

CA420 Databases II

Distributed Recovery

Atomic Commitment

Atomicity:

- transactions are all or nothing:
 - either they commit and their effects become permanent,
 - or they abort and their effects are removed from the database

Distributed atomicity:

- each transaction must either commit at each site, or abort at each site

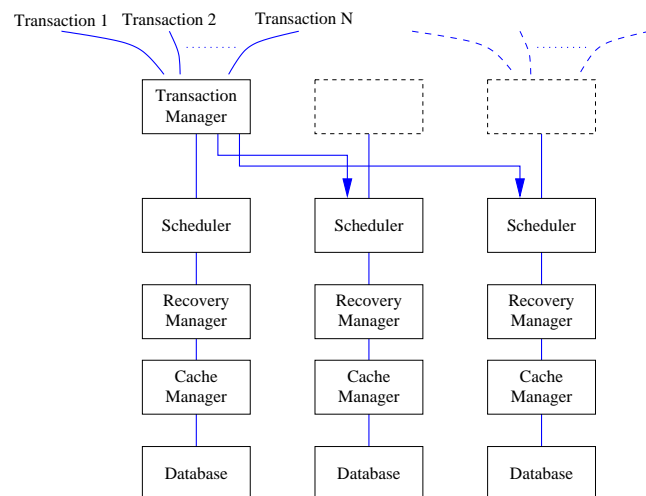
Atomic commitment protocol:

- a protocol to achieve distributed atomicity

Problems:

- lost messages
- system failures

Distributed Recovery



System Model Assumptions

System model:

- no replication of data items
only a single copy of each data item
- reads and writes sent to site containing the relevant data item
- commits and aborts sent to *all* sites participating in the transaction

Failure Modes

Failure modes:

- site failures:
 - partial failures:
 - operational sites may be uncertain about the status of remote sites
 - total failures
- communication failures:
 - lost messages
 - network partition

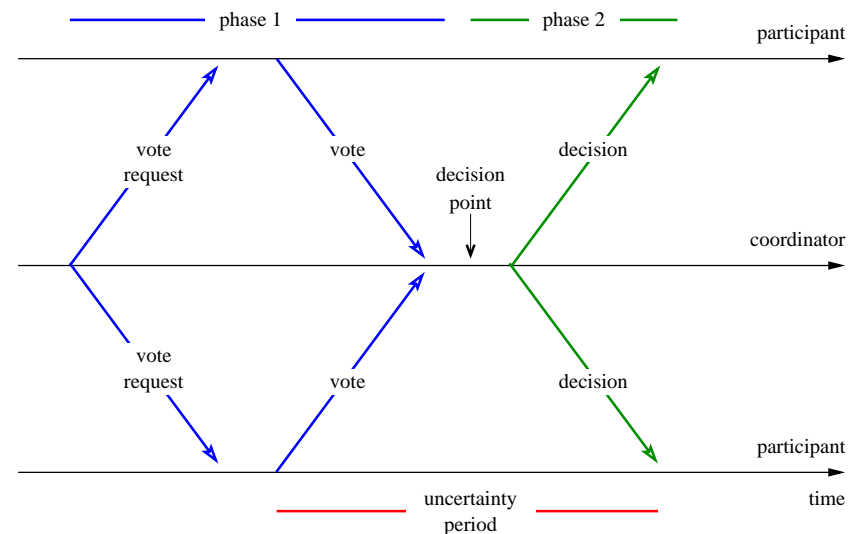
Operational systems may be unable tell the difference between failure types

Atomic Commitment Protocols (ACP)

An *atomic commitment protocols* is a voting protocol such that:

1. the decision is either *commit* or *abort*
2. all sites that reach a decision reach the *same* decision
3. a site cannot reverse its decision once it has reached one
4. the *commit* decision may only be reached if all participant vote to commit
5. if there are no failures and all sites vote to commit:
 - then the decision will be *commit*
6. if all failures are repaired and no new failures occur for sufficiently long:
 - then all sites will eventually reach a decision

Two-Phase Commit (2PC)



The 2PC Protocol

Each transaction is assigned a coordinator

When coordinator receives commit request:

1. coordinator sends *Vote-Req* message to all participants
2. when participant receives *Vote-Req* it:
 - respond with a message to the coordinator indicating *yes* or *no*
 - if *no*: then safe to abort local part of transaction unilaterally
3. if all votes are *yes* and coordinator's vote is also *yes*:
 - coordinator decides *Commit* and sends *Commit* message to all participants
 otherwise:
 - coordinator decides *Abort* and sends *Abort* message to participants that voted *yes*
4. each participant that voted *yes* waits for *Commit* or *Abort* message, and decides accordingly

2PC and Locking

Generally:

write locks must be held until commit or abort processing is complete

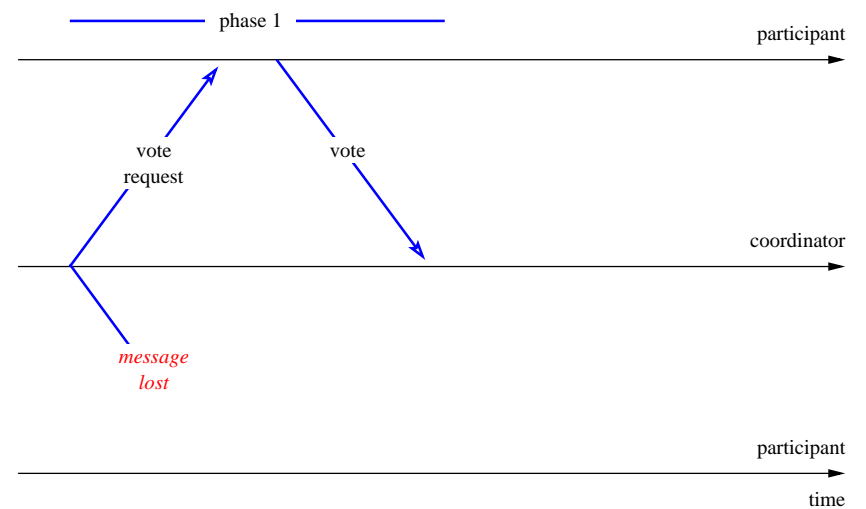
If site votes no:

- unilaterally Abort
- then release write locks

If site votes yes:

- write locks must be held until decision is known
- so other transactions may be blocked

Vote-Req Message Lost

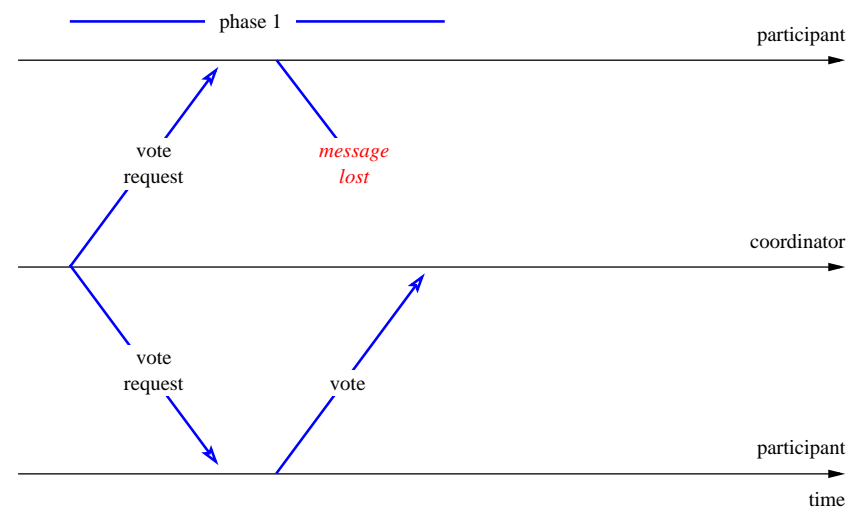


2PC – Timeout Actions

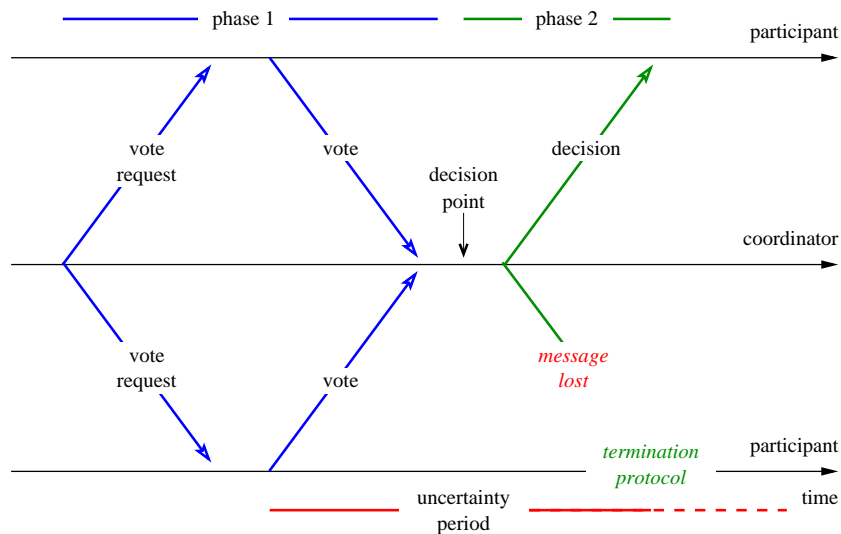
Timeout actions:

- when participant waiting for vote request:
 - participant has not yet voted
 - on timeout:
 - vote no and assume Abort
- when coordinator waiting for votes:
 - coordinator has not yet reached decision
 - on timeout:
 - vote no and assume Abort
 - send Abort messages to participants
- when participant waiting for decision:
 - participant is *uncertain*:
 - * cannot unilaterally commit
 - * cannot unilaterally abort

Vote Message Lost



Decision Message Lost



2PC – Cooperative Termination Protocol

Cooperative termination protocol:

- p is the initiator of the termination protocol
 q the responder
- protocol:
 1. if q has already decided:
 q sends this decision to p , and p decides accordingly
 2. if q has not yet voted
 q unilaterally decides Abort and informs p accordingly
 3. q has voted yes but has not yet reached decision
 q cannot help p

If p can communicate with some q for which either (1) or (2) above hold:
then p can learn the global outcome

This protocol reduces, *but does not eliminate*, the probability of blocking

2PC – Termination Protocols

Termination protocols:

- simplest
remain blocked until re-establish communication with coordinator:
 - participant may be blocked unnecessarily:
 - * there may be other participants who know the outcome, even if the participant at hand does not
- alternative
cooperative termination protocol:
 - assume coordinator includes a list of participants in the Vote-Req message
 - participants may be able learn global outcome from one-another

2PC – Logging

Logging:

1. when coordinator sends Vote-Req message:
write start-2PC record to the log
2. participant voting:
 - yes:**
write and flush “Yes” record to log *before* voting
 - no:**
write “No” record to log
3. coordinator decision:
 - commit:**
write and flush Commit record to log *before* informing participants
 - abort:**
write Abort record to log
4. participant receives Commit (or Abort) message:
write Commit (or Abort) record to log

2PC – Restart – Coordinator

If log contains a `start-2PC` record:

then site was/is coordinator

If log also contains `Commit` or `Abort` record:

then decision was made before failure

If neither is found:

then coordinator can unilaterally decide to `Abort`

Correctness depends upon:

writing commit decision to log *before* informing participants

2PC – Restart – Participant

If log does *not* contain a `start-2PC` record:

then site was/is a participant

If log also contains `Commit` or `Abort` record:

then decision was made before failure

If log does not contain a “Yes” record (or contains a “No” record):

participant can unilaterally abort

If log contains “Yes” record but no `Commit` record:

termination protocol must be executed

may end up blocked

At what point can the TM begin accepting new transactions?
