CS 408 - Computer Networks - Fall 2023
Course Project
DiSUcord, Simple Message Networking via Channels

Deadline: December 15, 2023, Friday, 22:00 (be aware that the deadline is not midnight)
Project Demo: week of December 18 (Schedule to be announced)

Second Chance Demos will be done within the last week of the classes (week of January 1, 2024).

Group Work

You can work in groups of **two** people. If you cannot find a group mate, you can do the entire project alone. We will not group people and will not facilitate groupings. However, you can use the WhatsApp group to find a group member.

Equal distribution of the work among the group members is essential. Both members of the group should submit all the codes for both the client and the server. Both members should be present during the demos. In case of any dispute within the group, please do not allow the problematic group member to submit the code, so that he/she will not be graded. However, if a group member submits the same code, then the other member automatically accepts his/her contribution.

Second chance demo rules

After the announcement of the project grades, you will have three days to correct the problematic parts to make a second demo. However, you will be able to claim only half of the deducted points in the second demo. No group changes are possible during the second demo.

Introduction

In this project, you are going to develop a message networking application called *DiSUcord* (SU version of Discord) by implementing *Client* and *Server* modules. (i) The *Server* module manages messages and channels and (ii) the *Client* module acts as a user who subscribes to channels, sends messages to channels, and view messages sent to the subscribed channels.

The server listens on a predefined port and accepts incoming client connections. There might be one or more clients connected to the server at the same time. Each client knows the IP address and the listening port of the server (to be entered through the Graphical User Interface (GUI)). Clients connect to the server through this port and identify themselves with their usernames. The server needs to keep the usernames of the currently connected clients in order to avoid the same name being connected more than once at a given time. Clients connect only via usernames (i.e. no password or other type of security).

In DiSUcord, there are two fixed channels, namely "IF 100" and "SPS 101". For the sake of simplicity, these channels' names will not change and you do not need to get their names as input. A client may subscribe to one or both of the channels through the client GUI at any time after the connection. A client can send messages only to the subscribed channel(s). Moreover, the clients subscribed to a particular channel will receive messages sent to that channel.

Details of Client and Server GUIs and Functionalities

Since there are two fixed channels in DiSUcord, you may design your client GUI with necessary components of each of them separately. The functionality for one channel includes a rich text box for displaying messages sent to the channel, a button to subscribe/unsubscribe (you may have two buttons depending on your design), a text box for composing messages, and a button for sending the composed message to the channel. Your client GUI should also be clear to understand the channel subscription statuses of that particular client.

Since the server will not accept duplicate connected client names, unsuccessful connections are possible. The client GUI should show the success/failure status for the server connection as well.

The server is the hub of everything. As mentioned above, connected clients and channel subscriptions are handled by the server. Moreover, messages sent to channels are multicast to all of the subscribers (including the one who sent the message) of that channel by the server.

Multicasting should be implemented explicitly. This means that the server should send a particular message only to the channel subscribers. In other words, sending a message to all of the clients (i.e. broadcasting) and letting the client side decide on whether to show a message based on subscription is definitely not acceptable (we mean it, such a project will not be accepted).

Here please remark that if a client has subscribed to a channel, it can view messages sent to that channel after its subscription, but not the messages sent before.

Additionally, clients have the capability to unsubscribe from a channel they have previously subscribed. Upon unsubscribing, the client will cease to receive new messages sent to that specific channel and of course, will not be able to send messages to it. On the other hand, the client later can subscribe again after it unsubscribes; these subscribe/unsubscribe operations can be done several times.

The server's GUI acts as a monitoring tool to track and display these server-client interactions for administrative purposes. For this purpose, on the server GUI, there should be a rich text box that displays all actions taking place on the server. This includes (but not limited to) events like client connections, clients subscribing channels, sending messages, unsubscribing from channels, disconnections, and any other relevant actions related to client interactions with full details. Moreover, the server GUI should have three more rich text boxes that show (i) the list of currently connected clients, (ii) the list of clients currently subscribed to channel IF 100, and (iii) the list of clients currently subscribed to channel SPS 101.

In addition, all reasonable/unreasonable user actions must be handled properly and rationally. We cannot list all of them here; use your commonsense for them! Moreover, client and server GUIs must reflect every detail (even if it is not written explicitly here).

For programming rules and submission specifications, please read the corresponding sections at the end of this document.

Other General Specifications

Server Specifications:

- There is only one server running on this system.
- The port number on which the server listens is not to be hardcoded; it should be taken from the Server GUI.
- The server will start listening on the specified port. It has to handle multiple clients simultaneously. To do so, whenever a client is connected to the listening port, the corresponding socket should be added to a list and the server should continuously accept other client sockets while listening.
- All activities of the server should be reported on the Server GUI in full detail. We cannot grade your project if we cannot follow what is going on; so the details contained in this text box are very important.
- Server must handle multiple connections. At the same time, one or more clients can send messages and subscribe to channels.
- Connected clients' usernames must be unique; therefore, the server must keep track of connected clients' names. If a new client comes with an existing username, the server must not accept this new client.
- When the server application is closed (even abruptly), nothing should crash! Also, the process in the operating system regarding the server should be terminated properly.

Client specifications:

- The server's IP address and the port number must not be hardcoded and must be entered via the client GUI.
- There could be any number of clients in the system.
- If the server or the client application closes, the other party should understand disconnection and act accordingly. Your program must not crash!
- All activities of the client should be reported on the client GUI including connection status, subscription/unsubscription information, messages, etc. We cannot grade your project if we cannot follow what is going on, so the details contained in this GUI are very important. All reasonable information must be shown even if they are not explicitly mentioned here.
- Each client must have a unique username. This username must be entered using the client GUI. This username is also sent to the server. The server identifies the clients using their usernames.
- Each client can disconnect from the system at any time. Disconnection can be done by pressing a disconnect button on the client GUI or by just closing the client window.
- If the client application is closed (even abruptly), nothing should crash! Also, the process in the operating system regarding the client should be terminated properly.
- Both connection and message transfer operations will be performed using TCP sockets.

Programming Rules

- Official course language is C#, but you can use any other language that supports TCP sockets.
- Your application should have a graphical user interface (GUI). **It is not a console application!**
- You must use TCP sockets as mentioned in the socket lab sessions. You are not allowed to use any other type of socket. Especially, please do **not** write us to ask for permission to use Websocket.
- In your application when a window is closed, **all threads related to this window should be terminated.**
- Client and server programs should be working when they are run on at least two separate computers. So please test your programs accordingly.
- Your code should be clearly commented. This affects up to 5% of your grade.
- Your program should be portable. It should not require any dependencies specific to your computer. We will download, compile and run it. If it does not run, it means that your program is not running. So do test your program before submission.

Submission

- Submit your work to SUCourse+. Both group members must submit.
- Delete the content of debug folders in your project directory before submission.
- Create a folder named **Server** and put your server related codes here.
- Create a folder named **Client** and put your client related codes here.
- Create a folder named **XXXX_Lastname_OtherNames**, where XXXX is your SUNet ID (e.g. yasinughur_Ughur_Yasin). Put your Server and Client folders into this folder.
    - Compress your XXXX_Lastname_OtherNames folder **using ZIP or RAR**.
- You will be invited for a demonstration of your work. Date and other details about the demo will be announced later.
- 24 hours late submission is possible with 10 points penalty (out of 100).

We encourage the use of our WhatsApp group for general questions related to the project. For personal questions and support, you can send email to course TAs.
Good luck!

Yasin Ughur - yasinughur@sabanciuniv.edu
Alize Sevgi Yalçınkaya - alizesevgi@sabanciuniv.edu
Simge Demir - simgedemir@sabanciuniv.edu
Albert Levi