



PROJECT: TCP Chat Room Using Socket programming

Computer Communication Network
Department of ECE
PES University Bangalore
Faculty in charge: Prof.Prajeesha

Team Members :

Jacob V Sanoj [PES1UG20EC083]
Kaushik Narayan [PES1UG20EC093]
Meghana M [PES1UG20EC114]

Sem: 5th sem
Section: B

Sl no	Description	Page no
1.	Problem Statement	
2.	Introduction	
3.	Procedure	
4.	Code	
5.	Result	
6.	Conclusion	
7.	References	

INTRODUCTION

Chat room is a space for people across the globe to communicate with each other about various topics. The topics can range from anything such as app development to any recipes made in the kitchen. Chat rooms are great platforms for learning how to communicate.

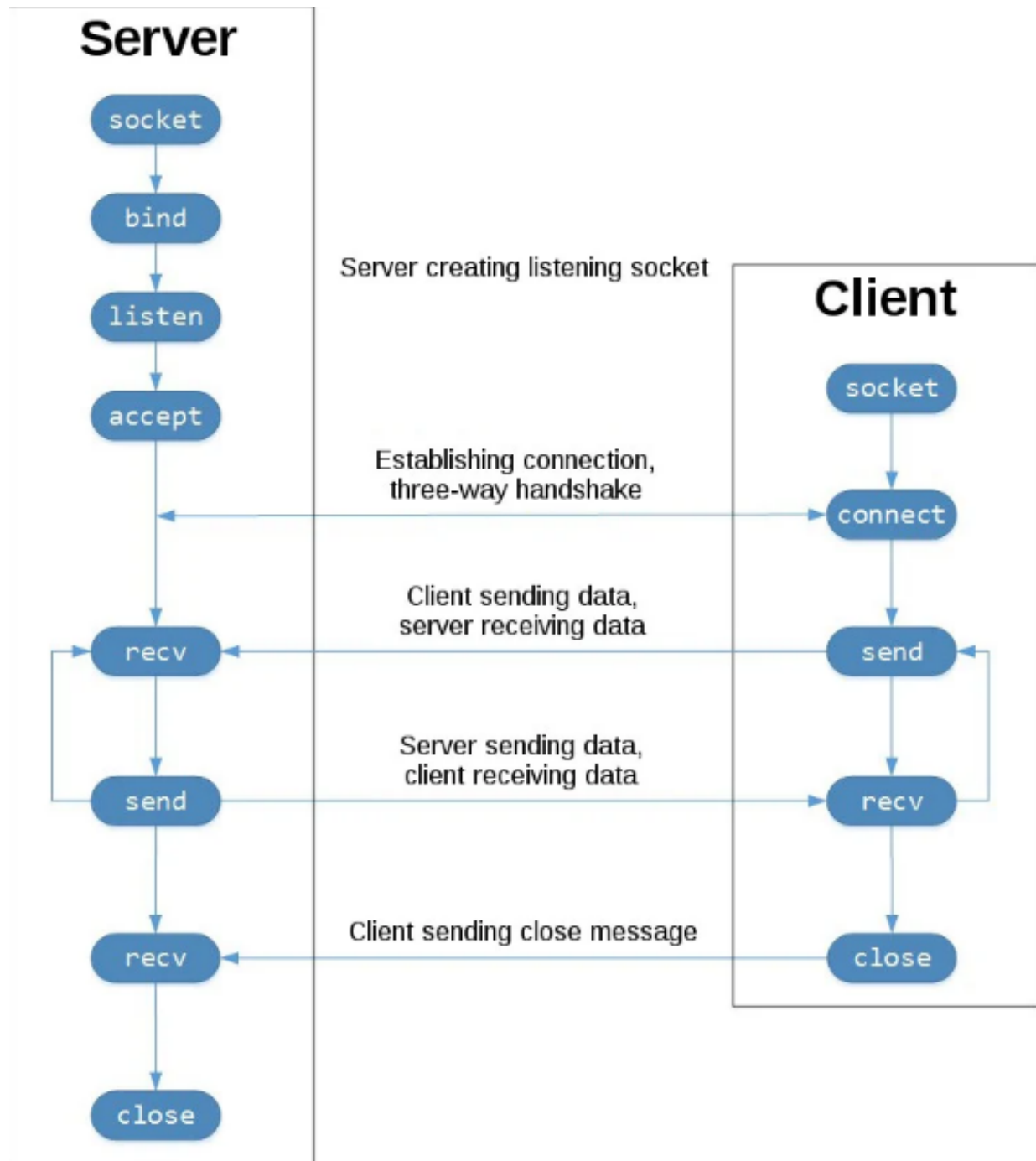
SOCKET PROGRAMMING

A socket is a communications connection point (endpoint) that you can name and address in a network. Socket programming shows how to use socket APIs to establish communication links between remote and local processes.

The processes that use a socket can reside on the same system or different systems on different networks. Sockets are useful for both stand-alone and network applications. Sockets allow you to exchange information between processes on the same machine or across a network, distribute work to the most efficient machine, and they easily allow access to centralised data. Socket application program interfaces (APIs) are the network standard for TCP/IP.

The type of sockets that are most suitable are **TCP sockets** as the TransmissionControl Protocol (TCP):

- **Is reliable:** packets dropped in the network are detected and retransmitted by the sender.
- **Has in-order data delivery:** data is read by your application in the order it was written by the sender.



Procedure

Implementing The Server

Now let's start by implementing the server first. For this we will need to import two libraries, namely socket and threading. The first one will be used for the network connection and the second one is necessary for performing various tasks at the same time.

```
import threading
import socket
```

The next task is to define our connection data and to initialise our socket. We used the localhost address (127.0.0.1) and the port 55555. The port is actually irrelevant but you have to make sure that the port you are using is free and not reserved. If you are running this script on an actual server, specify the IP-address of the server as the host.

```
host = '127.0.0.1' # localhost
port = 55555

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind((host, port))
server.listen()

clients = []
nicknames = []
```

When we define our socket, we need to pass two parameters. These define the type of socket we want to use.

AF_INET indicates that we are using an internet socket rather than an unix socket. The second parameter stands for the protocol we want to use.

SOCK_STREAM indicates that we are using TCP and not UDP.

After defining the socket, we bind it to our host and the specified port by passing a tuple that contains both values. We then put our server into listening mode, so that it waits for clients to connect. At the end we create two empty lists, which we will use to store the connected clients and their nicknames later on.

```
def broadcast(message):          # broadcast message to all
    clients connected to the server
    for client in clients:
        client.send(message)
```

We define a function to send a message to each client that is connected and therefore in the clients list.

The following function will be responsible for handling messages from the clients.

```
def handle(client):
    while True:
        try:
            message = client.recv(1024)      # receive
            message from the client at a rate of 1024 bytes
            broadcast(message)                #Try to receive
            message from the client and broadcast it to all other
            clients
        except:
            index = clients.index(client)
            clients.remove(client)
            client.close()
            nickname = nicknames[index]
            broadcast(f'{nickname} left the
chat!'.encode('ascii'))                    #Any error in broadcasting
            message will result in the client being removed from the
            list
            nicknames.remove(nickname)
```

```
break
```

The function is running in a while-loop. The function accepts a client as a parameter. Everytime a client connects to our server we run this function for it and it starts an endless loop.

It receives a message from the client and broadcasts it to all connected clients.

Now if for some reason there is an error with the connection to this client, we remove it and its nickname, close the connection and broadcast that this client has left the chat. After that we break the loop and this thread comes to an end.

```
def receive():
    while True:
        client, address = server.accept()      #Accept
connection
        print(f"Connected with {str(address)}")

        client.send('NICK'.encode('ascii'))    #Send
nickname to the client
        nickname = client.recv(1024).decode('ascii')
#Receive nickname from the client

        nicknames.append(nickname)
        clients.append(client)

        print(f"Nickname of the client is {nickname}!")
        broadcast(f"{nickname} joined the
chat!".encode('ascii')) #Broadcast nickname to all other
clients that the client with that nickname has joined the
chat

        client.send('Connected to the
server!'.encode('ascii'))
```

```
thread = threading.Thread(target=handle,  
args=(client,)) # Each client will have its own thread  
thread.start()
```

When we are ready to run our server, we will execute this receive function. It also starts an endless while-loop which constantly accepts new connections from clients.

Once a client is connected it sends the string 'NICK' to it, which will tell the client that its nickname is requested.

After that it waits for a response (which hopefully contains the nickname) and appends the client with the respective nickname to the lists.

After that, we print and broadcast this information.

Finally, we start a new thread that runs the previously implemented handling function for this particular client.

```
print("Server is listening...")  
receive()
```


Implementing The Client

We shall import the same libraries as we did for the server for the client.

```
import socket
import threading
```

The first steps of the client are to choose a nickname and to connect to our server. We will need to know the exact address and the port at which our server is running.

```
nickname = input("Choose your nickname: ")

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('127.0.0.1', 55555))
```

Here we connect to an existing server. A client needs to have two threads that are running at the same time.

The first one will constantly receive data from the server.

The second one will send our own messages to the server.

Below is the receive function of the client.

```
def receive():
    while True:
        try:
            message = client.recv(1024).decode('ascii')
            #Receive message from the server
            if message == 'NICK':
                client.send(nickname.encode('ascii')) #Send
                #nickname to the server
            else:
                print(message)
        except:
```

```
print("An error occurred!")
client.close()
break
```

It constantly tries to receive messages and to print them onto the screen using an infinite while loop. If the message is 'NICK' however, it doesn't print it but it sends its nickname to the server. In case there is some error, we close the connection and break the loop.

```
def write():
    while True:
        message = f'{nickname}: {input("")}' # Define
        message to be sent to the server
        client.send(message.encode('ascii')) #Send message
        to the server
```

The writing function is quite a short one. It also runs in an endless loop which is always waiting for input from the user. Once it gets some, it combines it with the nickname and sends it to the server. Now we create 2 threads.

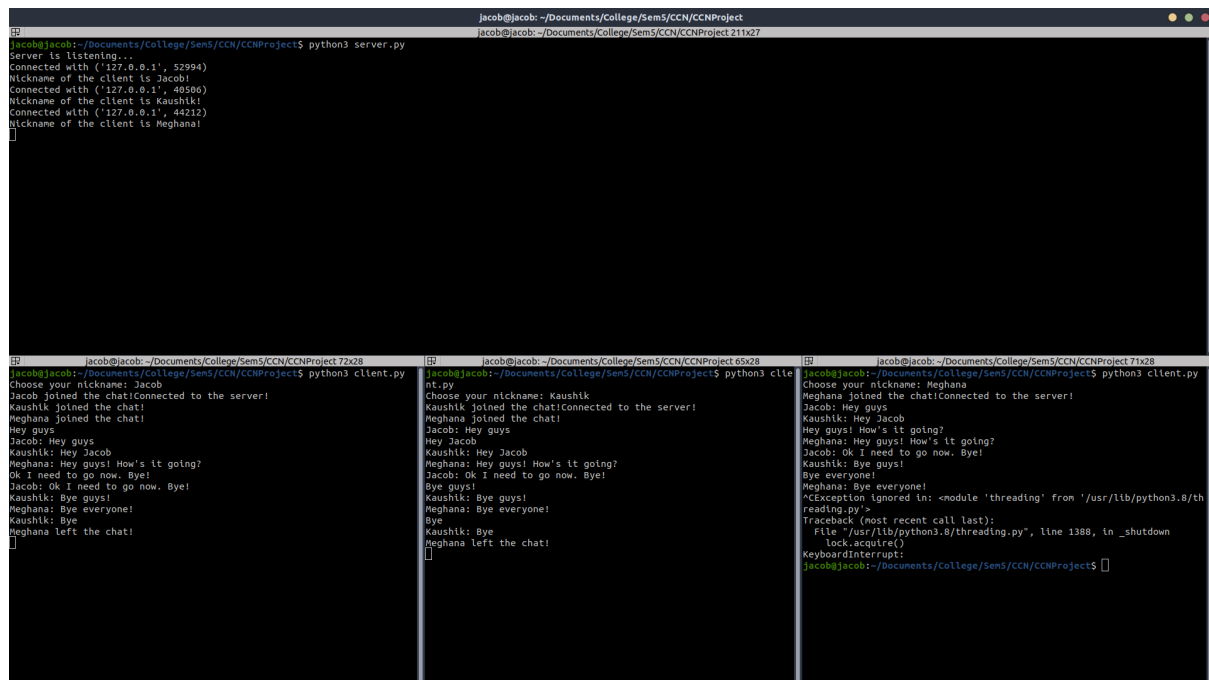
One thread is to run the receive function on an infinite while-loop.

The other is for the write function which works on another infinite while-loop.

```
receive_thread = threading.Thread(target=receive)
#Create a thread for receiving messages
receive_thread.start()

write_thread = threading.Thread(target=write) #Create a
thread for sending messages
write_thread.start()
```

Results



```
jacob@jacob: ~/Documents/College/Sem5/CCN/CCNProject
jacob@jacob: ~/Documents/College/Sem5/CCN/CCNProject 21x27

jacob@jacob:~/Documents/College/Sem5/CCN/CCNProject$ python3 server.py
Server is listening...
Connected with ('127.0.0.1', 52994)
Nickname of the client is Jacob!
Connected with ('127.0.0.1', 40506)
Nickname of the client is Kaushik!
Connected with ('127.0.0.1', 44212)
Nickname of the client is Meghana!

jacob@jacob:~/Documents/College/Sem5/CCN/CCNProject 72x28
jacob@jacob:~/Documents/College/Sem5/CCN/CCNProject$ python3 client.py
Choose your nickname: Jacob
Jacob joined the chat!connected to the server!
Kaushik joined the chat!
Meghana joined the chat!
Hey guys
Jacob: Hey guys
Kaushik: Hey Jacob
Meghana: Hey guys! How's it going?
Ok I need to go now. Bye!
Jacob: Ok I need to go now. Bye!
Kaushik: Bye guys!
Meghana: Bye everyone!
Kaushik: Bye
Meghana left the chat!

jacob@jacob:~/Documents/College/Sem5/CCN/CCNProject 65x28
jacob@jacob:~/Documents/College/Sem5/CCN/CCNProject$ python3 client.py
Choose your nickname: Kaushik
Kaushik joined the chat!connected to the server!
Meghana joined the chat!
Jacob: Hey guys
Hey Jacob
Kaushik: Hey Jacob
Meghana: Hey guys! How's it going?
Jacob: Ok I need to go now. Bye!
Bye guys!
Kaushik: Bye guys!
Meghana: Bye everyone!
Bye
Kaushik: Bye
Meghana left the chat!

jacob@jacob:~/Documents/College/Sem5/CCN/CCNProject 71x28
jacob@jacob:~/Documents/College/Sem5/CCN/CCNProject$ python3 client.py
Choose your nickname: Meghana
Meghana joined the chat!connected to the server!
Jacob: Hey guys
Kaushik: Hey Jacob
Hey guys! How's it going?
Meghana: Hey guys! How's it going?
Jacob: Ok I need to go now. Bye!
Kaushik: Bye guys!
Bye everyone!
Meghana: Bye everyone!
*Exception ignored in: <module 'threading' from '/usr/lib/python3.8/threading.py'>
Traceback (most recent call last):
  File "/usr/lib/python3.8/threading.py", line 1388, in _shutdown
    lock.acquire()
KeyboardInterrupt:
jacob@jacob:~/Documents/College/Sem5/CCN/CCNProject$
```

Conclusion

We developed a chat room application in python using sockets and threads. This software is extremely light as is the bare concepts of how chatting systems like WhatsApp, Telegram and others work. Because of how light the system is, it is easy for anyone to join the chat if they are in the same network with a few steps. We learnt how TCP communication works and how to program sockets. We learnt also how to multithread our application so that it worked smoothly and we were able to add multiple users into the chat room.

References

- <https://www.neuralnine.com/tcp-chat-in-python/>
- <https://www.ibm.com/docs/en/i/7.1?topic=communications-socket-programming>