Assignment 3: Quorum

Description

We will build upon the system designed in Assignment 2.

In this assignment, we will assume any replica can commit updates. A client can send an **add_update(key,value)** request to any replica and the replica will update its local data store only when it achieves a quorum. We define a quorum as the majority of members within a set of nodes. For a size of N nodes, a quorum requires votes from at least Nw members, where Nw >= (N/2)+1.

A leader is responsible for achieving a quorum on any add_update request to the state. You will implement the Bully algorithm to elect a leader. Each replica will be assigned a unique id on startup and leader election will be held (i) on startup (ii) or, when the leader becomes unresponsive. Note that the replicas will forward the add_update request to the leader, which will achieve quorum.

Once the leader replica achieves a quorum (e.g., majority vote), it will COMMIT the changes from the add_update request to its local data store. And, send COMMIT changes to other replicas. On receiving the COMMIT message for that entry, all replicas commit the updates to their local data store.

Similarly, any **read(key)** will require a quorum before the value is sent to the client. A client can connect to any replica to read the value. The replica will randomly select NR replicas and request for values. If there is no disagreement in value among the replicas, it will send the value to the client. Else it will randomly select N_R replicas and repeat until replicas reach an agreement. You should select N_R such that $N_R + N_W > N$.

Extra credit

It is likely that the leader (or replicas) may crash, and the state is left inconsistent. We will also implement a *restore protocol* where the replicas will roll back to the known consistent state that replicas agree upon with the leader. To do so, a replica will compare its state with the leader and duplicate the leader's state by copying the difference. Note that the replicas need to use vector clocks and should persist the state in order to send the difference.

Implementation

N_R and N_w are parameters in your server implementation.

Evaluation

1. Design a test to show that your system only commits changes when quorum is achieved.
2. Design a test to show read works as intended.
3. (Extra credit) Design a test to show failure cases are handled and replicas are restored to the leader's state.

Please put appropriate print statements.

Submission

Please upload your code and your report on Gradescope. Your report should discuss your implementation and design choices (pros and cons). Your submission should also contain all the code including the test cases, dockerfile and log files of your execution.

Grading Criteria

Component	**%**
Implementation	60
Testing	30
Code documentation	10
Extra credit	10