CONCORDIA UNIVERSITY

DEPARTMENT OF
COMPUTER SCIENCE AND SOFTWARE ENGINEERING

COMP 6231, Fall 2013 Instructor: R. Jayakumar

PROJECT

Issued: Oct. 31, 2013 Due: Dec. 2, 2013

Failure Tolerant/Highly Available Distributed Police Information System

In this project, you are going to enhance your CORBA implementation of the Distributed Police Information System (DPIS) developed in Assignment 2 to be software failure tolerant or highly available using process replication. There are two projects, suitable for teams of 3 or 4 students, as described in the following.

Project 1 (for teams of 3)

Software Failure Tolerant CORBA Distributed Police Information System

For this project, extend your Distributed Police Information System implementation from Assignment 2 to tolerate either a single software (non-malicious Byzantine) failure or a single replica crash at a time using active replication. Your actively replicated DPIS server system should have at least three replicas each running a different implementation (in different hosts on the network). The front end (FE) receives a CORBA request from a client, atomically broadcasts the request to all the replicas, receives the replies from the replicas, and sends a single correct result to the client. The FE 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. If the FE does not receive a reply from a replica within a reasonable time, it assumes that replica has crashed and informs the replica manager which replaces the crashed replica after confirming that crash failure. The replicas in this server subsystem execute the client requests according to a total order selected using a sequencer reducing the number of messages. 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 in order to avoid message loss. Specifically do the following.

- (*Team*) Assuming that server failure could be due to software bugs (i.e. non-malicious Byzantine failures) or process crashes, design your actively replicated, software failure-tolerant server system.
- (*Student 1*) Design and implement the front end (FE) which receives requests from the clients as CORBA invocations, atomically broadcasts the requests to the server replicas, and sends a single correct result back to the client by properly combining the results received from the replicas. The FE should also inform the replica manager (RM) if any replica produces incorrect result. If the FE does not receive a reply from a replica within a reasonable time, it assumes that replica has crashed and also informs the replica manager about this potential crash.

- (*Student 2*) 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 (which the FE uses for request forwarding) and replaces a failed replica that produces incorrect result three times successively with another good one. If the RM receives information from a FE about a potentially crashed replica, it confirms that replica has indeed crashed and then replaces the crashed replica with a good one.
- (*Student 3*) Design and implement a request ordering subsystem which totally orders the client requests received by the replica using a modified sequencer reducing the number of messages.
- (*Team*) Modify the individual implementations of the server replica from Assignment 2, 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 process failure by returning a random result.

Project 2 (for teams of 4)

Highly Available CORBA Distributed Police Information System

In this project, you are going to implement a highly available CORBA Distributed Police Information System which tolerates process crashes only (no software bugs) using unreliable failure detection. Thus, there is a group of (at least 3) server processes (typically running on different hosts) providing the redundancy for high availability and periodically checking each other for failure detection. One of the processes in the group is the elected leader and receives requests from clients through a CORBA front end and sends responses back to them through the front end. The leader of the server group broadcasts the client request atomically to all the servers in the group using a reliable FIFO broadcast mechanism, receives the responses from them and sends a single response back to the client as soon as possible. Since the replicated servers are usually on a local area network, they communicate using the unreliable UDP protocol. However, the communication among them should be reliable and FIFO. Specifically do the following.

- (*Team*) Assuming that processor failures are benign (i.e. crash failures) and not Byzantine, design your highly available active replication scheme using process group replication and reliable group communication.
- (*Student 1*) Design and implement the group leader process which receives a request from the front end, FIFO broadcasts the request to all the server replicas in the group using UDP datagrams, receives the responses from the server replicas and sends a single response back to the front end as soon as possible.
- (*Student 2*) Design and implement a reliable FIFO broadcast subsystem over the unreliable UDP layer.
- (*Student 3*) Design and implement a failure detection subsystem in which the processes in the group periodically check each other and remove a failed process from the group. If the group leader has failed, a new leader is elected using a distributed election subsystem.

- (*Student 4*) Design and implement a distributed leader election subsystem (based on the bully algorithm) which will be called when the current leader has crashed to elect a new leader for the process group.
- (*Team*) Modify the individual implementations of the server replica from Assignment 2, 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 process crash by killing that process while the application is running.

Marking Scheme

Before implementation, you should submit your design documentation by November 10, 2013, and get the TA's approval. This is a DESIGN project; if you implement a bad design, even though it may work, you'll still lose marks up to 50%.

[30%] Design Documentation (by group). ***Due by Sunday, November 10, 2013.***

- [20%] Describe and explain your design and architecture clearly, including theories (protocol, algorithm) you apply, how to implement, and also describe dataflow (how your modules interact/cooperate with each other to achieve the function).
- [10%] Design proper and sufficient test scenarios, which should include testing data and results.
- Please indicate the student ID of each student and the module that student is responsible for.
- Print the documentation and bring it to your demo.

[70%] Demo in the lab. ***On Thursday, December 5, 2013.***

- Please come to the lab session and choose your preferred demo time (20 minutes per group) in advance. There is no instant registration during the demo week, so if you cannot register before demo week, you will lose 40% of the mark.
- Make sure your application can work in the LAB during demo; otherwise you'll lose 70%. There will be no second chance.
- [50%] For group demo:
  - Introduce your application architecture.
  - 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%.
- [20%] For individual demo:
  - Introduce the module you are responsible for and answer questions on your module.

Questions

If you are having difficulties understanding sections of this project, feel free to email your Teaching Assistant at alexandre.hudon@sympatico.ca. It is strongly recommended that you attend the tutorial sessions as various aspects of the assignment will be covered.