ITF23019 - Parallel and Distributed Programming

Final Project

Submission deadline

Friday 8 May 2020, 10:00 at Canvas.

Instruction

The report of this final project must be submitted as a single PDF file. You can choose between scanning handwritten notes or typing the solutions directly on a computer. Scanned pages must be clearly legible. The PDF file must contain your name, course name, assignment number, page number on EVERY page.

The source code of the project must be submitted along with the PDF report. All of the files must be put into one folder and compressed into one ZIP file.

Students do not require to make an oral presentation for the project. However, you have to demonstrate that your program can run properly. The demonstration may take about 15 to 20 minutes. Questions may be asked during the demonstration.

Final Project Description

In this project, the students are asked to develop a MiniChat application. Students are free to choose your preferred programming languages to implement your application as long as the language supports the concepts about parallel, concurrent and distributed programming discussed in class. The MiniChat to be developed in this project is a GUI software that enables users to exchange text messages with each other. You can refer to Yahoo Messenger, Skype or Facebook Messenger as examples of a text messenger. Also, you can have a look at this tutorial [1] to understand the features of Yahoo Messenger.

The main focus of this project is only sending text messages. You are free to choose either Peer-To-Peer or Client-Server communication model for the protocol of your MiniChat. In Peer-To-Peer model, each user can act as a client or a server when exchanging messages with each other. In the Client-Server model, there will be a central server that receives messages from all senders and distributes them to the corresponding receivers. In either communication model, there must be one central server, called MiniChatServer, to handle user credentials (i.e, username, password, status, avatar, etc.) as well as other functions related to the clients (i.e., user login, user creation, get the list of added users, etc.). A

[1]https://www.youtube.com/watch?v=qXRzBPtXuI8

MiniChatClient which is used by the users to log in to the server and retrieve information about their accounts. This is similar to the usage of Skype or Yahoo Messenger where the users have to sign in to the server, get a list of their added friends and send messages to their chosen friends.

1. GUI Design - 10%

Figure 3 illustrates a basic GUI for the MiniChatClient, based on the GUI of Yahoo Messenger. You are free to have your own design of the GUI as well as add more features to the application. When implementing the underlying functionalities of a GUI application, please take into account the application of multi-threading programming in order to reduce the response time of the GUI. Using only one main thread in the implementation may freeze the GUI while the thread is executing underlying functions.

2. MiniChatServer - 20%

The MiniChatServer, as described above, manages information of its users. MiniChatServer needs not to have graphical user interfaces. Figure 1 presents the basic data structure for the MiniChat application. You could add more classes or attributes necessary to your implementation. Figure 2 illustrates the communication between a user using MiniChatClient and the MiniChatServer. The MiniChatServer has to be implemented as a REST server with APIs presented in Table 1.

Table 1: REST API of MiniChatServer

Function	URL subpath	Method	Input	Output
Login and update online status	/user/login/	POST	Username, Password, Online status	OK/BAD REQUEST
Logout and set online status to offline	/user/logout	POST	Username	OK/BAD REQUEST
Create a new user	/user/create/	POST	Username, Password	OK/BAD REQUEST
Get user information	/user/{username}/	GET	-	user information
Add new friend	/user/{username}/	PUT	list of usernames to be added	OK/BAD REQUEST

You are free to update or add more functions into Table 1 in accordance with your implementation. Please use JSON as the format for the Input/Output content of these functions.

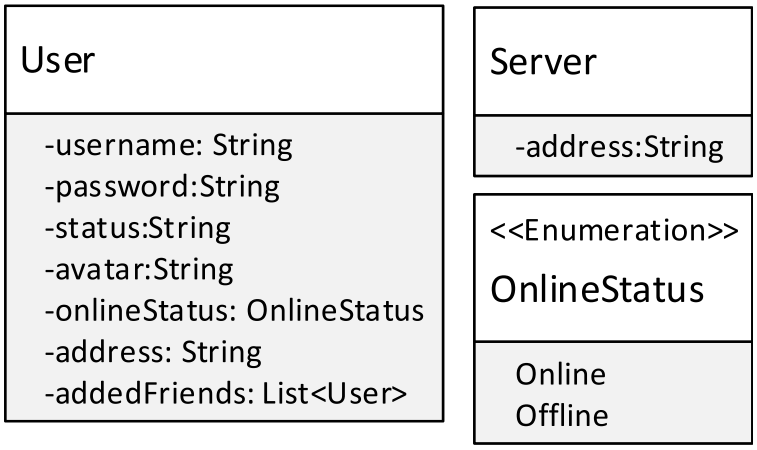


Figure 1: MiniChat Data Structure

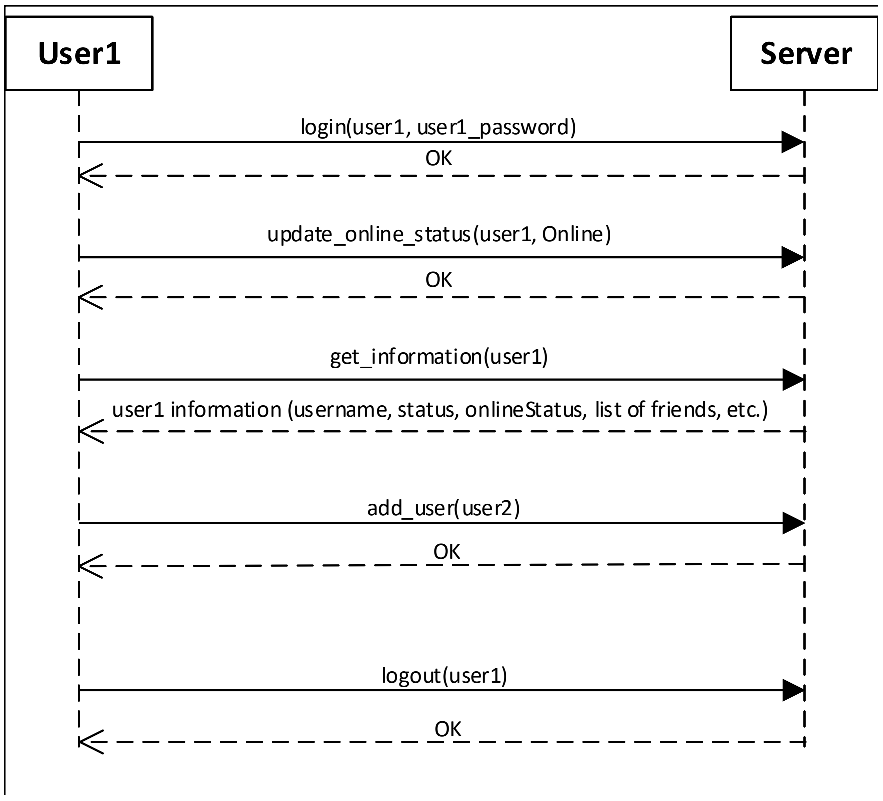
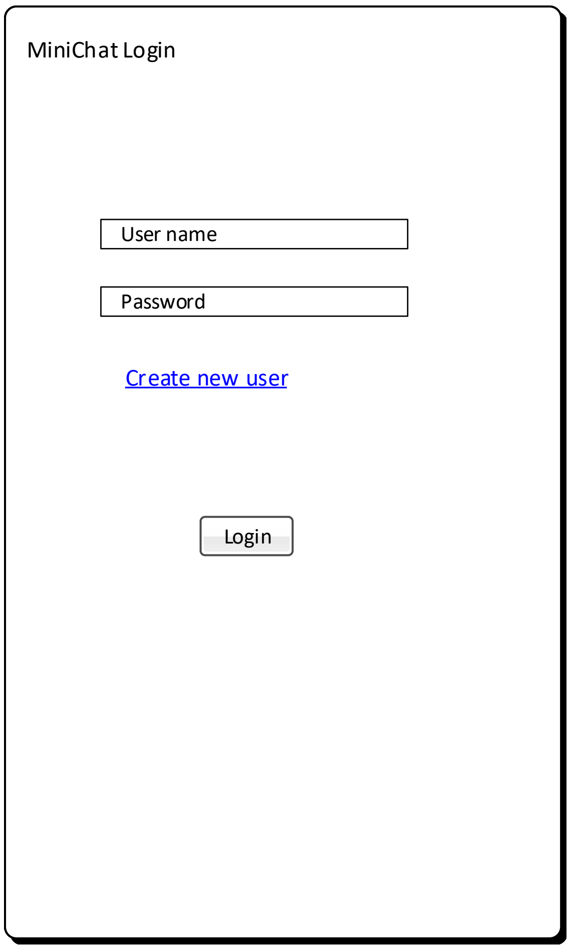

User1
Server
login(user1, user1_password)
OK
update_online_status(user1, Online)
OK
get_information(user1)
user1 information (username, status, onlineStatus, list of friends, etc.)
add_user(user2)
OK
logout(user1)
OK

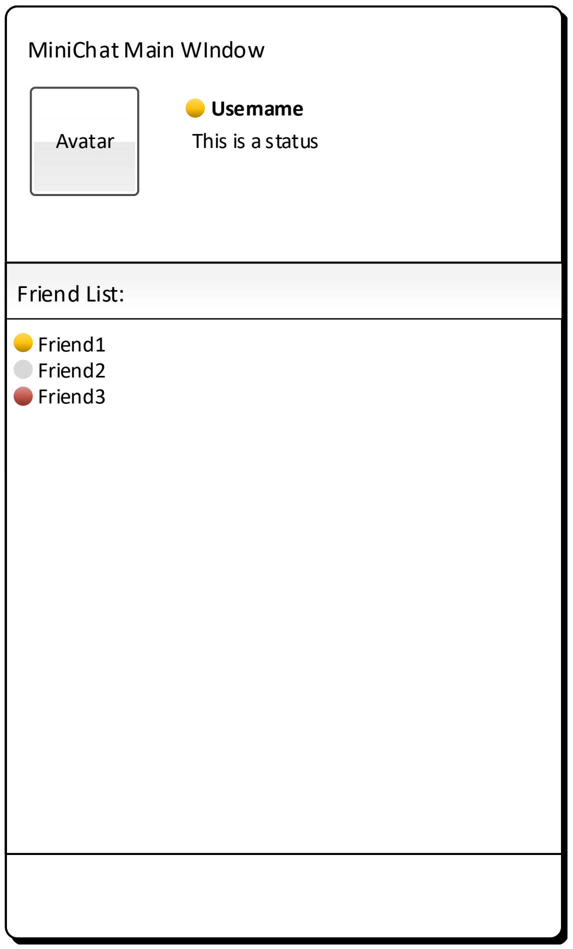
Figure 2: Communication between MiniChatClient and MiniChatServer

3. MiniChatClient - 20%

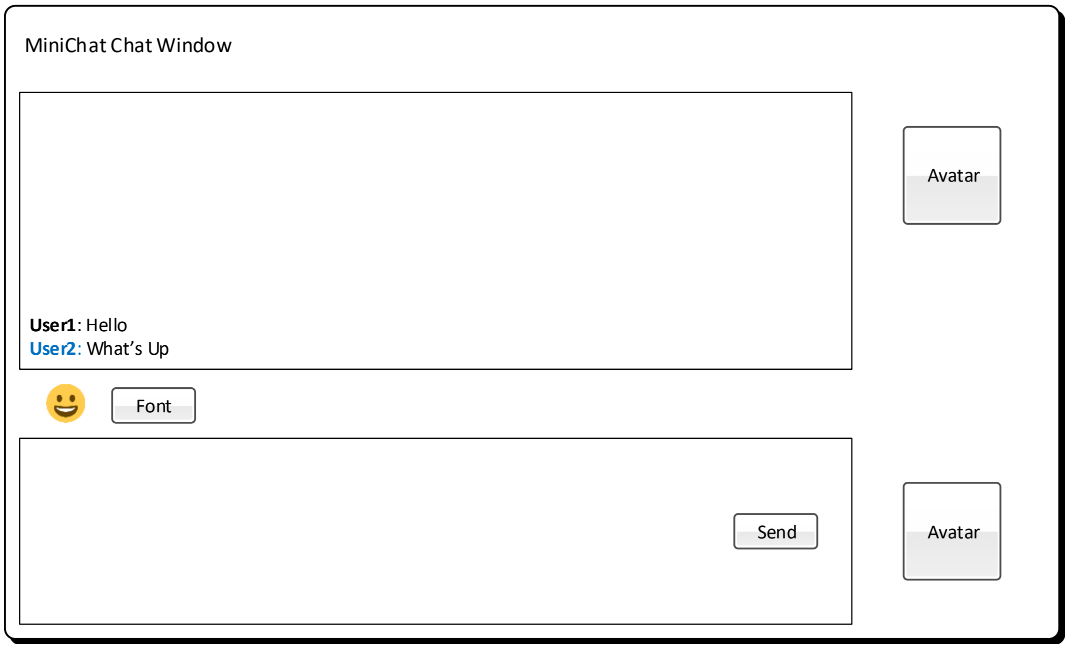
The MiniChatClient is the main GUI application that enables the users to login/logout into the MiniChatServer, update their information, and send messages with other added friends. An example of a graphical interface of the MiniChatClient is showed in Figure 3.



(a) Login window



(b) Main window



User1: Hello
User2: What's Up

(c) Chat window

Figure 3: MiniChatClient example user interface

Regarding the communication model between the users, as mentioned above, you can choose either Peer-To-Peer or Client-Server models for your MiniChat application. Figure 4 and 5 present the interaction of the users in both models.

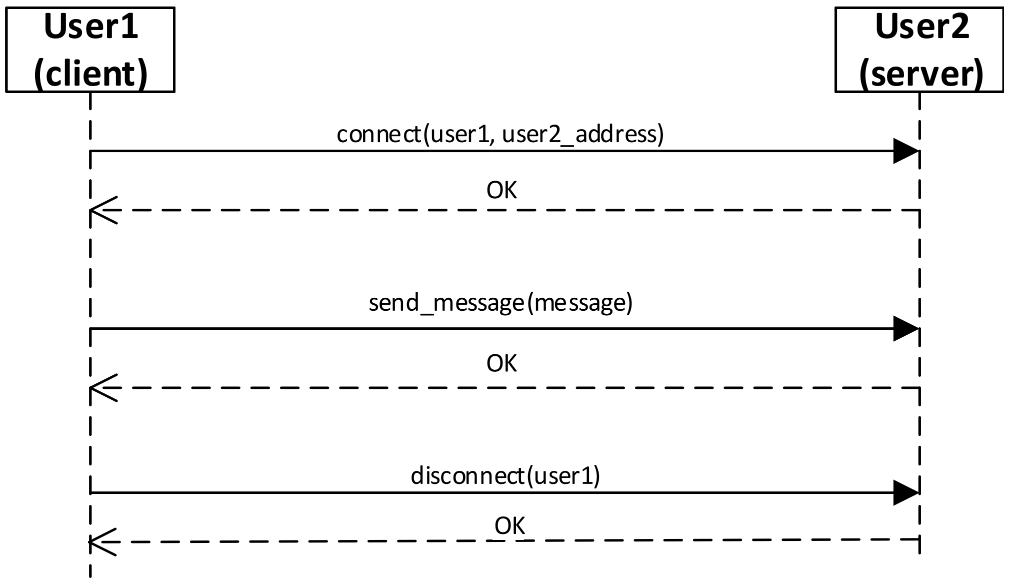


Figure 4: Peer-To-Peer Communication Model

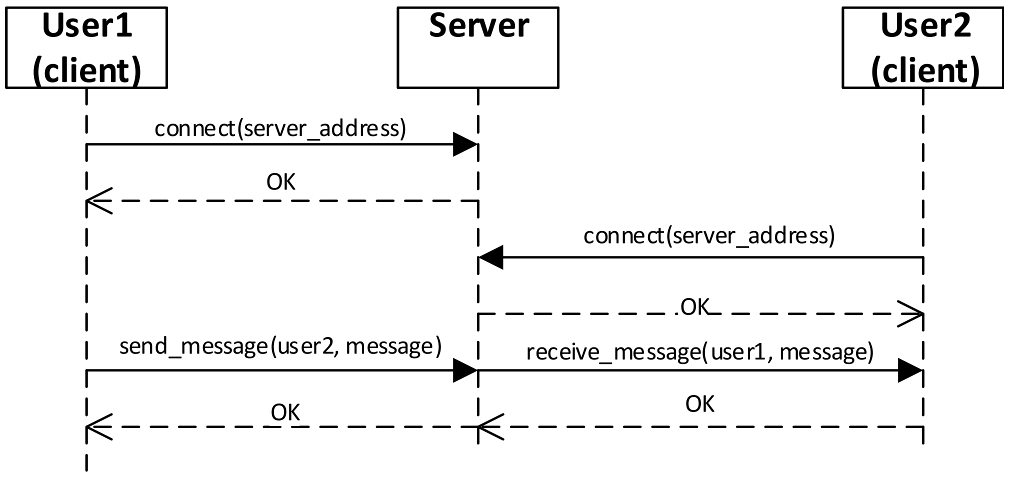


Figure 5: Client-Server Communication Model

For the Peer-To-Peer model, one user would act as a server (TCP Socket server) and the other would be the client. In this model, every MiniChatClient has to start a local server in order for the other users to connect and exchange messages. The attribute `address` of the `User` class in Figure 1 is meant for socket address of the user. If you do not employ this model, there is no need to use this attribute.

For the Client-Server model, there will be another central server (different from the REST MiniChatServer) which acts as an intermediate node that receives messages from all senders and forwards the messages to the corresponding receivers.

4. Advance Features - 10%

You are free to add more advance features to your MiniChatClient application. Several suggestions are updating avatar for the user, adding style and font for the message, emoji, buzzing [2], etc.

5. Report - 20%

The report should be from 10 - 30 pages. In the report please add any kind of diagrams, models that describe the implementation of your application. A manual of how to set up and run your application should also be added to the report.

6. Demonstration - 20%

For the demonstration, you do not need to make any formal presentation. Instead, you just need to demonstrate the features of your application. Questions could be asked during the demonstration just for clarification of the implementation. There will be no question beyond the scope of your work. Please note that the demonstration is mandatory in order to determine if you fulfill the requirements of the project.

[2] https://www.youtube.com/watch?v=WfcX2ise7ro