Large Scale Systems
CS 410 / 510

Lecture 5:

Distributed Commit Protocols

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 1 is Out!

- **Assignment 1 is out!**
- Please work with your groups to understand the underlying system.
- Assignment 1 report **deadline** → April 16, 2026 at 11:59pm.

Useful Readings

- In general, a good practice is to read the recommended books end-to-end.

- From **Principles of Distributed Database Systems**:
  - Chapters 1, 10
  - Chapter 11.1, 11.2, 11.3, 11.6, 11.7
  - Chapters 12, and 13 (*to be covered*)

Last Class

- Last class we looked at:

- Distributed Transaction Processing
- Concurrency Control → 2PL
- Necessary actions when running a distributed transaction → Undo, Redo

Transaction Fate

- Once a distributed transaction is executed by all the participants, we need to decide whether to commit or abort the transaction.
- When will a transaction commit or abort?
- A transaction **commits** when all the participants agree.
- A transaction **aborts** when one of the participant disagrees.

Abort or Commit Decision Making

- How to decide whether to abort or commit a transaction?
- Remember, we are in a distributed setting and decisions of all the participants need to be taken into consideration.
- One participant may say commit, while other says abort.
- How should we proceed?

Commit Protocols

Commit Protocols

- Commit protocols help to reach an **agreement** among the partitions.

- Agreement implies that somehow every partition gets a chance to vote and based on the votes a final decision is reached.
  - Every partition promises to follow this decision.

Commit Protocols

- Commit protocols help to reach an **agreement** among the partitions.

- Agreement implies that somehow every partition gets a chance to vote and based on the votes a final decision is reached.
  - Every partition promises to follow this decision.

- Commit Protocols come in several flavor:
  - Two Phase Commit, Three Phase Commit, Easy Commit, etc.

Two-Phase Commit Protocol

- First commit protocol.

- As the name suggests, in the absence of any failures, it helps participants reach an agreement in two phases.

Two-Phase Commit Protocol

- First commit protocol.

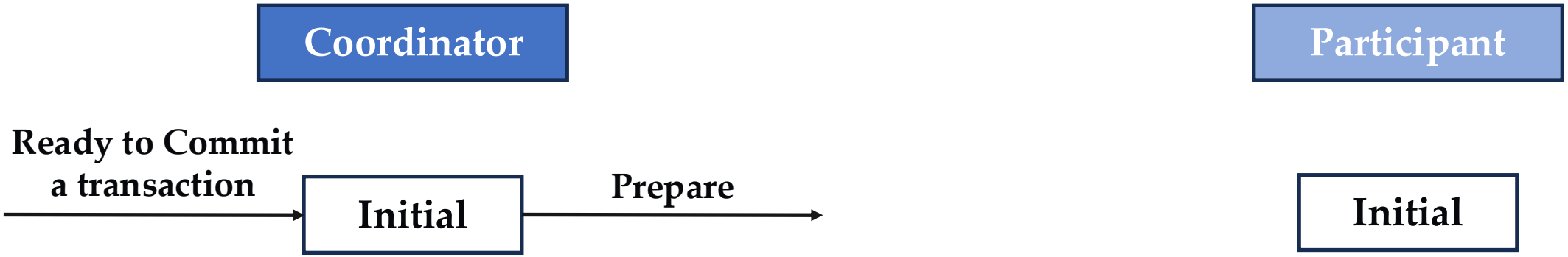
- As the name suggests, in the absence of any failures, it helps participants reach an agreement in two phases.

- The logic behind Two-Phase Commit (**2PC**) protocol is simple:
  - If **all the participants** agree to commit the transaction, then global decision is commit.
  - If **any one participant** decides to abort the transaction, then global decision is abort.

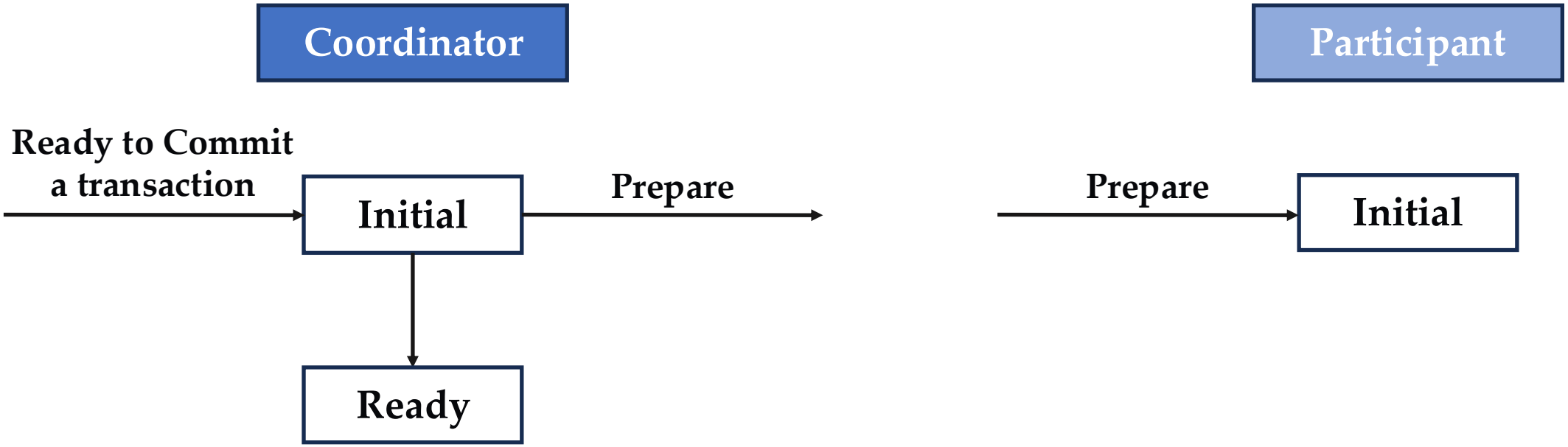
2PC Protocol Flow

2PC Protocol Flow

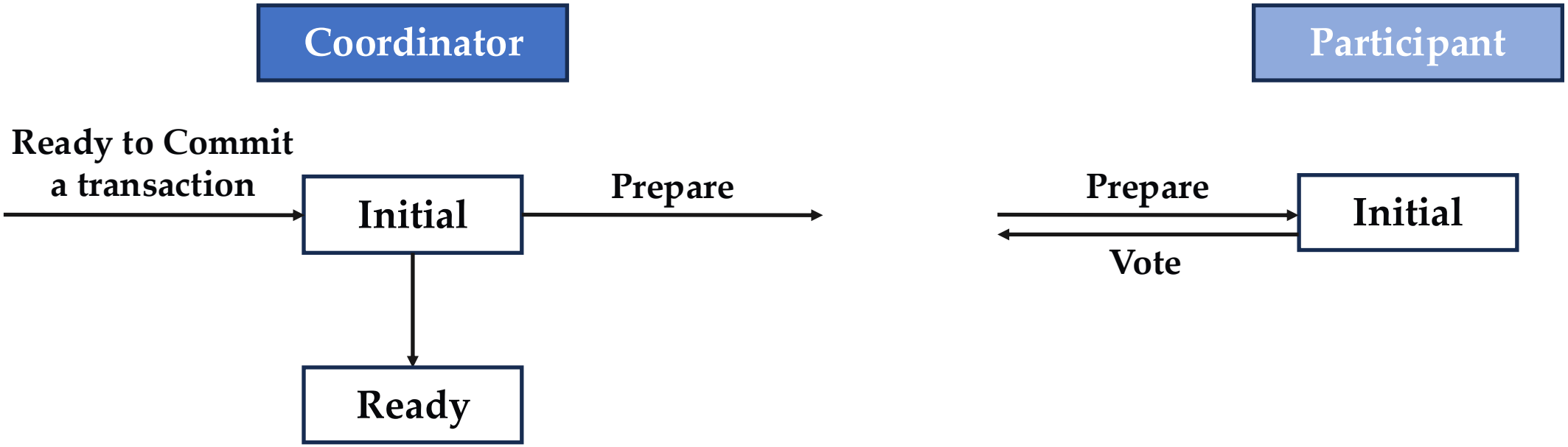
2PC Protocol Flow



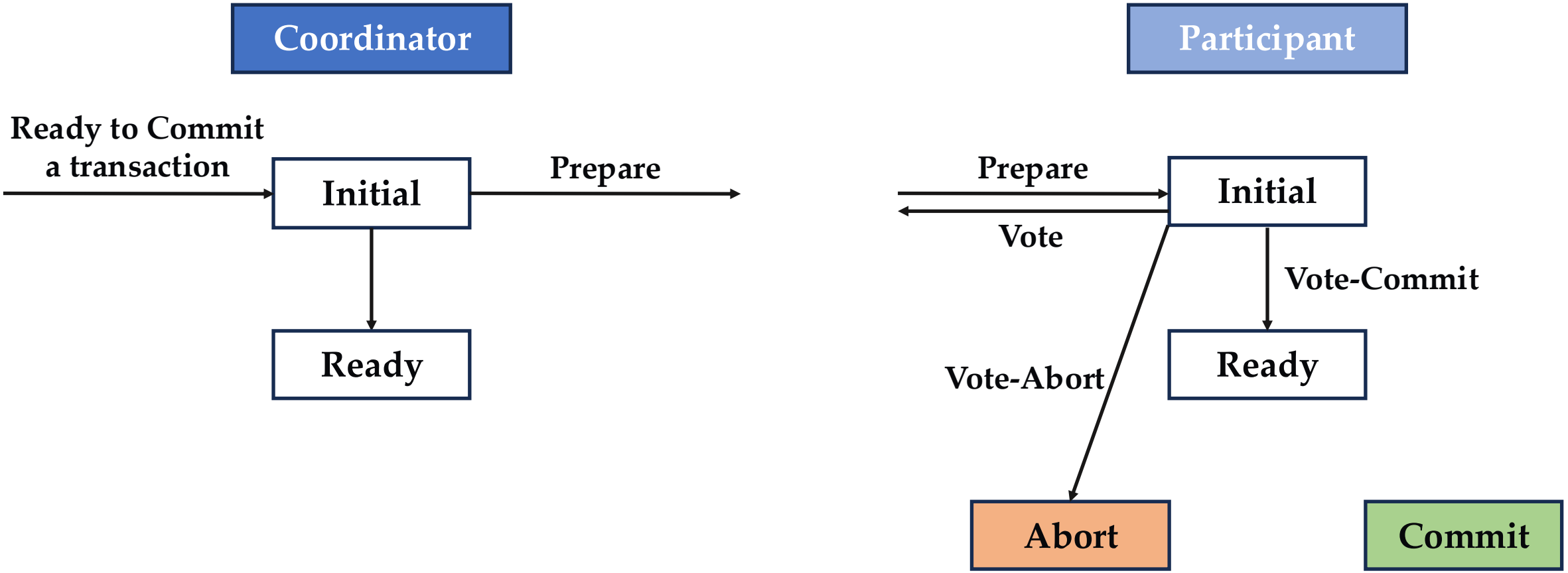
2PC Protocol Flow



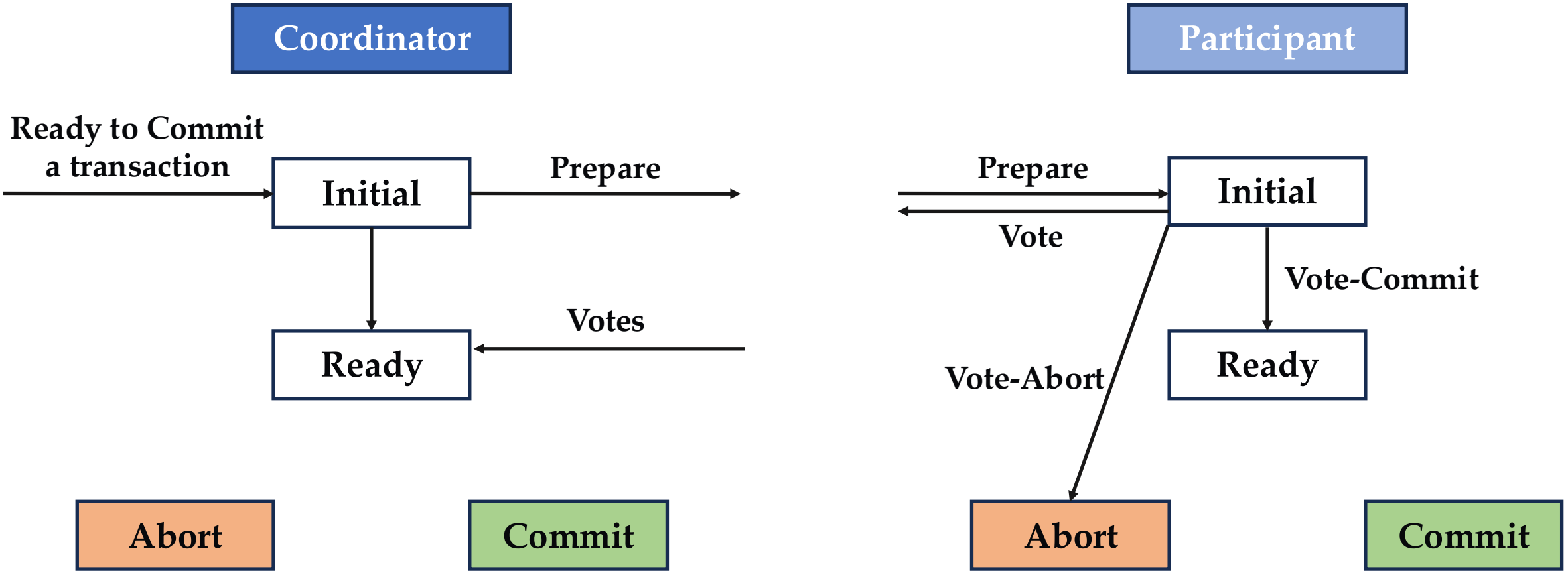
2PC Protocol Flow



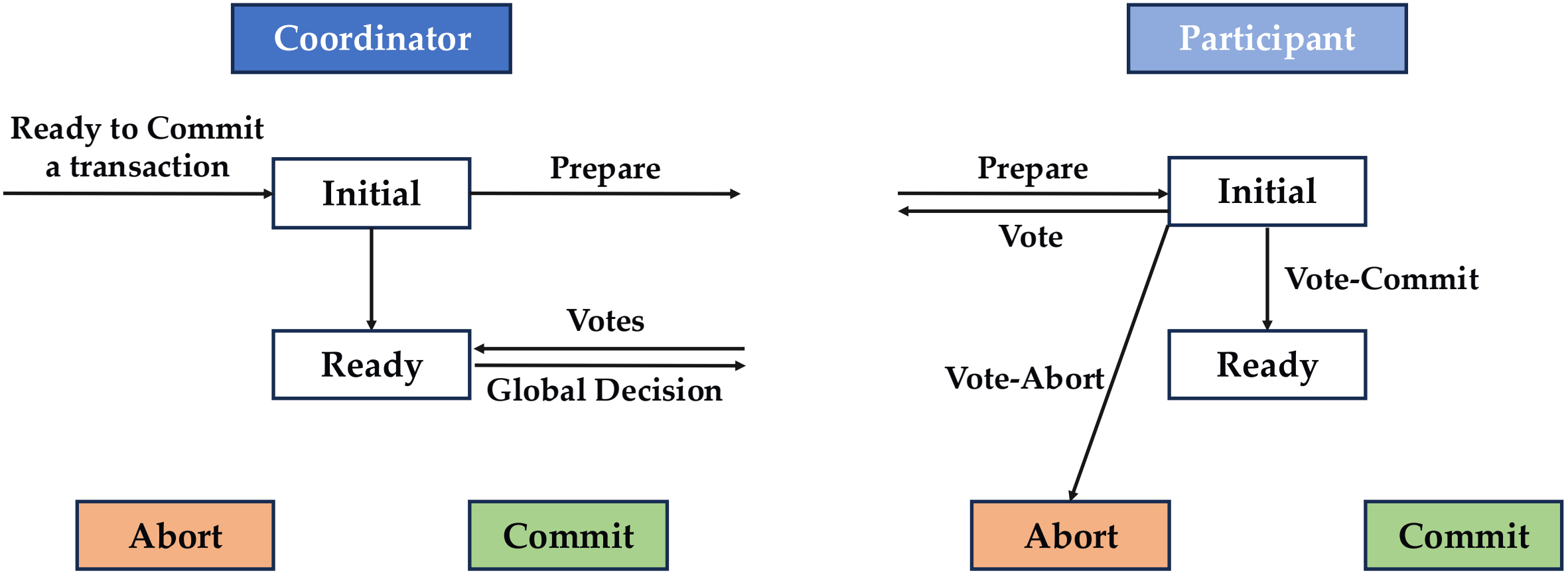
2PC Protocol Flow



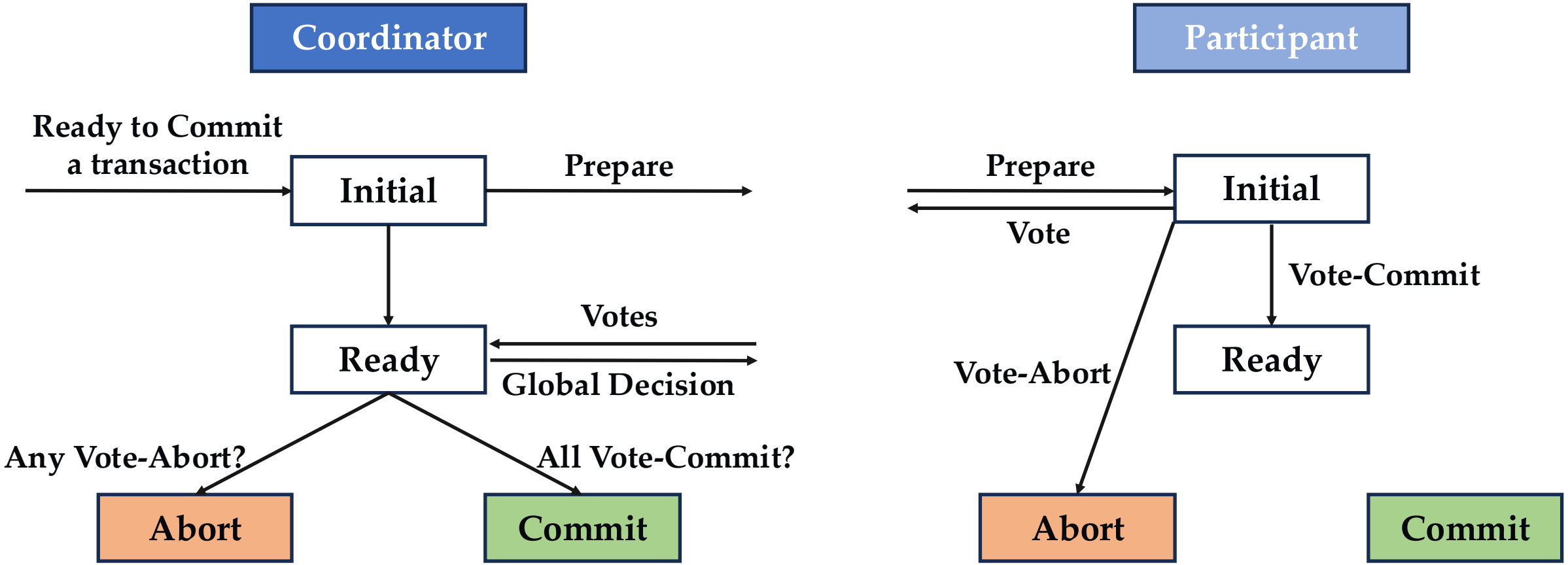
2PC Protocol Flow



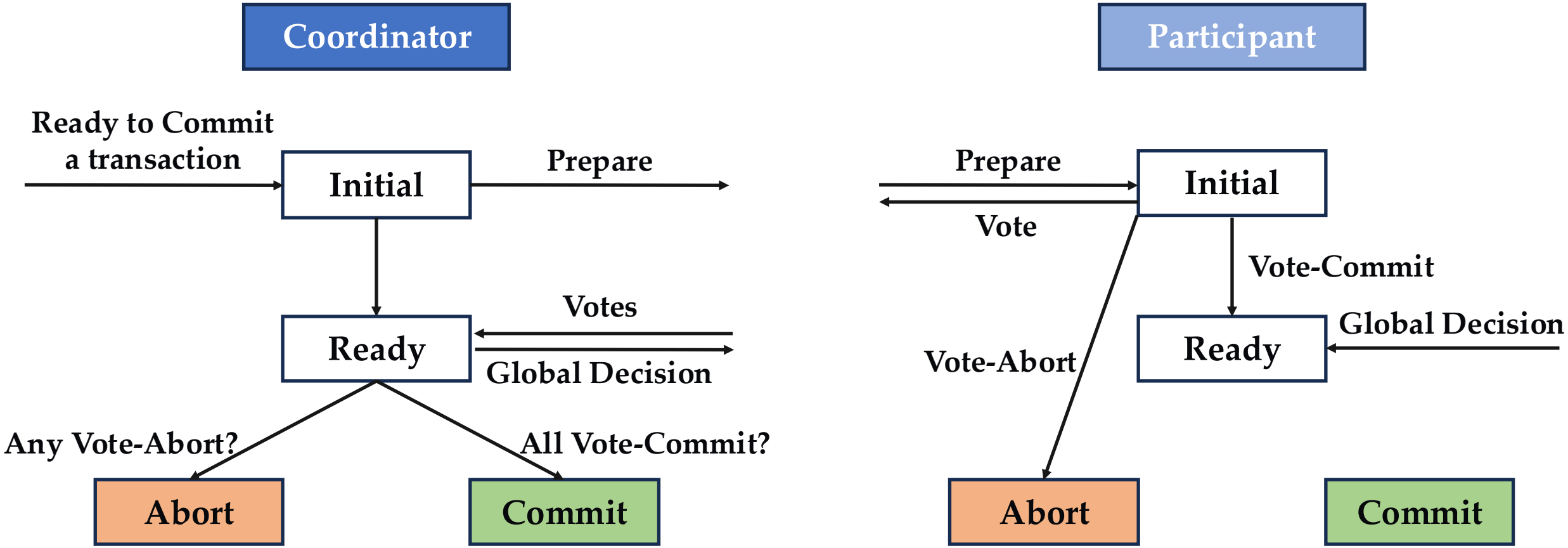
2PC Protocol Flow



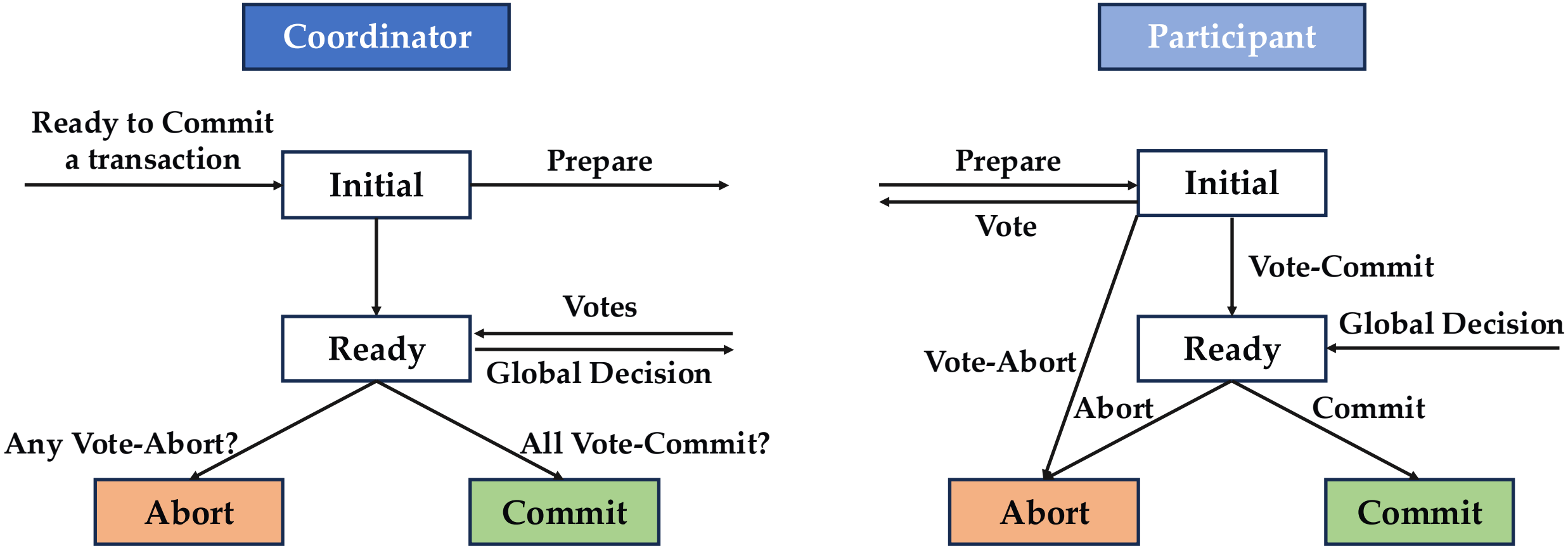
2PC Protocol Flow



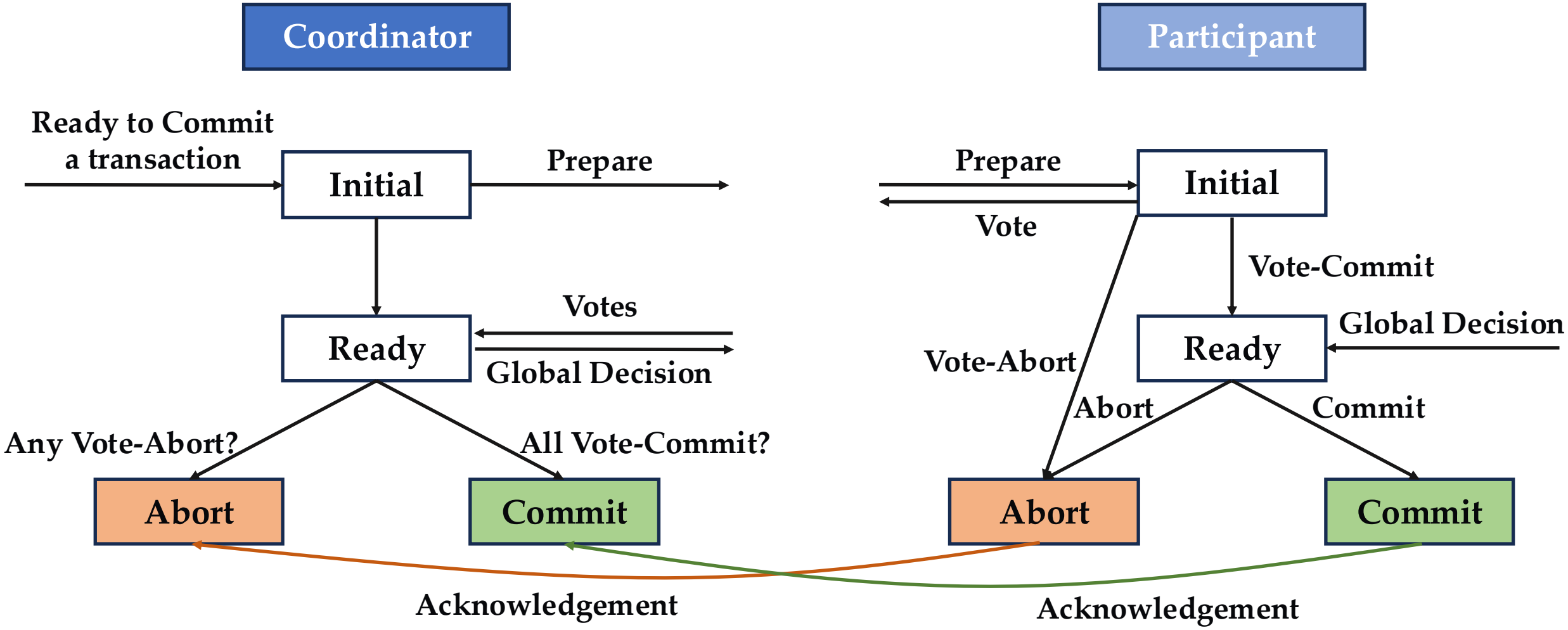
2PC Protocol Flow



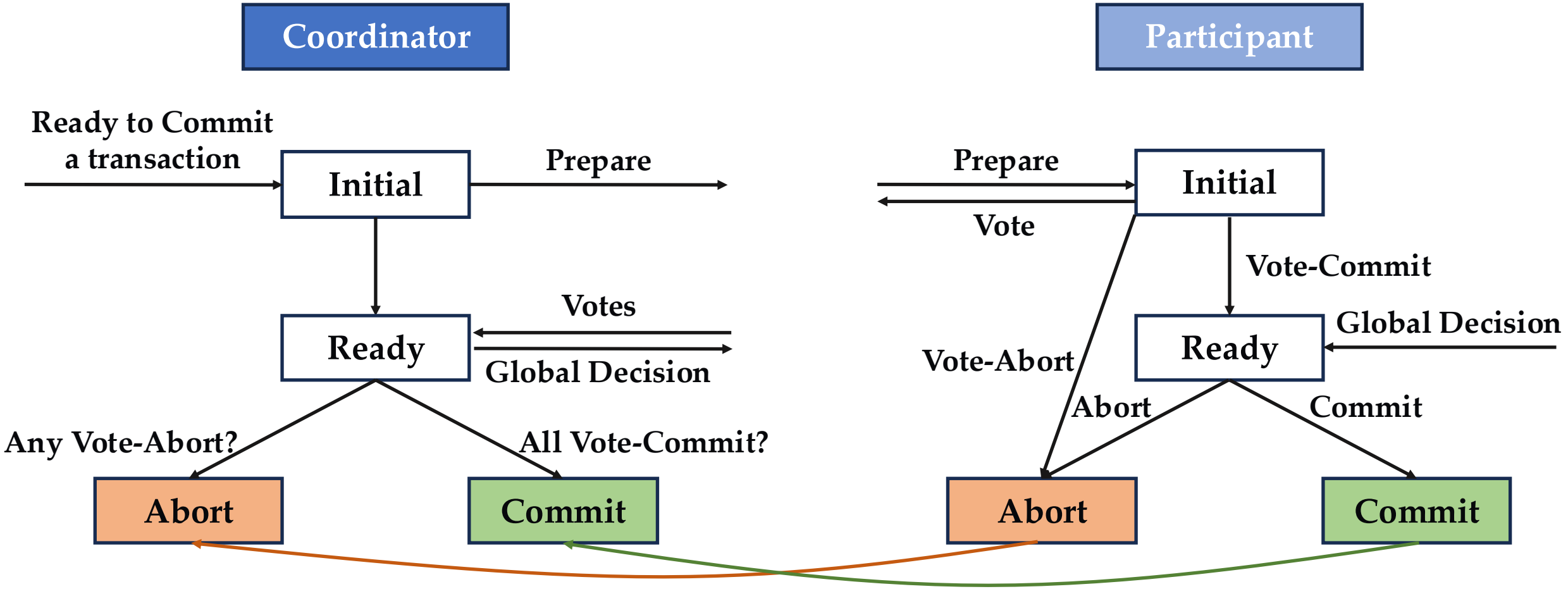
2PC Protocol Flow



2PC Protocol Flow

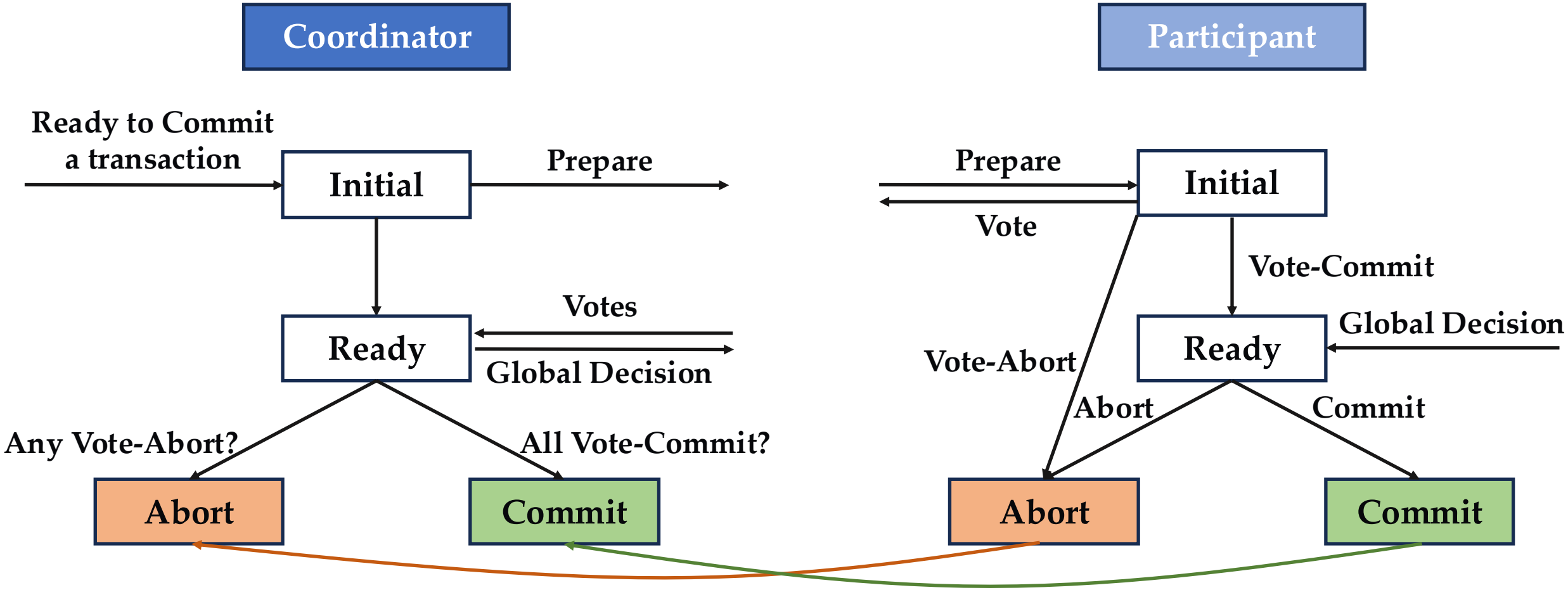


2PC Protocol Key Takeaways



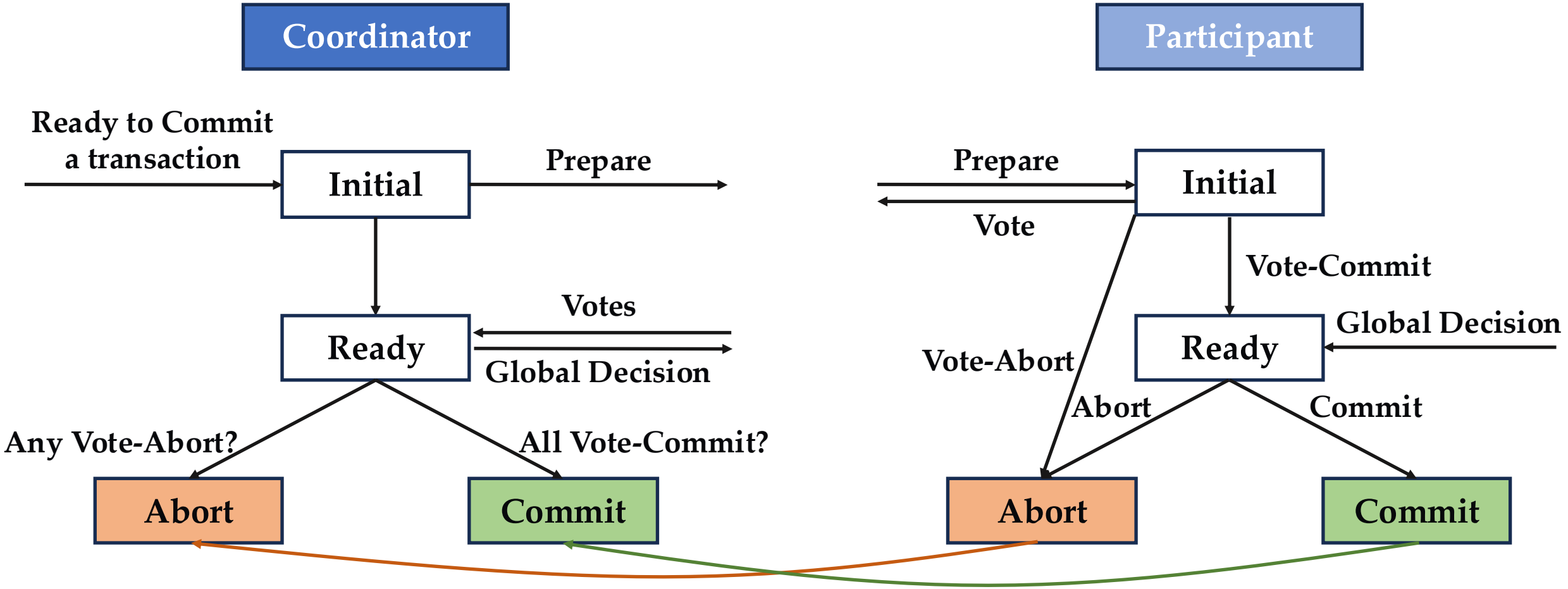
- 2PC permits a participant to unilaterally abort a transaction.

2PC Protocol Key Takeaways



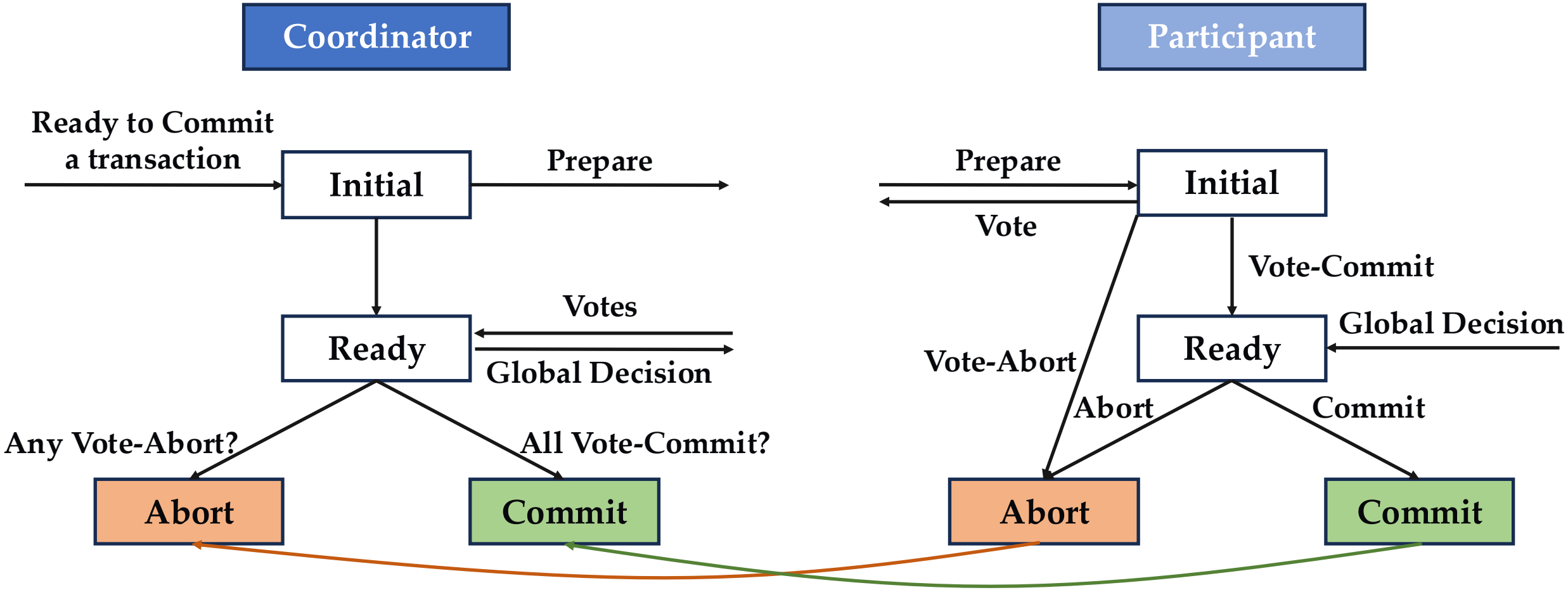
- Once a participant votes to commit or abort a transaction, it cannot change its vote.

2PC Protocol Key Takeaways



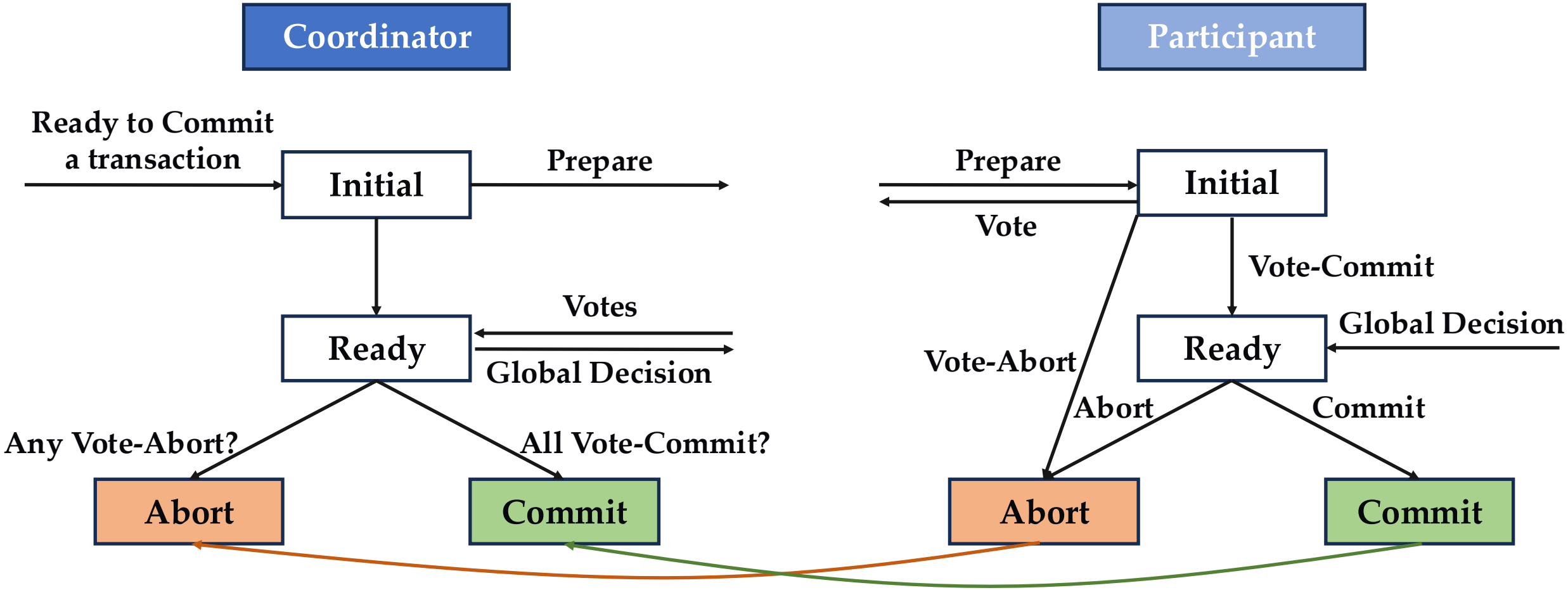
- In the Ready state, based on the global decision it receives from the coordinator, a participant can move either to abort or to commit state.

2PC Protocol Key Takeaways



- Before sending any message or changing states, the coordinator/participants need to write an entry to the log.

2PC Protocol Key Takeaways



- Some states allow the coordinator/participants to exit only when they receive messages from others. How can they exit if they do not receive required messages

2PC Termination

- How can 2PC protocol manage coordinator/participant failures?

2PC Termination Protocol

- The termination protocol allows 2PC to make progress under failures.

- How to start a termination protocol?

2PC Termination Protocol

- The termination protocol allows 2PC to make progress under failures.

- How to start a termination protocol?
  - Use timers.

  - Whenever the coordinator/participant enters a new state, it starts a timer.

2PC Termination Protocol

- The termination protocol allows 2PC to make progress under failures.

- How to start a termination protocol?
  - Use timers.

  - Whenever the coordinator/participant enters a new state, it starts a timer.

  - The timer defines the maximum amount of time the coordinator/participant should wait before starting the termination protocol.

  - When the timer timeouts → initiate the termination protocol.

2PC Termination Protocol

- The termination protocol allows 2PC to make progress under failures.

- How to start a termination protocol?
  - Use timers.

  - Whenever the coordinator/participant enters a new state, it starts a timer.

  - The timer defines the maximum amount of time the coordinator/participant should wait before starting the termination protocol.

  - When the timer timeouts → initiate the termination protocol.

  - Termination protocol is a state transition protocol → allows moving across states.

2PC Coordinator Timeout

- **Timeout in Ready State:**

2PC Coordinator Timeout

- **Timeout in Ready State:**
  - Implies that the coordinator did not receive votes from all the participants.

2PC Coordinator Timeout

- **Timeout in Ready State:**
  - Implies that the coordinator did not receive votes from all the participants.
  - Coordinator cannot unilaterally commit the transaction. Why?

2PC Coordinator Timeout

- **Timeout in Ready State:**
  - Implies that the coordinator did not receive votes from all the participants.
  - Coordinator cannot unilaterally commit the transaction. Why?
  - Coordination aborts the transaction and sends Abort as the global decision.

2PC Coordinator Timeout

- **Timeout in Ready State:**
  - Implies that the coordinator did not receive votes from all the participants.
  - Coordinator cannot unilaterally commit the transaction. Why?
  - Coordination aborts the transaction and sends Abort as the global decision.
- **Timeout in Abort/Commit State:**

2PC Coordinator Timeout

- **Timeout in Ready State:**
  - Implies that the coordinator did not receive votes from all the participants.
  - Coordinator cannot unilaterally commit the transaction. Why?
  - Coordination aborts the transaction and sends Abort as the global decision.
- **Timeout in Abort/Commit State:**
  - Implies that the coordinator did not receive acknowledgments from all the participants.

2PC Coordinator Timeout

- **Timeout in Ready State:**
  - Implies that the coordinator did not receive votes from all the participants.
  - Coordinator cannot unilaterally commit the transaction. Why?
  - Coordination aborts the transaction and sends Abort as the global decision.
- **Timeout in Abort/Commit State:**
  - Implies that the coordinator did not receive acknowledgments from all the participants.
  - Coordinator repeatedly sends the global decision to participants that have not responded.

2PC Participant Timeout

- **Timeout in Initial State:**

2PC Participant Timeout

- **Timeout in Initial State:**
  - Implies → participant is waiting for Prepare message

2PC Participant Timeout

- **Timeout in Initial State:**
  - Implies → participant is waiting for Prepare message
  - Participant unilaterally aborts the transaction.

2PC Participant Timeout

- **Timeout in Initial State:**
  - Implies → participant is waiting for Prepare message
  - Participant unilaterally aborts the transaction.
  - What if the participant receives a Prepare in the future?

2PC Participant Timeout

- **Timeout in Initial State:**
  - Implies → participant is waiting for Prepare message
  - Participant unilaterally aborts the transaction.
  - What if the participant receives a Prepare in the future?
    - Either ignore or send a vote-abort.

2PC Participant Timeout

- **Timeout in Initial State:**
  - Implies → participant is waiting for Prepare message
  - Participant unilaterally aborts the transaction.
  - What if the participant receives a Prepare in the future?
    - Either ignore or send a vote-abort.
- **Timeout in Ready State:**

2PC Participant Timeout

- **Timeout in Initial State:**
  - Implies → participant is waiting for Prepare message
  - Participant unilaterally aborts the transaction.
  - What if the participant receives a Prepare in the future?
    - Either ignore or send a vote-abort.
- **Timeout in Ready State:**
  - Implies → participant voted to commit but has not received the global decision.

2PC Participant Timeout

- **Timeout in Initial State:**
  - Implies → participant is waiting for Prepare message
  - Participant unilaterally aborts the transaction.
  - What if the participant receives a Prepare in the future?
    - Either ignore or send a vote-abort.
- **Timeout in Ready State:**
  - Implies → participant voted to commit but has not received the global decision.
  - Cannot unilaterally make a decision, or change its own vote (to vote abort) .

2PC Participant Timeout

- **Timeout in Initial State:**
  - Implies → participant is waiting for Prepare message
  - Participant unilaterally aborts the transaction.
  - What if the participant receives a Prepare in the future?
    - Either ignore or send a vote-abort.

- **Timeout in Ready State:**
  - Implies → participant voted to commit but has not received the global decision.
  - Cannot unilaterally make a decision, or change its own vote (to vote abort) .
  - Participant remains blocked until it hears a decision.

2PC Blocking

- Participant remaining blocked impacts system liveness!
- What can we do?

2PC Blocking

- Participant remaining blocked impacts system liveness!
- What can we do?
- Several important scenarios to consider?
  - Can participants communicate with each other?
  - Who all have failed?
  - Can unilateral decisions be made?

2PC Blocking

- If the communication is not centralized, then participants can communicate with each other and try to make progress.

- If a participant timeouts in Initial State (did not get a chance to vote) and has not received a global decision, it can unilaterally abort.
  - Post unilaterally aborting, forward the decision to everyone.

2PC Blocking

- If the communication is not centralized, then participants can communicate with each other and try to make progress.

- If a participant timeouts in Initial State (did not get a chance to vote) and has not received a global decision, it can unilaterally abort.
  - Post unilaterally aborting, forward the decision to everyone.

- If a participant timeouts in Ready State (voted to commit) and it can determine that only the coordinator failed, it can still make progress.

2PC Blocking

- If the communication is not centralized, then participants can communicate with each other and try to make progress.

- If a participant timeouts in Initial State (did not get a chance to vote) and has not received a global decision, it can unilaterally abort.
  - Post unilaterally aborting, forward the decision to everyone.

- If a participant timeouts in Ready State (voted to commit) and it can determine that only the coordinator failed, it can still make progress.
  - Elect a new coordinator.

2PC Blocking

- If the communication is not centralized, then participants can communicate with each other and try to make progress.

- If a participant timeouts in Initial State (did not get a chance to vote) and has not received a global decision, it can unilaterally abort.
  - Post unilaterally aborting, forward the decision to everyone.

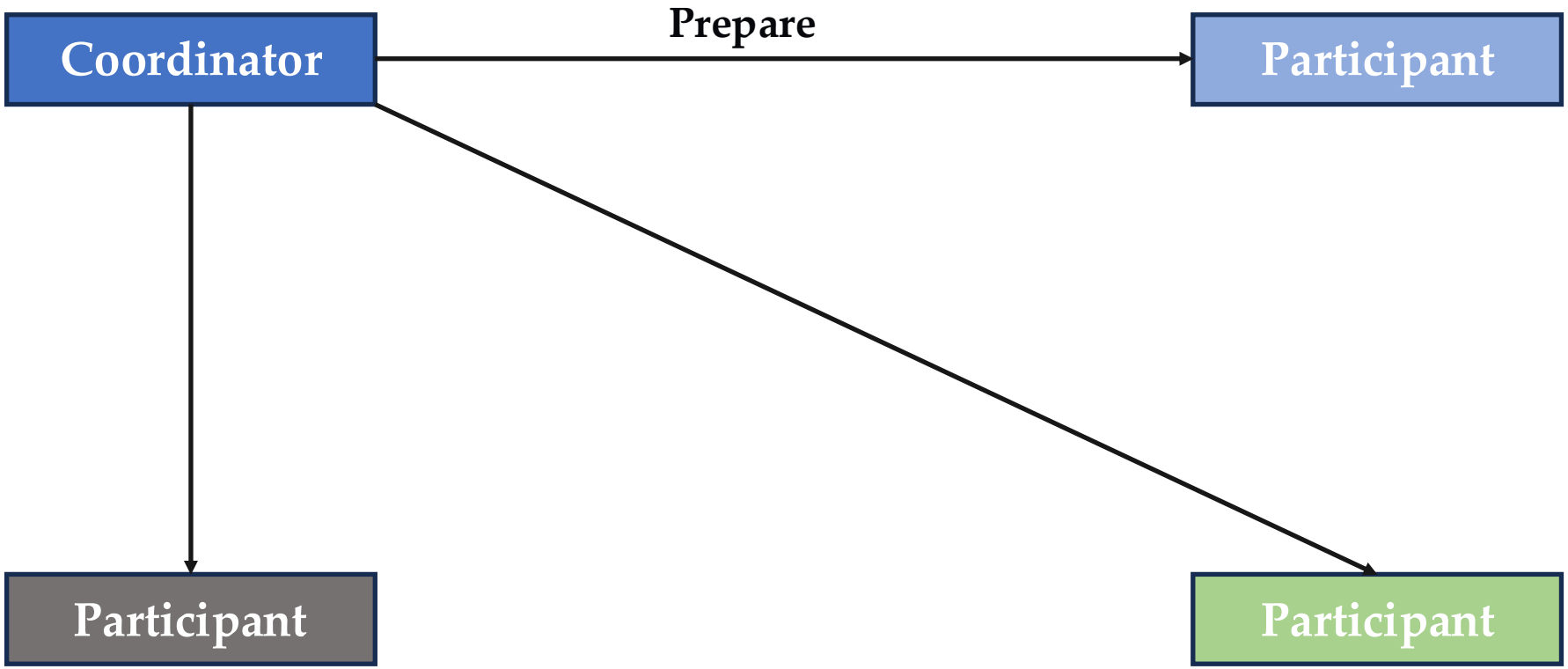
- If a participant timeouts in Ready State (voted to commit) and it can determine that only the coordinator failed, it can still make progress.
  - Elect a new coordinator.

  - New Coordinator collects all the votes and makes a decision.

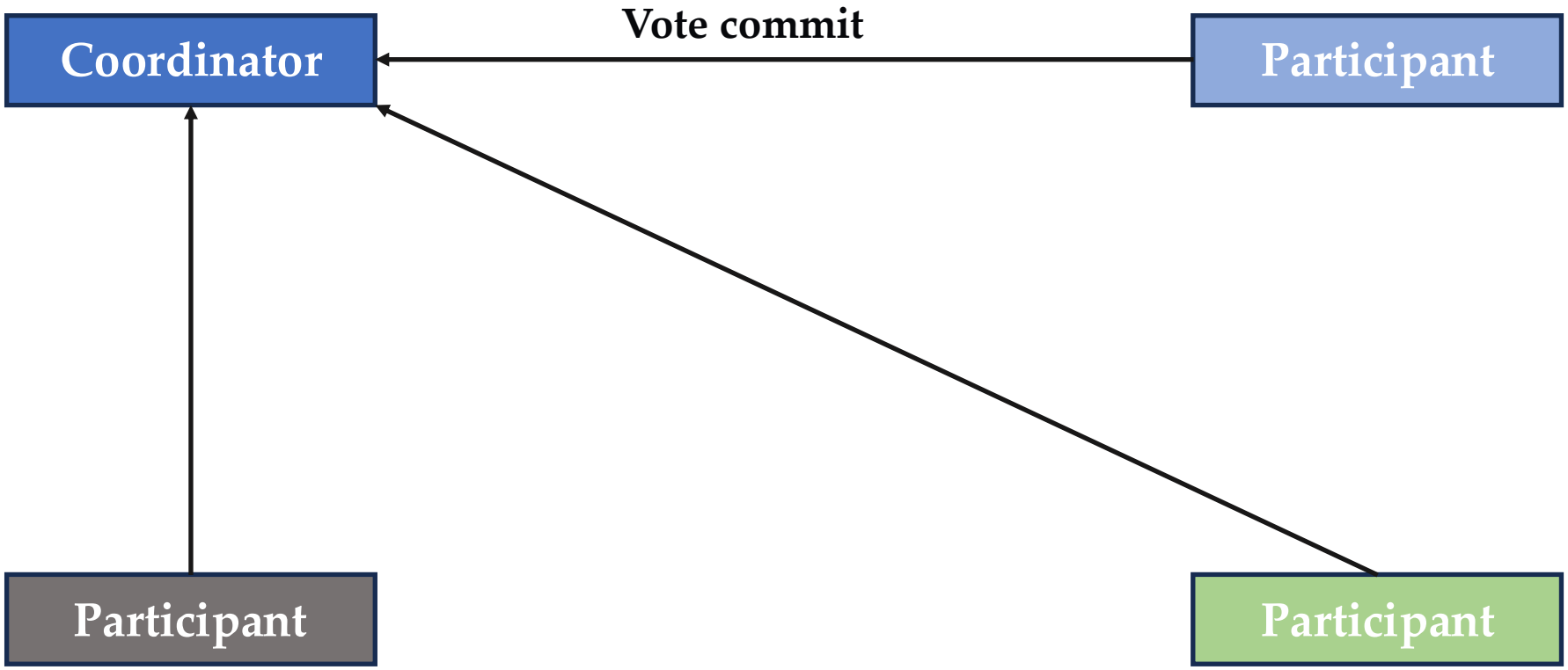
Any situation where 2PC always blocks?

Any situation where 2PC always blocks?

Any situation where 2PC always blocks?



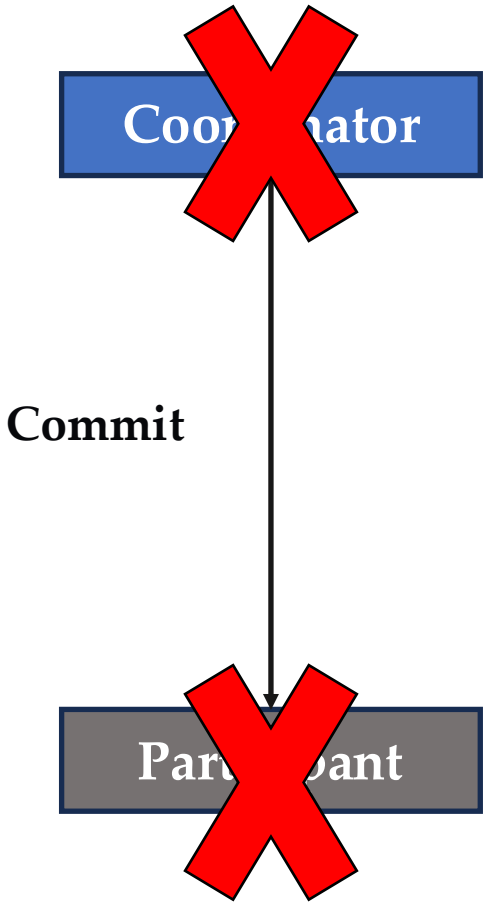
Any situation where 2PC always blocks?



Any situation where 2PC always blocks?

Any situation where 2PC always blocks?

Any situation where 2PC always blocks?



How should these participants make progress?

Any situation where 2PC always blocks?

- Thus, 2PC is termed as a blocking protocol!
- Is there any alternative?

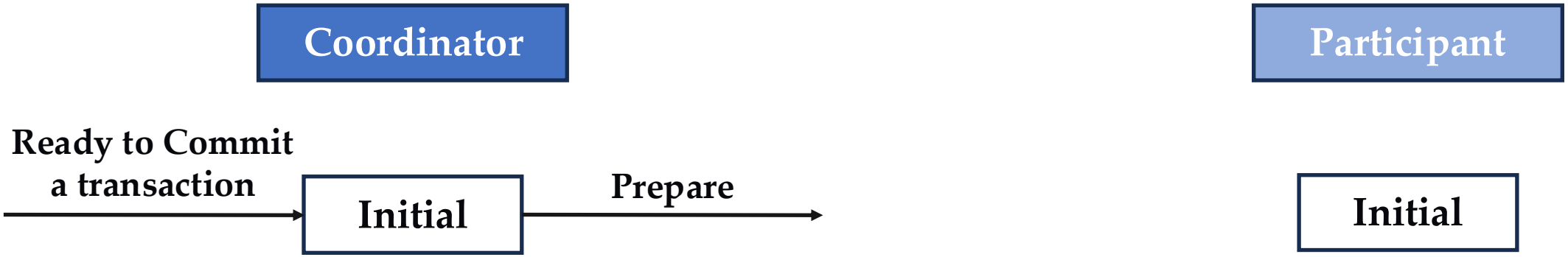
Three Phase Commit Protocol

- The Three Phase Commit (3PC) protocol is non-blocking under node failures.

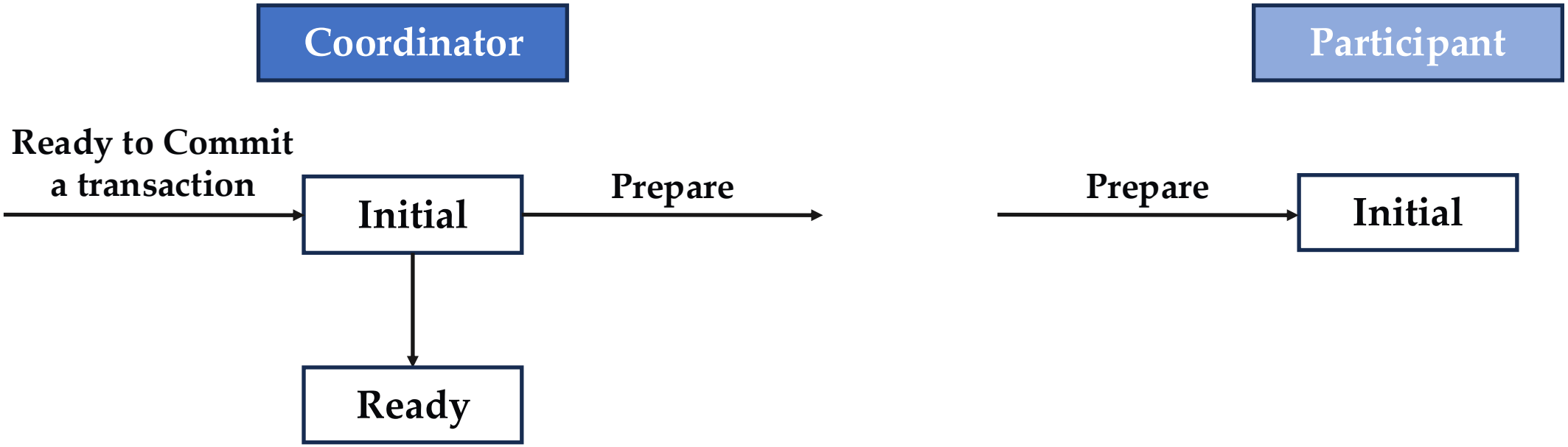
3PC Protocol Flow

3PC Protocol Flow

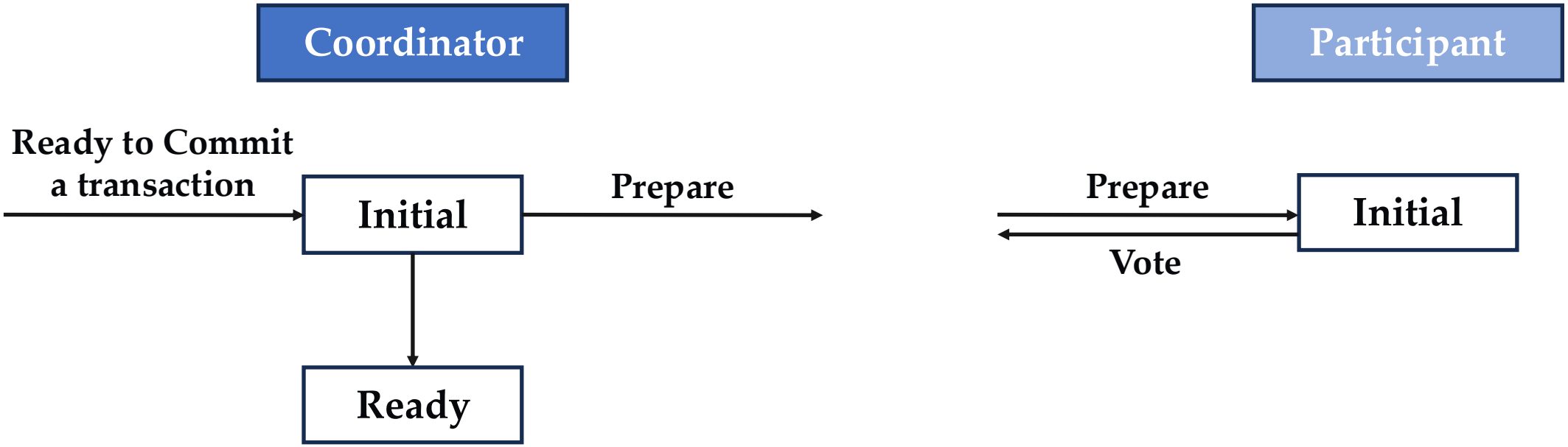
3PC Protocol Flow



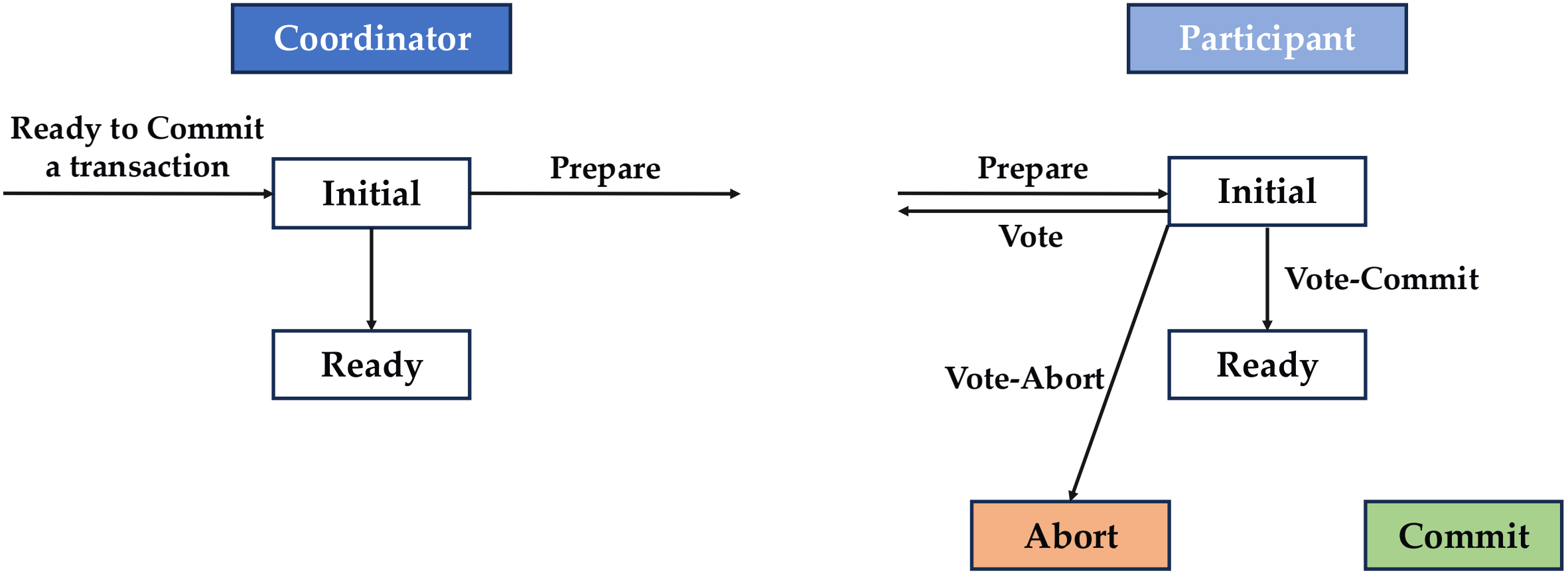
3PC Protocol Flow



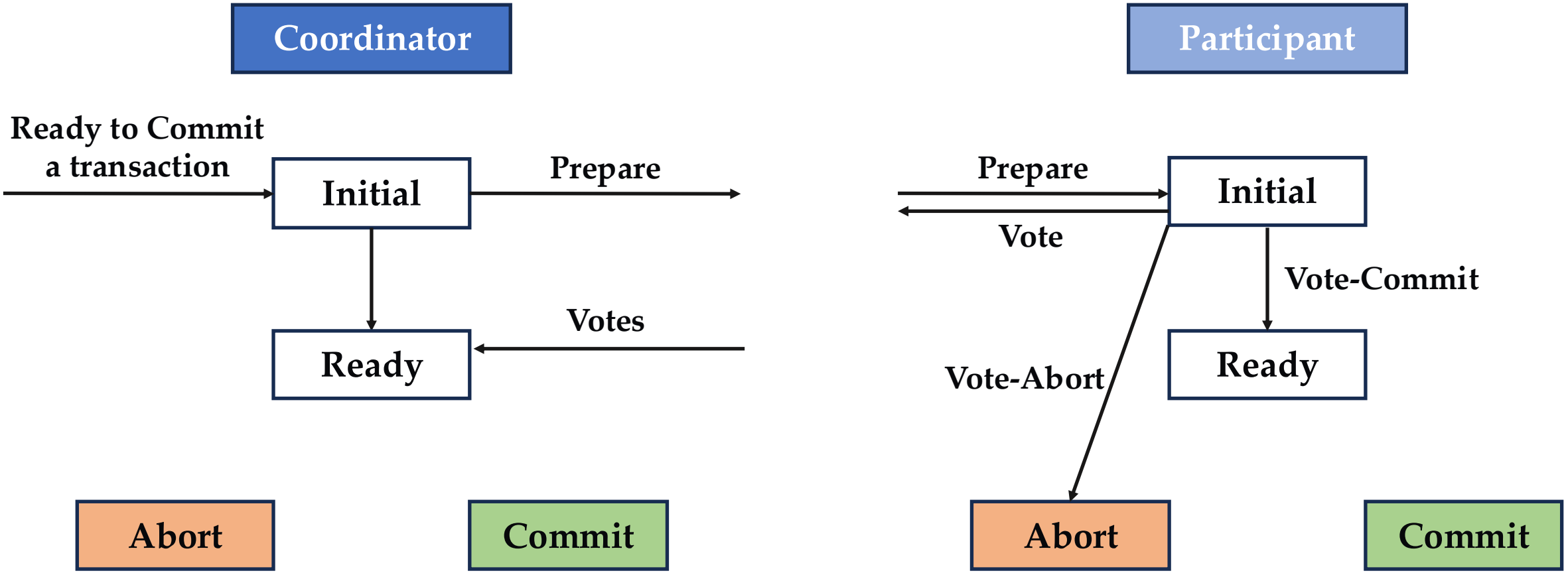
3PC Protocol Flow



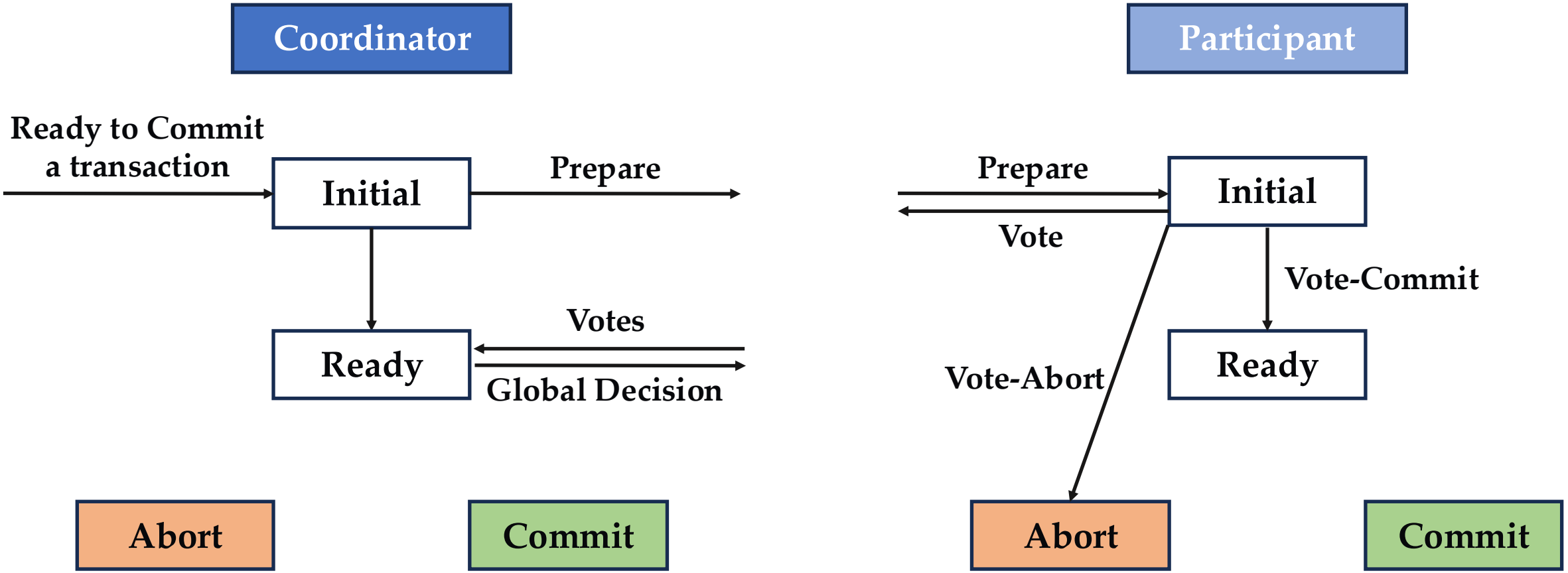
3PC Protocol Flow



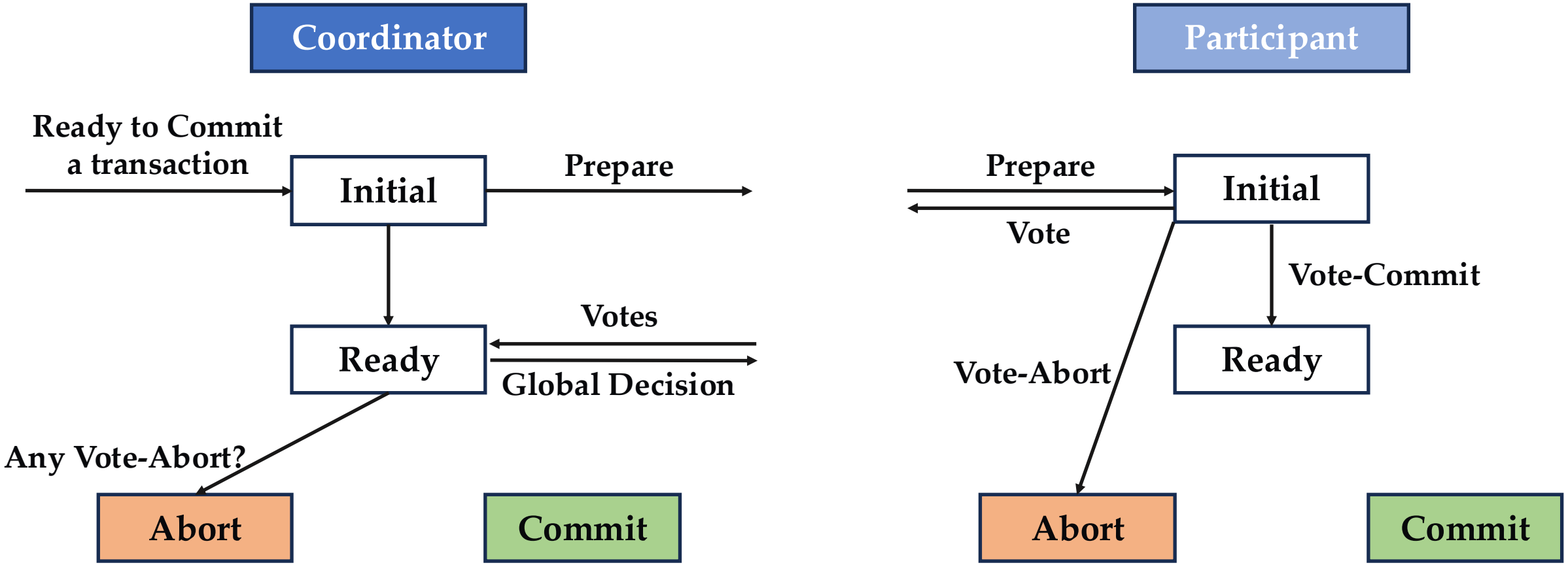
3PC Protocol Flow



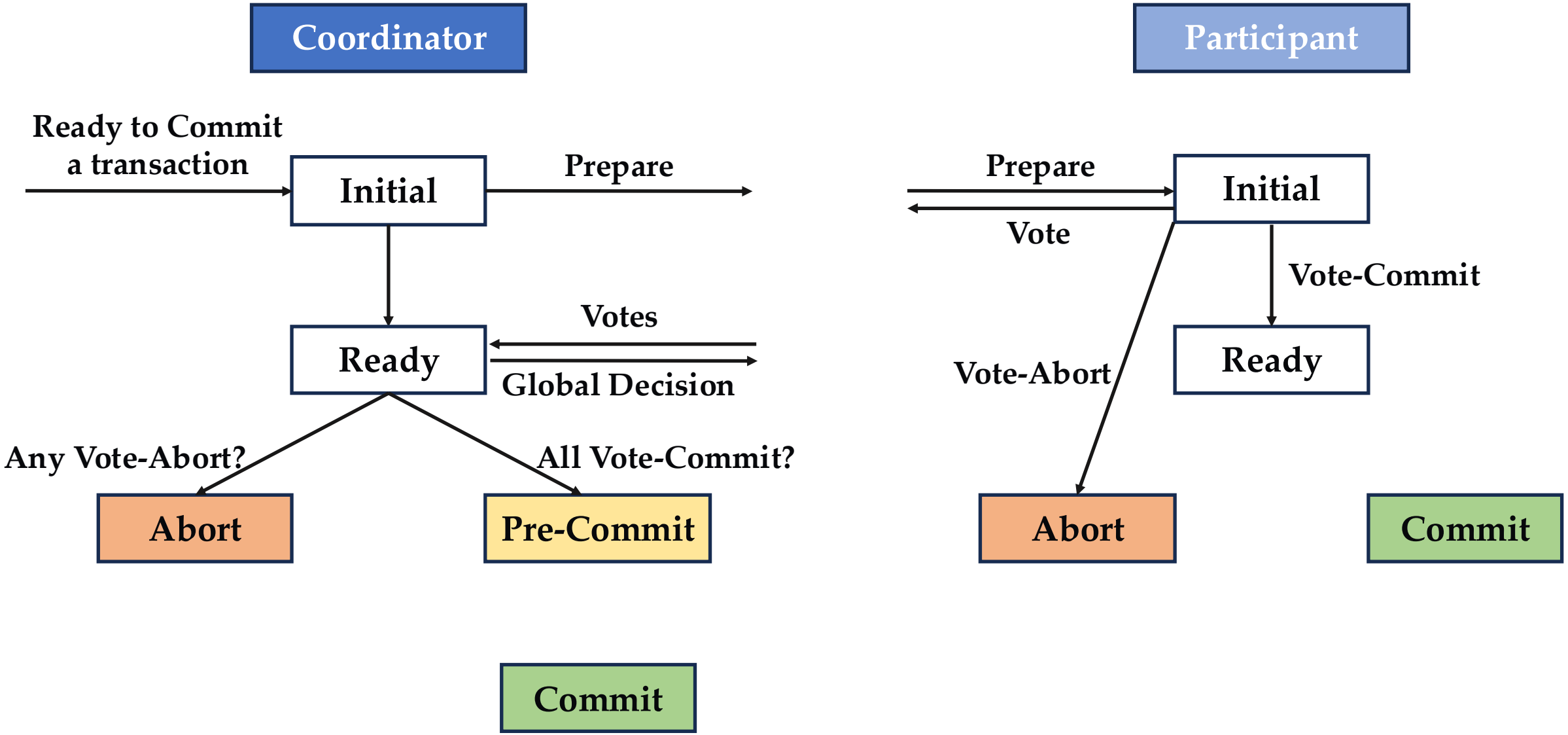
3PC Protocol Flow



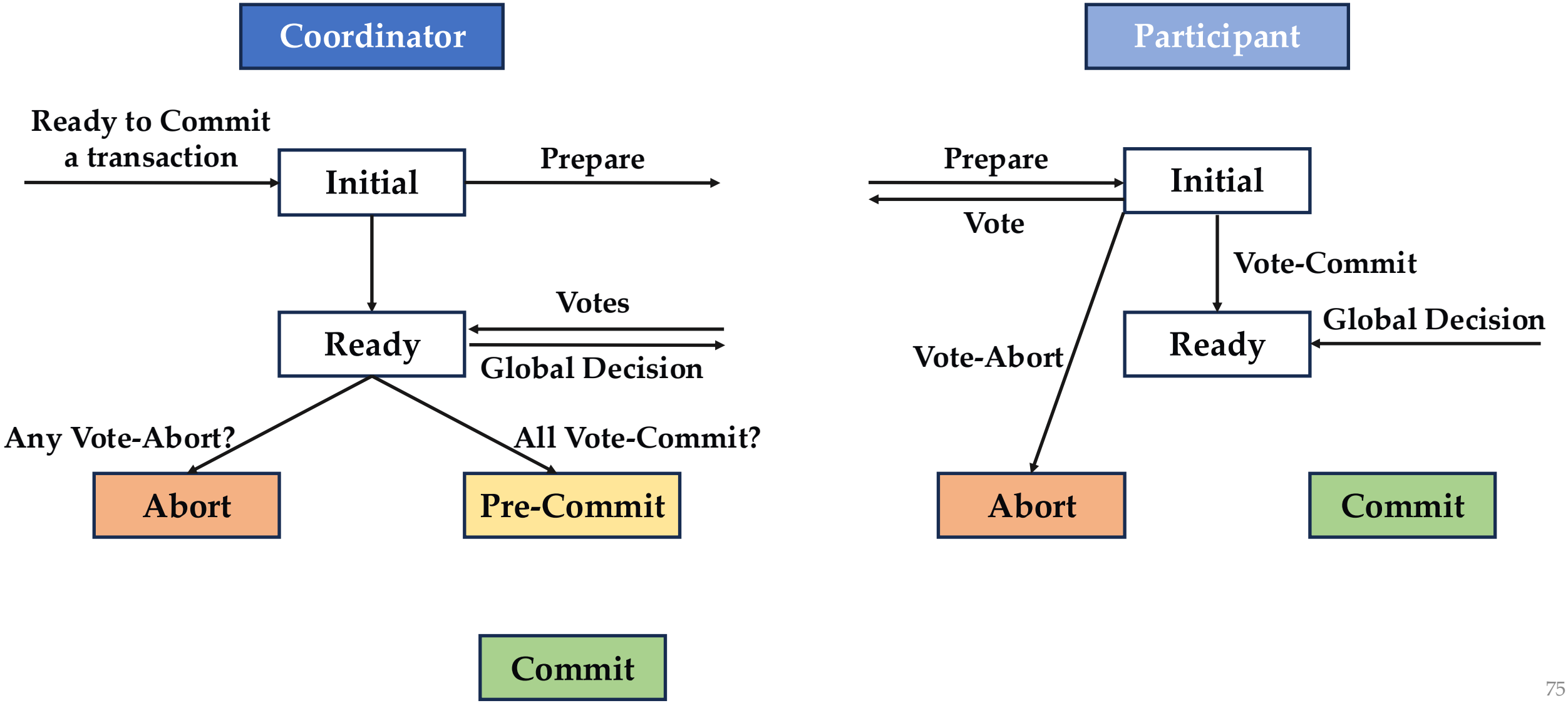
3PC Protocol Flow



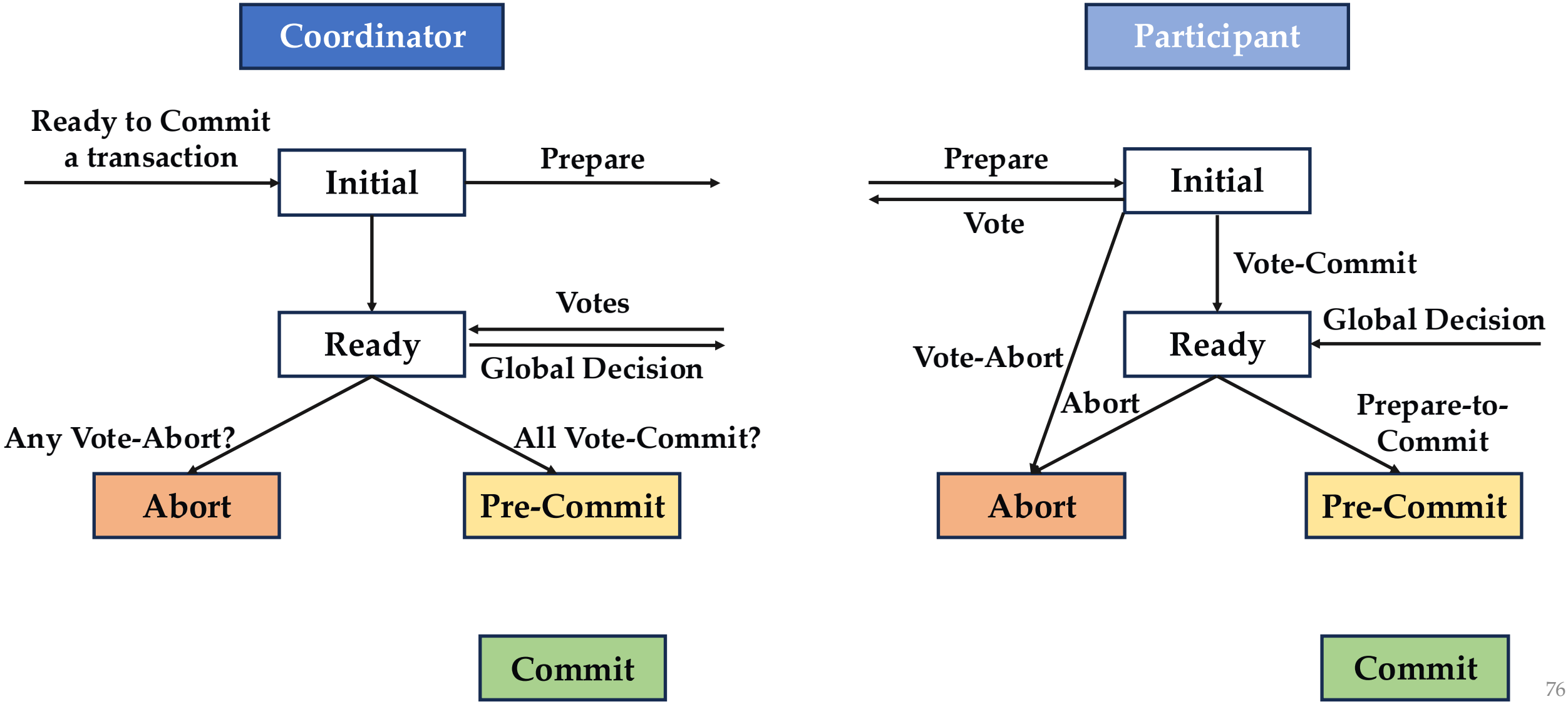
3PC Protocol Flow



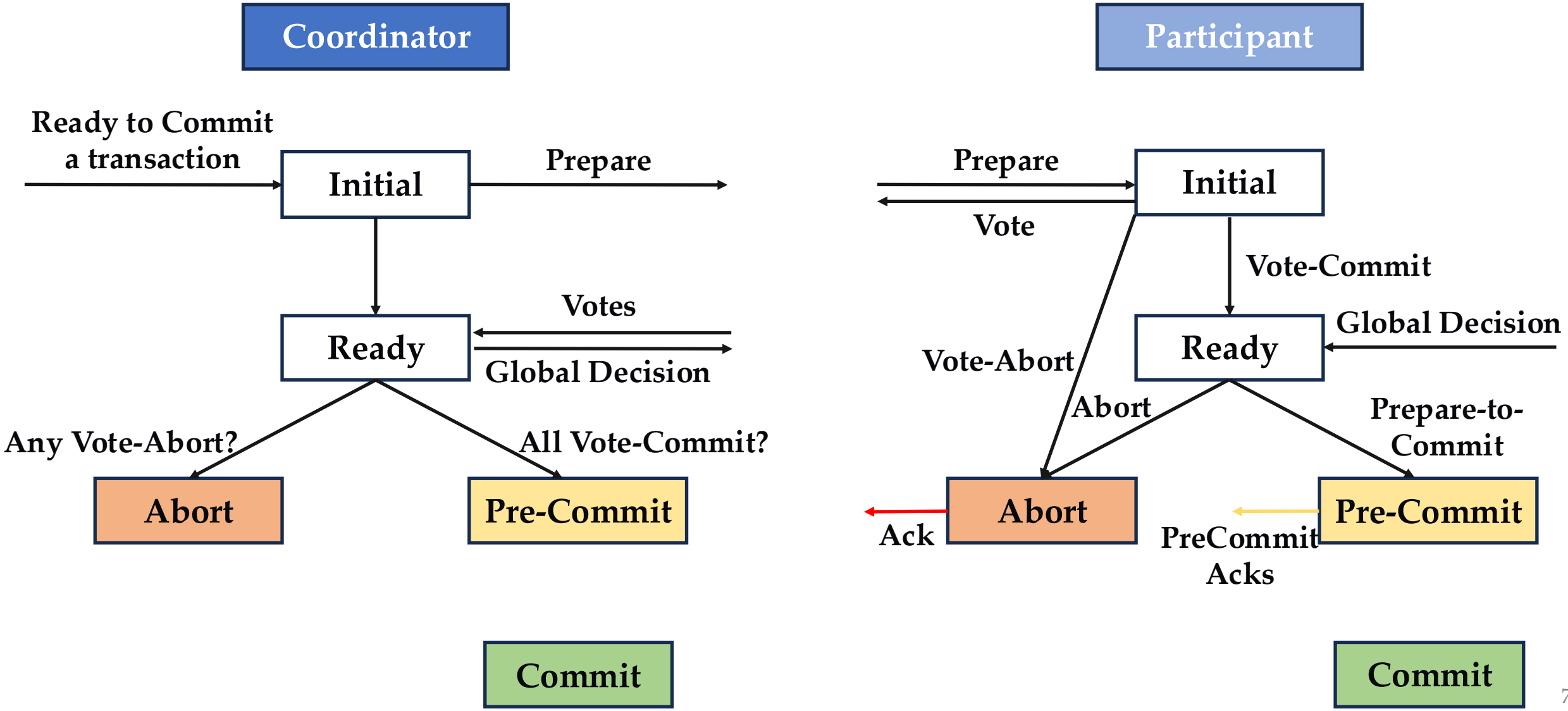
3PC Protocol Flow



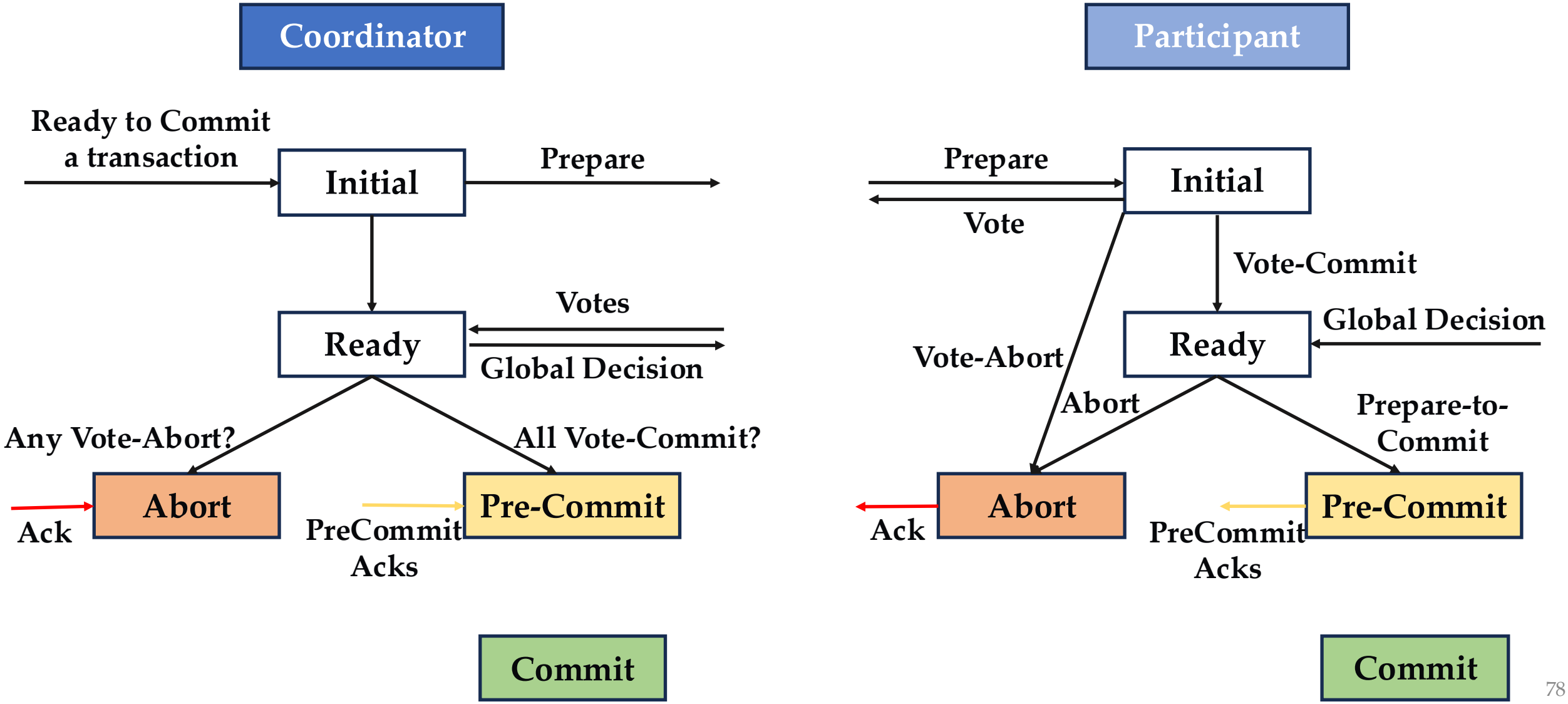
3PC Protocol Flow



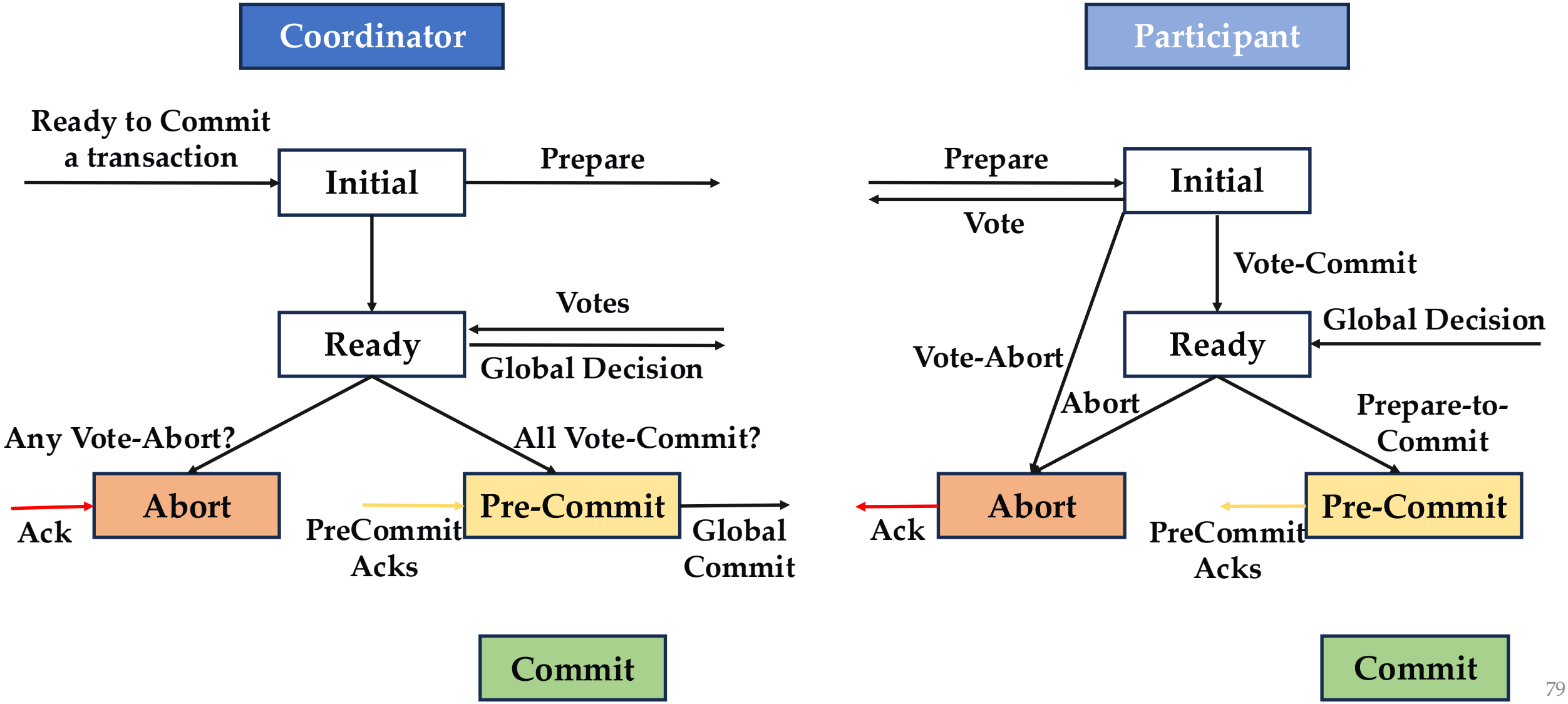
3PC Protocol Flow



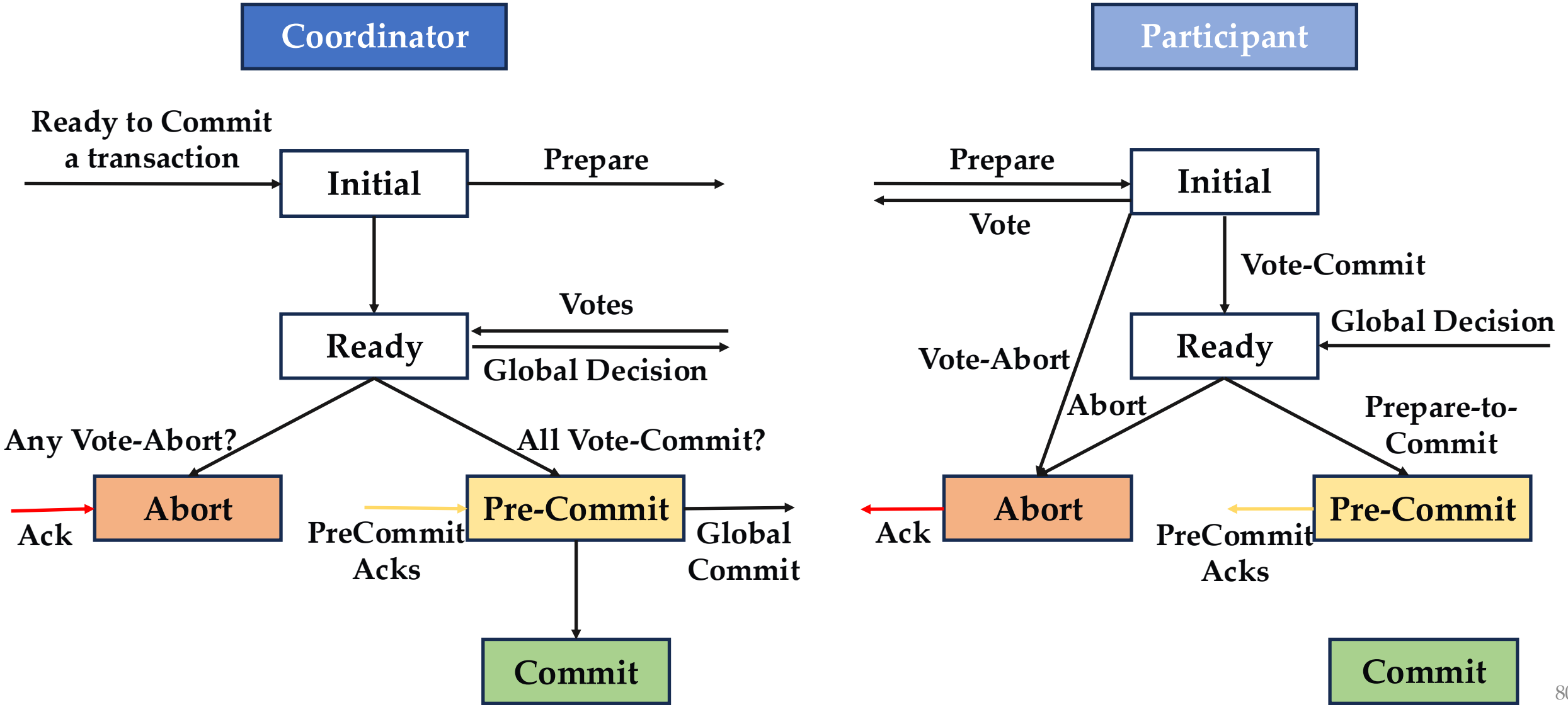
3PC Protocol Flow



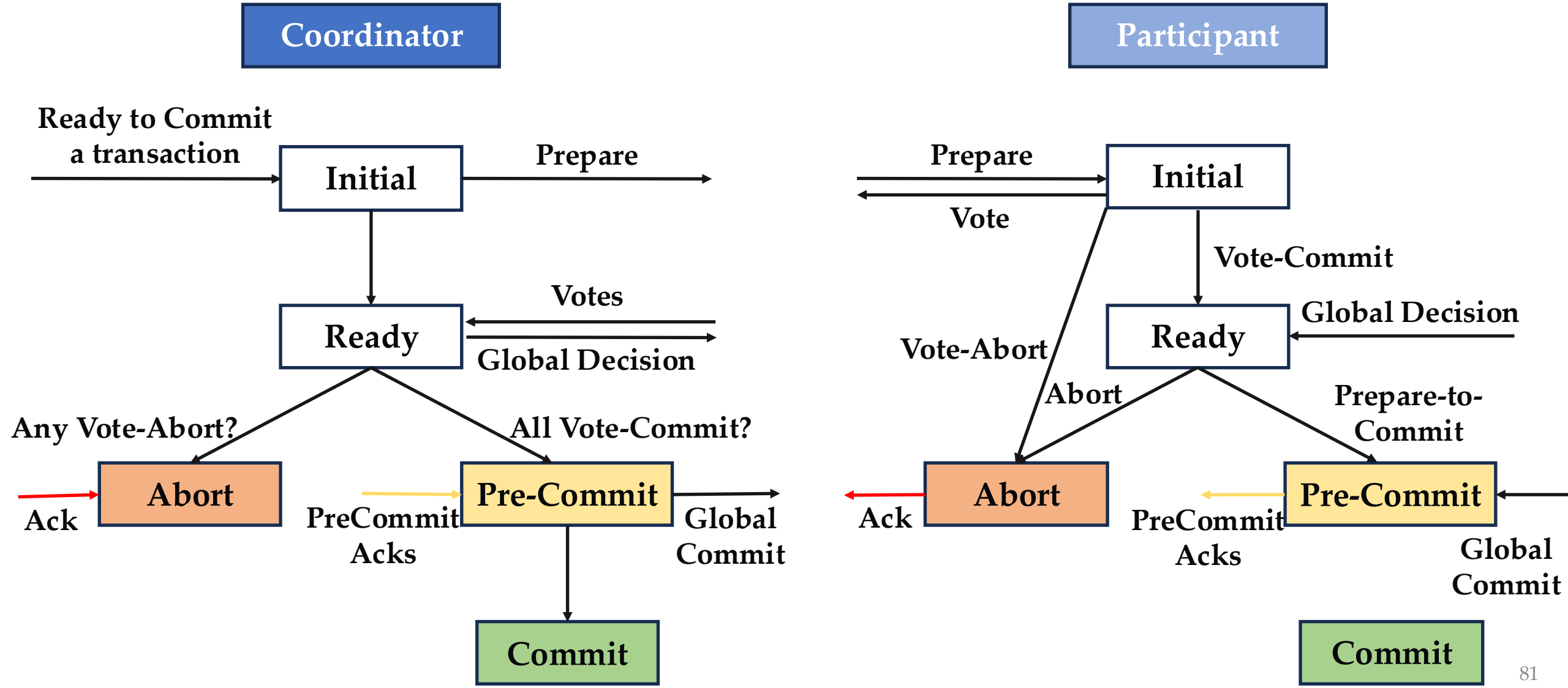
3PC Protocol Flow



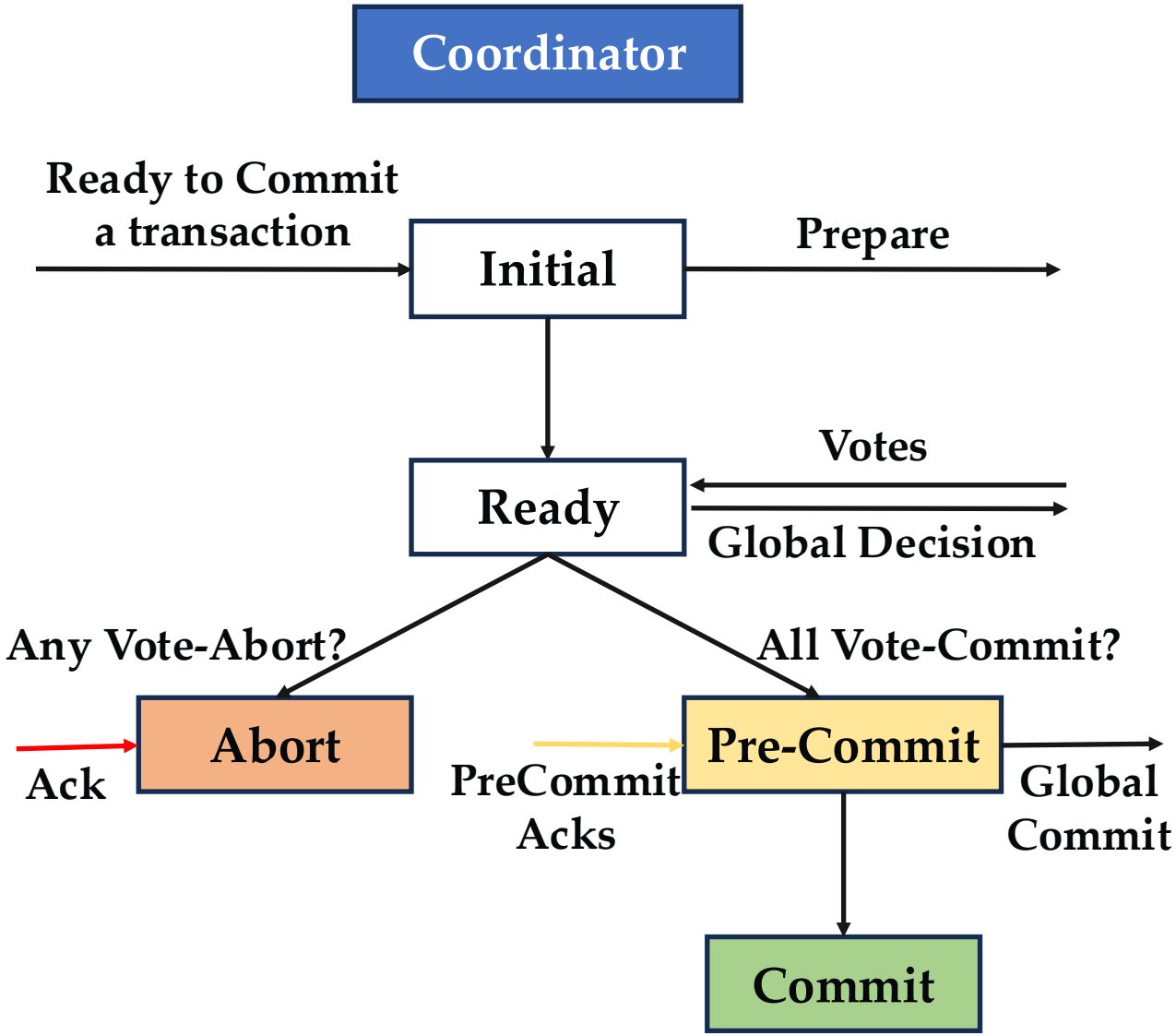
3PC Protocol Flow

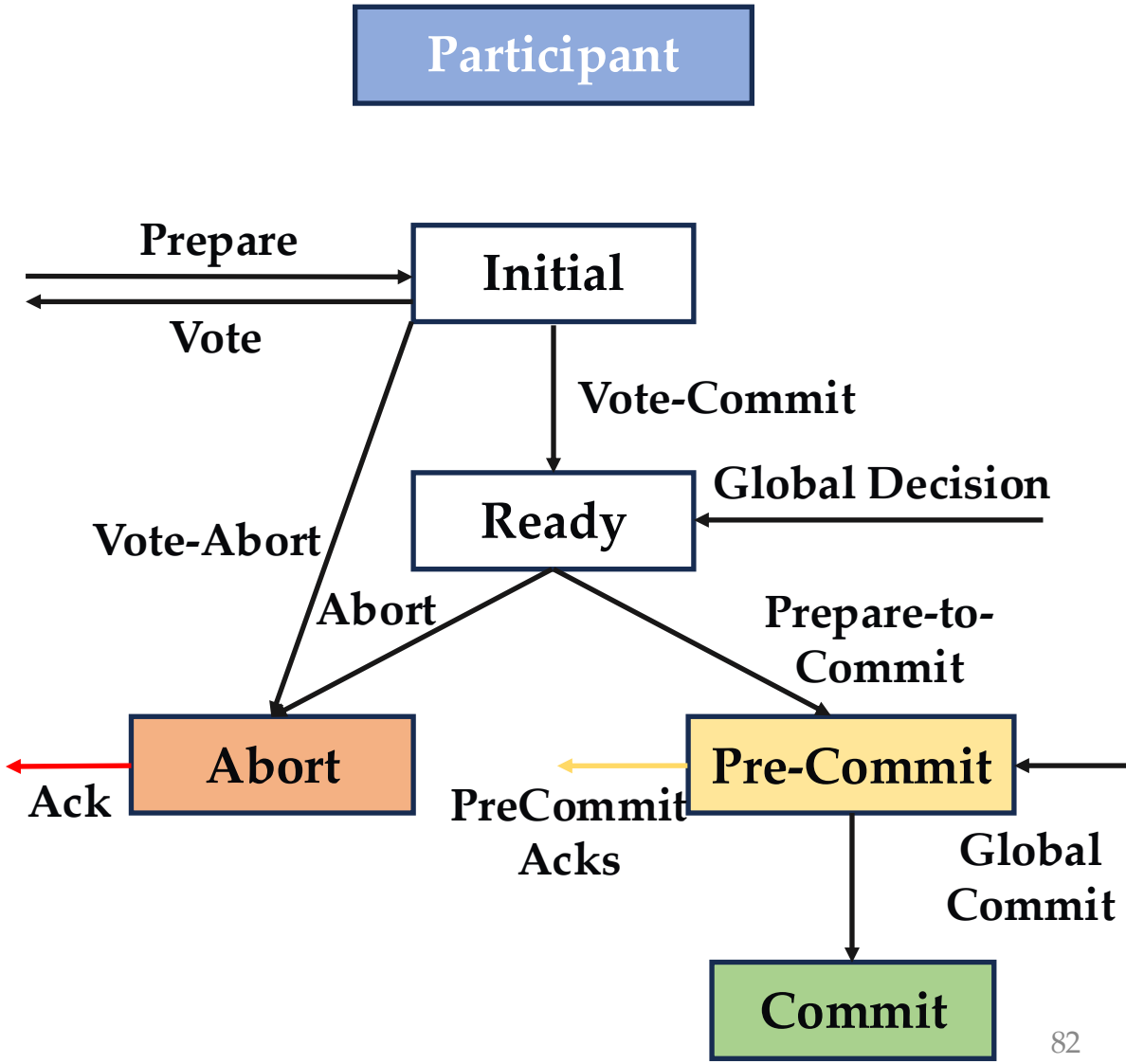


3PC Protocol Flow

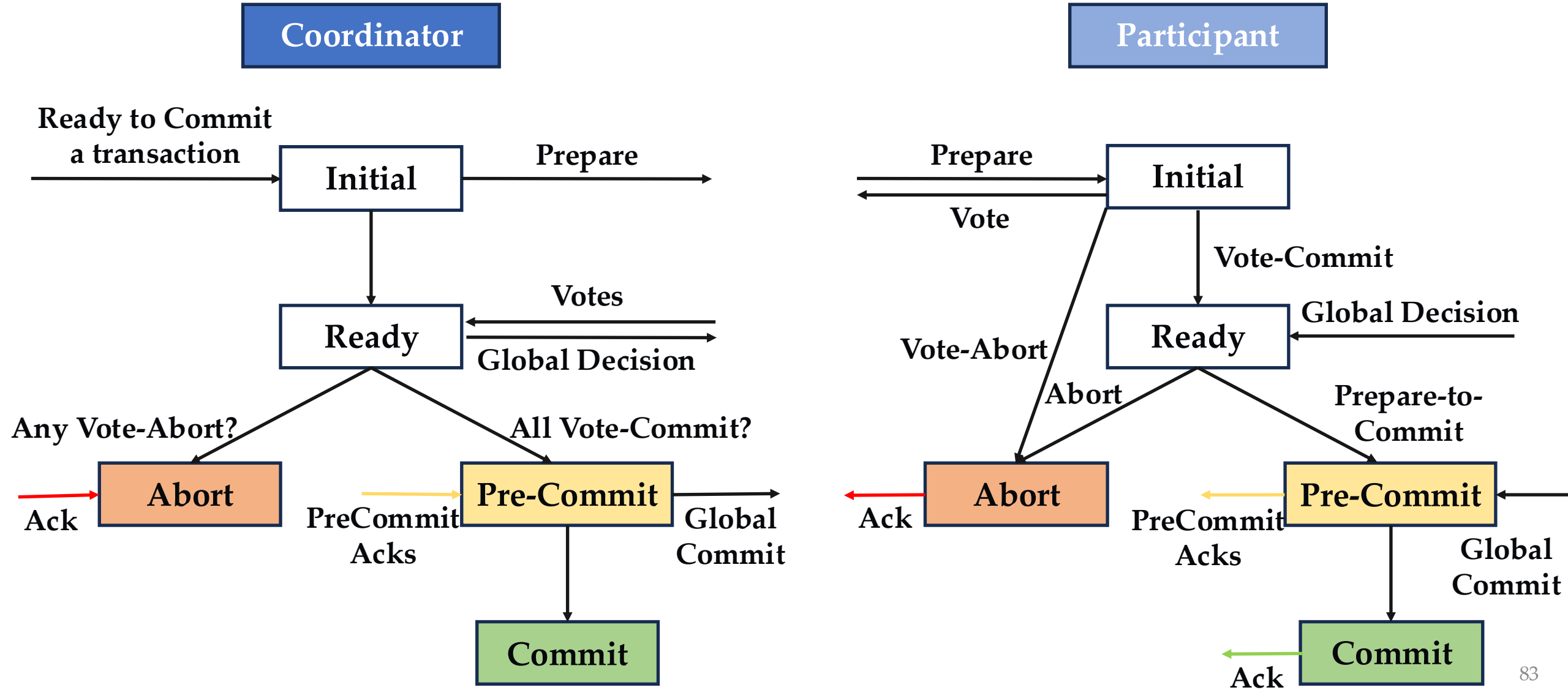


3PC Protocol Flow

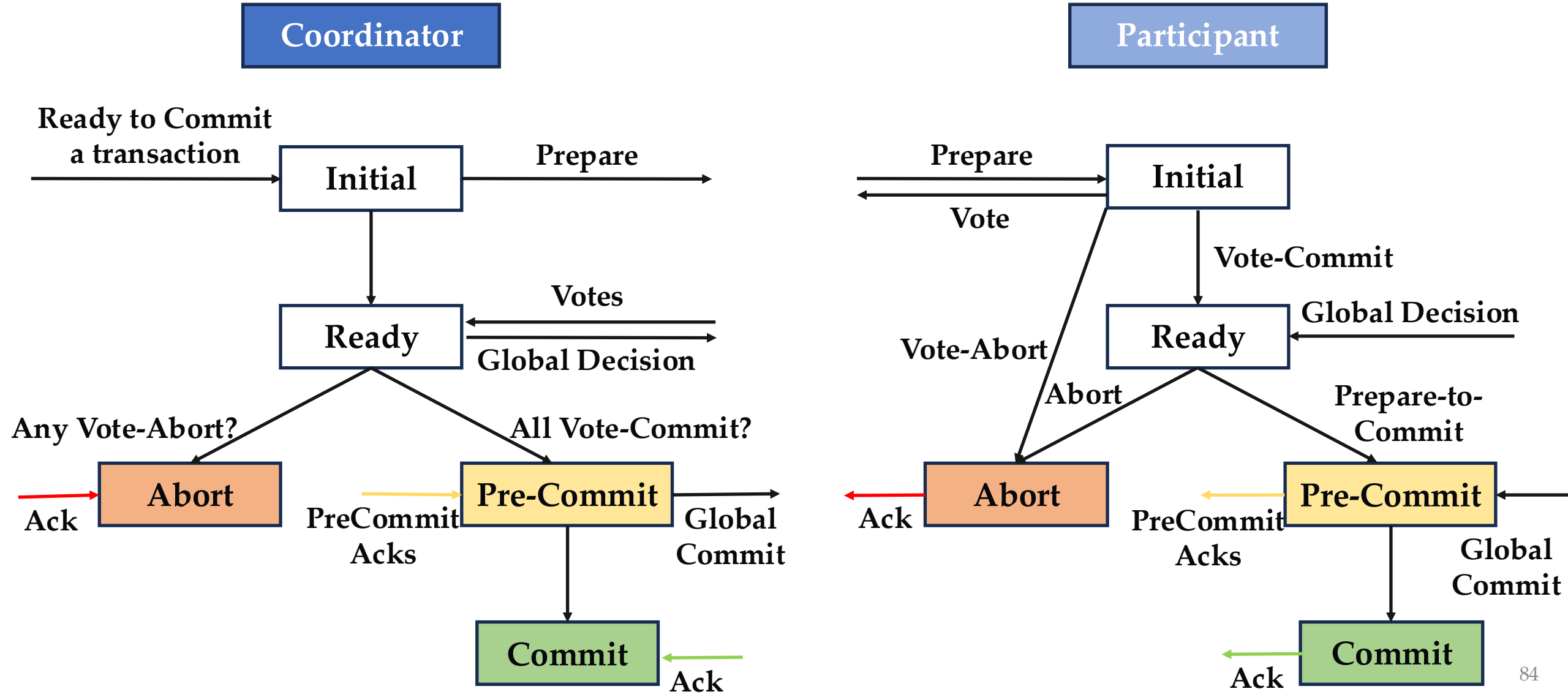




3PC Protocol Flow



3PC Protocol Flow



Three Phase Commit Protocol

- So, why is 3PC protocol non-blocking?

Three Phase Commit Protocol

- So, why is 3PC protocol non-blocking?
- **Two key properties** necessary for non-blocking behavior.

Three Phase Commit Protocol

- So, why is 3PC protocol non-blocking?
- **Two key properties** necessary for non-blocking behavior.

  1) No state is adjacent to both a commit and an abort state.

Three Phase Commit Protocol

- So, why is 3PC protocol non-blocking?
- **Two key properties** necessary for non-blocking behavior.

  1) No state is adjacent to both a commit and an abort state.

  2) No non-committable state is adjacent to a commit state.

Three Phase Commit Protocol

- So, why is 3PC protocol non-blocking?
- **Two key properties** necessary for non-blocking behavior.
  1) No state is adjacent to both a commit and an abort state.
  2) No non-committable state is adjacent to a commit state.
- Notice that 2PC **violates** these properties.

3PC Termination Protocol

- What changes do we need to do for the termination protocol of 3PC?

3PC Termination Protocol

- **Coordinator timeout in Pre-commit State**:
  - Coordinator timeouts in this state → Waiting for PreCommit acknowledgements from all the participants.

3PC Termination Protocol

- **Coordinator timeout in Pre-commit State**:
  - Coordinator timeouts in this state → Waiting for PreCommit acknowledgements from all the participants.

  - Coordinator knows that all the participants are either in Ready or PreCommit state.

3PC Termination Protocol

- **Coordinator timeout in Pre-commit State**:
  - Coordinator timeouts in this state → Waiting for PreCommit acknowledgements from all the participants.
  - Coordinator knows that all the participants are either in Ready or PreCommit state.
  - Simply re-sends Global decision and Global Commit messages.

3PC Termination Protocol

- **Participant timeout in Ready or Pre-commit State**:
  - Attempt to elect a new coordinator.

3PC Termination Protocol

- **Participant timeout in Ready or Pre-commit State**:
  - Attempt to elect a new coordinator.
  - The new coordinator can be Initial, Ready, Commit or Abort.

3PC Termination Protocol

- **Participant timeout in Ready or Pre-commit State**:
  - Attempt to elect a new coordinator.
  - The new coordinator can be Initial, Ready, Commit or Abort.
  - The new coordinator in Ready state → Abort's the transaction.
    - Some other participant many have to **transition from Precommit to Abort.**

3PC Termination Protocol

- **Participant timeout in Ready or Pre-commit State**:
  - Attempt to elect a new coordinator.
  - The new coordinator can be Initial, Ready, Commit or Abort.
  - The new coordinator in Ready state → Abort's the transaction.
    - Some other participant many have to **transition from Precommit to Abort**.
  - The new coordinator in Precommit state → Globally commits the transaction.

3PC Termination Protocol

- **Participant timeout in Ready or Pre-commit State**:
  - Attempt to elect a new coordinator.
  - The new coordinator can be Initial, Ready, Commit or Abort.
  - The new coordinator in Ready state → Abort's the transaction.
    - Some other participant many have to **transition from Precommit to Abort**.
  - The new coordinator in Precommit state → Globally commits the transaction.
  - The new coordinator in Abort state → Globally aborts the transaction.