CISC 5597 Distributed Systems

Lab 2: Consensus and Consistency— Basic Paxos Algorithm

Background:

Paxos is an algorithm that enables a distributed computing system to achieve consensus over an asynchronous network. To achieve agreement, one or more of the computers proposes a value to Paxos. Consensus is achieved when a majority of the computers running Paxos agrees on one of the proposed values.

Lab Assignment:

Based on RPC examples and the network model that you created in Lab-1, we are going to simulate a simplified distributed file system.

1. There are **THREE** "computing nodes" in your cluster. They are fully connected through RPC calls, meaning that they know each other's IPs and corresponding ports.
2. The cluster maintains a file, **CISC5597**, that has replicas on each of the computing nodes.
3. It uses **Basic Paxos (e.g., no distinguished proposer)** to maintain consistency on this file. It is a two-phase algorithm that contains **proposers** and **acceptors (NO learners in the system).** In this cluster, we assume the computing nodes are acting as proposers and acceptors at the same time.

The cluster needs to simulate the following items.

1. Create a file on three nodes.
2. The client can submit ONE value to the system and it will be stored in the distributed file.
    a. For example, the client can call the RPC, SubmitValue, on one of the nodes.
    b. When the cluster ensures consistency within the system, it sends a success message back to the client.
3. Simulate the basic Paxos as discussed during lecture 8. Please check out the 7 steps on *Single Decree Paxos: Protocol* page in Slide-8. There are 3 scenarios to be simulated.
    a. A single proposer. In this case, everything is well-behaved.
    b. There are two proposers, e.g., two clients, A and B, who want to change the same value.
        i. Simulate a scenario that A wins, e.g., Slide-8 page 23, *Previous value already chosen*
        ii. Simulate a scenario that B wins, e.g., Slide-8 page 24 or 25, *Previous value not chosen, but new proposer sees it (24) OR Previous value not chosen, new proposer doesn't see it (25)*
        iii. In order to simulate two proposers, you will need to add a random delay to see the effect.
        iv. *Hint: You can use the sleep function to simulate network delay.*

Grading Rubric: *This lab allows a group of 2*.
Your code should be elegant and well-documented (with comments).

1. The submission can not run successfully. (0 points)
2. Create a file on three nodes. (10%)
3. The three computing nodes can communicate with each other through RPCs. (20%)
4. Single proposer and a success message to the client. (50%)
5. A wins. (10%)
6. B wins in the scenario on page 24. (10%)
7. Bonus-1: B wins in the scenario on page 25. (5%)
8. Bonus-2: Simulate a scenario of livelock as described on slide 8 page 26 and solve it through a randomized restart. (5%)