ADJZAI001
MCHWIL006
RMJHRS001

CSC3002F ASSIGNMENT 1 NETWORKS-2019

-

Introduction

Many messaging applications have been created which implements various forms of communication such as group chats, broadcast chats (Chat Room) or private messaging. These forms of chatting implements various messaging protocols which is determined by the intended features and functionality of the application.

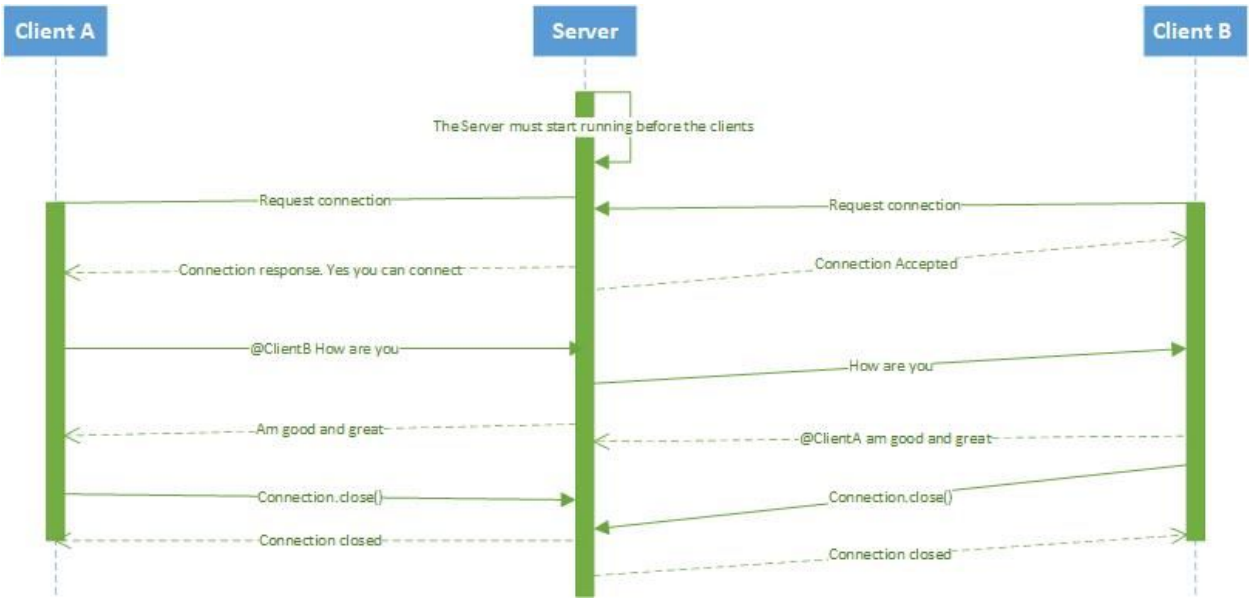
The Chat Application which we have implemented is a one to one private chat which means that the sender has to specify which receiver is intended for the message sent.

Chat Application Design

Protocol Design

Three classes have been made namely Server,Client and ServerThread. The Server class is responsible for handling the connection requests coming from all the clients. The transport protocol used is TCP meaning that the server has to send an acknowledgement to the client that the connection has been accepted. When the server starts running the clients can then connect to the server through the server address and the port number. The process of connecting server and the client is illustrated below in text and also sequence diagram;

1. The server socket is created and port number is assigned to the server. This done by instantiating a server object where Java ServerSocket library is used. The serversocket object takes in the port of where the server will be running from.
2. The server is now set and running and clients can start connecting through by specifying the server IP address and port number where the server is running. The server keeps running even when some clients disconnect from the server.
3. When the client class is run; the client object instantiation takes in two parameters that is the port number and the IP Address of the server. A try and catch block is used to first verify whether a server is running or not and when the machine realises a server is running; it indicates that a connection exists. If the server is not running; the client will indicate that no server is running. When the server receives the client connection requests, a socket is assigned to the particular socket where the server will be communicating through that particular client.



Messaging Process

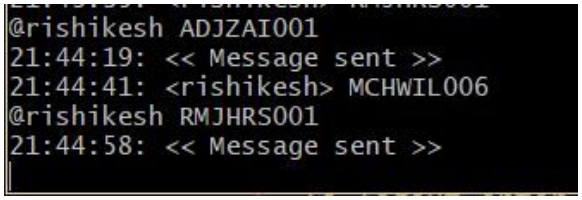
The server has a static variable for the maximum number of clients who can connect to the server. When the client wants to connect to the server; the server first checks whether the maximum number of clients allowed in the server is reached and if not the connection from the client is accepted.

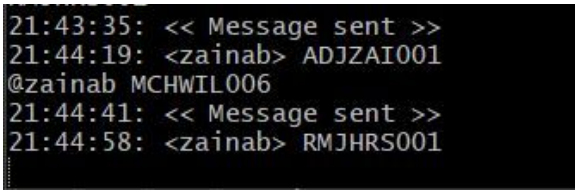
```
Client accepted /196.42.122.148:57172
Client accepted /196.42.122.148:57173
Client accepted /196.42.122.148:57186
```

This shows there are three client who have connected to the server. The client has an option to see who is already there to chat with them by asking who is here.

```
<< Welcome to NetChatter, zainab! >>
To logout enter 'EXIT' on a new line
To see who is in the chatroom enter 'WHOISHERE' on a new line
To send a private message type '@username' followed by your message
Admins are allowed to broadcast to all users at once

21:47:47: << rishikesh is the admin >>
WHOISHERE
> rishikesh
> zainab
> willie
```

The client specifies the receiver the message is intended for by putting a “@” at the beginning of the message. After the message leaves the client host; the message is received by the server.

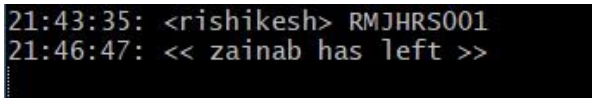




Zainab (left) texts Rishikesh (right) privately by inserting the “@” at the beginning of the username and the message is delivered to him privately without the other clients seeing the message.

The server receives connection request from different clients and every client is assigned a thread on the server to run on that thread. This is to allow easy implementation where communication between a pair cannot affect another conversation being done by another pair.

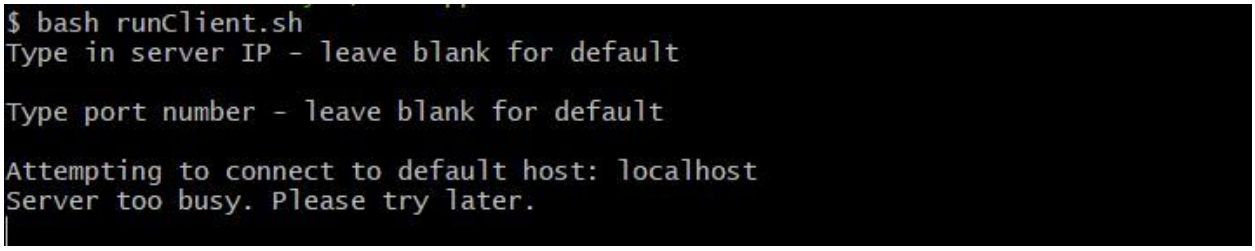
When the user wants to log off and leave the chat, they can enter ‘EXIT’ to close the chat application and the server disconnects the client appropriately.



Constraints Implemented

1. **Bandwidth**
   Bandwidth is the maximum rate of data transfer across a given path. This constraint is implemented as the many people in the third world countries are affected by bandwidth issues.

   Our application implements this constraint by limiting the amount of users that can connect to the server at any time to 10 users. If the number of the clients has reached maximum, the client is notified that the server is at full capacity. This is done to prevent flooding the server which will reduce the flow rate of the messages and impact the user experience with the chat application.



2. **Authentication**

   A user would need to either login or sign up before using the service. A class called Login handles this. It reads and stores user data in a text file called “db.txt”. The loadUsers() function loads in all the user data, makes User objects out of them and stores them in an ArrayList in memory.

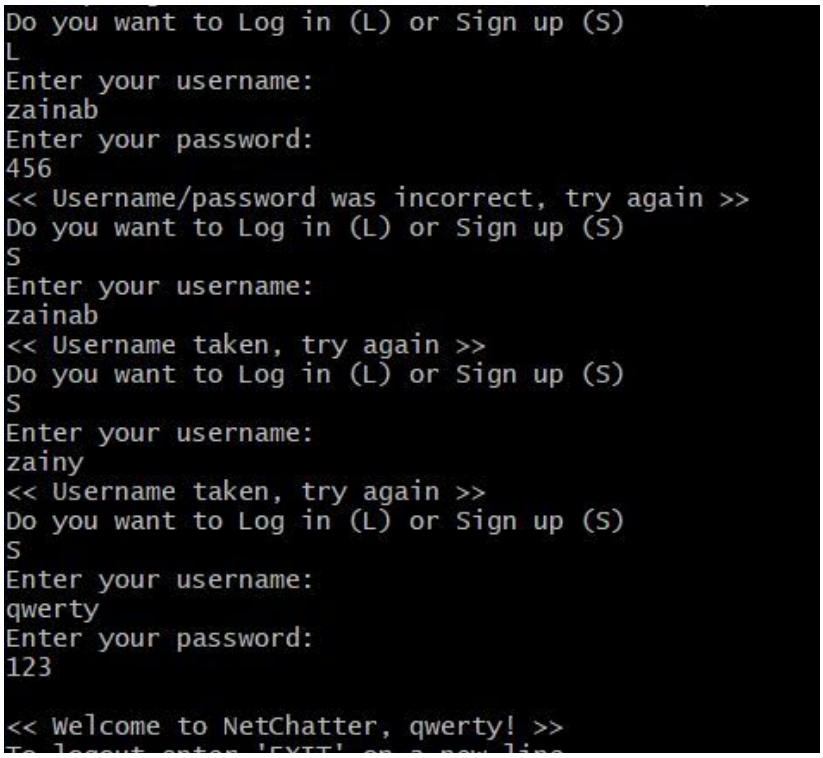
   When signing, up a user must enter a desired username and password. The username is checked with other usernames if it isn’t already in use. If not, then a new user is created in stored in the ‘database’.

   When logging in, a user is asked for a username and then for a password, these are checked by the ValidateLogin() function. If either user and/or username is wrong, an error message is displayed and the user may try to login again.

Features Included and Creativity

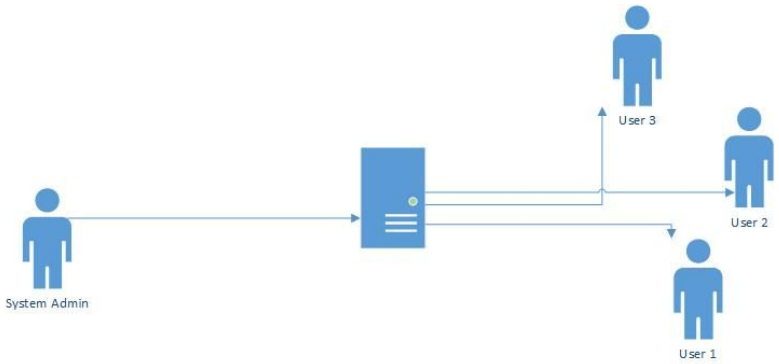
1. **LogIn Feature**

   This is for authentication purposes to make sure no two users share the same name. When the server starts; it loads all the users first whose names and passwords have been saved in a text file. The server then creates an array of users and every other client who request to connect to the server; their username is checked against the array and if two names exists ; the server prompts the user to input another name. This is to avoid messages from going to the same users



2. **Broadcast Feature**

   This feature is to be used by the system admins for the purpose of broadcasting messages to the users of the platform. The server assigns the role of admin to the first user who connects and thus only this user is allowed to broadcast messages to the other users who subsequently join. If the current admin leaves the chat, the next available client on the list will become the admin.

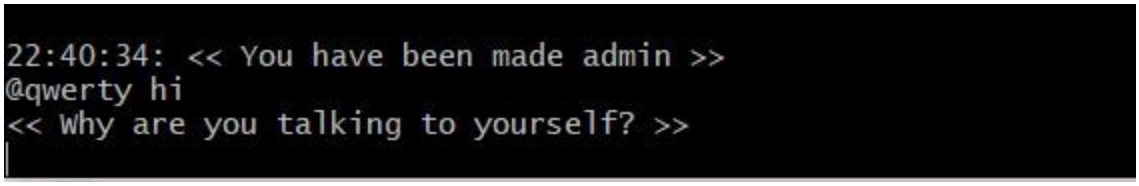



```
<< Welcome to NetChatter, rishikesh! >>
To logout enter 'EXIT' on a new line
To see who is in the chatroom enter 'WHOISHERE' on a new line
To send a private message type '@username' followed by your message
Admins are allowed to broadcast to all users at once

21:36:53: << You have been made admin >>
Hey I'm admin!
21:38:44: << Message sent >>
```

```
<< Welcome to NetChatter, willie! >>
To logout enter 'EXIT' on a new line
To see who is in the chatroom enter 'WHOISHERE' on a new line
To send a private message type '@username' followed by your message
Admins are allowed to broadcast to all users at once

21:37:29: << rishikesh is the admin >>
21:38:44: <rishikesh> Hey I'm admin!
I'm not admin :(
<< You cannot broadcast as you are not admin >>
```

3. **Error Handling**

   When the user chooses a username during sign up, the server rejects usernames beginning with '@' as this would cause errors later in the application. The user is also ushered into using the application correctly by catching various incorrect actions the user could do such as broadcast messaging when they are not an admin, trying to message themselves and trying to message users who are not in the chatroom.



The application also catches any errors with the arguments presented to the Client and Server applications. They both take input in a specific order and will notify the user if this is not adhered to. To make this easier, we have created bash scripts to do the inserting of arguments for the user.

In the case of errors occurring independent of the user input, such as IOExceptions, the client experiencing the error will display the error instead of crashing and it would then close the connection and the server would continue as if the user has logged off. On the other hand, if the server experiences an error, the clients will display that they have lost connection to the server.