Distributed Systems
COMP90015 2016 SM2
Project 2 - Extended Multi-Server Chat System

1. Synopsis

This assignment is to extend the Multi-Server Chat System in the assignment 1. The project primarily focuses on addressing the following challenges of the system:

- **Security**
- **Failure**
- **Scalability**

Moreover, this project should be demonstrated using **Virtual Machines (VMs)** running in the **Cloud** environment. More details can be found in the later part of the description.

Optionally you can design and implement a **Graphical User Interface (GUI)** for the Chat Client. Extra points for this part will be awarded. The source code of the chat client program with **Command Line Interface (CLI)**, used in the previous assignment, will be shared with you.

In this assignment you have the freedom to design and implement your own protocols and to add new components to the previous project.

This is a **group** assignment. You need to make a group of 3 or 4 people.

In the following sections we provide requirements regarding each of the above.

2. Security

To address security at least three main issues must be addressed.

1) All communications (between servers and between clients and servers) in the system should be happening via **secure channels** (encrypted communication).

- **Recommendation**: Use *TSL/SSL* TCP connections with self-signed certificates.

2) You need to provide an **authentication** mechanism for your project. Only registered and authorized users with valid *username* and *password* can log in into the system (No need to impelement a registration method. This can be manual.)

- **Recommendation**: Create a central authentication server with some predefined users.

Extra 5 points will be awarded if you can enable authentication using *OAuth2* protocol with Facebook, Google or Twitter accounts. This is **only recommended** to those who have good programming skills.

3) You need to add the authentication feature to the given chat client program to support your authentication mechanism.

Note that chat servers only allow connections for authorized clients passing the authentication phase.

The username is not necessarily the same as *client identity*. For example my username might be *anadjaran@unimelb.edu.au* or *anadjaran* and my chat client id is *adel*.

3. Failure Handling

Your project should address the following failure:

If a chat server **crashes** or **stops responding**, other chat servers should **detect** this situation. All chatrooms on that server should be removed from the list and clients should not be redirected to that server anymore.

- **Recommendation**: Use heartbeat signals generated at regular intervals to check if the other servers are working correctly.

You do not need to handle failures for the chat clients. Clients are simply getting disconnected in the case of a chat server crash. The chatrooms and other data associated to the user will be deleted in a similar way to the previous project.

4. Scalability

For the sake of scalability, your system has this feature that the *system admin* can add **new servers** to the system manually.

The newly added servers become fully functional components of the chat system and clients can directly connect to these chat servers. Thus, these servers participate in all the *Lock* and *Release* communications among chat servers to handle client identities and chat rooms.

- **Recommendation**: Add new JSON messages to the communication protocol of servers to support this feature that a new server can introduce itself to the system.

5. A Chat client program with GUI

Design and implement a **chat client** program with a **GUI**.

Please note that creating chat client with GUI is **optional**. You will receive extra 10 marks (100 + 10) for addressing the requirements of this section.

Source code for the chat client program used in the previous assignment will be shared with you as a sample. You can reuse the code or develop your own code from scratch. If you decide to implement your own chat client, you can use the programming language of your choice for this purpose.

The chat client should support all the features of the previous project plus the authentication.

6. Demo

You have to **demonstrate** the project. For the demo purpose you need to use an Infrastructure-as-a-Service (IaaS) **Cloud** to run your chat servers on multiple **VMs**.

- **Recommendation**: Use Nectar or alternative cloud services to host your Chat Servers and run the demo.

More details about Nectar Cloud will be given in the tutorials.

7. Your Report

Use 10pt font, double column, 1 inch margin all around. Put the names of all the group members, as well as their university username, and student number at the top of your report.

In up to 1500 words discuss:

- The overall **architectural model** of you application (**use diagrams**).
- How did you implement the **communications** between components of the system? Use **UML Sequence diagrams** to illustrate the main communication patterns.
- **Technologies** did you use and why?
- Which functionality did each **one of the team** complete? You will be asked specific questions based on that.

Note that you will receive marks based on your contributions to the project. You should be familiar with all the components of the project even if you take care of only one of the required features. Just preparing the report is not enough contribution for a member of the team to pass the assignment.

- **Recommendation**: It is strongly recommended that the report is prepared by all team members.

8. Submission

You need to submit the following via LMS:

- Your report in **PDF** format only.
- Your source and executable files in a **.ZIP** archive only.

Submissions will be due on **Wednesday 19th of October 11:59pm**.

This is a hard deadline that will not be extended as the week after is SWOT Vac. Demo time, date, and place for each group will be arranged on the 20th and 21st of October and will be announced close to the submission date.