BLG 433E

COMPUTER COMMUNICATIONS

PROJECT 1

AHMET GÖKTUĞ SEVİNÇ

150140120

In this project we implemented an instant messaging application using python socket programming. Sockets are bi-directional communication channels that establishe communication between a server and one or more clients. For this application we had one server and multiple clients. The socket on the server side associates itself with a port number and clients can communicate with each other from that port number via server. Since there are multiple clients I used multi-threading for clients side. So, for each client connecting to the server, a seperate thread was created. For this application I created two scripts: *Server.py* and *Client.py.*

*Server.py:*

Server script firstly, establishes a socket and binds it to an ip address and port 12000. Then it starts to wait for clients and when it receives a connection, it keeps those connections in a list (*list\_of\_clients*). For each connection a threat is created and in each thread, server waits for a message from clients and after receiving the message it sends the message to other clients.

|  |
| --- |
| **serverSocket = socket(AF\_INET, SOCK\_STREAM)** // AF\_INET is the address domain of the  socket. This is used when we have an Internet Domain with any two hosts. The second argument is the type of socket. SOCK\_STREAM means that data or characters are read in  a continuous flow.  **serverSocket.setsockopt(SOL\_SOCKET, SO\_REUSEADDR, 1)** //allow reuse of ip addresses    **serverPort = 12000** //port number is 12000  **serverSocket.bind(('', serverPort))** //bind the server to ‘0.0.0.0’ and port: 12000  **serverSocket.listen(50)** //listen for 50 active connections    **list\_of\_clients = []** //client list  **def client\_thread(connectionSocket, nameofClient):** //thread for clients    **connectionSocket.send("You are connected to chatroom.".encode())**    **while True:**  **try:**    **message = connectionSocket.recv(2048)** //receive a message from a client    **if message:**  **message\_to\_send = "<" + nameofClient.decode("utf-8") + "> " + message.decode("utf-8")** // if there is a message concatenate it with senders username  **broadcast(message\_to\_send, connectionSocket)** //call a function to broadcast the message    **else:**  **remove(connectionSocket)** //else remove connection    **except:**  **continue**    **def broadcast(message, connection):** //this function sends message to whole clients except the one who is sending the message  **for clients in list\_of\_clients:**  **if clients!=connection:**  **try:**  **clients.send(message.encode())**  **except:**  **clients.close()** //close and remove client  **remove(clients)**    **def remove(connection):** //this function removes the connection from the list of clients  **if connection in list\_of\_clients:**  **list\_of\_clients.remove(connection)**    **while True:**  **connectionSocket, addr = serverSocket.accept()**  **list\_of\_clients.append(connectionSocket)**    **message = ("Please enter your username: ")** //ask for username  **connectionSocket.send(message.encode())**  **nameofClient = connectionSocket.recv(2048)** //receive username  **nameofClient = nameofClient.rstrip()** //remove newline character from username  **if nameofClient:**  **print(nameofClient.decode("utf-8") + " connected")** //print username  **else:**  **print("Could not receive")**    **start\_new\_thread(client\_thread,(connectionSocket,nameofClient))** //create new thread    **connectionSocket.close()**  **serverSocket.close()** |

*Client.py:*

Client-side script firstly tries to access the server socket created at the specified IP Address and port number. After connecting to the server, it continuosly checks for input. There are two possible inputs: input from user and input from server. If user types a message client sends this message to the server. If message comes from server, it prints the message.

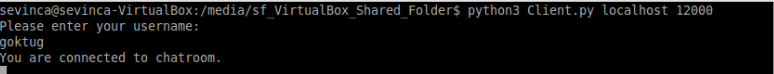
|  |
| --- |
| **clientSocket = socket(AF\_INET, SOCK\_STREAM)**  **if len(sys.argv) != 3:** // receive ip address and port number from the command line  **print ("Please enter IP address and port number")**  **exit()**  **serverName = str(sys.argv[1])**  **serverPort = int(sys.argv[2])**  **clientSocket.connect((serverName, serverPort))** //connect to the server    **while True:**    **sockets\_list = [sys.stdin, clientSocket]** //list of inputs  **readSockets, writeSocket, errorSocket = select.select(sockets\_list,[],[])**    **for socks in readSockets:**  **if socks == clientSocket:** //input from the server  **message = socks.recv(2048)** //receive message  **print(message.decode("utf-8"))** //print message  **else:** //input from the user  **message = sys.stdin.readline()** //read user input  **clientSocket.send(message.encode())** //send it to the server to broadcast  **sys.stdout.flush()**  **clientSocket.close()** |

**COMPILATION PROCEDURE**

Firstly we run Server.py:



Then we run Client.py script with IP and Port number arguments:

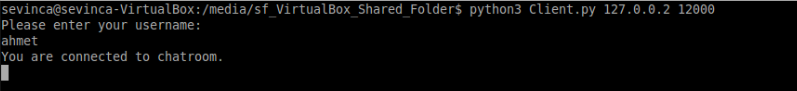


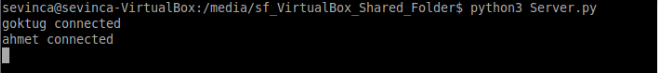
It asks for username and after entering username it gives a message saying connection is successful.



Also after connection, server-side gives information about client.

Now we can connect another client:





We can start messaging:

