CSE 3461/5461 Computer Networking and Internet Technologies
Project 2 - A Simple Bulletin Board Using Socket Programming

Instructor: Giovani Abuaitah

Due: Tuesday November 24, 4:30 PM

1 Overview

A bulletin board, in its simplest forms, is a discussion board where users can join a certain group and post messages that can only be seen by that group. The purpose of this project is to implement a fully fledged client-server application (i.e., the bulletin board) using pure **unicast** sockets. In this project, you can write both the client and server implementations using the same programming language, or you can have the client programm written in a langauge different from the one used for the server program. Additional credit ($\sim 5\%$) will be given if you choose to do the latter. You can use Java, Python, and/or C/C++. **IMPORTANT**: you **MUST** use **sockets** regardless of the language you choose (no other built-in or third party libraries/modules are allowed for networking purposes), and your code should run on the cse linux environments for demo purposes (see Demo section). The project can be done individually or in groups of **maximum 2 students**. If you decide to work in groups, send me an email of both names by **Tuesday November 3, 4:30 PM**. I advice you to pick your partner as soon as you can and **start early** on the project.

1.1 Part 1: One Unique Discussion Group

In the first part of this project, you will consider that all clients belong to one and only one group. A client joins by connecting to a dedicated server (a standalone process) and is prompted to enter a non-existent user name in that group. Note: in this project, you are *not* required to implement any user authentication mechanisms. The server listens on a specific non-system port endlessly. A client keeps track of all users that join or leave the group (when a user joins or leaves the group, all other connected clients get notified). When a user (or client) joins the group, he/she can only see the last 2 messages being posted on the board by other clients who joined earlier. A list of users belonging to the group is displayed once a new user joins (in this part, the list represents all users/clients that have joined earlier). When a user posts a new message, other users in the same group should see the message being posted. Messages are displayed in the following format: "Message ID, Sender, Post Date, Subject." A user can retrieve the content of a message by contacting the server and providing the message ID as a parameter. Your client program should also provide the option to leave the group. Once a user leaves the group, the server notifies all other users in the same group of this event.

1.2 Part 2: Multiple Discussion Groups

Extend Part 1 to allow users to join multiple discussion groups. Once a user is connected to the server, the server provides a list of 5 groups. The user can then select the desired group by id or by name. A user can join multiple groups. Remember that a user in one group cannot see users in other groups as well as the messages they have posted.

2 Demo over CSE environment

A demo may be required on the due date.

- You should use the cse environment to run both the server application and a number of client applications (minimum two clients). You can use a subset of the following machines: alpha.cse.ohio-state.edu, beta.cse.ohio-state.edu, gamma.cse.ohio-state.edu, epsilon.cse.ohio-state.edu, zeta.cse.ohio-state.edu, and eta.cse.ohio-state.edu. **IMPORTANT**: To avoid possible last-minute compilation or run-time issues close to the demo time, you

should test your code on such machines beforehand. I do not recommend using the machines during development (use your own workstation and run both the server process and client processes locally). This avoids any possible memory leaks or crashes to your cse environment during development.

- If you choose to work in a group of two, both students are expected to demo different parts of the project (you should plan with your partner accordingly).

2.1 Connecting To Your Environment

For information on how to connect to such machines, please refer to `https://cse.osu.edu/computing-services/resources/remote-access`. In a nutshell,

2.1.1 UNIX/Mac

```
ssh -X {username}@{machine-name}.cse.ohio-state.edu
```

2.1.2 Windows

Use PuTTY (`http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html`)

2.2 Installing Compilers

If your CSE environment does not have the compiler you are looking for (e.g., javac), do the following:

- Connect to one of the CSE machines mentioned above (e.g., alpha.cse.ohio-state.edu)
- In the terminal, type **subscribe**
- Choose the number corresponding to the package you want to install (e.g., 7 for latest version of JDK, 16 for PYTHON-3) and hit enter. Remember, since you will be using your own workstation for development as recommended above, make sure that your code will compile at all times using the packages you are installing on your CSE environment. Best way is to install the same version of the compiler, that you subscribe to, onto your own workstation to avoid any compilation errors when you port your code to your CSE environment
- Type **q** and hit enter
- Confirm the installation of the chosen package(s)
- Disconnect by typing **exit**
- Re-connect to machine using ssh, and now you should be able to run the compiler command (e.g., javac, python3)

2.3 Uploading Your Code

If you are wondering how to upload your code to the CSE machines mentioned above for testing and demo purposes, these machines also act as FTP servers. Therefore, you may use an FTP client (e.g., Filezilla Client `https://filezilla-project.org/download.php?type=client`). As we learned in class, FTP operates over ports 20 and 21. However, for security reasons, the standard TCP ports 20 and 21 are blocked. To still be able to upload your code, one way is to use the SFTP protocol to run FTP over SSH (port 22).

2.3.1 UNIX/Mac

ftps command comes pre-installed. You can still use the FileZilla Client (see next) if you prefer a GUI-based FTP client.

2.3.2 Windows

Use FileZilla Client (`https://filezilla-project.org/download.php?type=client`). When connecting to machine, use **22** as the port number.

3 Some Hints

3.1 Protocol Design

Like in any client-server application, both the client and server have to communicate using an agreed-upon protocol. Protocol messages should have a format understood by the client and server alike (as an example, think about HTTP request and response messages we studied in this course and in project 1; HTTP request messages contain header information parsed and understood by the server, and response messages have body section carrying the data back to clients). You may use plain text, or consider using XML or JSON representations for all or parts of the protocl messages.

3.2 Server Implemenation

- Since you are required to use **unicast** sockets, the server should keep a list of all connected clients (i.e., all client sockets).
- Consider using mulithreading, one thread for each TCP connection. Similar to what you did in project 1 "Multithreaded Web Server". In addition, the TCP connection should always be open until the client decides to leave the group and disconnects from the server.

4 Grading (Total 50 Points)

- Correctness (70%): The program should correctly perform the tasks described above. You will risk receiving a **zero** or very low credit for the entire project if your code does not compile, or the program does not perform as expected. 50% will be given to the first part and 20% to the second part.
- Usability (15%): Your program should be user friendly. You can use special input commands to handle user's requests. For example, you can use the following set of console commands with options:
  - a %connect command followed by the address and port number of a running bulletin board server to connect to.
  - a %join command to join the single discussion board
  - a %post command followed by the message subject and the message content or main body to post a message to the board.
  - a %users command to retrieve a list of users in the same group.
  - a %leave command to leave the group.
  - a %message command followed by message ID to retrieve the content of the message.
  - an %exit command to disconnect from the server and exit the client program.

  For Part 2:
  - a %groups command to retrieve a list of all groups that can be joined.
  - a %groupjoin command followed by the group id/name to join a specific group.
  - a %grouppost command followed by the group id/name, the message subject, and the message content or main body to post a message to a specific discussion group.
  - a %groupusers command followed by the group id/name to retrieve a list of users in the given group.
  - a %groupleave command followed by the group id/name to leave a specific group.
  - a %groupmessage command followed by the group id/name and message ID to retrieve the content of the message posted earlier on a specific discusion group.

  There will be **additional points** given to a graphical user interface (**GUI**) implementation.
- Documentation (15%): Documenting your code and using more expressive variables will be taken seriously. In addition, provide a **Makefile** and a **README** file. The README should contain instructions on how to compile and run your server/client programs, usability instructions if different from suggested above, and a description of any major issues you came across and how you handled them (or why you could not handle them).

5 Submission Instructions

- Place both your cleint and server source code, README, and a Makefile in one directory and submit by the due date (Do NOT submit binaries):

  ```
  submit c3461ad lab1 <your-code-directory>
  ```

  For group work, only one submission is adequate but make sure you provide your name and your partner's name in the README.