Commit Algorithms

Hamid Al-Hamadi

CS 5204

November 17, 2009

Virginia Tech

Agenda

- Fault Tolerance
- Transactional Model
- Commit Algorithms
    - 2-Phase Commit Protocol
    - Failure and Timeout Transitions
    - 3-Phase Commit Protocol
- Summary

Fault tolerance

Causes of failure in a distributed system:

- process failure
- machine failure
- network failure

How to deal with failures:

- transparent: transparently and completely recover from all failures
- predictable: exhibit a well defined failure behavior

Transaction Model

Transaction

- A sequence of actions (typically read/write), each of which is executed at one or more sites, the combined effect of which is guaranteed to be atomic.

A transaction is said to be ATOMIC when it satisfies the ACID properties:

- Atomicity: either all or none of the effects of the transaction are made permanent.
- Consistency: the effect of concurrent transactions is equivalent to some serial execution.
- Isolation: transactions cannot observe each other's partial effects.
- Durability: once accepted, the effects of a transaction are permanent (until changed again, of course).

Commit Algorithms

What is a Commit Algorithm?

Possible definition: Algorithm run by all nodes involved in a distributed transaction s.t. :

- Either **all nodes agree to commit** (transaction as a whole commits) or
- **All nodes agree to Abort** (transaction as a whole Aborts).

Variations:

- blocking vs. non-blocking protocols (non-failed sites must wait (can continue) while failed sites recover)
- independent recovery (failed sites can recover using only local information)
- Type of failures which can be tolerated

Commit Algorithms

Environment

Each node is assumed to have:

- data stored in a partially/full replicated manner
- stable storage (information that survives failures)
- logs (a record of the intended changes to the data: write ahead, UNDO/REDO)
- locks (to prevent access to data being used by a transaction in progress)

Generals Paradox :

- 2 Generals need to agree to attack at the same time
- Each general needs to confirm that the other general has agreed to attack.

Since message loss is possible, confirmations can get loss-> need to get confirmation

Result is that the 2 generals **can never agree on attacking**.

Commit Algorithms

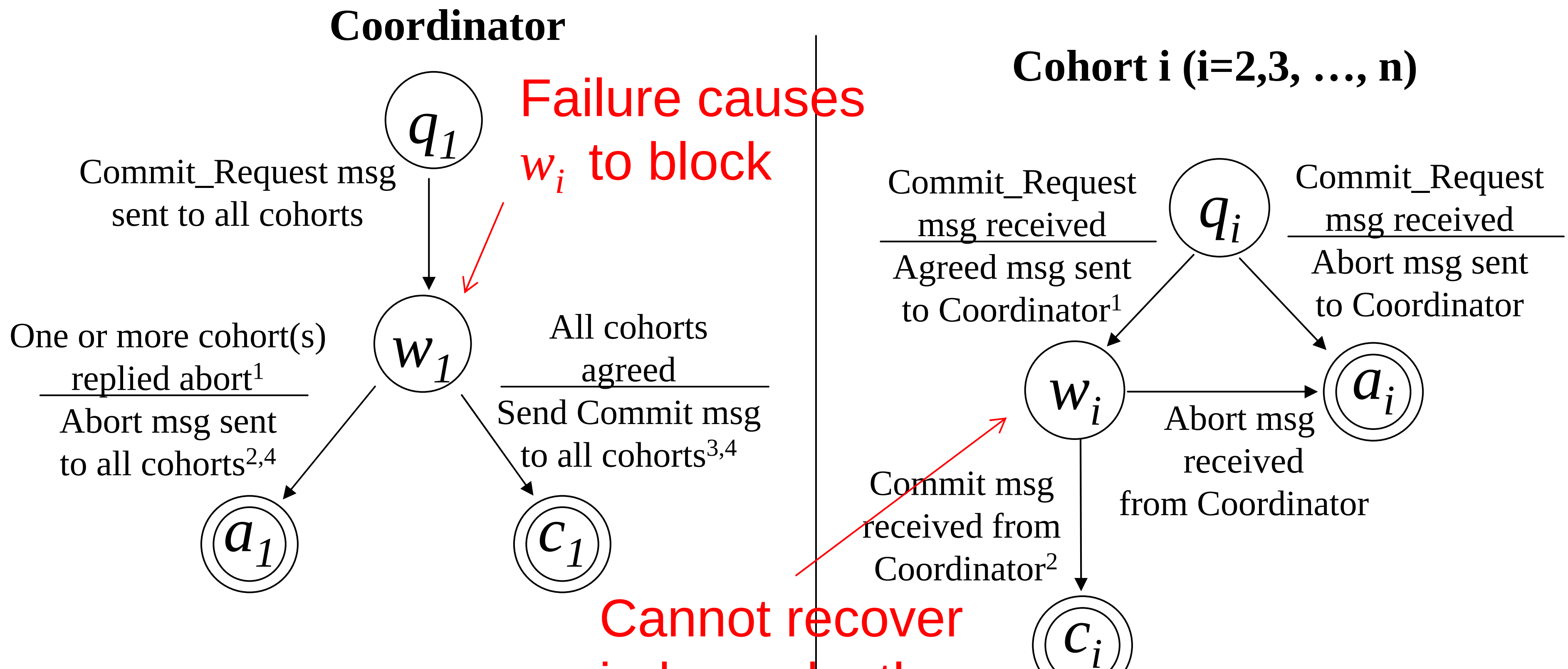
Goal:

Build a commit algorithm that is correct in the presence of failure such that either all nodes involved in the distributed transaction commit or they all abort.

Topology:

- n nodes:
  - 1 Coordinator
  - (n -1) Cohorts

2-phase Commit Protocol



1. Assume ABORT if there is a timeout
2. First, writes ABORT record to stable storage.
3. First, writes COMMIT record to stable storage.
4. Write COMPLETE record when all msgs confirmed.

1. First, write UNDO/REDO logs on stable storage.
2. Writes COMPLETE record; releases locks

Site Failures

Who Fails	At what point	Actions on recovery
Coordinator	before writing Commit	Send Abort messages
Coordinator	after writing Commit but before writing Complete	Send Commit messages
Coordinator	after writing Complete	None.
Cohort	before writing Undo/Redo	None. Abort will occur.
Cohort	after writing Undo/Redo	Wait for message from Coordinator.

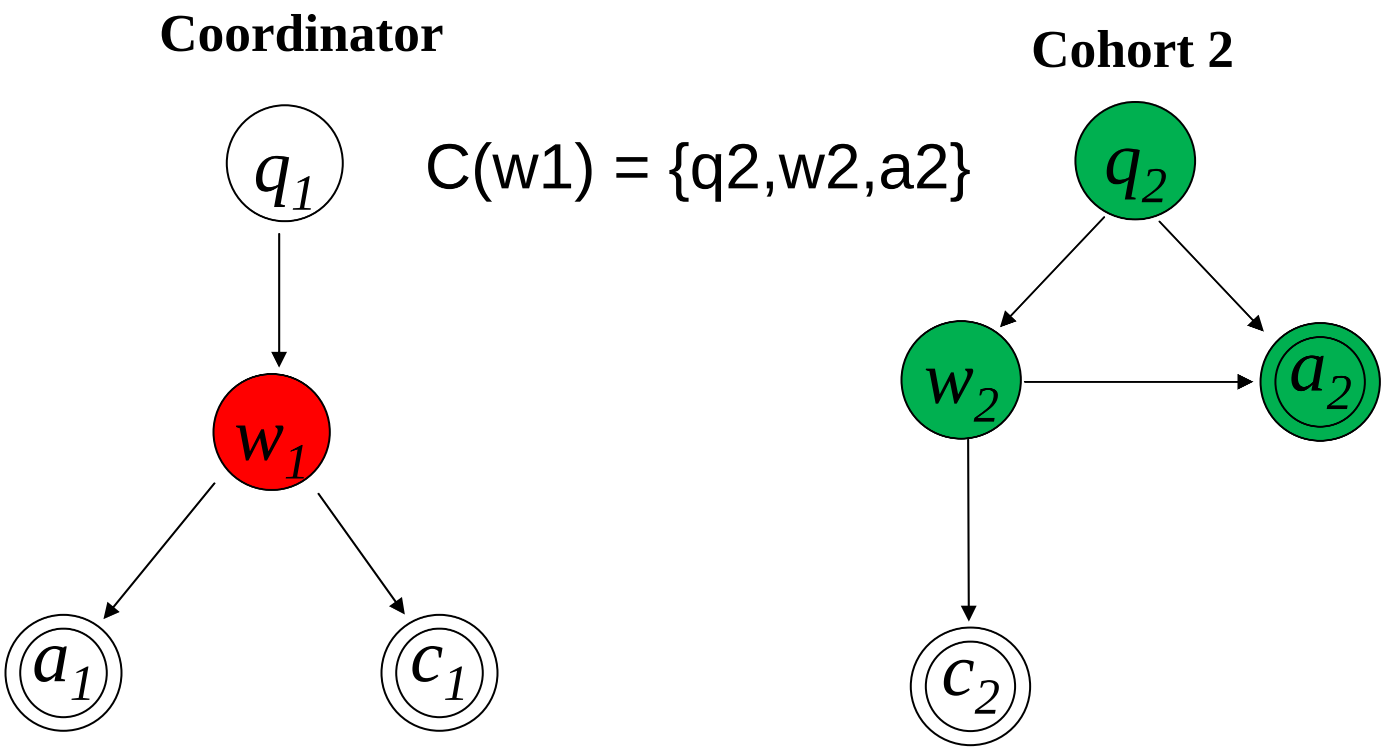
Definitions

Synchronous

A protocol is synchronous if any two sites can never differ by more than one transition.

Concurrency Set

For a given state, s, at one site the concurrency set, C(s), is the set of all states in which all other sites can be.



Sender set

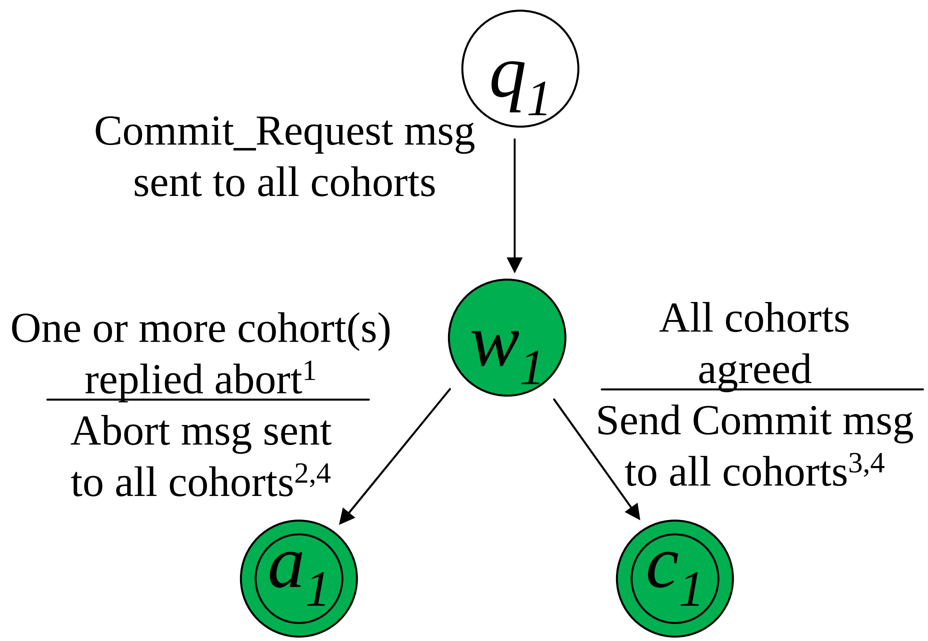
For a given state, s, at one site, the sender set, S(s), is the set of all other sites that can send messages that will be received in state s.

What causes blocking

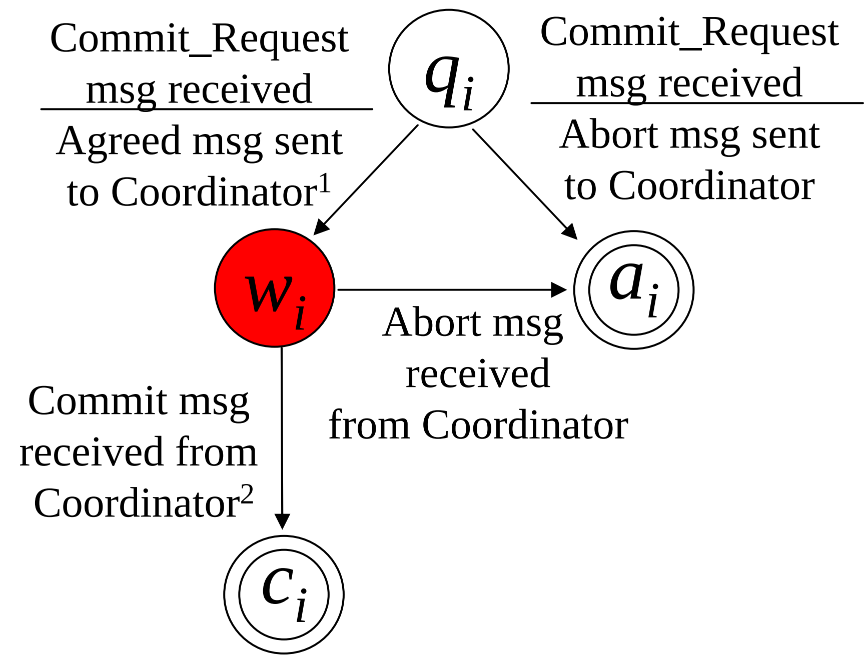
Blocking occurs when a site's state, s, has a concurrency set, C(s), that contains both commit and abort states.

Blocking of 2-phase Commit Protocol

Coordinator



Cohort i (i=2,3, …, n)



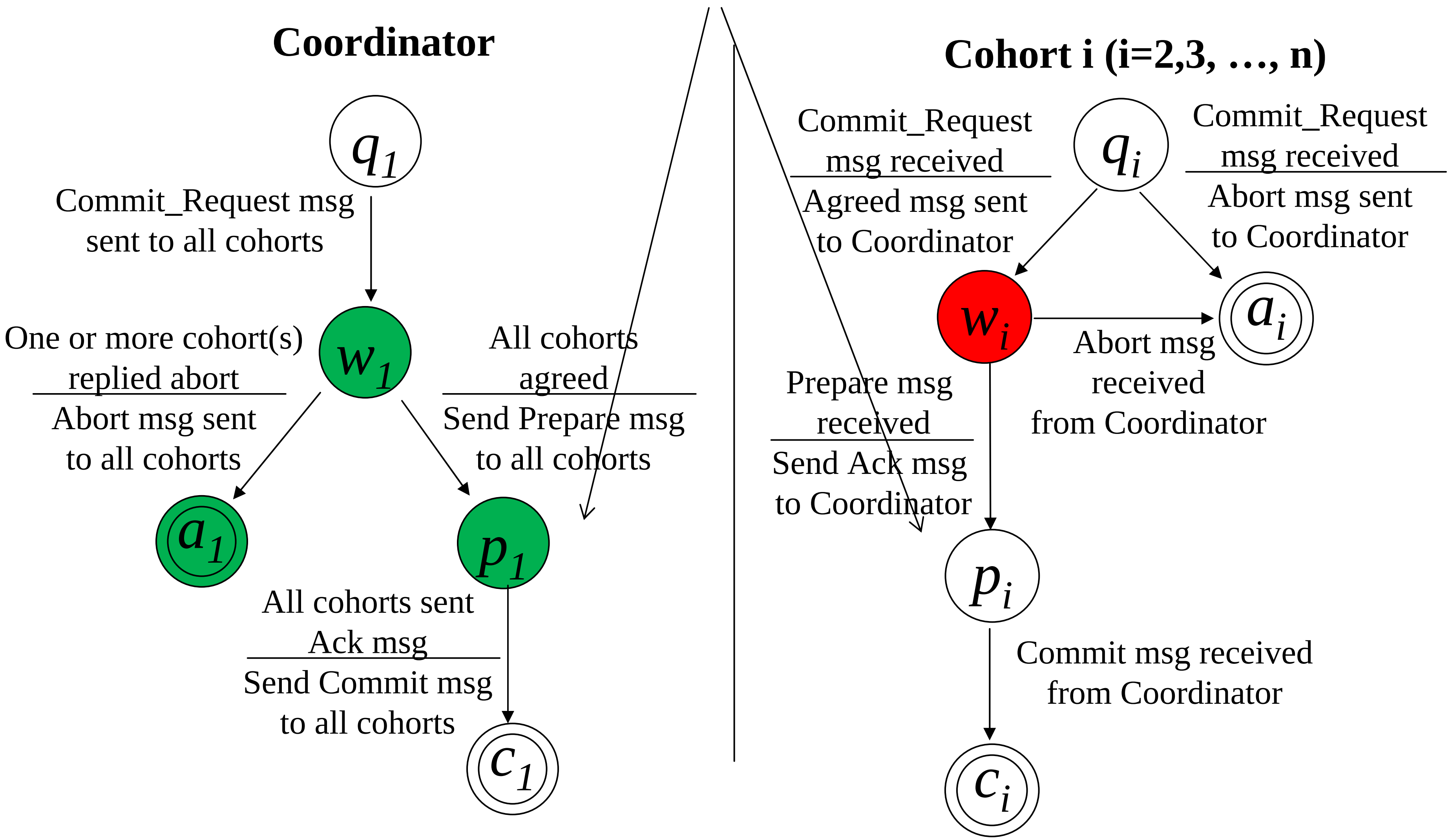
1 <u>Solution:</u>

2 Introduce additional states -> additional messages (to allow transitions to/from these new states). -> adding at least one more "phase". storage.

3

4 s

Added prepare states

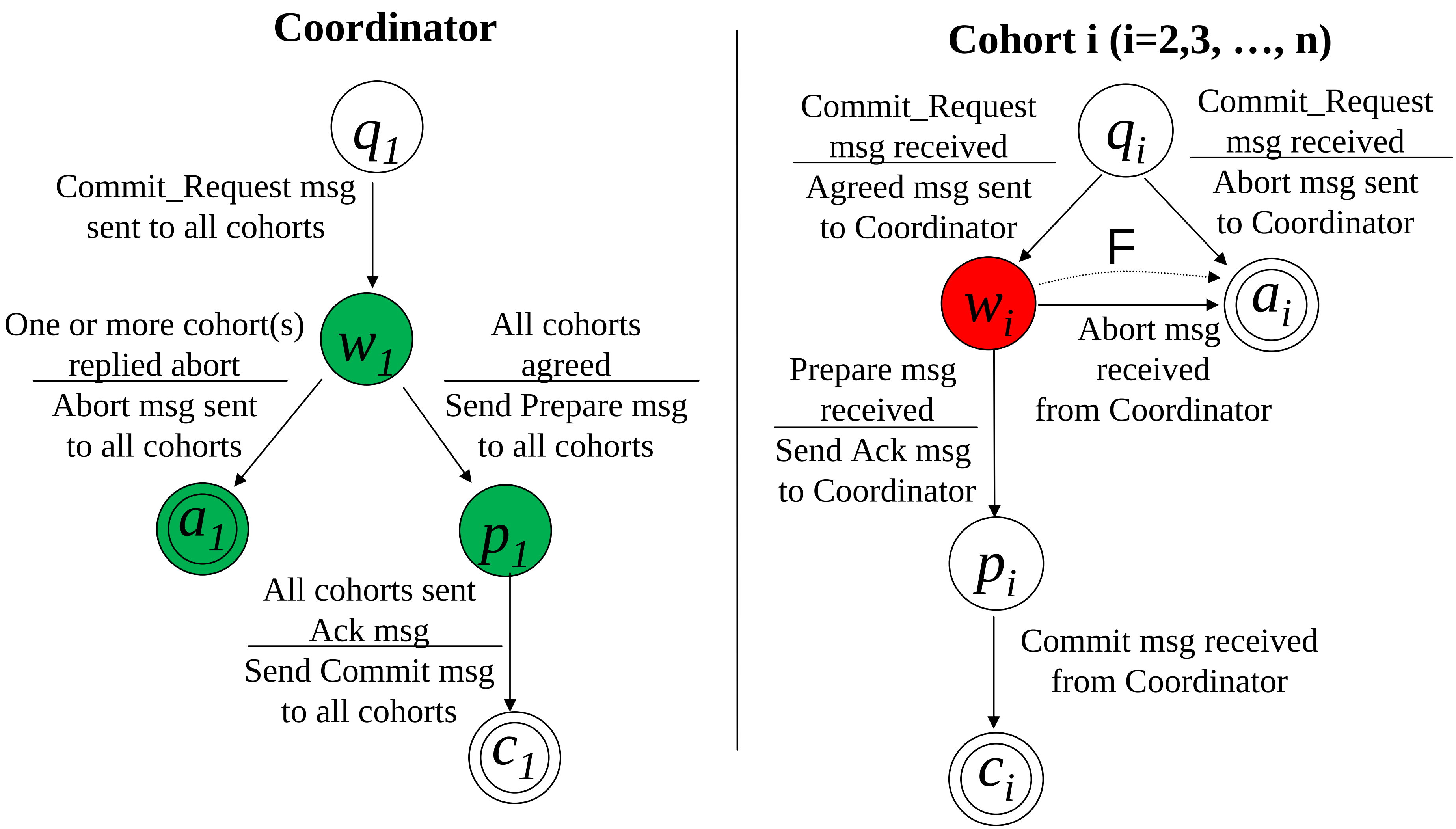


Failure and Timeout Transitions

Failure Transition Rule

For every nonfinal state s, if C(s) contains a commit, then add failure transition to a commit state; otherwise, add failure transition from s to an abort state

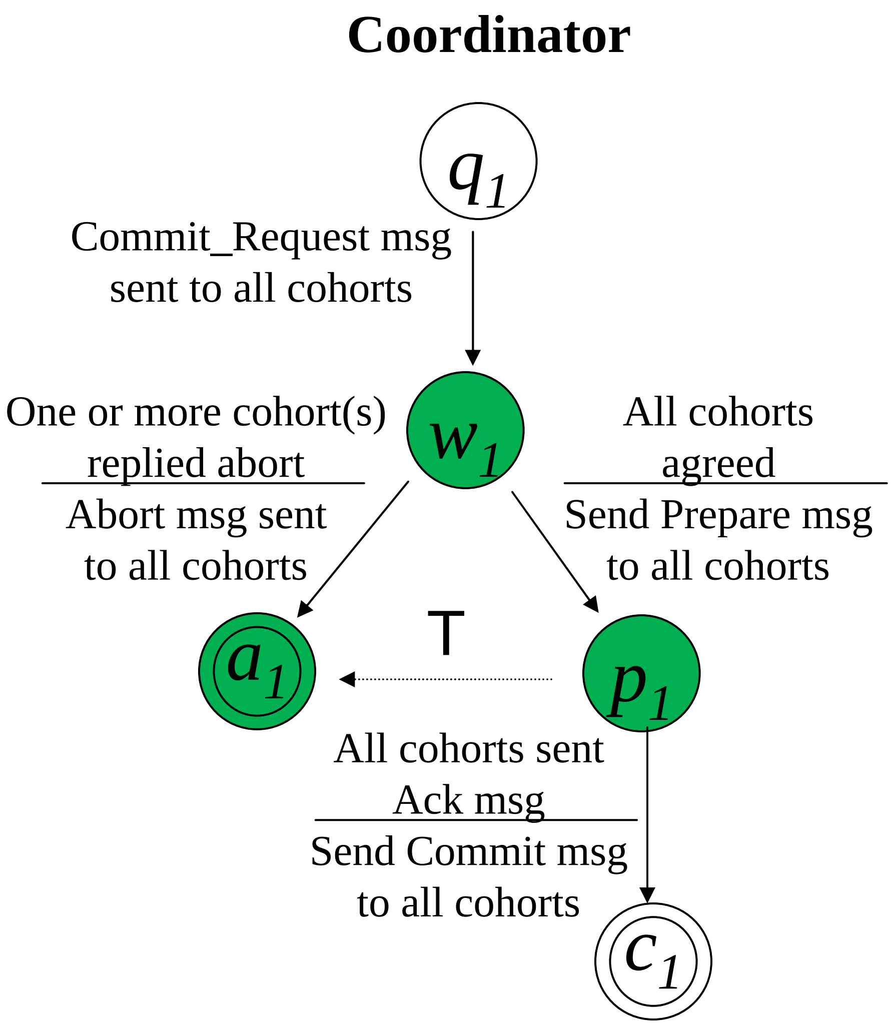
Adding a Failure Transition

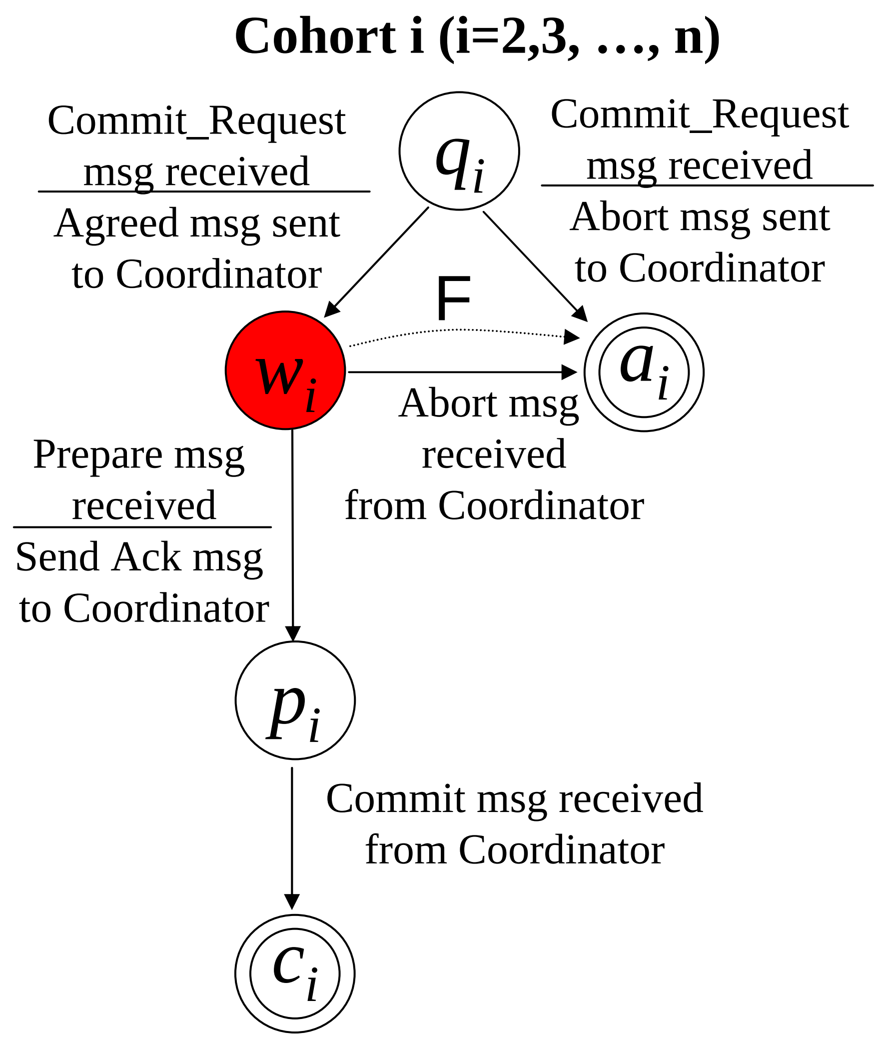


Timeout Transition Rule

For every nonfinal state s, if j is in S(s) and j has failure transition to commit (abort) state then add timeout transition from s to commit (abort) state

Adding a Timeout Transition

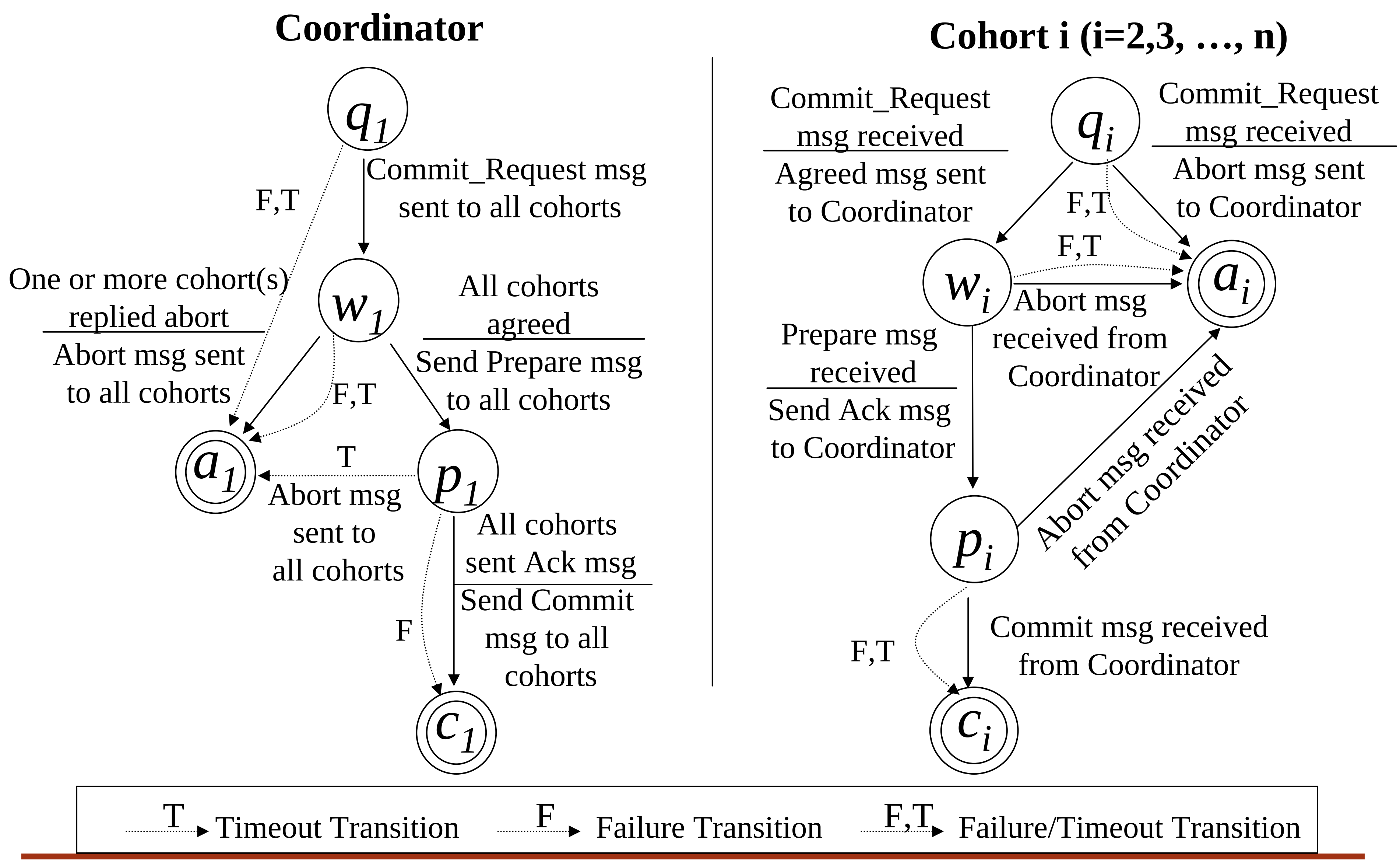




Adding a prepared state, and using Failure and Timeout transmissions in the 3PC protocol allows the protocol to be resilient to a **single site** failure.

After adding all transitions we get:

3-Phase Commit Protocol



Summary

- Commit Algorithms are used to commit distributed transactions across multiple nodes S.T either all nodes commit or all abort.

- Commit algorithms differ in aspects of blocking, independent recovery, and types of failures which can be tolerated.

- 2-phase commit algorithm suffers from blocking and lacks independent recovery.

- 3-phase commit algorithm uses prepared states and applies transition rules, this gives it the properties of:
- Non-blocking
- Can recovery independently (-> only resilient to a single site failure).

Questions?