Parallel Project
Chat and Networking System
Review – 2
Slot : D2

Submitted by-
Yash Gupta(15BCE0047)
Payal Pankhuri(15BCE0065)
Sakshi Singh(15BCE0243)
Arpit Jain(15BCE0602)

The given code and screenshots are a serial implementation for the chat system made using the pychus library of python. The paralle implementation will be executed in and during Review 3.

INTRODUCTION

Now-a-days communication has become a major part of a person's life and it is very important as it keep aware what is happening around the world and in the region where one lives. Keeping that in mind and our Network and Communication Project, we have done a project where we can establish a connection between a server and a client so that they can share things between them and could talk to each other by the means of texting each other.

In this project, we have written code in python to make our program. In that, the two systems need to connect to the same Wi-Fi-hotspot and one of the user is server while other is client and after the connection been established they can chat to each other by the means of texting, all a client needs to do is to connect on the same IP as of server and on the same port used by the server while making server and they can chat with each other.

PROBLEM DEFINITION

The problem is that can we do chat with each other without using any internet connection and just by a simple hotspot, mobile hotspot or any other and can share thoughts of two different peoples sitting in front of two different machines (one is server and another one is client).

We have created such an application which can do this and we can actually do that chatting by using that program which is written in Python.

But we can connect from server to client only up to certain range that is a drawback we can extend it to a building if we have a strong router so that there is no lagging in the network and both machines can connect to same network.

SERIAL CODE

```
import sys
if not sys.hexversion > 0x03000000:
    version = 2
else:
    version = 3
if len(sys.argv) > 1 and sys.argv[1] == "-cli":
    print("\n Starting command line chat")
    isCLI = True
else:
    isCLI = False


if version == 2:
    from Tkinter import *
    from tkFileDialog import asksaveasfilename
if version == 3:
    from tkinter import *
    from tkinter.filedialog import asksaveasfilename
import threading
import socket
import random
import math


conn_array = []
secret_array = dict()

username_array = dict()
contact_array = dict()  # key: ip address as a string, value: [port, username]

username = "Self"

location = 0
port = 0
top = ""

main_body_text = 0

def binWord(word):
    print("\n entered into binWord")
    master = ""
    for letter in word:
        temp = bin(ord(letter))[2:]
        while len(temp) < 7:
            temp = '0' + temp
        master = master + temp
    return master

def xcrypt(message, key):
    print("\n entered into xcrypt")
    count = 0
```

```
    master = ""
    for letter in message:
        if count == len(key):
            count = 0
        master += str(int(letter) ^ int(key[count]))
        count += 1

    return master
def x_encode(string, number):
    print("\n \nentered into x_encode")
    return xcrypt(binWord(string), bin(number)[2:])

def refract(binary):
    print("\n entered into refract")
    master = ""
    for x in range(0, int(len(binary) / 7)):
        master += chr(int(binary[x * 7: (x + 1) * 7], 2) + 0)
    return master


def formatNumber(number):
    print("\n \nentered into formatnumber")
    temp = str(number)
    while len(temp) < 4:
        temp = '0' + temp
    return temp

def netThrow(conn, secret, message):
    print("\n \n entered into netThrow")
    try:
        conn.send(formatNumber(len(x_encode(message, secret))).encode())
        conn.send(x_encode(message, secret).encode())
    except socket.error:
        if len(conn_array) != 0:
            writeToScreen(
                "Connection issue. Sending message failed.", "System")
            processFlag("-001")

def netCatch(conn, secret):
    print("\n entered into netCatch")
    try:
        data = conn.recv(4)
        if data.decode()[0] == '-':
            processFlag(data.decode(), conn)
            return 1
        data = conn.recv(int(data.decode()))
        return refract(xcrypt(data.decode(), bin(secret)[2:]))
    except socket.error:
        if len(conn_array) != 0:
            writeToScreen(
                "Connection issue. Receiving message failed.", "System")
        processFlag("-001")

def isPrime(number):
```

```
    print("\n entered into isPrime")
    x = 1
    if number == 2 or number == 3:
        return True
    while x < math.sqrt(number):
        x += 1
        if number % x == 0:
            return False
    return True

def processFlag(number, conn=None):
    print("\n entered into processFlag")
    global statusConnect
    global conn_array
    global secret_array
    global username_array
    global contact_array
    global isCLI
    t = int(number[1:])
    if t == 1:  # disconnect
        # in the event of single connection being left or if we're just a
        # client
        if len(conn_array) == 1:
            writeToScreen("Connection closed.", "System")
            dump = secret_array.pop(conn_array[0])
            dump = conn_array.pop()
            try:
                dump.close()
            except socket.error:
                print("\n Issue with someone being bad about disconnecting")
            if not isCLI:
                statusConnect.set("Connect")
                connecter.config(state=NORMAL)
            return

        if conn != None:
            writeToScreen("Connect to " + conn.getsockname()
                          [0] + " closed.", "System")
            dump = secret_array.pop(conn)
            conn_array.remove(conn)
            conn.close()

    if t == 2:  # username change
        name = netCatch(conn, secret_array[conn])
        if(isUsernameFree(name)):
            writeToScreen(
                "User " + username_array[conn] + " has changed their username to " +
name, "System")
            username_array[conn] = name
            contact_array[
                conn.getpeername()[0]] = [conn.getpeername()[1], name]

    # passing a friend who this should connect to (I am assuming it will be
    # running on the same port as the other session)
```

```
    if t == 4:
        data = conn.recv(4)
        data = conn.recv(int(data.decode()))
        Client(data.decode(),
               int([conn.getpeername()[0]][0])).start()

#2nd one
def processUserCommands(command, param):
    """Processes commands passed in via the / text input."""
    print("\n entered into processUserCommands")
    global conn_array
    global secret_array
    global username

    if command == "changeUsername":  # change nickname
        for letter in param[0]:
            print("Letter is ",letter)
            if letter == " " or letter == "\n":
                if isCLI:
                    error_window(0, "Invalid username. No spaces allowed.")
                else:
                    error_window(root, "Invalid username. No spaces allowed.")
                return
        if isUsernameFree(param[0]):
            writeToScreen("Username is being changed to " + param[0], "System")
            print("Conn_array is ", conn_array)
            for conn in conn_array:
                conn.send("-002".encode())
                netThrow(conn, secret_array[conn], param[0])
            username = param[0]
        else:
            writeToScreen(param[0] +
                          " is already taken as a username", "System")
    if command == "disconnect":  # disconnects from current connection
        for conn in conn_array:
            conn.send("-001".encode())
        processFlag("-001")
    if command == "connect":  # connects to passed in host port
        if(options_sanitation(param[1], param[0])):
            Client(param[0], int(param[1])).start()
    if command == "host":  # starts server on passed in port
        if(options_sanitation(param[0])):
            Server(int(param[0])).start()

#3rd one
def isUsernameFree(name):
    print("\n entered into isUsernameFree")
    global username_array
    global username
    print("username_array is ", username_array)
    for conn in username_array:
        if name == username_array[conn] or name == username:
            return False
    return True
```

```
def passFriends(conn):
    print("\n entered into passFriends")
    global conn_array
    for connection in conn_array:
        if conn != connection:
            conn.send("-004".encode())
            conn.send(
                formatNumber(len(connection.getpeername()[0])).encode())  # pass the ip
address
            conn.send(connection.getpeername()[0].encode())
            # conn.send(formatNumber(len(connection.getpeername()[1])).encode())
#pass the port number
            # conn.send(connection.getpeername()[1].encode())


#10th
def client_options_window(master):
    print("\n entered into client_options_window")
    top = Toplevel(master)
    top.title("Connection options")
    top.protocol("WM_DELETE_WINDOW", lambda: optionDelete(top))
    top.grab_set()
    Label(top, text="Server IP:").grid(row=0)
    location = Entry(top)
    location.grid(row=0, column=1)
    location.focus_set()
    Label(top, text="Port:").grid(row=1)
    port = Entry(top)
    port.grid(row=1, column=1)
    go = Button(top, text="Connect", command=lambda:
                client_options_go(location.get(), port.get(), top))
    go.grid(row=2, column=1)

def client_options_go(dest, port, window):
    print("\n entered into client_options_go")
    if options_sanitation(port, dest):
        if not isCLI:
            window.destroy()
        Client(dest, int(port)).start()
    elif isCLI:
        sys.exit(1)

#Check validation of port number.
#8th one
def options_sanitation(por, loc=""):
    print("\n entered into options_sanitation")
    global root
    if version == 2:
        por = unicode(por)
    if isCLI:
        root = 0
    if not por.isdigit():
        error_window(root, "Please input a port number.")
```

```
        return False
    if int(por) < 0 or 65555 < int(por):
        error_window(root, "Please input a port number between 0 and 65555")
        return False
    if loc != "":
        if not ip_process(loc.split(".")):
            error_window(root, "Please input a valid ip address.")
            return False
    return True

#Upto port number.
#Checking port number entered is valid or not.
def ip_process(ipArray):
    """Checks to make sure every section of the ip is a valid number."""
    print("\n entered into ip_process")
    if len(ipArray) != 4:
        return False
    for ip in ipArray:
        if version == 2:
            ip = unicode(ip)
        if not ip.isdigit():
            return False
        t = int(ip)
        if t < 0 or 255 < t:
            return False
    return True

#6th one
def server_options_window(master):
    """Launches server options window for getting port."""
    print("\n entered into server_options_window")
    top = Toplevel(master)
    top.title("Please enter Connection options")
    top.grab_set()
    print("Top grab_set is ",top.grab_set)
    top.protocol("WM_DELETE_WINDOW", lambda: optionDelete(top))
    Label(top, text="Port No(4 digit):").grid(row=0)
    port = Entry(top)
    port.grid(row=0, column=1)
    port.focus_set()
    print("top is ",top)
    go = Button(top, text="Start", command=lambda:
                server_options_go(port.get(), top))
    go.grid(row=1, column=1)

#7th one
def server_options_go(port, window):
    print("\n entered into server_options_go")
    if options_sanitation(port):
        if not isCLI:
            window.destroy()
        Server(int(port)).start()
    elif isCLI:
        sys.exit(1)
```

```
def username_options_window(master):
    print("\n entered into username_options_window")
    top = Toplevel(master)
    top.title("Username options")
    top.grab_set()
    Label(top, text="Username:").grid(row=0)
    name = Entry(top)
    name.focus_set()
    name.grid(row=0, column=1)
    go = Button(top, text="Change", command=lambda:
            username_options_go(name.get(), top))
    go.grid(row=1, column=1)


def username_options_go(name, window):
    print("\n entered into username_options_go")
    """Processes the options entered by the user in the
    server options window.
    """
    processUserCommands("nick", [name])
    window.destroy()

#9th one
def error_window(master, texty):
    """Launches a new window to display the message texty."""
    print("\n entered into error_window")
    global isCLI
    if isCLI:
        writeToScreen(texty, "System")
    else:
        window = Toplevel(master)
        print("window is ",window)
        window.title("ERROR")
        window.grab_set()
        Label(window, text=texty).pack()
        go = Button(window, text="Understood", command=window.destroy)
        go.pack()
        go.focus_set()

def optionDelete(window):
    print("\n Entered into optionDelete")
    connecter.config(state=NORMAL)
    window.destroy()

def contacts_connect(item):
    print("\n Entred into contacts_connect")
    """Establish a connection between two contacts."""
    Client(item[1], int(item[2])).start()

def dump_contacts():
    """Saves the recent chats to the persistent file contacts.dat."""
    print("\n entred into dump_contacts")
```

```
    global contact_array
    try:
        filehandle = open("data\\contacts.dat", "w")
    except IOError:
        print("\n Can't dump contacts.")
        return
    for contact in contact_array:
        filehandle.write(
            contact + " " + str(contact_array[contact][0]) + " " +
            contact_array[contact][1] + "\n")
    filehandle.close()

def placeText(text):
    print("\n entered into placeText")
    global conn_array
    global secret_array
    global username
    print(conn_array)
    writeToScreen(text, username)
    for person in conn_array:
        netThrow(person, secret_array[person], text)

#4th one
def writeToScreen(text, username=""):
    """Places text to main text body in format "username: text"."""
    print("\n entered into writeToScreen")
    global main_body_text
    global isCLI
    if isCLI:
        print("Here is isCli",isCLI)
        if username:
            print(username + ": " + text)
        else:
            print(text)
    else:
        main_body_text.config(state=NORMAL)
        main_body_text.insert(END, '\n')
        if username:
            #main_body_text.insert(END, "Vinit" + ": ")
            main_body_text.insert(END, username + ": ")
        main_body_text.insert(END, text)
        main_body_text.yview(END)
        main_body_text.config(state=DISABLED)

#1st one
def processUserText(event):
    print("\n entered into processUserText")
    """Takes text from text bar input and calls processUserCommands if it
    begins with '/'.
    """
    data = text_input.get()
    print("Entered data is : ",data)
    if data[0] != "/":
        #This / is acting like enter
```

```
            placeText(data)
        else:
            if data.find(" ") == -1:
                command = data[1:]
            else:
                command = data[1:data.find(" ")]
            print("Command is :",command)
            print("data[] is : ", data[data.find(" ")])
            params = data[data.find(" ") + 1:].split(" ")
            print("Complete data[] is :",data[data.find(" ") + 1:])
            print("Params are : ",params)
            processUserCommands(command, params)
        text_input.delete(0, END)

    def changeUsernameText(event):
        print("\n entered into chnageUsernameText")
        """Takes text from text bar input and calls processUserCommands if it
        begins with '/'.
        """
        data = text_input2.get()
        print("Entered data is : ",data)
        if data[0] == "/":
            #This / is acting like enter
            placeText(data)
        else:
            if data.find(" ") == -1:
                command = data[1:]
            else:
                command = data[1:data.find(" ")]
            print("Command is :",command)
            print("data[] is : ", data[data.find(" ")])
            params = data[data.find(" ") + 1:].split(" ")
            print("Complete data[] is :",data[data.find(" ") + 1:])
            print("Params are : ",params)
            command = "changeUsername"
            processUserCommands(command, params)
        text_input2.delete(0, END)


    def processUserInput(text):
        print("\n entered into processUserInput")
        """ClI version of processUserText."""
        if text[0] != "/":
            placeText(text)
        else:
            if text.find(" ") == -1:
                command = text[1:]
            else:
                command = text[1:text.find(" ")]
            params = text[text.find(" ") + 1:].split(" ")
            processUserCommands(command, params)

    class Server (threading.Thread):
        print("\n entered into Server")
```

```
    "A class for a Server instance."""
    def __init__(self, port):
        print("\n Entered into Server init")
        threading.Thread.__init__(self)
        #Threading in python is used to run multiple threads (tasks, function calls) at the
same time
        self.port = port

    def run(self):
        print("\n entered into run Server")
        global conn_array
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.bind(('', self.port))
        print("Enterd intp run Server 2")

        if len(conn_array) == 0:
            writeToScreen(
                "Waiting for connections on port: " +
                str(self.port), "System")
        s.listen(1)
        global conn_init
        conn_init, addr_init = s.accept()
        serv = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        serv.bind(('', 0))  # get a random empty port
        serv.listen(1)

        portVal = str(serv.getsockname()[1])
        if len(portVal) == 5:
            conn_init.send(portVal.encode())
        else:
            conn_init.send(("0" + portVal).encode())

        conn_init.close()
        conn, addr = serv.accept()
        conn_array.append(conn)  # add an array entry for this connection
        writeToScreen("Connected by " + str(addr[0]), "System")

        global statusConnect
        statusConnect.set("Disconnect")
        connecter.config(state=NORMAL)

        # create the numbers for my encryption
        prime = random.randint(1000, 9000)
        while not isPrime(prime):
            prime = random.randint(1000, 9000)
        base = random.randint(20, 100)
        a = random.randint(20, 100)
        # send the numbers (base, prime, A)
        conn.send(formatNumber(len(str(base))).encode())
        conn.send(str(base).encode())

        conn.send(formatNumber(len(str(prime))).encode())
        conn.send(str(prime).encode())
```

```
            conn.send(formatNumber(len(str(pow(base, a) % prime))).encode())
            conn.send(str(pow(base, a) % prime).encode())

            # get B
            data = conn.recv(4)
            data = conn.recv(int(data.decode()))
            b = int(data.decode())

            global secret_array
            secret = pow(b, a) % prime
            secret_array[conn] = secret

            conn.send(formatNumber(len(username)).encode())
            conn.send(username.encode())

            data = conn.recv(4)
            data = conn.recv(int(data.decode()))
            if data.decode() != "Self":
                username_array[conn] = data.decode()
                contact_array[str(addr[0])] = [str(self.port), data.decode()]
            else:
                username_array[conn] = addr[0]
                contact_array[str(addr[0])] = [str(self.port), "No_nick"]

            passFriends(conn)
            threading.Thread(target=Runner, args=(conn, secret)).start()
            Server(self.port).start()


    class Client (threading.Thread):
        print("\n entered into Client")
        """A class for a Client instance."""
        def __init__(self, host, port):
            print("\n Entered into Client")
            threading.Thread.__init__(self)
            self.port = port
            self.host = host

        def run(self):
            print("\n entered into run Client")
            global conn_array
            global secret_array
            conn_init = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            conn_init.settimeout(5.0)
            try:
                conn_init.connect((self.host, self.port))
            except socket.timeout:
                writeToScreen("Timeout issue. Host possible not there.", "System")
                connecter.config(state=NORMAL)
                raise SystemExit(0)
            except socket.error:
                writeToScreen(
                    "Connection issue. Host actively refused connection.", "System")
                connecter.config(state=NORMAL)
```

```
            raise SystemExit(0)
        porta = conn_init.recv(5)
        porte = int(porta.decode())
        conn_init.close()
        conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        conn.connect((self.host, porte))

        writeToScreen("Connected to: " + self.host +
                      " on port: " + str(porte), "System")

        global statusConnect
        statusConnect.set("Disconnect")
        connecter.config(state=NORMAL)

        conn_array.append(conn)
        data = conn.recv(4)
        data = conn.recv(int(data.decode()))
        base = int(data.decode())
        data = conn.recv(4)
        data = conn.recv(int(data.decode()))
        prime = int(data.decode())
        data = conn.recv(4)
        data = conn.recv(int(data.decode()))
        a = int(data.decode())
        b = random.randint(20, 100)
        conn.send(formatNumber(len(str(pow(base, b) % prime))).encode())
        conn.send(str(pow(base, b) % prime).encode())
        secret = pow(a, b) % prime
        secret_array[conn] = secret

        conn.send(formatNumber(len(username)).encode())
        conn.send(username.encode())

        data = conn.recv(4)
        data = conn.recv(int(data.decode()))
        if data.decode() != "Self":
            username_array[conn] = data.decode()
            contact_array[
                conn.getpeername()[0]] = [str(self.port), data.decode()]
        else:
            username_array[conn] = self.host
            contact_array[conn.getpeername()[0]] = [str(self.port), "No_nick"]
        threading.Thread(target=Runner, args=(conn, secret)).start()


def Runner(conn, secret):
    print("\n entered into Runner")
    global username_array
    while 1:
        data = netCatch(conn, secret)
        if data != 1:
            writeToScreen(data, username_array[conn])
```

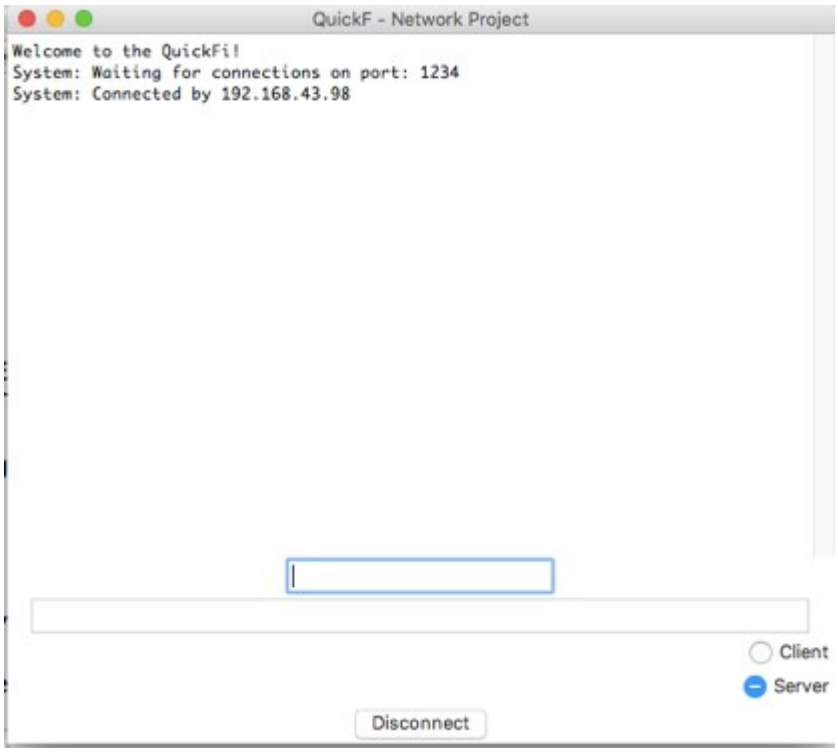
```
def QuickClient():
    """Menu window for connection options."""
    print("\n entered into QuickClient")
    window = Toplevel(root)
    window.title("Connection options")
    window.grab_set()
    Label(window, text="Server IP:").grid(row=0)
    destination = Entry(window)
    destination.grid(row=0, column=1)
    go = Button(window, text="Connect", command=lambda:
            client_options_go(destination.get(), "9999", window))
    go.grid(row=1, column=1)


def QuickServer():
    print("\n entered into QuickServer")
    """Quickstarts a server."""
    Server(9999).start()

#5th one
def connects(clientType):
    print("\n entered into connects")
    global conn_array
    connecter.config(state=DISABLED)
    if len(conn_array) == 0:
        print("Client type is ",clientType)
        if clientType == 0:
            client_options_window(root)
        if clientType == 1:
            server_options_window(root)
    else:
        # connecter.config(state=NORMAL)
        for connection in conn_array:
            connection.send("-001".encode())
        processFlag("-001")


def toOne():
    print("\n entered into toOne")
    global clientType
    clientType = 0


def toTwo():

    global clientType
    clientType = 1

    print("\n entered into toTwo")


if len(sys.argv) > 1 and sys.argv[1] == "-cli":
    print("Starting command line chat")
```

```
else:
    root = Tk()
    root.title("QuickF - Network Project")
    print("else part of toTow")
    #root.iconbitmap(r'C:\Users\Yash\Downloads\Logo2.ico')

    menubar = Menu(root)
    print(menubar)

    file_menu = Menu(menubar, tearoff=0)
    print(file_menu)
    file_menu.add_command(label="Change username",
                          command=lambda: username_options_window(root))
    file_menu.add_command(label="Exit", command=lambda: root.destroy())
    print(file_menu)
    menubar.add_cascade(label="File", menu=file_menu)

    connection_menu = Menu(menubar, tearoff=0)
    print("connection_menu")
    connection_menu.add_command(label="Quick Connect", command=QuickClient)
    connection_menu.add_command(
       label="Connect on port", command=lambda: client_options_window(root))
    connection_menu.add_command(
       label="Disconnect", command=lambda: processFlag("-001"))
    print(connection_menu)
    print("Over")

    menubar.add_cascade(label="Connect", menu=connection_menu)
    root.config(menu=menubar)
    print("mail_body")
    print("-----------------------")
    main_body = Frame(root, height=20, width=50)
    main_body_text = Text(main_body)
    body_text_scroll = Scrollbar(main_body)
    main_body_text.focus_set()
    body_text_scroll.pack(side=RIGHT, fill=Y)
    main_body_text.pack(side=LEFT, fill=Y)
    body_text_scroll.config(command=main_body_text.yview)
    main_body_text.config(yscrollcommand=body_text_scroll.set)
    main_body.pack()


    main_body_text.insert(END, "Welcome to the QuickFi!")
    main_body_text.config(state=DISABLED)

    #   changeUsername here
    text_input2 = Entry(root, width=20)
    text_input2.bind("<Return>", changeUsernameText)
    text_input2.pack()


    text_input = Entry(root, width=60)
    text_input.bind("<Return>", processUserText)
    text_input.pack()
```

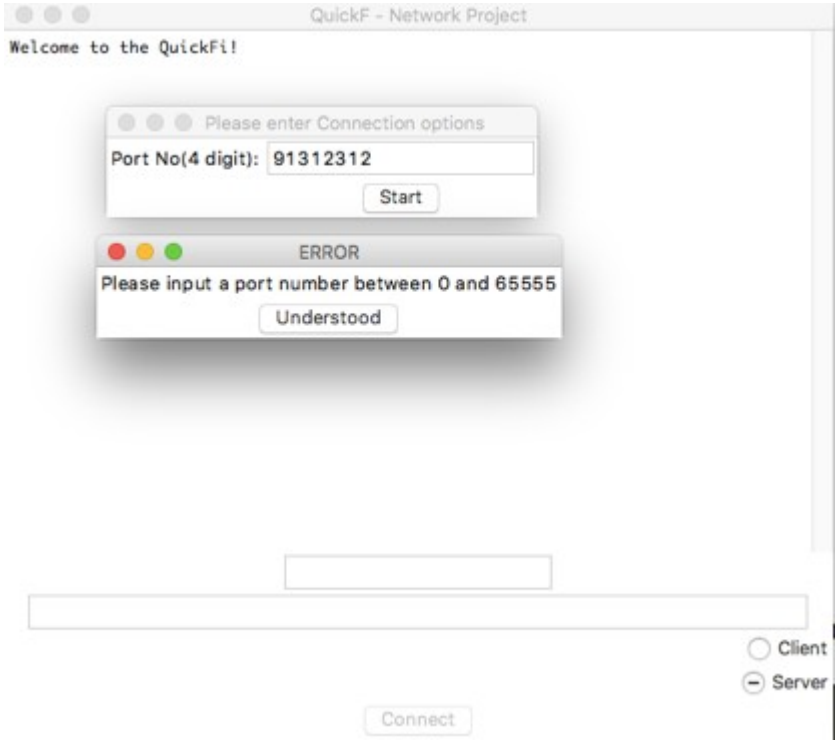
```
statusConnect = StringVar()
statusConnect.set("Connect")
clientType = 1
Radiobutton(root, text="Client", variable=clientType,
        value=0, command=toOne).pack(anchor=E)
Radiobutton(root, text="Server", variable=clientType,
        value=1, command=toTwo).pack(anchor=E)
connecter = Button(root, textvariable=statusConnect,
            command=lambda: connects(clientType))
connecter.pack()

root.mainloop()

dump_contacts()
```

RESULTS OF SERIAL IMPLEMENTATION

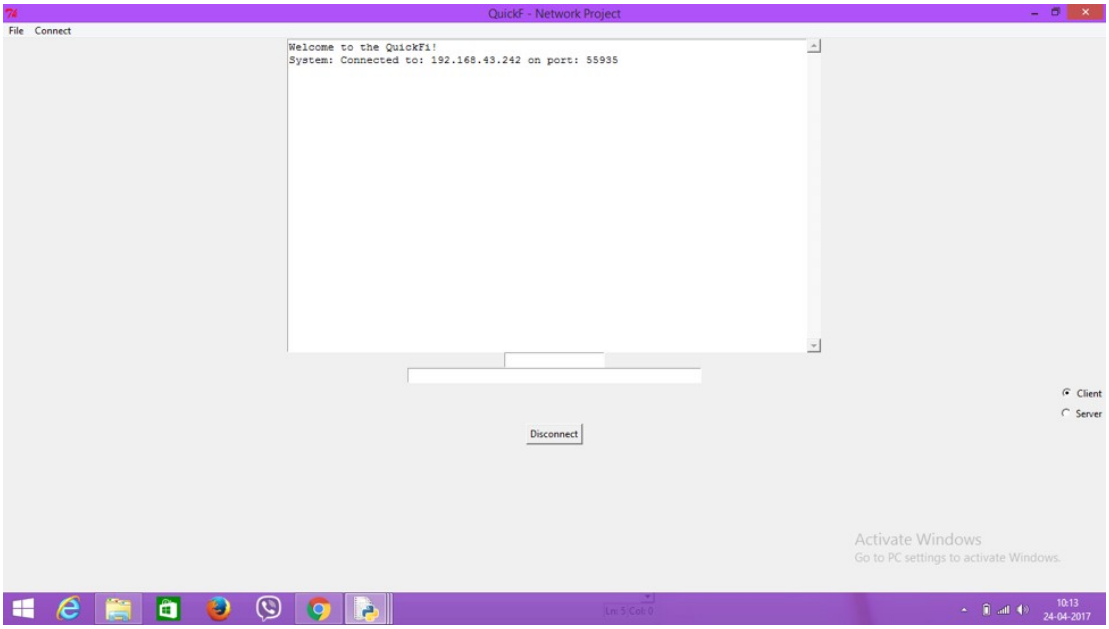
Acting as a server :
Entered port no - 1234.
Showing error messages on entering wrong port no :



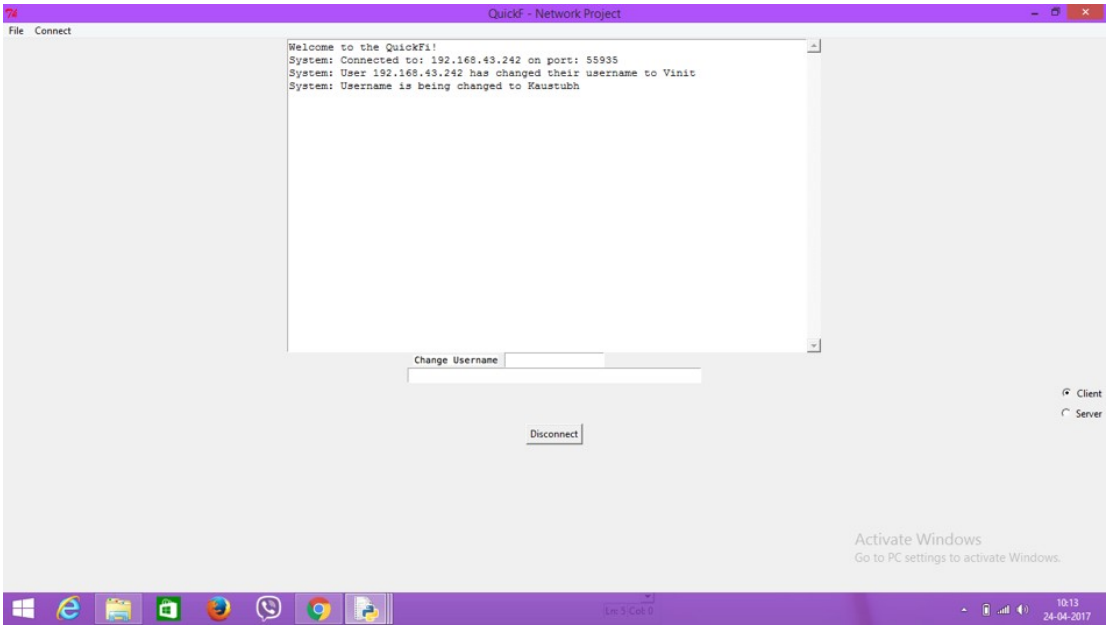
Showing error messages on entering wrong port no :



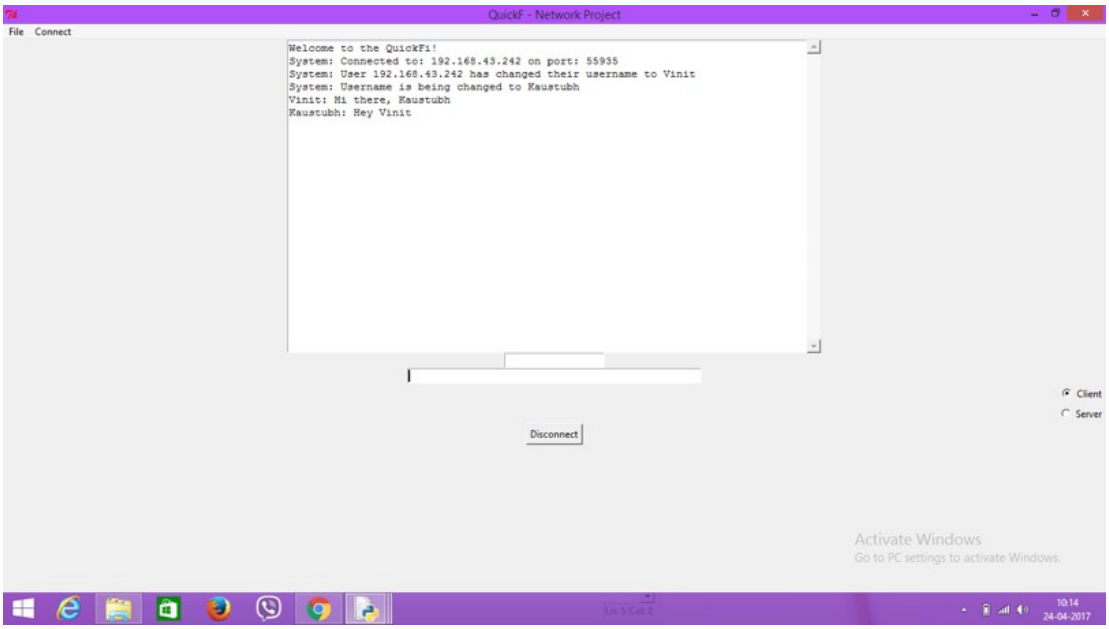
Acting as a client :
Connected to the server. Has to enter two things:
1.Port No
2.IP Address



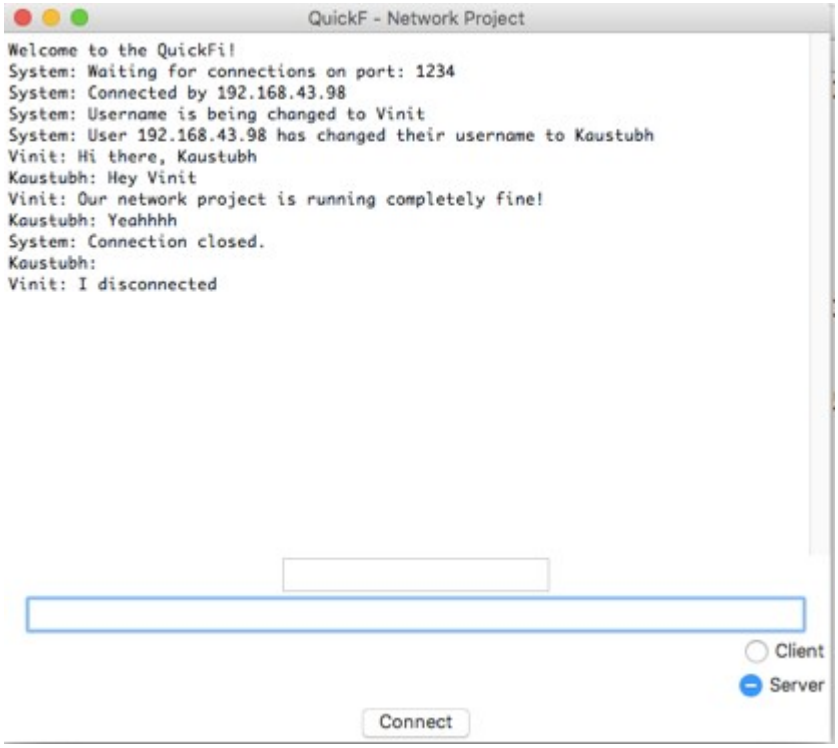
Changing Username :
Give value in the field of “Change Username”
Also displaying, about the another user has changed its username.



Receiving messages by client side :



Disconnecting from server side :



After disconnecting, messages are not received by the client :