Commit Protocols

Fault Tolerance

Causes of failure:

- process failure
- machine failure
- network failure

Goals:

- transparent: mask (i.e., completely recover from) all failures, or
- predictable: exhibit a well defined failure behavior

Elements:

- Atomic Transactions
- commitment (commit protocols)
  - generals paradox (message loss)
  - 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)

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.

Atomic Transactions

- 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).

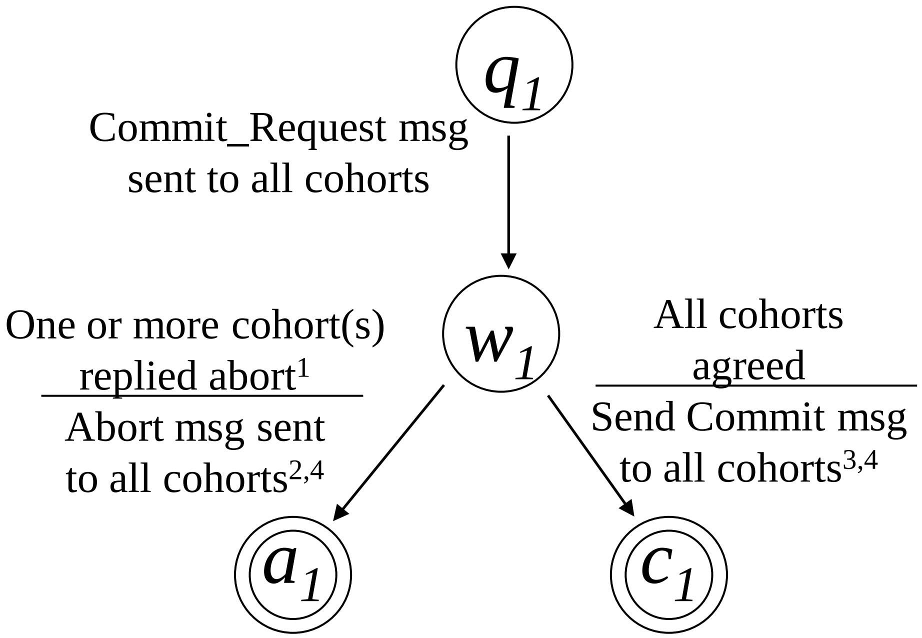
Environment

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)

2-phase Commit Protocol

Coordinator



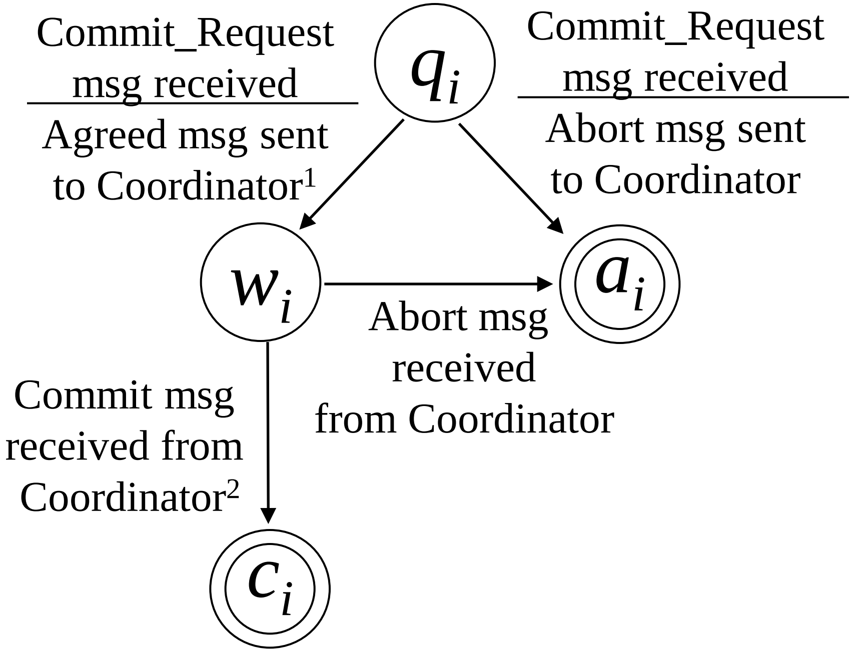
Commit_Request msg sent to all cohorts

One or more cohort(s) replied abort[1]
Abort msg sent to all cohorts[2,4]

All cohorts agreed
Send Commit msg to all cohorts[3,4]

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.

Cohort i (i=2,3, …, n)



Commit_Request msg received
Agreed msg sent to Coordinator[1]

Commit_Request msg received
Abort msg sent to Coordinator

Abort msg received from Coordinator

Commit msg received from Coordinator[2]

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. A state transition is caused by sending or receiving a message.

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??

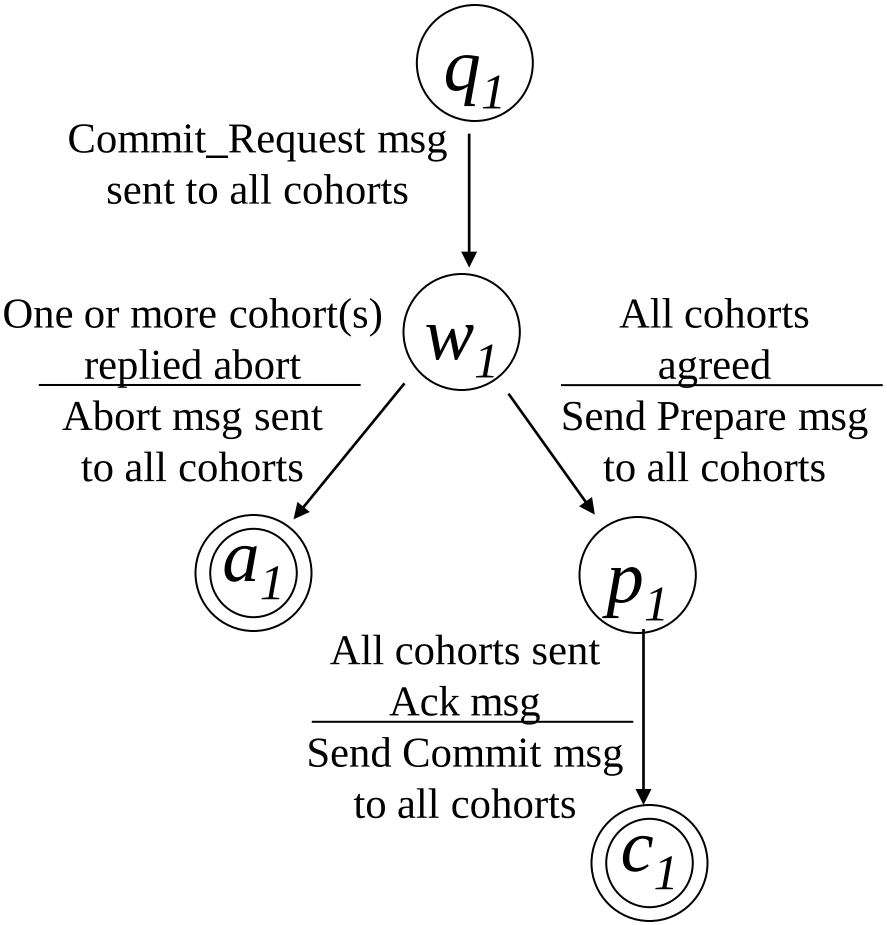
Blocking occurs when a site’s state, s, has a concurrency set, C(s), that contains both commit and abort states.

Solution:

Introduce additional states. This implies adding additional messages (to allow transitions to/from these new states). This implies adding at least one more “phase”.

3-phase Commit Protocol

Coordinator



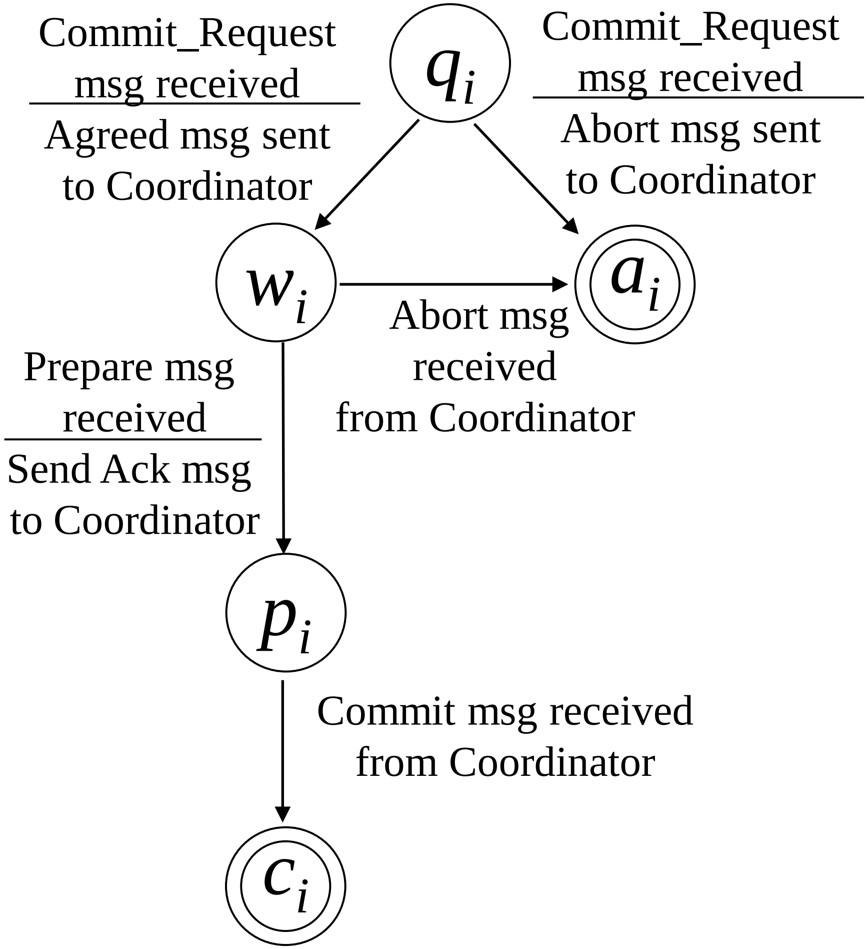
Commit_Request msg
sent to all cohorts

One or more cohort(s)
replied abort
Abort msg sent
to all cohorts

All cohorts
agreed
Send Prepare msg
to all cohorts

All cohorts sent
Ack msg
Send Commit msg
to all cohorts

Cohort i (i=2,3, …, n)



Commit_Request
msg received
Agreed msg sent
to Coordinator

Commit_Request
msg received
Abort msg sent
to Coordinator

Abort msg
received
from Coordinator

Prepare msg
received
Send Ack msg
to Coordinator

Commit msg received
from Coordinator

Rules for Adding New Transitions

Failure Transition Rule

For every nonfinal state, s, in the protocol, if C(s) contains a commit, then assign a failure transition from s to a commit state; otherwise, assign a failure transition from s to an abort state.

Timeout Transition Rule

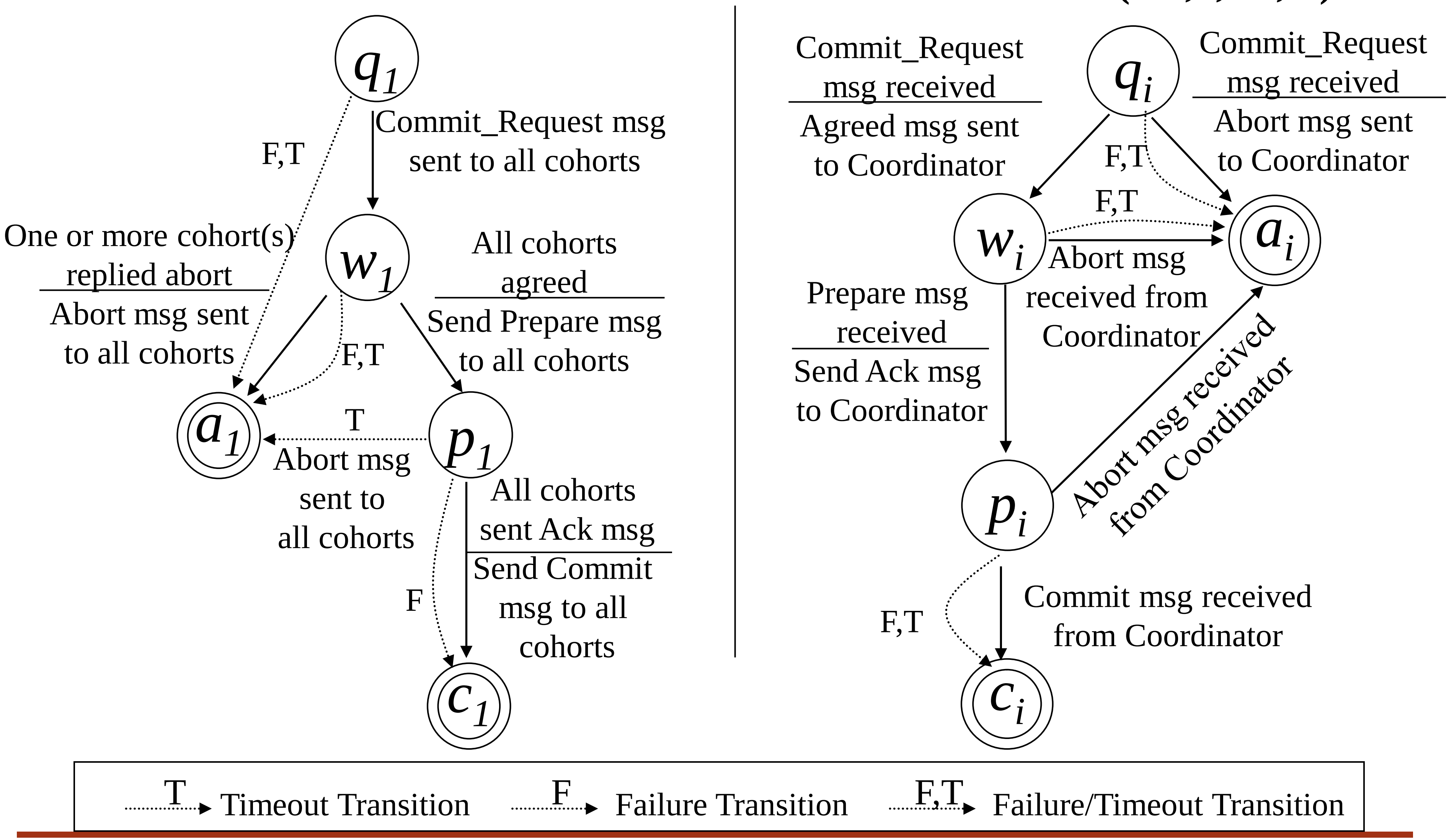
For each nonfinal state, s, if site j is in S(s), and site j has a failure transition to a commit (abort) state, then assign a timeout transition from state s to a commit (abort) state.

Using these rules in the three phase commit protocol allows the protocol to be resilient to a single site failure.

Timeout and Failure Transitions

Coordinator

Cohort i (i=2,3, …, n)



Commit_Request msg sent to all cohorts

One or more cohort(s) replied abort
Abort msg sent to all cohorts

All cohorts agreed
Send Prepare msg to all cohorts

Abort msg sent to all cohorts

All cohorts sent Ack msg
Send Commit msg to all cohorts

Commit_Request msg received
Agreed msg sent to Coordinator

Commit_Request msg received
Abort msg sent to Coordinator

Abort msg received from Coordinator

Prepare msg received
Send Ack msg to Coordinator

Abort msg received from Coordinator

Commit msg received from Coordinator

T → Timeout Transition F → Failure Transition F,T → Failure/Timeout Transition