Large Scale Systems
CS 410 / 510

Lecture 13:

Speculation in Consensus

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 3 is Out!

- Assignment 3 is due on **May 18, 2026** at **11:59pm PST**.
- Please start working with your groups.

Last Class

- Last class we looked at:

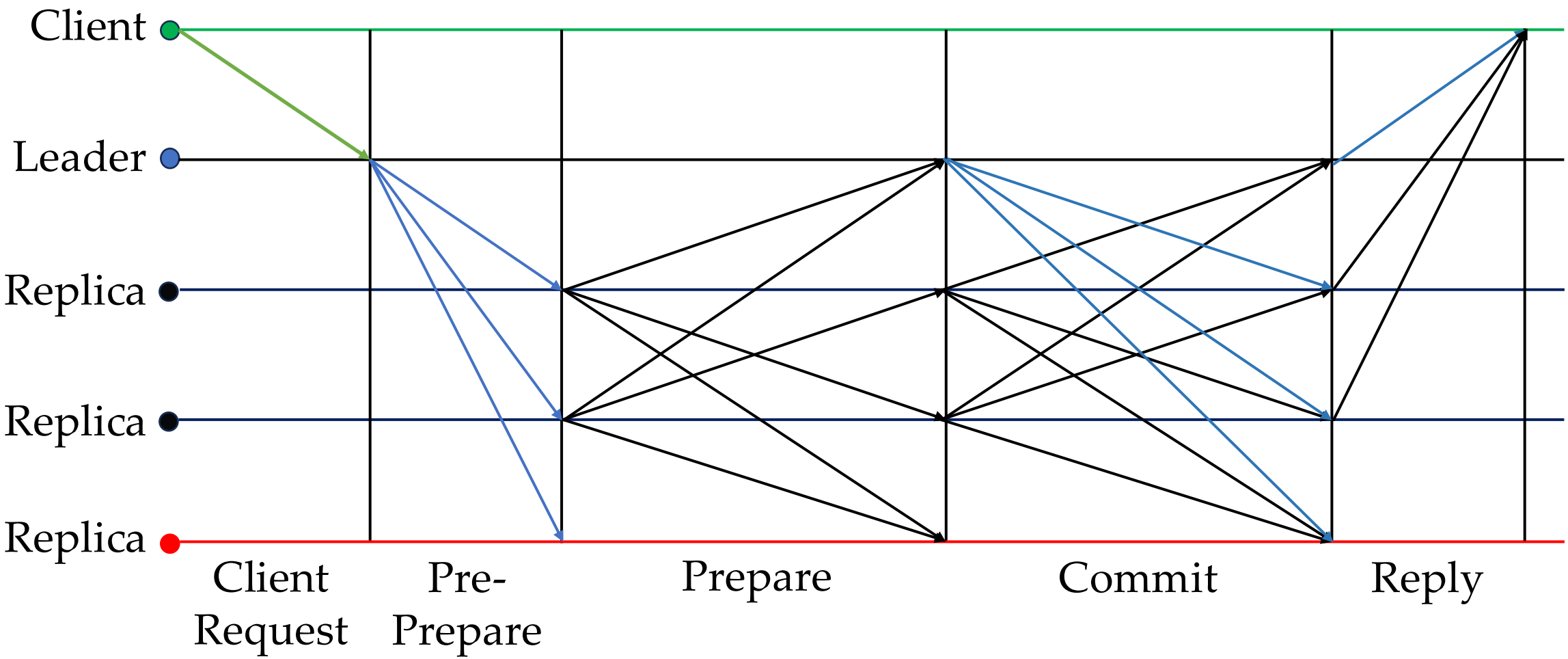
- PBFT Intricacies and view change

Optimizing PBFT

- Today, our goal is to optimize PBFT.

- How can we make PBFT more performant?

PBFT Protocol



Out-of-Order

- Can we allow replicas to process message out-of-order in PBFT?

Out-of-Order

- Can we allow replicas to process message out-of-order in PBFT?
  - Yes, we can allow replicas to process messages out-of-order.

- Will allowing replicas to process out-of-order impact safety?

Out-of-Order

- Can we allow replicas to process message out-of-order in PBFT?
  - Yes, we can allow replicas to process messages out-of-order.

- Will allowing replicas to process out-of-order impact safety?
  - No, because we need to start separating the consensus from execution.

- How can out-of-order message processing help increase PBFT's throughput?

Out-of-Order

- Can we allow replicas to process message out-of-order in PBFT?
  - Yes, we can allow replicas to process messages out-of-order.

- Will allowing replicas to process out-of-order impact safety?
  - No, because we need to start separating the consensus from execution.

- How can out-of-order message processing help increase PBFT's throughput?
  - We can allow the leader to send the next Pre-prepare message without waiting for the previous message to reach the Commit.
  - Essentially, replicas will no longer be idle → parallel-pipelined architecture.

Out-of-Order Processing in PBFT

Out-of-Order Processing in PBFT

- Essentially, the leader can send Pre-Prepare messages at sequence numbers 1, 2, 3, and so on, without waiting.

Out-of-Order Processing in PBFT

- Essentially, the leader can send Pre-Prepare messages at sequence numbers 1, 2, 3, and so on, without waiting.

- When a replica receives a Pre-Prepare message (say sequence number **k**), it starts participating in PBFT assuming that it is interested in only that request.

Out-of-Order Processing in PBFT

- Essentially, the leader can send Pre-Prepare messages at sequence numbers 1, 2, 3, and so on, without waiting.

- When a replica receives a Pre-Prepare message (say sequence number **k**), it starts participating in PBFT assuming that it is interested in only that request.

- When the replica receives **n-f** Commit messages for the **k**-th Pre-Prepare message, it puts the transaction in an execution queue.

Out-of-Order Processing in PBFT

- Essentially, the leader can send Pre-Prepare messages at sequence numbers 1, 2, 3, and so on, without waiting.

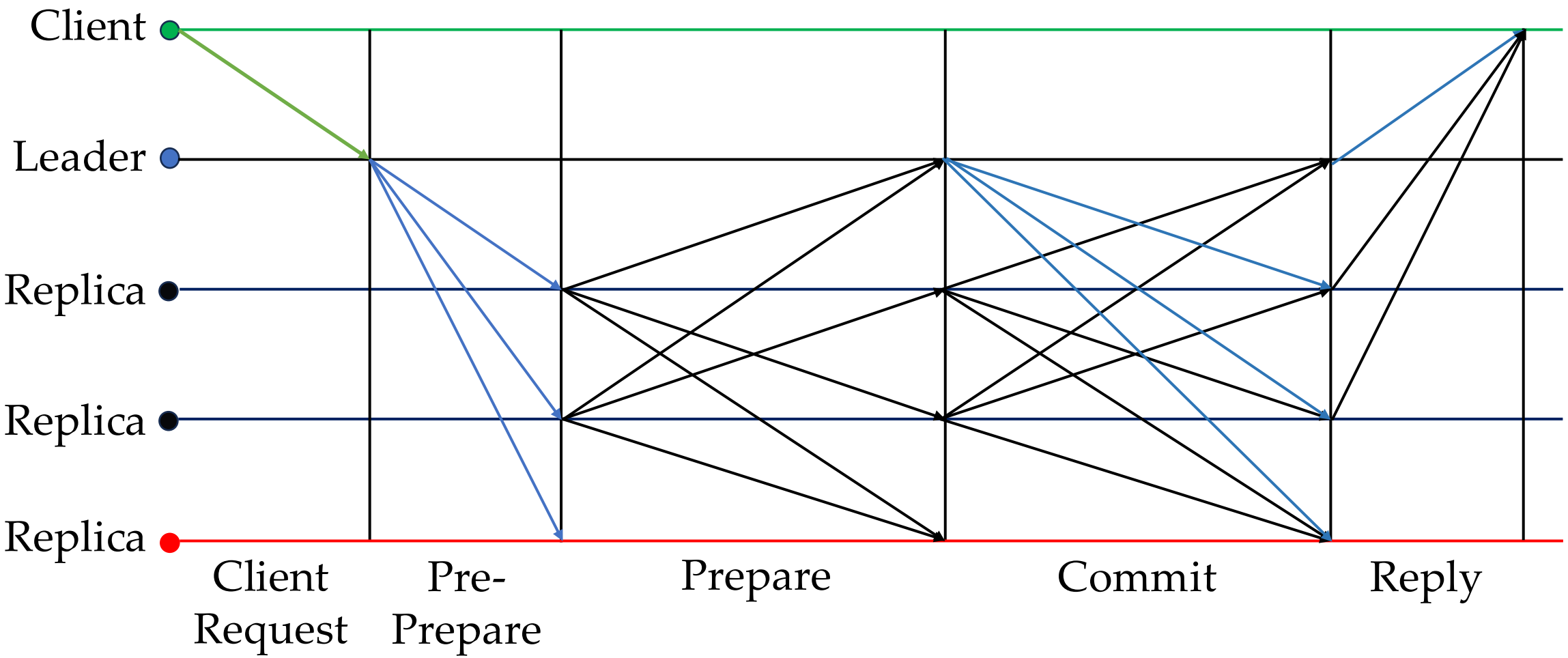
- When a replica receives a Pre-Prepare message (say sequence number **k**), it starts participating in PBFT assuming that it is interested in only that request.

- When the replica receives n-f Commit messages for the **k**-th Pre-Prepare message, it puts the transaction in an execution queue.

- The replica executes **k**-th transaction once it has executed (**k-1**)-th transaction.
  - Ex: Cannot execute 4^{th} transaction without executing the 3^{rd} transaction.

Execution and Reply under Out-of-Order Processing

Execution and Reply under Out-of-Order Processing

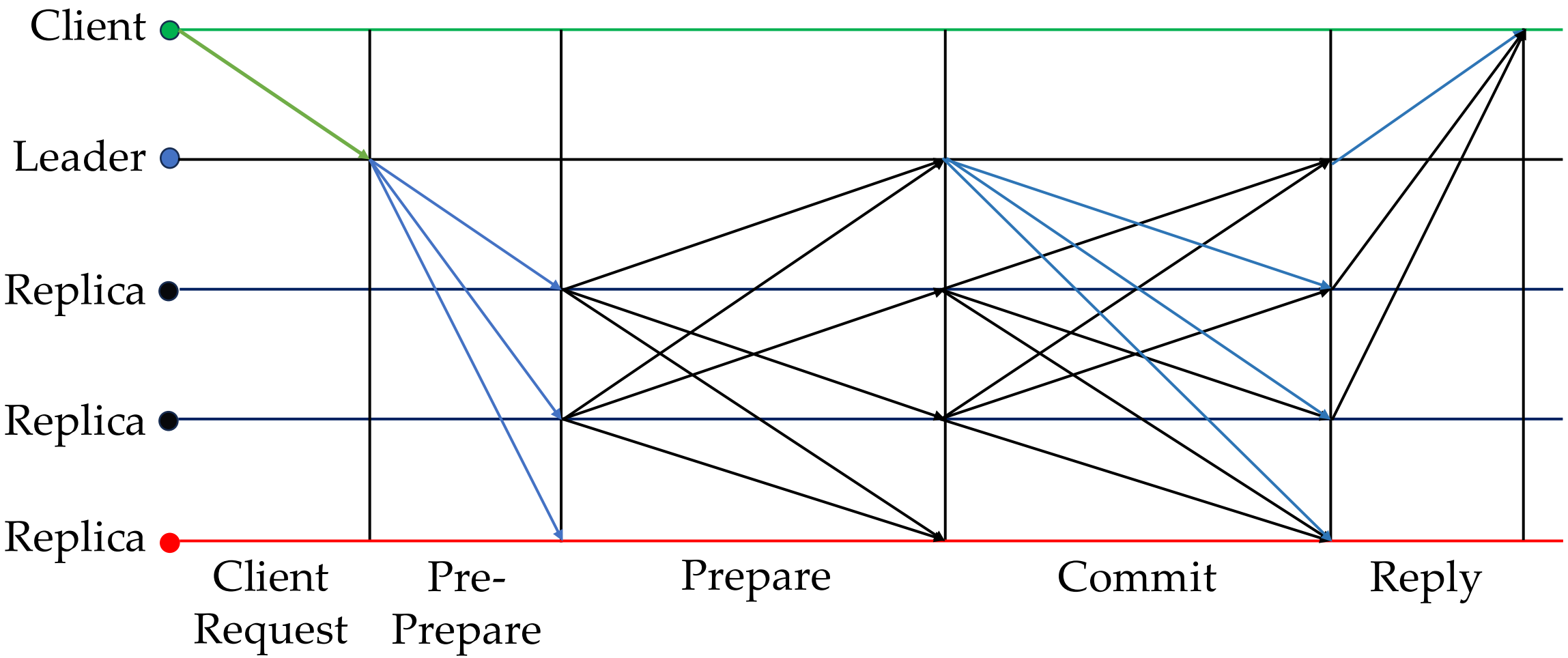


Execution and Reply under Out-of-Order Processing

- No changes in execution and client response procedures.

View Change under Out-of-Order Processing

View Change under Out-of-Order Processing



View Change under Out-of-Order Processing

- View change algorithm becomes trickier under out-of-order processing.

- A replica may have committed a message for the larger sequence number, while waiting for a request for a smaller sequence number.
  - Notice that the replica is keeping track of each sequence number now.

View Change under Out-of-Order Processing

- View change algorithm becomes trickier under out-of-order processing.

- A replica may have committed a message for the larger sequence number, while waiting for a request for a smaller sequence number.
  - Notice that the replica is keeping track of each sequence number now.

- In its view change message, the replica may have **gaps**.
  - Gaps represent sequence numbers for which a replica does not have n-f Prepare messages or even Pre-prepare message.

- The new leader tries to fill all the gaps or ensure that everyone agrees on the gaps.

Speculation in PBFT

Speculation in PBFT

- Permitting speculation in PBFT implies that we want replicas to **eagerly execute** the transactions without waiting for the commit phase to complete.

Speculation in PBFT

- Permitting speculation in PBFT implies that we want replicas to **eagerly execute** the transactions without waiting for the commit phase to complete.

- Where can we apply speculation in PBFT?

Speculation in PBFT

- Permitting speculation in PBFT implies that we want replicas to **eagerly execute** the transactions without waiting for the commit phase to complete.

- Where can we apply speculation in PBFT?

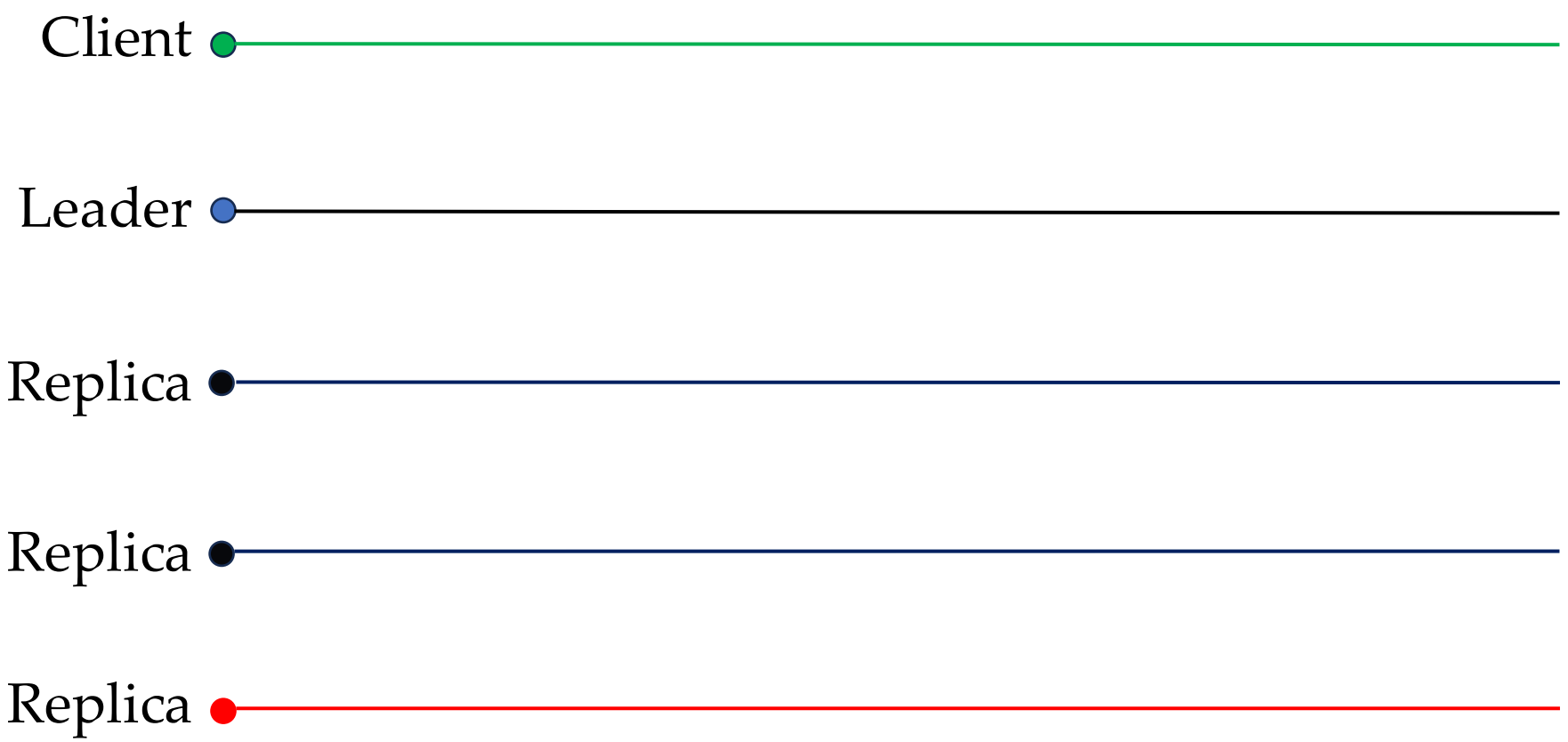
- We can speculate after the Prepare phase → Castro's Thesis and PoE [EDBT'22].

- We can speculate after the Pre-Prepare phase → Zyzzyva [SOSP'07].

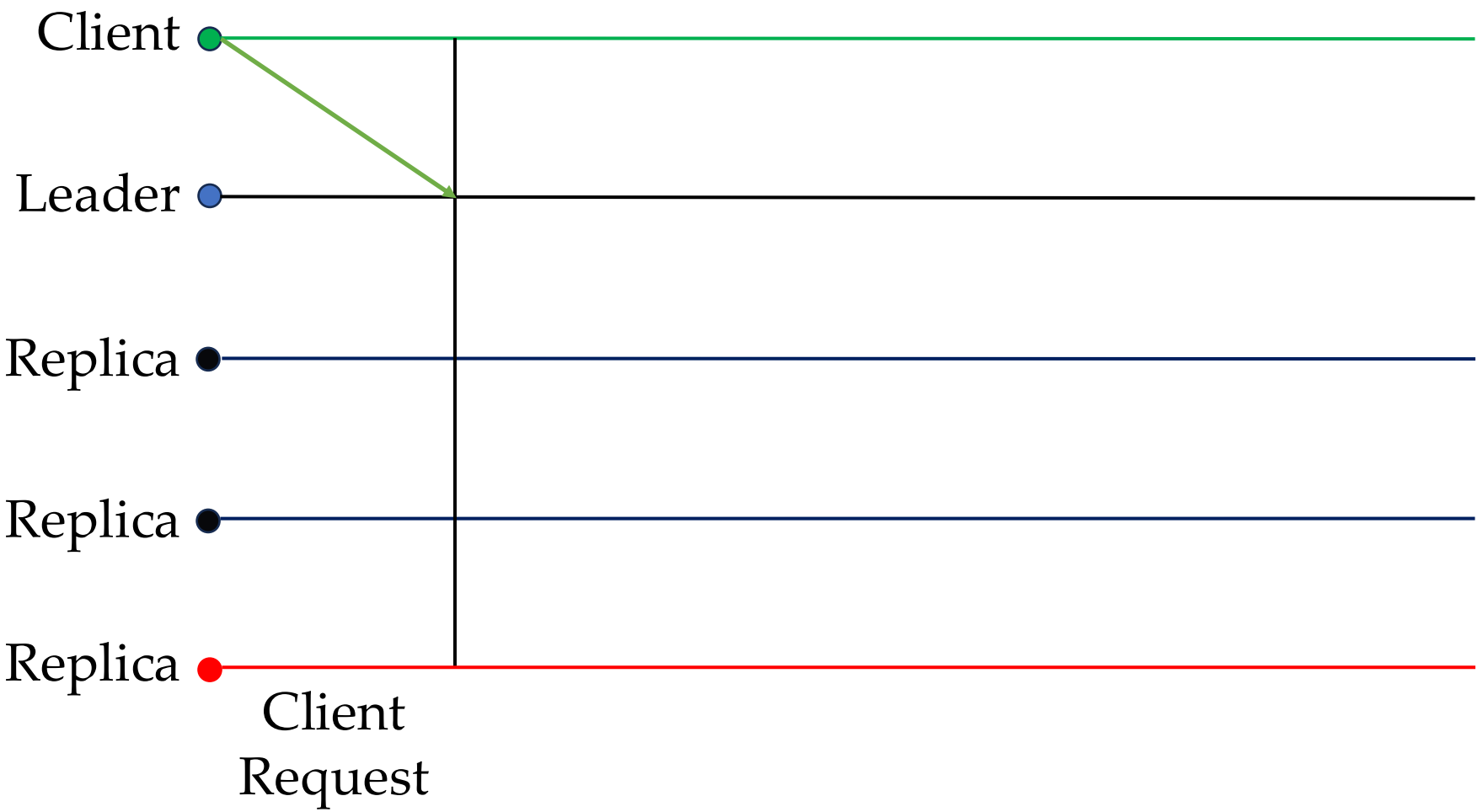
Speculative Execution after Prepare Phase

- The safest route is to speculatively execute the transaction after the Prepare phase.
- This idea was first mentioned in Miguel Castro's thesis.
- **Proof-of-execution (PoE)** expanded this idea, illustrated its limitation, and presented a rigorous evaluation.

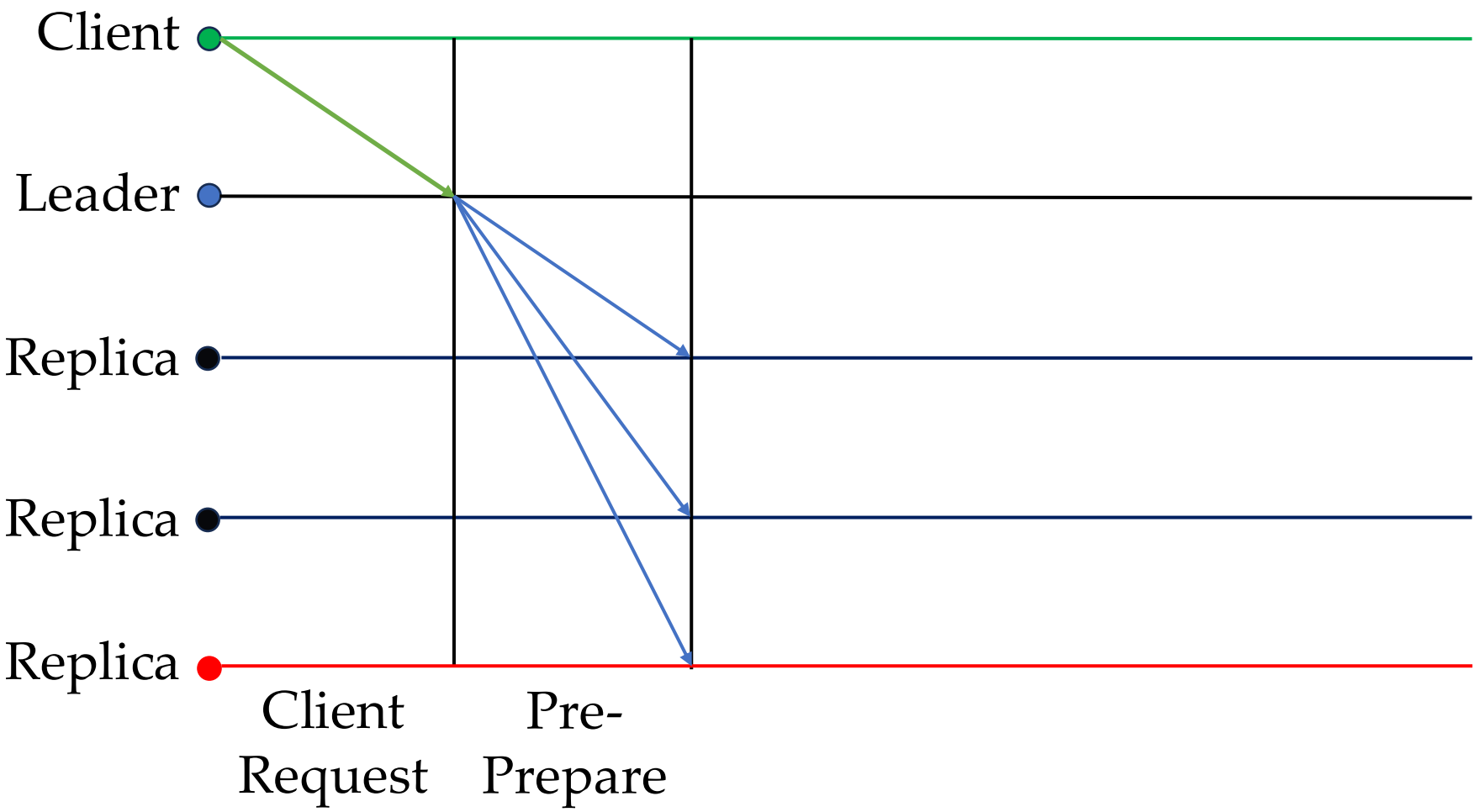
PoE Protocol



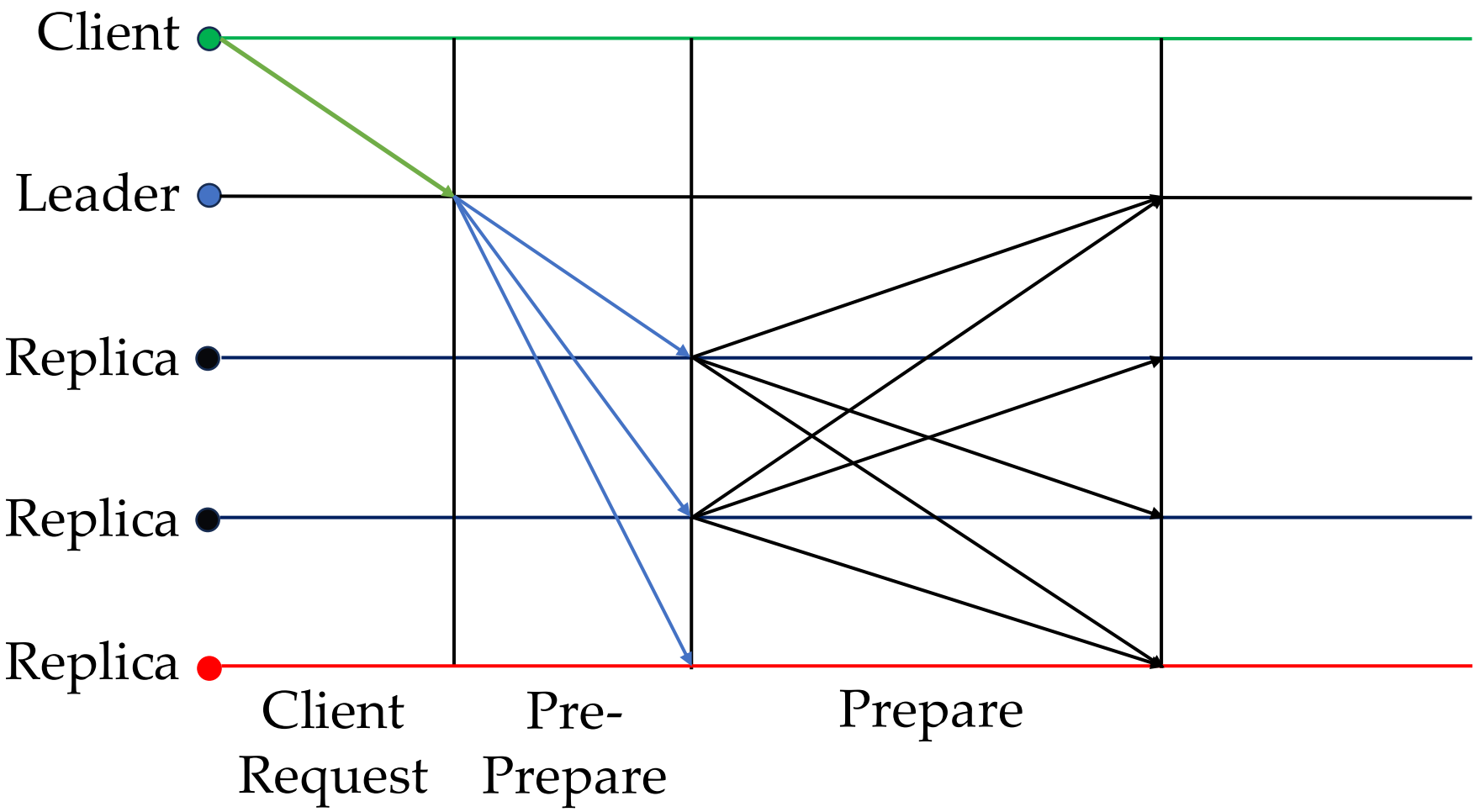
PoE Protocol



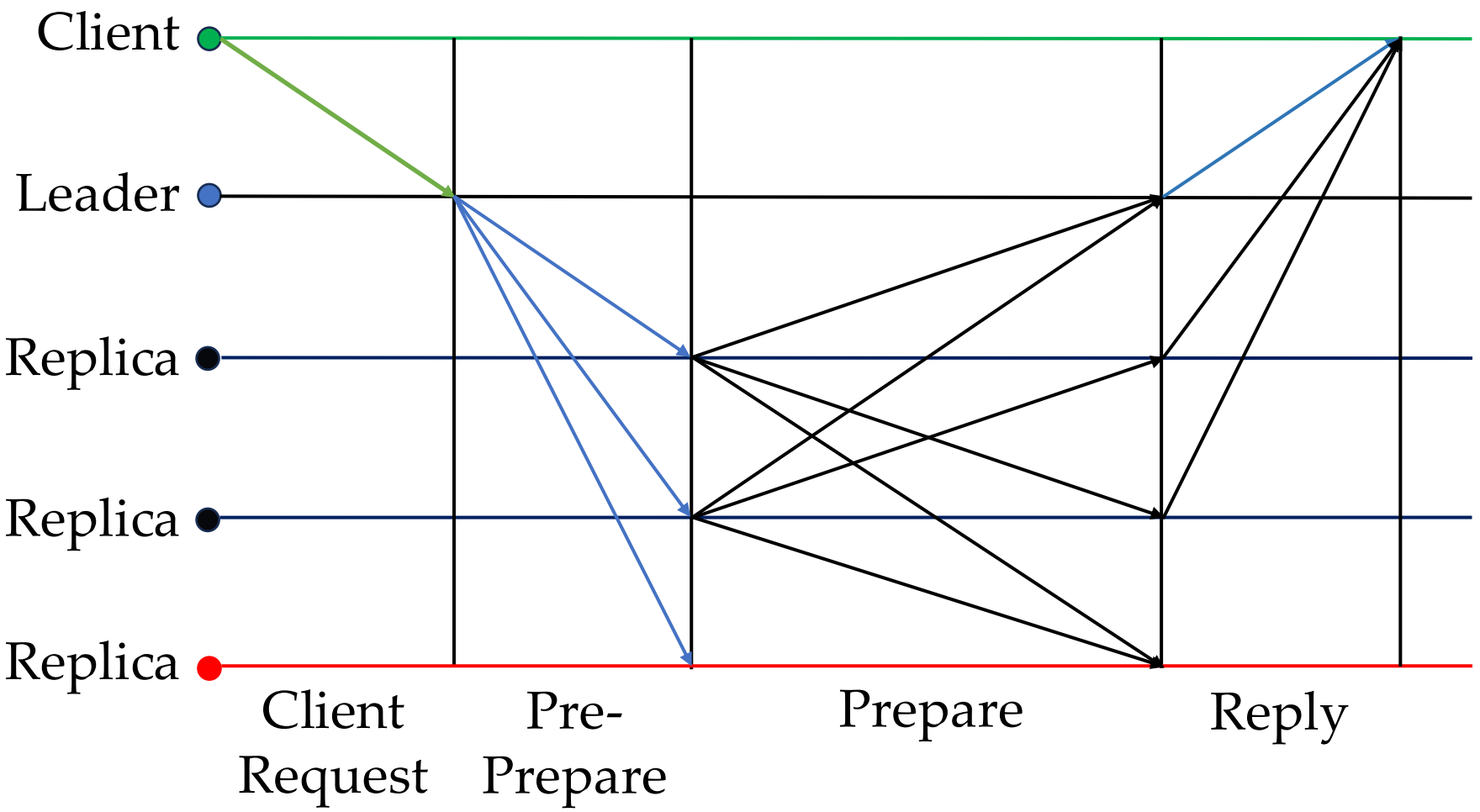
PoE Protocol



PoE Protocol



PoE Protocol



PoE Protocol

- **Key Points:**
  - Once a replica receives a Pre-Prepare message, it sends a corresponding Prepare message.
  - Once a replica receives n-f Prepare messages, it executes the request.
  - Finally, the replica replies to the client.

Key Challenges for PoE

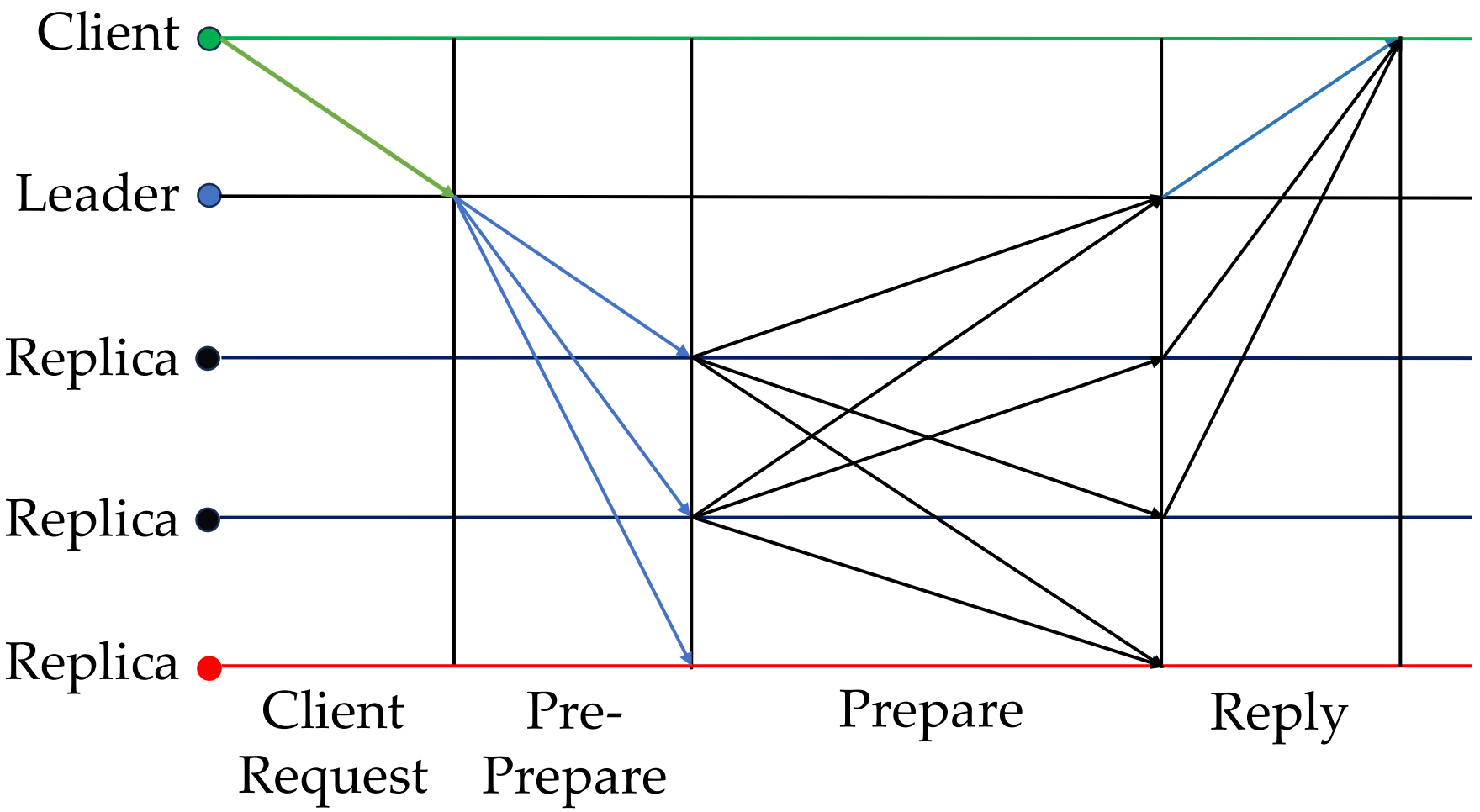
- What are the key challenges for PoE?

Key Challenges for PoE

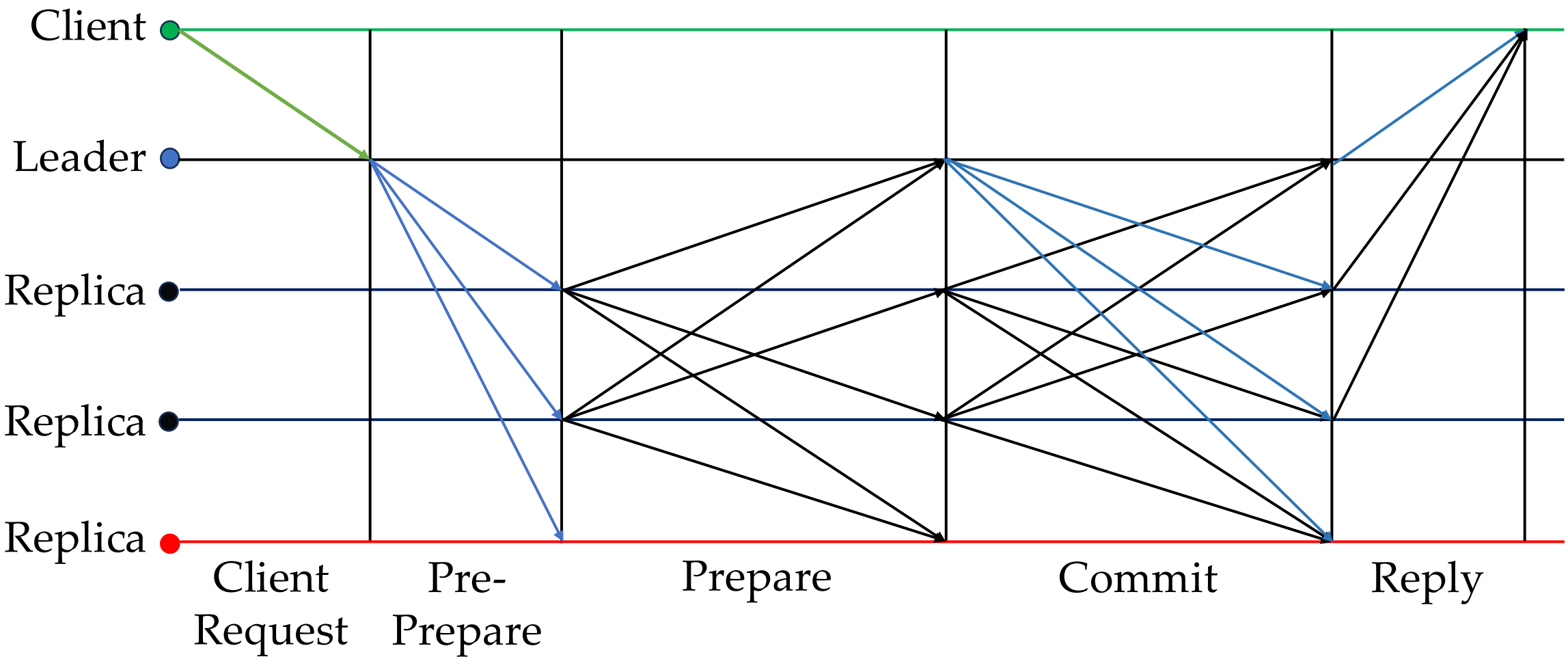
- What are the key challenges for PoE?
  - How can a replica know if the request is committed?
  - How can a client know if the request is committed?

Commit Information for Replica

PoE Protocol



PBFT Protocol



Commit Information for Replica

- A replica only comes to know if a request has committed, once the subsequent request gets committed.
  - Why?

Commit Information for Replica

- A replica only comes to know if a request has committed, once the subsequent request gets committed.
  - Why?
  - Because the assumption is that requests are being processing in order, so the state is known to all.
  - Acts like prepare and commit phases of PBFT.

Commit Information for Replica

- A replica only comes to know if a request has committed, once the subsequent request gets committed.
  - Why?
  - Because the assumption is that requests are being processing in order, so the state is known to all.
  - Acts like prepare and commit phases of PBFT.

- But what happens under out-of-order message processing?

Commit Information for Replica

- A replica only comes to know if a request has committed, once the subsequent request gets committed.
  - Why?
  - Because the assumption is that requests are being processing in order, so the state is known to all.
  - Acts like prepare and commit phases of PBFT.

- But what happens under out-of-order message processing?
  - It is possible that (**k+1**)-th request got prepared but **k**-th request did not arrive?
  - How to handle this?
  - Allow replicas to **rollback** their state.

Commit Information for Client

Commit Information for Replica

- A client cannot rollback → we need to give client precise information.

Commit Information for Replica

- A client cannot rollback → we need to give client precise information.

- For clients, PoE sets the following rule:
  - Each client needs to wait for **n-f** responses instead of **f+1** responses.

  - If a client receives **n-f** responses, it has the guarantee that its request will eventually commit.

  - If a client does not receive **n-f** responses, then it does not have any information about the fate of the request.

State Rollback

- When should a replica rollback its state?

State Rollback

- When should a replica rollback its state?
  - A replica rollbacks its state when it receives from the leader of the current view a conflicting state.

- How many maximum requests a replica may need to rollback.

State Rollback

- When should a replica rollback its state?
  - A replica rollbacks its state when it receives from the leader of the current view a conflicting state.

- How many maximum requests a replica may need to rollback.
  - All the requests since the last checkpoint.

- So when is the replica's state stable?

State Rollback

- When should a replica rollback its state?
  - A replica rollbacks its state when it receives from the leader of the current view a conflicting state.

- How many maximum requests a replica may need to rollback.
  - All the requests since the last checkpoint.

- So when is the replica's state stable?
  - At the time of checkpoint.

- When is it guaranteed that a replica's state will not be rollbacked.

State Rollback

- When should a replica rollback its state?
  - A replica rollbacks its state when it receives from the leader of the current view a conflicting state.

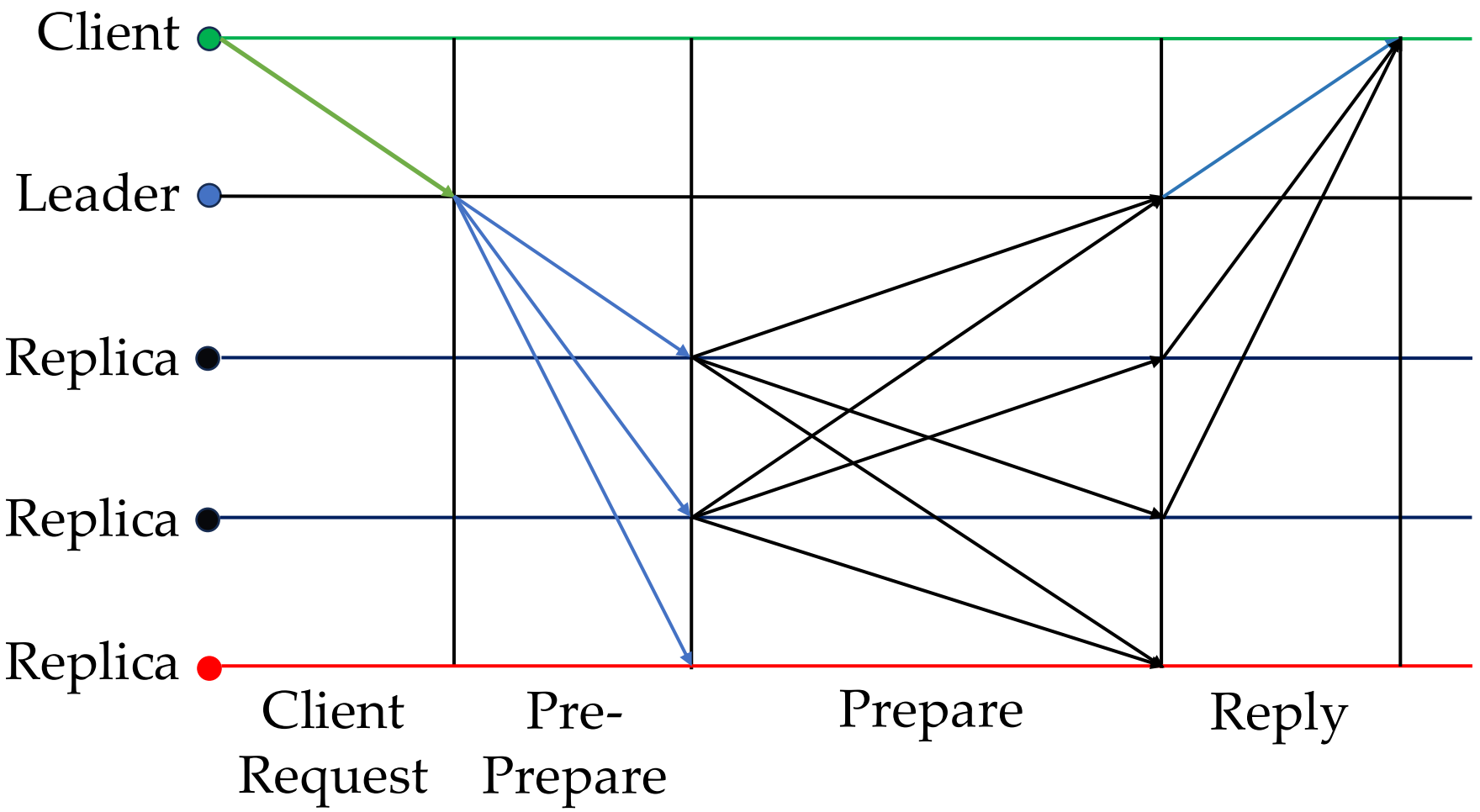
- How many maximum requests a replica may need to rollback.
  - All the requests since the last checkpoint.

- So when is the replica's state stable?
  - At the time of checkpoint.

- When is it guaranteed that a replica's state will not be rollbacked.
  - When a request has been executed by at least f+1 replicas, it is guaranteed to persist.

PoE's Responsiveness Dilemma

PoE Protocol



PoE's Responsiveness Dilemma

- PoE faces a responsiveness challenge:
  - The replicated system is both safe and live.
  - But clients do not receive n-f responses.
  - The leader cannot be replaced.

- Why?

PoE's Responsiveness Dilemma

- Recall that to replace the leader, support of **n-f** replicas is needed.
- The Byzantine leader plays cleverly, only includes **f+1** honest replicas in consensus.
- These **f+1** honest replicas will keep on preparing one request after another.
- However, client will never receive **n-f** responses because the **f** Byzantine replicas that helped to Prepare the requests will not send a response back to the client.
- Further, view change cannot take place as the **f+1** honest replicas have not observed any Byzantine behavior from the leader.