COMP6231 /1 – Sections BB – Summer 2020

PROJECT
Due: August 12th

Important Note

- Your program should be compiled, executed and return the expected results; otherwise a mark 0 (zero) will be assigned.
- The work can be realized individually.
- The delivery must be made no later the date due using the Website submission as mentioned in the course outline.
- If you are having difficulties understanding sections of this project, feel free to email the Teaching Assistant. It is strongly recommended that you attend the tutorial sessions which will cover various aspects of the assignment.

Software Failure Tolerant/Highly Available Distributed Player Status System (DPSS)

In this project, you are going to enhance your CORBA implementation of the Distributed Player Status System (DPSS) developed in Assignment #2 to be software failure tolerant or highly available using process replication. This project is suitable for teams of 4 students, as described in the following:

For this project, extend your Distributed Player Status System (DPSS) implementation from Assignment #2 to tolerate a single software (non-malicious Byzantine) failure using active replication.

Your actively replicated DPSS server system should have at least three replicas each running a different implementation (in different hosts on the network).

One of the replicas in the group is the designated leader and receives requests from clients through a CORBA Front End (FE). The leader of the server group broadcasts a client request (received through the front end) atomically to all the other replicas in the group using a reliable FIFO broadcast mechanism, receives the responses from them and sends a single correct response back to the client (through the front end). The leader also informs the Replica Manager (RM) if a replica produces incorrect result so that the RM can replace a failed replica that produced incorrect result three times successively with another good one.

Since the entire server system (Replicas, FE, and RM) usually runs on a local area network, the server replicas, FE, and RM communicate among them using the unreliable UDP protocol. However, this communication should be made reliable and FIFO in order to avoid message loss and guarantee correctness. Specifically do the following:

- Assuming that server failure could be due to software bugs (i.e. non-malicious Byzantine failures), design your actively replicated, software failure-tolerant server system;
- Modify the server implementation from *Assignment #2* so that it can work either as the group leader or as a regular server. The group leader receives a request from the CORBA front end, FIFO broadcasts the request to all the server replicas, receives the responses from the server replicas and sends a single correct response back to the client. The regular server receives requests from the leader, executes the request and sends the response back to the leader;
- Design and implement the front end (FE) which receives a client request as a CORBA invocation, forwards the request to the leader process, receives the result from the leader and sends it back to the client;
- Design and implement the Replica Manager (RM), which creates and initializes the actively replicated server subsystem. The RM also manages the server replica group information, maintains if a replica has produced incorrect result and replaces a failed replica that produces incorrect result three times successively with another good one;
- Design and implement a reliable FIFO broadcast subsystem over the unreliable UDP layer;
- Integrate all the modules properly, deploy your application on a local area network, and test the correct operation of your application using properly designed test runs. You may simulate a software failure by returning a random result.

Submitting Project

- Naming convention for zip file: Create one zip file, containing all source files (.java, .doc or .pdf or .txt, etc.) for your project using the following naming convention:

  The zip file should be called *P_studentID*, where # is the number of the project *studentID* is your student ID number. For example, for the project, student 123456 would submit a zip file named *P_123456.zip*.

- Submit your zip file at: https://fis.encs.concordia.ca/eas/ as **Project** and submission. The project submitted to the wrong directory would be discarded and no replacement submission will be allowed.
- Submit only **ONE version** of a project. If more than one version is submitted the last one, before the deadline date, will be graded and all others will be disregarded.

Evaluation Criteria of the project (100 points)

Activities	Points
Design Documentation: - Describe and explain your design and architecture clearly, including theories (protocol, algorithm) you apply, how to implement. **[20 pts.]** - Design proper and sufficient test scenarios, which should include testing data and results. **[10 pts.]**	30 pts.
Q2: Demo on Zoom platform: - Introduce your application architecture: **[20 pts.]** - Demo your designed testing scenarios to illustrate the correctness of your design. If your testing scenarios do not cover all possible issues, you'll lose part of mark up to 40%. **[50 pts.**]	70 pts.