Aleena Anna Plessey
21BCE5767


# RTE Socket Programming

# Calculator, Chat and Multiuser Chat

*Submitted by Aleena Anna Plessey*

Reg no: 21BCE5767

Course Program: B.Tech

Course code: BCSE308L

Course Title: Computer Networks

Batch:2021-25

*Submitted to:* Dr. Swaminathan A

*Submitted on 07/06/2023*

**Problem Statement:**

5.1 Write a TCP Client Server program to perform Mathematical Calculations.

5.2 Write a UDP Client Server program to perform a simple chat.

5.3 Using the given components and devices, such as routers, switches and cat6 cables, create a of any topology. Configure the router and configure the IP for all n nodes connected in the network using dynamic DHCP. Write a UDP Client Server program to perform a multiuser chat program.

**Aim and Objective:**

a) To create a TCP Client Server program

b) To create a UDP client server program

c)To write and complete a UDP client server program to make a multiuser chat program using all the computers connected in a network.

**5.1 TCP Client Server program to perform Mathematical Calculations.**

**Code:**

**Client side:**

```python
import socket
def start_client():
    host="127.0.0.1"
    port=12345

    client_socket=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    client_socket.connect((host,port))

    print("Connected to server.")

    while True:
        expression=input("Enter an expression to calculate (q to leave):")
        if expression=="q":
            break
        client_socket.send(expression.encode())
        result=client_socket.recv(1024).decode()
        print("Result:",result)
    client_socket.close()
    print("Connect closed")
if __name__=="__main__":
    start_client()


```

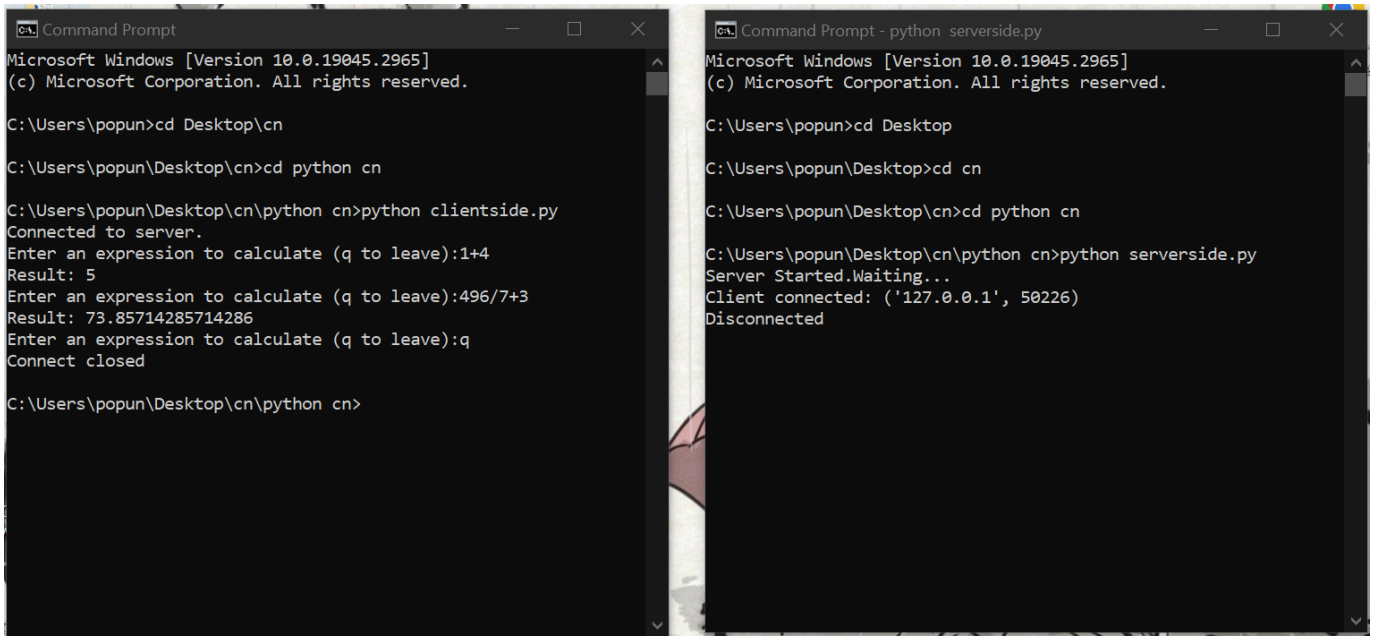
**Server side:**

```python
import socket
def calculate(expression):
    try:
        result=eval(expression)
        return str(result)
    except:
        return "ERROR"
def start_server():
    host="127.0.0.1"
    port=12345

    server_socket=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    server_socket.bind((host,port))
    server_socket.listen(1)
    print("Server Started.Waiting...")
    while True:
        client_socket,addr=server_socket.accept()
        print("Client connected:",addr)

        while True:
            data=client_socket.recv(1024).decode()
            if not data:
                break
            result=calculate(data)
            client_socket.send(result.encode())

        client_socket.close()
        print("Disconnected")

if __name__=="__main__":
    start_server()
```

1)Run the server-side code first to start the server.

2)Run the client-side server and input the expressions the output will be given.

**Output:**




## 5.2 UDP Client Server program to perform a simple chat.

**Code:**

**Server:**

```python
import socket
SERVER_IP = 'localhost'
SERVER_PORT = 12345

server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind((SERVER_IP, SERVER_PORT))

print("UDP server is running on", SERVER_IP + ":", SERVER_PORT)

while True:
    data, client_address = server_socket.recvfrom(1024)
    message = data.decode()
    print("[Received from", client_address[0] + ":", client_address[1], "]:", message)
    response = input("Enter response (or 'bye' to quit): ")
    server_socket.sendto(response.encode(), client_address)

    if response.lower() == "bye":
        break

server_socket.close()
```

**Client:**

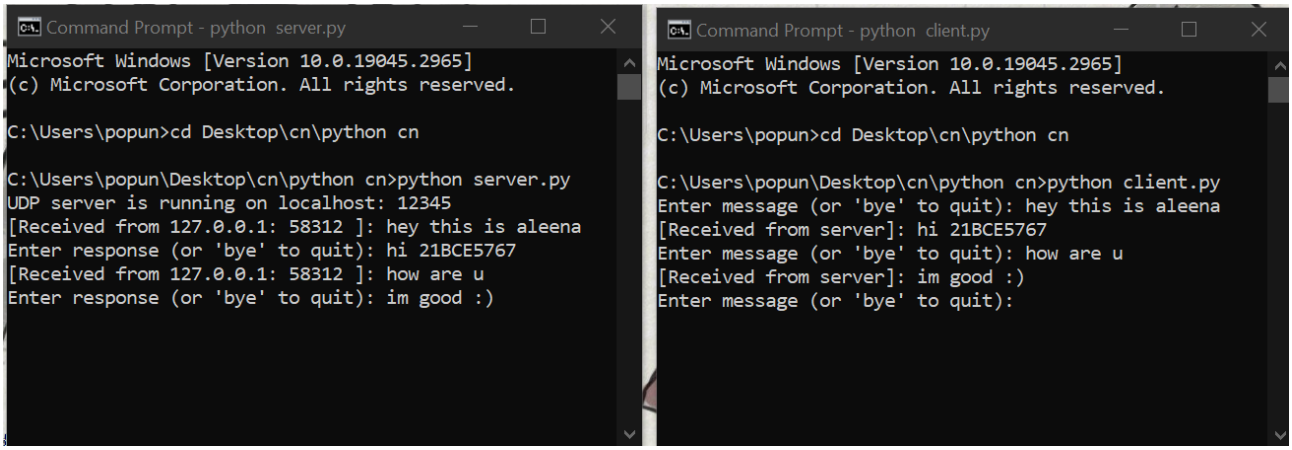
```python
import socket
SERVER_IP = 'localhost'
SERVER_PORT = 12345
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
while True:
    message = input("Enter message (or 'bye' to quit): ")
    client_socket.sendto(message.encode(), (SERVER_IP, SERVER_PORT))

    if message.lower() == "bye":
        break

    data, _ = client_socket.recvfrom(1024)
    response = data.decode()
    print("[Received from server]:", response)

client_socket.close()

```

1)Run the server-side code first to start the server.

2)Run the client-side server and send the first message. It will be reflected and the server side will get the message.

3)Then the server side can send the message which will be received in the client side.

**Output:**




**5.3 UDP Client Server program to perform a multiuser chat program.**

**Code:**

**Server:**

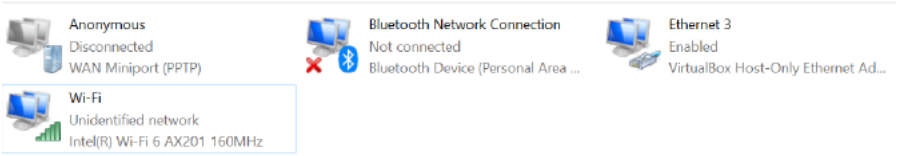
```python
import socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_address = ('192.168.1.6', 8888)
server_socket.bind(server_address)
print('UDP server listening on {}:{}'.format(*server_address))
while True:
message, client_address = server_socket.recvfrom(1024)
print('Received from {}: {}'.format(client_address, message.decode()))
response = 'Message received: {}'.format(message.decode())
server_socket.sendto(response.encode(), client_address)
```

**Client:**

```python
import socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_address = ('192.168.1.6', 8888)
while True:
message = input("Enter a message: ")
client_socket.sendto(message.encode(), server_address)
response, server = client_socket.recvfrom(1024)
print('Received from {}: {}'.format(server, response.decode()))
client_socket.close()
```

**Output:**

1)Connect to the router and connect the system to the router




2)Verify that the network is connected using ping ipaddress

3)Then assign unique IP addresses to each of the computers manually

4)Run the code.


**Result:**

We were able to successfully create a TCP client server program, a UDP client server single user chat and a multiuser chat as well.