Project 4 Executive Summary

Assignment Overview

This assignment required extending our Project 3 key-value store by replacing the non-fault-tolerant two-phase commit protocol with the Paxos consensus algorithm. The goal was to create a replicated key-value store that continues to function correctly even when some server nodes fail. We needed to implement the core Paxos roles (proposers, acceptors, and learners) and configure the acceptors to randomly fail and restart, demonstrating how the system maintains consistency despite these failures. The project challenged us to understand and implement complex distributed systems concepts in a practical setting, showing how theoretical consensus algorithms translate into robust, fault-tolerant applications.

Technical Impression

When I first approached this project, I severely underestimated the complexity of Paxos. I spent almost a week just trying to understand the algorithm, reading Lamport's "Paxos Made Simple" paper multiple times and watching various YouTube explanations. What seemed "simple" in the title was anything but! The breakthrough came when I started drawing diagrams of the message flows and state transitions for each role.

My first implementation attempt failed miserably. I initially tried to create separate classes for each Paxos role, but quickly ran into synchronization nightmares. After hitting a wall, I realized a more integrated approach was needed, with each server instance performing all three Paxos roles. Refactoring the code took another three days.

The random failure simulation was particularly tricky. I implemented a separate thread that would periodically "kill" the acceptor functionality, but initially, this caused deadlocks. I discovered that I hadn't properly handled the case where a server tried to contact a failed acceptor. Adding timeouts and retries solved this issue, but exposed another problem: servers occasionally formed "dueling proposers" scenarios, where they kept proposing conflicting values. Adding exponential backoff to my retry logic finally addressed this.

Testing revealed more subtle issues. Sometimes, even with a majority of servers running, operations would fail. The bug turned out to be in how I was counting responses - I was waiting for exactly the majority number instead of at least the majority. This small logical error caused sporadic failures that were hard to diagnose.

The most satisfying moment was watching the system recover seamlessly after multiple nodes failed. Seeing the Paxos algorithm actually maintain consistency across the distributed system, despite all the chaos I was throwing at it, gave me a profound appreciation for the elegance of the algorithm.