IT-559 Distributed Systems Project

Group 7

Suyash Bhagat (202101085),
Jeel Viradiya (202101164),
Nishith Parekh (202101449)

Distributed Consensus

April 20, 2024

1 Problem Statement

Distributed systems often require multiple processes to agree on a single value. This problem, known as Distributed Consensus, is crucial for ensuring data consistency and system-wide coordination. PAXOS is a well-known consensus algorithm designed to address this challenge by allowing a group of nodes to agree on a single value.

This project aims to implement the PAXOS distributed consensus algorithm within a simplified environment consisting of one server and multiple clients. The primary objective is to achieve consensus among the clients regarding a proposed value.

We will focus on achieving the following objectives:

- **Implement PAXOS Algorithm:** Develop the necessary components to execute the PAXOS algorithm, including proposal generation, promise and acceptance handling, and the final consensus phase.
- **Handle Network Communication:** Establish communication channels between the server and clients using TCP sockets to exchange messages required for the PAXOS protocol.
- **Demonstrate consensus:** Showcase how the PAXOS protocol guarantees agreement on a single value, ensuring consistency within the distributed system.

Through this project, we will better understand distributed consensus and gain practical experience with the PAXOS algorithm. The successful implementation will provide a valuable tool for understanding and evaluating this fundamental concept in distributed computing.

2 Software Architecture

The distributed system will be built using several interacting modules to achieve consensus through the PAXOS algorithm. Below is a breakdown of the key modules and their data flows:

Note: This is a simplified overview. The actual implementation may involve additional messages and logic for handling failures and timeouts.

2.1 Modules

1. **Proposer**
   - Inputs: Client request with a proposed value.
   - Outputs: PAXOS messages (Prepare and Accept) containing the proposed value. Broadcasted to all Acceptors.
2. **Acceptor**
   - Inputs: PAXOS messages (Prepare and Accept) from Proposers.
   - Outputs: PAXOS messages (Promise and Accepted) sent back to Proposers.
3. **Learner**
   - Inputs: PAXOS messages (Accepted) containing the chosen value from any Proposer.
   - Outputs: Agreed-upon value reported to the system or application.
4. **Communication Channel**
   - Inputs: PAXOS messages from any module (Proposer, Acceptor, Learner).
   - Outputs: Delivers PAXOS messages to the intended recipient module.

2.2 Data Flow

1. The **Client** submits a request with a proposed value to the **Proposer**.
2. The **Proposer** initiates a round of consensus by sending a **Prepare** message containing a unique proposal number and the proposed value to all **Acceptors**.
3. Each **Acceptor** responds with a **Promise** message indicating it hasn't accepted a higher-numbered proposal yet. The Promise may also include the highest accepted proposal number and its corresponding value (if any).
4. Upon receiving a majority of Promises, the **Proposer** sends an **Accept** message containing the proposal number, value, and any learned highest accepted values to all **Acceptors**.
5. **Acceptors** that haven't responded with a conflicting Promise accept the proposal and send an **Accepted** message back to the **Proposer** and potentially all **Learners**.
6. Once the **Proposer** receives a majority of **Accepted** messages for its proposal, it knows consensus has been reached.
7. **Learners** collect **Accepted** messages from any Proposer. Once a Learner receives a majority of **Accepted** messages for the same proposal number and value, it considers that value to be the agreed-upon outcome and reports it to the system or application.

3 Screenshots

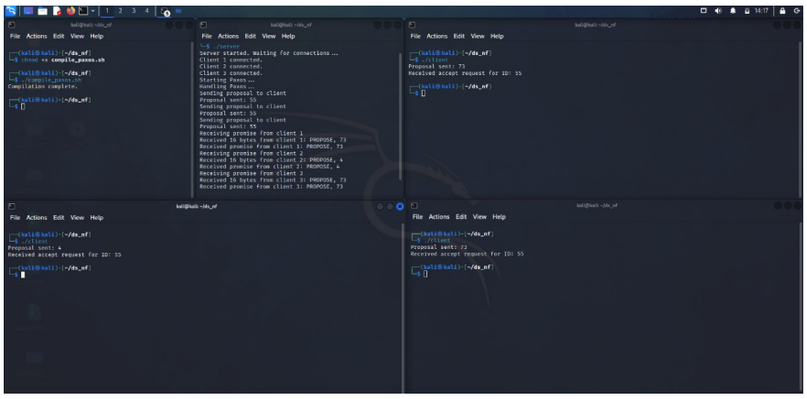

Figure 1: Input and Output screenshot

4 Conclusion

This project focused on implementing a simplified distributed system using the PAXOS consensus algorithm. While traditionally envisioned for fully distributed systems with multiple servers, we adapted the core principles for a scenario with a single server and three clients.

We successfully demonstrated how the server, acting as the sole Proposer and Acceptor, can achieve consensus with the three clients. The clients submit proposed values, and the server facilitates the PAXOS protocol to ensure all clients eventually agree on a single chosen value.

The core PAXOS components were implemented:

- **Server (Combined Proposer and Acceptor):** This module initiates consensus rounds, proposes values from clients, and acts as the sole acceptor, maintaining consistency.
- **Clients:** These modules submit proposed values to the server and wait for the agreed-upon value to be communicated back.

The communication channel facilitates message exchange between the server and clients.

This implementation serves as a valuable proof-of-concept for understanding PAXOS and its potential for achieving consensus in various system designs, even with a single server coordinating with multiple clients.