Assignment 2: Reconfigurable Paxos

Distributed Systems Class

May 11, 2012

1 Specification

In the next stage of the single person project, you will support reconfigurability within Paxos. This would allow for a Paxos group to drop failed nodes and add new nodes in order to replace the failed ones. As a consequence, the resulting Paxos group would have a greater degree of availability than the version with a fixed static set of nodes. You do not have to provide any additional functionality from the client facing API; instead, your implementation would just be improving the reliability of the underlying storage.

As with any distributed systems project, your first step would be to design the various components that is necessary to add reconfiguration to your Paxos implementation. Feel free to follow the design suggestions in the "Paxos made simple" paper by Lamport or the "Paxos made practical" paper by Mazieres. You would likely need to add the following components or address related issues:

- Failure detector: develop a failure detector that can recognize when a node fails.
- Reconfigure by removing failed nodes: change the configuration of the Paxos group by coming to a consensus on the membership set for the next epoch. Consider what are the implications of having inconsistent failure detectors – i.e., what happens when A thinks B has failed, B thinks A has failed, and C thinks that both A and B are alive.
- Reconfigure by adding a new node to the system: consider a node that wants to join the group. This node can submit a request to the existing Paxos group, say with its location information, indicating that it wants to join the group.
- State transfer: upon adding a new node to the system, how is the state transferred from previous set of nodes to the new node? It is ok if your implementation is blocking during this transfer, i.e., it does not have to accept any client requests while the state transfer takes place.

As with the previous assignment, you can organize your code in whatever way that you think is appropriate. While we have provided you with a Java framework, you are welcome to use the language of choice that you are most comfortable with.

2 Deliverables

2.1 Code

Students will need to provide the source code that implements the above requirements. The source code should be well-documented in order to ease grading and prevent misunderstandings about functionality. Compile/execution scripts must be included as well, and you can assume that they will run on support managed machines in the lab from the directory just above the jars/ and proj/ directories. The groups will meet with the TA or the instructor to demonstrate the code and explain how it works.

2.2 Write-up

In addition to the source code, a major component of your grade will be based on a project write-up, in which your group should explain the protocol and any relevant implementation details. You should include, at a minimum, the following:

- a detailed description of your group's implementation
- any assumptions you made
- how to use your implementation
- any outstanding issues
- anything else that you feel is important to discuss (e.g. quirks, interesting behavior, performance characteristics, etc)