https://github.com/cong-n-tran/CSE5306-PA3

Course: CSE5306 - Distributed Systems Assignment: Programming Assignment 3

Team Members

1. Cong Tran (ID: 1002046419)
2. Mohamed Mohamed (ID: 1001375427)

Work Distribution

Cong Tran

- Two-Phase Commit (2PC):
  - Implemented the Coordinator logic in the Trip Service (Q1).
  - Implemented the Participant logic in the Location Service (Q2), ensuring atomic updates for ride completion and driver availability.
  - Defined the gRPC Protobuf definitions for the voting and commit phases.
- Testing:
  - Developed and executed the unit testing suites for both the 2PC implementation (verifying commit/abort scenarios) and the Raft implementation (verifying leader election and partition tolerance).

Mohamed Mohamed

- Raft Implementation:
  - Implemented the core Raft Consensus algorithm (Q3 & Q4).
  - Developed the Leader Election mechanism, including randomized timeouts and state transitions (Follower -> Candidate -> Leader).
  - Implemented the Log Replication logic to ensure data consistency across the distributed nodes.
  - Handled the logic for managing network partitions and ensuring quorum safety.

Summary of Approach

Two-Phase Commit (2PC) We chose a decentralized approach where the Trip Service acts as the Coordinator because it owns the lifecycle of the ride. We used Redis to maintain state during the transaction. A key challenge was ensuring that if the coordinator crashed after voting but before the global decision, participants wouldn't remain locked indefinitely; we addressed this by implementing simple timeouts on the participant side.

Raft Consensus Our Raft implementation focuses on correctness over performance. We implemented the core state machine strictly following the Raft paper.

- Election Safety: We verified that split votes are handled by randomized election timeouts.
- Log Replication: We implemented basic AppendEntries consistency checks. If a follower's log doesn't match the leader's prevLogIndex, the leader decrements the index and retries.
- Split Brain: We verified via our testing suite that when the network is partitioned, the minority partition pauses operations while the majority partition elects a new leader, ensuring the system remains available and consistent.