Distributed Systems Programming
ICT2107

Group Project 1

WhatsChat: A Fully Distributed Messaging Software

1. Introduction

WhatsChat is a software that allows clients within the same network domain to chat. It aims to make use of broadcasting and multicasting messaging to enable seamless messaging across nodes with minimum setup required. The challenge of this project is in the design and implementation of protocols that support complete decentralized organization of directory services and maintenance of group membership. As such, there is no centralized server and that peer nodes can enter and leave the messaging group dynamically.

In this project, your group is tasked to extend the multicast chat application to create WhatsChat. The requirements are as follow:

(a) **Client**: The Client is an autonomous node that exercises distributed processing and management of messaging sessions across the communicating peer nodes. It requires no centralized coordination. Instead, it coordinates with other instances of clients to implement the distributed messaging software.

(b) **Well-known Address**. Each Client must run a thread that listens to a *well-known multicast address* of 230.1.1.1. This multicast channel serves several purposes as described below.

   a. **User Registration**. A user must register with the system by entering a unique user id of not more than 8 characters (e.g. SiuLam). The user id is case sensitive, and must not contain any spaces and must not begin with a numeric. When registering, the uniqueness of the user id is checked by the Client against other clients in the network via the well-known multicast address (see point (b)). If there is a duplicate, the client node holding the user id will inform the Client. In which case, the user can try to re-enter another user name. Subsequently, if the holder of the original user id leaves the network, another Client can request to register the same user id.

   b. **Successful Registration**. On successful registration, the Client will announce its user id to all the client nodes in the network. In which case, all the client nodes (including the Client) will update its list of all online user ids.

(c) **User Profile**: This feature is optional and will be considered as an extension. Each user may create a profile of himself or herself that is linked to the user id. User may upload a photo and short description. Another user may click on a user id and be able to view the profile.

(d) **Group Messaging**. A user initiates a group communication session by first creating a messaging group, which is comprised of a selection of group members (much like WhatsApp group messaging). The user is able to pick and select the members to be in the group. Each group is assigned a name by the user and it will be checked against all running processes (of the same app) for any duplicate in the group name (Hint: use 230.1.1.1 to check, see point(b)). If there is a duplicate, it will flag a message to the user. Each new group will be assigned a new multicast address with the format of 230.1.X.Y. Your group is free to design a method of deriving X and Y. Once selected, online

members that are selected will receive a notification (via 230.1.1.1, see point(b)), and their group list will automatically be updated to show that they are now members of the newly created group.

(e) **Communication**. A user can select the group that he/she wants to be active on. For example, John may be a member of ICT2101, ICT2102 and ICT2107. He may select ICT2107 as the active group to participate. In which case, only messages pertaining to ICT2107 will be displayed within the conversation text box. Messages pertaining to other groups will be stored (but not displayed in the conversation text box). Subsequently, John may switch to another group, say ICT2101 and make it the active group that he desires to participate in.

(f) **Membership**. Once a member is enlisted to a group, he/she is able to add or delete members to and from the group, and to change the name of the group (e.g. change ICT2107 to Distributed Computing). Also, a user may choose to leave the group. Once a user leaves the group, he or she can only become a member again by having an existing member add him or her to the group. When a member first joins the group, the Client will request from one or more member(s) (via 230.1.1.1, see point(b)) the latest 10 messages (or less) that were circulated within the group.

(g) **User Interface**. Shown below is an example of the user interface (Figure 1). Feel free to design your own unique interface with the ultimate aim to improve interactivity and usability.

(h) **Storage.** When the user quits the program, all state information, including the user id, profile (if implemented), group id, membership, etc, will be saved to a local file and will be retrieved when the user starts the program again. It is optional but will be considered as extension to store and retrieve the historical messages in the group.

(i) **Extensions**. If you have made significant extension(s) to the project, please include a chapter in your report to describe the extension(s) made (e.g. point (c), point (h)). This will help me to quickly zoom in and assess the extra efforts made to the project.

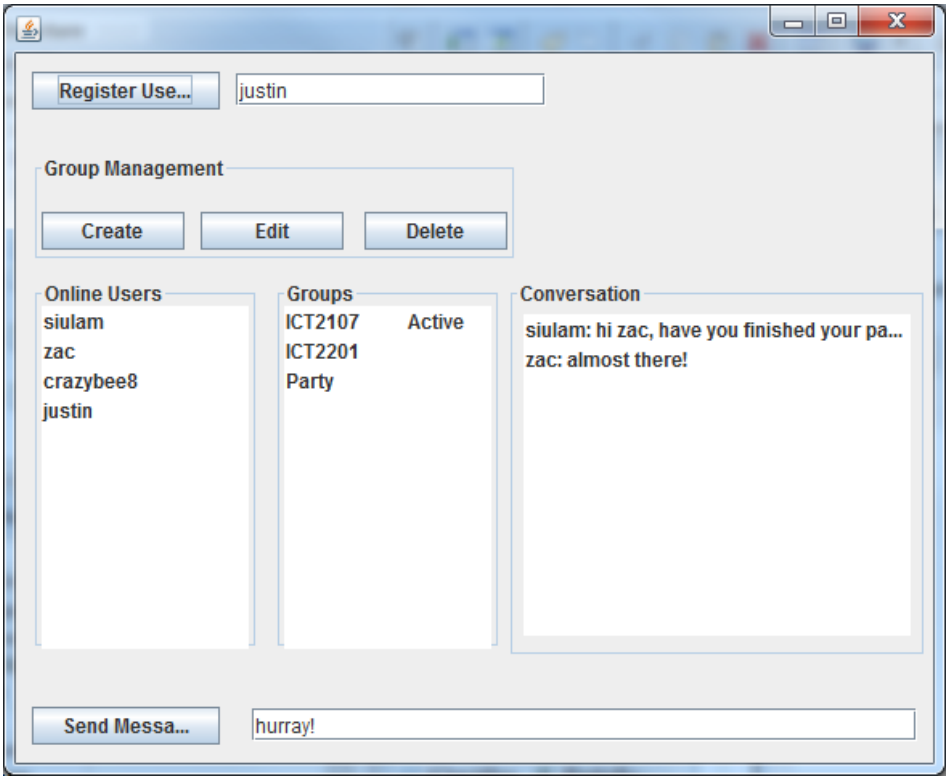

Figure 1 Sample GUI

2. Report

Each group is required to write a report detailing the design of the system. The report should not be just a user manual. It should detail the software architecture of the design in terms of the major software components, as well as the system architecture. Make use of notation tools such as UML, data flow or flow control charts to describe the design.

The report should not be more than 10 pages in length.

On the last page, detail the contributions made by each member of the group. For example:

> Angela Chan: In this project, I am responsible for the development….

3. Deliverables and Deadlines

The deadline of submission for the group project is 11:59PM, 17 Feb 2020 (Monday). You are required to submit a zipped file with a filename named after your group ID. For example, if your group is Group 8, the zip file should be named as Group08.zip. Failure to properly name the file will result in marks being deducted.

The zip file should contain the following directory structure:

Level 1	Level 2	Level 3	Level 4
GroupX.zip			
	GroupX		
		JAR	GroupX.jar
		Source	Put all your source code here
		Reports	GroupX.pdf
		Presentation	GroupX.ppt

As shown above, the zip file should have a root directory named after the group ID, e.g. Group08. Following, it should contain four directories, namely JAR, Source, Report and Presentation:

- The *JAR* directory should contain the executable file. You should export your program to an executable jar file (google the web to find out how to make an executable jar file) so that it makes it easy for me to run and test your program.
- The *Source* should contain the source files (including codes).
- The *Report* directory should contain only the PDF report.
- The *Presentation* directory should contain only the slides used in your project demonstration.

Failure to adhere to the structure may result in marks being deducted.

Upload the GroupX.zip file to the LMS on or before the deadline. A penalty will be imposed for each day that you are late in submitting the file.

4. Marking and Peer Evaluation

The marking will be structured as follows:

- Design and Implementation of Application (80%)
  - User registration 10%
  - Adding a friend 10%
  - Creating group 10%
  - Communicating in group 15%
  - Extensions 15%
  - Software stability 10%
  - Software usability and HCI 10%
- Report
  - Organization and Structure 5%
  - System Design, Evaluation and Methods 10%
  - User Manual 5%

Total	100%

A peer evaluation will be conducted. The purpose is to recognize individual effort and contributions in a team project by peer members. It is important to recognize that it is not peer grading but really recognizing and acknowledging differential peer contributions by members in the project.