CISC 5597/6935 Distributed Systems

Lab 2: Consensus and Consistency— Raft Consensus Algorithm

Background:

Raft is a consensus algorithm used in distributed systems to ensure that data is replicated safely and consistently. It allows the members of a distributed system to agree on a sequence of values in the presence of failures.

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 log file, **CISC6935**, that has replicas on each of the computing nodes.
3. It uses **RAFT** to maintain consistency on this simulated log file. The nodes have three states: Follower, Candidate, and Leader.

The cluster needs to simulate the following items.

1. Create a file on three nodes.
2. The client can submit MULTIPLE values, one at a time, to the system and the values will be stored in the log file distributedly.
    a. For example, the client can call the RPC, SubmitValue, such as 1, on one of the nodes.
    b. The system needs to write one line into the file, such as *write 1.*
    c. When the system ensures consistency, it sends back a success message to the client.

3. Simulate the basic RAFT as discussed during lecture 8. There are 3 scenarios to be simulated.

    a. Leader election phase and everything goes well, NO crashes or slow followers.
    b. Simulate leader change in a *perfect situation*, which means NO left entries partially replicated (slide-8 page 54). It's basically the same as item a with another round of leader election, but *picking the best candidate*. This is a simplified selection due to the perfect situation.
        i. The leader change process can start with the client, e.g., by typing a command
    c. Simulate a scenario that the leader crashed (e.g., log file deleted) with an inconsistency log (e.g., *missing only ONE log entry*) on the other two followers. Check out slide-8 page 62.
        i. Again, it can be started by the client's command.

d. **Optional BONUS**: Simulate the crashed node rejoins the cluster. In this case, the leader needs to synchronize with the latest log files. The situation is similar to item c (slide-8 page 62), but with multiple rounds.

e. *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 log file on three nodes. (10%)
3. The three computing nodes can communicate with each other through RPCs. (20%)
4. Leader election phase and everything goes well. (50%)
5. Item b. (10%)
6. Item c. (10%)
7. Bonus. (5%)