CS 4850/7850 Computer Networks I

Project: ChatRoom Version 2

Due Date: Friday, March 21, before 11:00am.

1. Overview

In this project, you will extend version 1 by using threads to implement a chat room that includes multiple clients and a server that utilizes the socket API. The socket API is implemented in many programming languages. You are permitted to use your language of choice as long as it utilizes the socket API.

The client program provides commands: **login** (allow users to join the chat room), **newuser** (create a new user account), **send all** (sends a message to the server and the server broadcasts the message to all logged in users), **send UserID** (sends a message to the server and the server send a unicast message to the user), **logout** (quit the chat room), and **who** (list all the users in the chat room).

The server runs a chat room service, manages all the clients and distributes the messages.

2. Description

You will implement a server and a client. The server will use 1 plus the last four digits of your student ID as the server port number to avoid conflicting with other students' server program. For example, if the last four digits of your student ID is 3456, then as the server port number is 13456. When running both the client and the server on the same computer, please use 127.0.0.1 as the server IP address.

In this project, there are multiple clients connecting to the server at the same time. *The server can support MAXCLIENTS number of concurrent clients. For the grading purpose, se*t *MAXCLIENTS = 3.* The commands listed in item 3 below are input by the user on the client side. The client checks for correct usage of the commands, then relays the commands to the server. The server implements the corresponding functions required to support these commands. When the server starts, it should first read the user account information from the given file users.txt. For grading purpose, the initial user accounts (UserID, Password) are (Tom, Tom11), (David, David22), and (Beth, Beth33).

3. Client/Server Functions to be implemented

1. **login** UserID Password (same command as in Project V1)

The client first checks the correct usage of the command, and, if correct, sends the command to the server. If the server can verify the UserID and the Password, the server will send a confirmation message to the client and inform all other clients that this client joins the chat room; otherwise, the server will decline login and send an error message to the client.

2. **newuser** UserID Password (same command as in Project V1)

Creates a new user account. A new user can invoke the newuser command to create an account (we don't assume an administrator in this scenario). The length of the UserID should be between 3 and 32 characters, and the length of the Password should be between 4 and 8 characters. UserID and Password need to be case-sensitive. UserID and Password need to be case-sensitive. Also, assume that UserID and Password do not contain spaces (your program does not need to test for potential spaces).

The client first checks the correct usage of the command (including correct lengths of UserID and Password), and, if correct, sends the command to the server. The server will reject the request if the UserID is already there. The users' IDs and passwords should be kept in the given file users.txt on the server side.

3. **send all** message

Send the "message" to the server. The server will precede the message with the UserID of the sender and broadcast the message to all other clients. Message size can be between 1 and 256 characters.

4. **send UserID** message

Send the "message" to the server. The server will precede the message with the UserID of the sender and unicast the message to the client "UserID". Message size can be between 1 and 256 characters.

5. **who**

List all the users in the chat room (including the user who issued the who command).

6. **logout** (same command as in Project V1)

Logout from the chat room. The connection between the server and client will be closed and the client should exit. The server should continue running and allow other clients to connect.

4. Program specifications

Client Side Specs

- While logged out, a user should only be able to either login or create a new user. All other commands should be invalid while logged out.
- While logged in, a user should only be able to send messages or log out. The user should not be able to login while already logged in or create a new user while logged in.
- Password length and username length restrictions should be implemented as outlined above.

Server Side Specs

- New user accounts should persist between sessions (i.e., the new user information needs to be stored in the users.txt file by the server). If the file does not exist, the server should create it when the first account is created.

- Usernames must be unique. A new user cannot be created with the same user name as an existing user

5. Programming Language

You can use any programming language you like (C, C++, Java, Python, Ruby,…etc). Server and client should be implemented as console applications using the socket API, so please do not add a Graphic User Interface to your program. You should run the code the same way as you did in Version 1 except that multiple clients will be open simultaneously. As most of you are familiar with C, client and server skeleton programs in C are posted on Canvas, including Visual Studio project files and compile instructions, as a starting point.

6. Grading

For Undergraduate Students: Total 50 EXTRA points

- 7.5 points for each of the four commands. You will lose points if the commands are not implemented as specified (30 points total)
- 20 points for neat source code and implementing appropriate error messages. Your source code must be well commented, including an overall header with student name, date, program description, etc.
- You will lose 40 points for any bug that causes the program to crash or makes the program exit abnormally even if all commands can be demonstrated.
- You will lose 50 points if you do not utilize the socket API.

These extra points will only be given to undergraduate students who also submitted Version 1 of the project (submitting Version 2 as Version 1 will not count).

For Graduate Students: Total 100 points

- 15 points for each of the four commands. You will lose points if the commands are not implemented as specified (60 points total)
- 40 points for neat source code and implementing appropriate error messages. Your source code must be well commented, including an overall header with student name, date, program description, etc.
- You will lose 80 points for any bug that causes the program to crash or makes the program exit abnormally even if all commands can be demonstrated.
- You will lose 100 points if you do not utilize the socket API.

7. Code submission

You have to submit your source code files through the course Canvas site. Late or email submissions, or submission of executables will not be accepted.

Please submit two Zip files, one for the client source code and one for the server source code (not executables). Also include any IDE-related files necessary to compile the programs and instructions on how to compile and run your code (including the version of the language, libraries, tools used, Java JRE version, etc).

8. Outputs

The client/server functions need to be implemented exactly as shown in Section 3, including the function calls (e.g., newuser Mike Mike11). You are not allowed to change these calls; i.e., do not change it to something like:

Newuser
Please enter user name: Mike
Please enter user password: Mike11

The following shows an example chat room session. **Your client/server programs must re-produce this example exactly**.

Client output for Tom:

My chat room client. Version Two.

>send
> Denied. Please login first.
>login Tom Tom11
> login confirmed
>who
> Beth, David, Tom
> Beth: when is project 2 due?
>David: I do not know.
>send all November 3.
>David: really?
>David left.

Client output for David:

My chat room client. Version Two.
>login David David22
> login confirmed
>who
> Beth, David
>send Beth are you there?
>Beth: yes
>Tom joins.
>Beth: when is project 2 due?
>send all I do not know.
>Tom: November 3.
>send all: really?
>logout

Server output:

My chat room server. Version Two.

Beth login.
David login.
David (to Beth): are you there?
Beth (to David): yes
Tom login.
Beth: when is project 2 due?
David: I do not know.
Tom: November 3.
David: really?
David logout.