

REPORT



MULTI-USER CHAT USING CLIENT SERVER MODEL

Contents

1. Introduction
2. Standard Chat Features (Join, leave)
3. Code
4. Output

1. Introduction

Socket programming is the key API for programming distributed applications on the Internet. It is the way of connecting two nodes of a network. The client-server model is one of the most used communication systems. Server is the main part by which clients communicate with each other through messages. Every client should register with the server to communicate through messages with other clients. TCP is a connection-oriented approach for clients and servers.

MULTI-USER CHAT:

It is the application where multiple users (clients) can exchange messages in the context of a room. This can be built using client-server model with socket programming.

CLIENT-SERVER MODEL:

This model tells how can the server provide the necessary requirements and offering the services to multiple clients.

The steps:

1. Server opens the socket connection and adds its address to it and will now be waiting for the new users.
2. Client now requests a connection to server.
3. If the Server accepted the connection, then the relationship between client and server is established using a protocol (TCP).
4. Clients can now communicate with each other by sending and receiving messages through server.

PROJECT: As we are using socket programming, we use port numbers which are in range and are free to be used (we used port '20'), and as we implemented our project on local host so the IP number will be '127.0.0.1' as it is standard for every system. We have implemented multi-threading in python - Pycharm environment and run it on command prompt.

2. Standard Chat Features (Join, leave)

JOIN: In our project, whenever a client wants to join the chat room they have to enter their name and join the room. A welcome message will be sent to the new user. The other clients who are already in the chatroom will be notified with a broadcast message that this new person has been added. Now they can exchange text messages with each other through server.

LEAVE: Suppose if they want to leave the chat room they need to type "LEAVE" to exit. Once they have been removed, the other users(clients) of the chat room will be notified with a broadcast message that this person has left the chat room. Whenever a user(client) is exit from chat room, they will be removed from the list of users and will be added when they join chat room(vice-versa). Here everything is being handled by multiple threads concept.

Functions in Server side:

1. Adding a new user.
2. Adding a new connection.

Functions in Client side:

1. Sending the text messages to other clients.
2. Receiving the text messages from other clients.

3. CODE

1. SERVER CODE:

```
from socket import AF_INET, socket, SOCK_STREAM
from threading import Thread

def addnewuser(newuser, adduser):
    username = newuser.recv(BufferSize).decode("utf8")
    usermessage = 'Welcome %s to chatroom! Enter LEAVE to exit the chatroom ' % username
    newuser.send(bytes(usermessage, "utf8"))
    userwelcomemessage = "%s has joined the chat room! " % username
    messageBroadcast(bytes(userwelcomemessage, "utf8"))
    userList[newuser] = username

    while True:
        userwelcomemessage = newuser.recv(BufferSize)
        if userwelcomemessage != bytes("LEAVE", "utf8"):
            messageBroadcast(userwelcomemessage, "Message from User - "+username+": ")
        else:
            newuser.send(bytes("LEAVE", "utf8"))
            newuser.close()
            del userList[newuser]
            messageBroadcast(bytes("%s left the chat room." % username, "utf8"))
            print("%s: %s has disconnected." % adduser)
            break

def addconnection():
    while True:
        server, userList = SERVER.accept()
        print("%s: %s has connected." % userList)
        server.send(bytes("Welcome to Chat room. To join enter your Name and begin chatting.", "utf8"))
        Thread(target=addnewuser, args=(server, userList)).start()

def messageBroadcast(userwelcomemessage, nameId=""):
    for socketConn in userList:
        socketConn.send(bytes(nameId, "utf8")+userwelcomemessage)

userList = {}
addressList = {}

port = int(input("Enter port number: "))
host = ""
BufferSize = 1024
Address = (host, port)
```

```
SERVER = socket(AF_INET, SOCK_STREAM)
SERVER.bind(Address)
```

```
if __name__ == "__main__":
    SERVER.listen(5)
    print("Waiting for new users....")
    ACCEPT = Thread(target=addconnection)
    ACCEPT.start()
    ACCEPT.join()
    SERVER.close()
```

2. CLIENT CODE:

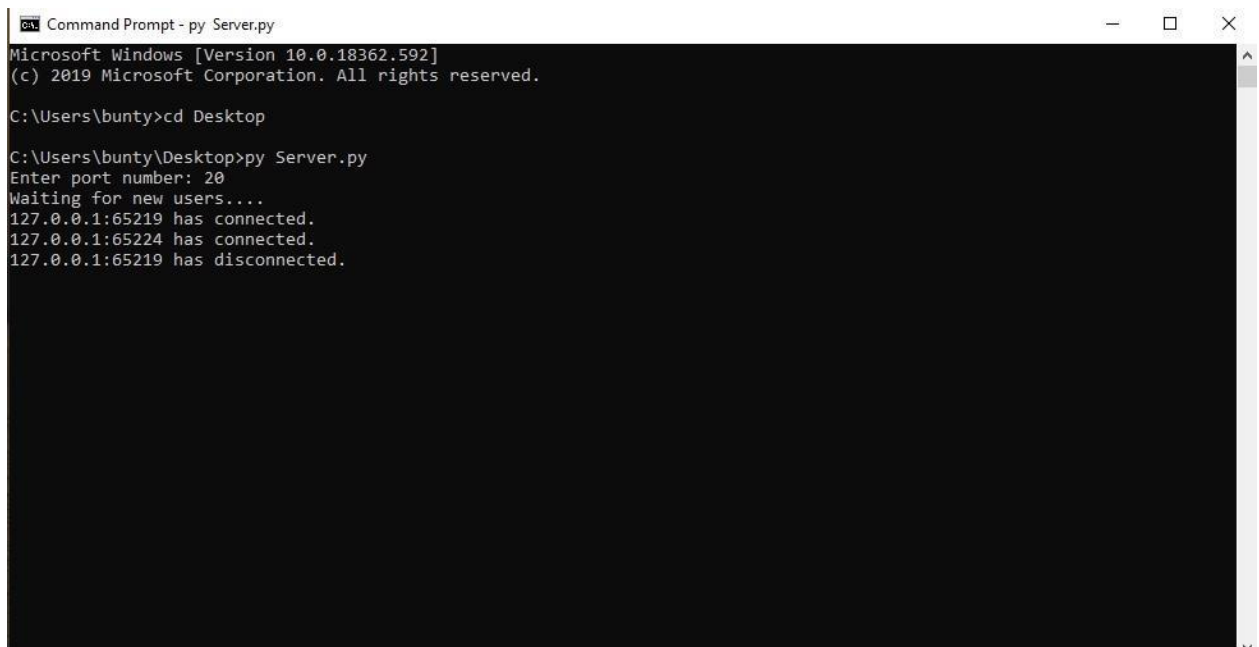
```
from socket import AF_INET, socket, SOCK_STREAM from
threading import Thread
userconnection = socket(AF_INET, SOCK_STREAM)
PORT_NO = int(input('Enter port number: '))
HOST_NO = input('Enter host number: ')
BufferSize = 1024
ADDR = (HOST_NO, PORT_NO)
userconnection.connect(ADDR) def
sendText():    sText=input()
userconnection.send(bytes(sText, "utf8"))
if sText == "LEAVE":
    userconnection.close()
    exit()
    sendText() def
receiveText():
while True:
try:
    rText = userconnection.recv(BufferSize ).decode("utf8")
    print(rText)    except OSError:        break

if __name__ == '__main__':
    Thread(target = receiveText).start()
    Thread(target = sendText).start()
```

4.OUTPUT

This can be implemented for “n” any number of clients. Here to simplify for understanding we used 2 clients (Keerthi Reddy Kandi, Raj Ingle).

1. First we will run server program that enter port number. It will wait for new users and it displays whenever a new user is connected or disconnected.



```
Command Prompt - py Server.py
Microsoft Windows [Version 10.0.18362.592]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\bunty>cd Desktop
C:\Users\bunty\Desktop>py Server.py
Enter port number: 20
Waiting for new users....
127.0.0.1:65219 has connected.
127.0.0.1:65224 has connected.
127.0.0.1:65219 has disconnected.
```

2. Now we open 2 command prompt (2 clients). First client (Keerthi Reddy Kandi) has joined the chat room and received a welcome message. Second client (Raj Ingle) has joined, now Keerthi user got notified that Raj has joined the chat

room. Now they begin chatting. After sometime Keerthi user wants to exit chat room so specified LEAVE and left the chat room. Raj got notified that Keerthi left the chat room.

```
Command Prompt
Microsoft Windows [Version 10.0.18362.592]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\bunty>cd Desktop

C:\Users\bunty\Desktop>py Client.py
Enter port number: 20
Enter host number: 127.0.0.1
Welcome to Chat room. To join enter your Name and begin chatting.
Keerthi Reddy Kandi
Welcome Keerthi Reddy Kandi to chatroom! Enter LEAVE to exit the chatroom
Raj Ingle has joined the chat room!
Hello Raj
Message from User - Keerthi Reddy Kandi: Hello Raj
Message from User - Raj Ingle: Hey Keerthi wassup
I'm good how are you?
Message from User - Keerthi Reddy Kandi: I'm good how are you?
Message from User - Raj Ingle: I'm doing well
ok then bye take care
Message from User - Keerthi Reddy Kandi: ok then bye take care
Message from User - Raj Ingle: Bye see you later
LEAVE

C:\Users\bunty\Desktop>
```

```
Command Prompt - py Client.py
Microsoft Windows [Version 10.0.18362.592]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\bunty>cd Desktop

C:\Users\bunty\Desktop>py Client.py
Enter port number: 20
Enter host number: 127.0.0.1
Welcome to Chat room. To join enter your Name and begin chatting.
Raj Ingle
Welcome Raj Ingle to chatroom! Enter LEAVE to exit the chatroom
Message from User - Keerthi Reddy Kandi: Hello Raj
Hey Keerthi wassup
Message from User - Raj Ingle: Hey Keerthi wassup
Message from User - Keerthi Reddy Kandi: I'm good how are you?
I'm doing well
Message from User - Raj Ingle: I'm doing well
Message from User - Keerthi Reddy Kandi: ok then bye take care
Bye see you later
Message from User - Raj Ingle: Bye see you later
Keerthi Reddy Kandi left the chat room.
-
```