Multi user chat server using TCP

Aim

To implement a multi user chat server using TCP as transport layer protocol.

Theory

TCP (Transmission Control Protocol) works with the Internet Protocol (IP),which defines how computers send packets of data to each other. Together, **TCP** and **IP** are the basic rules defining the Internet. It is a connection-oriented protocol,which means that a connection is established and maintained until the applicationprograms at each end have finished exchanging messages.

Server- In a simple multi user chat system, the server usually has the role toreceive the messages sent by the clients and send it to all other clients. So basically,he handles the routing of the messages sent by one client to all the other clients.

Client- The client here acts from the side of the user. He sends the messagesto the server, and the server sends this message to all the other clients to simulatea simple multi-user chat system.

Code

Server.py

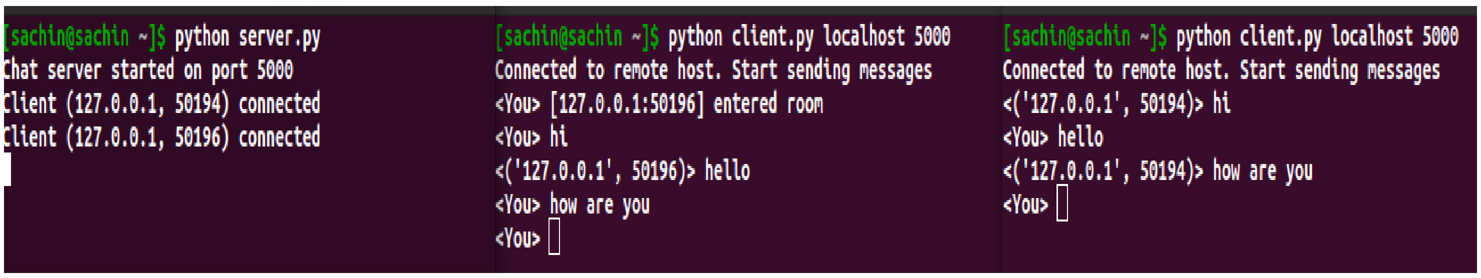
```
import socket, select


def broadcast_data (sock, message):
	for socket in CONNECTION_LIST:
		if socket != server_socket and socket != sock :
			try :
				socket.send(message)
			except :
				socket.close()
				CONNECTION_LIST.remove(socket)

if __name__ == "__main__":
	CONNECTION_LIST = []
	RECV_BUFFER = 4096
	PORT = 5000
	server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
	# this has no effect, why ?
	server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
	server_socket.bind(("0.0.0.0", PORT))
	server_socket.listen(10)

	# Add server socket to the list of readable connections
	CONNECTION_LIST.append(server_socket)

	print "Chat server started on port " + str(PORT)

	while 1:
		read_sockets,write_sockets,error_sockets = 
select.select(CONNECTION_LIST,[],[])

		for sock in read_sockets:
			#New connection
			if sock == server_socket:
					sockfd, addr = server_socket.accept()
				CONNECTION_LIST.append(sockfd)
				print "Client (%s, %s) connected" % addr

				broadcast_data(sockfd, "[%s:%s] entered room\n" % 
addr)
			else:
				# Data recieved from client, process it
				try:
					data = sock.recv(RECV_BUFFER)
					if data:
						broadcast_data(sock, "\r" + '<' 
+str(sock.getpeername()) + '> ' + data)

				except:
					broadcast_data(sock, "Client (%s, %s) is 
offline" % addr)
					print "Client (%s, %s) is offline" % addr
					sock.close()
					CONNECTION_LIST.remove(sock)
					continue

	server_socket.close()
```

Client.py

```
import socket, select, string, sys

def prompt() :
	sys.stdout.write('<You> ')
	sys.stdout.flush()

if __name__ == "__main__":

	if(len(sys.argv) < 3) :
		print 'Incorrect Usage. Usage : python client.py hostname 
port'
		sys.exit()
```

```
     host = sys.argv[1]
     port = int(sys.argv[2])

     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
     s.settimeout(2)

     try :
          s.connect((host, port))
     except :
          print 'Unable to connect'
          sys.exit()

     print 'Connected to remote host. Start sending messages'
     prompt()

     while 1:
          socket_list = [sys.stdin, s]
          read_sockets, write_sockets, error_sockets =
select.select(socket_list , [], [])

          for sock in read_sockets:
               if sock == s:
                    data = sock.recv(4096)
                    if not data :
                         print '\nDisconnected from chat server'
                         sys.exit()
                    else :
                         #print data
                         sys.stdout.write(data)
                         prompt()

               else :
                    msg = sys.stdin.readline()
                    s.send(msg)
                    prompt()
```

Output



```
[sachin@sachin ~]$ python server.py
Chat server started on port 5000
Client (127.0.0.1, 50194) connected
Client (127.0.0.1, 50196) connected
```

```
[sachin@sachin ~]$ python client.py localhost 5000
Connected to remote host. Start sending messages
<You> [127.0.0.1:50196] entered room
<You> hi
<('127.0.0.1', 50196)> hello
<You> how are you
<You>
```

```
[sachin@sachin ~]$ python client.py localhost 5000
Connected to remote host. Start sending messages
<('127.0.0.1', 50194)> hi
<You> hello
<('127.0.0.1', 50194)> how are you
<You>
```