CS4224/CS5424 Lecture 5
Distributed Commit Protocols

Transaction Management

- ACID properties
  1. **Atomicity**: Either all or none of the actions in Xact happen
  2. **Consistency**: If each Xact is consistent and the DB starts consistent, it ends up consistent
  3. **Isolation**: Execution of one Xact is isolated from other Xacts
  4. **Durability**: If a Xact commits, its effects persist
- The concurrency control manager component ensures isolation
- The recovery manager component ensures atomicity and durability

Recovery in Centralized DBMS

- **Recovery manager** guarantees atomicity and durability properties of Xacts
  - Undo: remove effects of aborted Xact to preserve atomicity
  - Redo: re-install effects of committed Xact for durability
- Processes three operations:
  - Commit(T) - installs T's updated pages into stable database
  - Abort(T) - restores all data that T updated to their prior values
  - Restart - recovers database to a consistent state from system failure
    - ★ aborts all active Xacts at the time of system failure
    - ★ installs updates of all committed Xacts that were not installed in the stable database before the failure

Log-based Database Recovery

- Log (aka trail/journal): history of actions executed by DBMS
  - Contains a log record for each write, commit, & abort
- Store as a sequential file of records in stable storage (i.e., non-volatile storage)

Implementing Abort

- Undo all updates by Xact to database pages
- Write-ahead logging (WAL) protocol
  Do not flush an uncommitted update to the stable database until the log record containing its before-image has been flushed to the log
- How to undo all the updates of Xact?
  - For each log record of Xact in reverse order, restore log record's before-image

Implementing Commit

- Need to ensure that that all the updates of Xact must be in stable storage (stable database or log) before Commit
- Force-at-commit protocol
  Do not commit a Xact until the after-images of all its updated pages are in stable storage (stable database or log)
- How to enforce force-at-commit protocol?
  - Writes commit log record for Xact
  - Flushes log

Implementing Restart

- Consists of redo phase & undo phase
- Redo phase:
  - Scans log records in forward direction to redo updates
  - Keeps track of active Xacts
- Undo phase: aborts all active Xacts

Distributed Transactions

- Transaction originating site - site where Xact is initiated
- Transaction coordinator (TC) - transaction manager (TM) at originating site
  - TC also referred to as coordinator
- TC of a distributed Xact *T* coordinates with other TMs to execute *T* at multiple sites
  - Other TMs are referred to as participants
- We use TM to refer to a coordinator or participant
- **Commit protocol** ensures **atomicity** of distributed transactions

Distributed Transactions (cont.)

- **Example**: Consider the following DDBMS

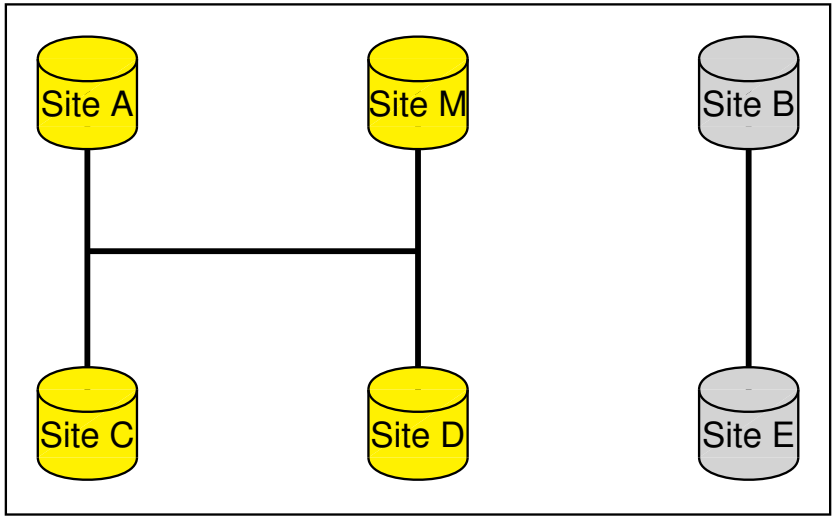
  Site A = $\{S_1\}$
  Site B = $\{R_1, S_2\}$
  Site C = $\{R_2, S_3\}$
  Site D = $\{R_3\}$

- Suppose Transaction T_1 is submitted at Site A:
  UPDATE R SET balance = balance * 1.01;
- TM at Site A is the transaction coordinator for T_1
- T_1 is executed as three local transactions:

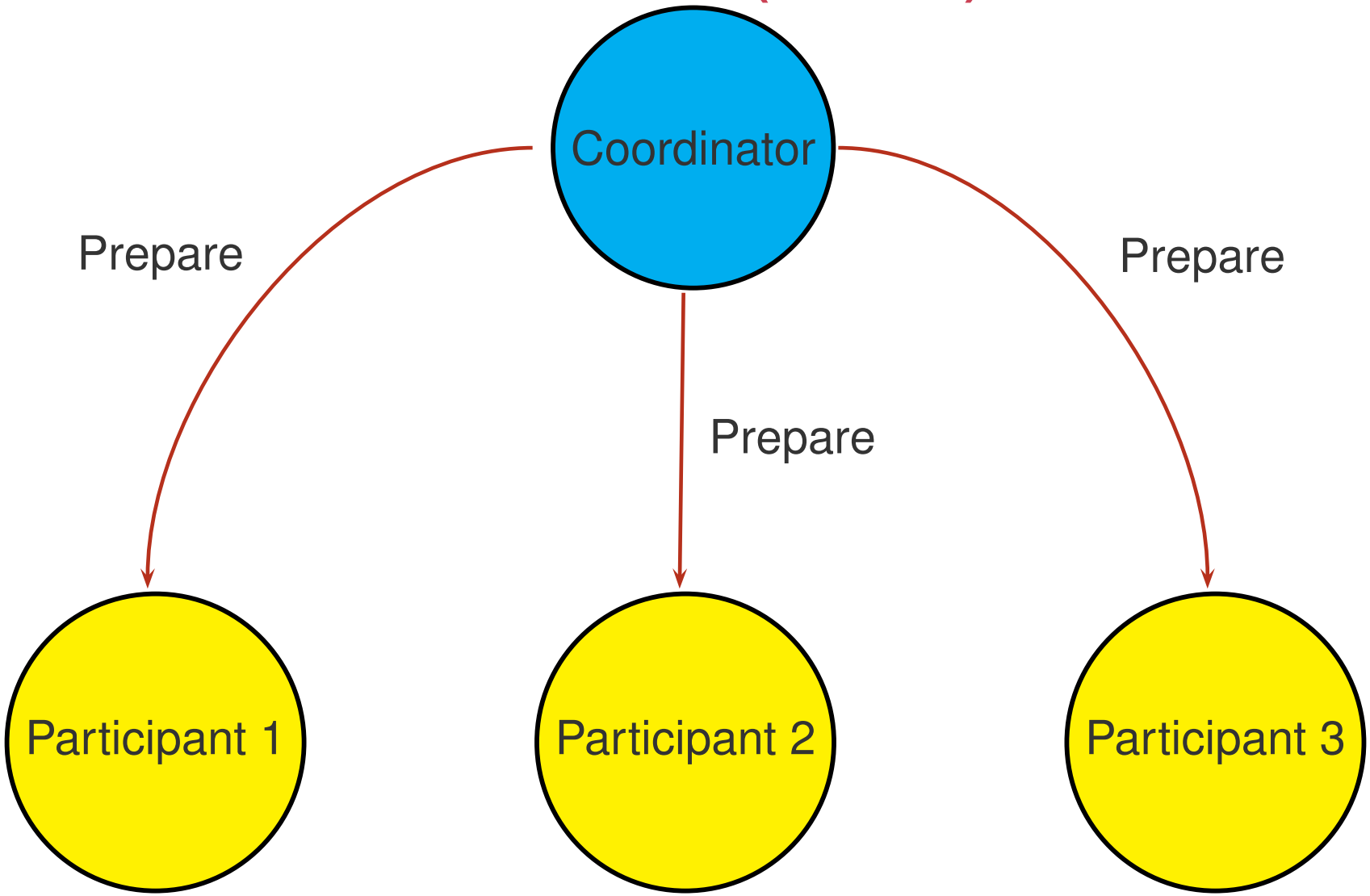
  $T_{1,B}$: UPDATE R_1 SET balance = balance * 1.01;
  $T_{1,C}$: UPDATE R_2 SET balance = balance * 1.01;
  $T_{1,D}$: UPDATE R_3 SET balance = balance * 1.01;

Failures in Distributed DBMS

- **Site failures**
  - Fail-stop model: A site is either working correctly (i.e. operational) or not working at all (i.e. failed)
  - Partial site failure: Some site(s) are operational & some site(s) are down
  - Total site failure: All sites are down
- **Communication failures**
  - Lost messages, network partitioning, etc.

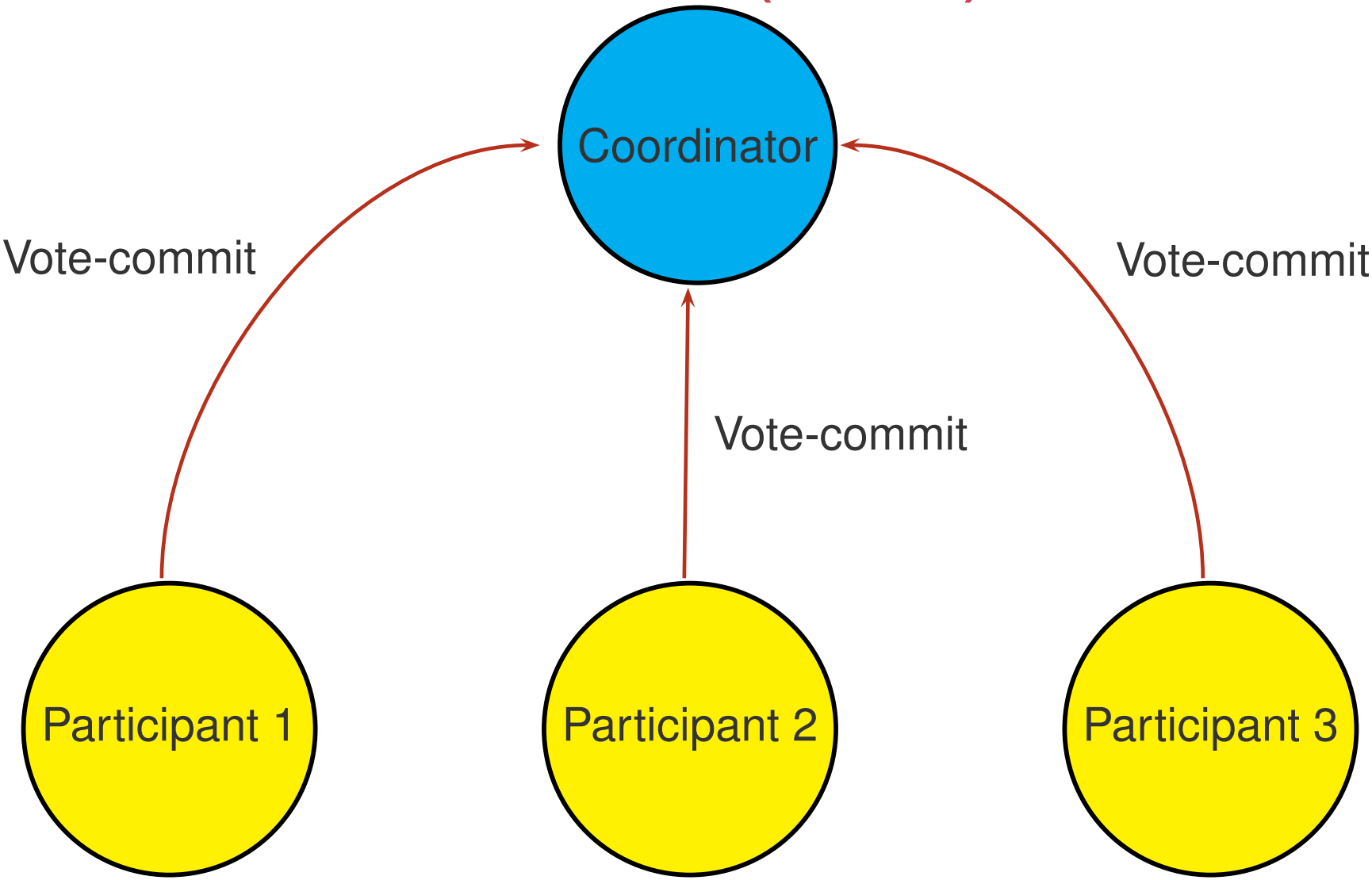


Two-Phase Commit (2PC) Protocol



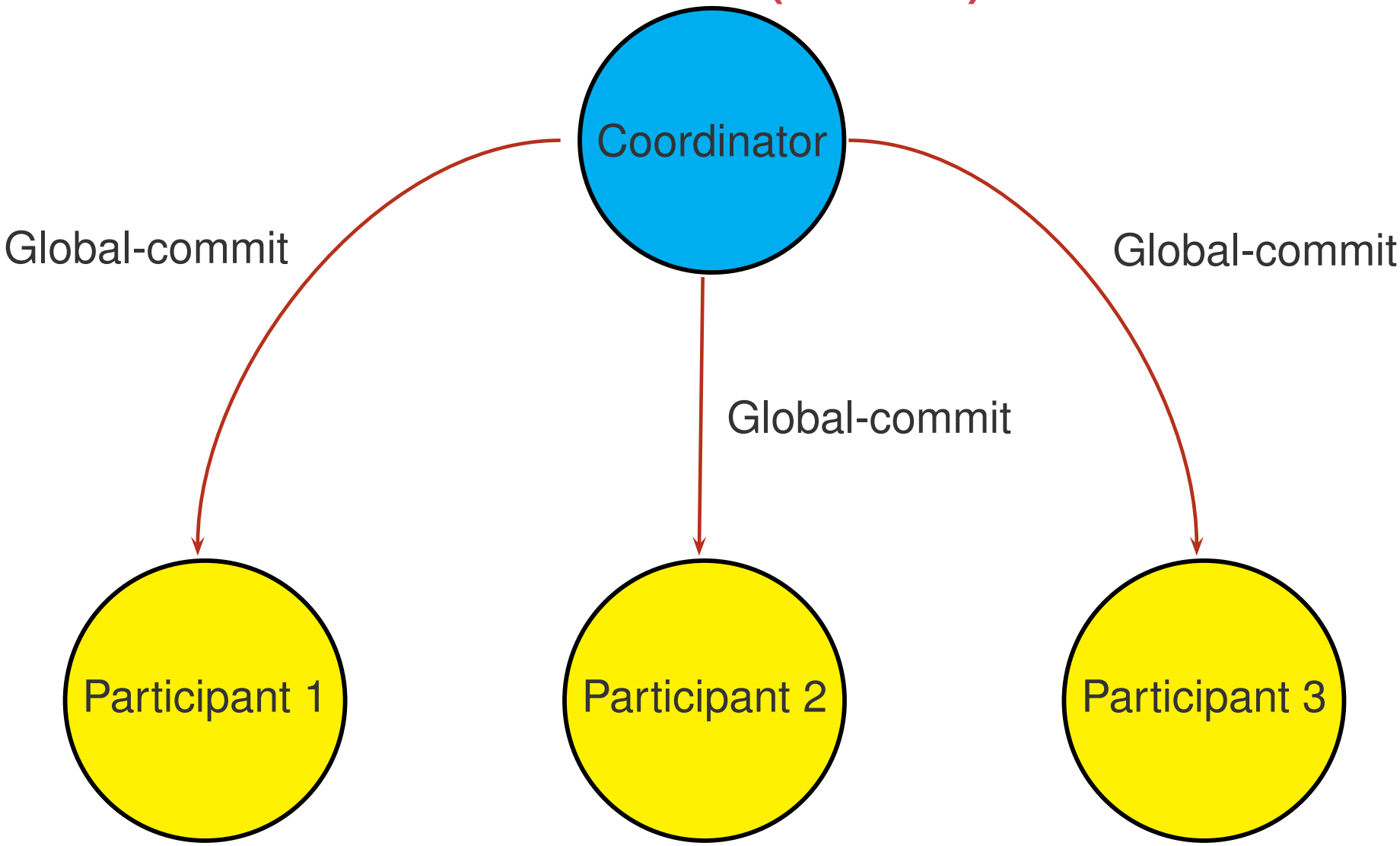
Phase 1: Voting Phase

Two-Phase Commit (2PC) Protocol



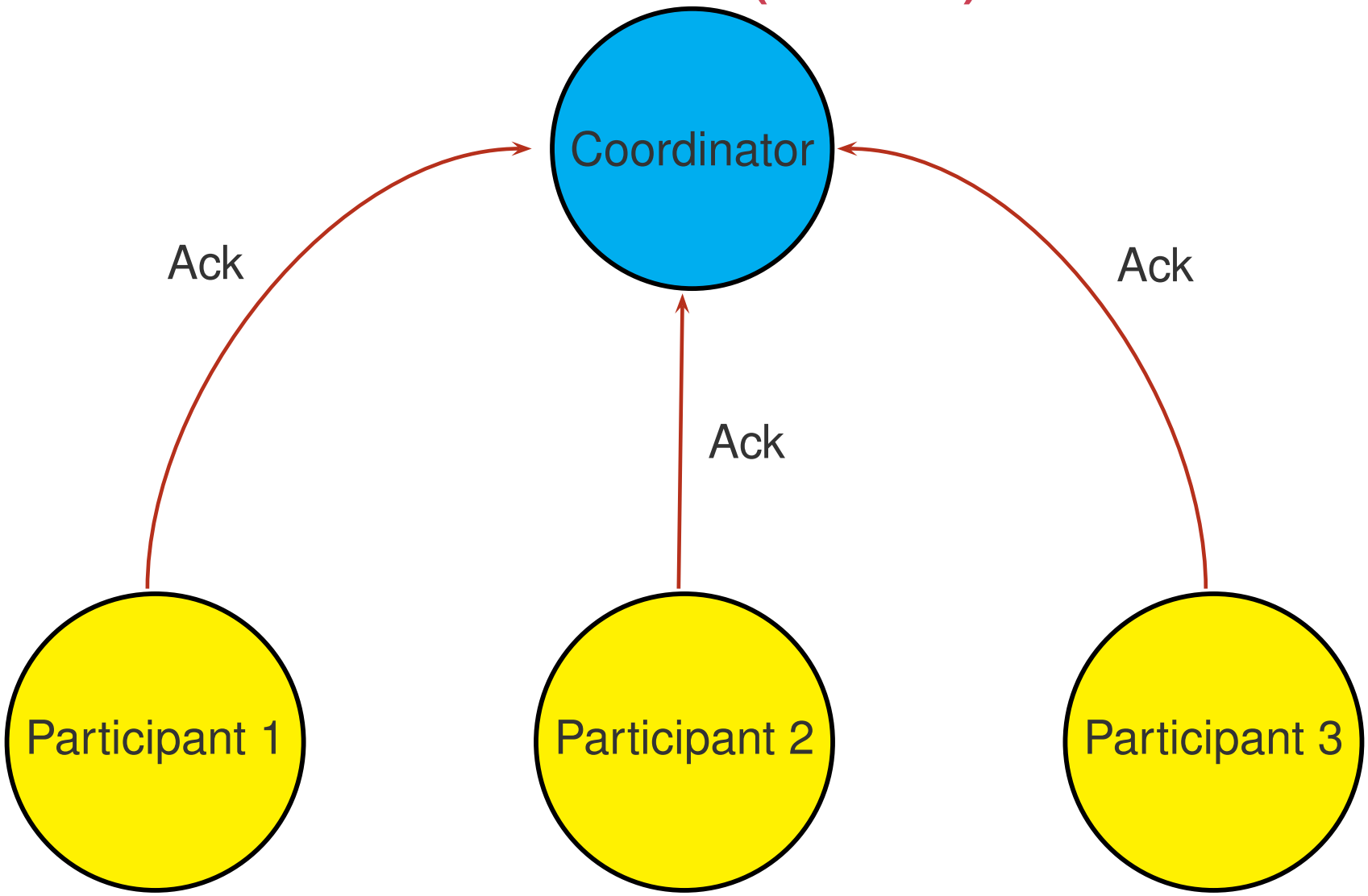
Phase 1: Voting Phase

Two-Phase Commit (2PC) Protocol



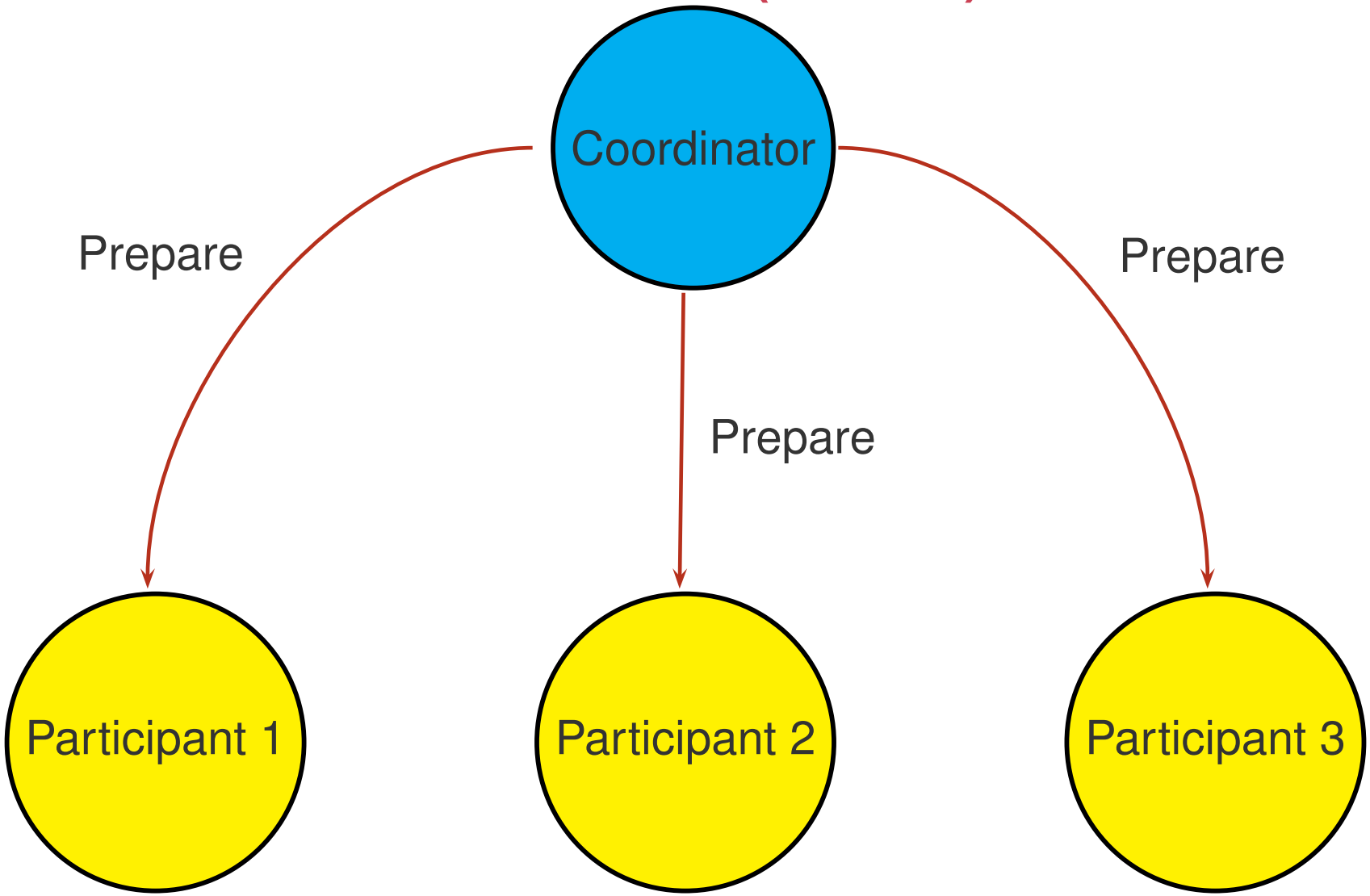
Phase 2: Decision Phase (commit scenario)

Two-Phase Commit (2PC) Protocol



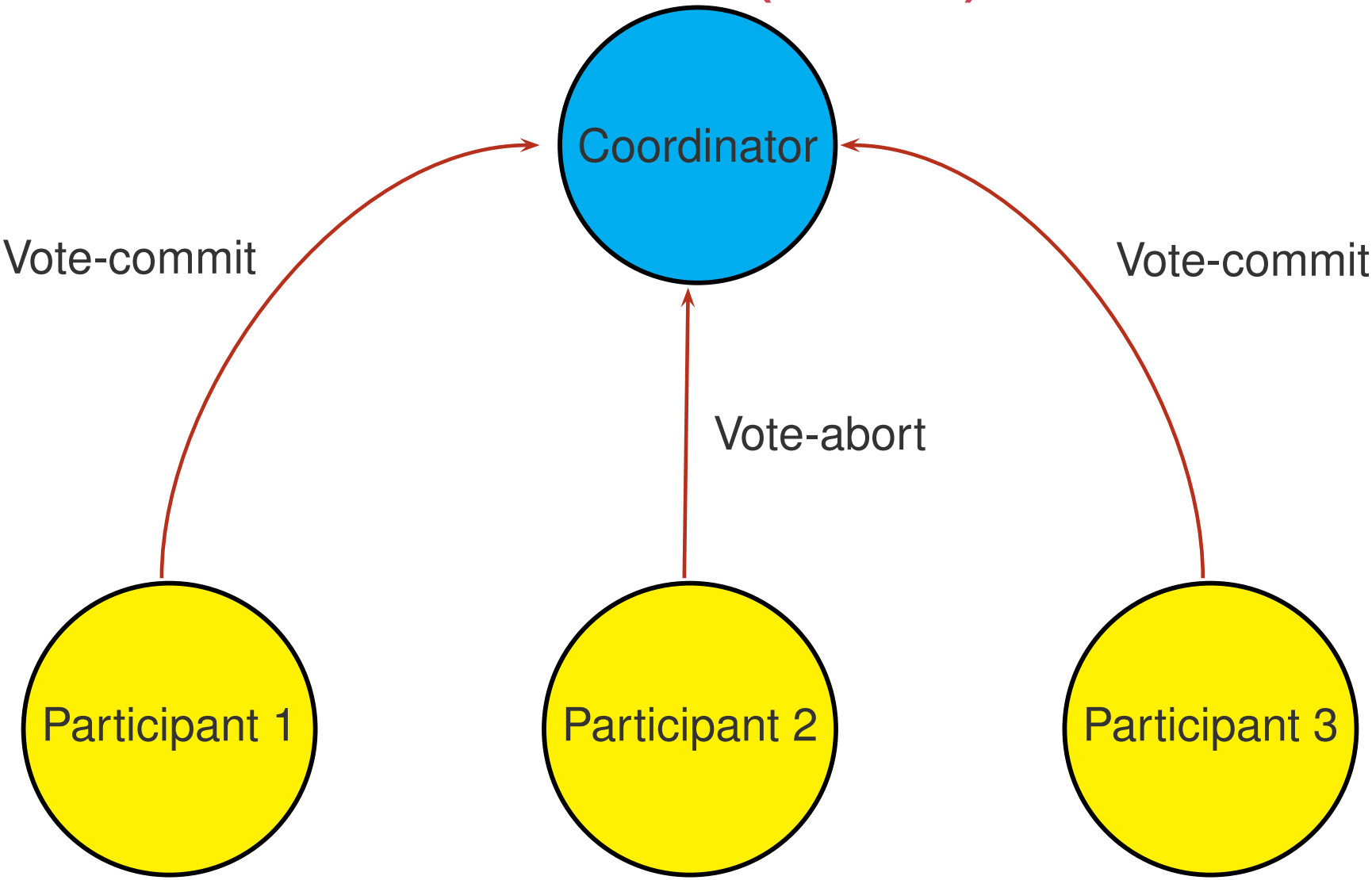
Phase 2: Decision Phase (commit scenario)

Two-Phase Commit (2PC) Protocol



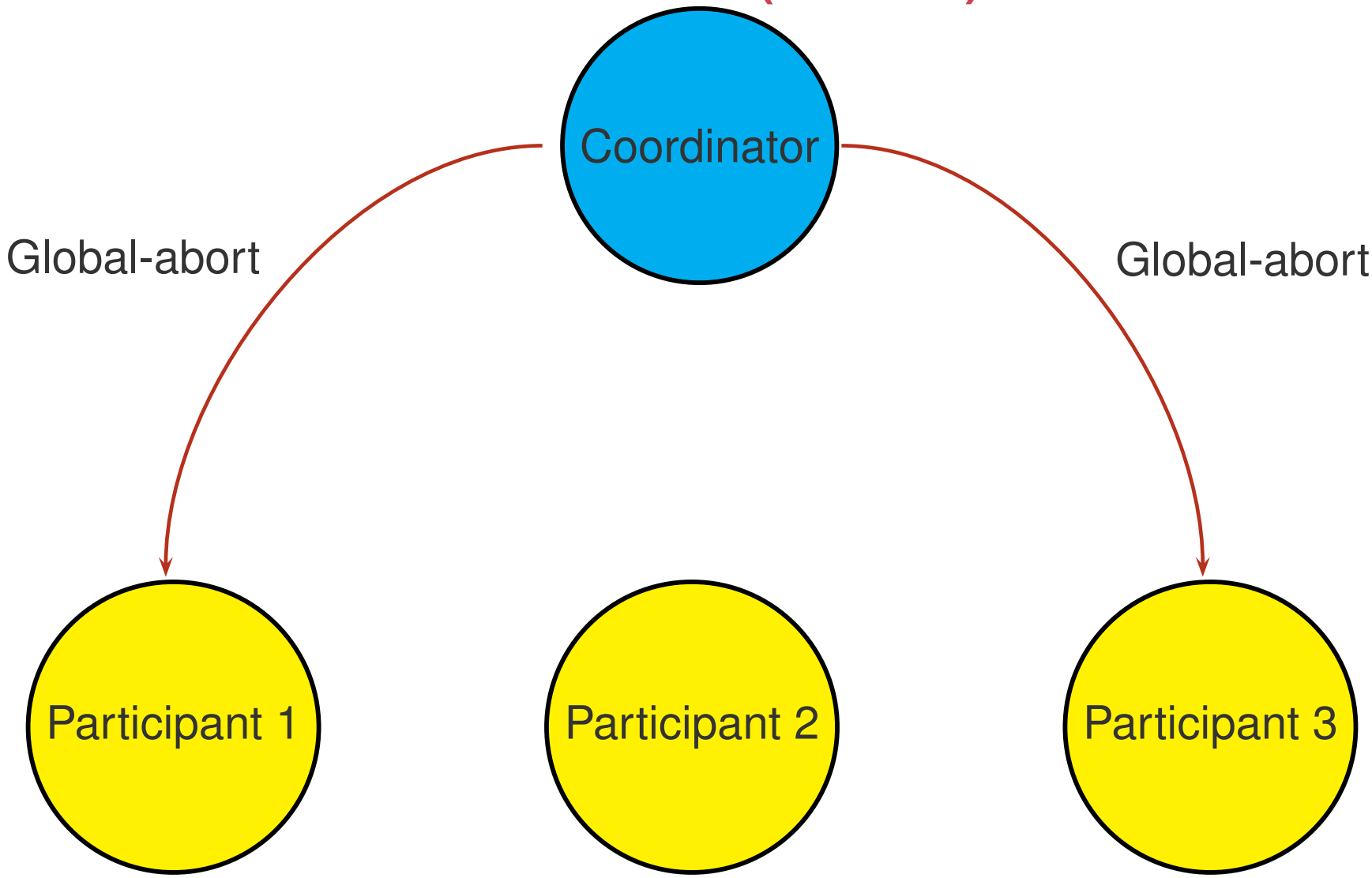
Phase 1: Voting Phase

Two-Phase Commit (2PC) Protocol



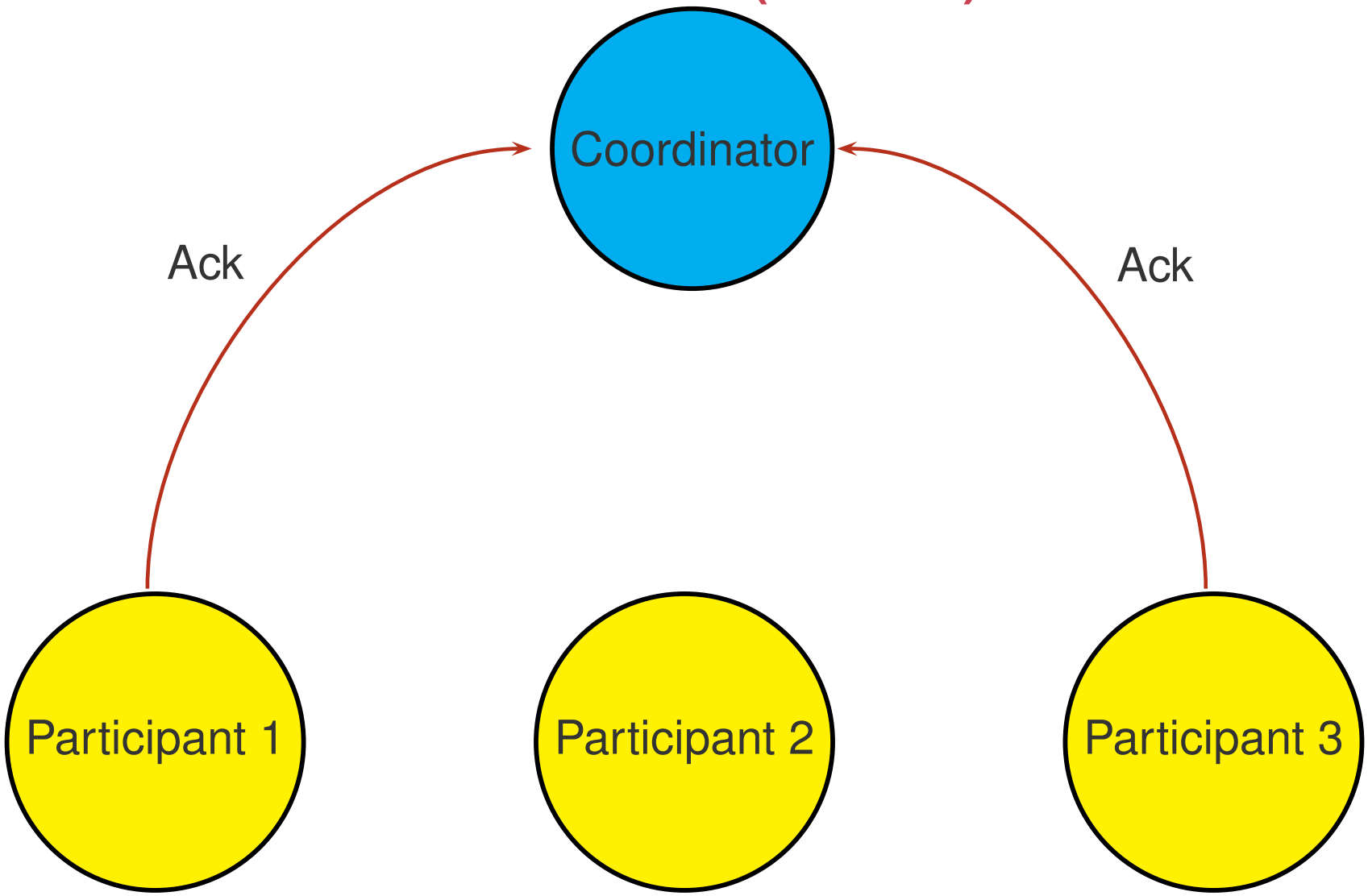
Phase 1: Voting Phase

Two-Phase Commit (2PC) Protocol



Phase 2: Decision Phase (abort scenario)

Two-Phase Commit (2PC) Protocol

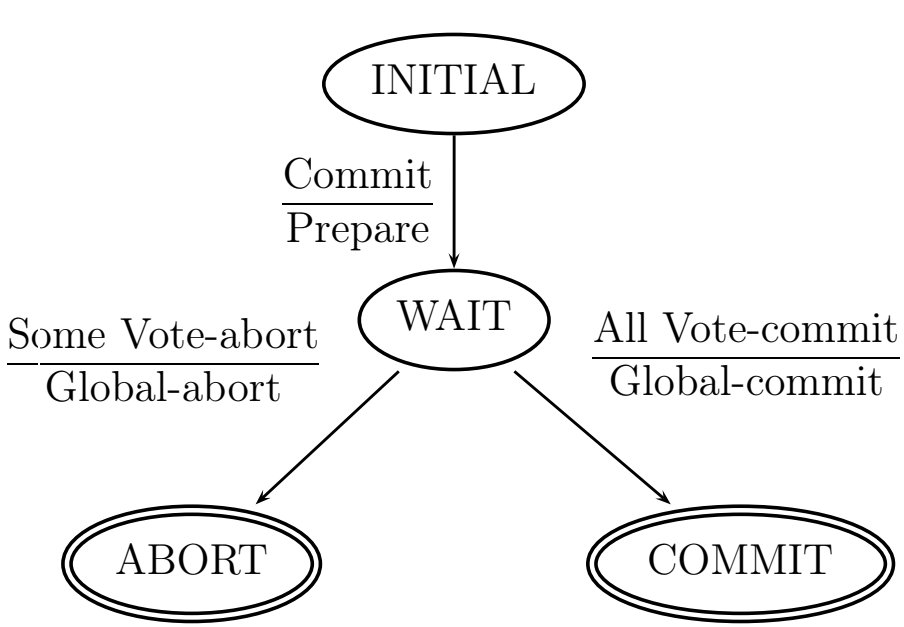


Phase 2: Decision Phase (abort scenario)

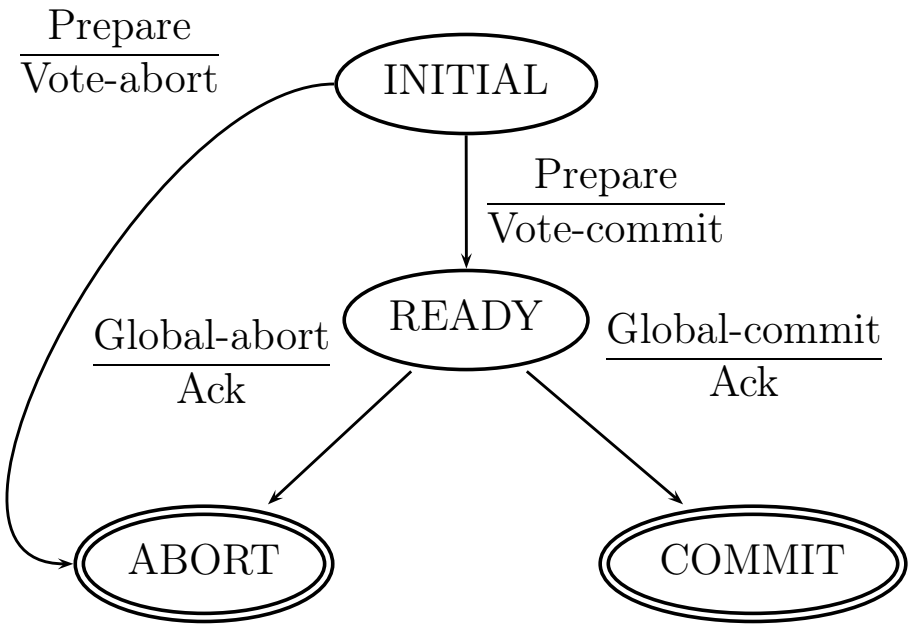
Two-Phase Commit (2PC) Protocol

- A.k.a. Presumed Nothing 2PC (PN-2PC) Protocol
- Ensures atomic commitment of distributed Xacts
- **First phase**: Voting/Preparation phase
  - Coordinator collects votes from participants
- **Second phase**: Decision phase
  - Coordinator sends global decision to participants
- Centralized communication model: Participants communicate only with coordinator

State Transition Diagrams for 2PC



Coordinator



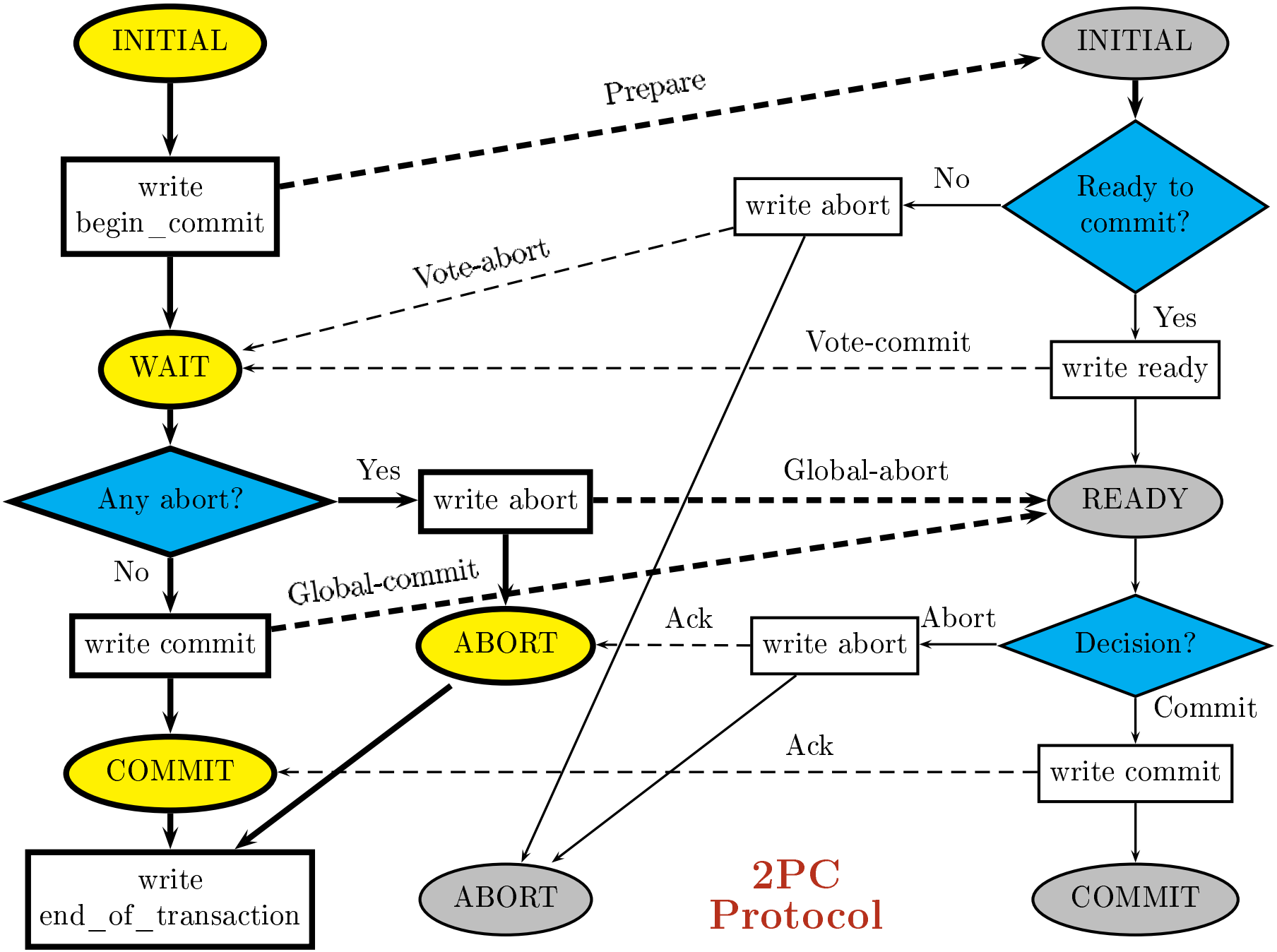
Participant

Properties of 2PC Protocol

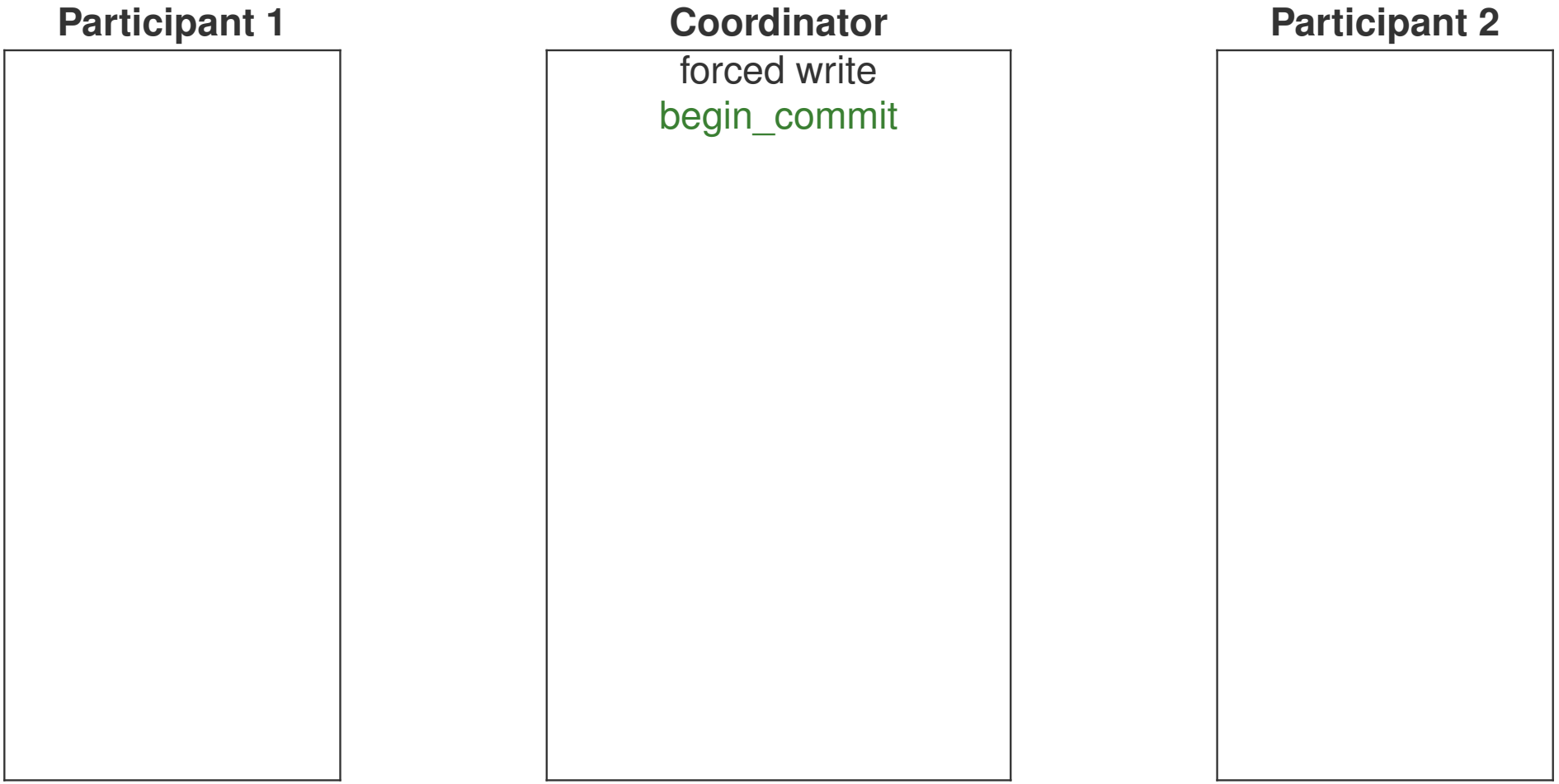
- Every participant must reach the same global decision
- Once a participant has voted, it can't change its vote
- If any participant has voted to abort, the global decision will be abort
- If the global decision is commit, then all participants must have voted to commit
- If there are no failures and all participants voted to commit, then the global decision will be commit
- A participant can unilaterally abort a Xact before it votes
- 2PC is **synchronous within one state transition**
  - No TM leads another TM by more than one state transition during 2PC execution

Log Records

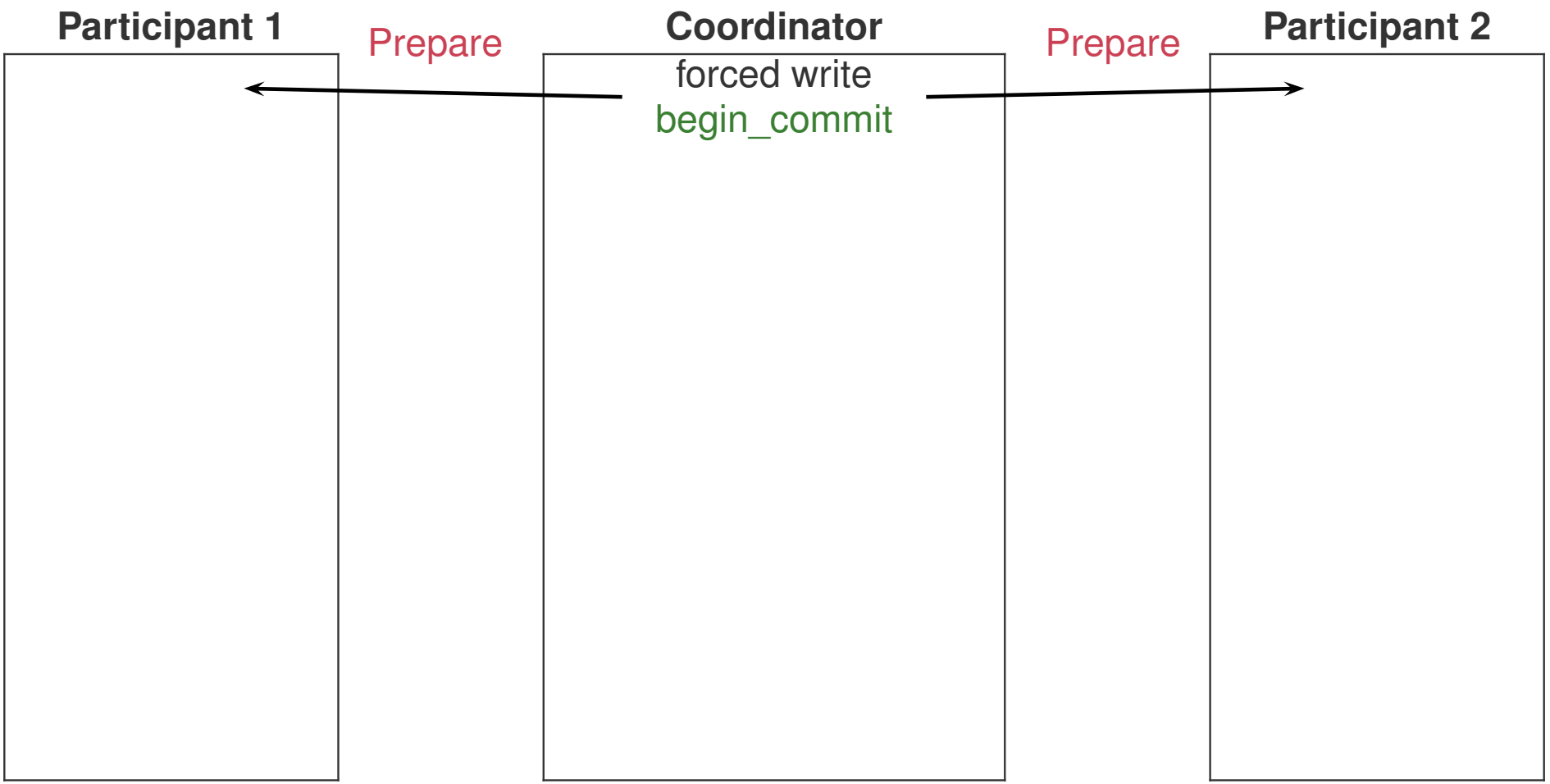
- Before a coordinator/participant process sends a message, it first writes a log record to reflect its state
- A log write is forced if the log record must be flushed to disk before sending the next message; otherwise, the log write is non-forced
  - forced writes = synchronous writes
  - non-forced writes = asynchronous writes
- begin_commit log record includes addresses of all participants


INITIAL
write begin_commit
Prepare
INITIAL
Ready to commit?
No
write abort
Vote-abort
Yes
WAIT
Vote-commit
write ready
Any abort?
Yes
write abort
Global-abort
READY
No
Global-commit
write commit
ABORT
Ack
write abort
Abort
Decision?
Commit
COMMIT
Ack
write commit
write end_of_transaction
ABORT
2PC Protocol
COMMIT

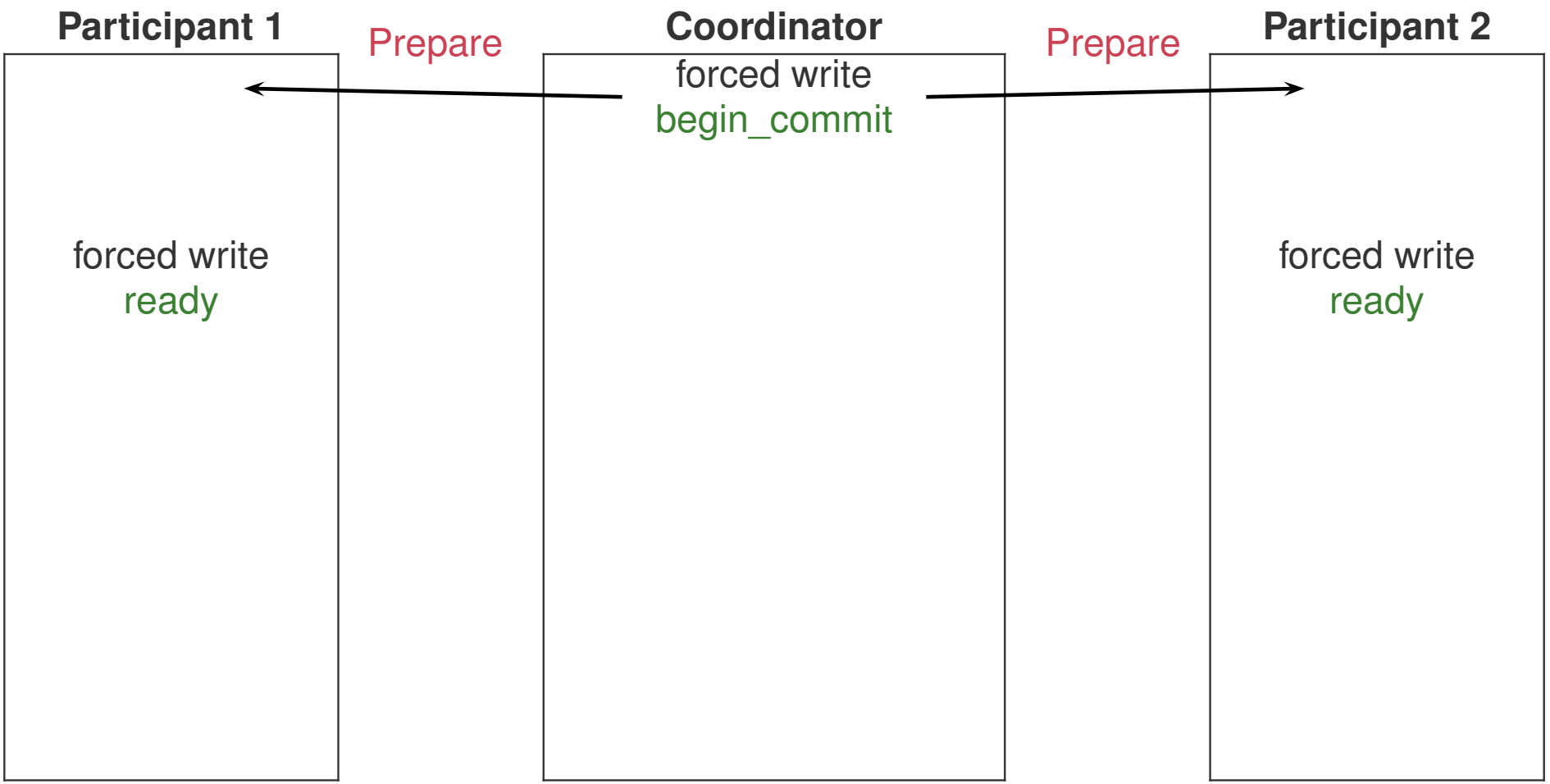
Commit Scenario



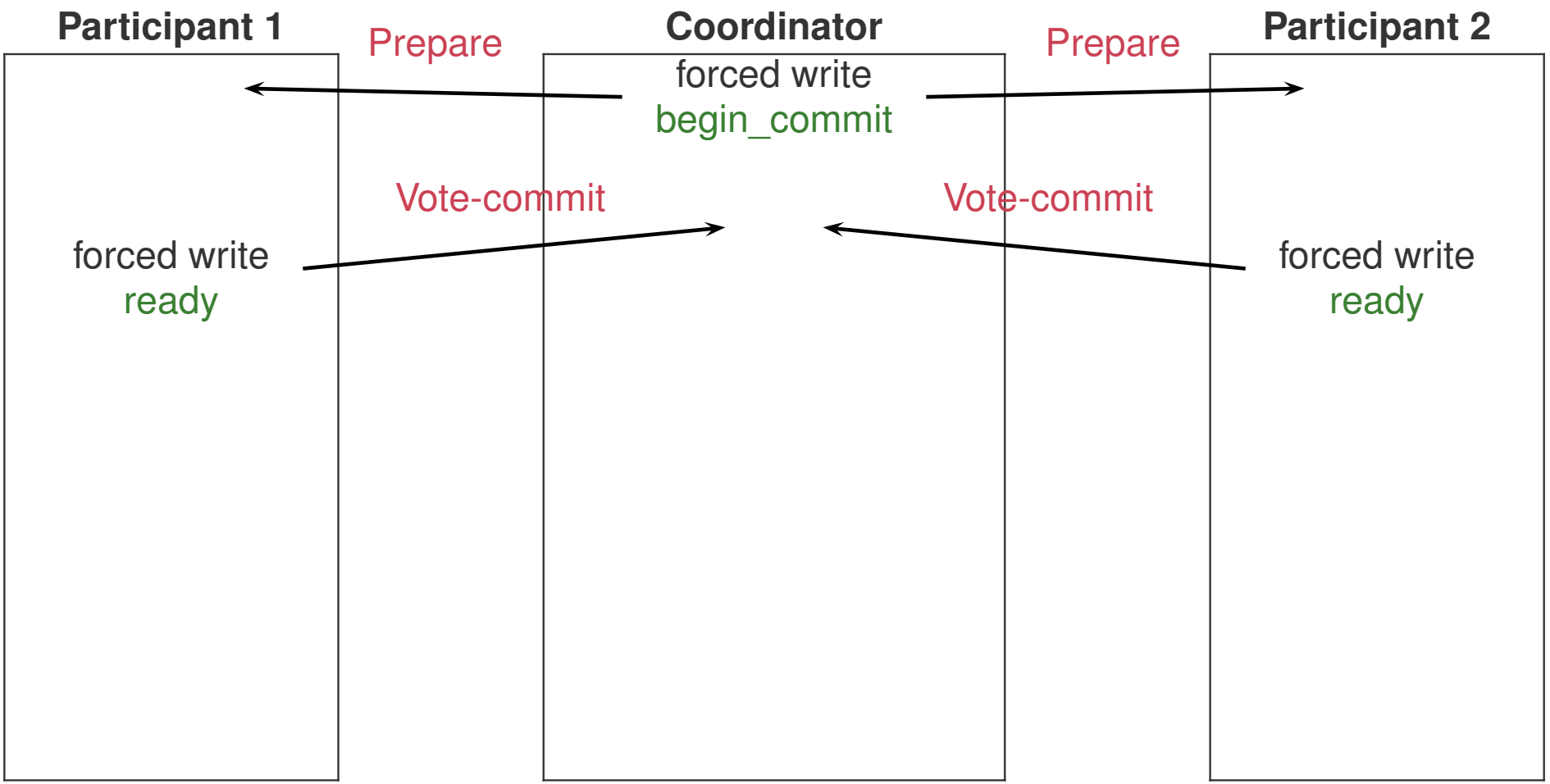
Commit Scenario



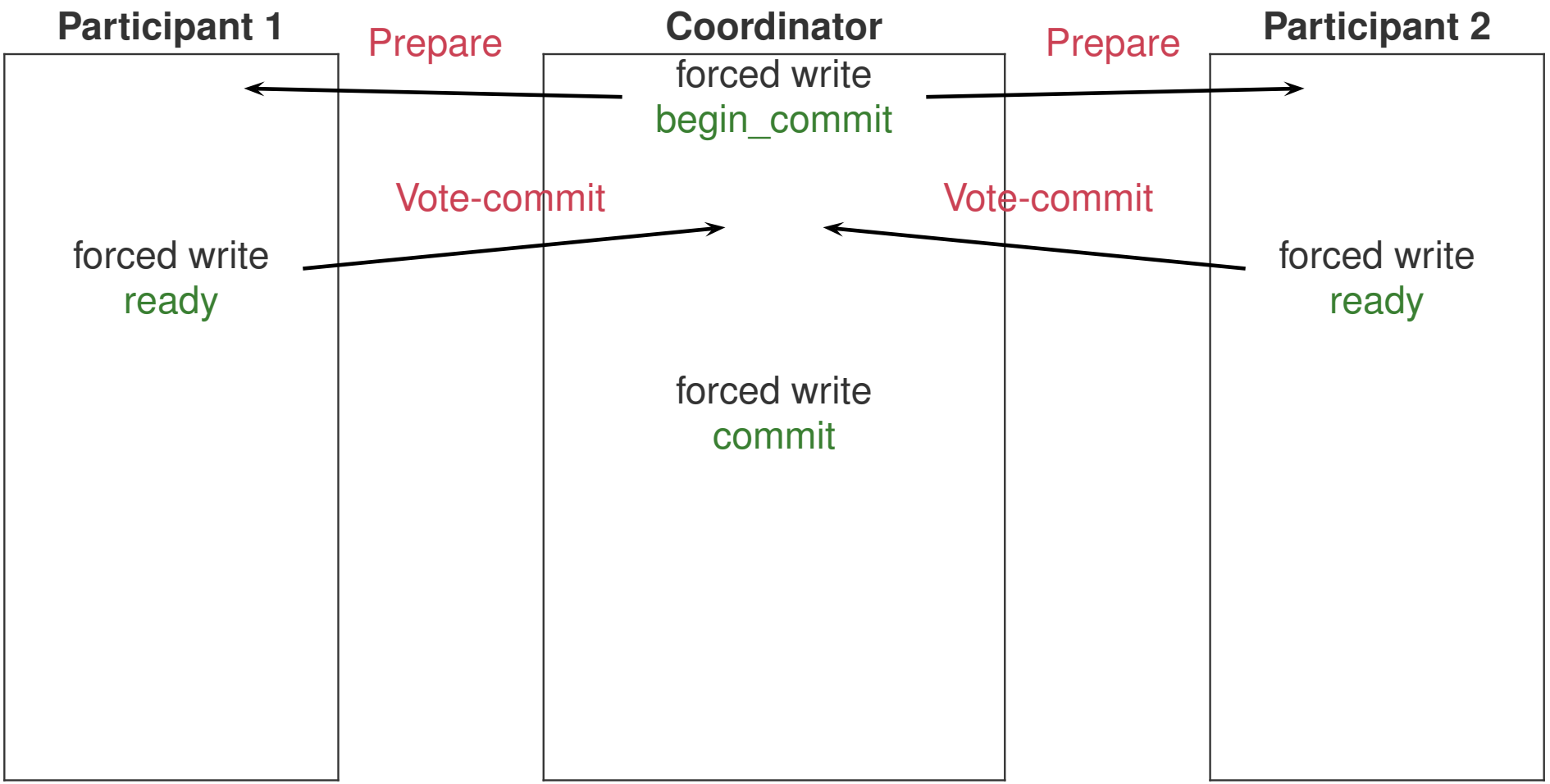
Commit Scenario



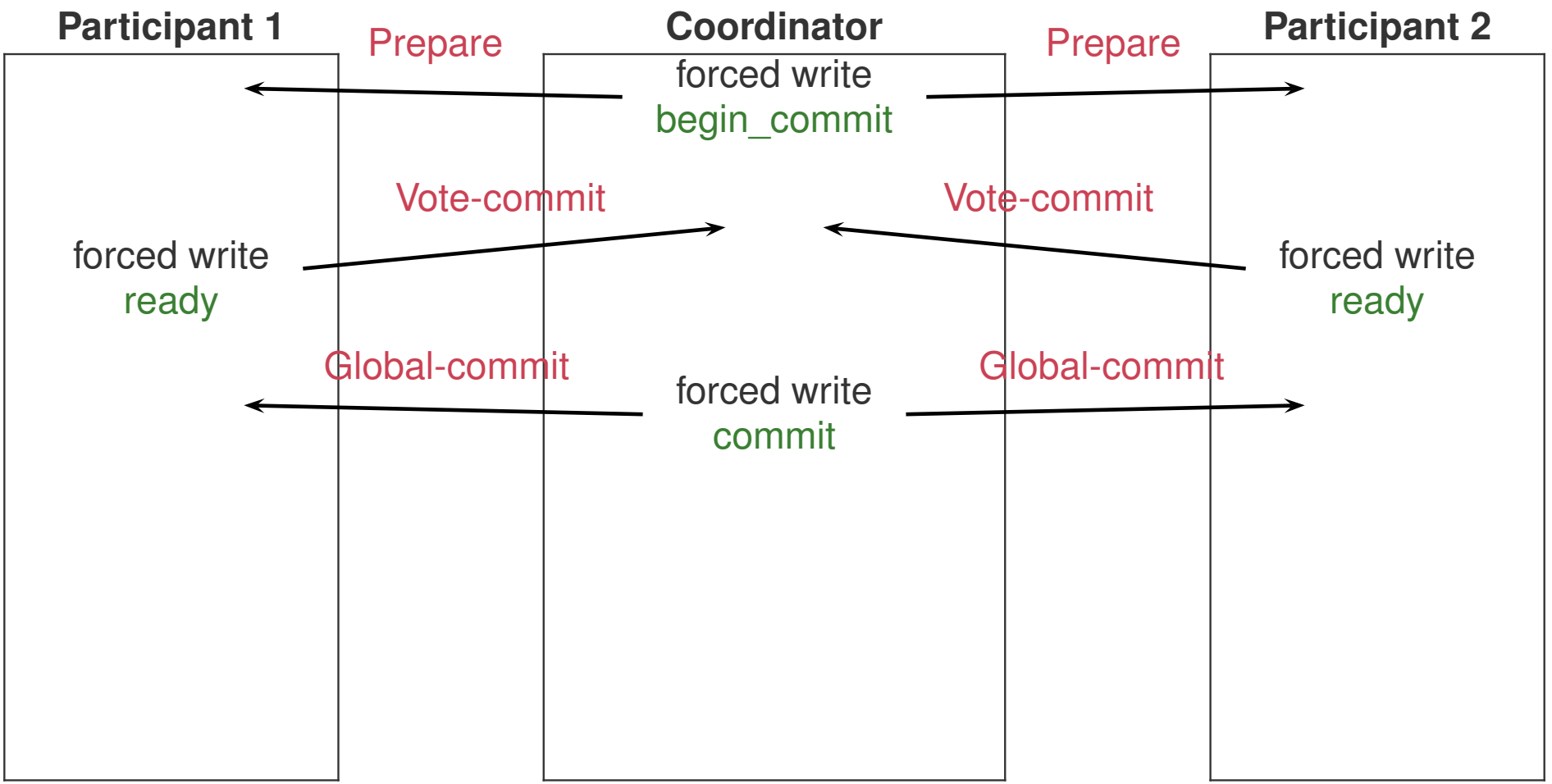
Commit Scenario



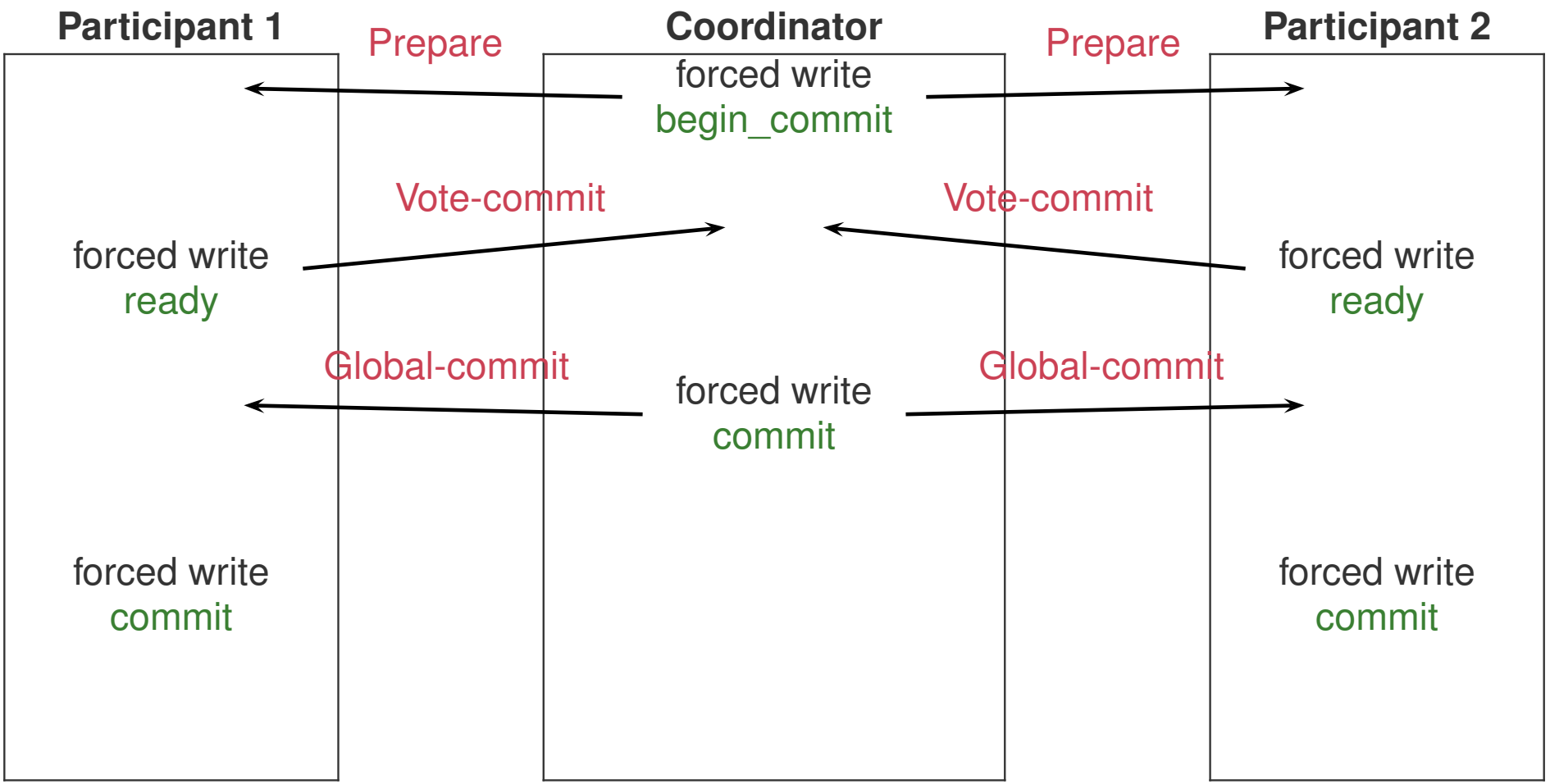
Commit Scenario



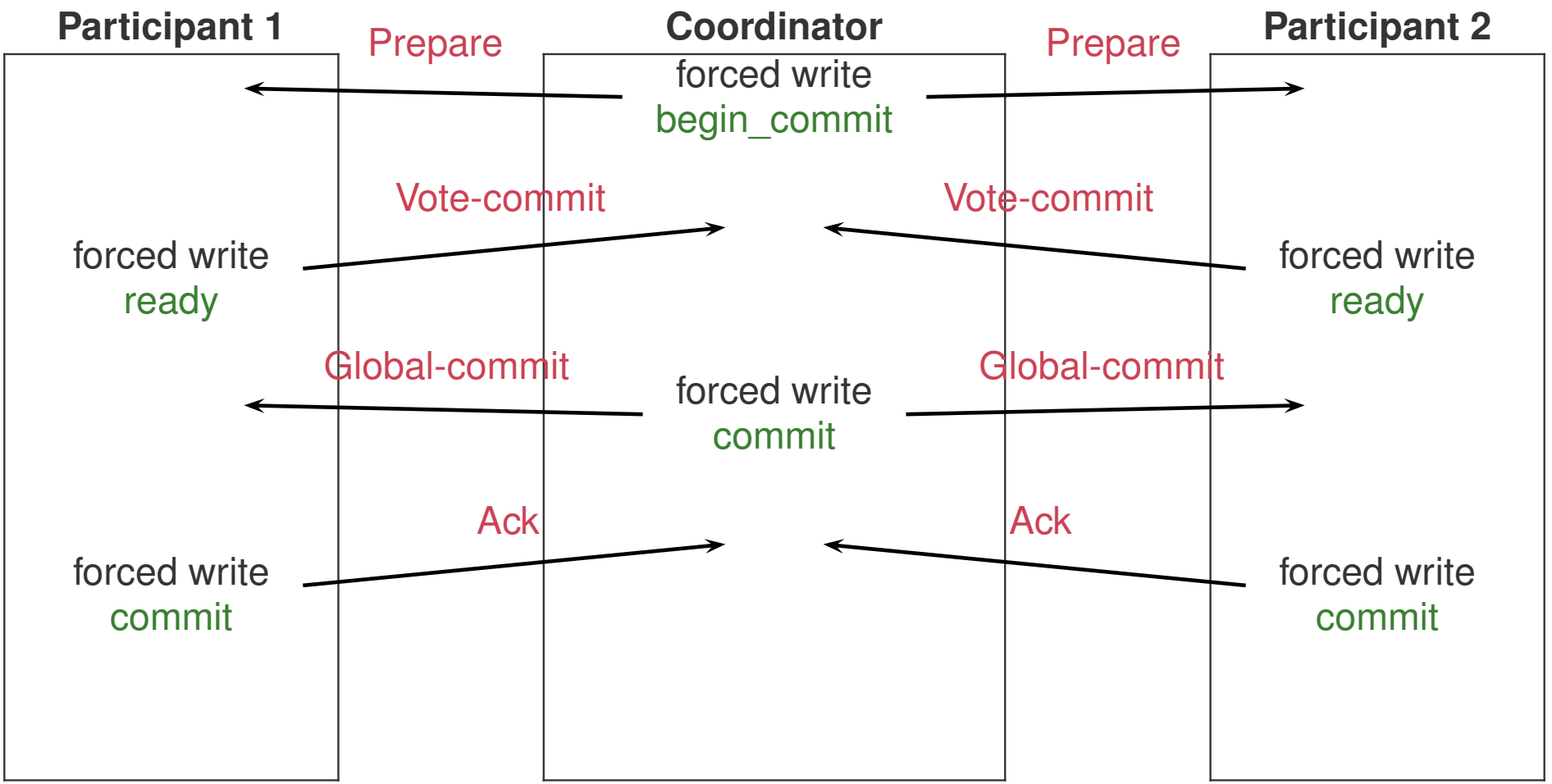
Commit Scenario



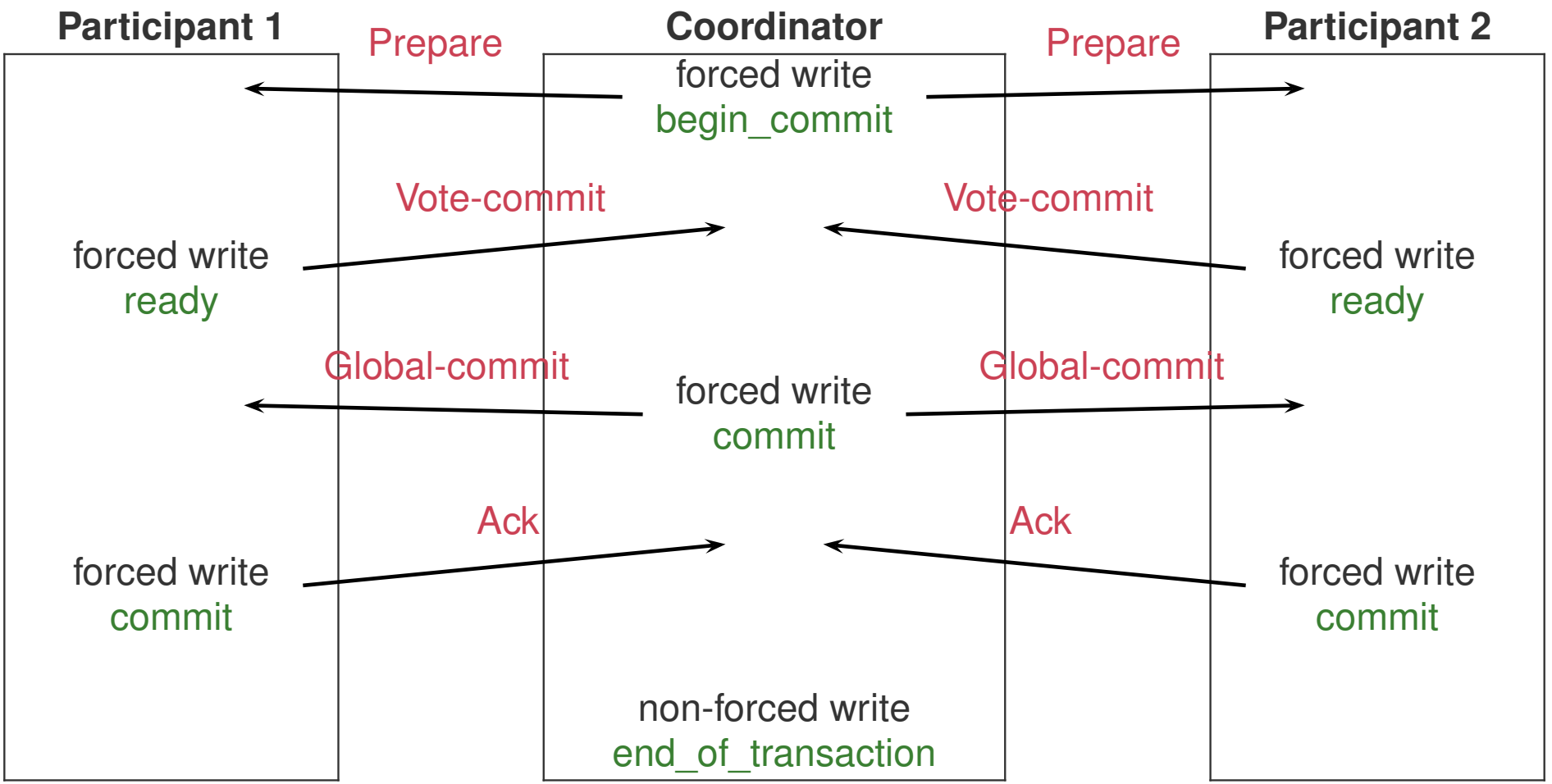
Commit Scenario



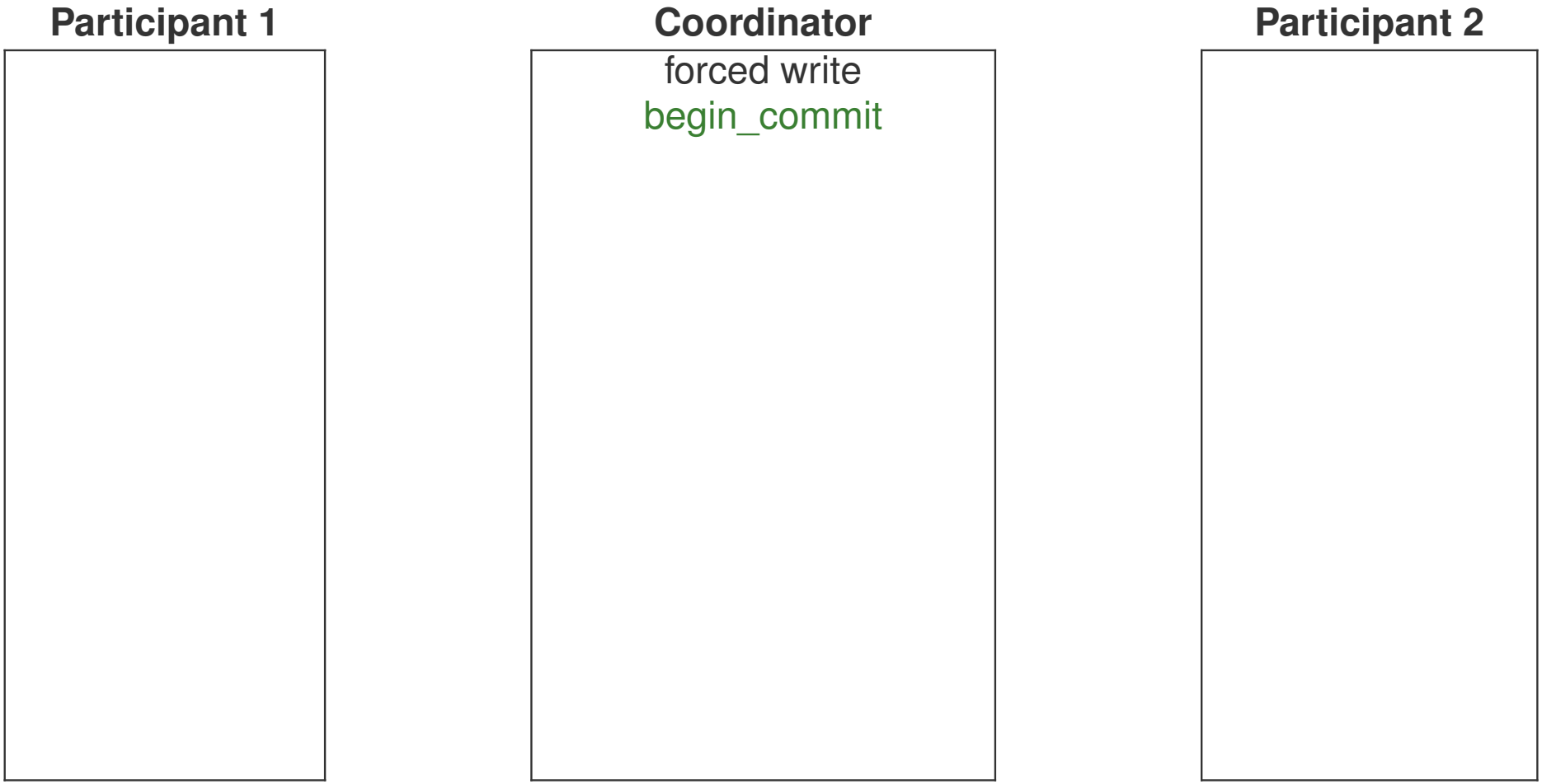
Commit Scenario



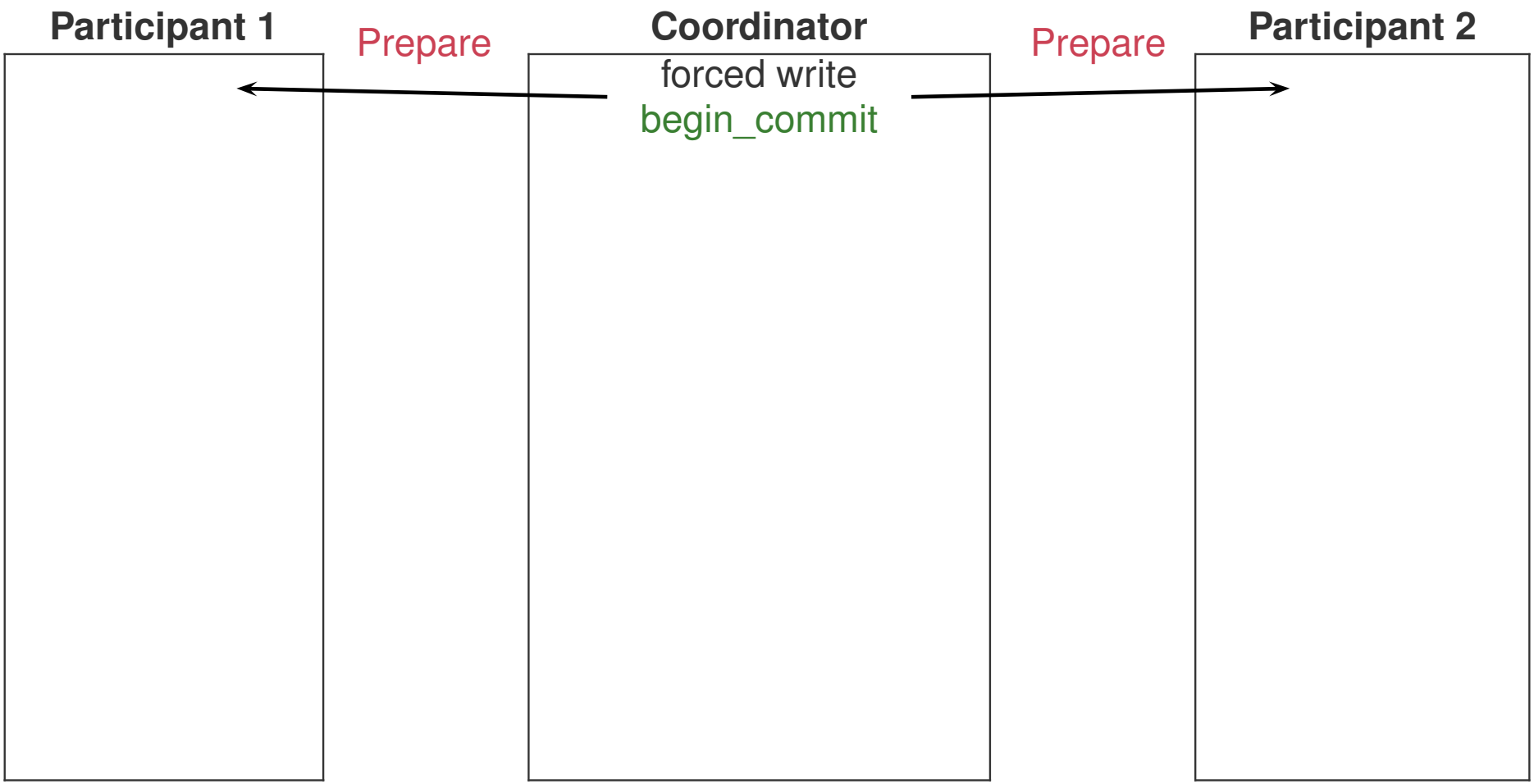
Commit Scenario



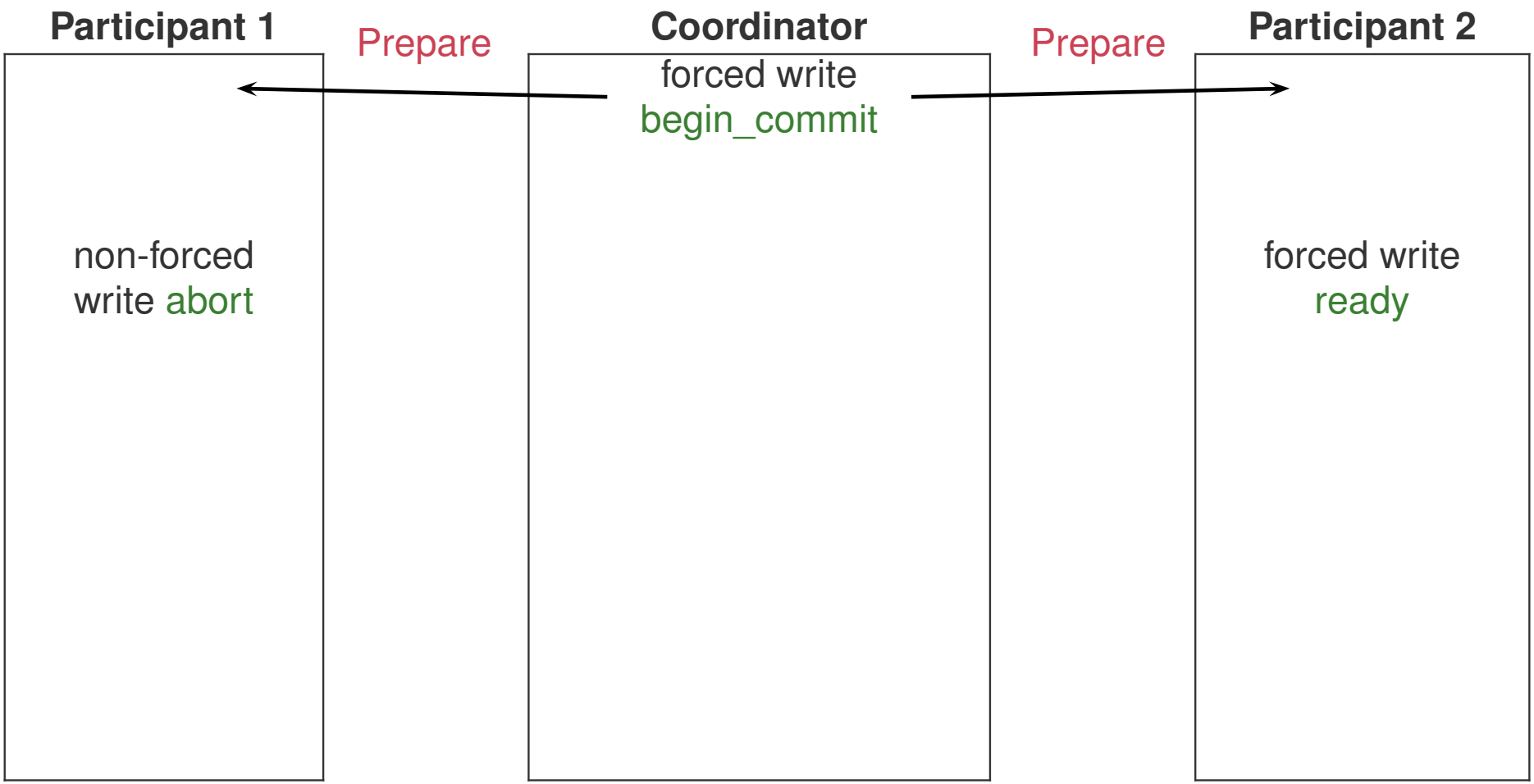
Abort Scenario



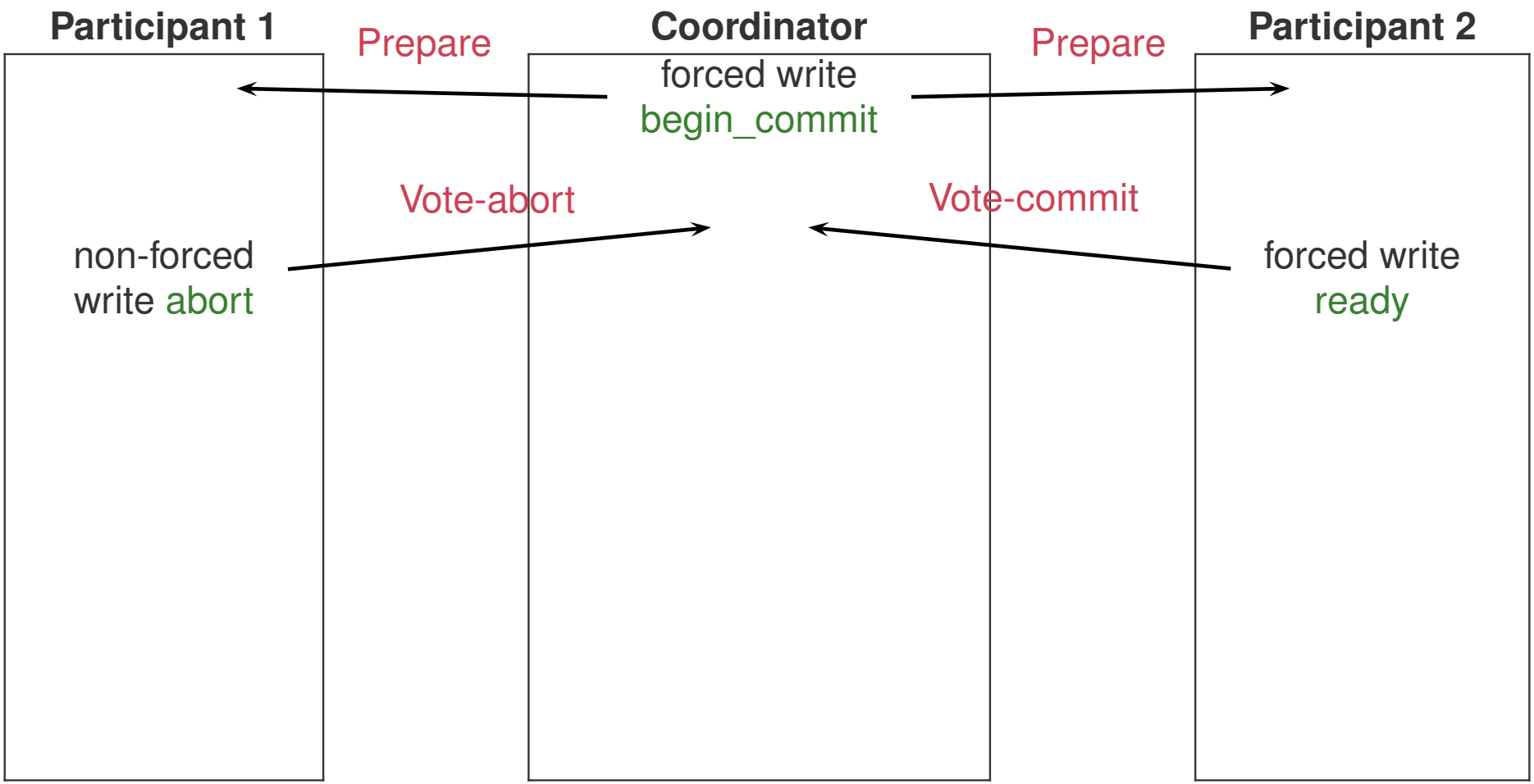
Abort Scenario



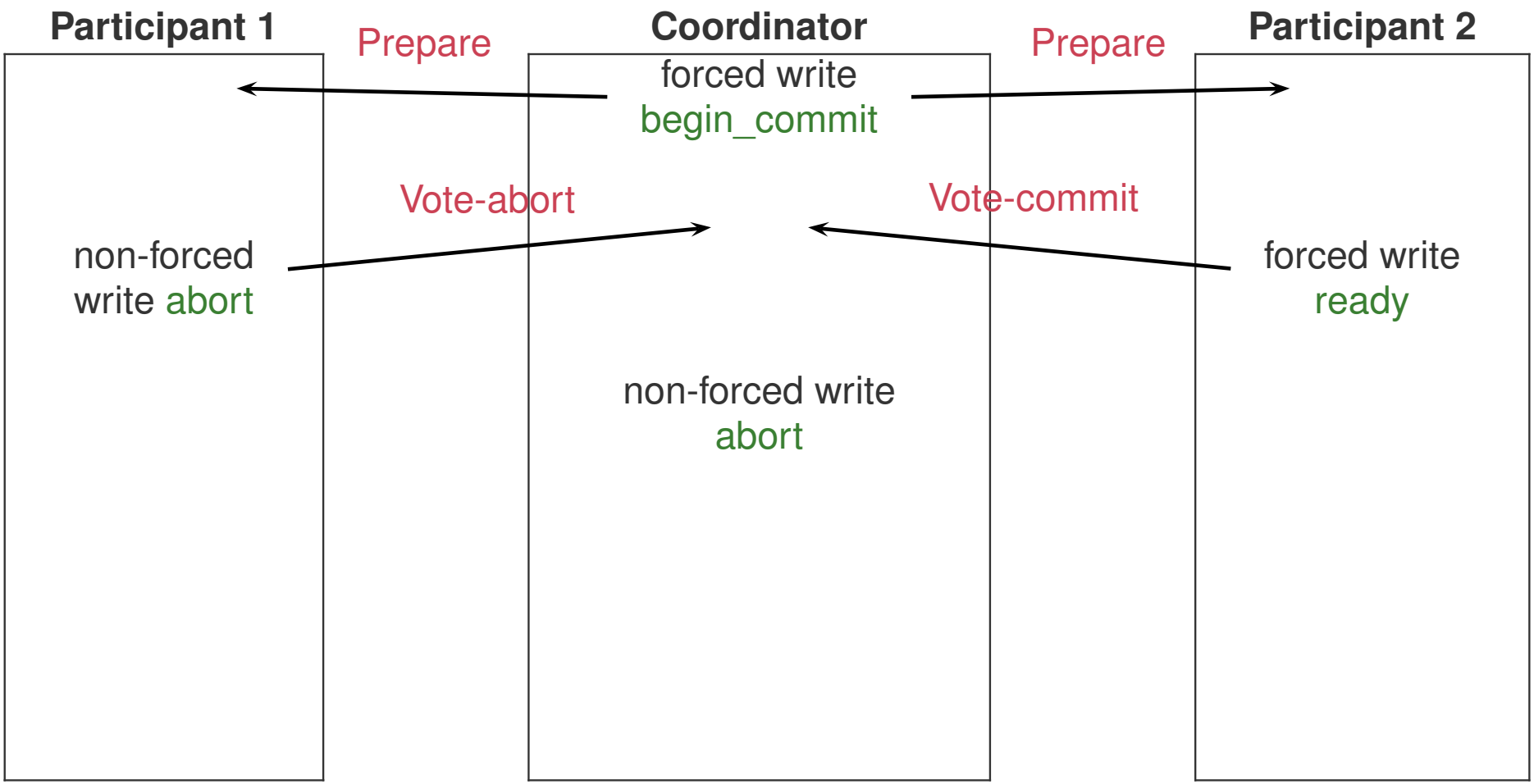
Abort Scenario



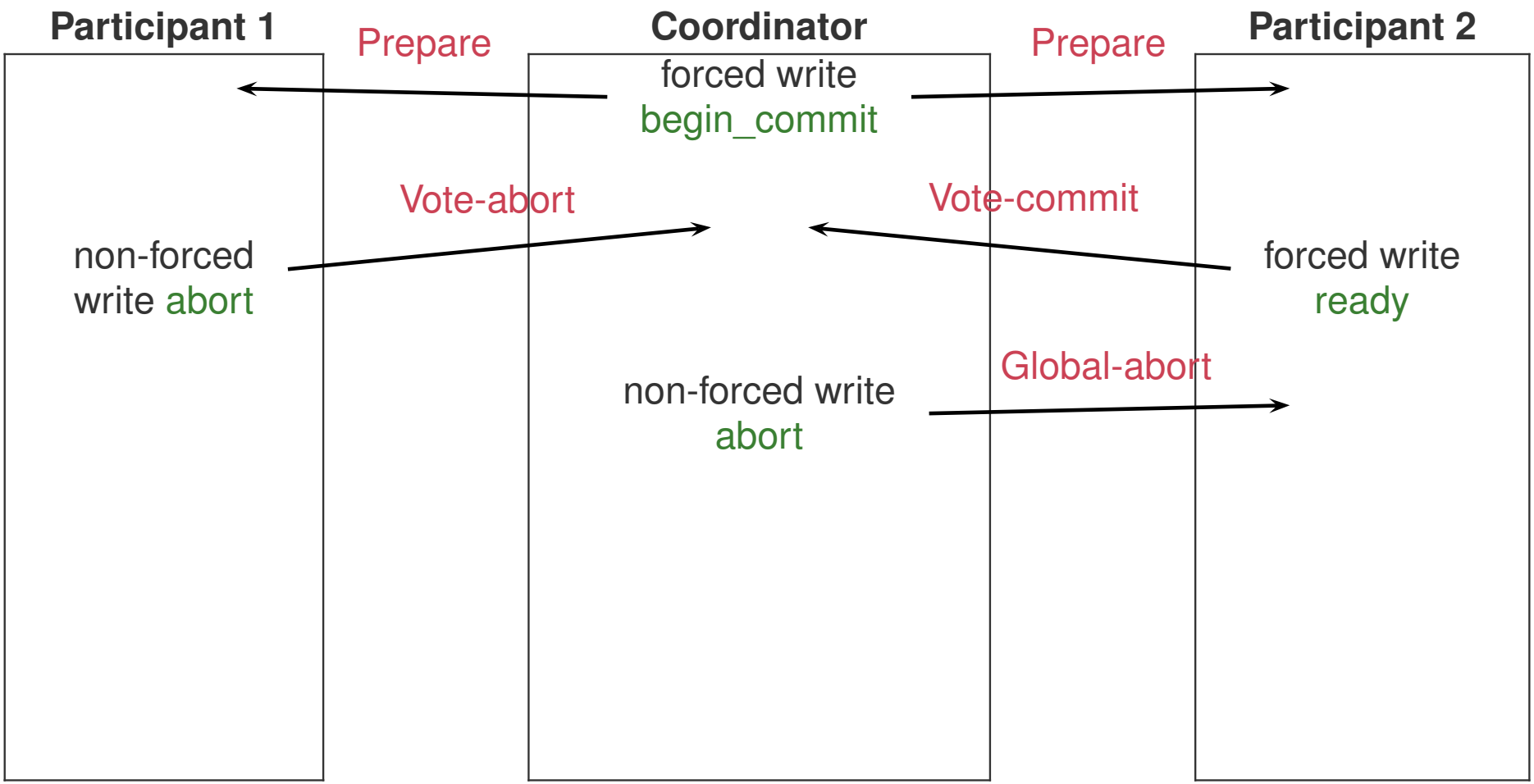
Abort Scenario



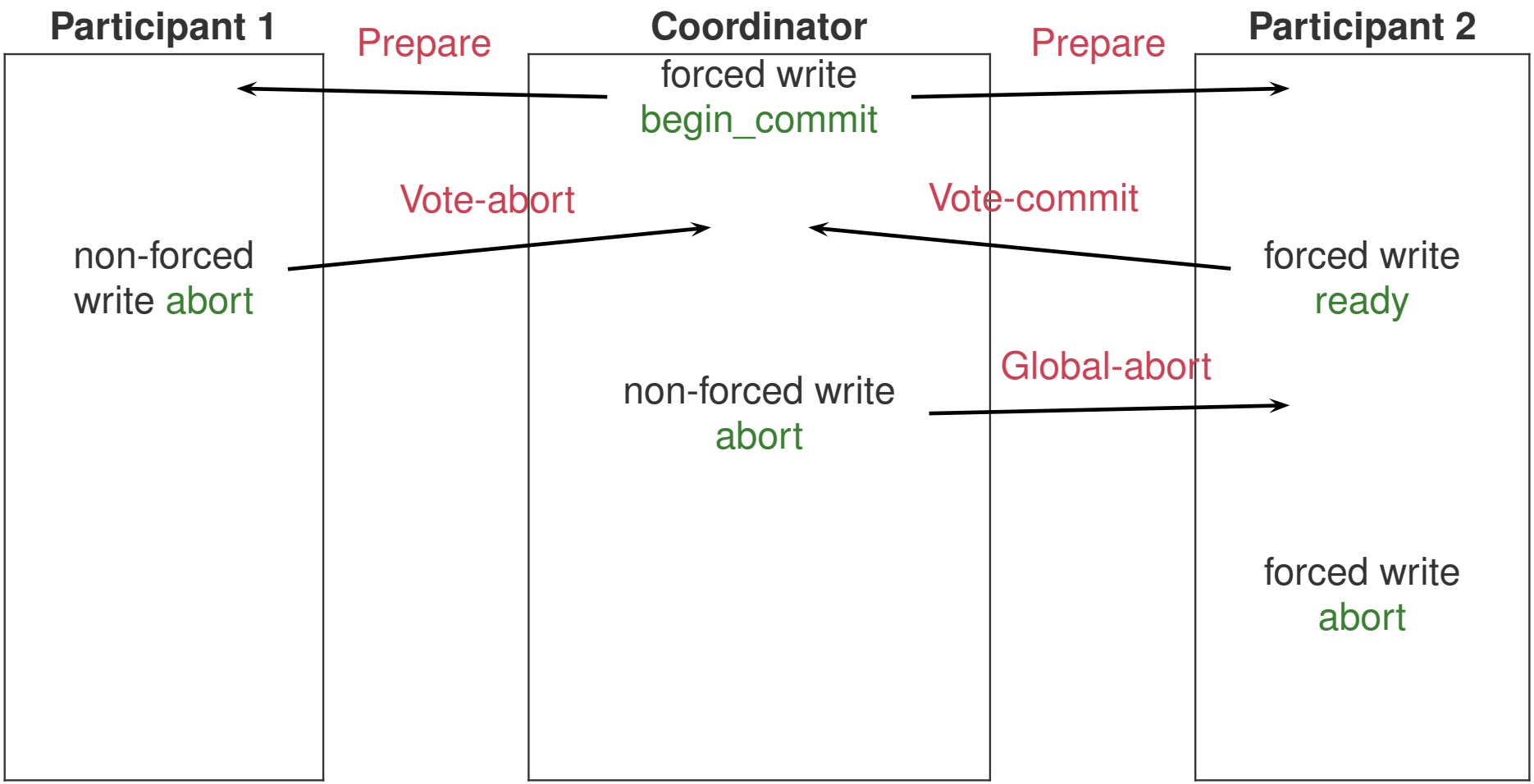
Abort Scenario



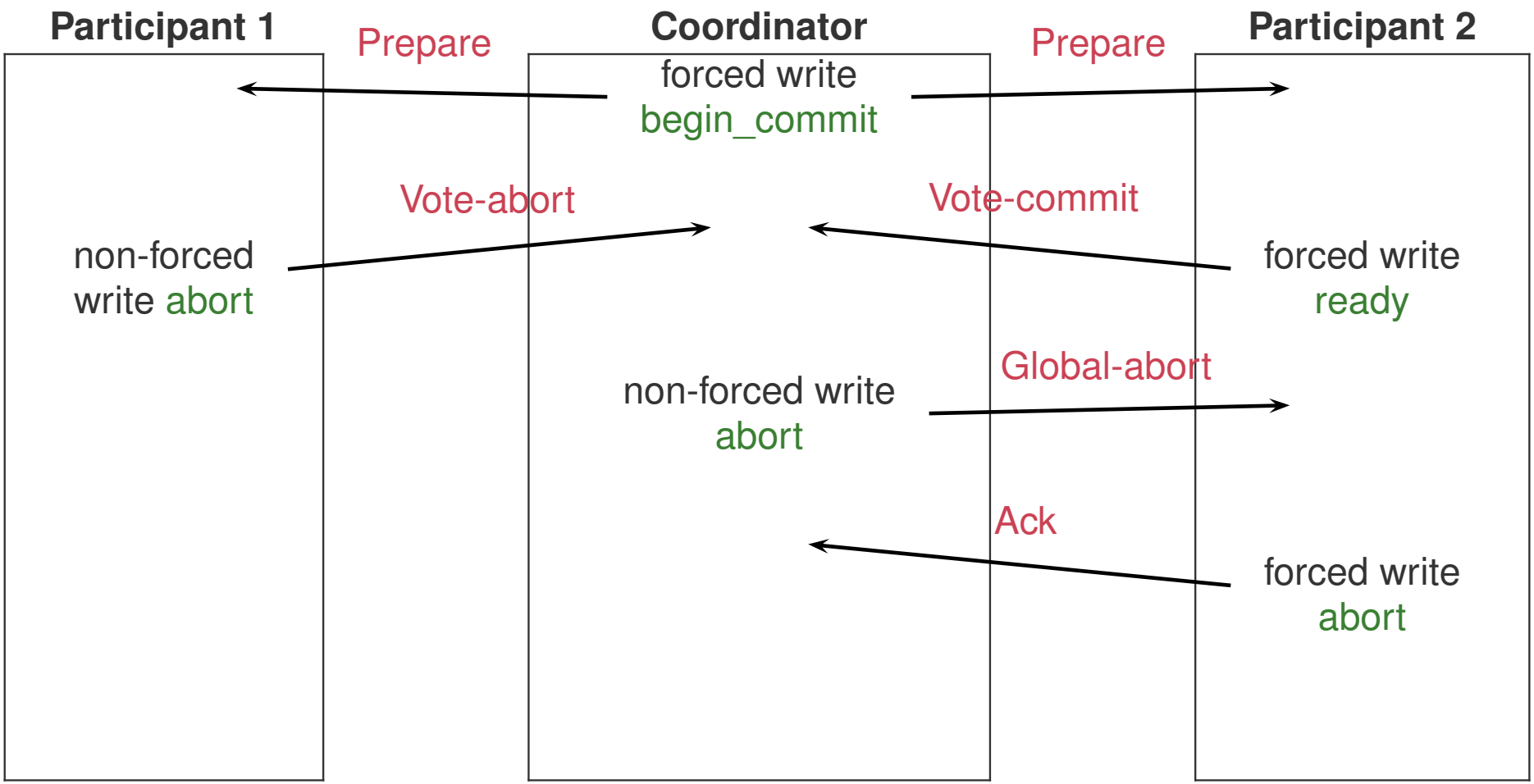
Abort Scenario



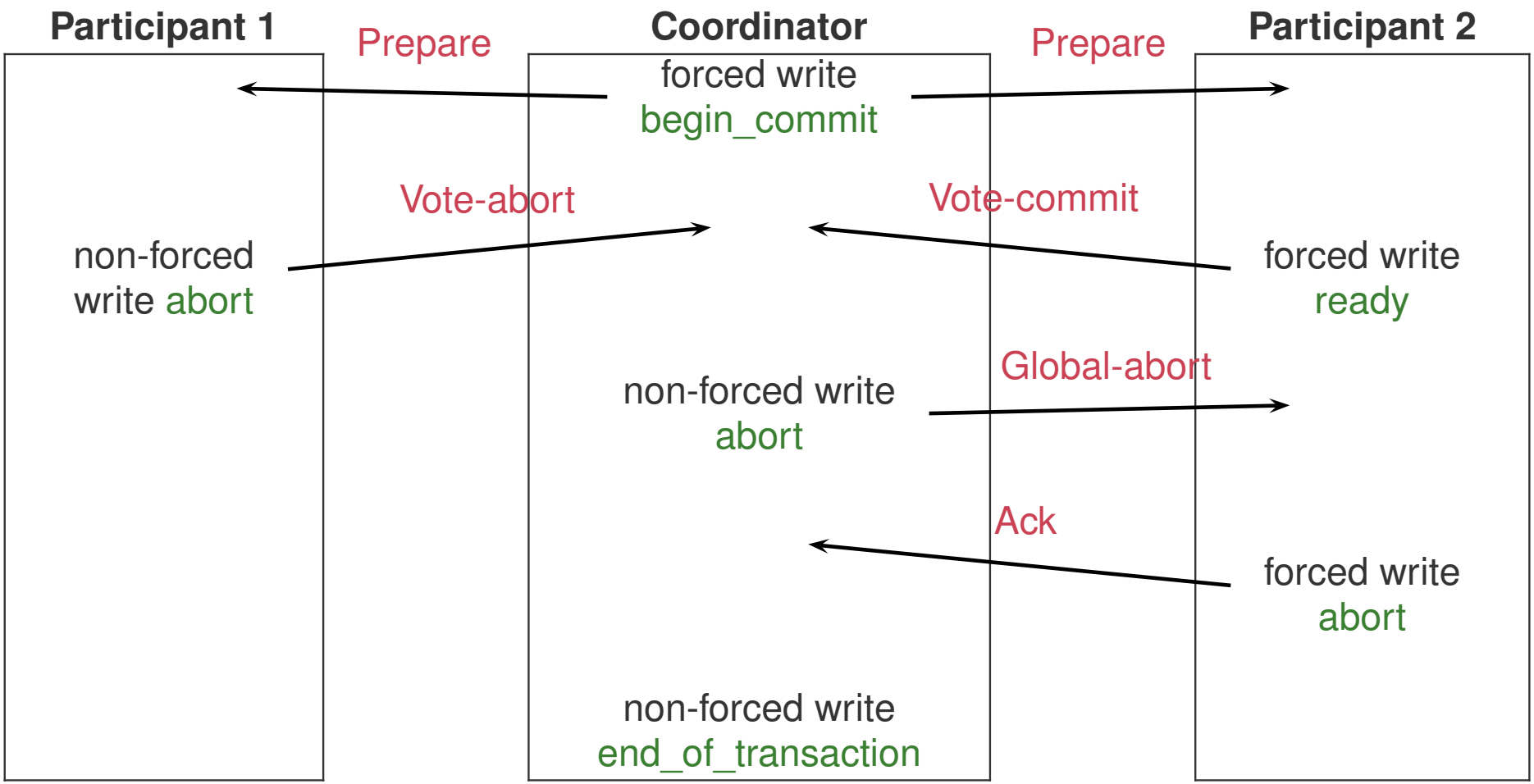
Abort Scenario



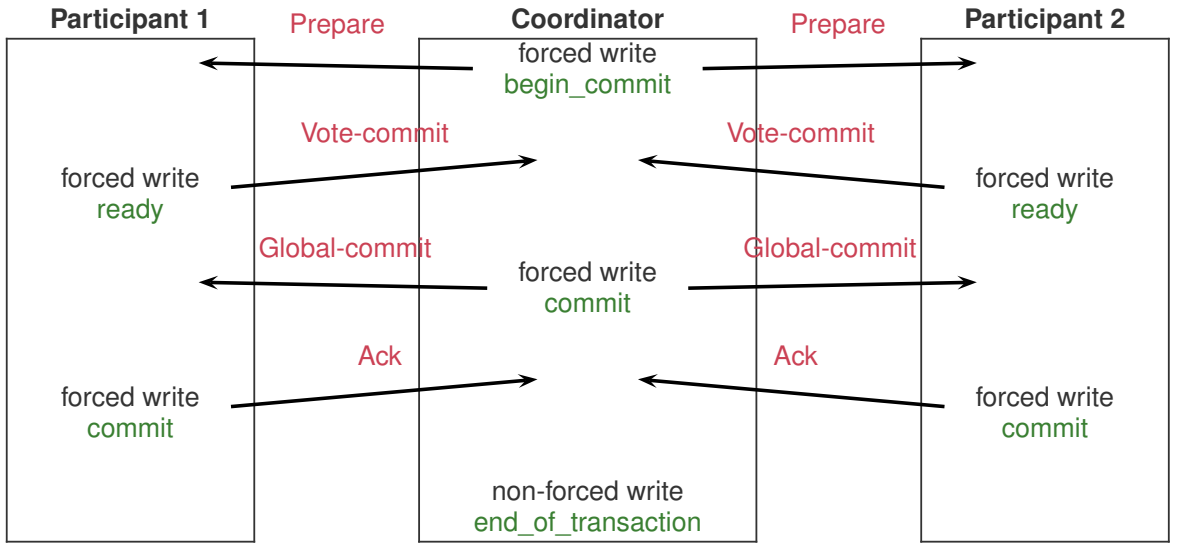
Abort Scenario

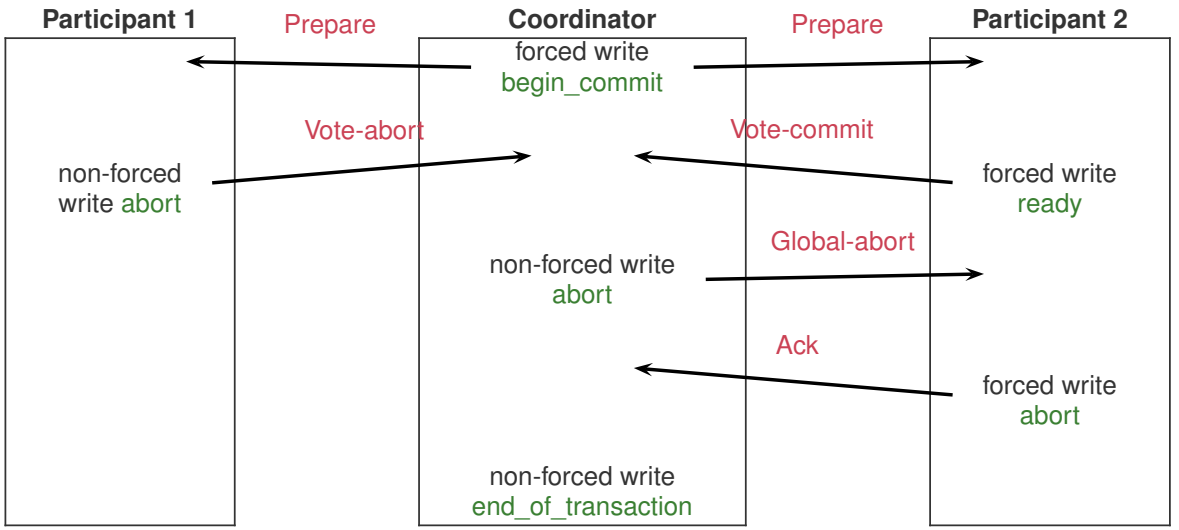


Abort Scenario



Commit vs Abort Scenario





Dealing with Site Failures

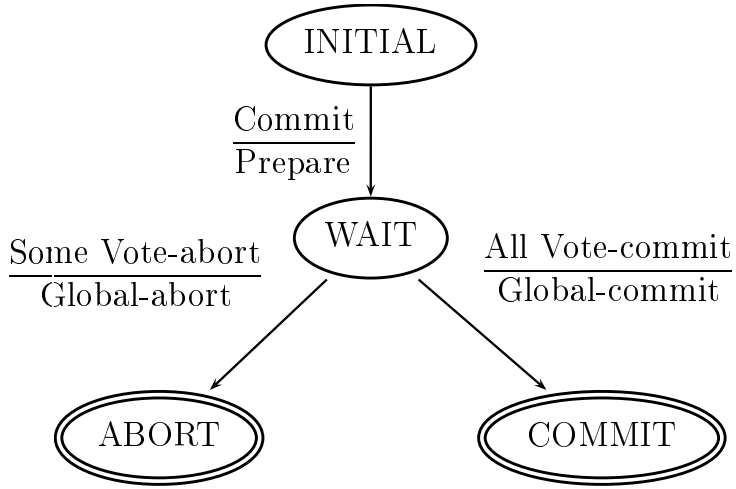
- Failures are detected by timeouts
- Each coordinator/participant process starts a timer when it enters a state that involves waiting for a message
- If expected message is not received before timer runs out, process times out & invokes termination protocol
- **Recovery protocol**
  - How a failed site recovers after being restarted
- **Termination protocol**
  - How operational sites deal with failures

Recovery Protocols for 2PC

- How a failed site recovers after being restarted
- A recovery protocol is independent if it can determine how to terminate a transaction that was executing at the time of a failure without having to consult any other site

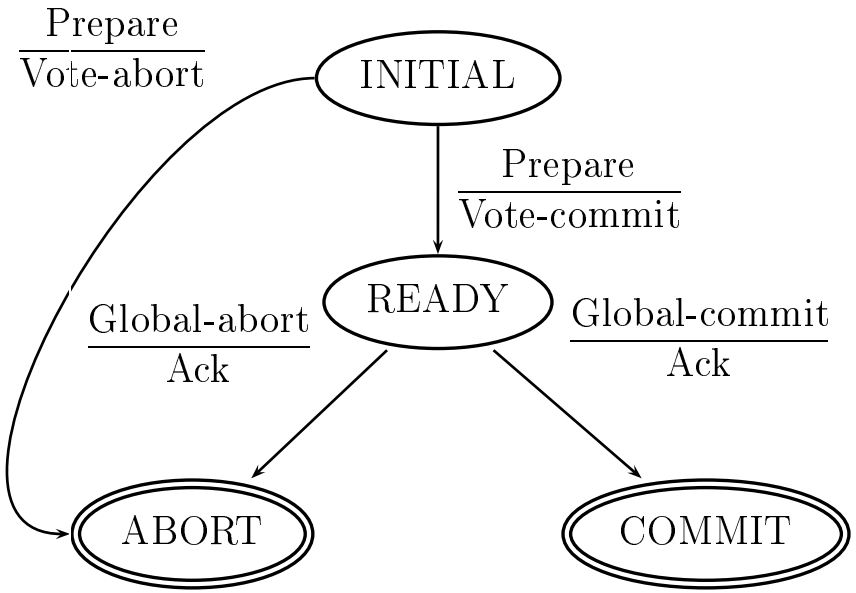
Recovery Protocol for Coordinator

Fails in state	Coordinator's recovery actions
INITIAL	Writes an abort record in log & sends "Global-abort" to all participants
WAIT	Writes an abort record in log & sends "Global-abort" to all participants
ABORT	Does nothing if all ACKs have been received; otherwise, sends "Global-abort" to all participants
COMMIT	Does nothing if all ACKs have been received; otherwise, sends "Global-commit" to all participants



Recovery Protocol for Participants

Fails in state	Participant's recovery actions
INITIAL	Aborts transaction unilaterally
READY	Sends "Vote-commit" to coordinator
ABORT	Does nothing
COMMIT	Does nothing

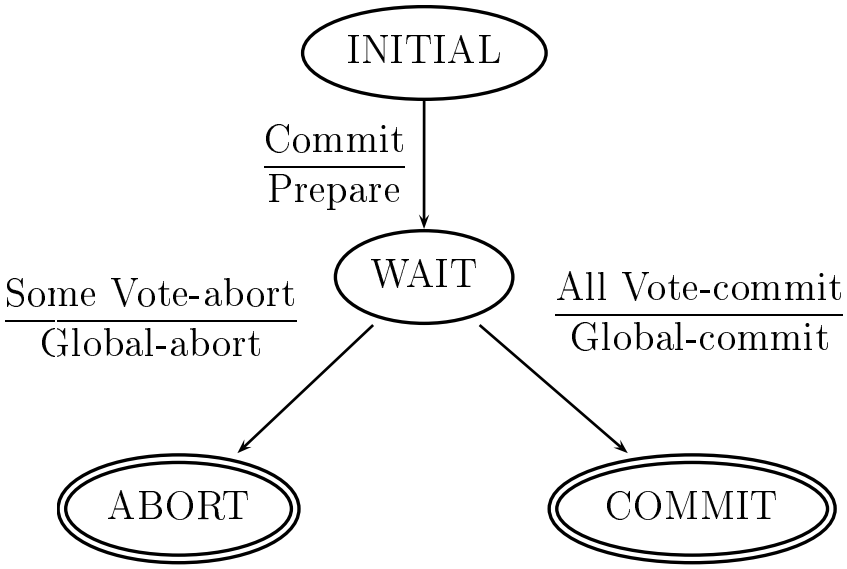


Termination Protocols for 2PC

- How operational sites deal with failures
- A termination protocol is non-blocking if it permits a transaction to terminate (i.e., commit/abort) at the operational sites without waiting for recovery of the failed site
- 2PC Termination Protocols
  - Basic Termination Protocol
  - Cooperative Termination Protocol

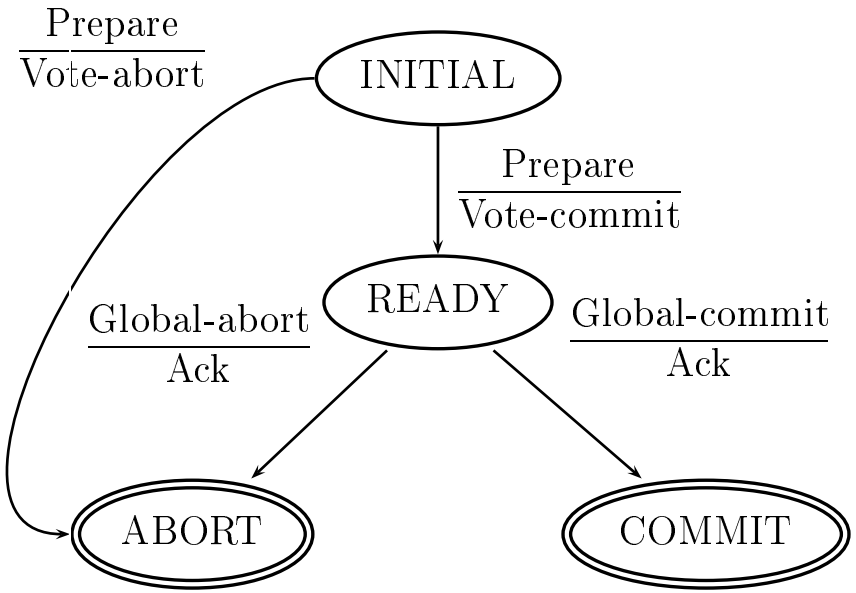
Termination Protocol for Coordinator

Timeout in state	Coordinator's termination actions
WAIT	Writes an abort record in log & sends "Global-abort" to all participants
ABORT	Sends "Global-abort" to participants who have not responded
COMMIT	Sends "Global-commit" to participants who have not responded



Termination Protocol for Participants

Timeout in state	Participant's termination actions
INITIAL	Aborts transaction unilaterally
READY	Participant is blocked!



Cooperative Termination Protocol

- **Goal**: Reduce probability of blocking by failed coordinator
- Participants can communicate with other participants
  - Coordinator includes addresses of all participants in "Prepare" message
- When a participant *P* timeout in READY state
  - *P* sends "Decision-request" message to all other participants

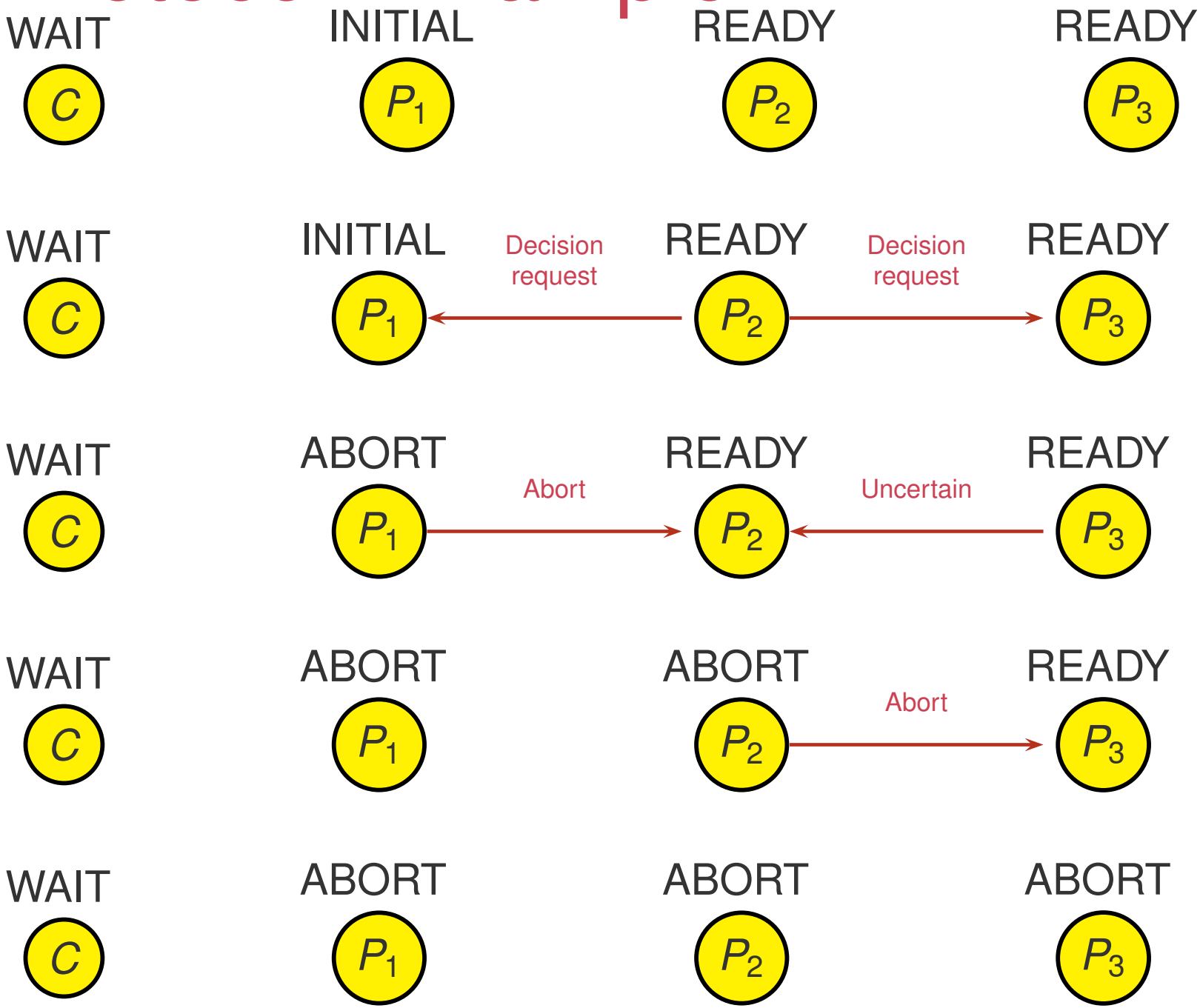
Cooperative Termination Protocol (cont.)

- When a participant *Q* receives a "Decision-request" message, it responds as follows

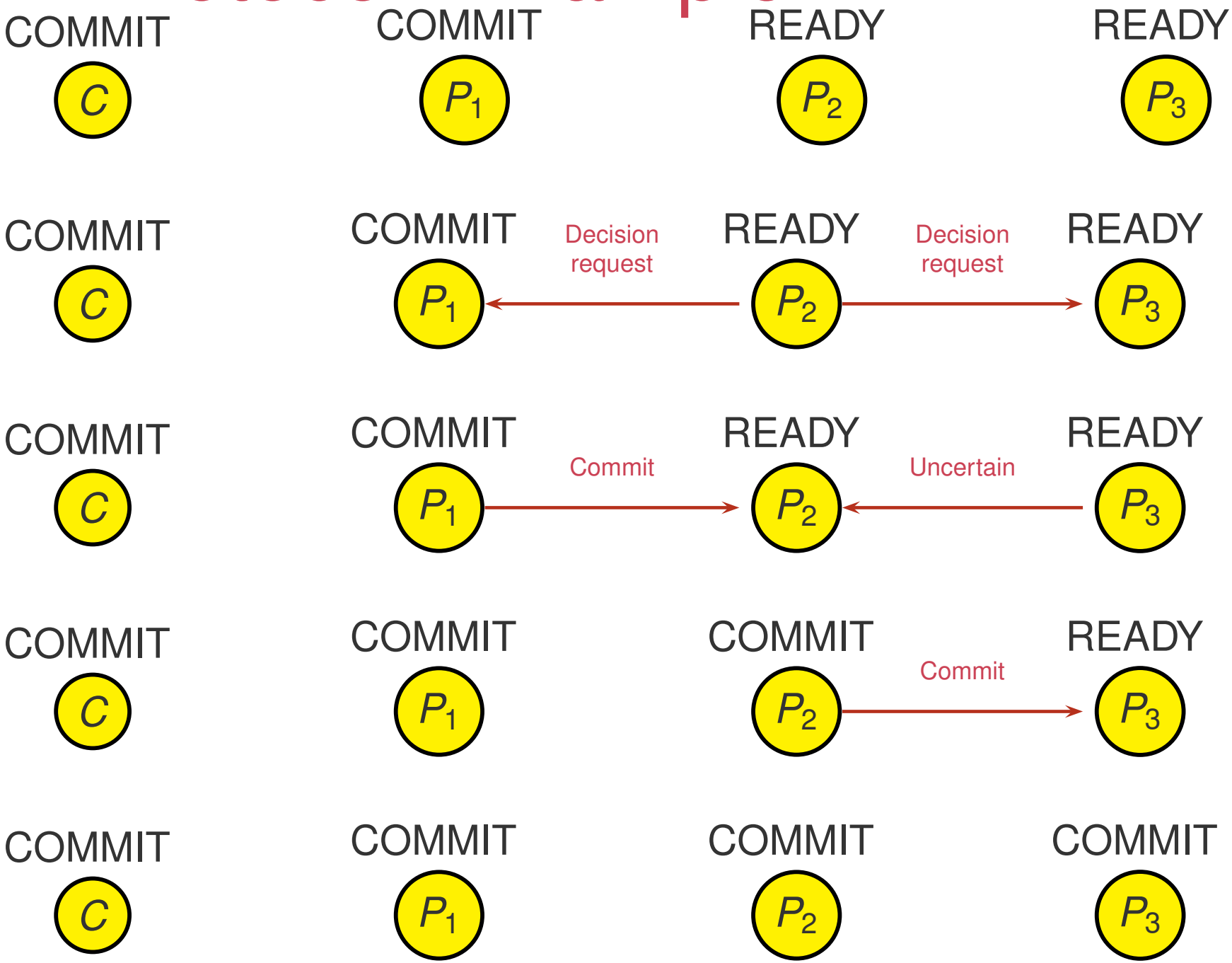
Q's state	Q's actions
INITIAL	Unilaterally aborts the transaction; replies "Abort"
READY	Replies "Uncertain"
COMMIT	Replies "Commit"
ABORT	Replies "Abort"

- If any participant replies with "Commit"/"Abort" to *P*
  - *P* terminates the transaction with that decision
  - *P* sends the decision to every participant that replied "Uncertain"

CT Protocol: Example 1



CT Protocol: Example 2



Optimizations of 2PC Protocol

- Complexity of 2PC Protocol:
  - Number of transmitted messages
  - Number of forced writes of log records
- Optimizations to reduce complexity/blocking
  - Presumed Abort 2PC (PA-2PC) Protocol
  - Presumed Commit 2PC (PC-2PC) Protocol
  - Three-Phase Commit (3PC) Protocol
- Both **Basic Termination Protocol** & **Cooperative Termination Protocol** in 2PC are blocking protocols

Examples of Blocking in 2PC

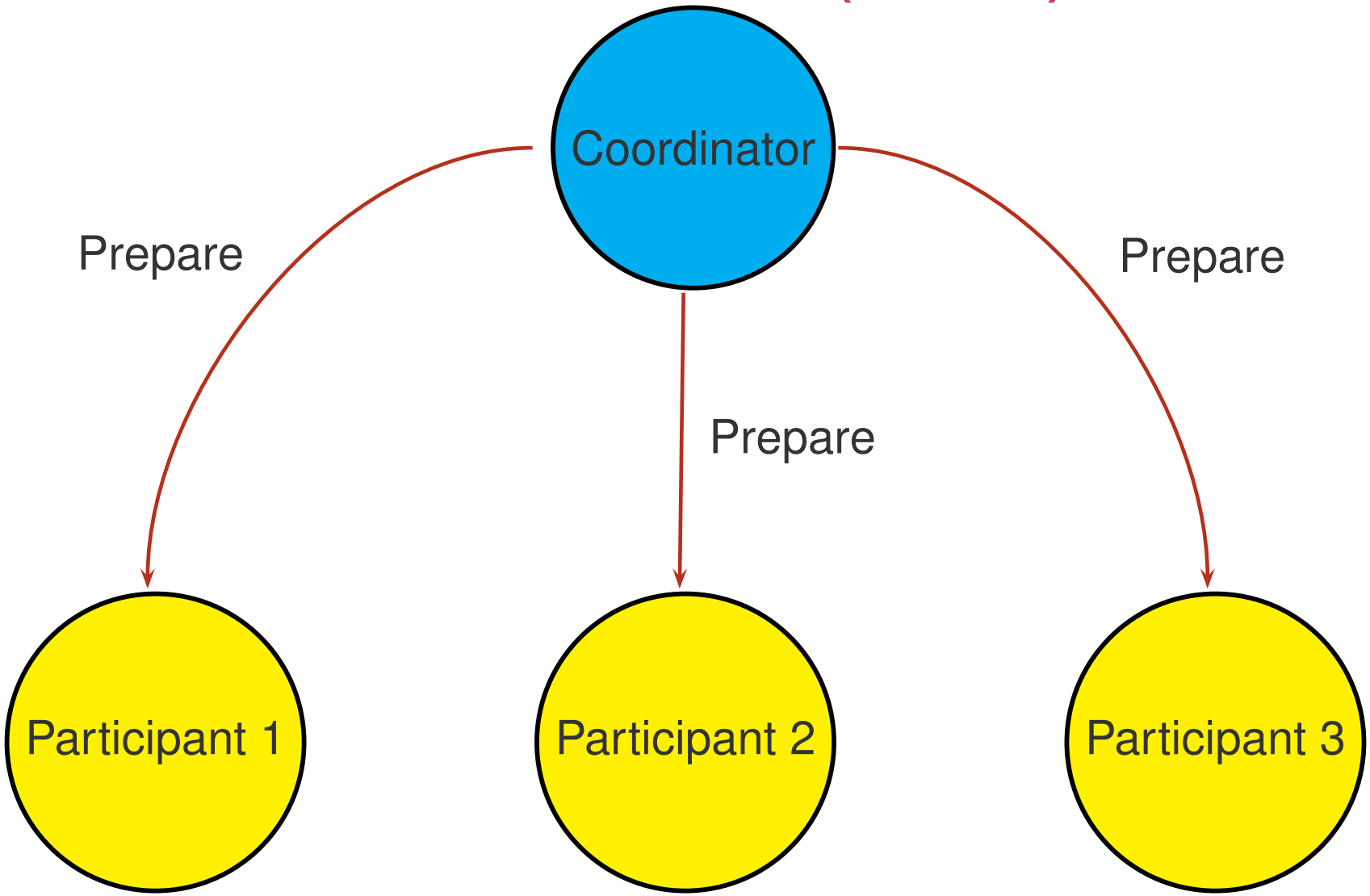
Scenario 1: *C* fails after sending Global-commit, P_1 fails after receiving Global-commit, P_2 & P_3 did not receive Global-commit

Scenario 2: *C* fails before receiving any vote & P_1 fails after voting

Three-Phase Commit (3PC) Protocol

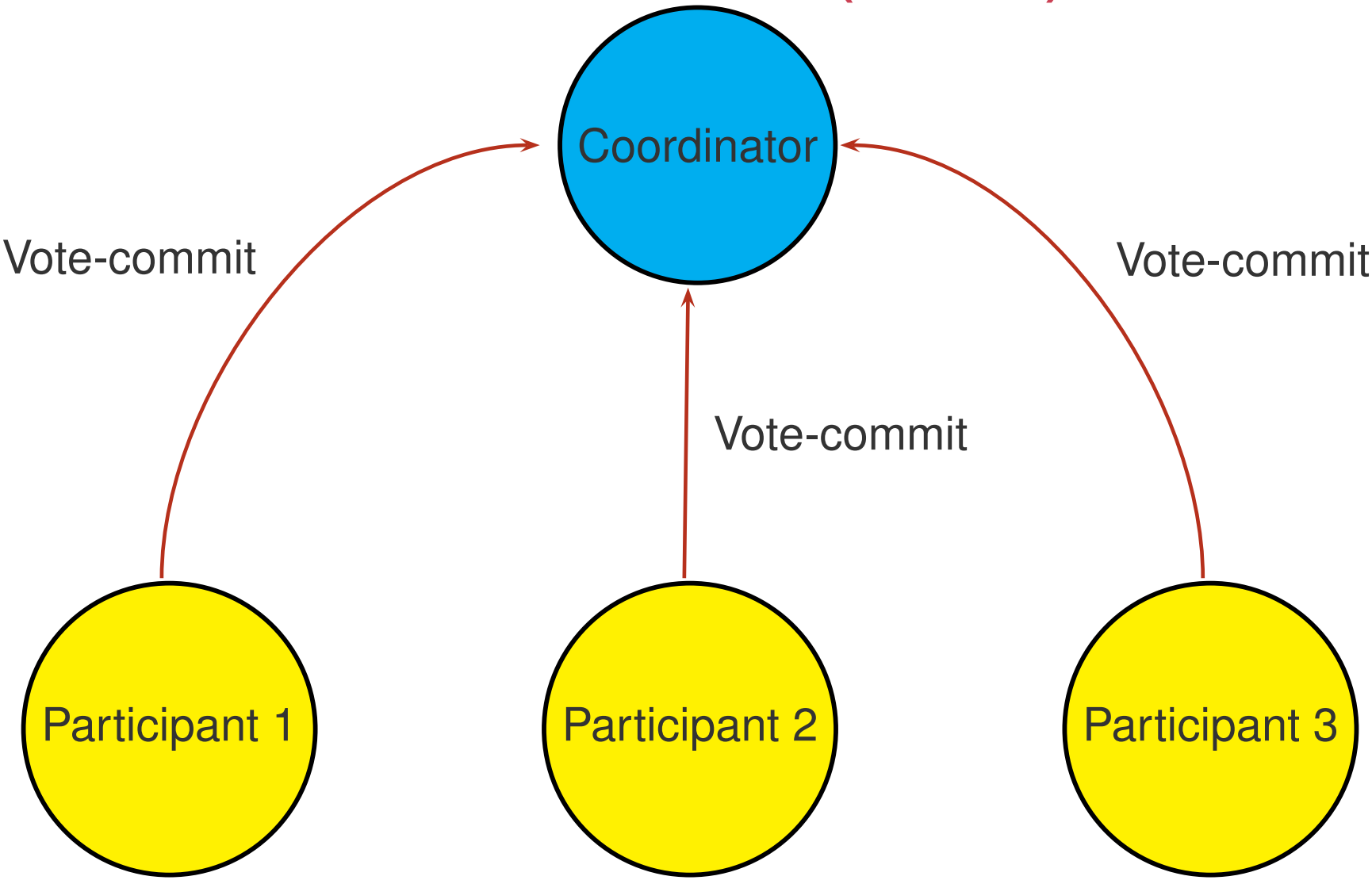
- Goal: Reduce the probability of blocking
  - In the absence of communication failure & total site failure, 3PC is non-blocking
- **First phase**: Voting/Preparation phase
  - Coordinator collects votes from participants
- **Second phase**: Dissemination phase
  - Coordinator disseminates voting outcome to participants if there is no abort vote
- **Third phase**: Decision phase
  - Coordinator sends global decision to participants

Three-Phase Commit (3PC) Protocol



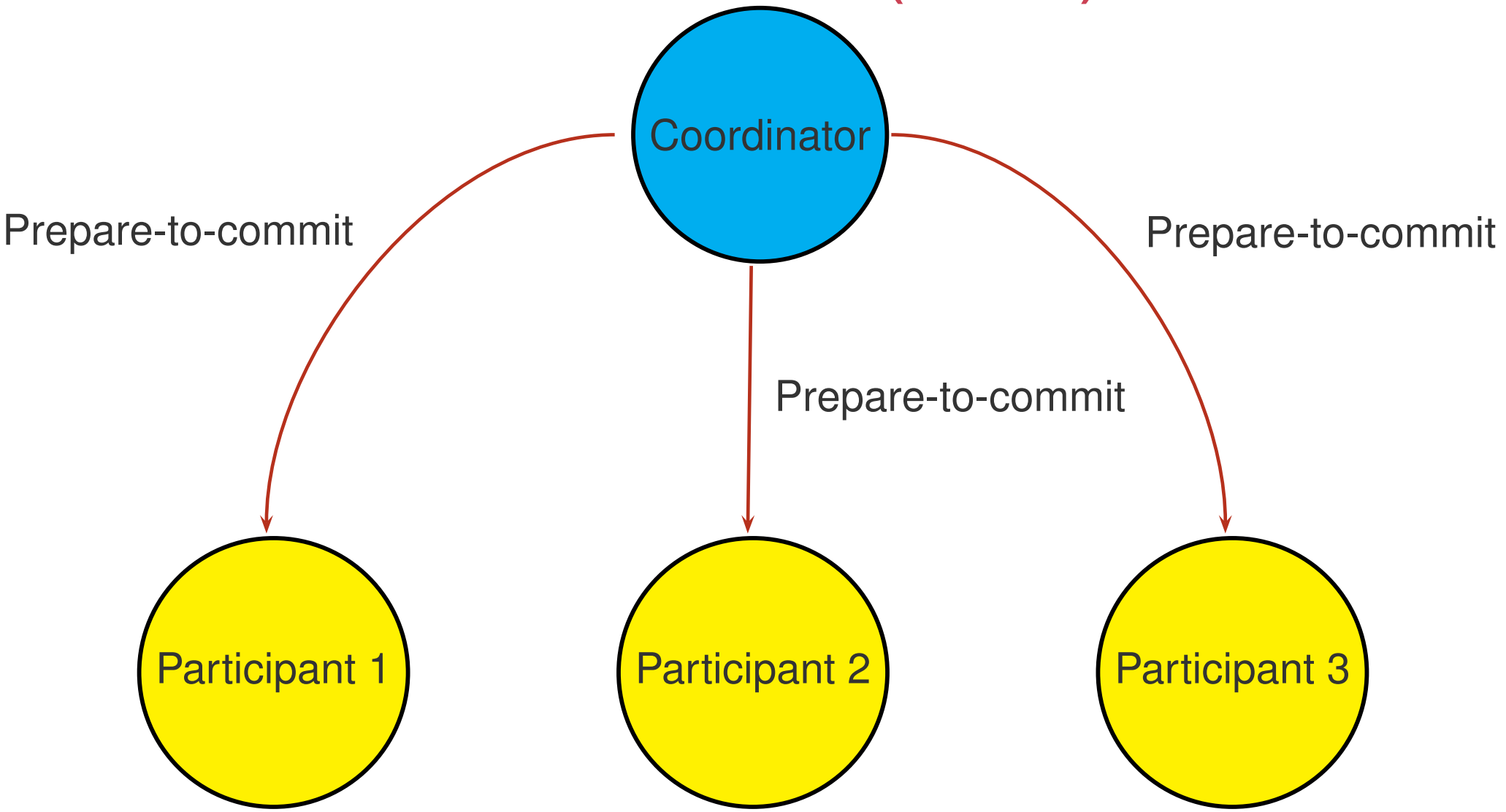
Phase 1: Voting Phase

Three-Phase Commit (3PC) Protocol



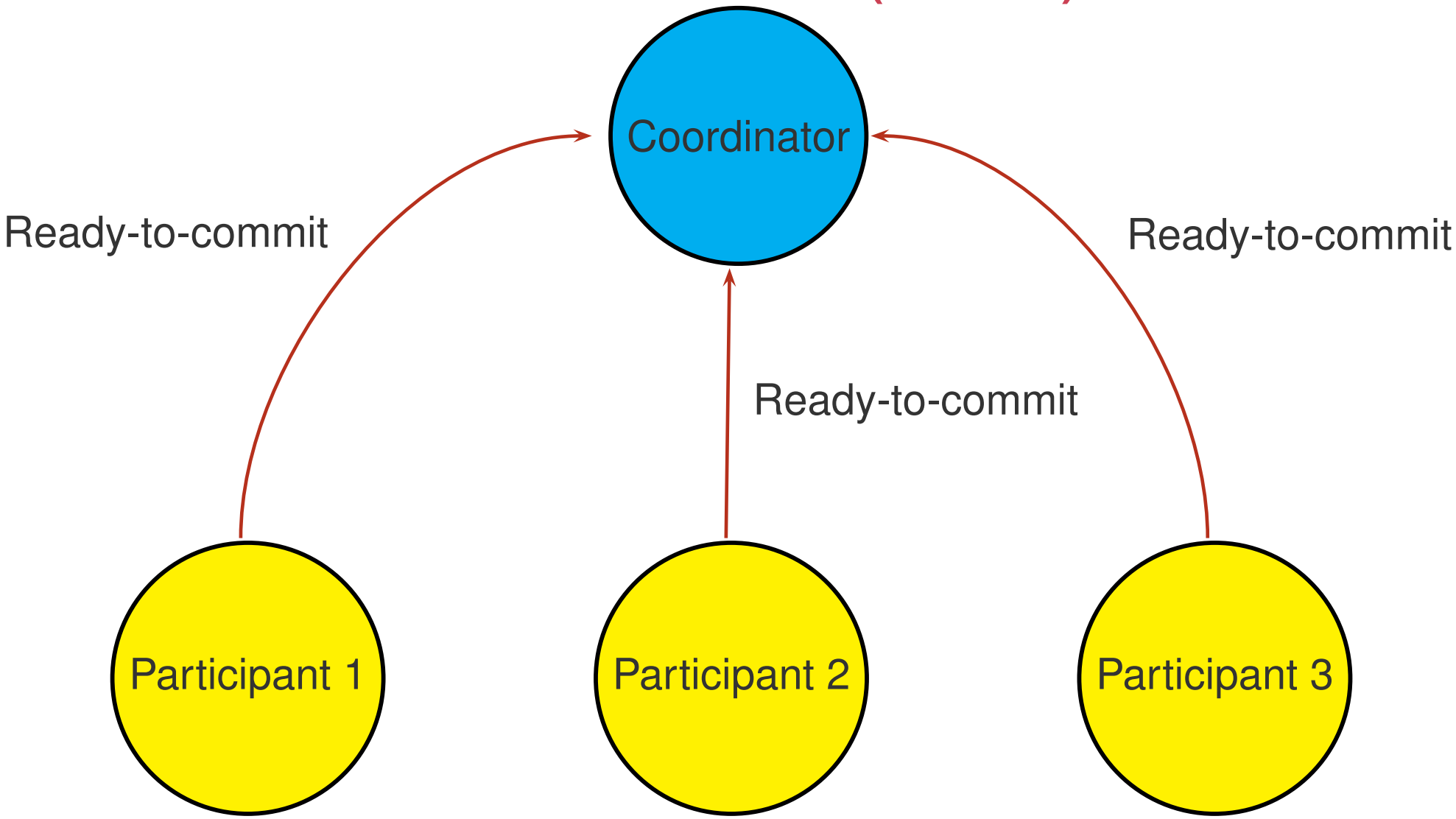
Phase 1: Voting Phase

Three-Phase Commit (3PC) Protocol



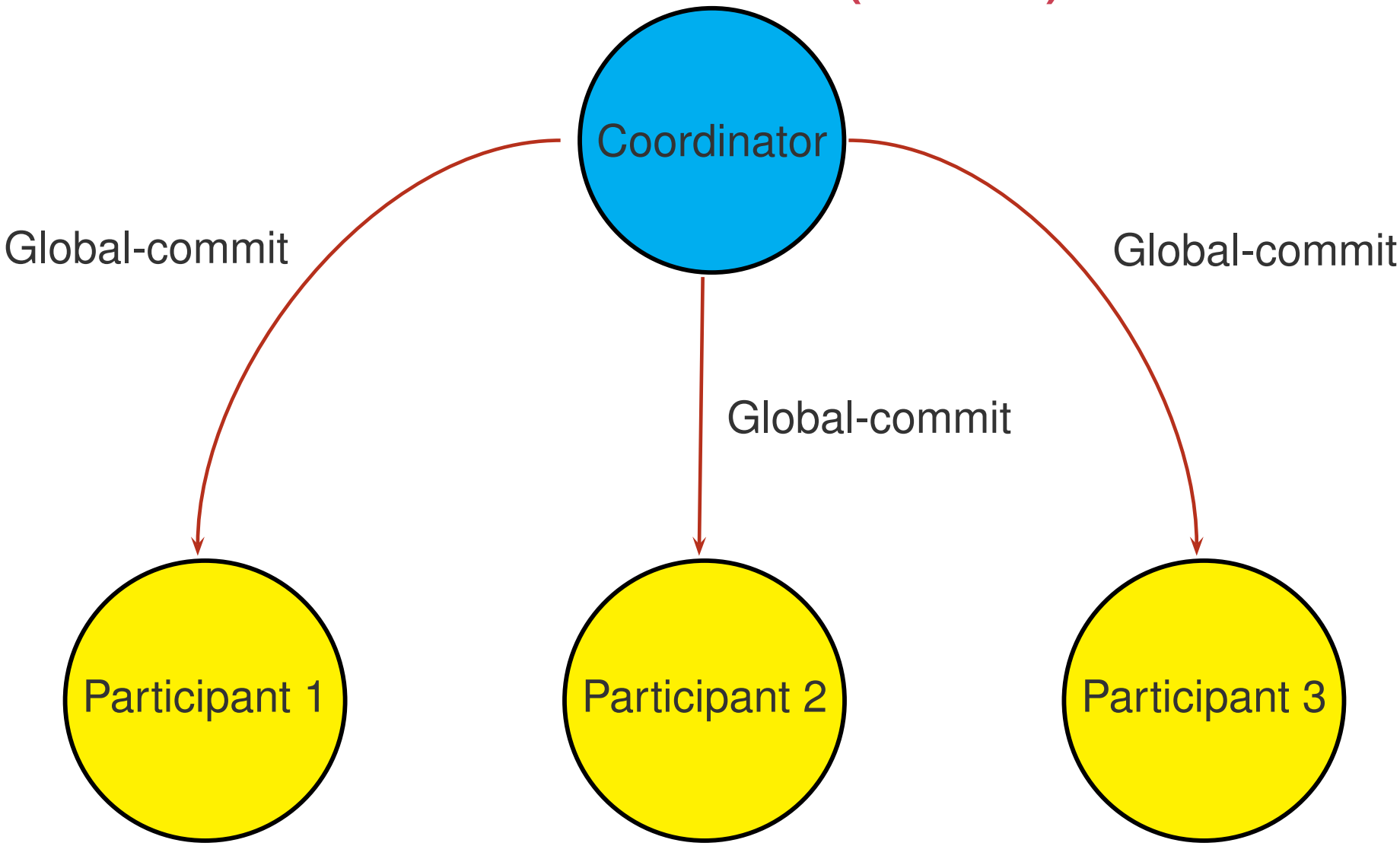
Phase 2: Dissemination Phase (commit scenario)

Three-Phase Commit (3PC) Protocol



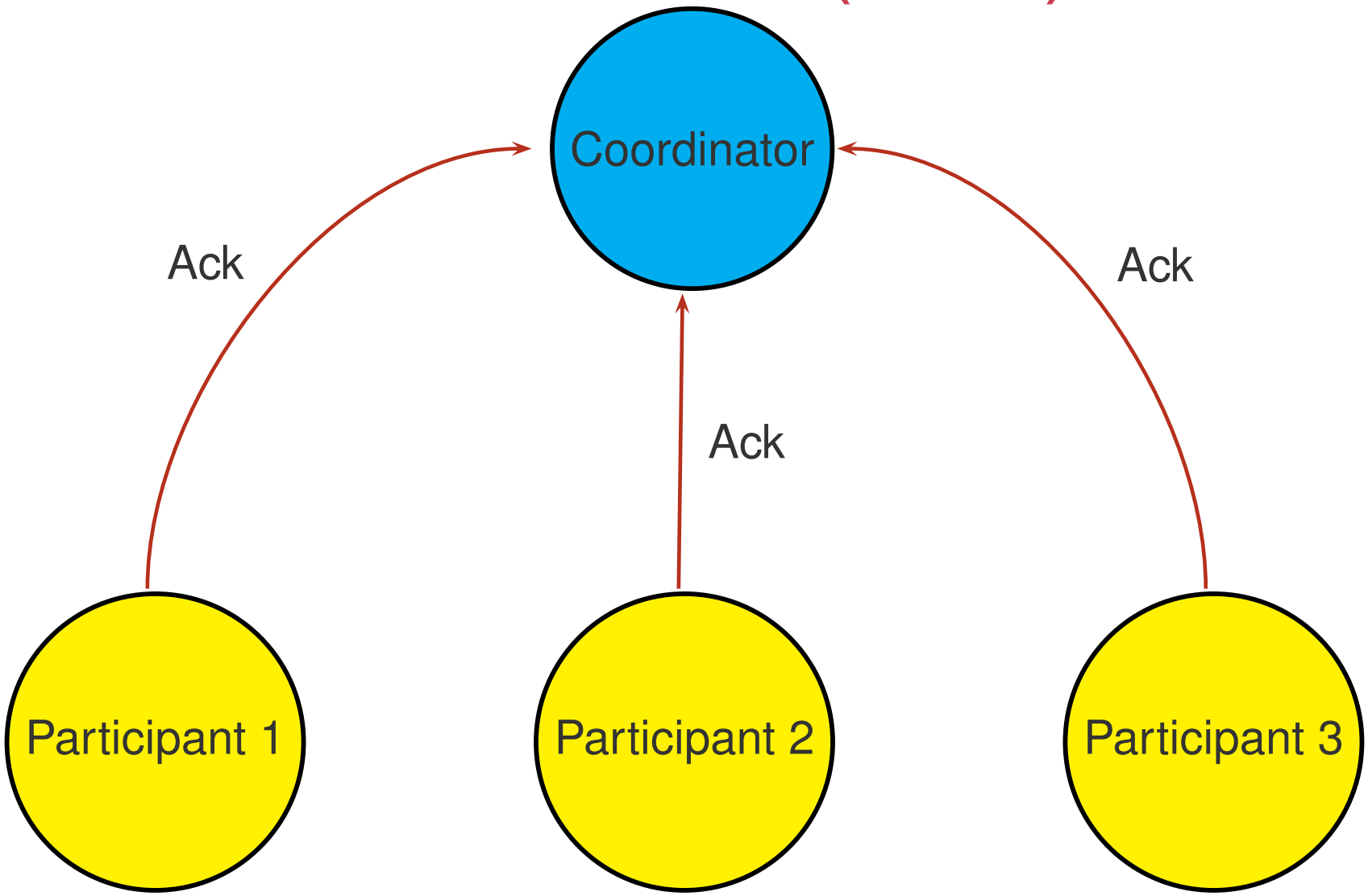
Phase 2: Dissemination Phase (commit scenario)

Three-Phase Commit (3PC) Protocol



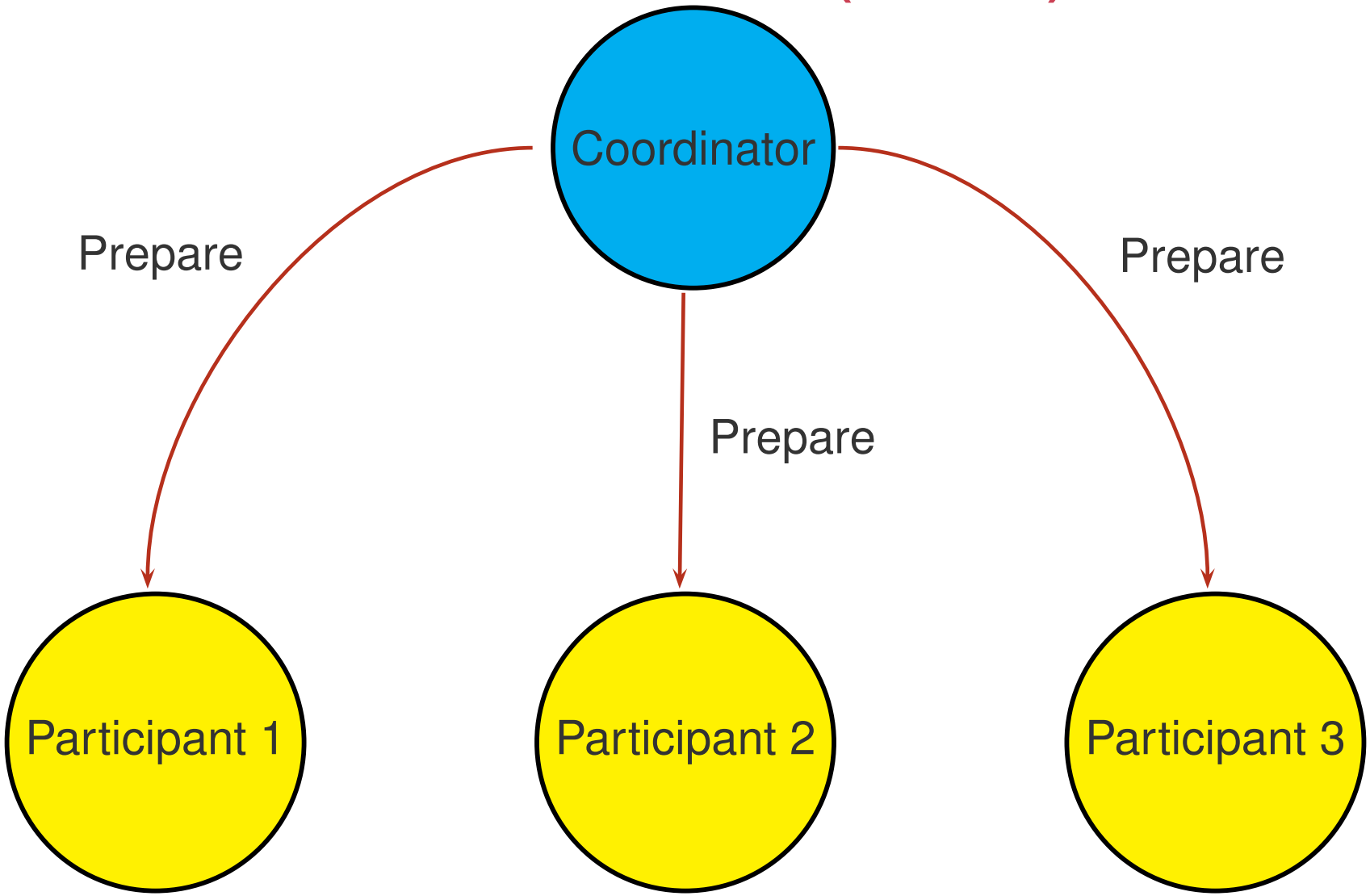
Phase 3: Decision Phase (commit scenario)

Three-Phase Commit (3PC) Protocol



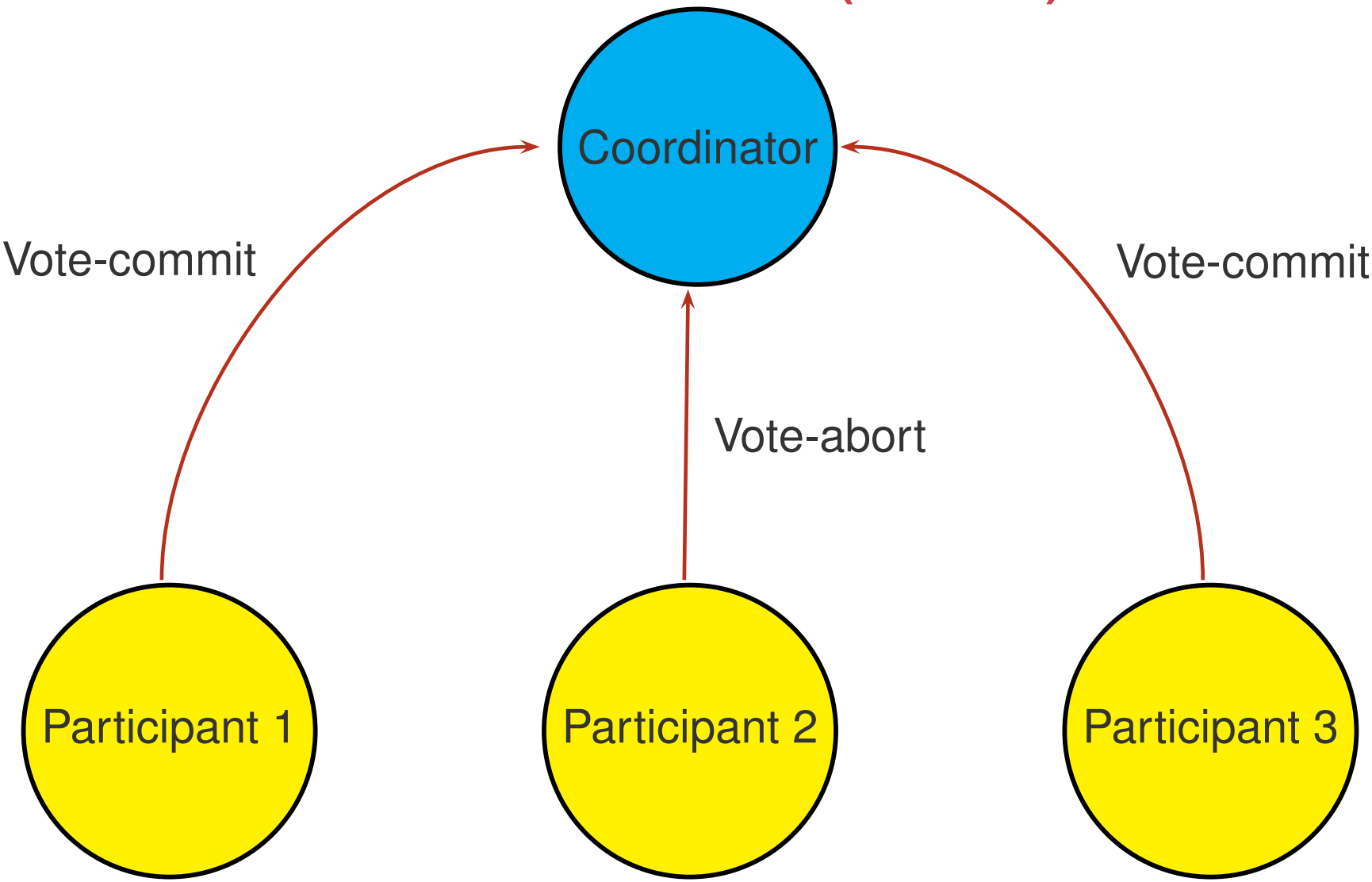
Phase 3: Decision Phase (commit scenario)

Three-Phase Commit (3PC) Protocol



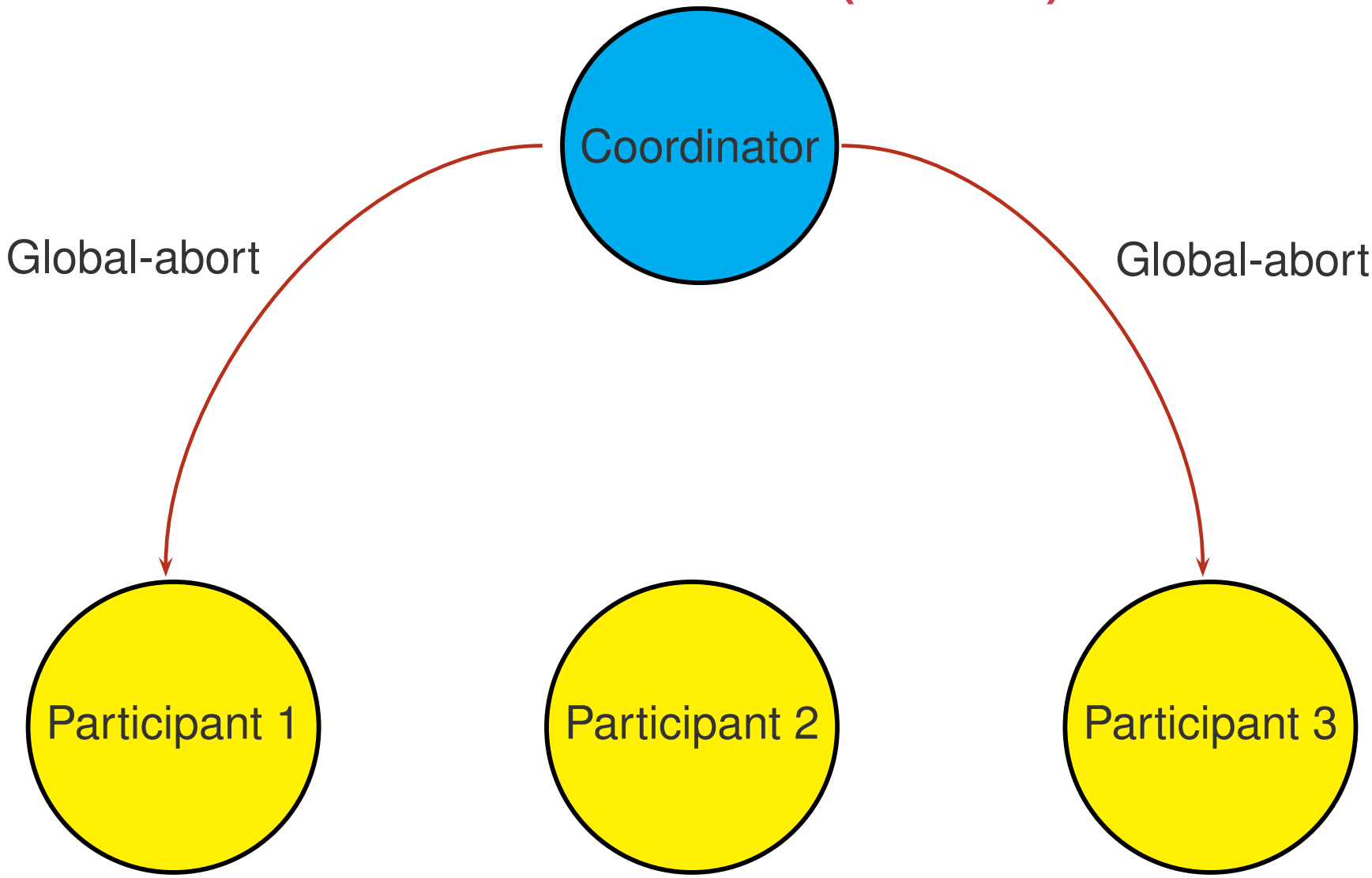
Phase 1: Voting Phase

Three-Phase Commit (3PC) Protocol



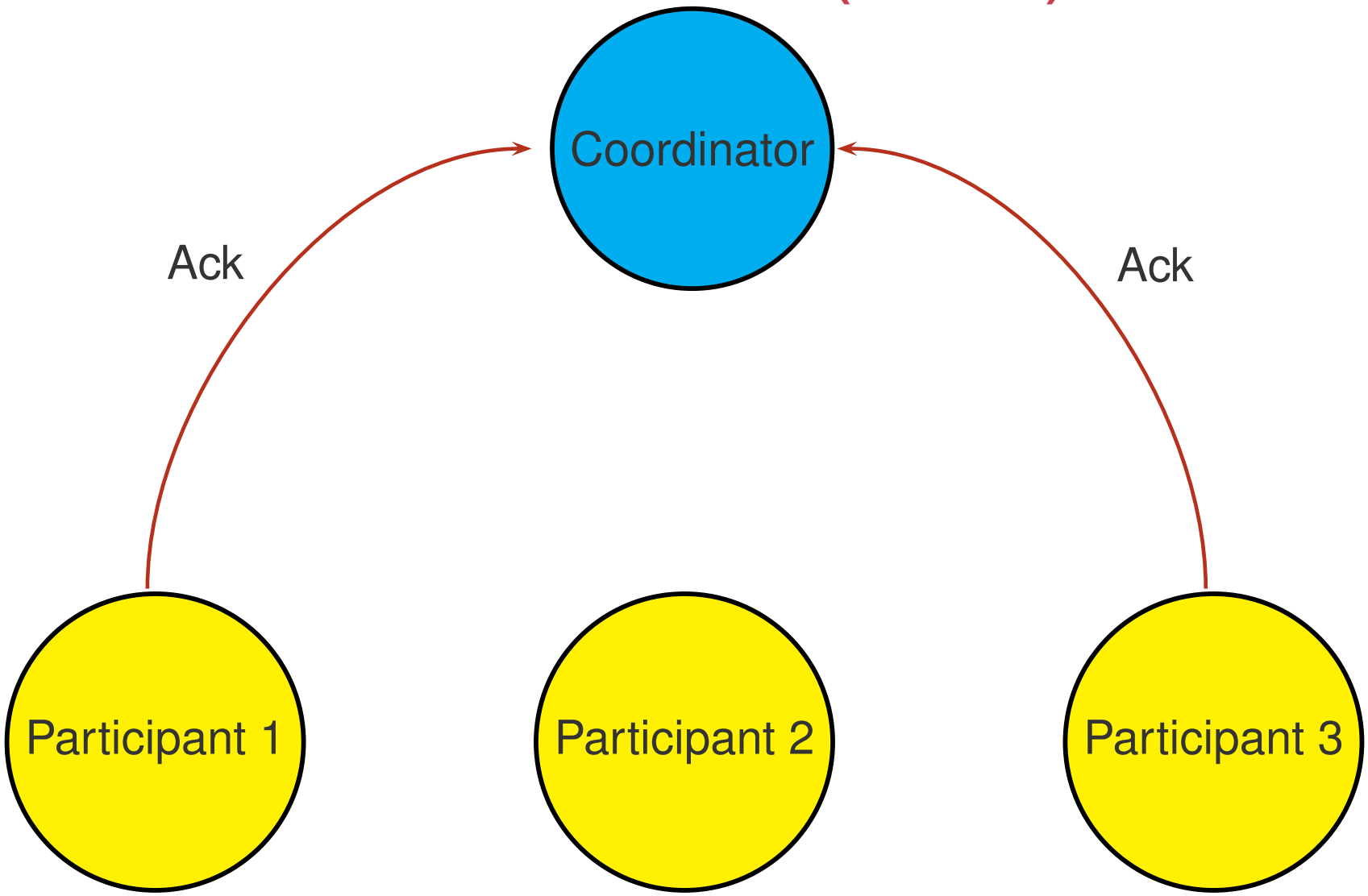
Phase 1: Voting Phase

Three-Phase Commit (3PC) Protocol



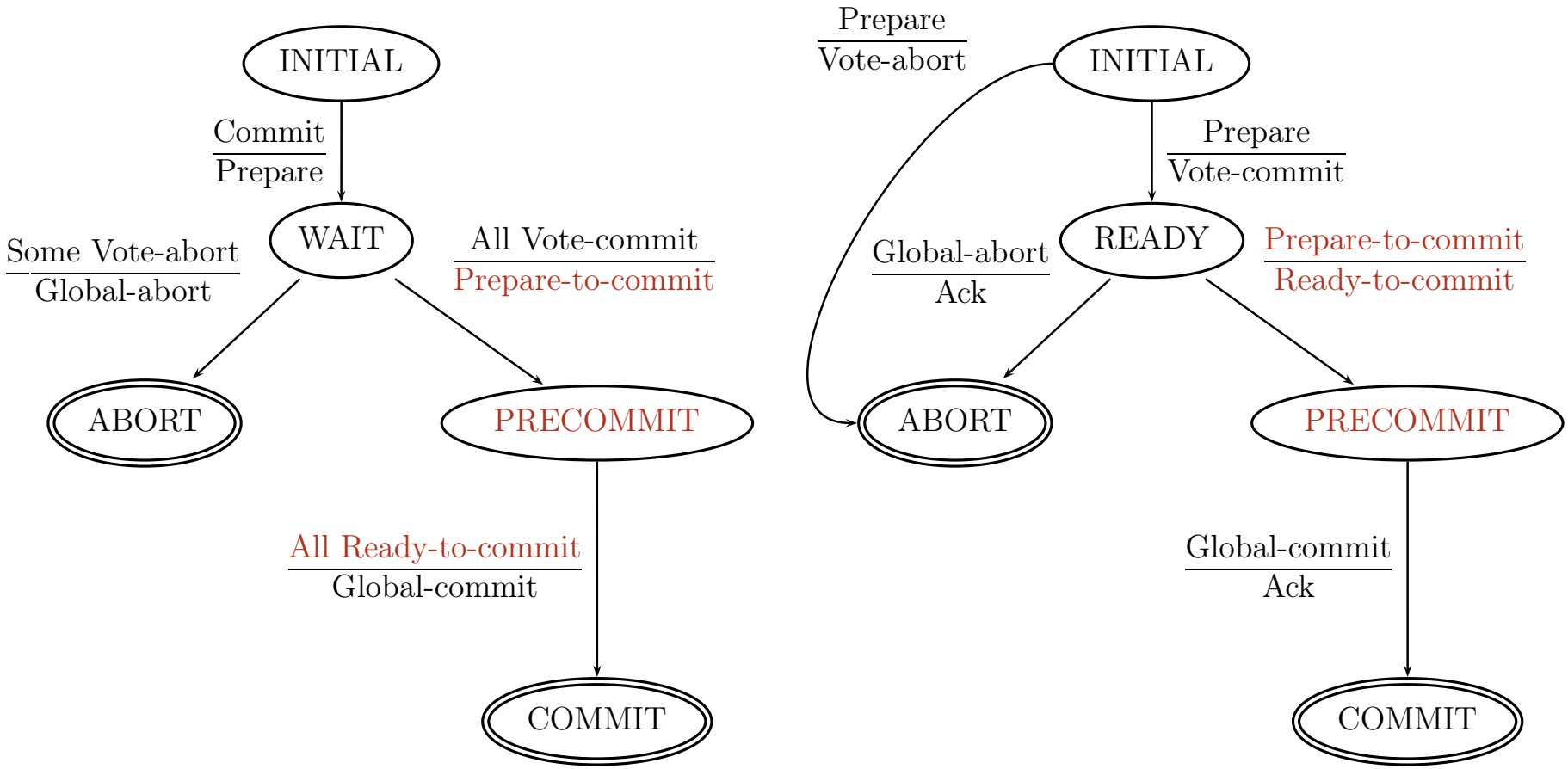
Phase 2: Decision Phase (abort scenario)

Three-Phase Commit (3PC) Protocol

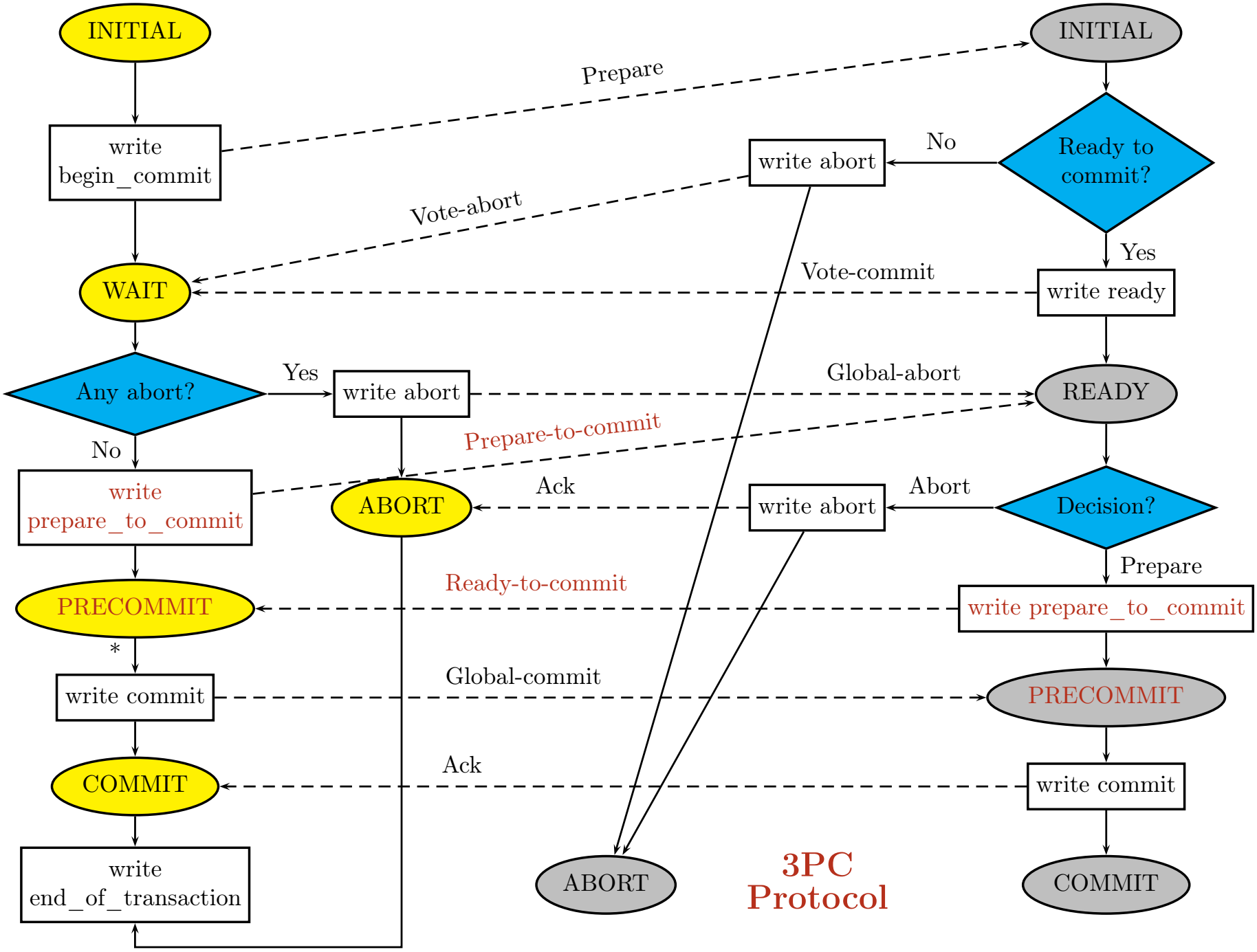


Phase 2: Decision Phase (abort scenario)

State Transition Diagrams for 3PC

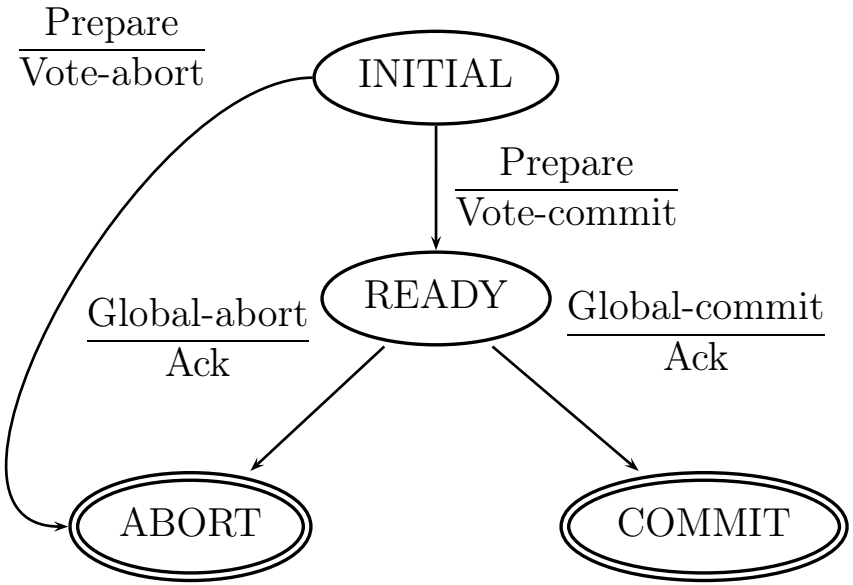


Coordinator **Participant**

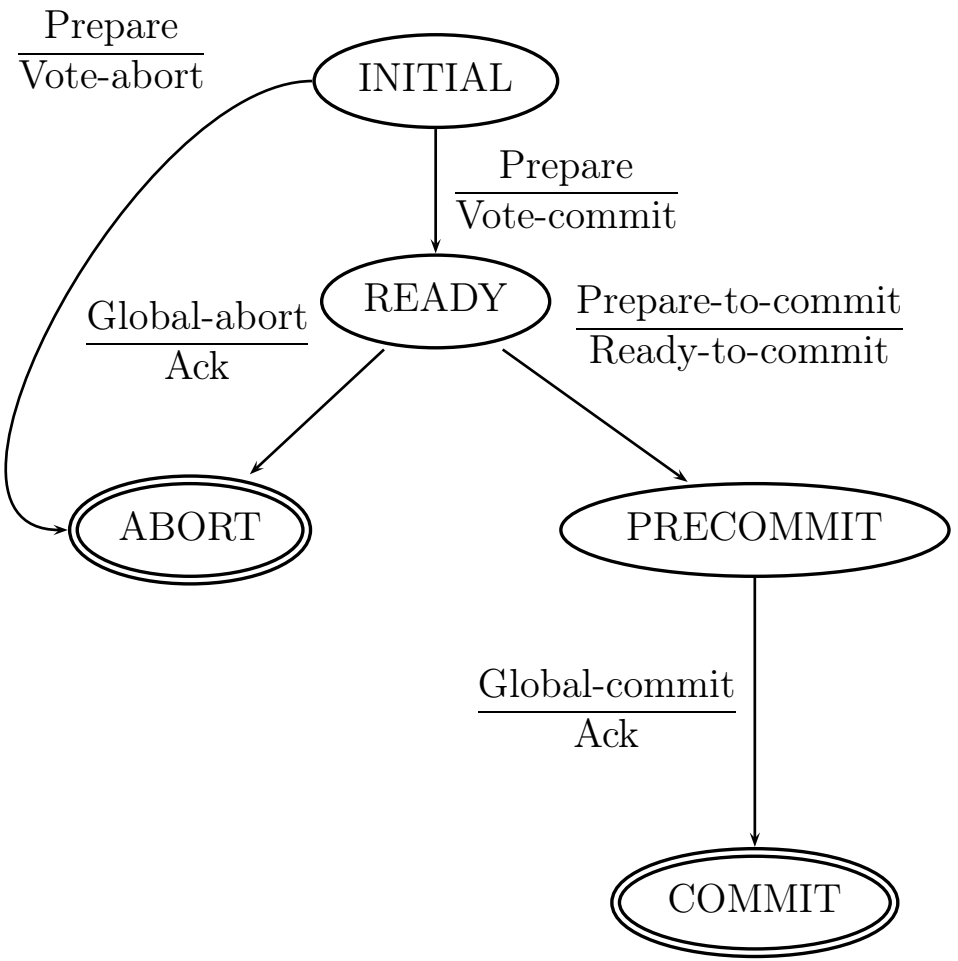


* When coordinator has received Ready-to-commit from all participants

2PC vs 3PC



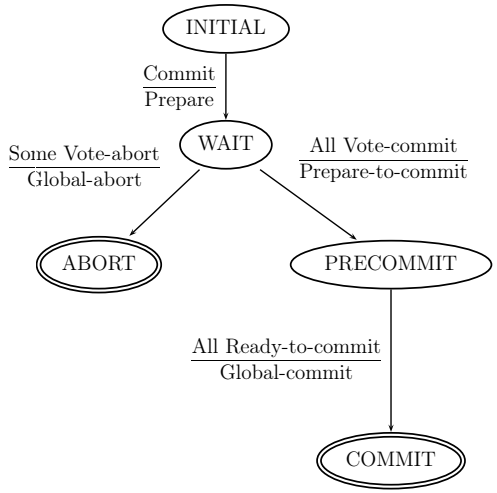
2PC Protocol



3PC Protocol

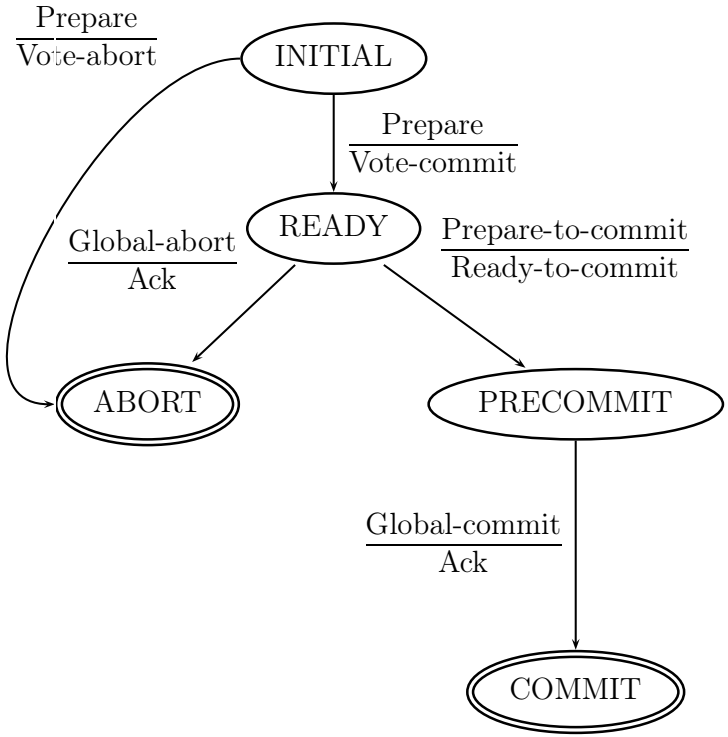
Termination Protocol for Coordinator

Timeout in state	Coordinator's termination actions
WAIT	Writes an abort record in log & sends "Global-abort" to all participants
PRECOMMIT	Writes a commit record in log & sends "Global-commit" to operational participants
ABORT	Sends "Global-abort" to participants who have not responded
COMMIT	Sends "Global-commit" to participants who have not responded

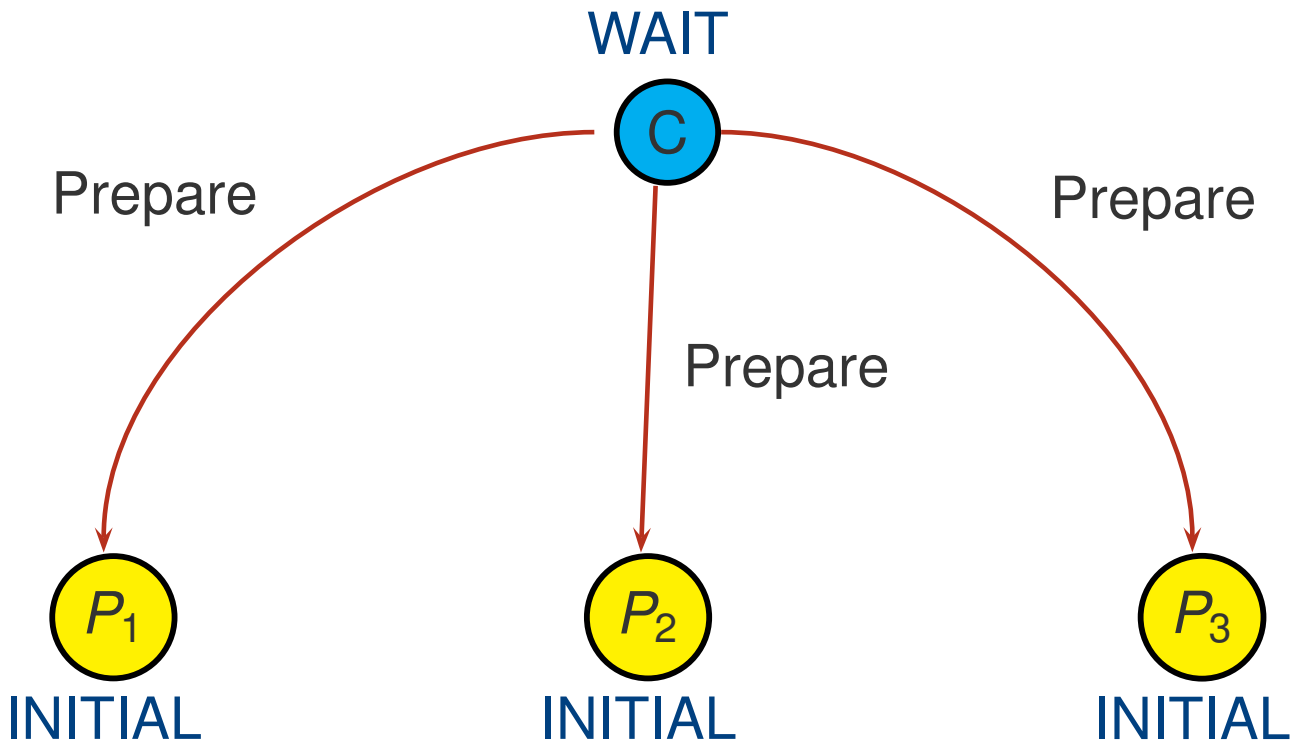


Termination Protocol for Participants

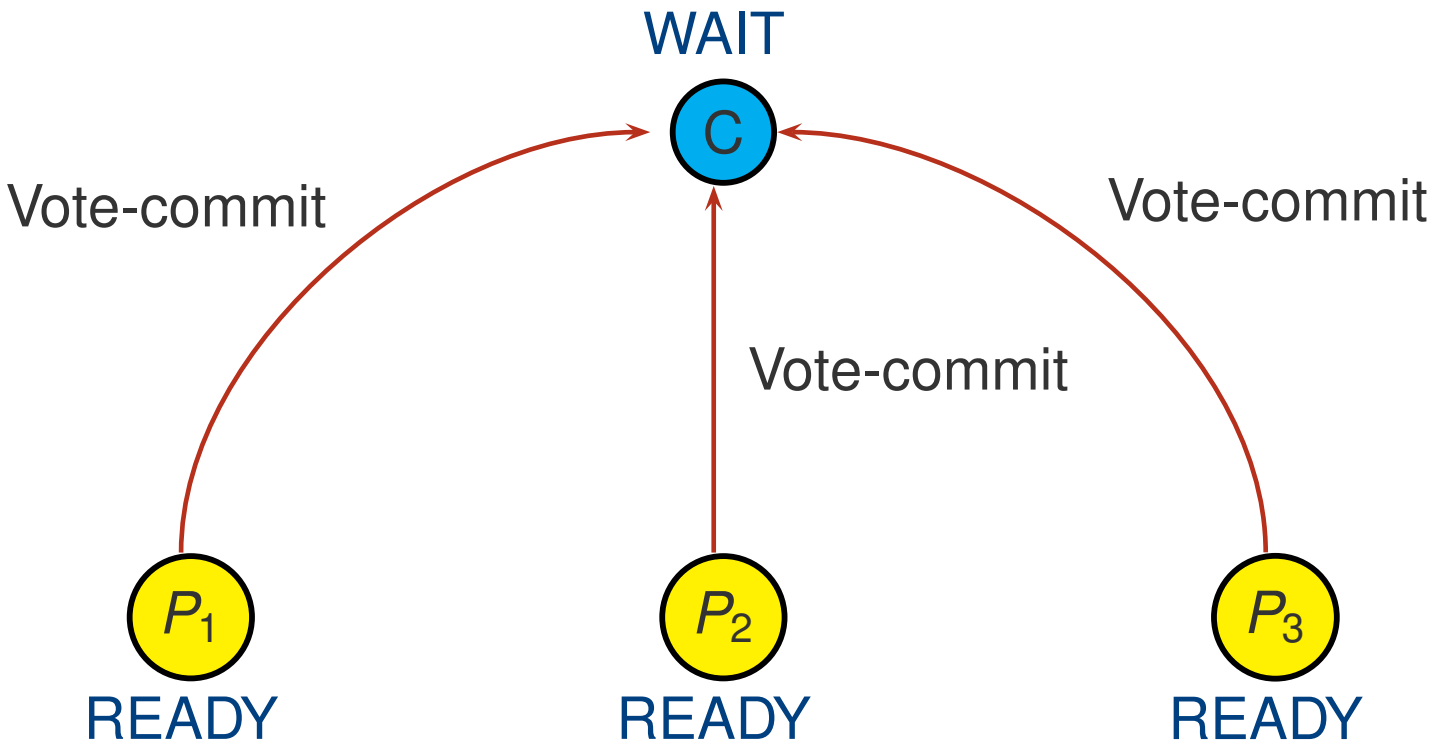
Timeout in state	Participant's termination actions
INITIAL	Aborts transaction unilaterally
READY	Executes Termination Protocol 1
PRECOMMIT	Executes Termination Protocol 1



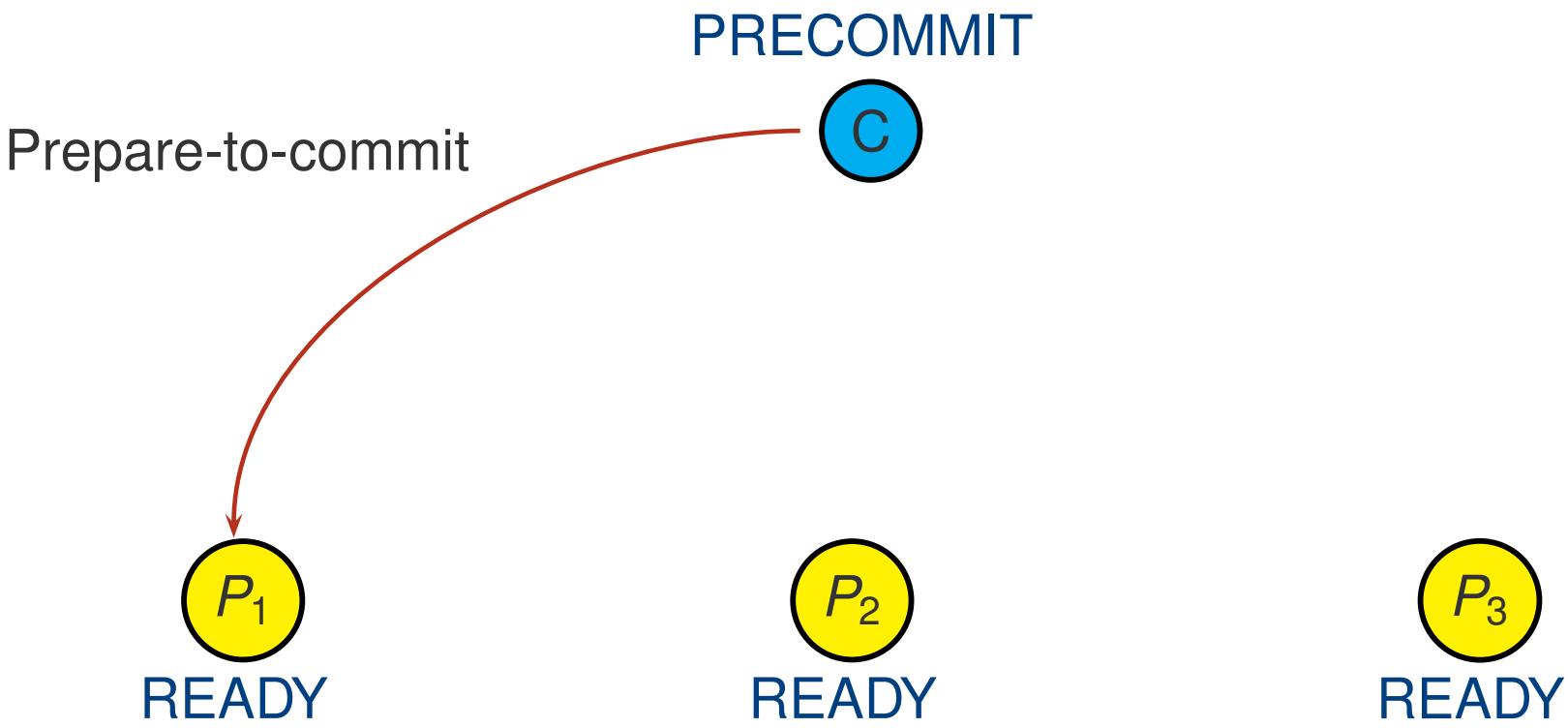
Example: 3PC Termination Protocol 1



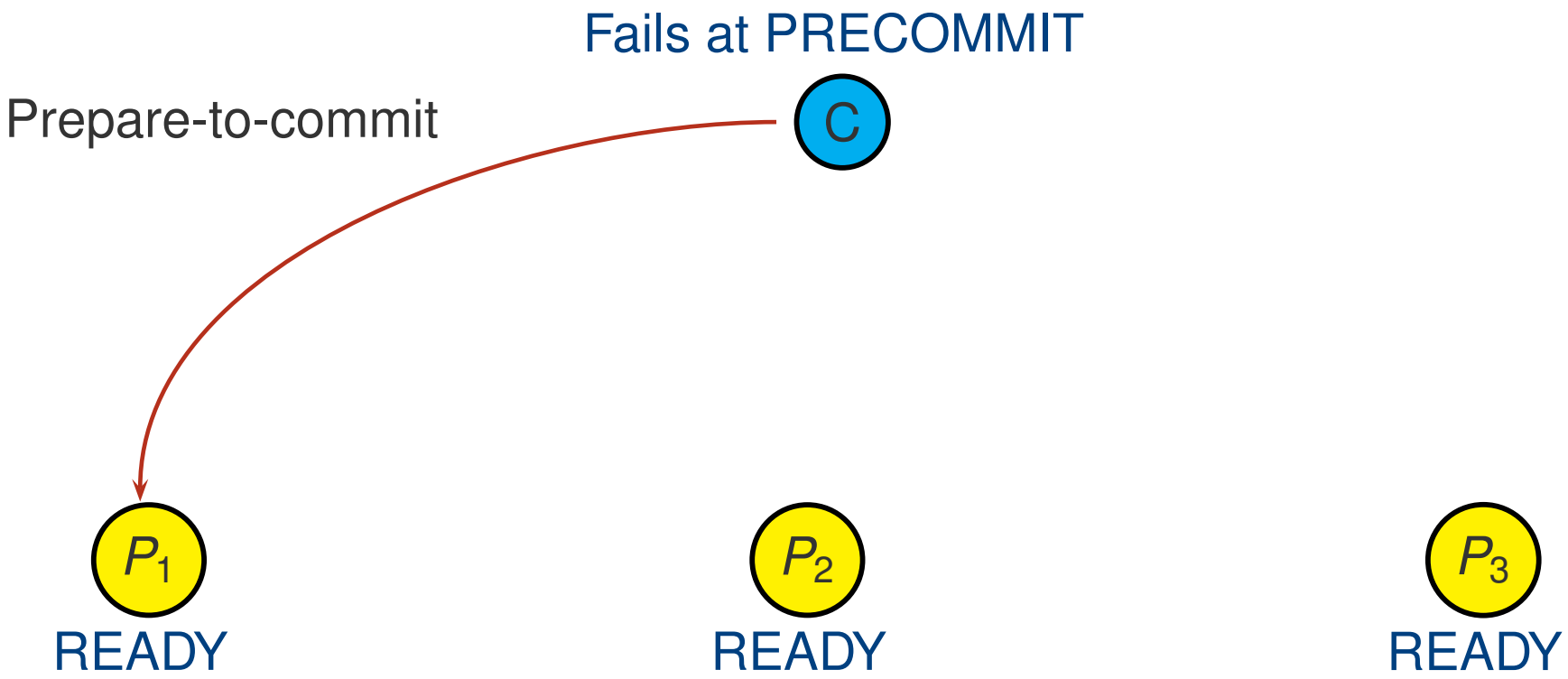
Example: 3PC Termination Protocol 1



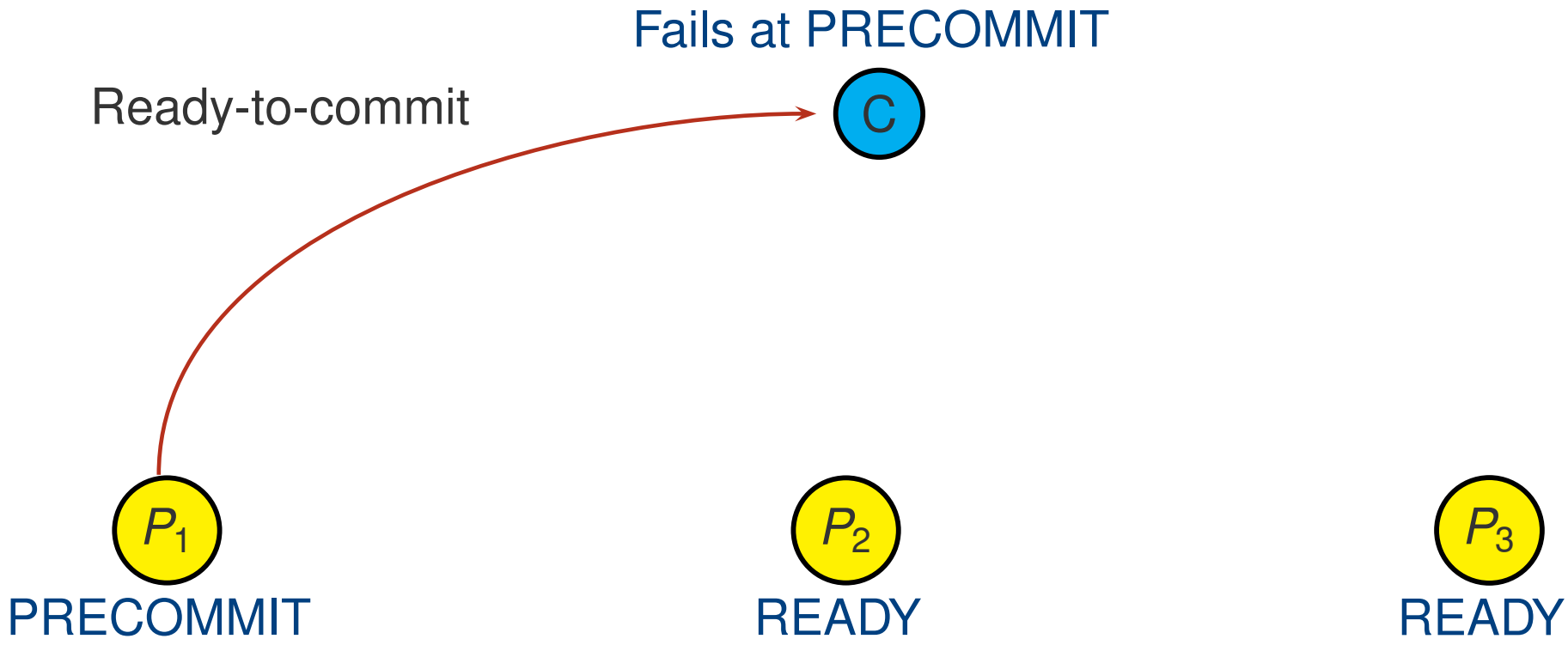
Example: 3PC Termination Protocol 1



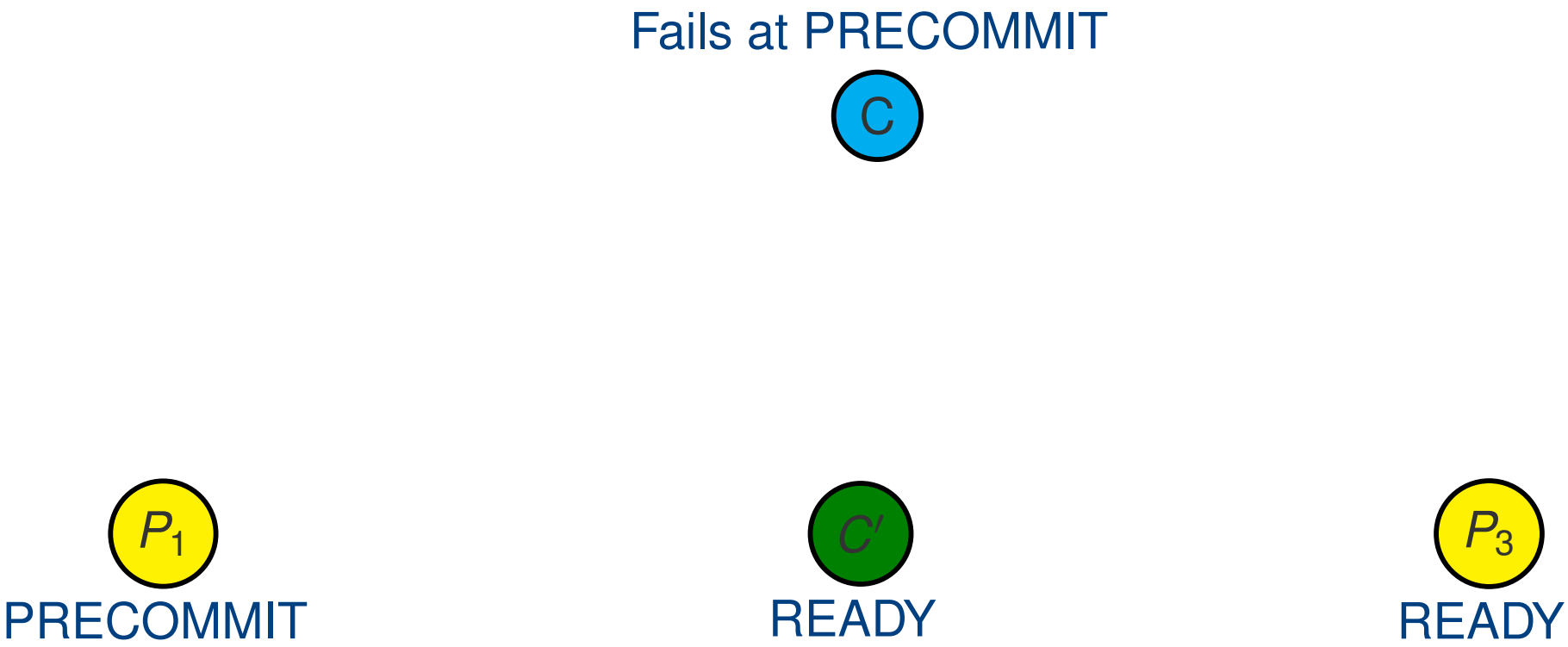
Example: 3PC Termination Protocol 1



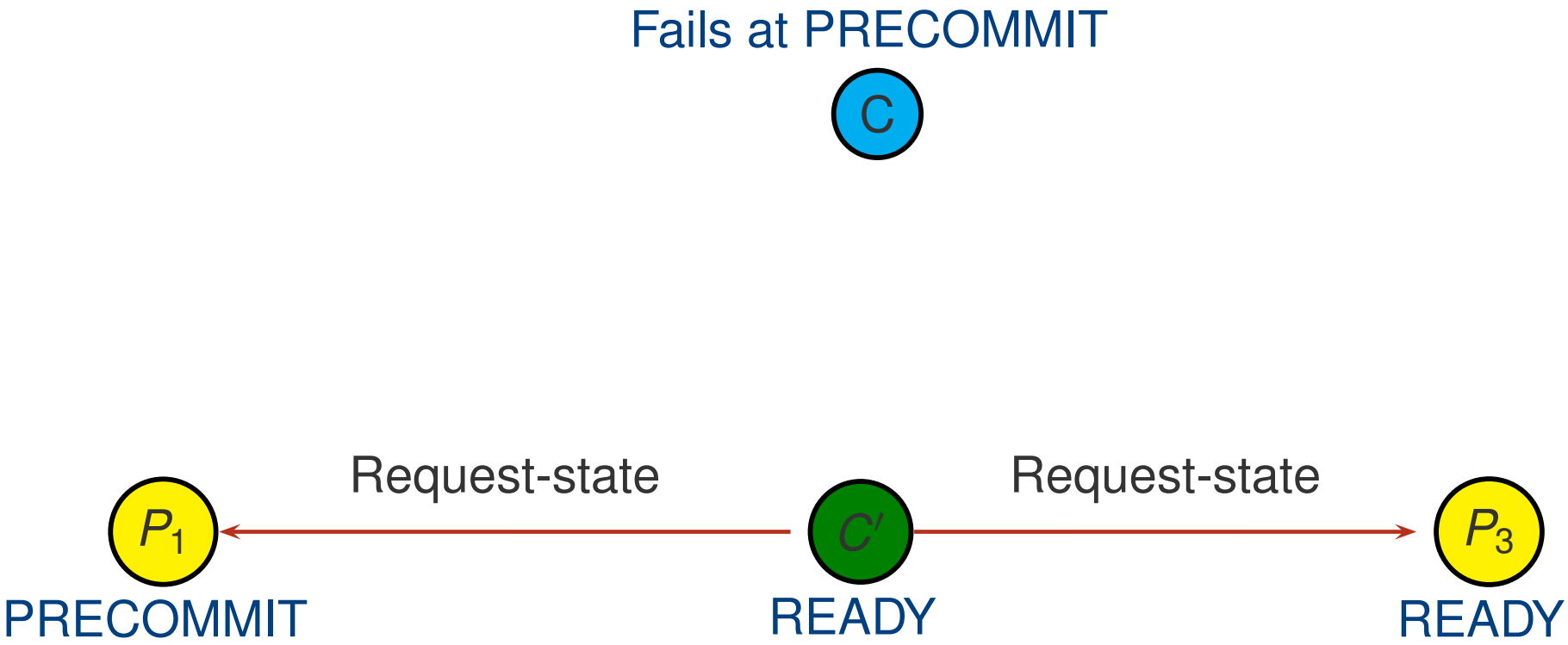
Example: 3PC Termination Protocol 1



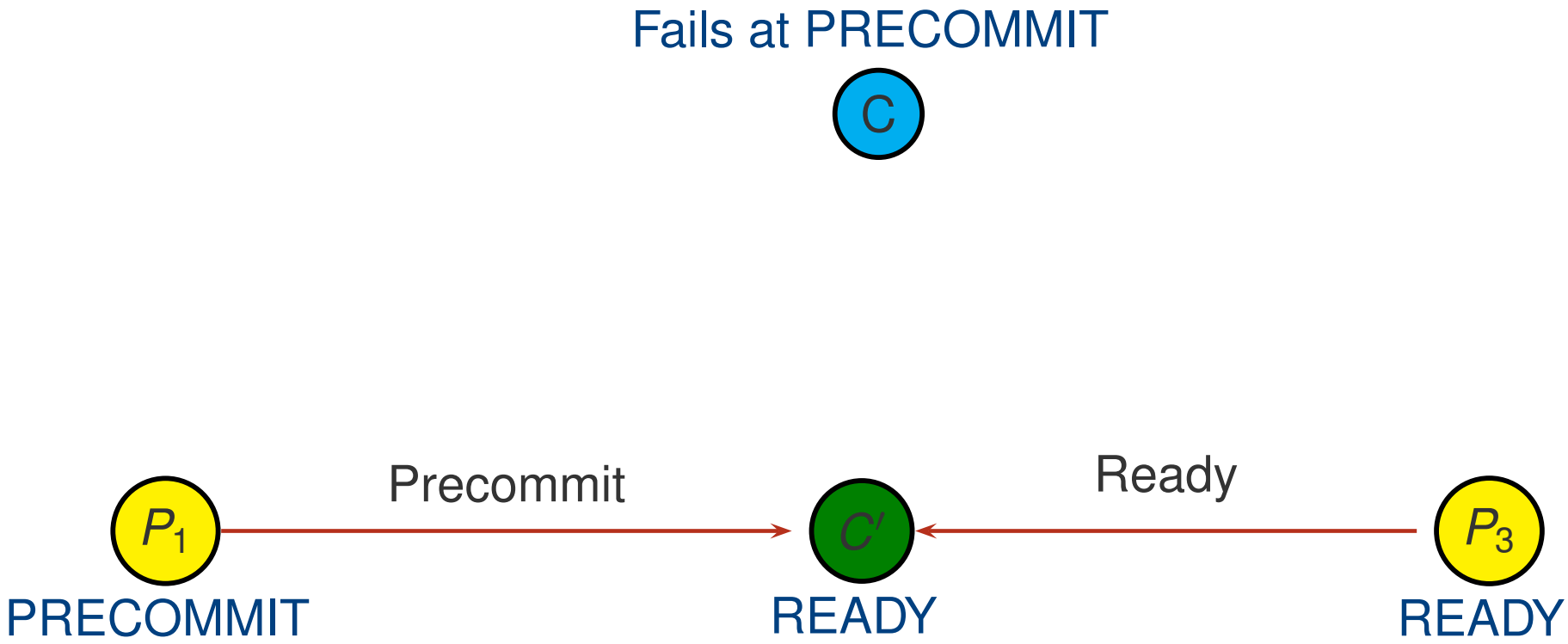
Example: 3PC Termination Protocol 1



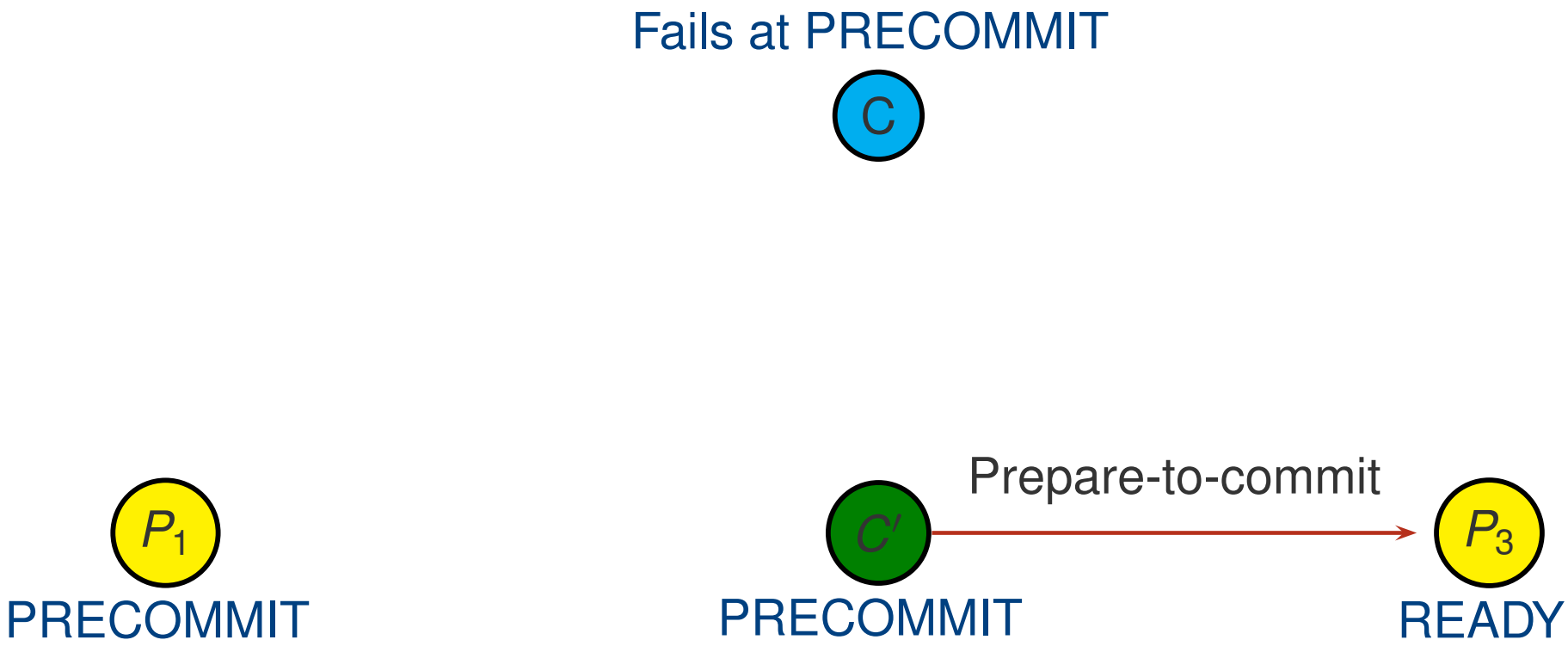
Example: 3PC Termination Protocol 1



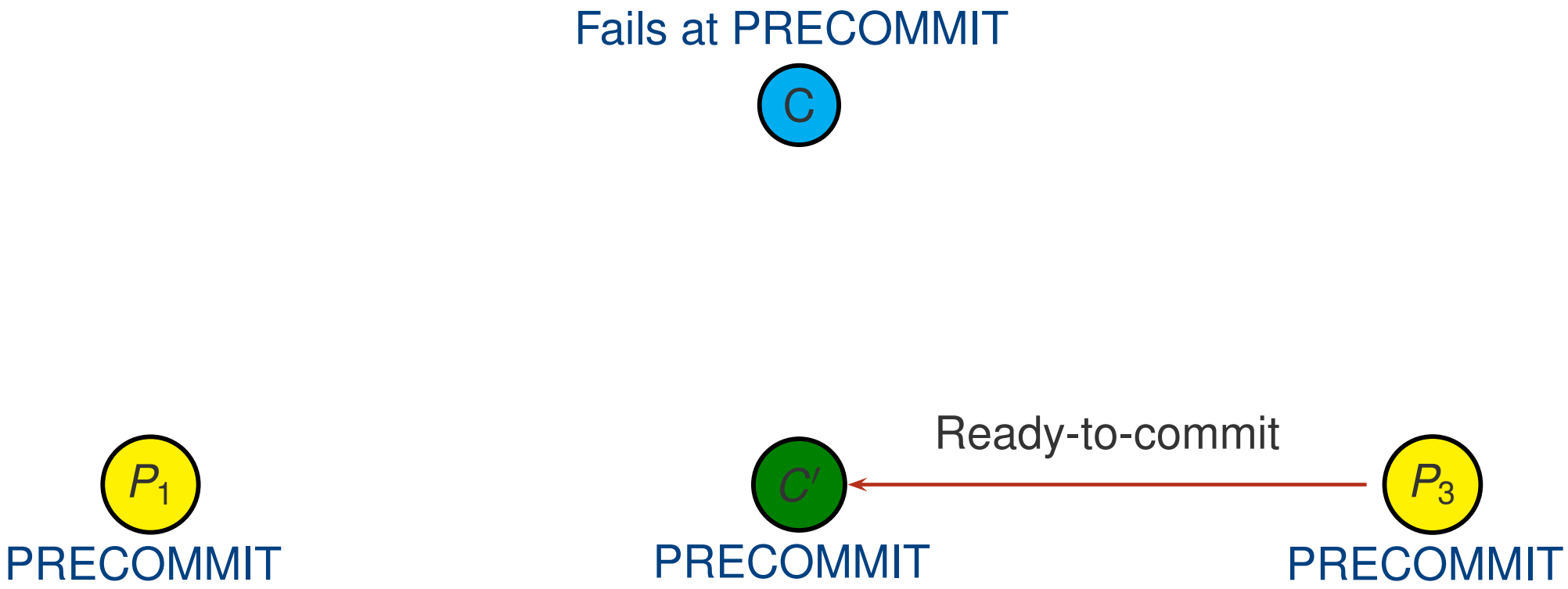
Example: 3PC Termination Protocol 1



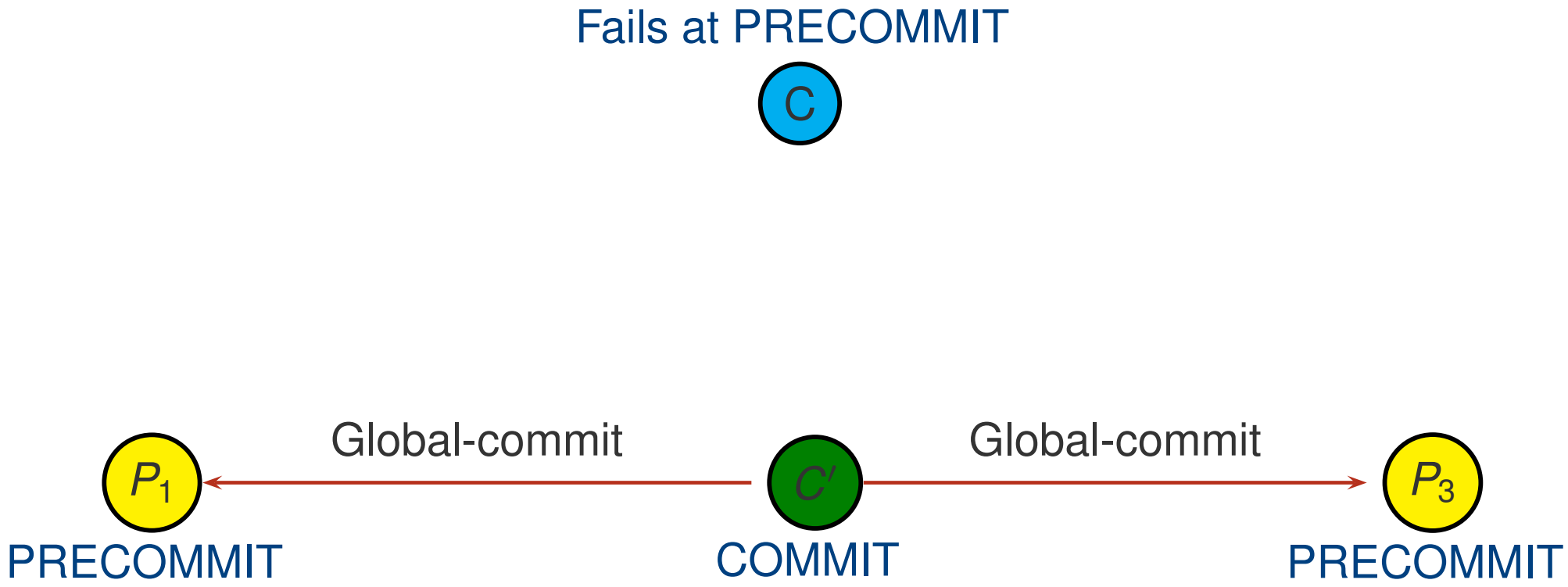
Example: 3PC Termination Protocol 1



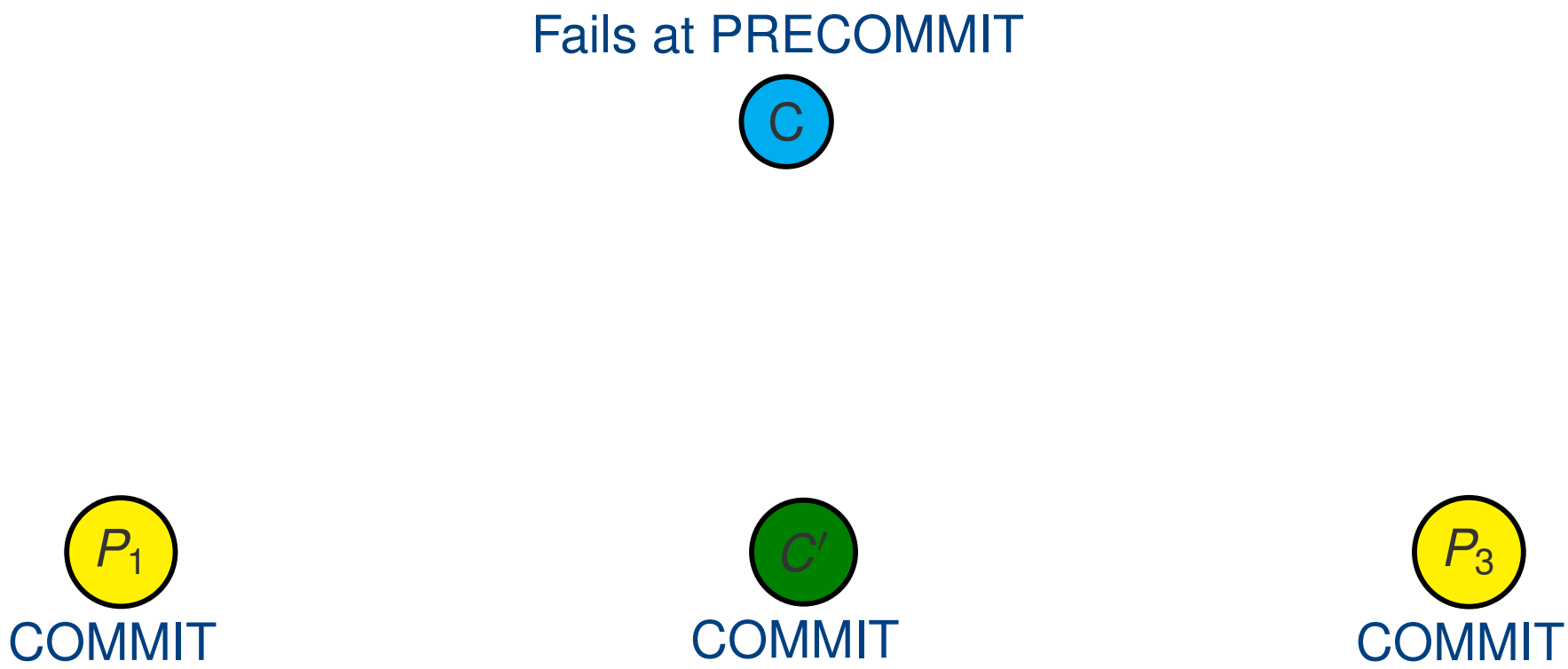
Example: 3PC Termination Protocol 1



Example: 3PC Termination Protocol 1



Example: 3PC Termination Protocol 1



3PC Termination Protocol 1

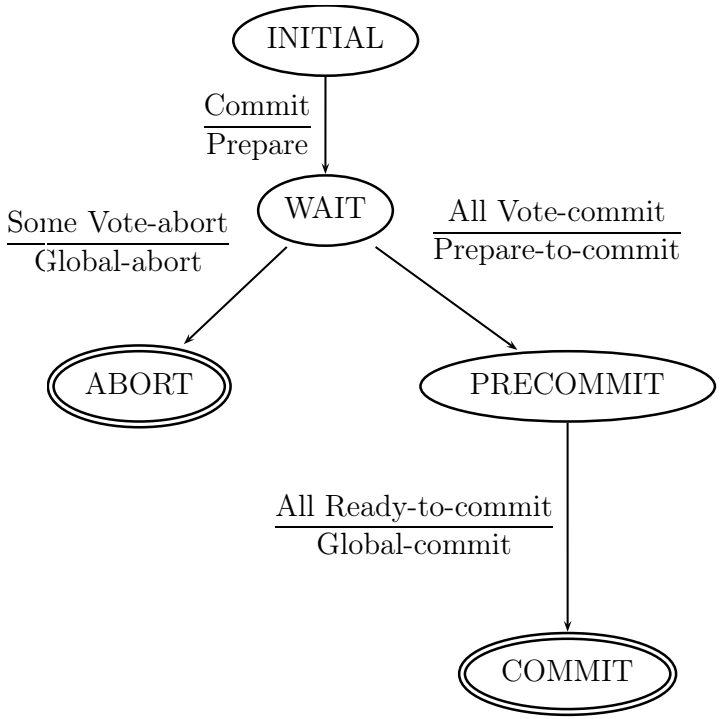
1. Participants elect a new coordinator C'
2. C' sends a State-request message to participants
3. Each participant responds to C' about its current state
4. C' terminates the transaction as follows:
   - 4.1 If there is some TM in COMMIT state, then
     - 4.1.1 C' sends “Global-commit" to all participants
   - 4.2 Else if no TM is in PRECOMMIT state, then
     - 4.2.1 C' sends “Global-abort" to all participants
   - 4.3 Otherwise,
     - 4.3.1 C' sends “Prepare-to-commit" to participants in READY state. After receiving “Ready-to-commit" from these participants, C' sends “Global-commit" to all participants

3PC Termination Protocol 1 (cont.)

- **Participant time out**: If any participant time out during the termination protocol, another new coordinator will be elected
- **Failed Participant**: Any participant that fails during the termination protocol will be ignored by the coordinator
- **Recovered Participant**: Any participant/coordinator X that fails and then recovers while the termination protocol is in progress will not be allowed to participate in the termination protocol; instead, X will be recovered following the recovery protocol

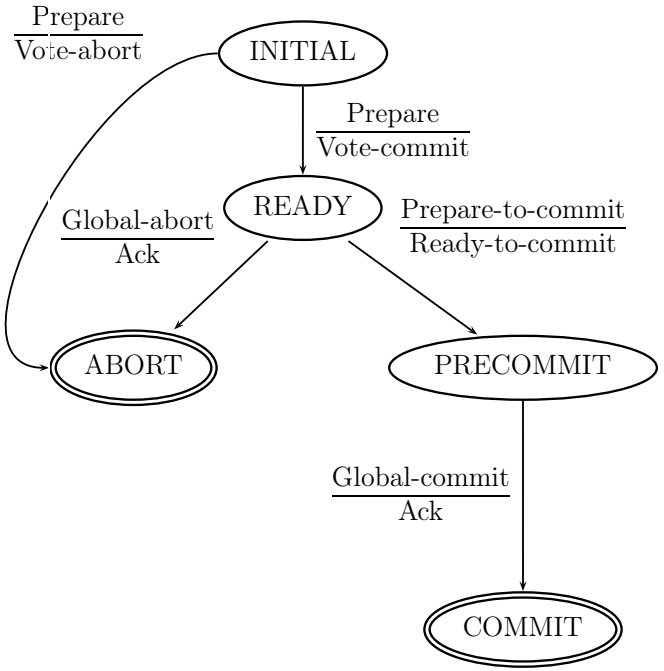
Recovery Protocol for Coordinator

Fails in state	Coordinator's recovery actions
INITIAL	Aborts transaction unilaterally
WAIT	Asks some TM for global decision
PRECOMMIT	Asks some TM for global decision
ABORT	Does nothing
COMMIT	Does nothing



Recovery Protocol for Participants

Fails in state	Participant's recovery actions
INITIAL	Aborts transaction unilaterally
READY	Asks some TM for global decision
PRECOMMIT	Asks some TM for global decision
ABORT	Does nothing
COMMIT	Does nothing



Handling Total Site Failure

- After a total site failure, the recovering TMs must remain blocked until a TM P recovers such that
  (1) either P can recover independently (i.e., P's state is INITIAL/ABORT/COMMIT),
  (2) or P was the last TM to fail
- For case (1), P simply notifies the recovered TMs of the global decision
- For case (2), P terminates the transaction by executing the termination protocol among the recovered TMs

3PC-1 Protocol

- Let's refer to the variant of 3PC protocol discussed so far as 3PC-1
- In the **absence of total site failure & communication failure**, 3PC-1 is a non-blocking protocol
- In the event of **total site failure**, blocking may occur but correctness is guaranteed by 3PC-1
- In the event of **communication failure**, 3PC-1 may not be correct!

Handling Communication Failure

- Communication failure could lead to inconsistent decisions made by multiple coordinators
- **Termination Protocol 2**: a new termination protocol that ensures correctness in the presence of communication failure

Timeout in state	Participant's termination actions
INITIAL	Aborts transaction unilaterally
READY	Executes Termination Protocol 2
PRECOMMIT	Executes Termination Protocol 2

- Key idea: A coordinator is allowed to make a decision only if it involves a majority of TMs

3PC Termination Protocol 2

- Participants elect a new coordinator *C′*
- *C′* sends a State-request message to participants
- Each participant responds to *C′* about its current state
- Case 1: If there's a COMMIT state, then *C′* decides **Commit** & sends "Global-commit" to all participants
- Case 2: If there's a ABORT state, then *C′* decides **Abort** & sends "Global-abort" to all participants

3PC Termination Protocol 2 (cont.)

- Case 3: If there's a PRECOMMIT state, no COMMIT/ABORT state, & a majority of READY/PRECOMMIT states, then
  - C' sends "Prepare-to-commit" to participants not in PRECOMMIT
  - Participant that receives "Prepare-to-commit" changes its state to PRECOMMIT & responds with "Ready-to-commit"
  - If the number of PRECOMMIT & Ready-to-commit responses forms a majority, then C' decides **Commit** & sends "Global-commit" to all participants; otherwise, C' and the participants become **blocked**

3PC Termination Protocol 2 (cont.)

- Case 4: If there's no COMMIT/ABORT state & a majority of INITIAL/READY/PREABORT states, then
  - *C′* sends "Prepare-to-abort" to participants not in PREABORT
  - Participant that receives "Prepare-to-abort" changes its state to PREABORT & responds with "Ready-to-abort"
  - If the number of PREABORT & Ready-to-abort responses forms a majority, then *C′* decides **Abort** & sends "Global-abort" to all participants; otherwise, *C′* and the participants become **blocked**
- Case 5: In all other cases, *C′* and the participants become **blocked**
- A blocked TM periodically executes the termination protocol. When a failed TM recovers, it executes the termination protocol

3PC-2 Protocol

- Let's refer to the second variant of 3PC protocol discussed as 3PC-2
- 3PC-2 is correct in the presence of total site failure and communication failure
- 3PC-2 is a non-blocking protocol in the absence of communication failure so long as a majority of TMs are operational

References

- T. Özsu & P. Valdureiz, *Distributed Transaction Processing*, Chapter 5, Principles of Distributed Database Systems, 4^{th} Edition, 2020
- P. Bernstein, V. Hadzilacos, N. Goodman, *Distributed Recovery*, Chapter 7, Concurrency Control and Recovery in Database Systems, 1987.

https://www.microsoft.com/en-us/research/wp-content/uploads/2016/05/ccontrol.zip