



CSE 512: Distributed and Parallel Data Systems

Lecture 7

Instructor: Mohamed Sarwat



Reliable Data Management

Reliability

Problem:

How to maintain

atomicity

durability

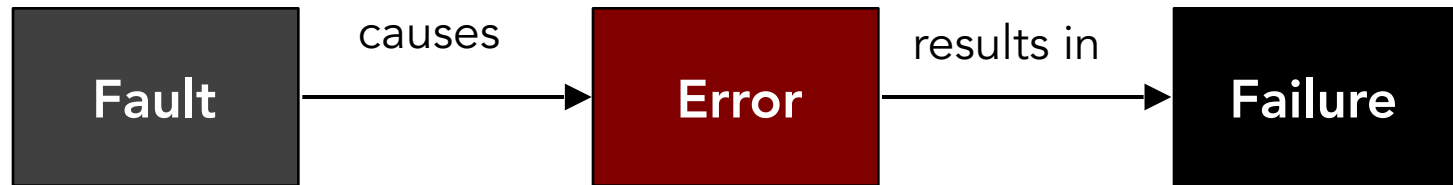
properties of transactions



Fundamental Definitions

- Reliability
 - Probability that the system has not experienced any failures within a given time period.
- Availability
 - The fraction of the time that a system meets its specification.
 - The probability that the system is operational at a given time t .

Faults to Failures



Types of Faults

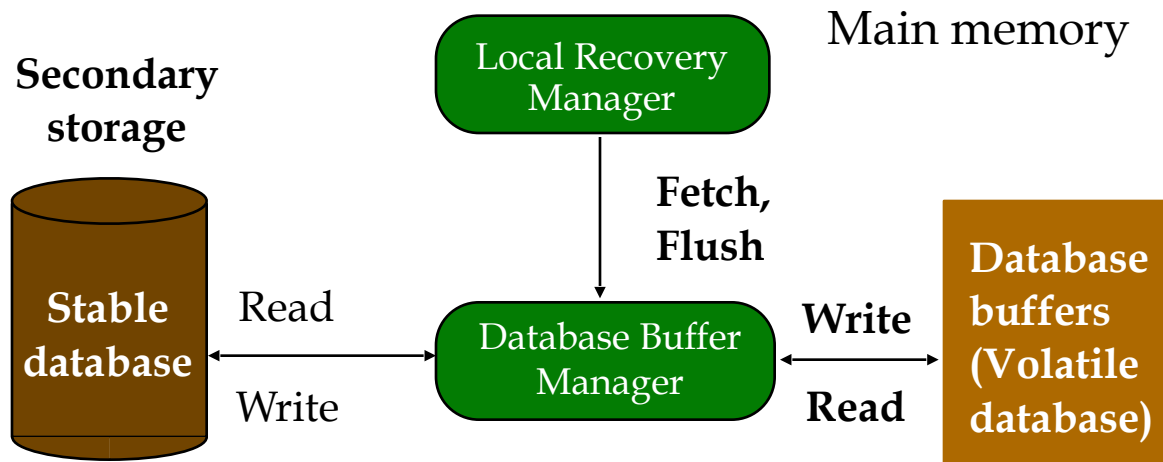
- Hard faults
 - Permanent
 - Resulting failures are called hard failures
- Soft faults
 - Transient or intermittent
 - Account for more than 90% of all failures
 - Resulting failures are called soft failures

Types of Failures

- Transaction failures
 - Transaction aborts (unilaterally or due to deadlock)
 - Avg. 3% of transactions abort abnormally
- System (site) failures
 - Failure of processor, main memory, power supply, ...
 - Main memory contents are lost, but secondary storage contents are safe
 - Partial vs. total failure
- Media failures
 - Failure of secondary storage devices such that the stored data is lost
 - Head crash/controller failure (?)
- Communication failures
 - Lost/undeliverable messages
 - Network partitioning

Local Recovery Management – Architecture

- Volatile storage
 - Consists of the main memory of the computer system (RAM).
- Stable/Persistent storage
 - Resilient to failures and loses its contents only in the presence of media failures (e.g., head crashes on disks).
 - Implemented via a combination of hardware (non-volatile storage) and software (stable-write, stable-read, clean-up) components.



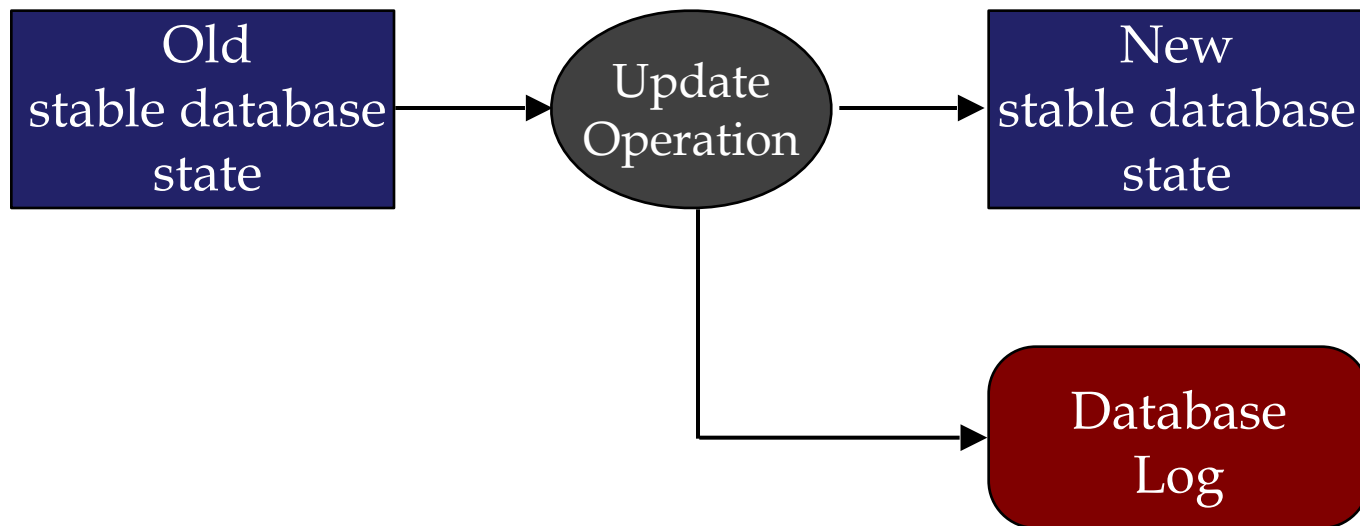
Update Strategies

- In-place update
 - Each update causes a change in one or more data values on pages in the database buffers
- Out-of-place update
 - Each update causes the new value(s) of data item(s) to be stored separate from the old value(s)

In-Place Update Recovery Information

Database Log

Every action of a transaction must not only perform the action, but must also write a *log* record to an append-only file.



Logging

The log contains information used by the recovery process to restore the consistency of a system. This information may include

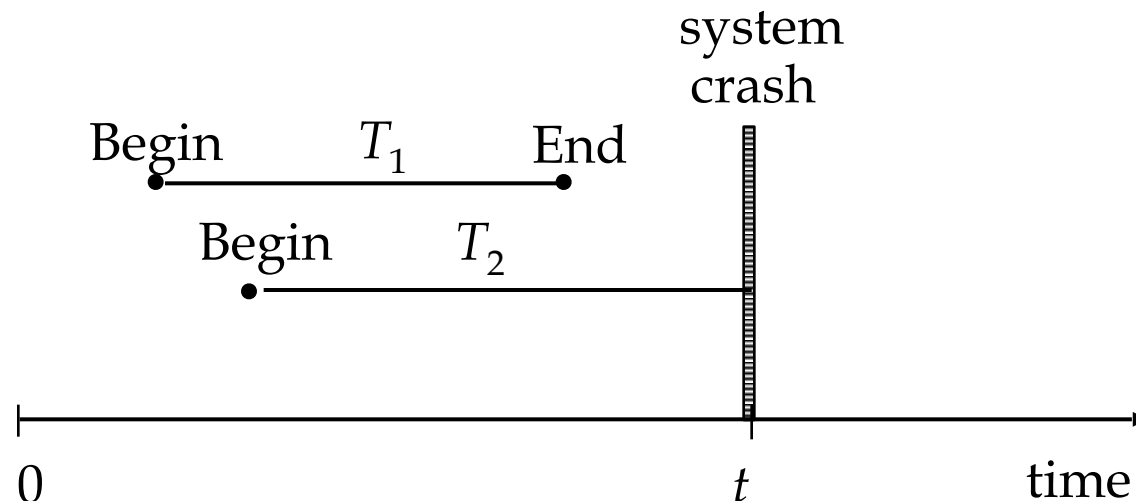
- transaction identifier
- type of operation (action)
- items accessed by the transaction to perform the action
- old value (state) of item (**before image**)
- new value (state) of item (**after image**)

...

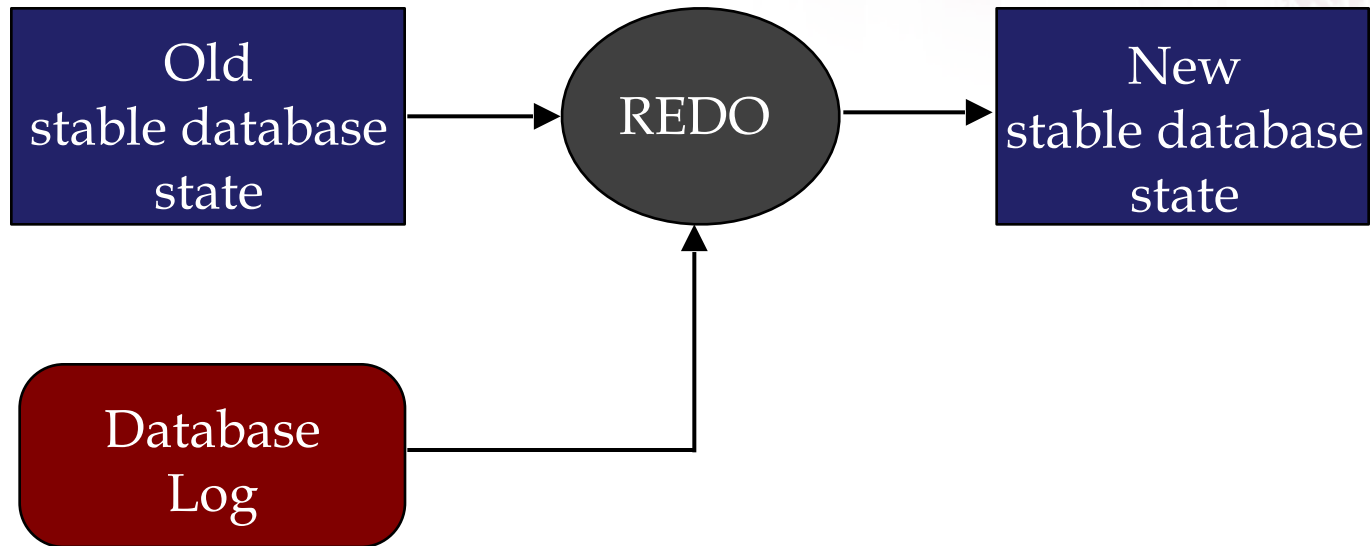
Why Logging?

Upon recovery:

- all of T_1 's effects should be reflected in the database (REDO if necessary due to a failure)
- none of T_2 's effects should be reflected in the database (UNDO if necessary)

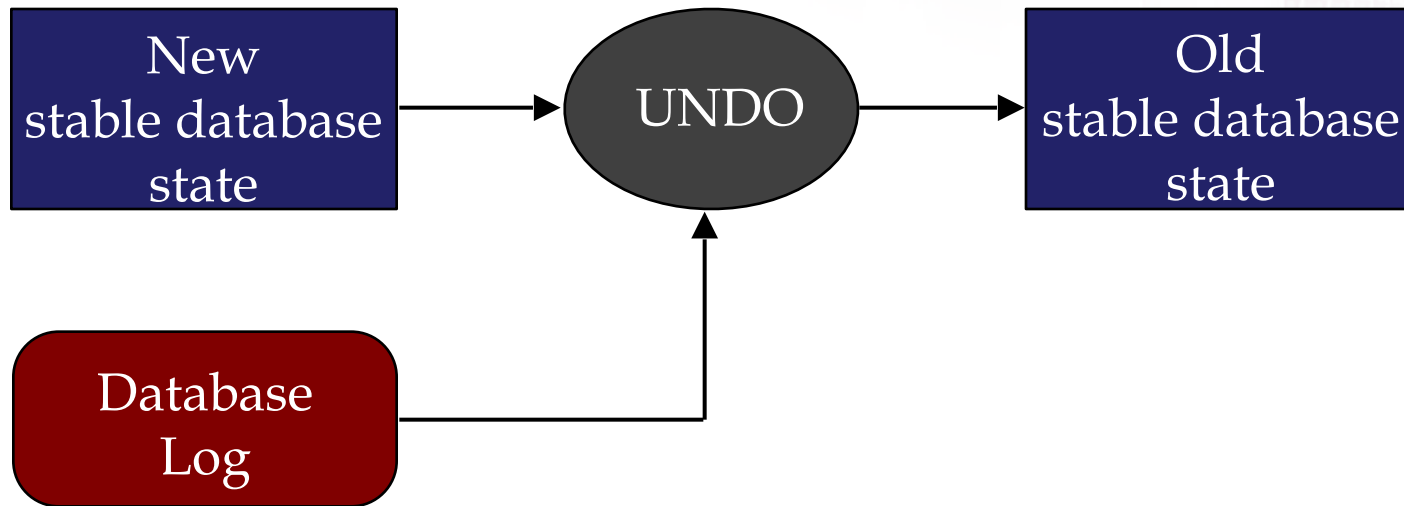


REDO Protocol



- REDO'ing an action means performing it again.
- The REDO operation uses the log information and performs the action that might have been done before, or not done due to failures.
- The REDO operation generates the new image.

UNDO Protocol



- UNDO'ing an action means to restore the object to its before image.
- The UNDO operation uses the log information and restores the old value of the object.

Write–Ahead Log Protocol

- Notice:
 - If a system crashes before a transaction is committed, then all the operations must be undone. Only need the before images (*undo portion* of the log).
 - Once a transaction is committed, some of its actions might have to be redone. Need the after images (*redo portion* of the log).
- WAL protocol :
 - ❶ Before a stable database is updated, the undo portion of the log should be written to the stable log
 - ❷ When a transaction commits, the redo portion of the log must be written to stable log prior to the updating of the stable database.

Out-of-Place Update Recovery Information

- Shadowing
 - When an update occurs, don't change the old page, but create a shadow page with the new values and write it into the stable database.
 - Update the access paths so that subsequent accesses are to the new shadow page.
 - The old page retained for recovery.
- Differential files
 - For each file F maintain
 - a read only part FR
 - a differential file consisting of insertions part DF^+ and deletions part DF^-
 - Thus, $F = (FR \cup DF^+) - DF^-$
 - Updates treated as delete old value, insert new value



Distributed Reliability Protocols

- Commit protocols
 - How to execute commit command for distributed transactions.
 - Issue: how to ensure atomicity and durability?
- Termination protocols
 - If a failure occurs, how can the remaining operational sites deal with it.
 - *Non-blocking* : the occurrence of failures should not force the sites to wait until the failure is repaired to terminate the transaction.
- Recovery protocols
 - When a failure occurs, how do the sites where the failure occurred deal with it.
 - *Independent* : a failed site can determine the outcome of a transaction without having to obtain remote information.
- Independent recovery \Rightarrow non-blocking termination

Two-Phase Commit (2PC)

Phase 1 : The coordinator gets the participants ready to write the results into the database

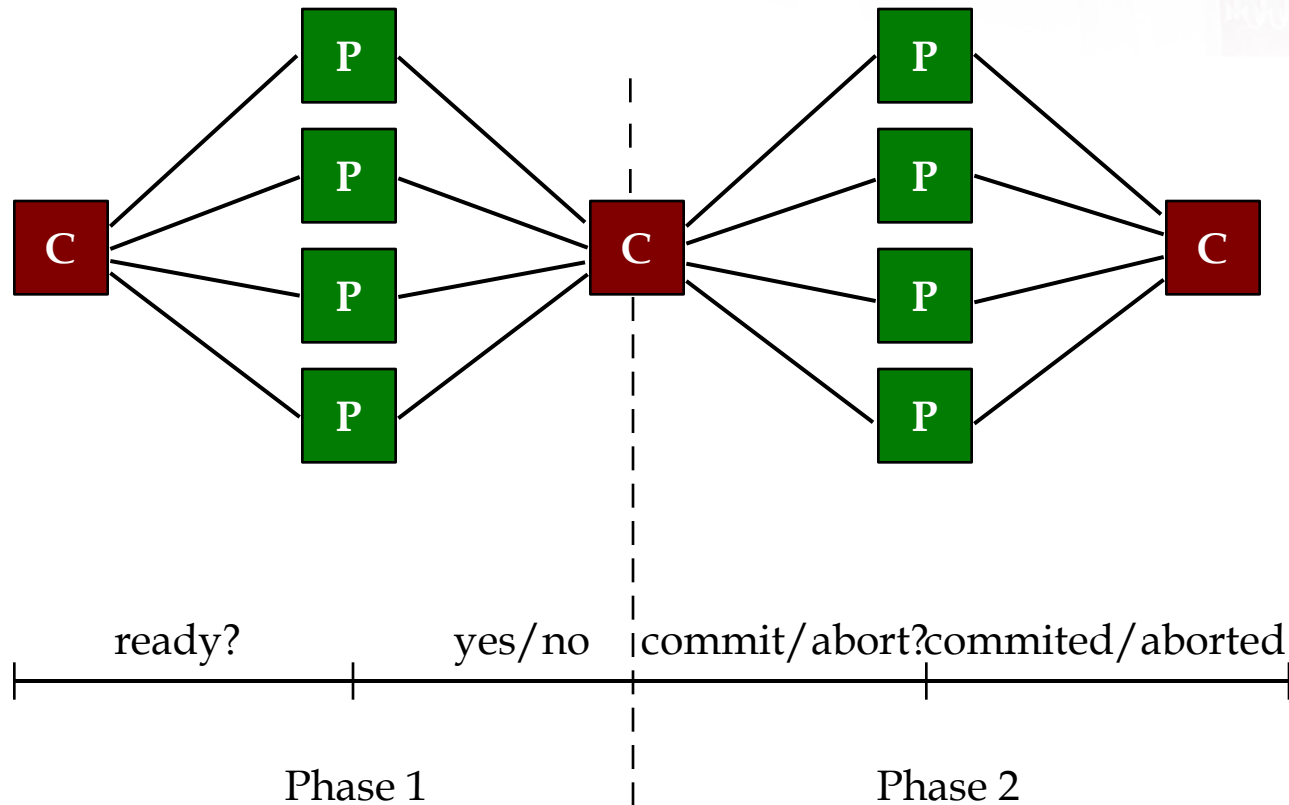
Phase 2 : Everybody writes the results into the database

- **Coordinator** :The process at the site where the transaction originates and which controls the execution
- **Participant** :The process at the other sites that participate in executing the transaction

Global Commit Rule:

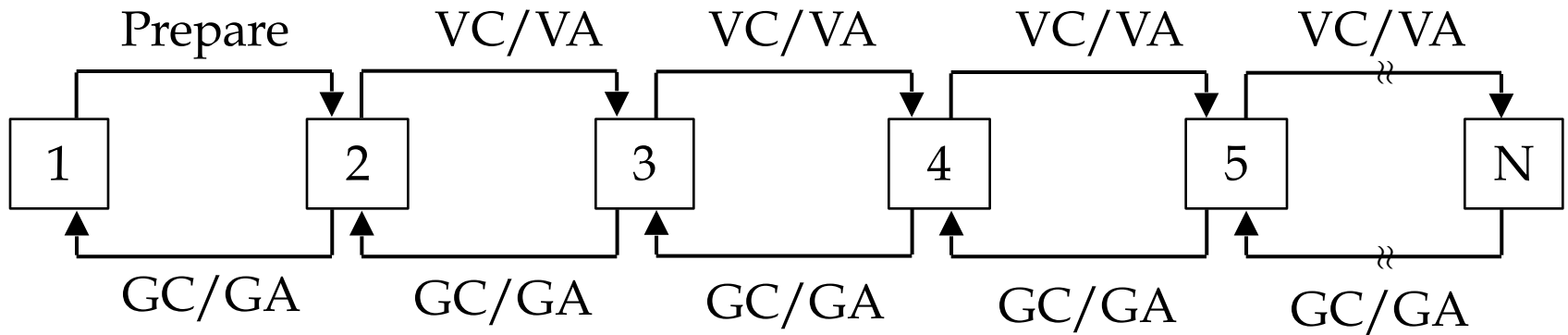
- ❶ The coordinator aborts a transaction if and only if at least one participant votes to abort it.
- ❷ The coordinator commits a transaction if and only if all of the participants vote to commit it.

Centralized 2PC



Linear 2PC

Phase 1



Phase 2

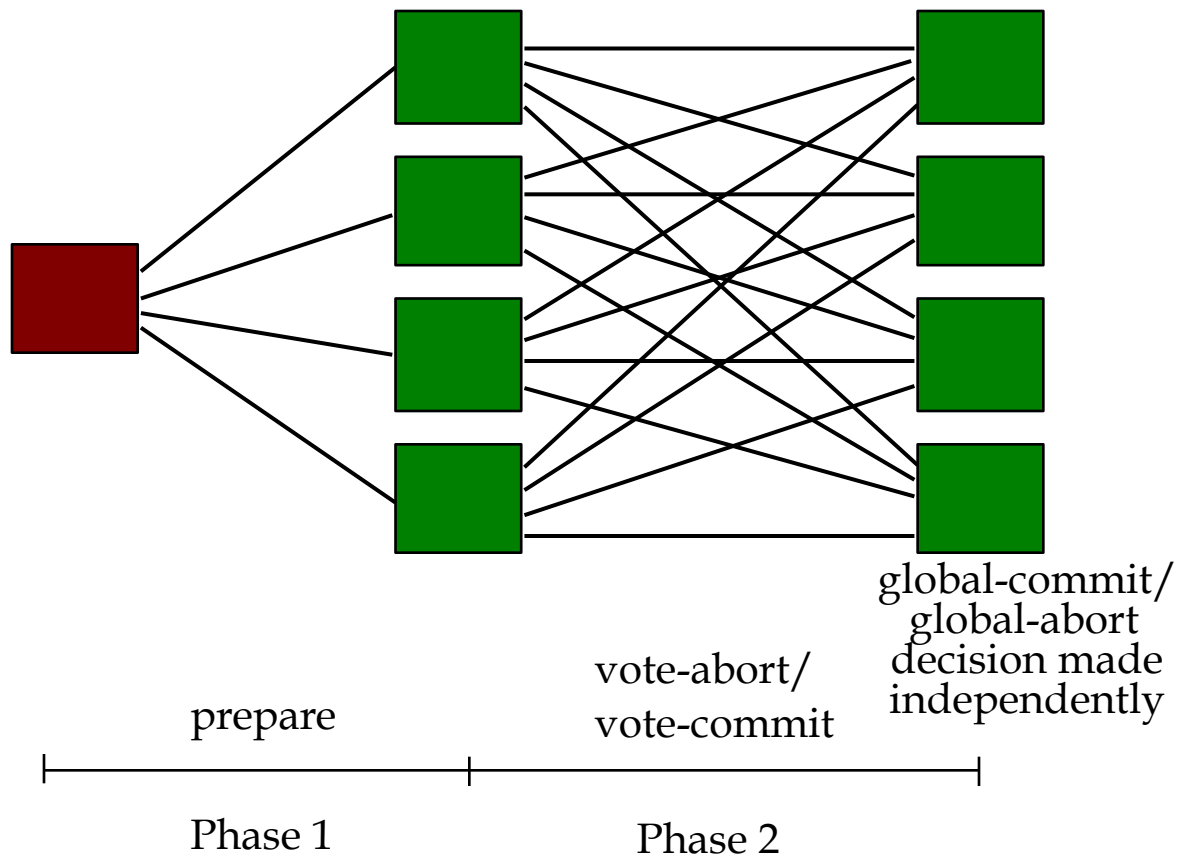
VC: Vote-Commit, VA: Vote-Abort, GC: Global-commit, GA: Global-abort

Distributed 2PC

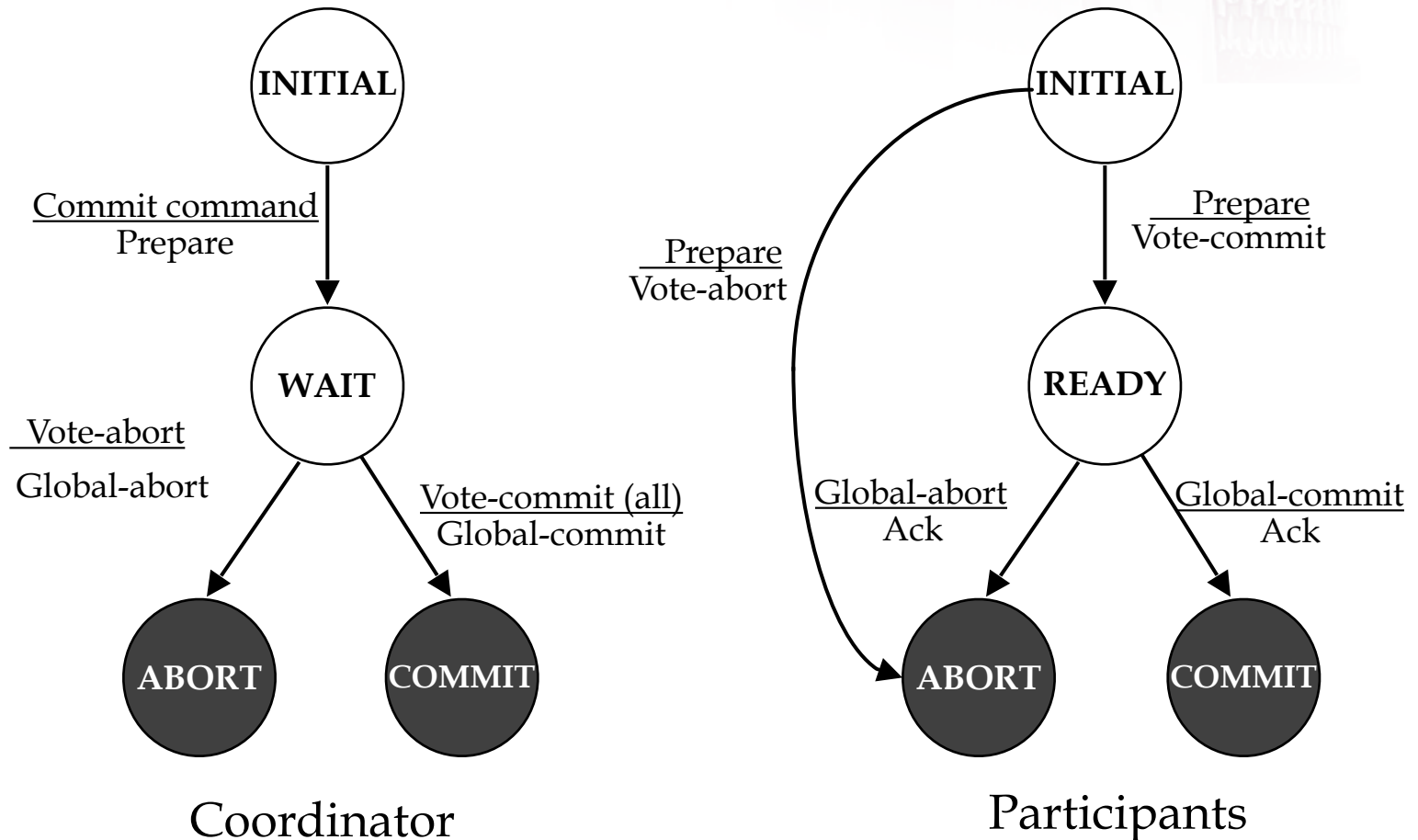
Coordinator

Participants

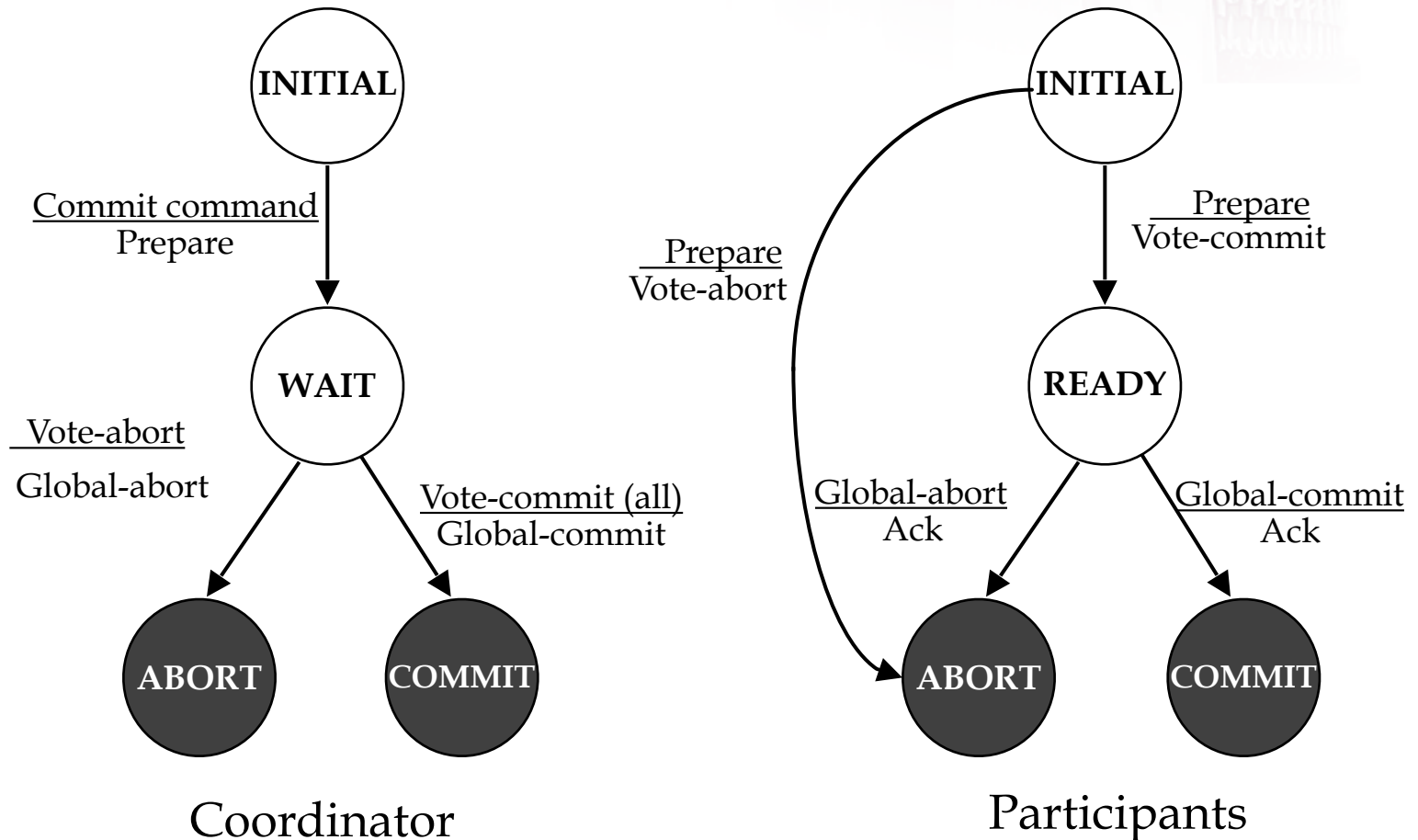
Participants



State Transitions in 2PC

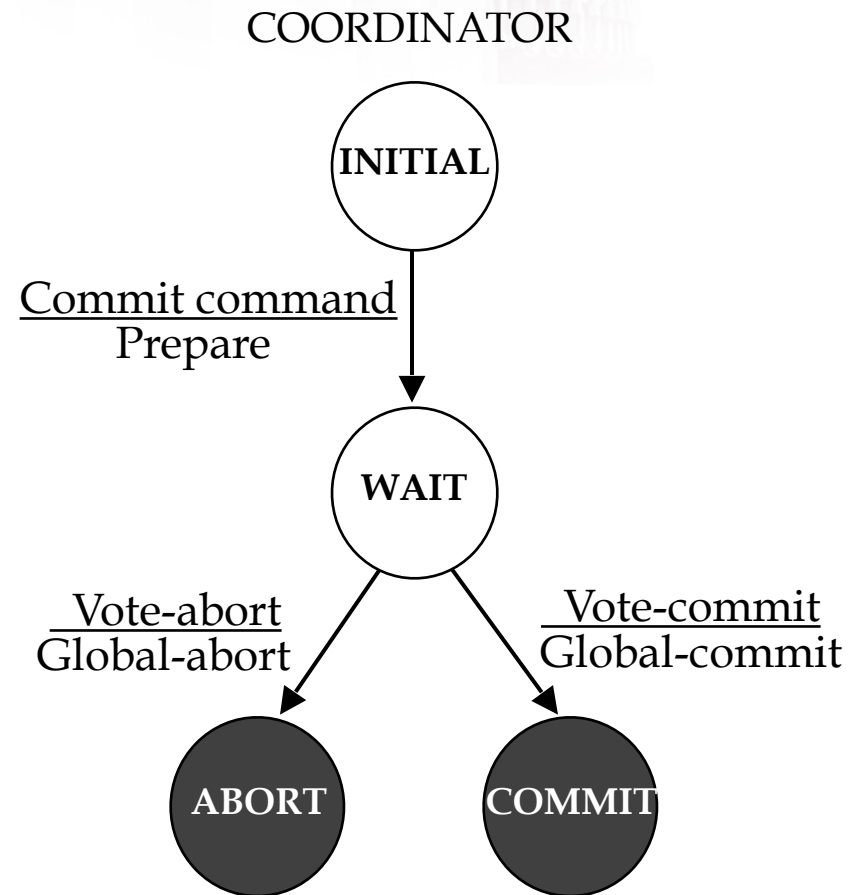


Handling Failures



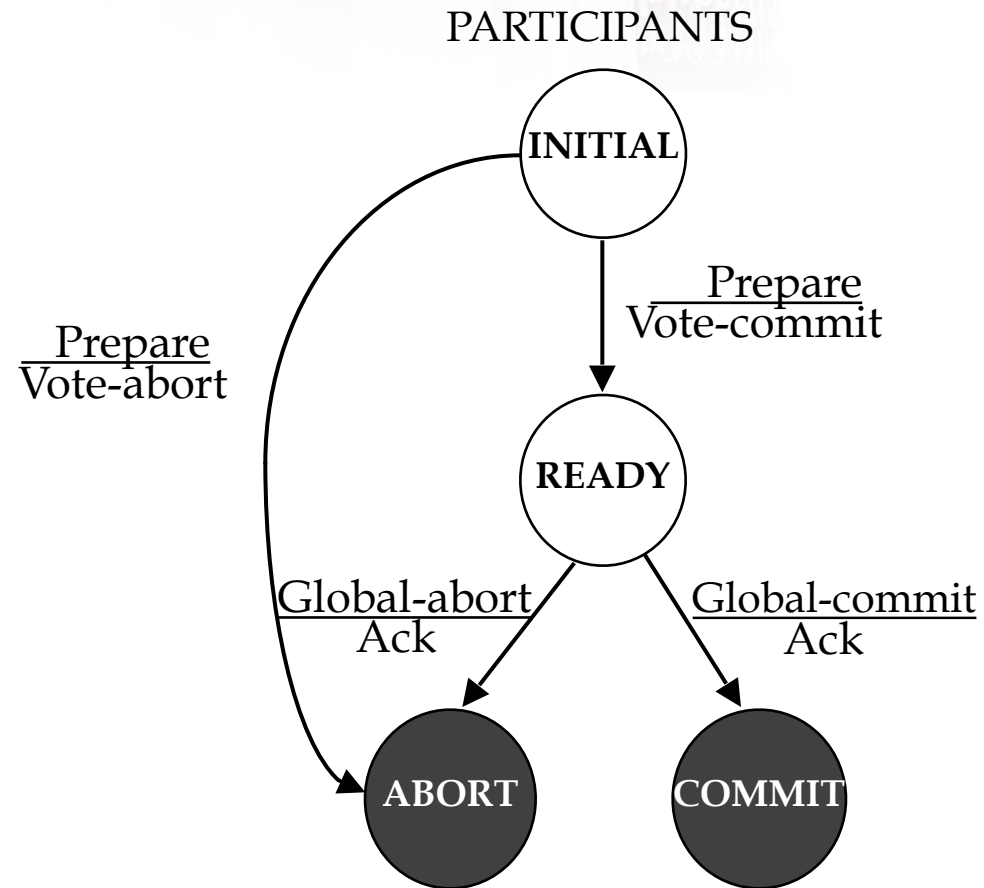
Site Failures - 2PC Termination

- Timeout in INITIAL
 - Who cares
- Timeout in WAIT
 - Cannot unilaterally commit
 - Can unilaterally abort
- Timeout in ABORT or COMMIT
 - Stay blocked and wait for the Acks



Site Failures - 2PC Termination

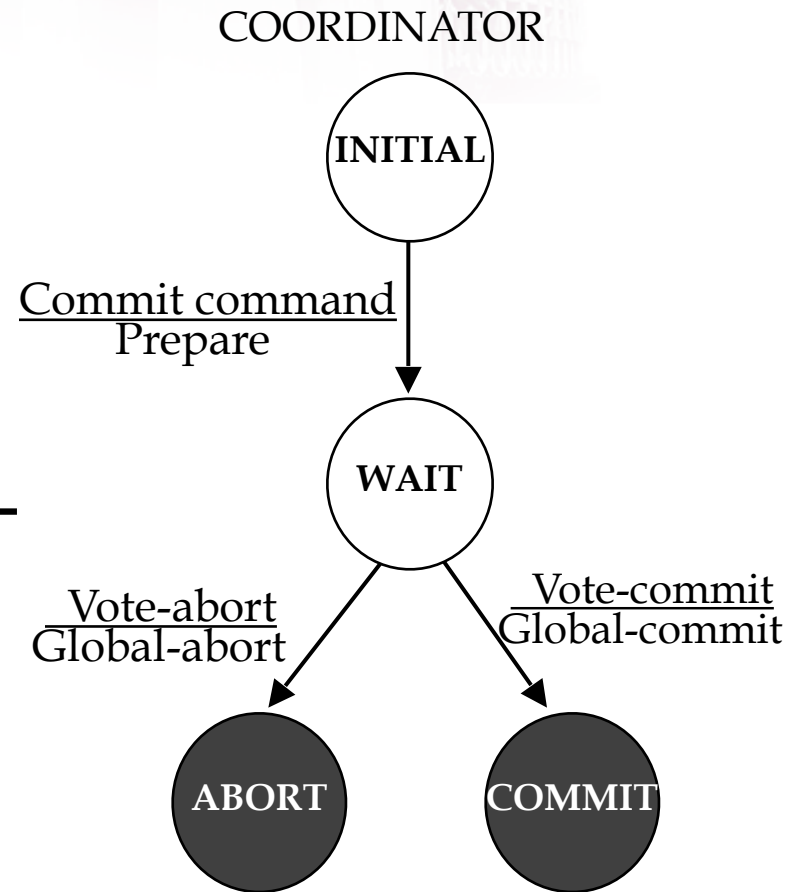
- Timeout in INITIAL
 - Coordinator must have failed in INITIAL state
 - Unilaterally abort
- Timeout in READY
 - Stay blocked



Recover From Failures

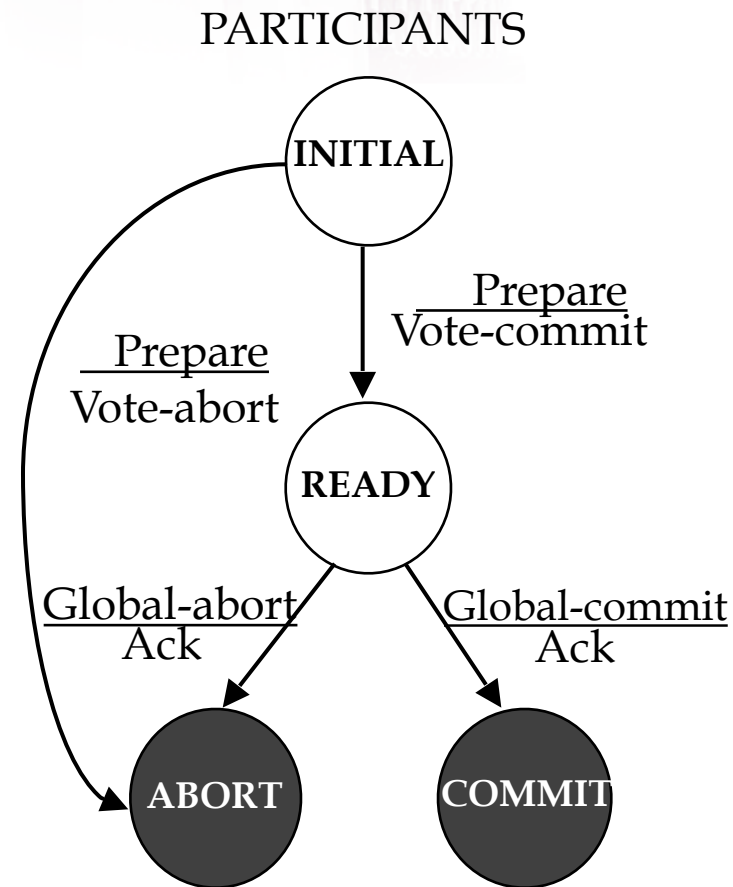
Site Failures - 2PC Recovery

- Failure in INITIAL
 - Start the commit process upon recovery
- Failure in WAIT
 - Restart the commit process upon recovery
- Failure in ABORT or COMMIT
 - Nothing special if all the acks have been received
 - Otherwise the termination protocol is involved



Site Failures - 2PC Recovery

- Failure in INITIAL
 - Unilaterally abort upon recovery
- Failure in READY
 - The coordinator has been informed about the local decision
 - Treat as timeout in READY state and invoke the termination protocol
- Failure in ABORT or COMMIT
 - Nothing special needs to be done



2PC Recovery Protocols – Additional Cases

Arise due to non-atomicity of log and message send actions

- Coordinator site fails after writing “begin_commit” log and before sending “prepare” command
 - treat it as a failure in WAIT state; send “prepare” command
- Participant site fails after writing “ready” record in log but before “vote-commit” is sent
 - treat it as failure in READY state
 - alternatively, can send “vote-commit” upon recovery
- Participant site fails after writing “abort” record in log but before “vote-abort” is sent
 - no need to do anything upon recovery

2PC Recovery Protocols – Additional Case

- Coordinator site fails after logging its final decision record but before sending its decision to the participants
 - coordinator treats it as a failure in COMMIT or ABORT state
 - participants treat it as timeout in the READY state
- Participant site fails after writing “abort” or “commit” record in log but before acknowledgement is sent
 - participant treats it as failure in COMMIT or ABORT state
 - coordinator will handle it by timeout in COMMIT or ABORT state

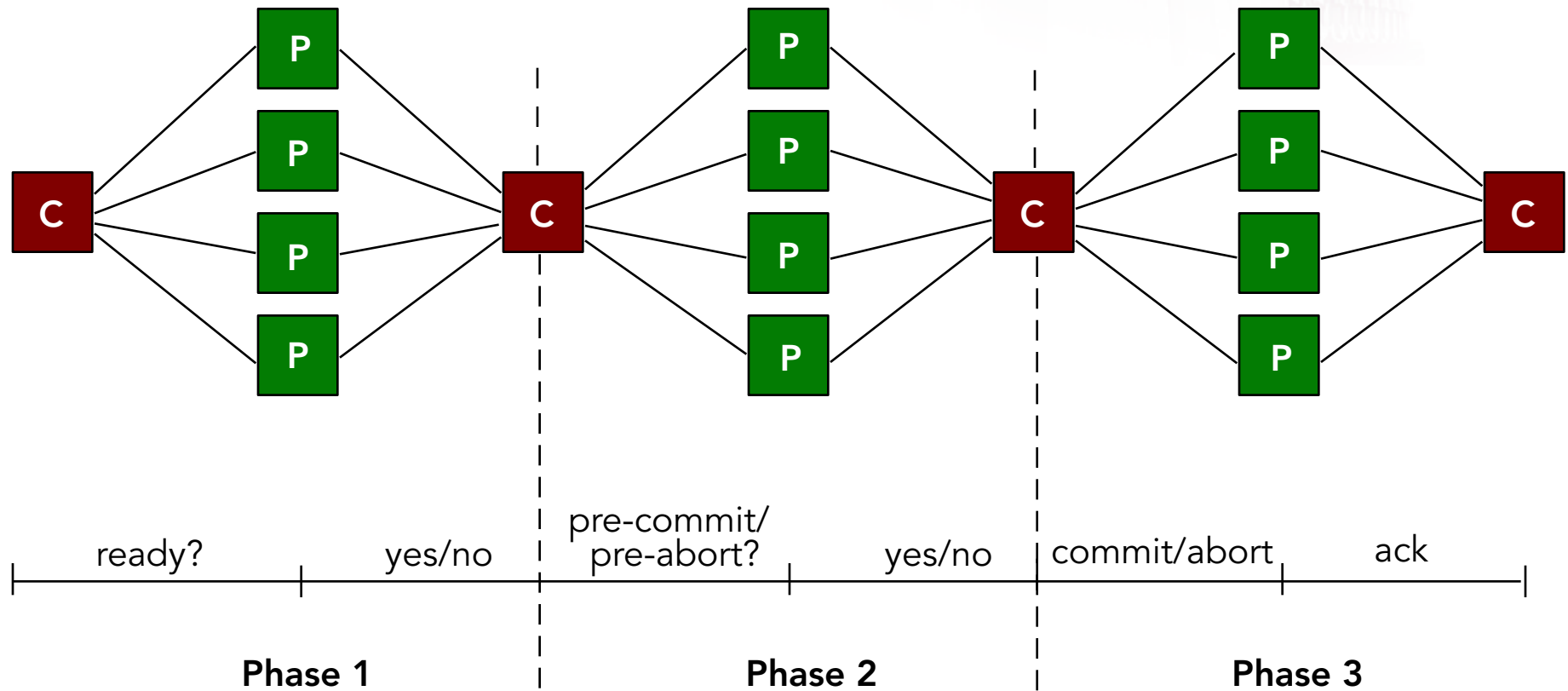
Problem With 2PC

- Blocking
 - Ready implies that the participant waits for the coordinator
 - If coordinator fails, site is blocked until recovery
 - Blocking reduces availability
- Independent recovery is not possible
- However, it is known that:
 - Independent recovery protocols exist only for single site failures; no independent recovery protocol exists which is resilient to multiple-site failures.
- So we search for these protocols – 3PC

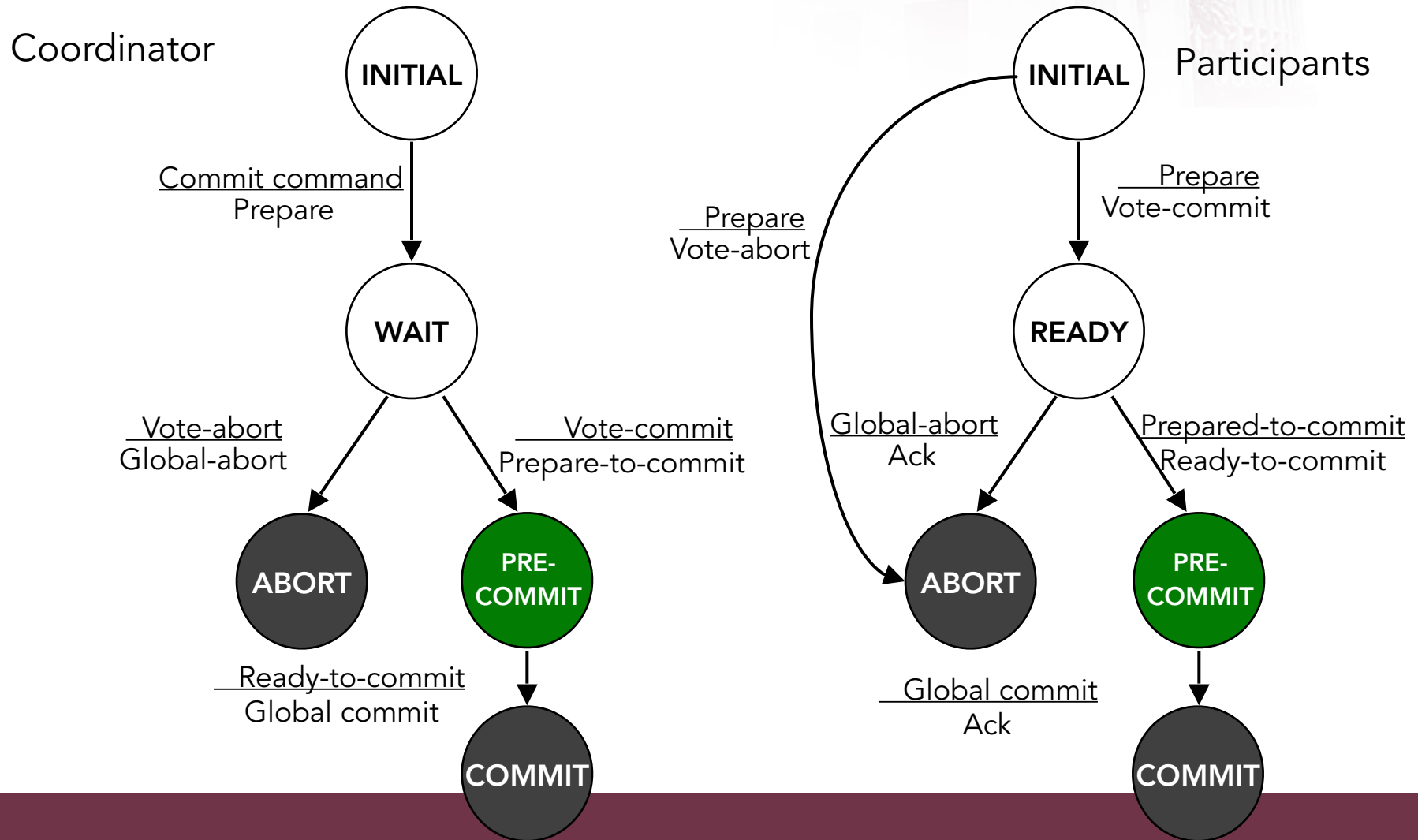
Three-Phase Commit

- 3PC is non-blocking.
- A commit protocols is non-blocking iff
 - it is synchronous within one state transition, and
 - its state transition diagram contains
 - no state which is “adjacent” to both a commit and an abort state, and
 - no non-committable state which is “adjacent” to a commit state
- Adjacent: possible to go from one stat to another with a single state transition
- Committable: all sites have voted to commit a transaction
 - e.g.: COMMIT state

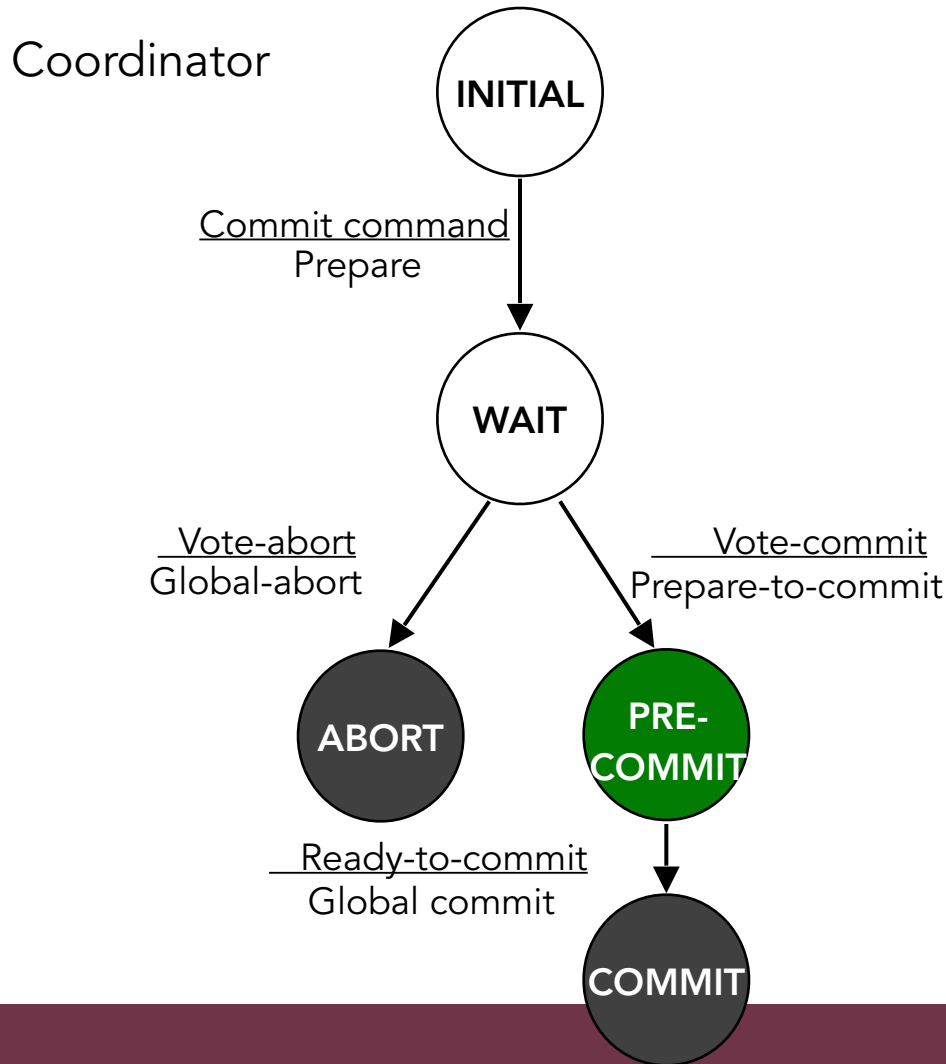
Communication Structure



State Transitions in 3PC



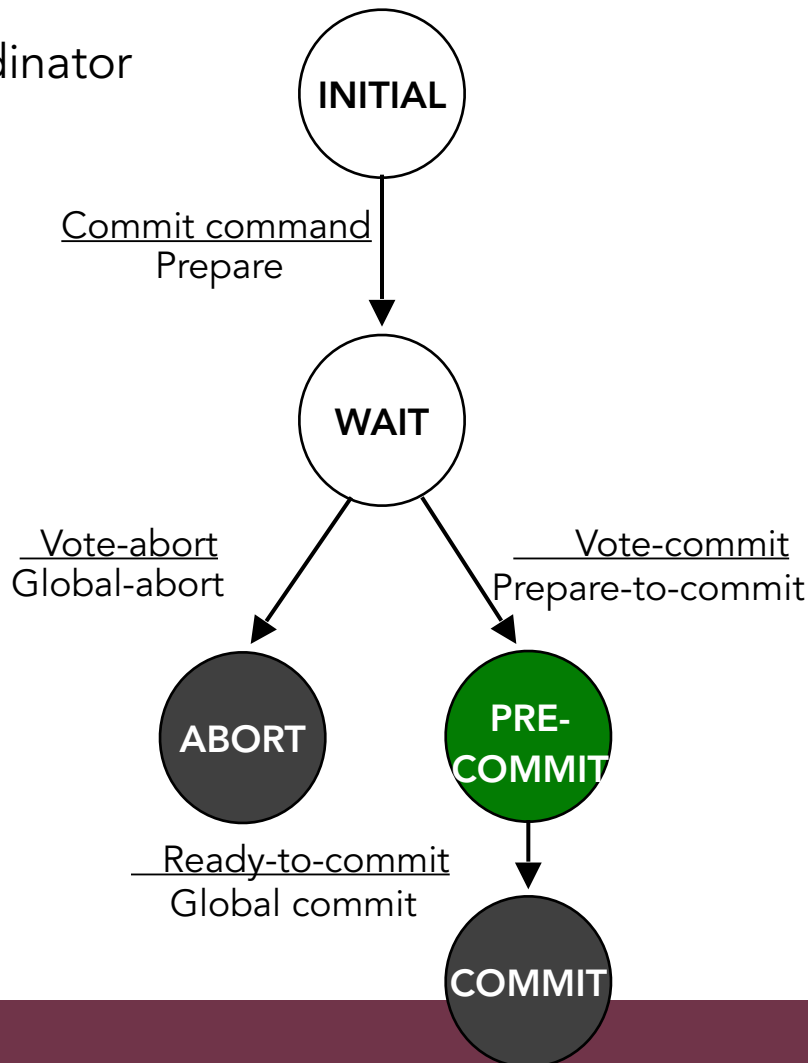
Site Failures – 3PC Termination



- Timeout in INITIAL
 - Who cares
- Timeout in WAIT
 - Unilaterally abort
- Timeout in PRECOMMIT
 - Participants may not be in PRE-COMMIT, but at least in READY
 - Move all the participants to PRECOMMIT state
 - Terminate by globally committing

Site Failures – 3PC Termination

Coordinator



- Timeout in **ABORT** or **COMMIT**
 - Just ignore and treat the transaction as completed
 - participants are either in **PRECOMMIT** or **READY** state and can follow their termination protocols

Site Failures – 3PC Termination



- Timeout in INITIAL
 - Coordinator must have failed in INITIAL state
 - Unilaterally abort
- Timeout in READY
 - Voted to commit, but does not know the coordinator's decision
 - Elect a new coordinator and terminate using a special protocol
- Timeout in PRECOMMIT
 - Handle it the same as timeout in READY state



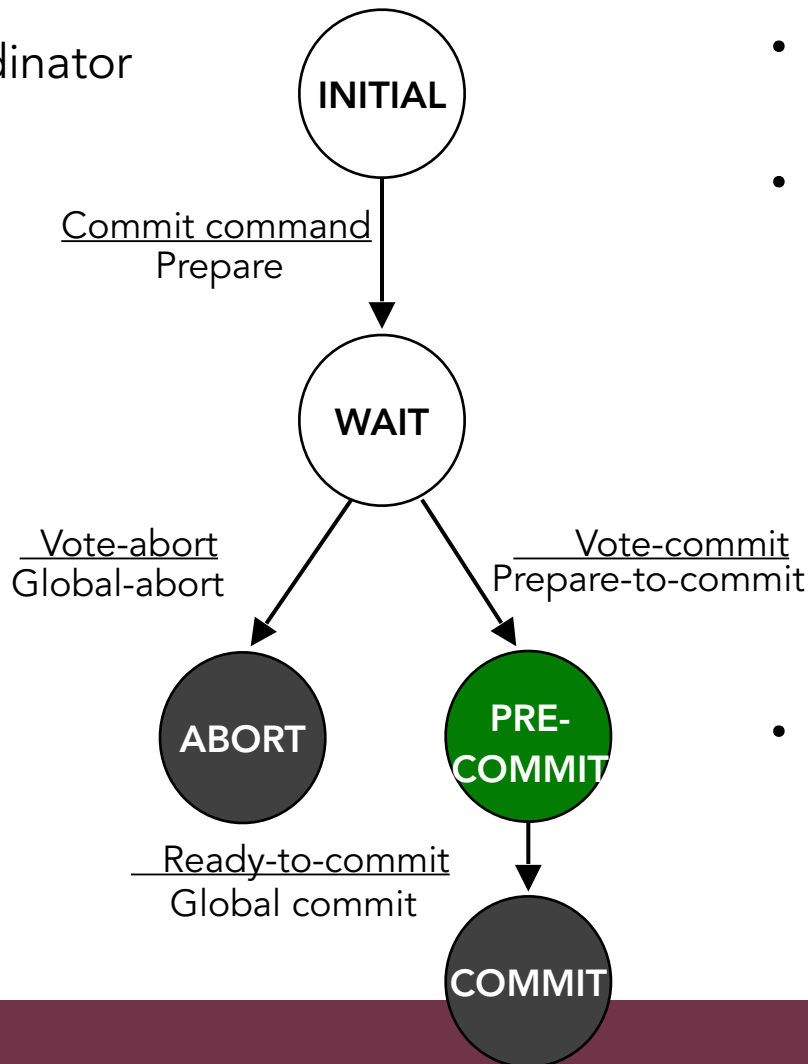
Termination Protocol Upon Coordinator Election

New coordinator can be in one of four states: WAIT, PRECOMMIT, COMMIT, ABORT

- ❶ Coordinator sends its state to all of the participants asking them to assume its state.
- ❷ Participants “back-up” and reply with appropriate messages, except those in ABORT and COMMIT states. Those in these states respond with “Ack” but stay in their states.
- ❸ Coordinator guides the participants towards termination:
 - If the new coordinator is in the WAIT state, participants can be in INITIAL, READY, ABORT or PRECOMMIT states. New coordinator globally aborts the transaction.
 - If the new coordinator is in the PRECOMMIT state, the participants can be in READY, PRECOMMIT or COMMIT states. The new coordinator will globally commit the transaction.
 - If the new coordinator is in the ABORT or COMMIT states, at the end of the first phase, the participants will have moved to that state as well.

Site Failures – 3PC Recovery

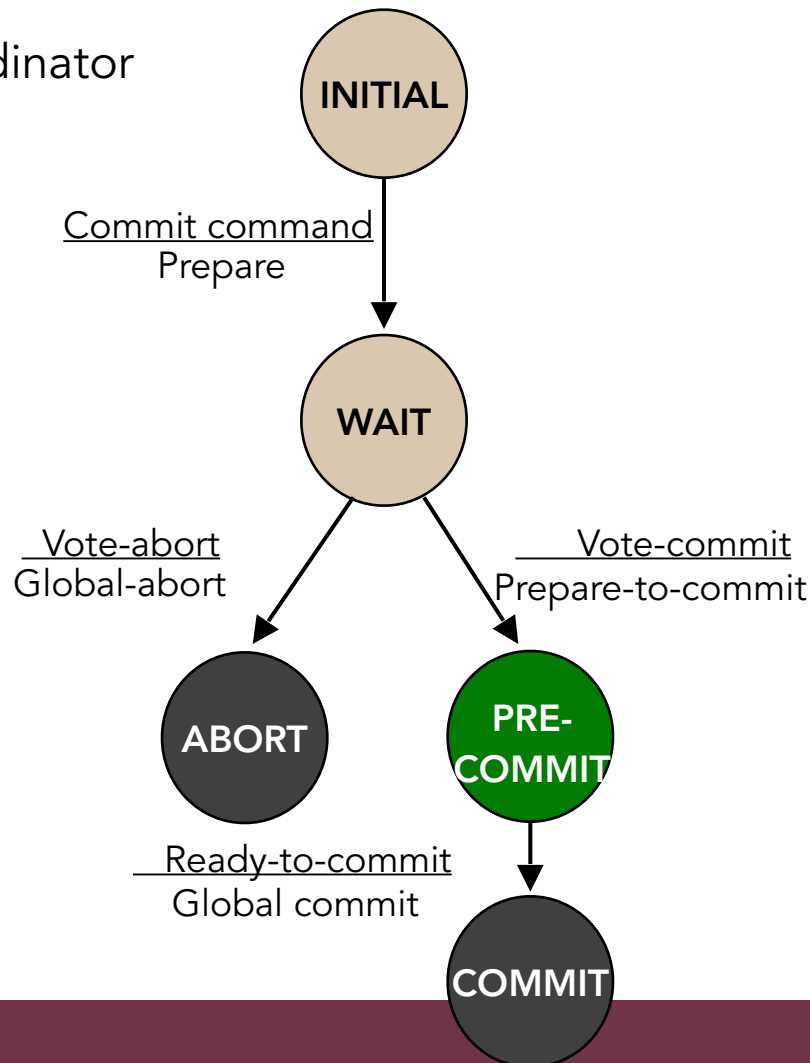
Coordinator



- Failure in INITIAL
 - start commit process upon recovery
- Failure in WAIT
 - the participants may have elected a new coordinator and terminated the transaction
 - the new coordinator could be in WAIT or ABORT states ⇒ transaction aborted
 - ask around for the fate of the transaction
- Failure in PRECOMMIT
 - ask around for the fate of the transaction

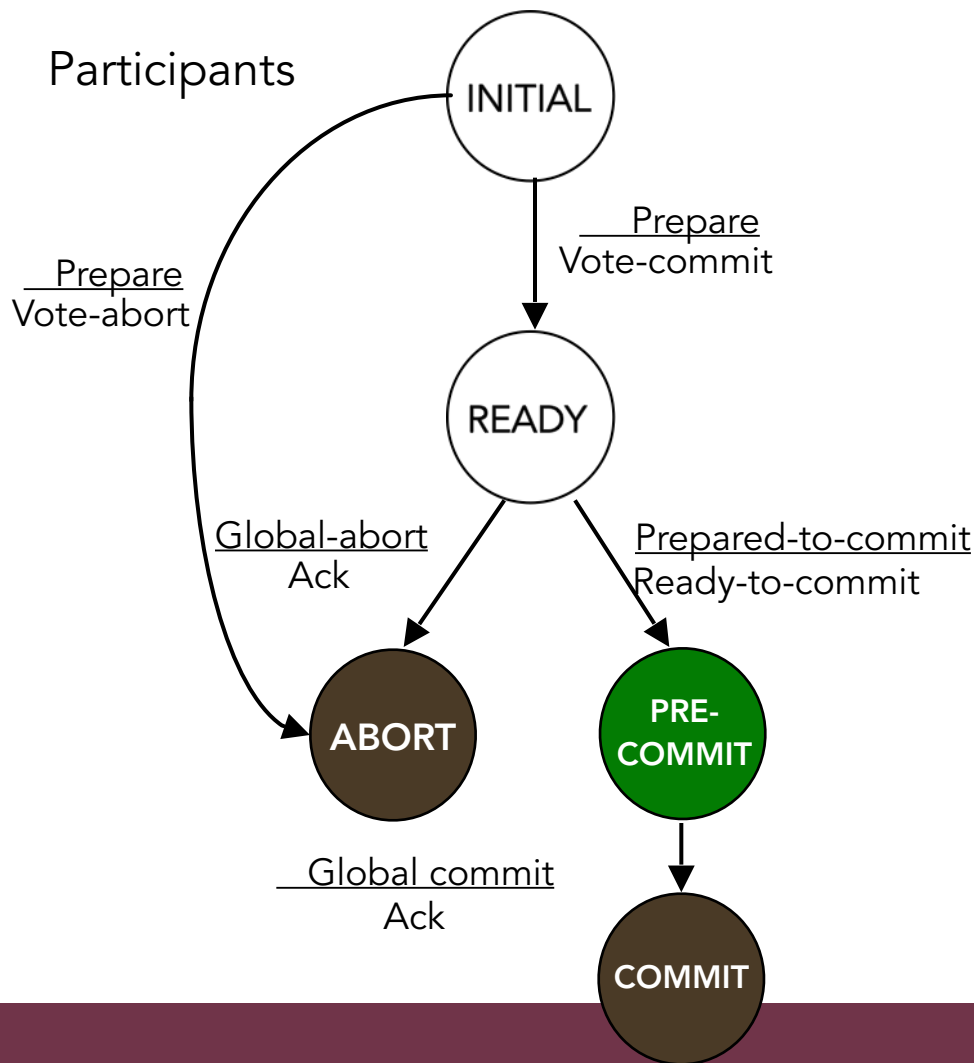
Site Failures – 3PC Recovery

Coordinator



- Failure in COMMIT or ABORT
 - Nothing special if all the acknowledgements have been received; otherwise the termination protocol is involved

Site Failures – 3PC Recovery



- Failure in INITIAL
 - unilaterally abort upon recovery
- Failure in READY
 - the coordinator has been informed about the local decision
 - upon recovery, ask around
- Failure in PRECOMMIT
 - ask around to determine how the other participants have terminated the transaction
- Failure in COMMIT or ABORT
 - no need to do anything

Network Partitioning

- Simple partitioning
 - Only two partitions
- Multiple partitioning
 - More than two partitions
- Formal bounds:
 - There exists no non-blocking protocol that is resilient to a network partition if messages are lost when partition occurs.
 - There exist non-blocking protocols which are resilient to a single network partition if all undeliverable messages are returned to sender.
 - There exists no non-blocking protocol which is resilient to a multiple partition.

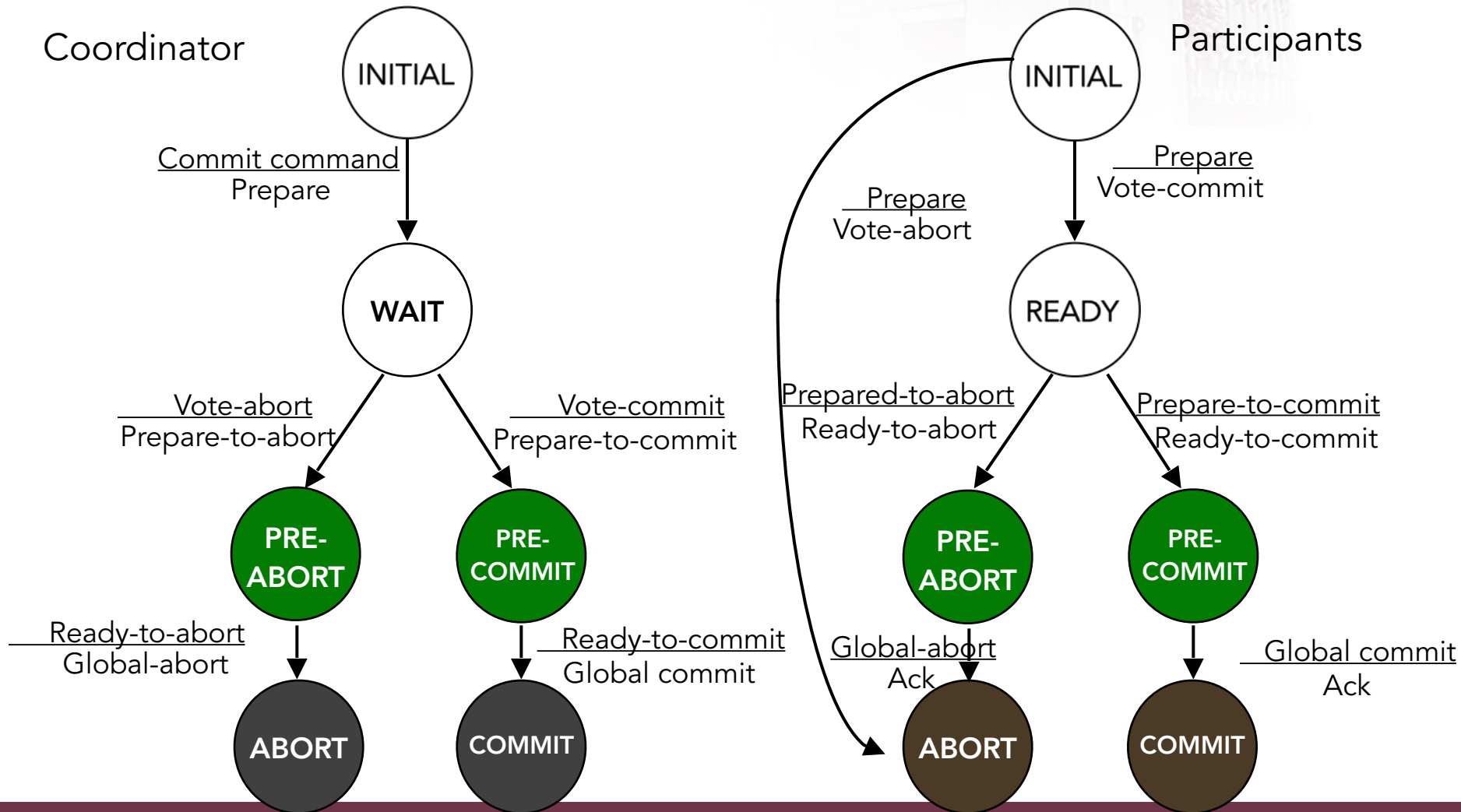
Independent Recovery Protocols for Network Partitioning

- No general solution possible
 - allow one group to terminate while the other is blocked
 - improve availability
- How to determine which group to proceed?
 - The group with a majority
- How does a group know if it has majority?
 - Centralized
 - Whichever partitions contains the central site should terminate the transaction
 - Voting-based (quorum)

Quorum Protocols

- The network partitioning problem is handled by the commit protocol.
- Every site is assigned a vote V_i .
- Total number of votes in the system V
- Abort quorums V_a , commit quorums V_c
 - $V_a + V_c > V$ where $0 \leq V_a, V_c \leq V$
 - Before a transaction commits, it must obtain a commit quorum V_c
 - Before a transaction aborts, it must obtain an abort quorum V_a

State Transitions in Quorum Protocols





Use for Network Partitioning

- Before commit (i.e., moving from PRECOMMIT to COMMIT), coordinator receives commit quorum from participants. One partition may have the commit quorum.
- Assumes that failures are “clean” which means:
 - failures that change the network's topology are detected by all sites instantaneously
 - each site has a view of the network consisting of all the sites it can communicate with



Questions