Large Scale Systems
CS 410 / 510

Lecture 9:

Crash Failures

Suyash Gupta

Assistant Professor

Distopia Labs and ONRG

Dept. of Computer Science

(E) suyash@uoregon.edu

(W) gupta-suyash.github.io

UNIVERSITY OF OREGON

Assignment 2 is Out!

- **Assignment 2 is out!**
- Please work with your groups to understand the underlying system.
- Assignment 2 report **deadline** → April 30, 2026 at 11:59pm.

Last Class

- Last class we looked at:

- How to evaluate a LSS

Reading Material

- For this class
  - Read **Chapter 8** from **Distributed System** by Maarten van Steen and Andrew Tanenbaum.
  - Authors provide students free online copy.

Failures

- What is meant by a failure in a large-scale system?

Failures

- What is meant by a failure in a large-scale system?

- A failure in a large-scale system implies that some component in the system is no longer working in the expected manner.

What can fail in a Large-Scale System?

What can fail in a Large-Scale System?

- In a replicated system:
  - One of more replicas can fail.

What can fail in a Large-Scale System?

- In a replicated system:
  - One of more replicas can fail.

- In a sharded system:
  - Replicas in a shard can fail.
  - Can full shard fail?

What can fail in a Large-Scale System?

- In a replicated system:
  - One of more replicas can fail.

- In a sharded system:
  - Replicas in a shard can fail.
  - Can full shard fail? Yes

- Can clients fail?

What can fail in a Large-Scale System?

- In a replicated system:
  - One of more replicas can fail.

- In a sharded system:
  - Replicas in a shard can fail.
  - Can full shard fail? Yes

- Can clients fail? Yes

- What about network?

What can fail in a Large-Scale System?

- In a replicated system:
  - One of more replicas can fail.

- In a sharded system:
  - Replicas in a shard can fail.
  - Can full shard fail? Yes

- Can clients fail? Yes

- What about network? Even network failures are possible.

Categorization of Failures

- How can you broadly categorize failures?

Categorization of Failures

- How can you broadly categorize failures?
- **Omission Failures**
- **Commission Failures**

Categorization of Failures

- How can you broadly categorize failures?

- **Omission Failures:**
  - Unexpected failures.
  - Failures due to something omitted, when it should have been done.

- **Commission Failures**
  - Caused Failures.
  - Failures due to doing some action that should not have been done.

Categorization of Failures

- **Crash Failure** → A node halts or fail stops.
- **Ommission Failure** → Failure to send, receive or respond to a message.
- **Timing Failure** → Message arrives beyond a specified time period.
- **Commission Failure** → Arbitrary failures like Byzantine attacks.

Detecting Failures

- Can a Node/Process/Partition **P** detect if another node **Q** has failed?

Detecting Failures

- Can a Node/Process/Partition **P** detect if another node **Q** has failed?
  - Unfortunately, **No**!
  - Possibly there is a lack of communication due to network failure.

Detecting Failures

- Can a Node/Process/Partition **P** detect if another node **Q** has failed?
  - Unfortunately, **No**!
  - Possibly there is a lack of communication due to network failure.

- What are the different types of network models?

Detecting Failures

- Can a Node/Process/Partition **P** detect if another node **Q** has failed?
  - Unfortunately, **No**!
  - Possibly there is a lack of communication due to network failure.

- What are the different types of network models?
  - **Asynchronous Systems**

  - **Synchronous Systems**

Detecting Failures

- Can a Node/Process/Partition **P** detect if another node **Q** has failed?
  - Unfortunately, **No**!
  - Possibly there is a lack of communication due to network failure.

- What are the different types of network models?
  - **Asynchronous Systems**
    - No timing assumptions made about process execution or message delivery.

  - **Synchronous Systems**
    - Process execution and message delivery are time bounded!

Asynchronous vs Synchronous System

- What are the key design issues with asynchronous and synchronous systems?

Asynchronous vs Synchronous System

- What are the key design issues with asynchronous and synchronous systems?
- A synchronous system is hard to practically **implement**.
- It is hard to **determine** the type of failure in an asynchronous system.
- In practice, no system is truly synchronous or asynchronous.

Partially Synchronous System

Partially Synchronous System

- Partially synchronous systems mostly behave as synchronous systems with some periods of asynchrony.

- There is no time bound, but the assumption is that the periods of asynchrony will resolve.

Partially Synchronous System

- Partially synchronous systems mostly behave as synchronous systems with some periods of asynchrony.

- There is no time bound, but the assumption is that the periods of asynchrony will resolve.

- Common method to detect failures is to use timers.
  - Incorrect failure detection possible → Network failure may be categorized as node failure.

Availability under Failures

- How can you guarantee availability under failures?

Availability under Failures

- How can you guarantee availability under failures?
  - Replication!

- How much replication or redundancy is needed to handle a failure?

Availability under Failures

- How can you guarantee availability under failures?
  - Replication!

- How much replication or redundancy is needed to handle a failure?
  - Depends on the type of failure and goal of the system.

  - **Crash failures** → For **f failures**, total replicas needed?
    - **Storage and Execution Systems** →

    - **Regular Systems** →

Availability under Failures

- How can you guarantee availability under failures?
  - Replication!

- How much replication or redundancy is needed to handle a failure?
  - Depends on the type of failure and goal of the system.

  - **Crash failures** → For **f failures**, total replicas needed?
    - **Storage and Execution Systems** →
      - If the goal is to just store and execute data, **f+1 replicas** sufficient.

    - **Regular Systems** →
      - Total **2f+1 replicas** needed.

Decision Making in Replicated Systems

Decision Making in Replicated Systems

- Once you have decided the replication factor, you need to enable agreement or consensus mechanism among the replicas.

- Consensus helps the system make progress under crash failures.

- **How to establish a consensus?**

Paxos

Paxos

- Published by Leslie Lamport in 1989.

- Considered as the first consensus protocol.

Paxos Preliminaries

- Partially synchronous setting (may also be asynchronous).
- Messages can get loss, delayed, duplicated or reordered.
- All participants behave deterministic → on same input, yield same output.
- Replicas can crash.

Paxos System Model

- Each replica plays three types of **roles**:

Paxos System Model

- Each replica plays three types of **roles**:
  - **Proposers**
    - Propose client transactions.

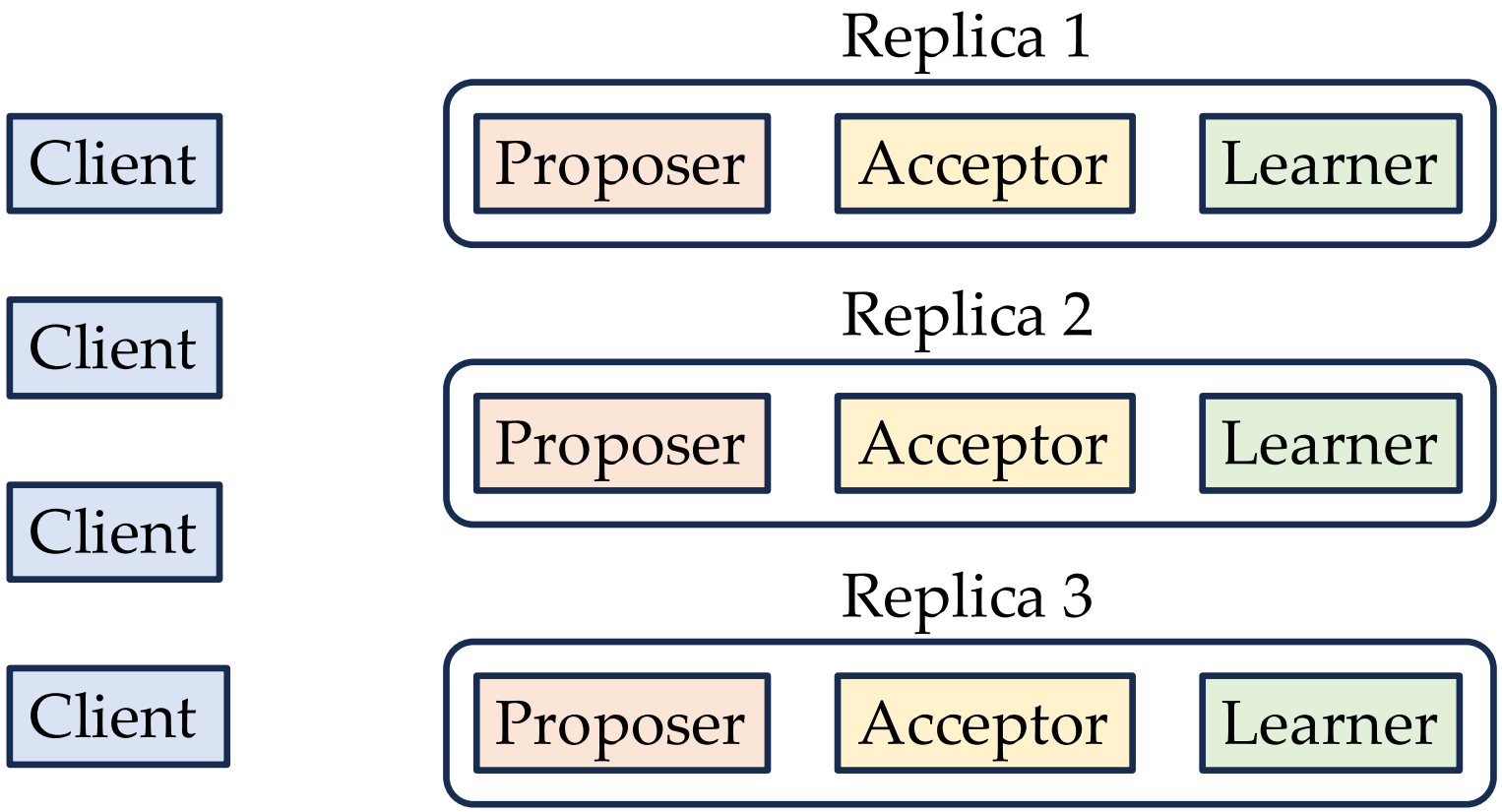
Paxos System Model

- Each replica plays three types of **roles**:
  - **Proposers**
    - Propose client transactions.
  - **Acceptors**
    - Decide whether to accept a proposal or not.

Paxos System Model

- Each replica plays three types of **roles**:
  - **Proposers**
    - Propose client transactions.
  - **Acceptors**
    - Decide whether to accept a proposal or not.
  - **Learners**
    - Learn accepted proposal and execute them.

Paxos System Model



Paxos Fault Tolerance

- To guard against up to **f** failures, Paxos needs at least **2f+1** replicas to reach consensus.

- A proposal becomes accepted if it is accepted by a majority of acceptors →**f+1**.

Paxos Protocol

Paxos Protocol

- **Step 1a :**

  A proposer selects a proposal number **n** and sends a prepare request with number **n** to a majority of acceptors.

Paxos Protocol

- **Step 1a :**

  A proposer selects a proposal number **n** and sends a prepare request with number **n** to a majority of acceptors.

- **Step 1b :**

  If an acceptor receives a prepare request with number **n** greater than that of any prepare request to which it has already responded, then it responds to the request with

  - a **promise** not to accept any more proposals numbered less than **n** and
  - with the highest-numbered proposal (if any) that it has accepted.

Paxos Protocol

- **Step 2a :**

  If the proposer receives a response to its prepare requests (numbered **n**) from a majority of acceptors, then it sends an accept request to each of those acceptors for a proposal numbered **n** with a value **v**,

  where **v** is the value of the highest-numbered proposal among the responses, or is any value if the responses reported no proposals.

Paxos Protocol

- **Step 2a :**

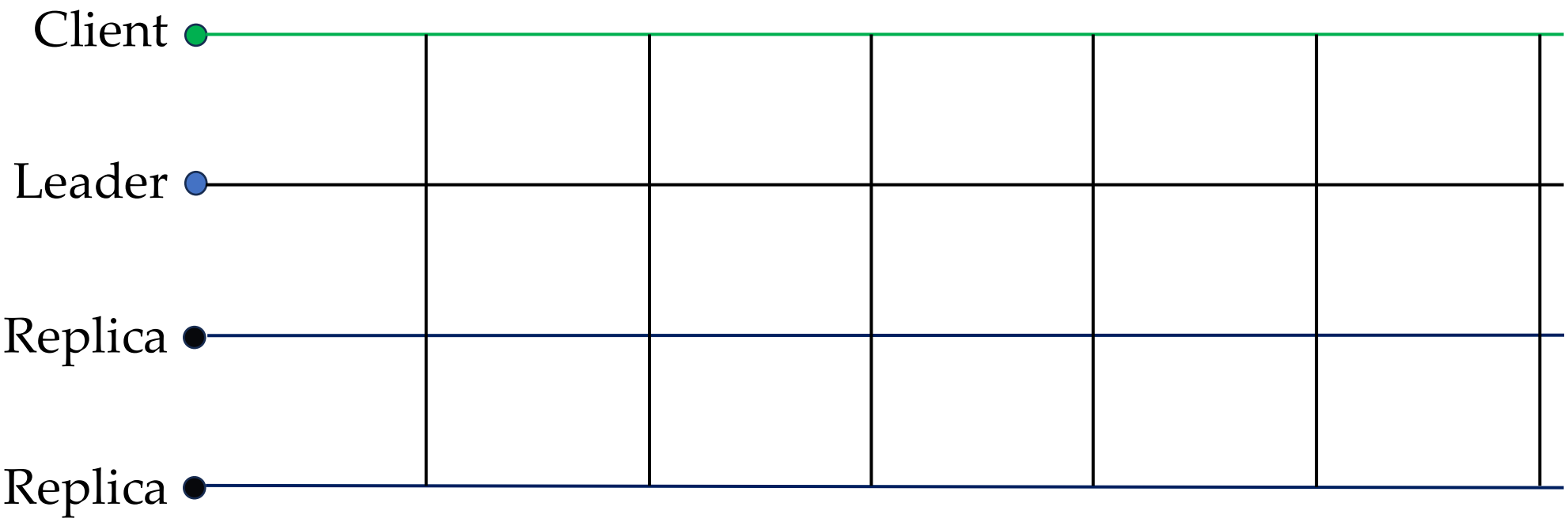
  If the proposer receives a response to its prepare requests (numbered **n**) from a majority of acceptors, then it sends an accept request to each of those acceptors for a proposal numbered **n** with a value **v**,

  where **v** is the value of the highest-numbered proposal among the responses, or is any value if the responses reported no proposals.

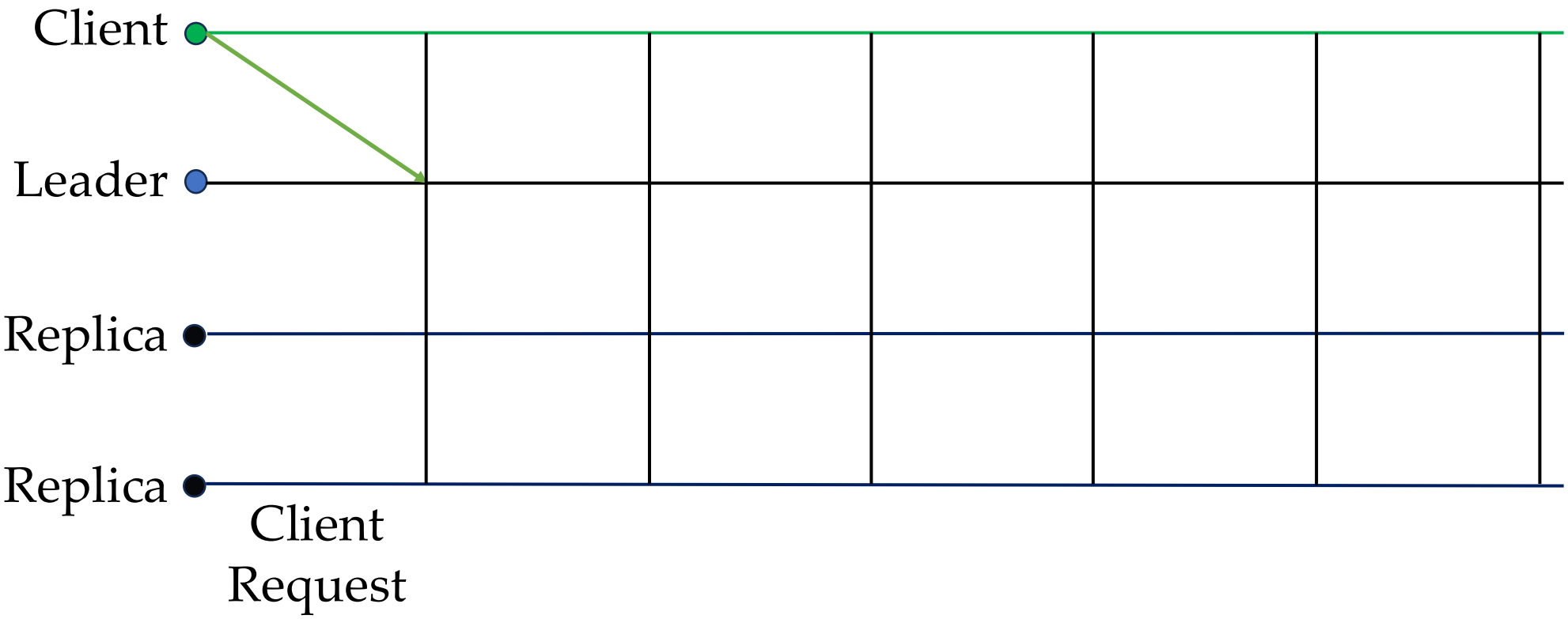
- **Step 2b :**

  If an acceptor receives an accept request for a proposal numbered **n**, it accepts the proposal unless it has already responded to a prepare request having a number greater than **n**.

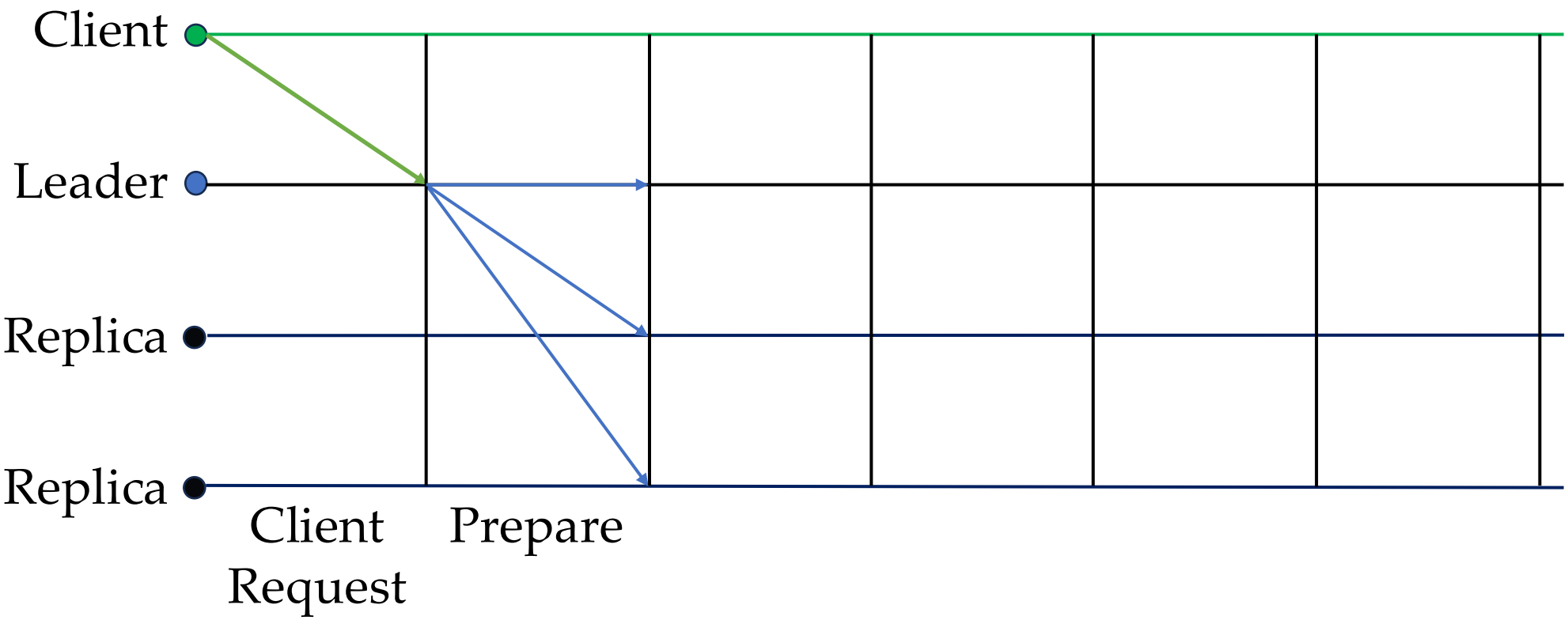
Paxos Protocol



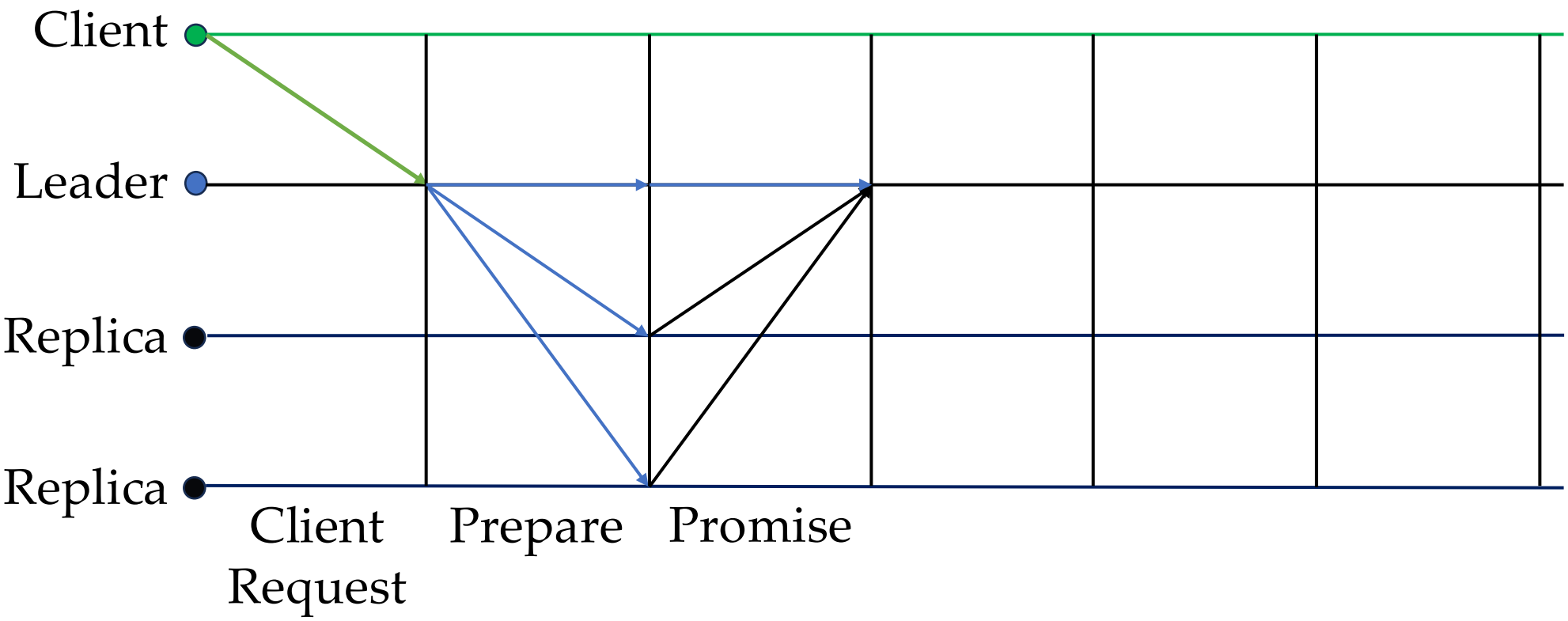
Paxos Protocol



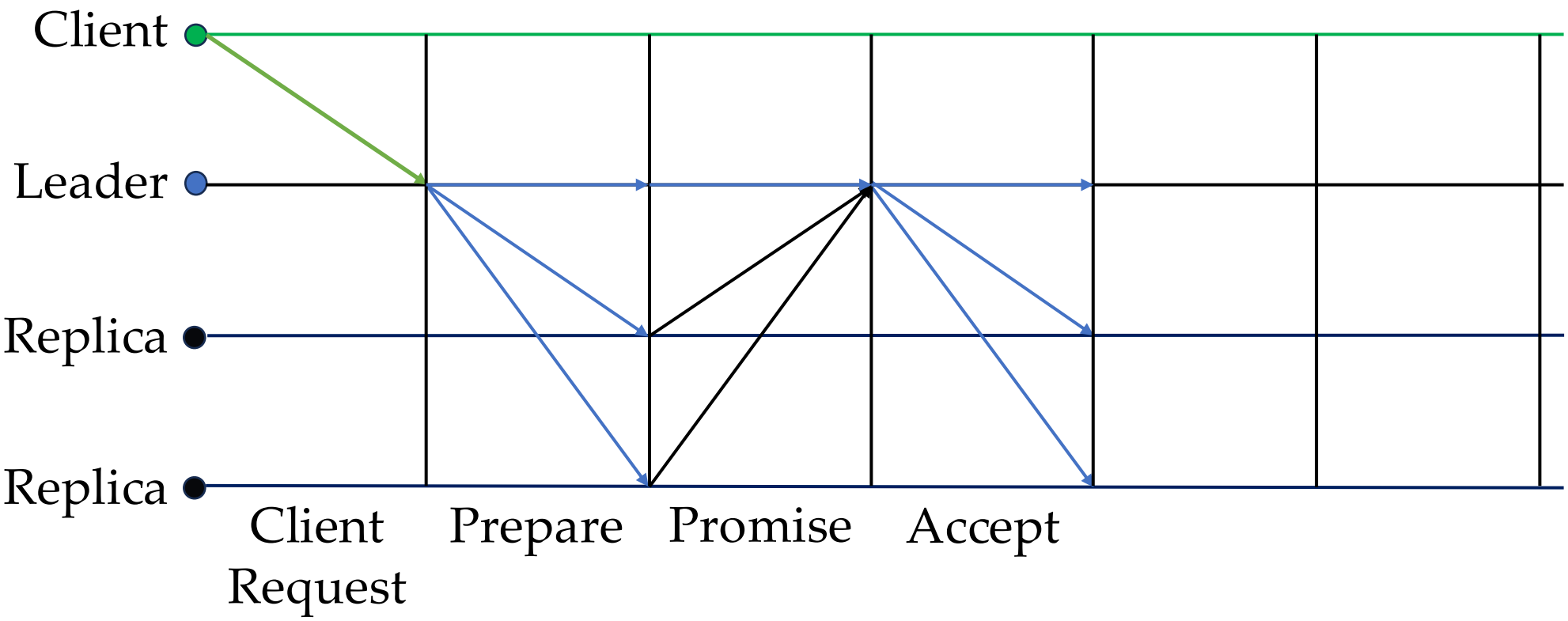
Paxos Protocol



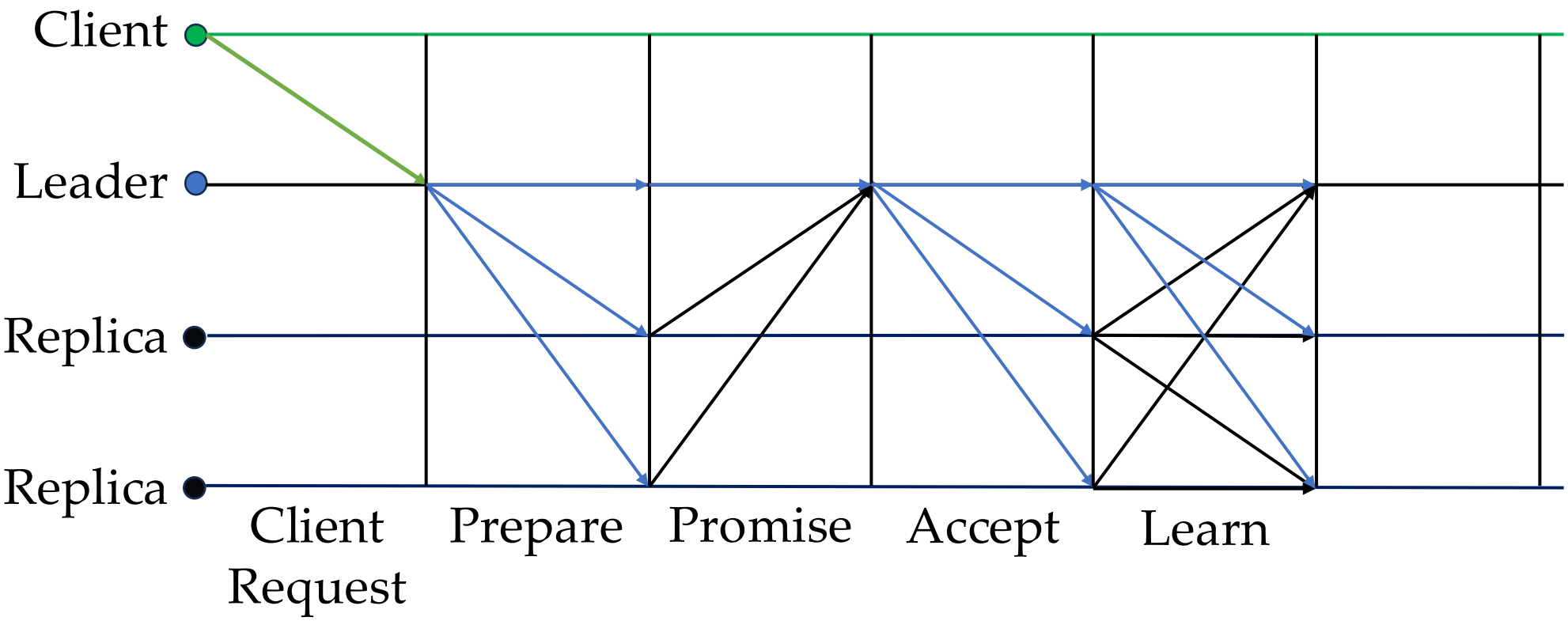
Paxos Protocol



Paxos Protocol



Paxos Protocol



Liveness Issue with Paxos

Liveness Issue with Paxos

- In the case of multiple competing proposers, Paxos can get stuck and not make any progress!

Liveness Issue with Paxos

- In the case of multiple competing proposers, Paxos can get stuck and not make any progress!

- Solution → Having a dedicated leader.