Outline

- Introduction
- Background
- Distributed DBMS Architecture
- Distributed Database Design
- Distributed Query Processing
- Transaction Management
  - Commit/Termination protocols – 2PC
- Building Distributed Database Systems (RAID)
- Mobile Database Systems
- Privacy, Trust, and Authentication
- Peer to Peer Systems

Useful References

- Textbook *Principles of Distributed Database Systems*,

  Chapter 12.4, 12.5.1
- D. Skeen and M Stonebraker, *A Formal Model of Crash Recovery in a Distributed System*, IEEE Trans. Software Eng. 9(3): 219-228, 1983.
- D. Skeen, *A Decentralized Termination Protocol*, IEEE Symposium on Reliability in Distributed Software and Database Systems, July 1981.

Byzantine General Problem

- Two generals are situated on adjacent hills and enemy is in the valley in between.
- Enemy can defeat either general, but not both.
- To succeed, both generals must agree to either attack or retreat.
- The generals can communicate via messengers who are subject to capture or getting lost.
- The general may themselves be traitors or send inconsistent information.

Byzantine Agreement

- Problem of a set of processors to agree on a common value for an object. Processors may fail arbitrarily, die and revive randomly, send messages when they are not supposed to etc.

Atomicity Control from Book

- 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

General Terminology for Commit/Termination/Recovery Protocols

Committed:	Effects are installed to the database.
Aborted:	Does not execute to completion and any partial effects on database are erased.
Consistent state:	Derived state from serial execution.

Inconsistency caused by:

1. Concurrently executing transaction.
2. Failures causing partial or incorrect execution of a transaction.

General Terminology for Commit/Termination/Recovery Protocols

- Commit protocols
  - Protocols for directing the successful execution of a simple transaction
- Termination protocols
  - Protocols at operational site to commit/abort an unfinished transaction after a failure
- Recovery protocols
  - Protocols at failed site to complete all transactions outstanding at the time of failure

General Terminology for Commit/Termination/Recovery Protocols

- Distributed Crash Recovery:
  - Centralized Protocols
  - Hierarchical Protocols
  - Linear Protocols
  - Decentralized Protocols
- Phase:
  - Consists of a message round where all Sites exchange messages.
- Two Phase Commit Protocol:
  - ARGUS, LOCUS, INGRES
- Four Phase Commit Protocol:
  - SSD-1
- Quorum:
  - Minimum number of sites needed to proceed with an action

Commit/Termination Protocols

- Two Phase Commit
- Three Phase Commit
- Four Phase Commit
- Linear, Centralized, Hierarchical, Decentralized Protocols

Two Phase Commit

	Site 1	Site 2
1.	Trans. arrives. Message to ask for vote is sent to other site(s)	
		Message is recorded. Site votes Y or N (abort) Vote is sent to site 1
2.	The vote is received. If vote = Y on both sites, then Commit else Abort	
		Either Commit or Abort based on the decision of site 1

Two-Phase Commit (2PC)

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

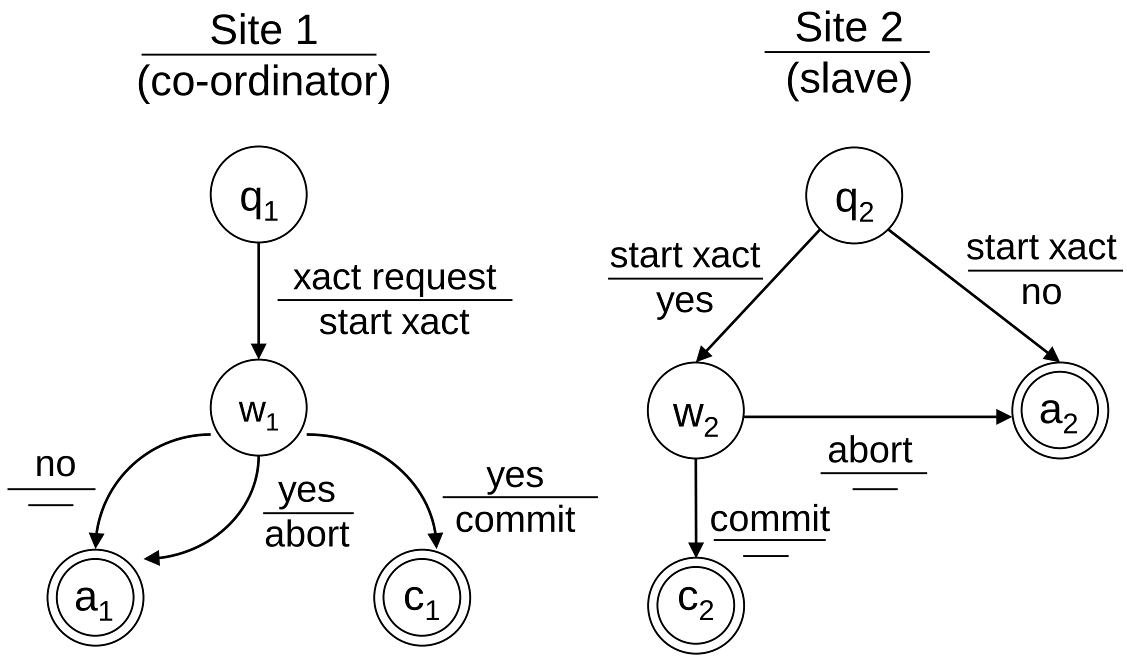
Phase 2 : Everybody writes the results into the database

- **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

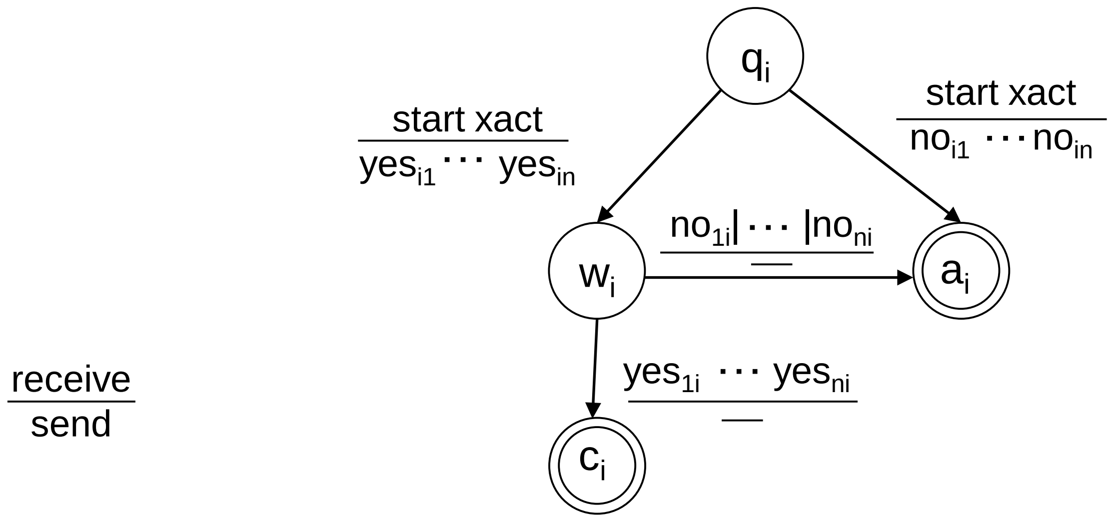
Global Commit Rule:

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

Local Protocols for the Centralized Two-Phase Commit Protocol

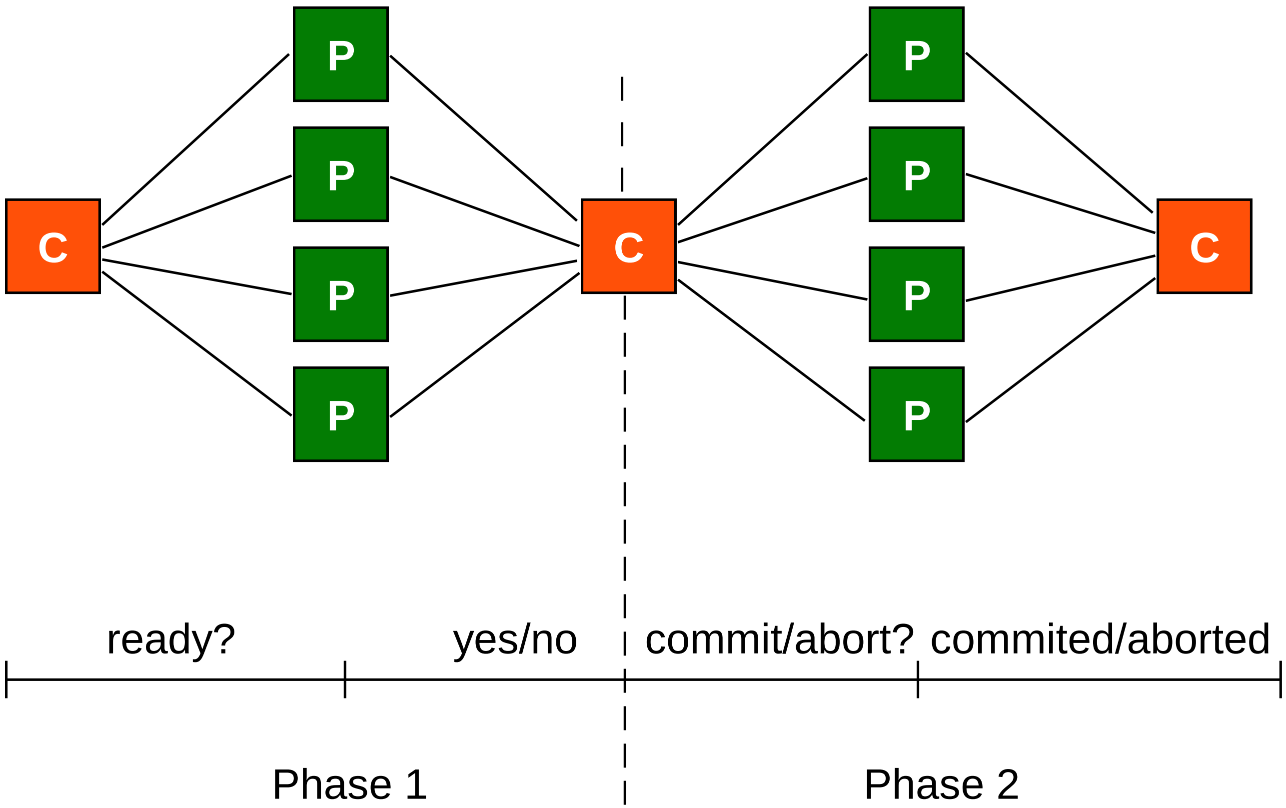


Decentralized Two-Phase Commit Protocol

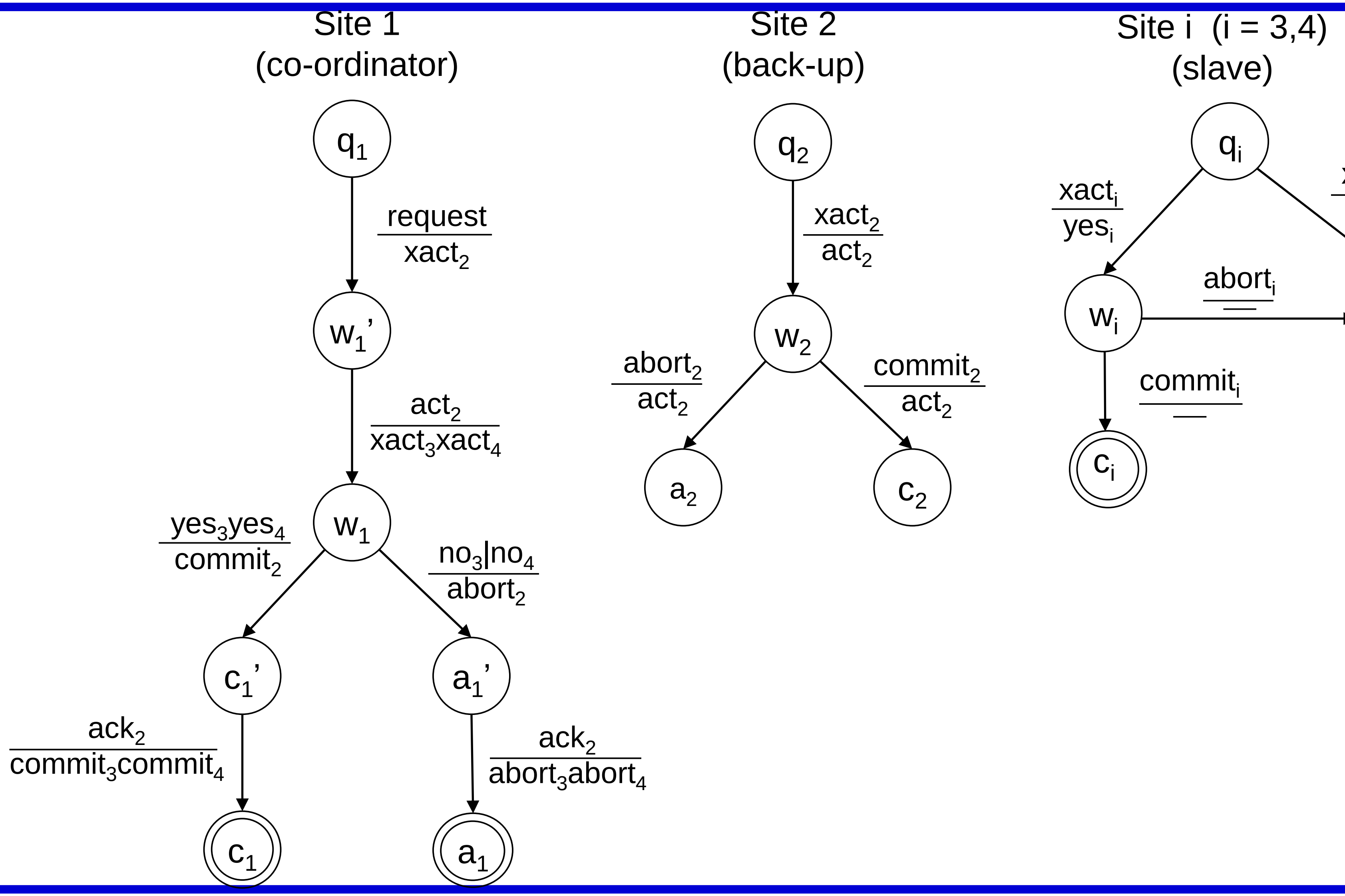


Site i (i = 1,2,…n)

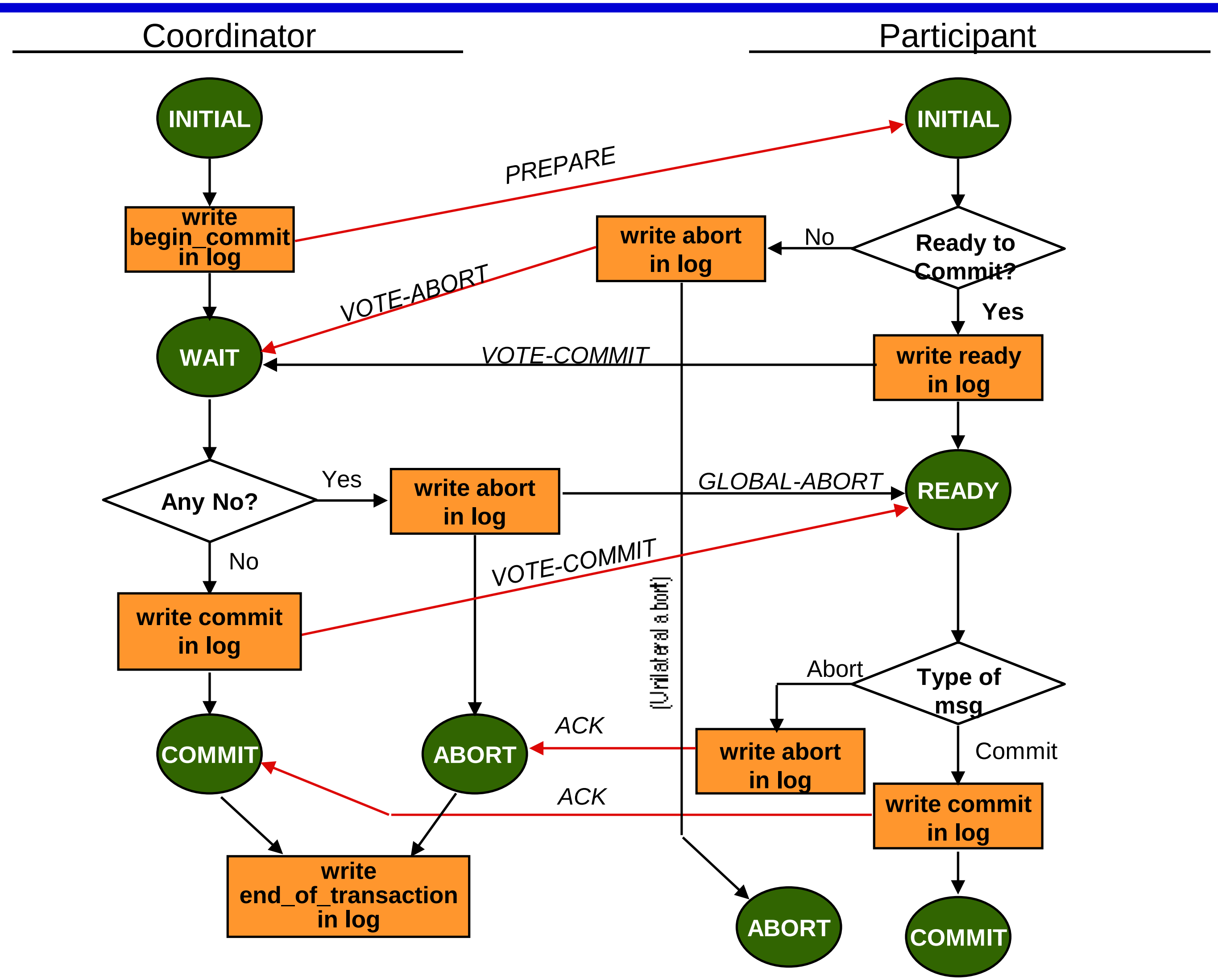
Centralized 2PC (see book)



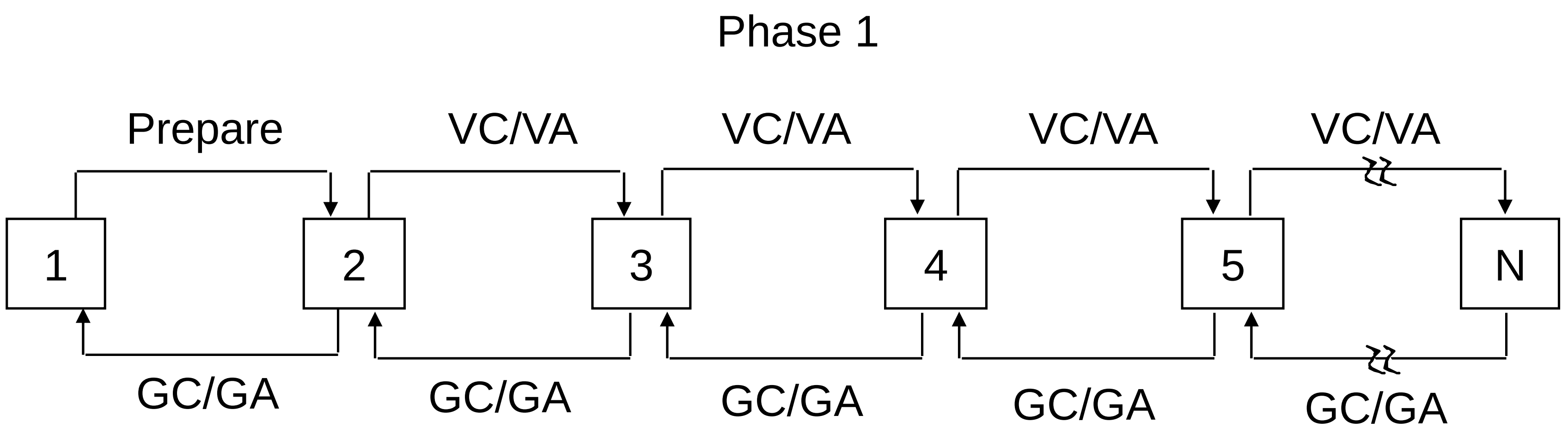
SDD-1 Four-Phase Commit Protocol



2PC Protocol Actions (see book)

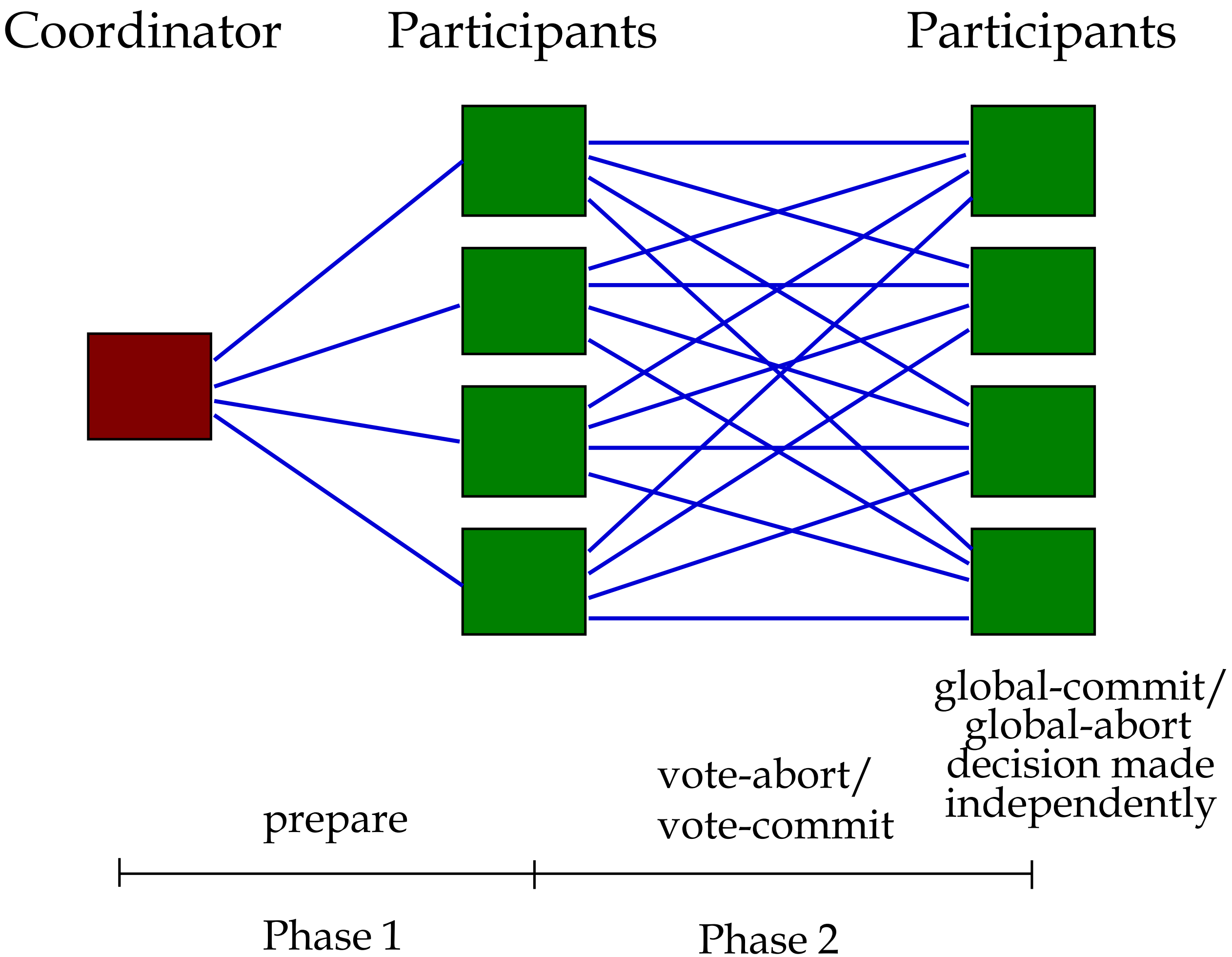


Linear 2PC

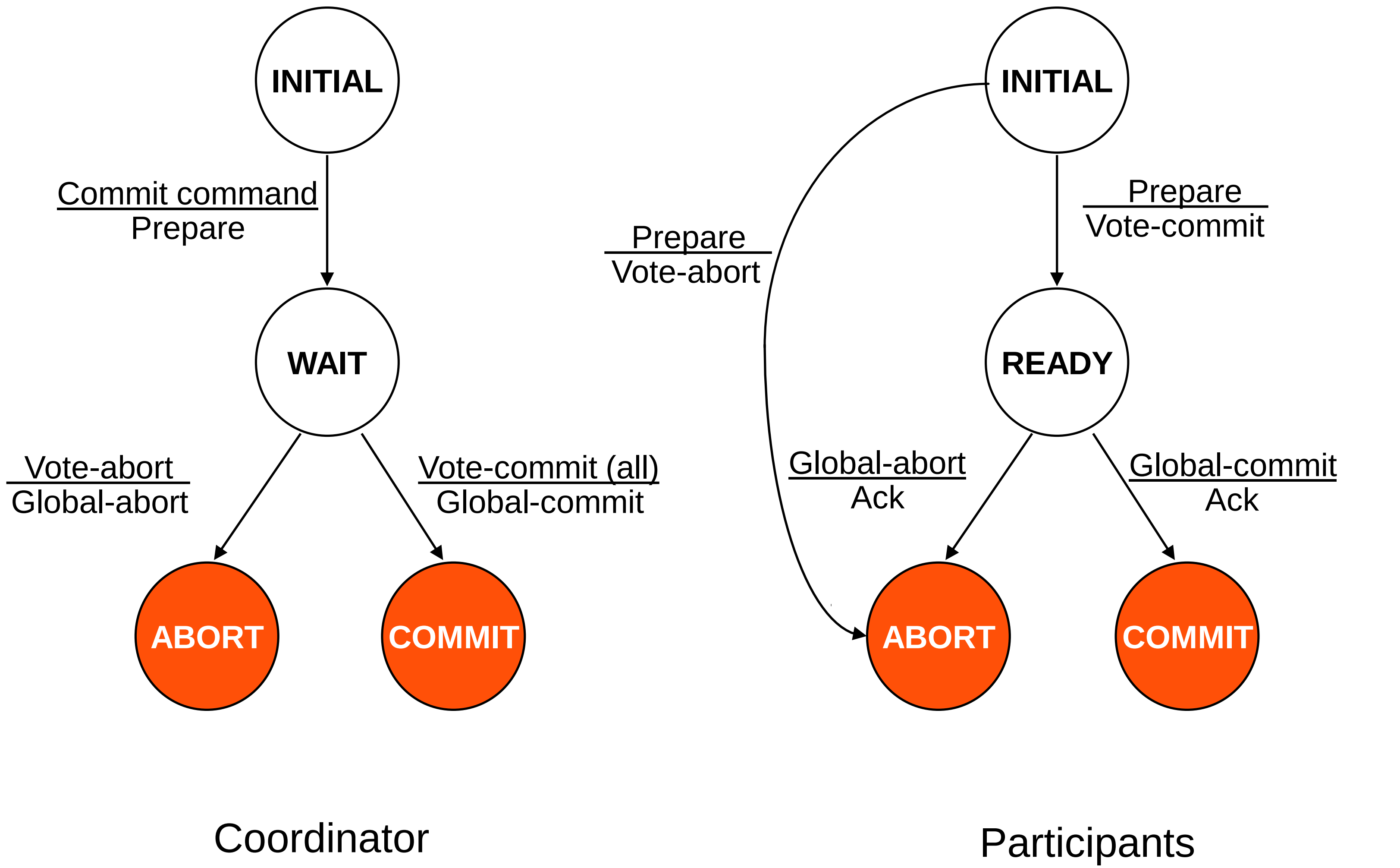


VC: Vote-Commit, VA: Vote-Abort, GC: Global-commit, GA: Global-abort

Distributed 2PC

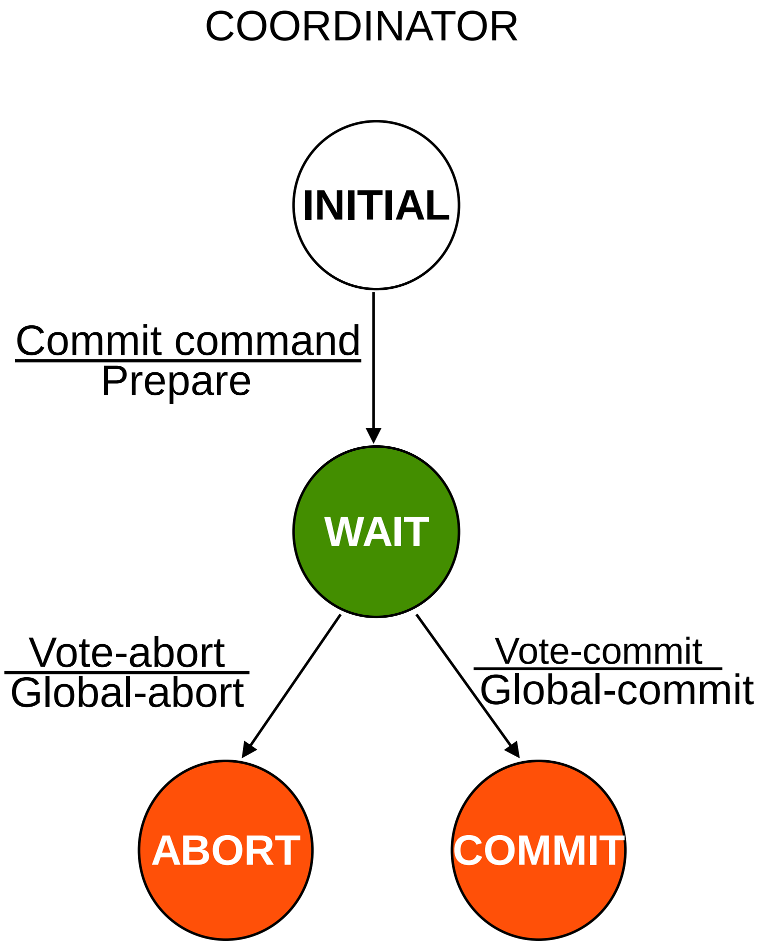


State Transitions in 2PC (see book)



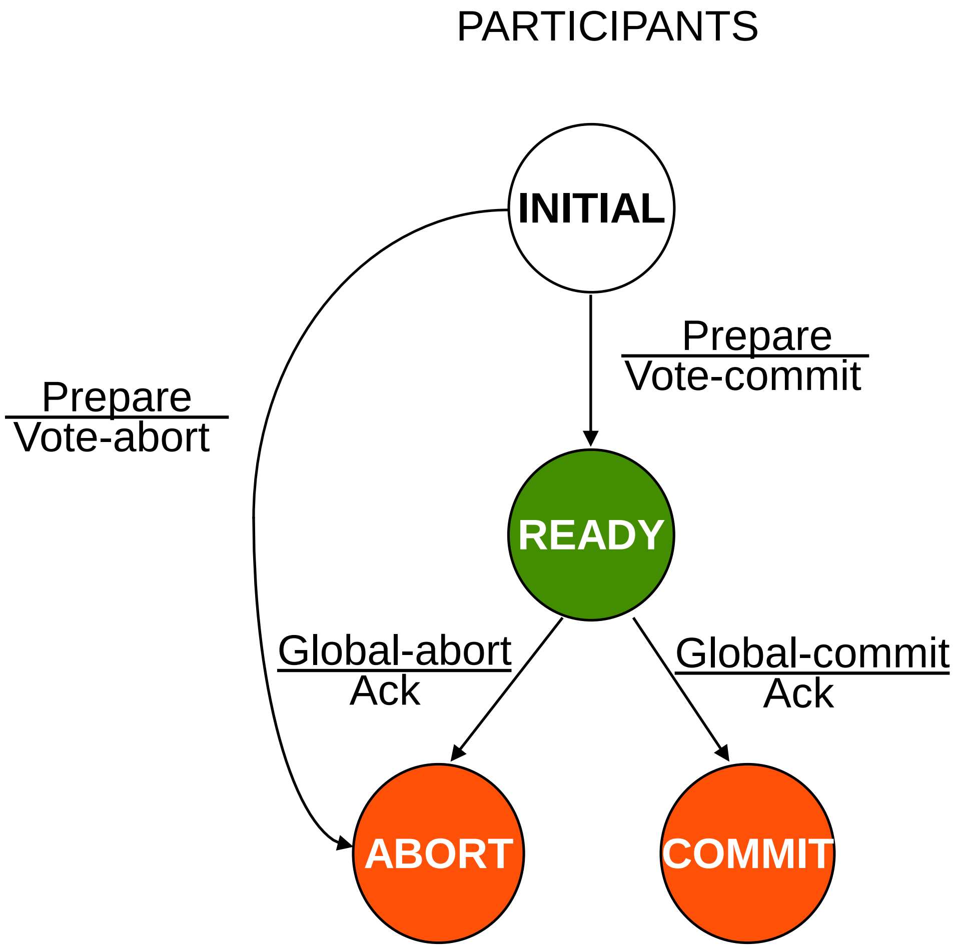
Site Failures - 2PC Termination (see book)

- 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



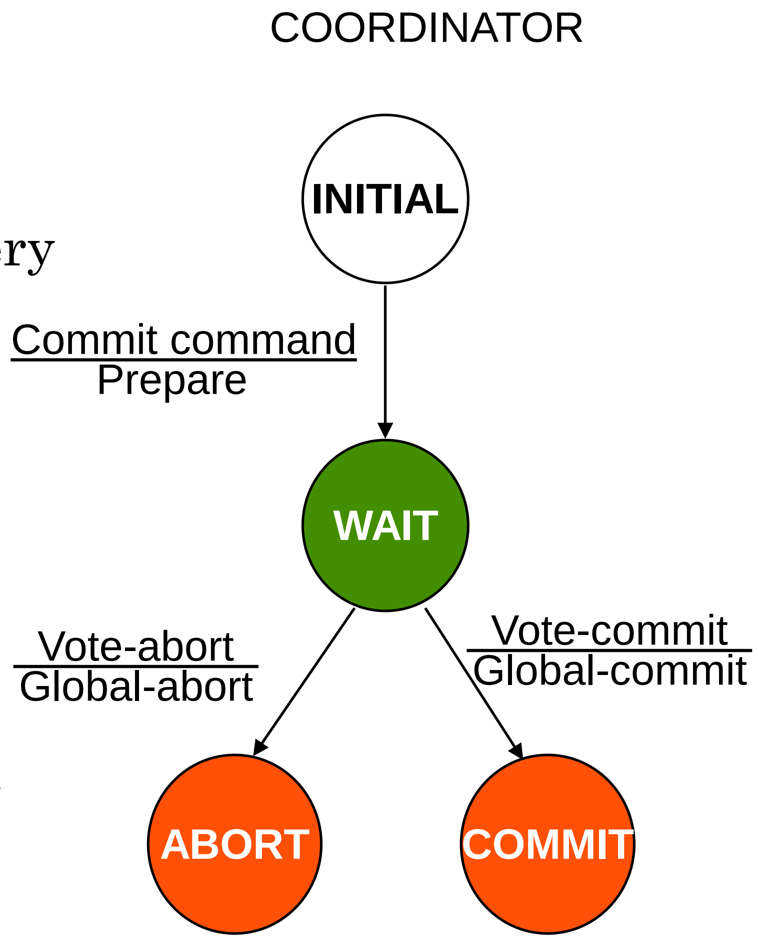
Site Failures - 2PC Termination

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



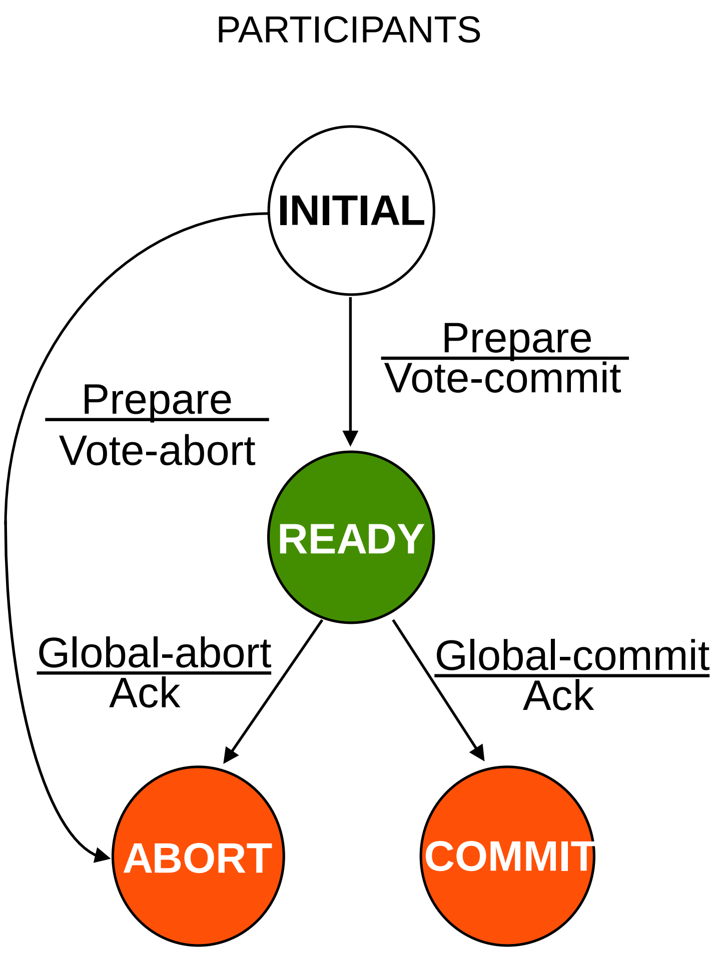
Site Failures - 2PC Recovery

- 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 (see book)

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 (see book)

- 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