



**A PROJECT REPORT**  
**On**  
**MULTI-USER CHAT SERVER**

**Team Members:**  
**MOHAMMAD SAIF**  
**SAI HARIKA PALURI**  
**HERLEEN KAUR SANHOTRA**

**Professor:**  
**Dr. Dewan T. Ahmed**  
**TA:**  
**Sahithi Priya Gutta**

## **ABSTRACT**

Socket programming is the way of connecting two nodes on the network. To establish a connection between a client and a server, the server first needs to open a socket connection and bind its address to the socket. The next follow up step would be it should be open to accepting incoming connections. After the server is up and running a client can now send a connection request to a server. It's only after the server accepts this client connection that a client can send and receive data to and from the server. This is the fundamental process for connection establishment. Our goal in this project is to build a multi-user chat room where any user can leave and join the chat room at any time. We implemented this functionality by using python as our primary language. We deployed two Python scripts that are the Client and the Server. By enabling multi-threading in our python script, we were able to account for multiple users joining a chat room.

## TABLE OF CONTENTS

1.1	Introduction	4
1.2	Sockets	4
1.3	Multithreading	4
1.4	Server Program	4
1.5	Client Program	5
1.6	Establishing a TCP connection	5
1.7	Code for Server	5-6
1.8	Code for Client	7
1.9	Implementation of Multiuser Client-Server chat in Terminal	7-9
1.9.1	Run Server program to open connection	7
1.9.2	Run Client program	8
1.9.3	Multi-user chat room	8-9
1.10	Flow of Server and Client Program	9
1.11	References	10

## TABLE OF FIGURES

1(a)	Code snippet for accepting new socket connections	5
1(b)	Server Code Snippet for accepting and broadcasting messages	6
2(a)	Client function to send message	7
2(b)	Client function to receive message	7
3(a)	Server running on host	7
3(b)	User 2 perspective	8
3(c)	User 1 perspective	8
3(d)	Multiple clients connecting to the server	8
3(e)	Multiple clients chat room	8
3(f)	Multiple clients chat room on local host	9
4(a)	Flow diagram of Client Server connection	9

## 1.1 Introduction

A multi-user chat room is a technology that enables multiple clients to join a chat room and communicate with one another on the same connection. In our project, we built a multi-user chat room using the concept of sockets. Each client must establish a connection with the server using a socket address. A socket address comprises two things: IP address and Port Number. After the connection is established, the client(user) can communicate with another client/s(user/s) connected on the same server. Our project enables a client to leave the chat at any point during the active connection. We have implemented the project by creating a client and server program using python on Visual Studios IDE. We have used TCP protocol instead of UDP because it is more reliable, and data read by the receiver is in order with the sender's write.

## 1.2 Sockets

We are using TCP sockets for our project. We are using two flags namely SOCK\_STREAM and AF\_INET. AF\_INET is used while creating a socket to specify the address types the socket can work with. SOCK\_STREAM is specified as an argument to declare that we are working with TCP sockets.

Few python socket API functions we have used for our project are:

```
bind()  
socket()  
listen()  
accept()  
connect()  
send()  
receive()  
close()
```

For the purpose of our project, the server can use any private or ephemeral port number. Any client that wants to connect with the server socket must enter this designated port number.

## 1.3 Multithreading

Multi-threading is used to enable execution of parts of a program concurrently. To establish our goal of a multiuser chat room we implemented multi-threading in Python. On the Server side to accept connections as well as simultaneously send and receive data to and from the server we used two threads. Similarly, in the Client side to send and receive data simultaneously we used two threads.

## 1.4 Server Program

The first task in the project is to create a Server program. A server can open a connection and accept incoming client requests. Since the project is a multi-user chat room, multiple clients can connect to this server. This connection can be established using TCP sockets. The server program uses multithreading to accept requests of multiple clients as well as to send and receive messages to and from the server. We have implemented two functions that is "incoming" and "new\_client" for this purpose.

## 1.5 Client Program

When we create a client program our first step would be to connect it to the desired host. To establish a connection to the host we open a TCP socket. To enable multiple clients to send and receive simultaneously on a server we use two threads for the “sendMessages” and “recvMessages” functions in the client script.

## 1.6 Establishing a TCP connection

Step one: The server instantiates a server socket object, denoting on which port number the communication is to occur. The bind() function takes in host address and port number.

Step two: The server now starts accepting connections using the accept() function. We have used an infinite loop that waits forever for all incoming connections.

Step three: At the client-side, the program asks the user to enter parameters: hostname and port number. After entering the correct details, the client is now connected to the server.

Step four: As soon as a client connects to the server, the server displays some details regarding the connection and sends a welcome message to the client that is connected. The client is asked to enter a username for the convenience of differentiating between multiple clients. Further instructions are given to the client in case the client wants to leave the chat (“!q” is used if the client wants to leave the chat room).

Step five: Multiple client programs can be run and connected to the server and the chat can now begin.

## 1.7 Code for Server

```
#Function for accepting incoming client connections
def incoming():
    #A loop that waits forever till it gets a client connection
    while True:
        server,clientList = SERVER.accept()
        print("%s:%s has connected." % clientList)
        server.send(bytes("Welcome to SHH Chat server, enter your name and join the room!", "utf8"))
        Thread(target=new_client, args=(server,clientList)).start()
```

*1(a) Code snippet for accepting new socket connections*

```

#Function for connecting with a new client in order to receive and send messages to and from server
def new_client(newclient,address):
    #Client enters name which has to be received by server.
    nameOfClient = newclient.recv(BUFSIZ).decode("utf8")
    #Welcome message from the server
    welcome = 'Hello %s! If you ever want to quit , type !q to log out of the chat room.' % nameOfClient
    #Server sending a welcome message to the client
    newclient.send(bytes(welcome, "utf8"))
    message = "%s has joined the room!" % nameOfClient
    #Sending a broadcast message that is visible to all the clients connected over the chat room.
    messageBroadcast(bytes(message, "utf8"))
    #Saving all client names in a dictionary
    listOfClients[newclient] = nameOfClient

    while True:
        msg = newclient.recv(BUFSIZ)
        if message != bytes("!q", "utf8"):
            messageBroadcast(message, "->" + nameOfClient + ": ")
        else:
            newclient.send(bytes("!q", "utf8")) #New client is sending !q to the server in order to quit the chat
            room.
            newclient.close()
            del listOfClients[newclient] #Closing connection for new client and deleting name entry from clients
            dictionary.
            messageBroadcast(bytes("%s has left the chat room." % nameOfClient, "utf8"))
            print("%s:%s has logged out ." % address)
            break

```

*1(b) Server Code Snippet for accepting and broadcasting messages*

## 1.8 Code for Client

**#Send Function call for the client to send messages on the server.**

```
def sendMessages():
    sendMessage=input()
    clientConnection.send(bytes(sendMessage, "utf8"))
    if sendMessage == "!q":
        clientConnection.close()
        exit()
    sendMessages()
```

*2(a)Client function to send message*

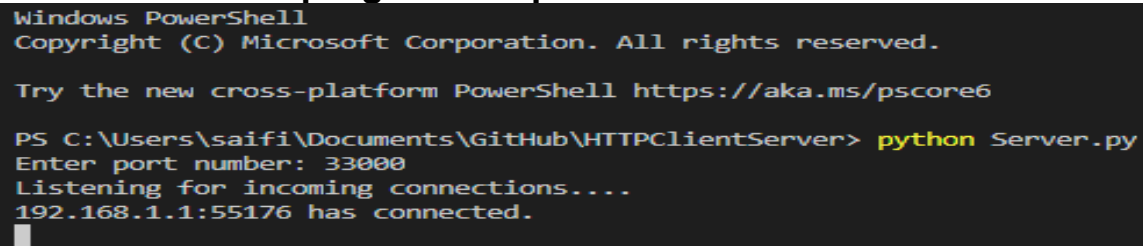
**#Receive function call for the client to receive messages from other clients connected to the server.**

```
def recvMessages():
    while True:
        try:
            messageRecieved = clientConnection.recv(BufferSize ).decode("utf8")
            print(messageRecieved)
        except OSError: #Chance of an exception arises if the client has decided to close his connection.
            break
```

*2(b)Client function to receive message*

## 1.9 Implementation of multiuser Client-Server chat in Terminal

### 1.9.1 Run Server program to open connection



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/powershell

PS C:\Users\saifi\Documents\GitHub\HTTPClientServer> python Server.py
Enter port number: 33000
Listening for incoming connections....
192.168.1.1:55176 has connected.
█
```

*3(a)Server running on 172.73.134.152*

## 1.9.2 Run Client program

```
Command Prompt - python Client.py
C:\Users\saifi\Documents\GitHub\HTTPClientServer>python Client.py
Enter port: 33000
Enter host: 172.73.134.152
Welcome to SHH Chat server, enter your name and join the room!
Saif
Hello Saif! If you ever want to quit , type !q to log out of the chat room.
```

*3(b) User 1 perspective*

```
PS C:\Users\harika paluri> & E:/ana/python.exe "c:/Users/harika paluri/Documents/GitHub/MultiUserChatRoom/Client.py"
Enter port: 33000
Enter host: 172.73.134.152
Welcome to SHH Chat server, enter your name and join the room!
harika
Hello harika! If you ever want to quit , type !q to log out of the chat room.
hi saif
->harika: hi saif
->Saif: Morning Harika
```

*3(c) User 2 perspective*

## 1.9.3 Multi-user chat room

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\saifi\Documents\GitHub\HTTPClientServer> python Server.py
Enter port number: 33000
Listening for incoming connections....
127.0.0.1:52533 has connected.
173.95.57.195:48314 has connected.
192.168.1.1:52620 has connected.
```

*3(d) Multiple clients connecting to the server*

```
C:\Users\saifi\Documents\GitHub\HTTPClientServer>python Client.py
Enter port: 33000
Enter host: 127.0.0.1
Welcome to SHH Chat server, enter your name and join the room!
Saif
Hello Saif! If you ever want to quit , type !q to log out of the chat room.
harika has joined the room!
->harika: hi saif
Morning Harika
->Saif: Morning Harika
->harika: where is herleen
Probably Sleeping
->Saif: Probably Sleeping
Fake Herleen has joined the room!
->Fake Herleen: Hello Friends
->harika: hi herleen nice to see you
Good Morn->harika: but ill have to leave now bye
->Saif: byee
->harika: !q!q
```

*3(e) Multiple clients chat room*



```

HTTPClientServer — python3 Server.py — 80x24
Last login: Sat Sep 28 12:37:16 on ttys013
(base) Herleens-MacBook-Pro:~ hsanhotra$ python3 Server.py
python3: can't open file 'Server.py': [Errno 2] No such file or directory
(base) Herleens-MacBook-Pro:~ hsanhotra$ cd /Users/hsanhotra/Documents/GitHub/HTTPClientServer
-bash: /Users/hsanhotra/Documents/GitHub/HTTPClientServer: is a directory
(base) Herleens-MacBook-Pro:~ hsanhotra$ cd /Users/hsanhotra/Documents/GitHub/HTTPClientServer
(base) Herleens-MacBook-Pro:HTTPClientServer hsanhotra$ python3 Server.py
Enter port number: 46000
Listening for incoming connections....
127.0.0.1:63619 has connected.
127.0.0.1:63622 has connected.
127.0.0.1:63630 has connected.
127.0.0.1:63619 has logged out .

```

```

HTTPClientServer — -bash — 80x24
(base) Herleens-MacBook-Pro:~ hsanhotra$ cd /Users/hsanhotra/Documents/GitHub/HTTPClientServer
(base) Herleens-MacBook-Pro:HTTPClientServer hsanhotra$ python3 Client.py
Enter port: 46000
Enter host: 127.0.0.1
Welcome to SHH Chat server, enter your name and join the room!
Saif
Hello Saif! If you ever want to quit , type !q to log out of the chat room.
Harika has joined the room!
Herleen has joined the room!
Hello guys
->Saif: Hello guys
->Harika: Hey saif, what are you doing
->Herleen: hi everyone
->Harika: I am studying for ccn midterm
->Herleen: Oh great!, we should study in a group
Sounds like a good plan herleen!
->Saif: Sounds like a good plan herleen!
i have to go now
->Saif: i have to go now
bye bye
->Saif: bye bye
!q
(base) Herleens-MacBook-Pro:HTTPClientServer hsanhotra$

```

```

HTTPClientServer — python3 Client.py — 80x24
Last login: Mon Sep 30 12:14:02 on ttys000
(base) Herleens-MacBook-Pro:~ hsanhotra$ cd /Users/hsanhotra/Documents/GitHub/HTTPClientServer
(base) Herleens-MacBook-Pro:HTTPClientServer hsanhotra$ python3 Client.py
Enter port: 46000
Enter host: 127.0.0.1
Welcome to SHH Chat server, enter your name and join the room!
Harika
Hello Harika! If you ever want to quit , type !q to log out of the chat room.
Herleen has joined the room!
->Saif: Hello guys
Hey saif, what are you doing
->Harika: Hey saif, what are you doing
->Herleen: hi everyone
I am studying for ccn midterm
->Harika: I am studying for ccn midterm
->Herleen: Oh great!, we should study in a group
->Saif: Sounds like a good plan herleen!
->Saif: i have to go now
->Saif: bye bye
Saif has left the chat room.

```

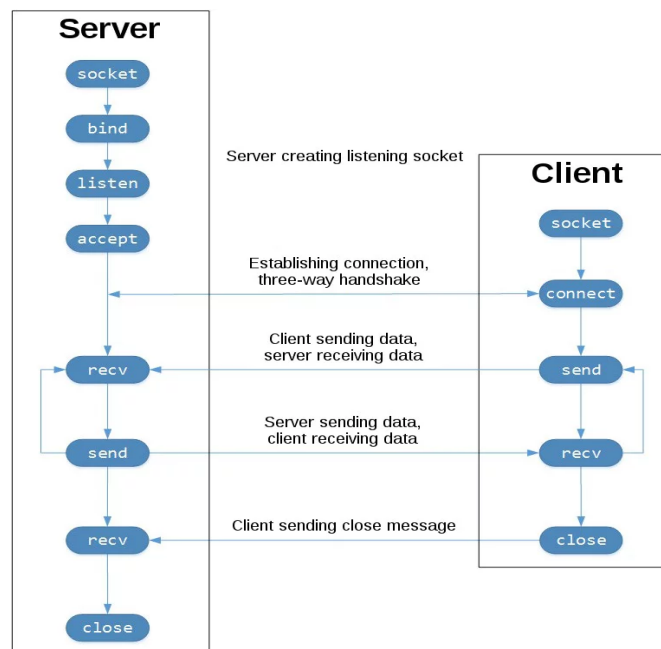
```

HTTPClientServer — python3 Client.py — 80x24
Last login: Mon Sep 30 12:14:05 on ttys001
(base) Herleens-MacBook-Pro:~ hsanhotra$ cd /Users/hsanhotra/Documents/GitHub/HTTPClientServer
(base) Herleens-MacBook-Pro:HTTPClientServer hsanhotra$ python3 Client.py
Enter port: 46000
Enter host: 127.0.0.1
Welcome to SHH Chat server, enter your name and join the room!
Herleen
Hello Herleen! If you ever want to quit , type !q to log out of the chat room.
->Saif: Hello guys
->Harika: Hey saif, what are you doing
hi everyone
->Herleen: hi everyone
->Harika: I am studying for ccn midterm
Oh great!, we should study in a group
->Herleen: Oh great!, we should study in a group
->Saif: Sounds like a good plan herleen!
->Saif: i have to go now
->Saif: bye bye
Saif has left the chat room.

```

3(f) Multiple clients chat room on local host

## 1.10 Flow of Server and Client Program



4(a) Flow diagram of Client Server connection

## 1.11 References

1. <https://realpython.com/python-sockets>
2. <https://docs.python.org/3/library/socket.html>
3. <https://docs.python.org/3/library/threading.html>