Multi user chat server using TCP as transport layer protocol

Amrith M

April 10, 2018

1 About

These programs work as following

Server must be running first. Any number of clients can be run then server randomly take request from any client and respond to it.To respond client server maintains that many number of threads each thread share the same object, which object's method will respond to client, thus synchronization is achieved.Client program runs until client exits.

2 Server.py

```python
import socket
import _thread

sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
sock.bind(('',3003))
sock.listen(10)

connected = []

def client(conn,addr):
    name = conn.recv(100)
    name = name.decode()
    print(name + " Joined Chat")
    conn.sendall("Welcome to Chat Room".encode())
    while True:
        data = conn.recv(100)
        if data:
            data = data.decode()
            print(name + " => " + data)
            msg = name + " => " + data
            broadCast(conn,msg.encode())
        else:
            break
    conn.close()

def broadCast(conn,data):
    for client in connected:
        if(client != conn):
            try:
                client.sendall(data)
            except:
                client.close()
                connected.remove(client)


while True:
    conn,cnt = sock.accept()
    connected += [conn]
    _thread.start_new_thread(client,(conn,cnt))
```

3 Client.py

```
import socket
import _thread

sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
sock.connect(('',3003))

def listening():
    while True:
        data = sock.recv(100)
        print(data.decode())

name = input()
name = name.encode()
sock.send(name)
welcome = sock.recv(100)
print(welcome.decode())

_thread.start_new_thread(listening,())

while True:
    data = input()
    sock.send(data.encode())
```