Client Server Application

November 2022

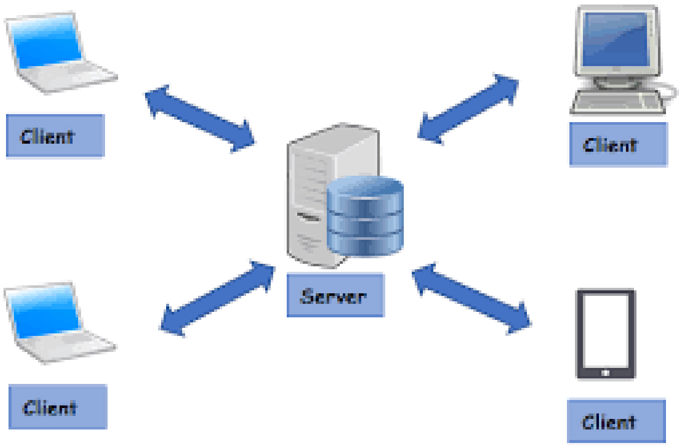
NOTE APPLICATION

Group Members:

1. U20CS079 Mihir Gandhi
2. U20CS083 Mihir Hemnani
3. U20CS086 Rajan Patel
4. U20CS093 Nehal Jhajharia

1 Introduction

How to set up a simple Note Room server and allow multiple clients to connect to it using a client-side script. The code uses the concept of sockets and threading.
Sockets can be thought of as endpoints in a communication channel that is bi-directional and establishes communication between a server and one or more clients. Here, we set up a socket on each end and allow a client to interact with other clients via the server. The socket on the server side associates itself with some hardware port on the server-side. Any client that has a socket associated with the same port can communicate with the server socket.
A thread is a sub-process that runs a set of commands individually of any other thread. So, every time a user connects to the server, a separate thread is created for that user, and communication from the server to the client takes place along individual threads based on socket objects created for the sake of the identity of each client. We will require two scripts to establish this chat room. One to keep the serving running, and another that every client should run in order to connect to the server.



2 Pseudo Code

Server side:

```
1. Create Server Socket
2. Bind
3. Listen
4. Accept the connection, connectSocket
5. Create a new thread and increment the thread count by 1
6. Take name as input from client
7. If the client exist then ask client to try with other name
8. Add the new user to the clientList
9. While input from client not equal to quit
   take input from client
   store the input in message buffer
```

10. drop the connection

11. decrement the thread count by 1

12. remove the client name from clientList

Client side:

1. Create clientSocket

2. Send the client name

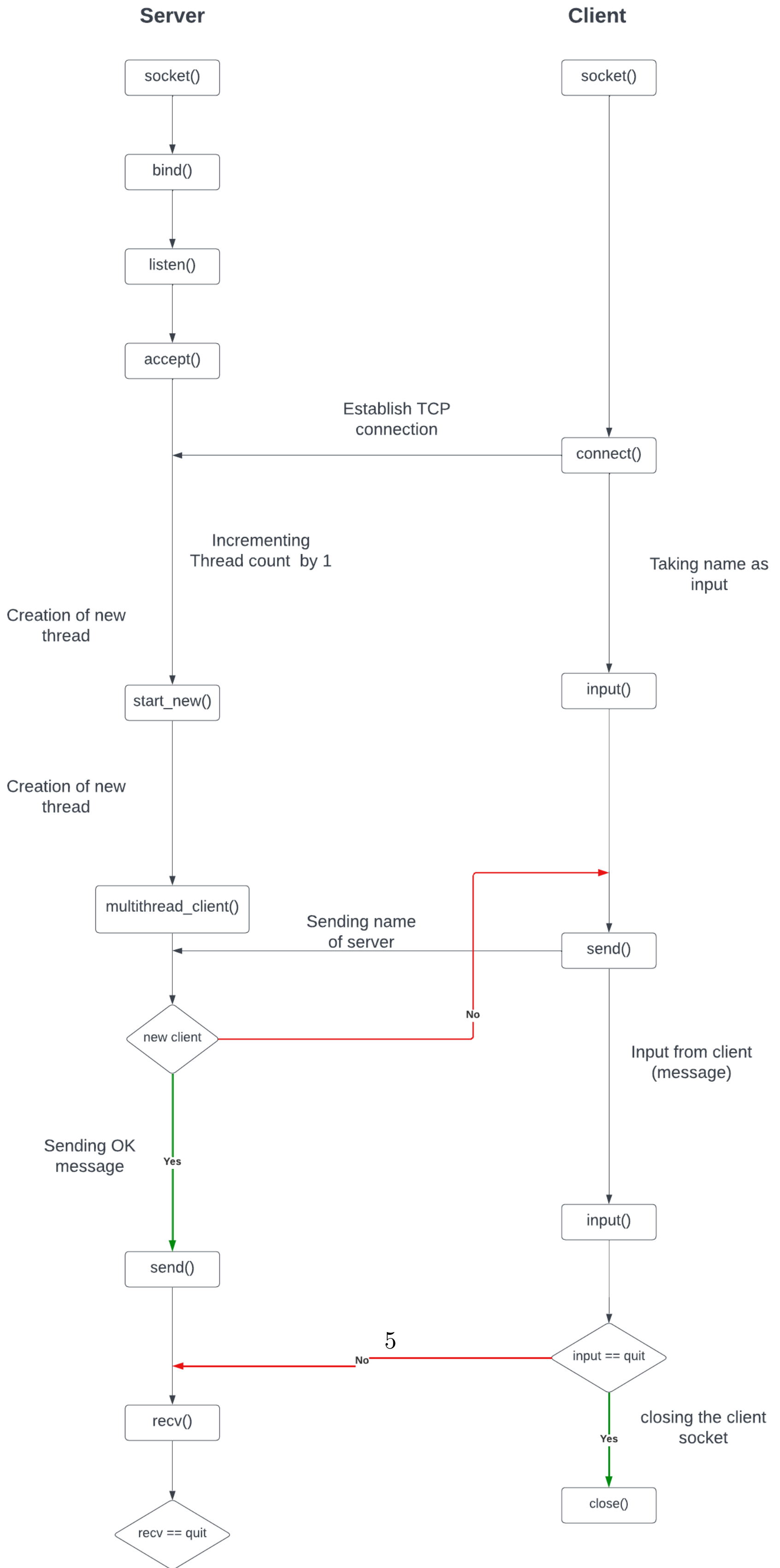
3. if recieved message is not equal to 'OK'
   repeat step 2

4. Sending message to server

5. continue step 4 until sent message is equal to 'quit'

6. close the clientSocket

3 Flowchart


Server
Client
socket()
socket()
bind()
listen()
accept()
Establish TCP connection
connect()
Incrementing Thread count by 1
Taking name as input
Creation of new thread
start_new()
input()
Creation of new thread
multithread_client()
Sending name of server
send()
No
new client
Input from client (message)
Sending OK message
Yes
input()
send()
No
input == quit
recv()
closing the client socket
Yes
close()
recv == quit

4 Performance of Application

Security is relatively easy to implement with this type of network model, since you can setup a single server computer to handle all information requests or login requests for the entire network.
As here we are using TCP connection of communication, its reliable and connection oriented. For every client a new socket is establish on the server side.

5 Conclusion

The Client-Server network model provides important services to the network safely and securely, it also allows the convenience of allowing the users to work on their own workstation machine. However, this network model can be very expensive, not only because the software can be expensive, but you also must provide adequate hardware for both the servers and the individual workstation machines, which can become very expensive with revolving hardware updates