IT-559 Distributed Systems

Group-6 Project Report

Topic: CONSENSUS ALGORITHM

- **Problem Statement:**
  - a) **System Description:** The implemented system is a software solution designed to execute the basic Paxos algorithm. The system facilitates agreement on a single value among a group of distributed processes, ensuring consistency and fault tolerance.

  - b) **Paper Implementation:**
    i. **Leslie Lamport**, Paxos Made Simple, ACM SIGACT News, December 2001
    ii. Understanding Paxos and other distributed consensus algorithms, **Victor Yodaiken**

- **Core concepts from the Paper that we implemented :**

1. Introduction:
   - Lamport introduces the Paxos algorithm, highlighting its significance in achieving fault-tolerant distributed consensus. Despite its perceived complexity, he argues that it is fundamentally simple.

2. The Consensus Algorithm:
   - **The Problem**: Defines the requirements for consensus, emphasizing the need for safety (ensuring only proposed values are chosen) and liveness (ensuring eventual selection of a value).
   - **Roles and Agents**: Describes the roles of proposers, acceptors, and learners. Proposers initiate proposals, acceptors accept proposals, and learners discover chosen values.
   - **Asynchronous Model**: Outlines the communication model, where agents operate asynchronously and may fail or restart.

3. Choosing a Value:
   - Phase 1: Proposers select a proposal number ‘ n ’ and send prepare requests to a majority of acceptors. Acceptors respond with promises never to accept proposals numbered less than ‘n ’ and the highest-numbered proposal they've accepted.
   - This phase establishes a proposal number and ensures that acceptors are ready to receive new proposals without compromising safety.
   - Phase2: If a proposer receives responses from a majority of acceptors in Phase 1, it sends accept requests to those acceptors with a proposal numbered ‘n ’ and

a value ’v ’, where ’v ’ is the value of the highest-numbered proposal among the responses.
- This phase finalizes the proposal by ensuring its acceptance by a majority of acceptors, thus guaranteeing that the proposal is chosen.

4. Learning a Chosen Value:
- Learners discover chosen values by receiving responses from acceptors or distinguished learners. This ensures that all learners eventually converge on the chosen value.
- This phase emphasizes fault tolerance and reliability in disseminating chosen values to all learners, even in the presence of failures or message loss.

5. Progress:
- Discusses scenarios where progress might be hindered by competing proposers and proposes solutions to ensure eventual progress, such as electing a distinguished proposer.
- Ensuring progress is crucial for the algorithm's effectiveness, and mechanisms like leader election mitigate potential bottlenecks.

6. The Implementation:
- Describes how the Paxos algorithm is implemented in a network of processes, with each process playing multiple roles.
- Implementing Paxos requires attention to detail, especially in maintaining state consistency and ensuring fault tolerance through mechanisms like stable storage.

- **Software Architecture:**

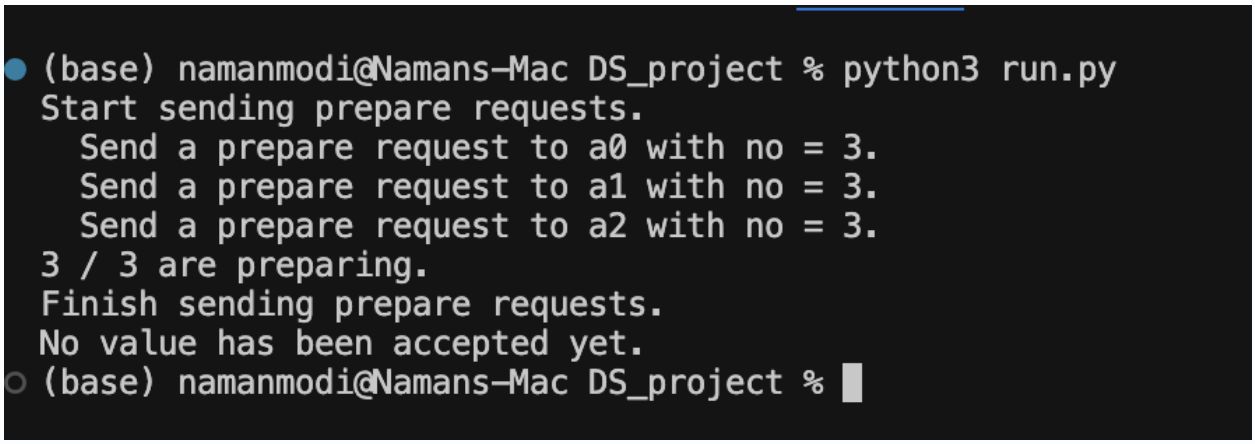
Modules:
- **Proposer**: Initiates proposals by sending prepare requests and accept requests to acceptors.
- **Acceptor**: Receives prepare requests and accept requests from proposers, decides whether to accept or reject proposals, and informs learners about accepted values.
- **Learner**: Receives accepted values from acceptors and determines the chosen value based on the majority acceptance.

Inputs:
- Configuration parameters: Number of acceptors, initial proposal number, initial proposal value.
- Network messages: Prepare requests, accept requests, responses from acceptors.

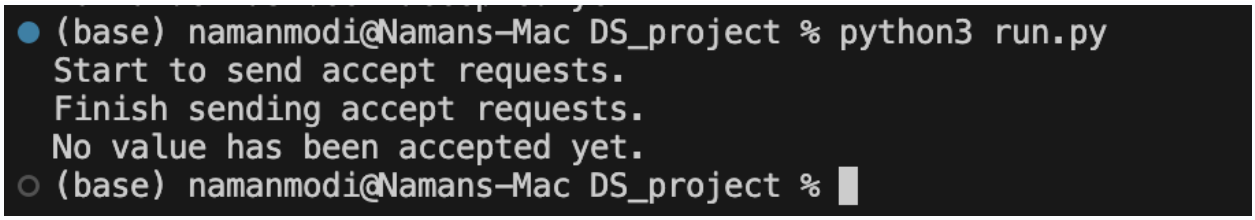
Outputs:
- Console outputs indicating the progress of prepare requests and accept requests. Chosen value determined by learners.

- **Code:(we have Hard coded paxos algorithm)**

Phase 1 simulation:

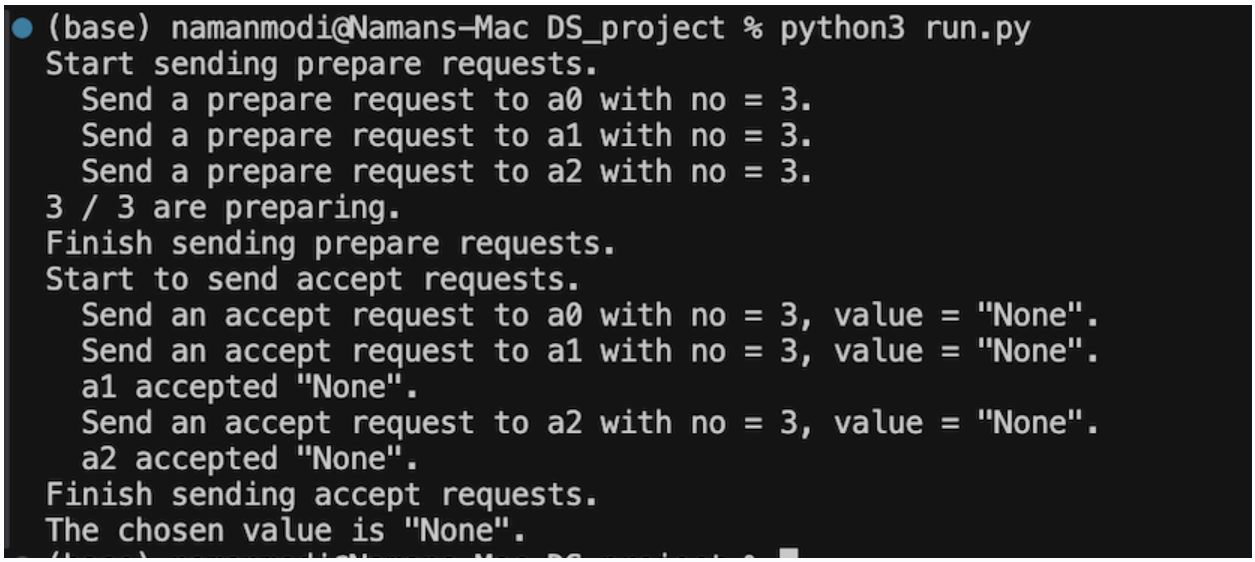


Phase 2 simulation:



Failure Simulation:

Here, a0 fails to respond in the simulation.



Multiple Proposers:

There are 3 proposers in the simulation with different proposal value and number.

```
(base) namanmodi@Namans-Mac DS_project % python3 run.py
Start sending prepare requests for proposer 1
  Send a prepare request to a0 with no = 1.
  Send a prepare request to a1 with no = 1.
  Send a prepare request to a2 with no = 1.
3 / 3 are preparing.
Finish sending prepare requests for proposer 1
Start to send accept requests for proposer 1
  Send an accept request to a0 with no = 1, value = "A".
  a0 accepted "A".
  Send an accept request to a1 with no = 1, value = "A".
  a1 accepted "A".
  Send an accept request to a2 with no = 1, value = "A".
  a2 accepted "A".
Finish sending accept requests for proposer 1
Start sending prepare requests for proposer 1
  Send a prepare request to a0 with no = 1.
  Send a prepare request to a1 with no = 1.
  Send a prepare request to a2 with no = 1.
3 / 3 are preparing.
Finish sending prepare requests for proposer 1
Start to send accept requests for proposer 1
  Send an accept request to a0 with no = 1, value = "B".
  a0 accepted "B".
  Send an accept request to a1 with no = 1, value = "B".
  a1 accepted "B".
  Send an accept request to a2 with no = 1, value = "B".
  a2 accepted "B".
Finish sending accept requests for proposer 1
Start sending prepare requests for proposer 2
  Send a prepare request to a0 with no = 2.
  Send a prepare request to a1 with no = 2.
  Send a prepare request to a2 with no = 2.
3 / 3 are preparing.
Finish sending prepare requests for proposer 2
Start to send accept requests for proposer 2
  Send an accept request to a0 with no = 2, value = "B".
  a0 accepted "B".
  Send an accept request to a1 with no = 2, value = "B".
  a1 accepted "B".
  Send an accept request to a2 with no = 2, value = "B".
  a2 accepted "B".
Finish sending accept requests for proposer 2
Start sending prepare requests for proposer 2
  Send a prepare request to a0 with no = 2.
  Send a prepare request to a1 with no = 2.
  Send a prepare request to a2 with no = 2.
3 / 3 are preparing.
Finish sending prepare requests for proposer 2
Start to send accept requests for proposer 2
  Send an accept request to a0 with no = 2, value = "B".
  a0 accepted "B".
  Send an accept request to a1 with no = 2, value = "B".
  a1 accepted "B".
  Send an accept request to a2 with no = 2, value = "B".
  a2 accepted "B".
Finish sending accept requests for proposer 2
Start sending prepare requests for proposer 3
  Send a prepare request to a0 with no = 3.
  Send a prepare request to a1 with no = 3.
  Send a prepare request to a2 with no = 3.
3 / 3 are preparing.
Finish sending prepare requests for proposer 3
Start to send accept requests for proposer 3
  Send an accept request to a0 with no = 3, value = "B".
  a0 accepted "B".
  Send an accept request to a1 with no = 3, value = "B".
  a1 accepted "B".
  Send an accept request to a2 with no = 3, value = "B".
  a2 accepted "B".
Finish sending accept requests for proposer 3
Start sending prepare requests for proposer 3
  Send a prepare request to a0 with no = 3.
  Send a prepare request to a1 with no = 3.
  Send a prepare request to a2 with no = 3.
3 / 3 are preparing.
Finish sending prepare requests for proposer 3
Start to send accept requests for proposer 3
  Send an accept request to a0 with no = 3, value = "B".
  a0 accepted "B".
  Send an accept request to a1 with no = 3, value = "B".
  a1 accepted "B".
  Send an accept request to a2 with no = 3, value = "B".
  a2 accepted "B".
Finish sending accept requests for proposer 3
The chosen value in the latest round is "B".
```

- **Results:**
    - The investigation into the Paxos algorithm provides a clear understanding of its functionality and applicability within fault-tolerant distributed systems. Paxos, often perceived as complex, is revealed to be one of the simplest and most fundamental distributed algorithms. The study unfolds the intricacies of Paxos, elucidating its consensus algorithm, which ensures the selection of a single value among proposed options. Safety requirements, including the necessity for only proposed values to be chosen and the uniqueness of the chosen value, are meticulously addressed.

- **Conclusion:**
    - In conclusion, this paper has presented a comprehensive analysis of the Paxos algorithm, shedding light on its underlying principles and operational mechanisms. By addressing the consensus problem inherent in distributed systems, Paxos offers a robust solution for achieving agreement among processes despite asynchronous communication and potential failures. Through the delineation of its phases and safety properties, Paxos emerges as a foundational tool for building reliable distributed systems. Moving forward, further exploration and implementation of Paxos promise to enhance the resilience and efficiency of distributed computing architectures.

References:

- **Leslie Lamport**, Paxos Made Simple, ACM SIGACT News, December 2001
- Understanding Paxos and other distributed consensus algorithms, **Victor Yodaiken**
- Multiple blogs from **Medium** for understanding

Team Members:

- Naman Modi - 202101421
- Manan Pareek - 202101018
- Rishi Arora - 202101025
- Raj Lutya - 202101032