CSE 544
Principles of Database Management Systems

Magdalena Balazinska

Fall 2007

Lecture 13 - Distribution: transactions

References

- **Transaction Management in the R* Distributed Database Management System.**

  C. Mohan, B. Lindsay, and R. Obermarck. TODS 11(4). 1986.

- **Database management systems.**

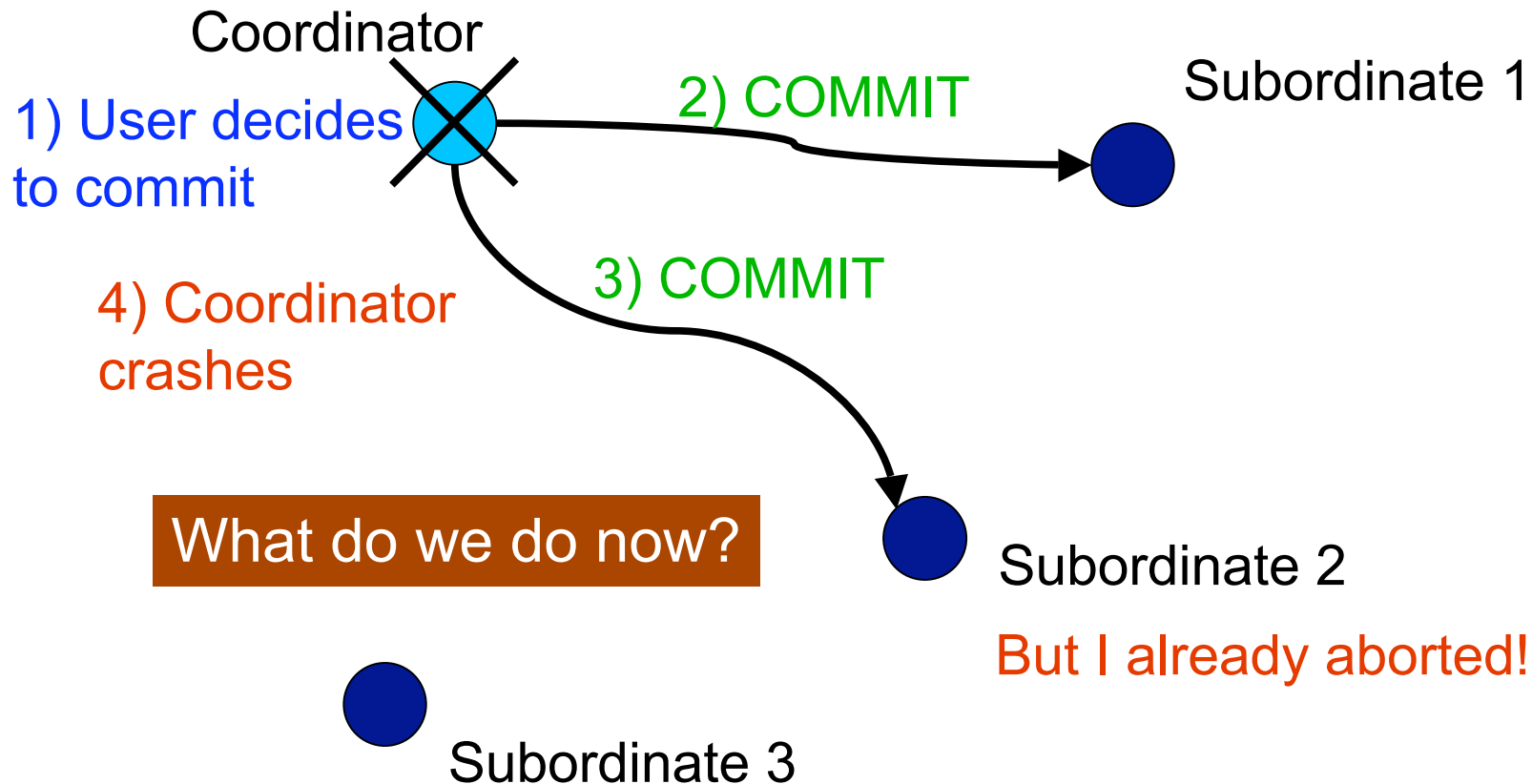 Ramakrishnan and Gehrke.

  Third Ed. **Chapter 22**

Distributed Transactions

- **Concurrency control**

- **Failure recovery**
  - Transaction must be committed at all sites or at none of the sites!
    - No matter what failures occur and when they occur
  - Two-phase commit protocol (2PC)

Distributed Concurrency Control

- In theory, different techniques are possible

  - Pessimistic, optimistic, locking, timestamps

- In practice, distributed two-phase locking

  - Simultaneously hold locks at all sites involved

- Deadlock detection techniques

  - Global wait-for graph (not very practical)

  - Timeouts

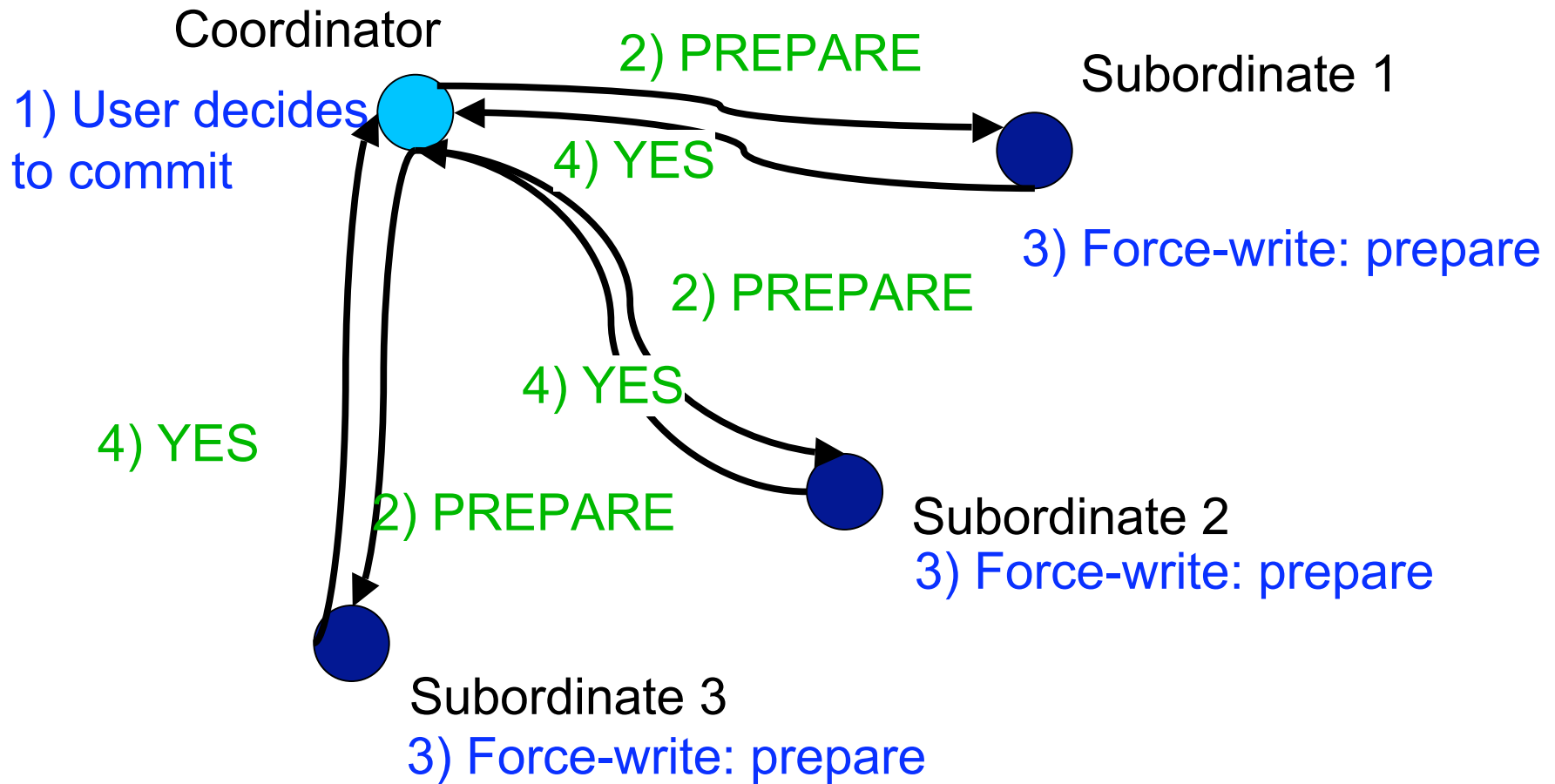- If deadlock: abort least costly local transaction

Two-Phase Commit: Motivation

Coordinator

1) User decides to commit

2) COMMIT

Subordinate 1

4) Coordinator crashes

3) COMMIT

What do we do now?

Subordinate 2

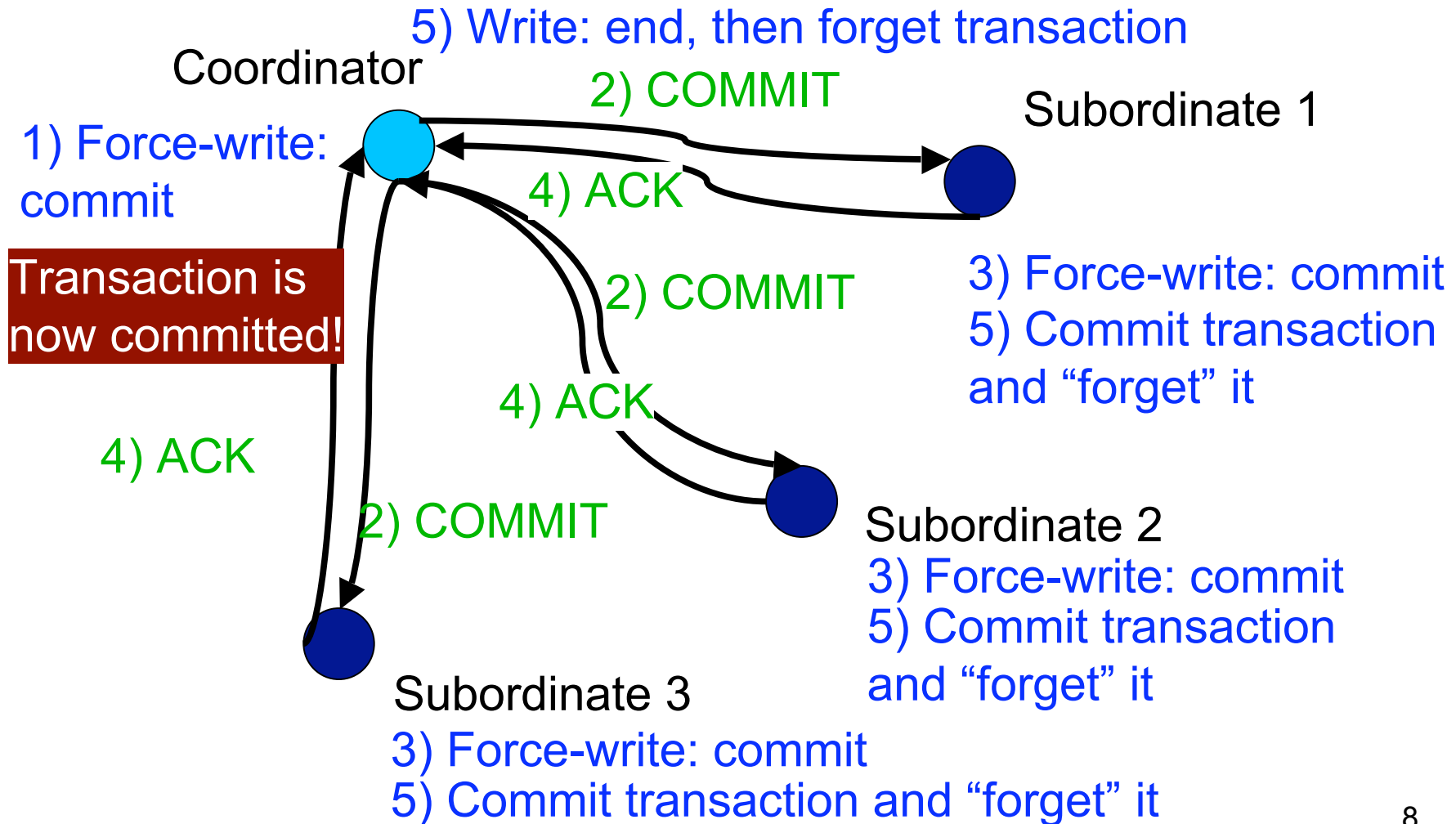But I already aborted!

Subordinate 3

Two-Phase Commit Protocol

- One coordinator and many subordinates
  - Phase 1: prepare
  - Phase 2: commit or abort
  - Log records for 2PC include transaction and coordinator ids
  - Coordinator also logs ids of all subordinates

- Principle
  - When a process makes a decision: vote yes/no or commit/abort
  - Or when a subordinate wants to respond to a message: ack
  - First force-write a log record (to make sure it survives a failure)
  - Only then send message about decision

2PC: Phase 1, Prepare

Coordinator

2) PREPARE

Subordinate 1

1) User decides to commit

4) YES

3) Force-write: prepare

2) PREPARE

4) YES

4) YES

Subordinate 2
3) Force-write: prepare

2) PREPARE

Subordinate 3
3) Force-write: prepare

2PC: Phase 2, Commit

5) Write: end, then forget transaction

Coordinator

2) COMMIT

Subordinate 1

1) Force-write: commit

4) ACK

Transaction is now committed!

3) Force-write: commit
5) Commit transaction and "forget" it

2) COMMIT

4) ACK

4) ACK

Subordinate 2
3) Force-write: commit
5) Commit transaction and "forget" it

2) COMMIT

Subordinate 3
3) Force-write: commit
5) Commit transaction and "forget" it

8
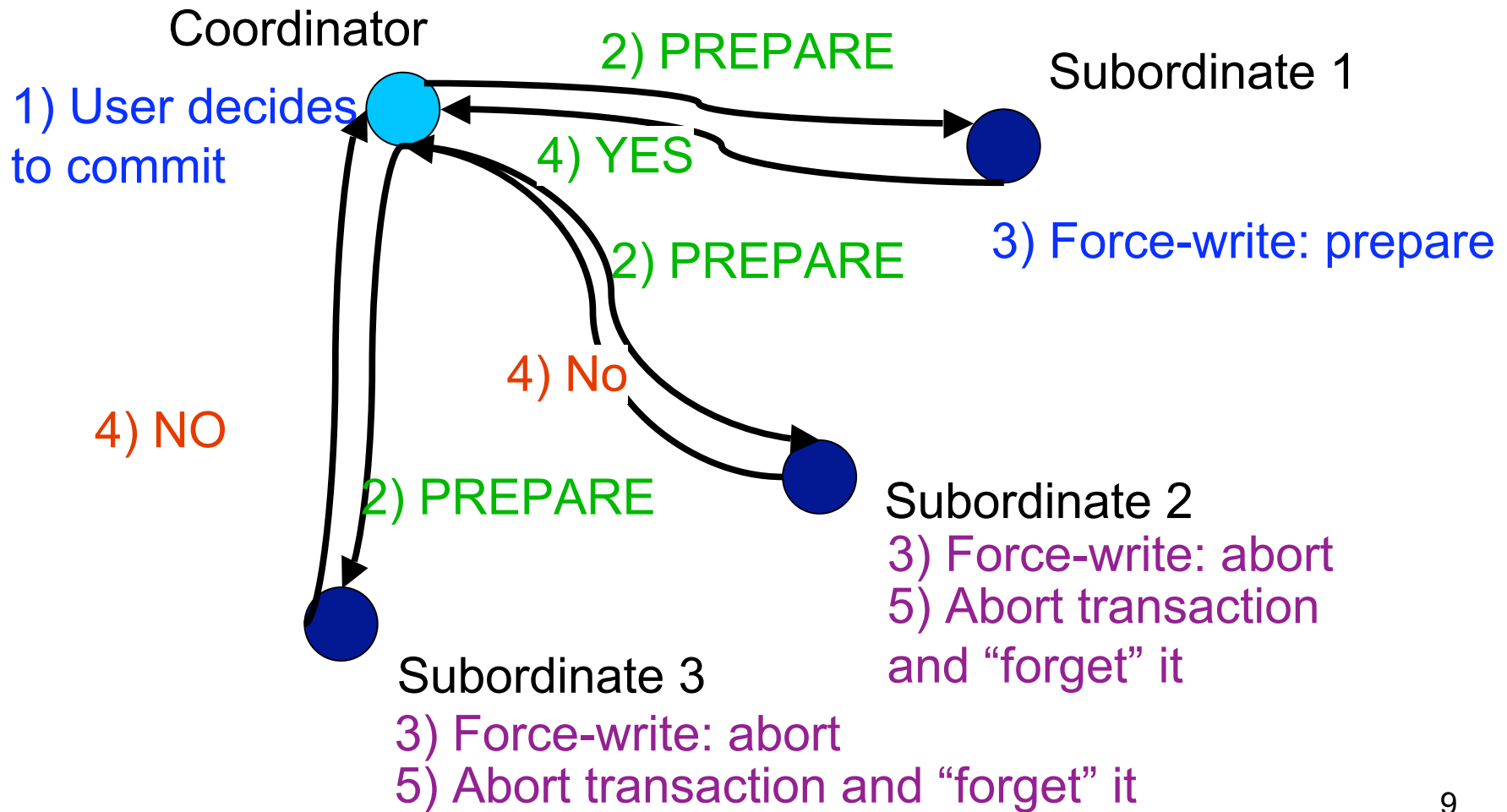
2PC with Abort

Coordinator

1) User decides to commit

2) PREPARE

4) YES

Subordinate 1

3) Force-write: prepare

2) PREPARE

4) No

4) NO

2) PREPARE

Subordinate 2
3) Force-write: abort
5) Abort transaction and "forget" it

Subordinate 3
3) Force-write: abort
5) Abort transaction and "forget" it

9

2PC with Abort

5) Write: end, then forget transaction

Coordinator

2) ABORT

Subordinate 1

1) Force-write: abort

4) ACK

3) Force-write: abort
5) Abort transaction and "forget" it

Subordinate 2

Subordinate 3
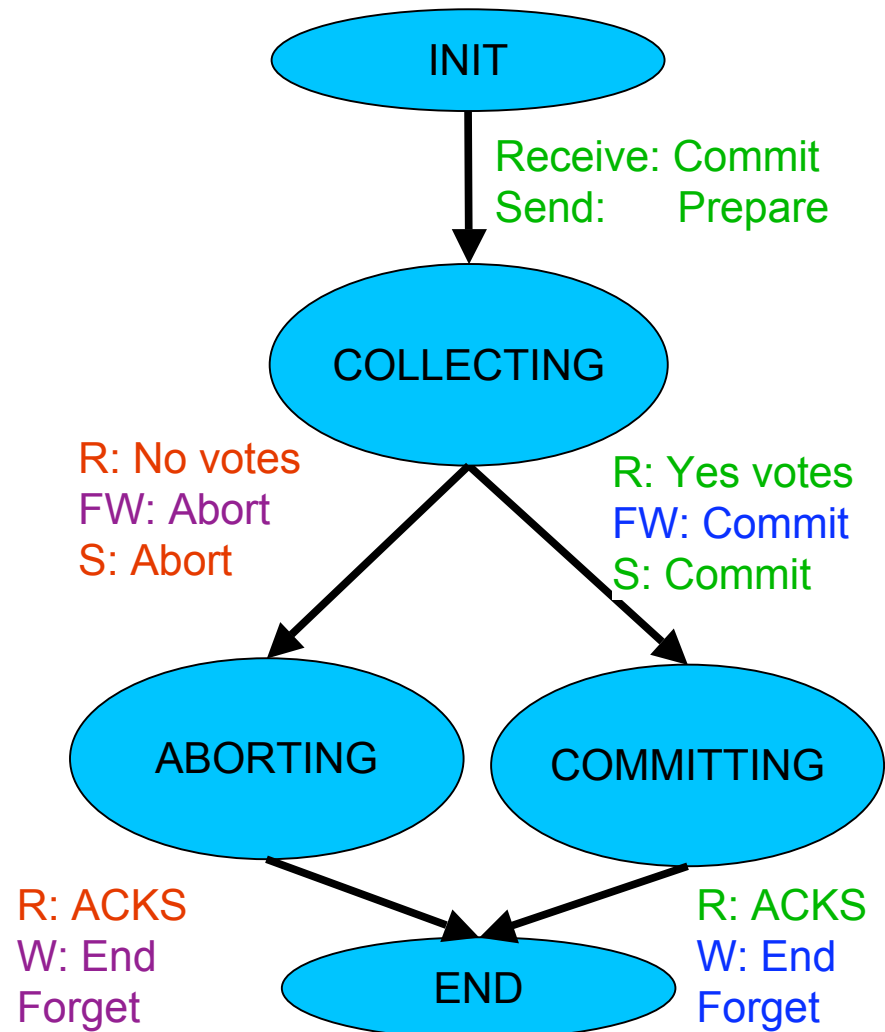
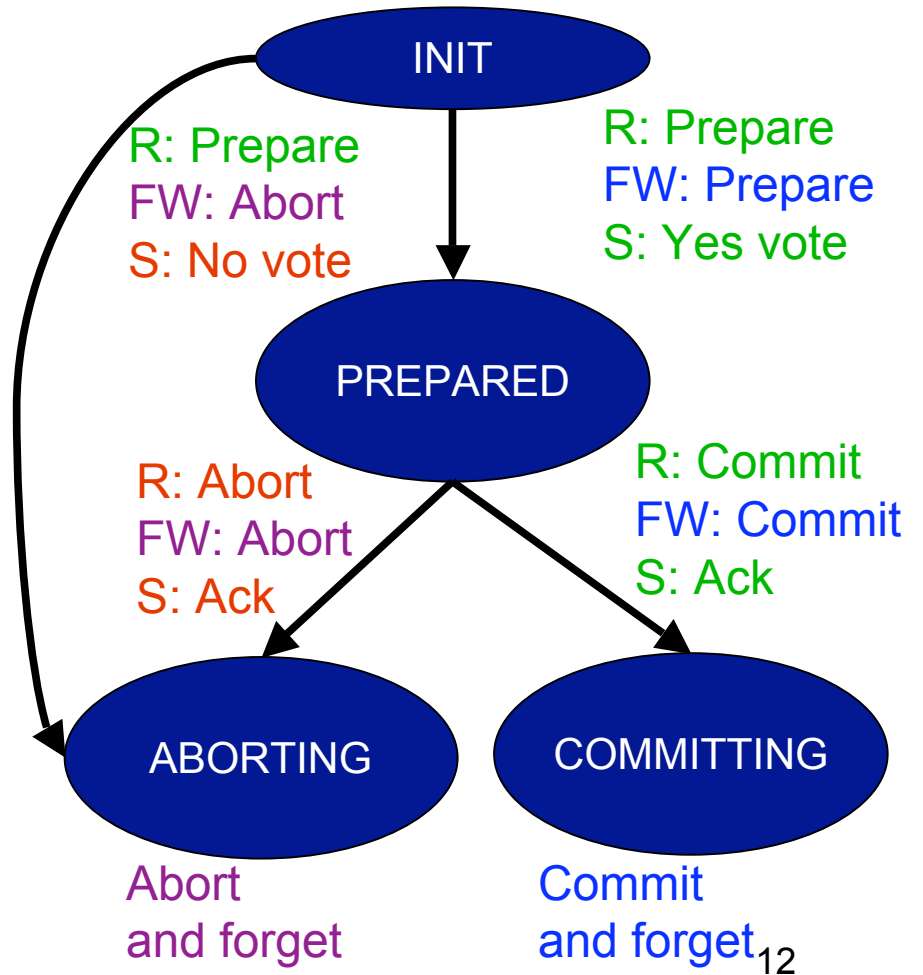# Coordinator State Machine

- All states involve waiting for messages

Subordinate State Machine

- INIT and PREPARED involve waiting

```
                          ┌──────────┐
                          │   INIT   │
                          └──────────┘
      R: Prepare                │         R: Prepare
      FW: Abort                 │         FW: Prepare
      S: No vote                │         S: Yes vote
                                ▼
                          ┌──────────┐
                          │ PREPARED │
                          └──────────┘
      R: Abort            ╱         ╲    R: Commit
      FW: Abort          ╱           ╲   FW: Commit
      S: Ack            ╱             ╲  S: Ack
                       ▼               ▼
               ┌──────────┐      ┌────────────┐
               │ ABORTING │      │ COMMITTING │
               └──────────┘      └────────────┘
               Abort             Commit
               and forget        and forget
```

12

Failure Handling

- Approach 1: no failure detection
  - Can only do retrying & blocking

- Approach 2: timeouts
  - Since unilateral abort is ok,
  - Subordinate can timeout in init state
  - Coordinator can timeout in collecting state
  - Prepared state is still blocking

- 2PC is a blocking protocol

Failure Handling Principles

- If doesn't know anything about transaction respond "abort" to inquiry messages about fate of transaction

- If there are no log records for a transaction after a crash then abort transaction and "forget" it

- Retry mechanism
  - In prepared state, periodically query coordinator
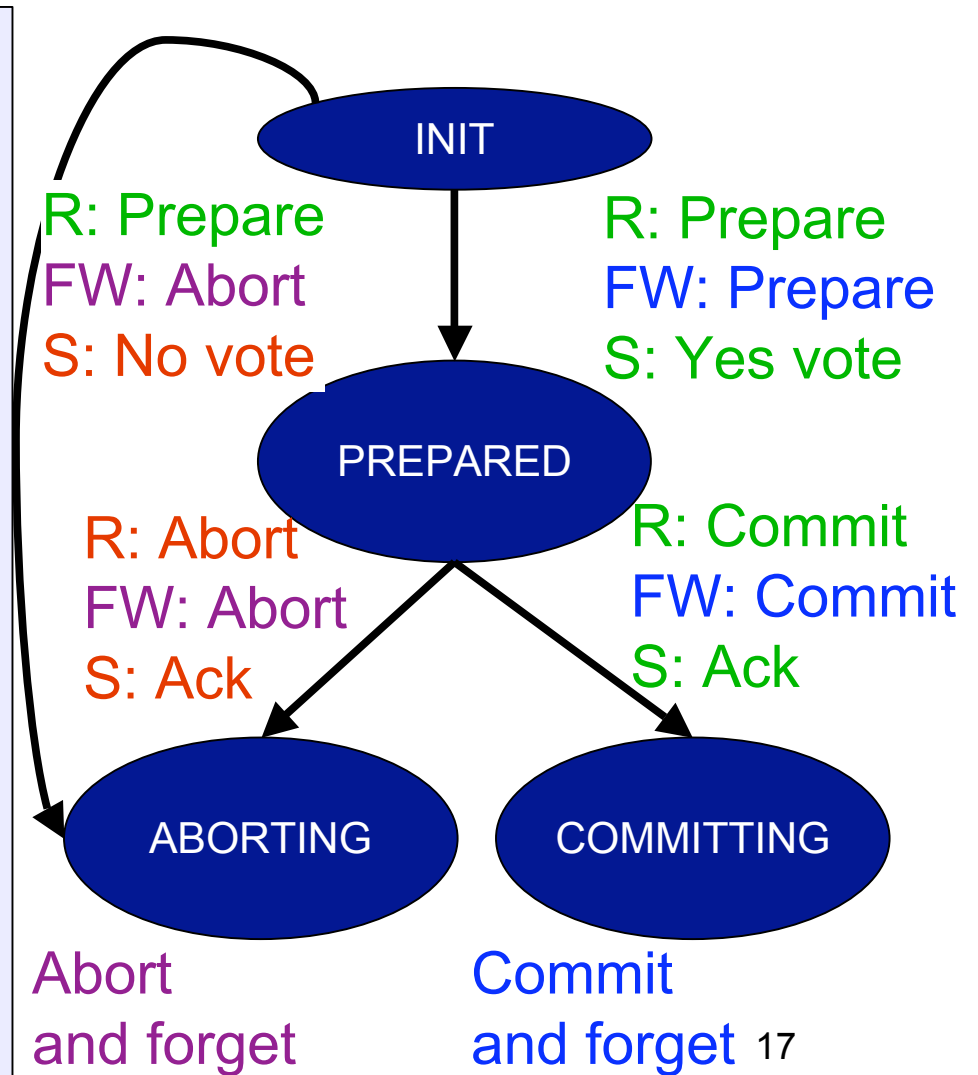 - In committing/aborting state, periodically resend messages to subordinates
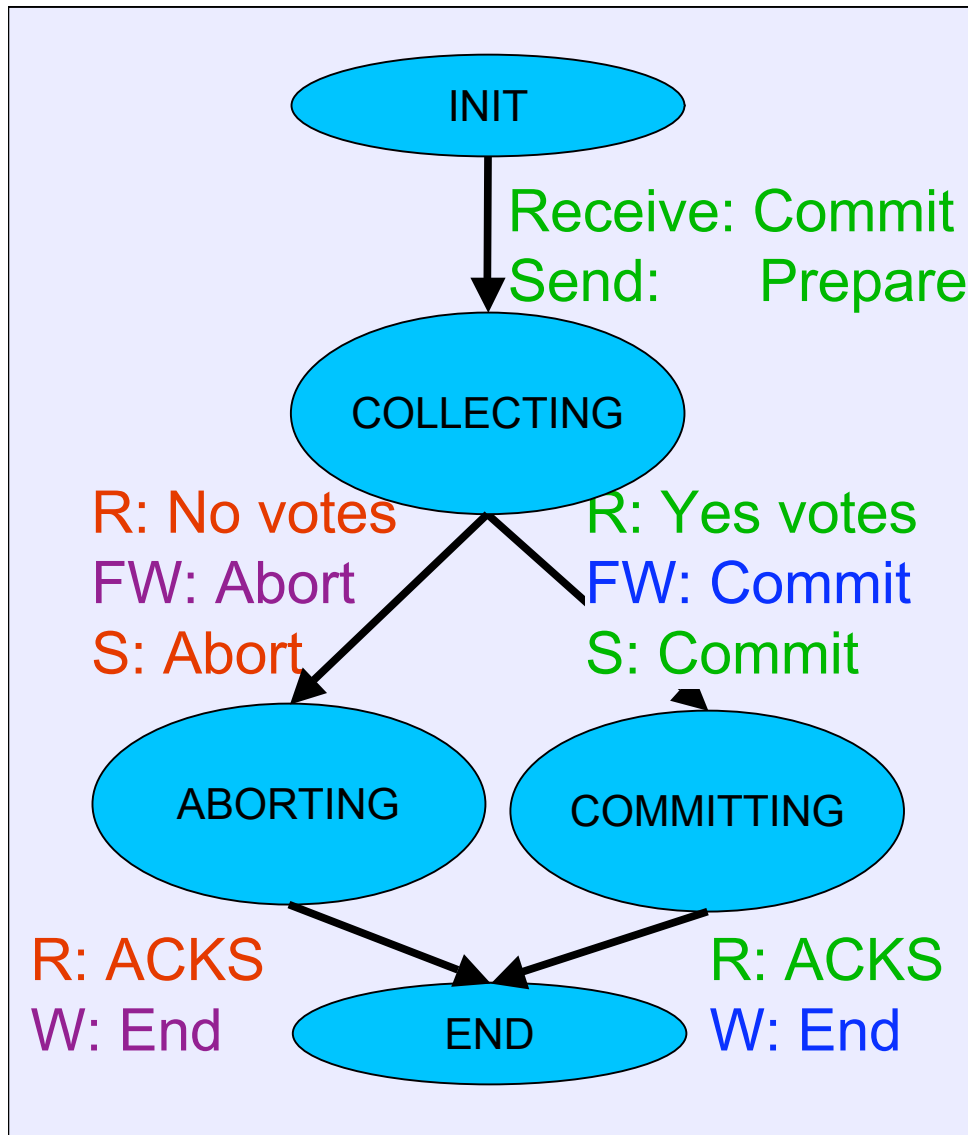
Observations

- Coordinator keeps transaction in transactions table until it receives all acks
  - To ensure all subordinates know whether to commit or abort
  - So acks enable coordinator to "forget" about transaction

- After crash, if recovery process finds no log records for a transaction, the transaction is presumed to have aborted

- Read-only subtransactions: no changes to undo or redo

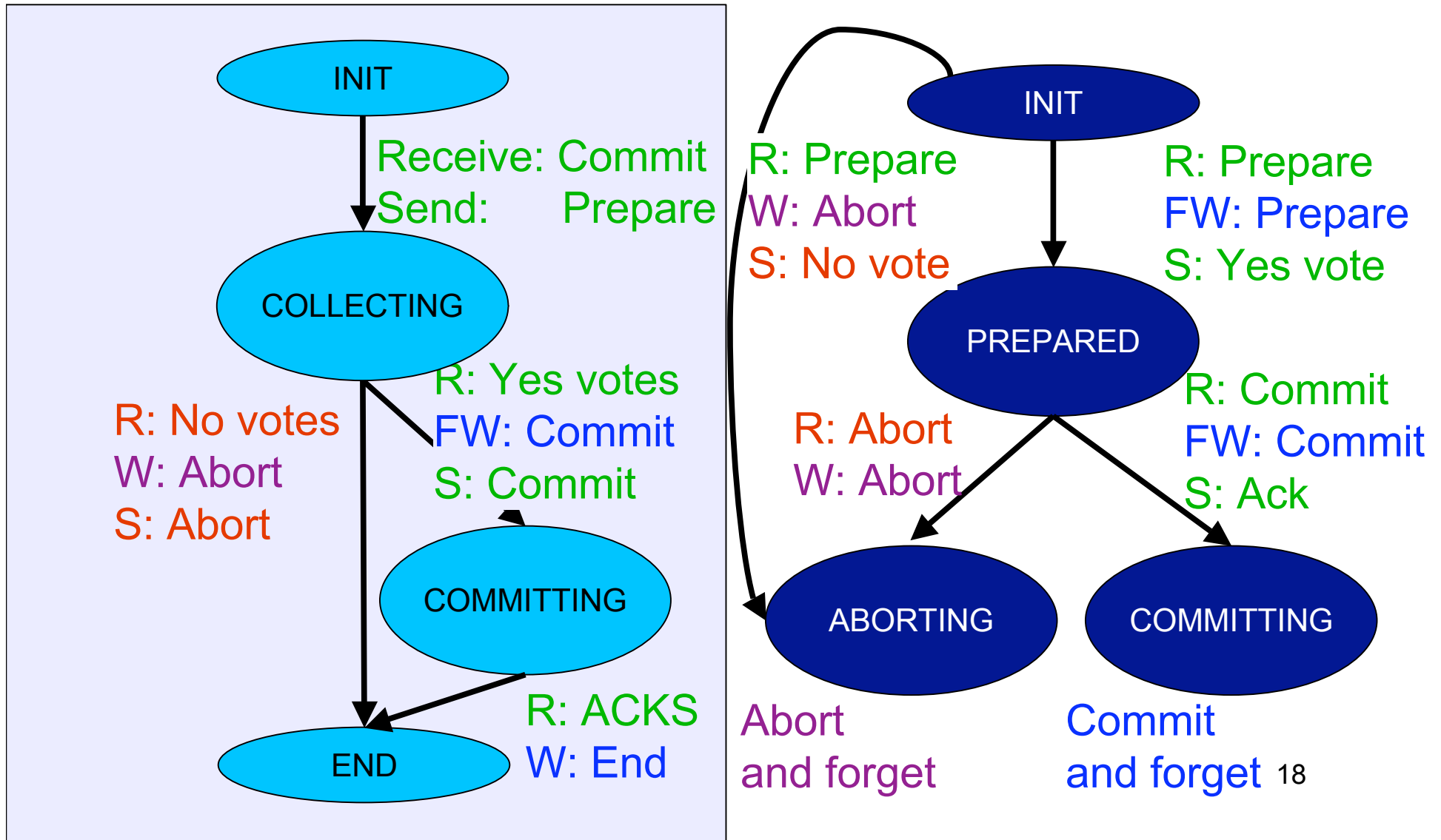# Presumed Abort Protocol

- Optimization goals
  - Fewer messages and fewer force-writes

- Principle
  - If nothing known about a transaction, assume ABORT
- Aborting transactions need no force-writing

- Avoid log records for read-only transactions
  - Reply with a READ vote instead of YES vote
- Optimizes read-only transactions

2PC State Machines (repeat)

INIT

Receive: Commit
Send: Prepare

COLLECTING

R: No votes
FW: Abort
S: Abort

R: Yes votes
FW: Commit
S: Commit

ABORTING

COMMITTING

R: ACKS
W: End

R: ACKS
W: End

END

INIT

R: Prepare
FW: Abort
S: No vote

R: Prepare
FW: Prepare
S: Yes vote

PREPARED

R: Abort
FW: Abort
S: Ack

R: Commit
FW: Commit
S: Ack

ABORTING

COMMITTING

Abort
and forget

Commit
and forget   17

Presumed Abort State Machines

INIT

Receive: Commit
Send: Prepare

COLLECTING

R: No votes
W: Abort
S: Abort

R: Yes votes
FW: Commit
S: Commit

COMMITTING

END

R: ACKS
W: End

R: Prepare
W: Abort
S: No vote

R: Abort
W: Abort

INIT

R: Prepare
FW: Prepare
S: Yes vote

PREPARED

R: Commit
FW: Commit
S: Ack

ABORTING

Abort
and forget

COMMITTING

Commit
and forget

18

Presumed Abort for Read-Only

INIT

Receive: Commit
Send: Prepare

COLLECTING

R: Read
Forget

END

INIT

R: Prepare
S: Read vote

DONE

Forget

19