. Menu Options



Selecting option 1 from menu and getting Server name and IP address from organisations.txt file

Selecting option 2 from menu and getting mean, maximum and minimum according to time

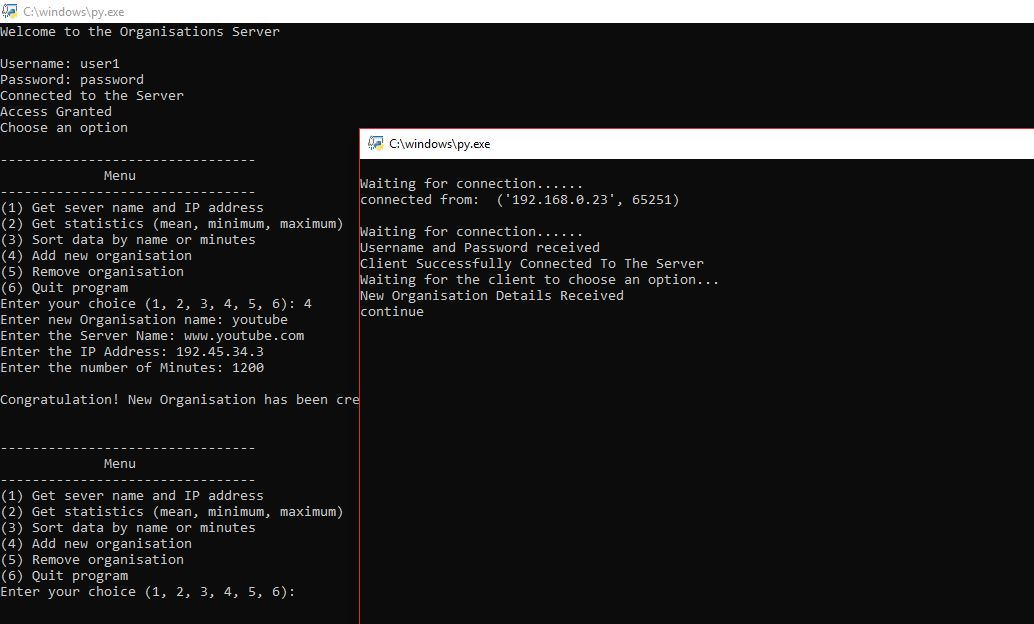


Selecting option 3 from menu

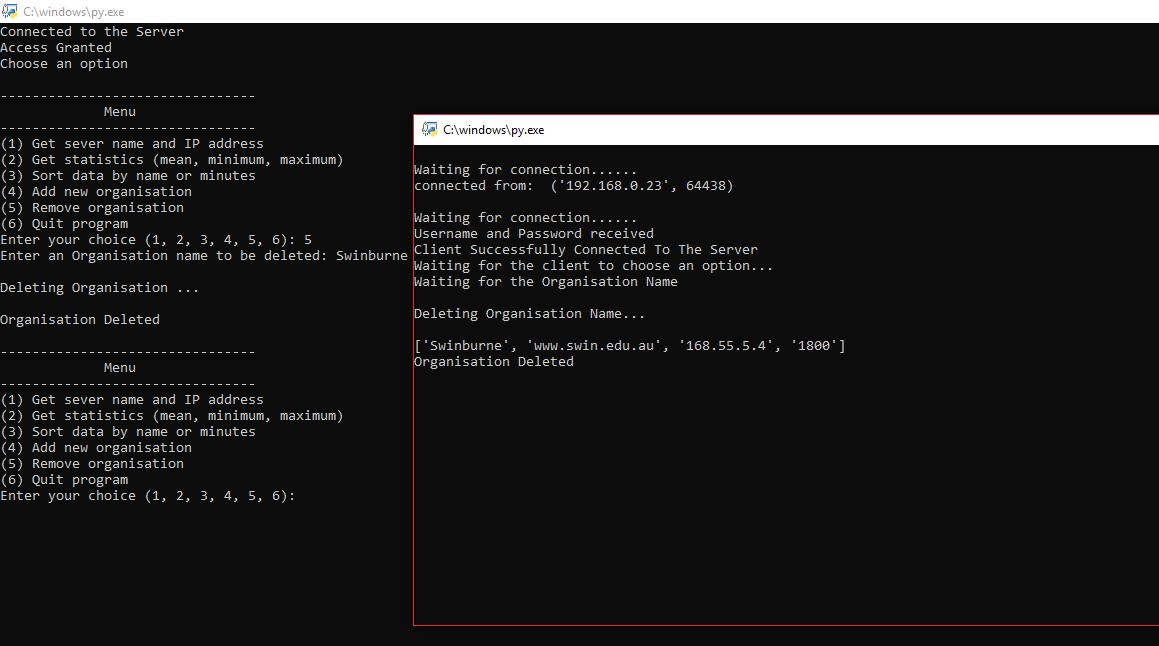
Again asking for two options

Sorting menu of organization details from organisations.txt

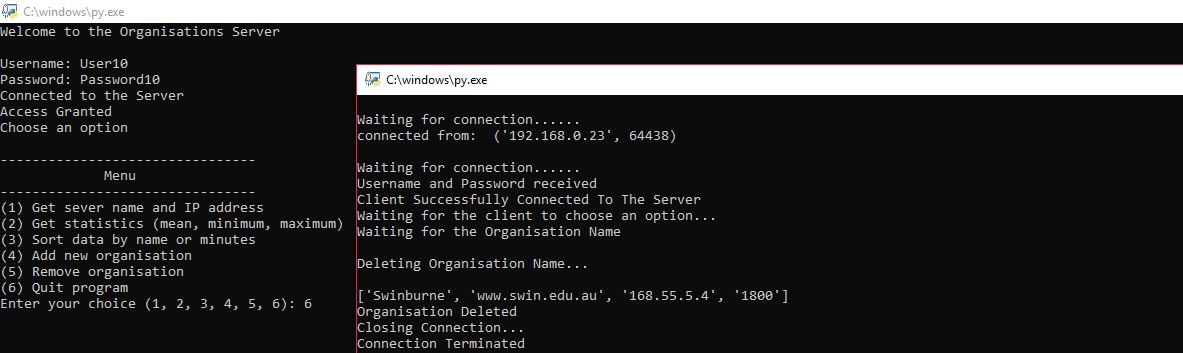
1. According to name
2. According to time



Selecting option 4 from menu and adding New organization details to organisations.txt



Selecting option 5 from menu and removing organization details from organisations.txt



Selecting option 6 from menu and exiting program

# Conclusion of program

The TCP is set for the functions which includes the links to the other hosts like the required packet fragmentation on the transmission medium. Here, the standards are set for the balancing of the traffic load with the unpredictable network behaviour, IP packets which are lost, duplicated or delivered when they are set out of order. The check is also on how the data is remaining undelivered with the source that is notified. The check is on the accuracy for the delivery measures with the check on how the file sharing is processed with the streaming of the media applications and the setup of the real-time applications with Voice over the IP. (Hamler et al, 2015). The setup is based on the Real Time Transport Protocols that tend to operate over the User Datagram sets and handle the delivery of the services which can easily guarantee for the bytes that will be for the bytes that are sent in a proper manner.

References

1. Puthukodan, S., Dadrasnia, E., Vinod, V.K.T., Nguendon, H.K., Lamela, H., Ducournau, G., Lampin, J.F., Garet, F., Coutaz, J.L., Lee, D.M. and Baik, S., 2014. Sub-THz characterisation of multi-walled carbon nanotube thin films using vector network analyser. *Electronics Letters*, *50*(4), pp.297-299.
2. Hammler, J. and Gallant, A., 2015. Balocco.“Simple de-embedding and simulation technique to find permittivity with a THz vector network analyser”. *Proceedings of IRMMW-THz*, pp.23-28.
3. Cha, H.Y., Rafiee, M. and Aftanasar, M.S., 2015, December. Cost effective TRL calibration technique on network analyser. In *RF and Microwave Conference (RFM), 2015 IEEE International* (pp. 1-6). IEEE.
4. Brennan, P.V., Lok, L.B., Nicholls, K. and Corr, H., 2014. Phase-sensitive FMCW radar system for high-precision Antarctic ice shelf profile monitoring. *IET Radar, Sonar & Navigation*, *8*(7), pp.776-786.
5. Gregory, A.P., 2013. Q-factor measurement using a vector network analyser. *NPL Report No. MAT*, *58*.
6. Nykl, J., Jakob, M. and Hrncir, J., 2014, August. Advanced public transport network analyser. In *Proceedings of the Twenty-first European Conference on Artificial Intelligence* (pp. 1229-1230). IOS Press.