ICS 421 Spring 2010

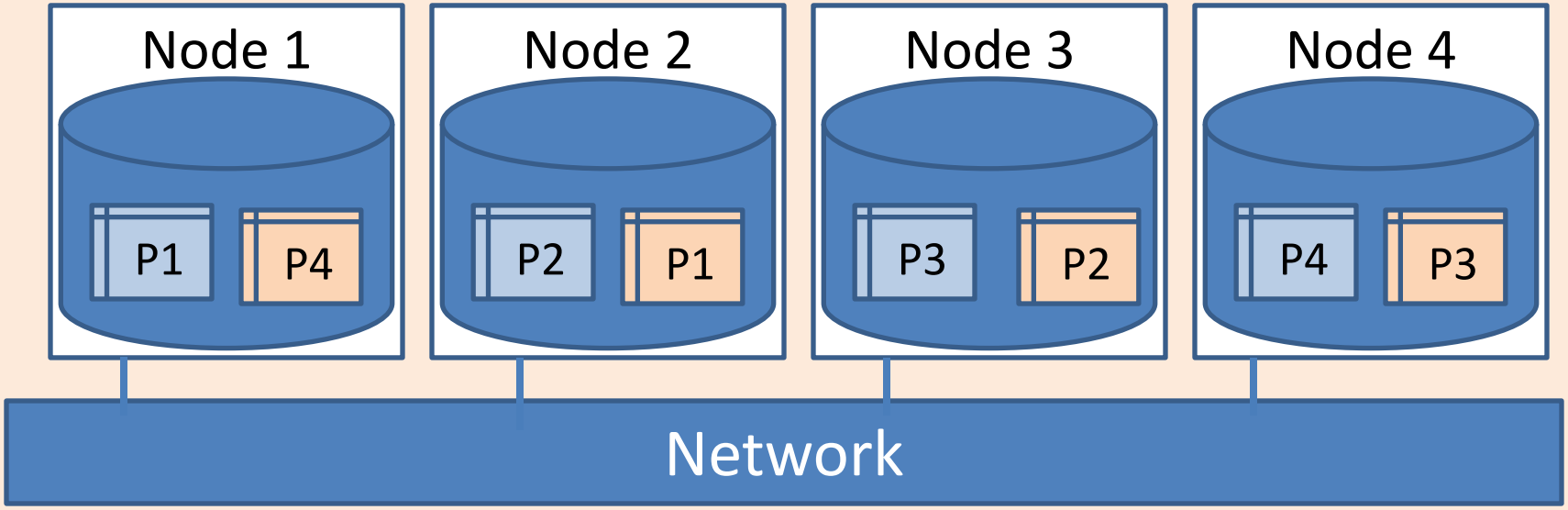
Distributed Transactions

Asst. Prof. Lipyeow Lim

Information & Computer Science Department

University of Hawaii at Manoa

Isolation: what is different ?



Consider shared nothing parallel/distributed DB

- Node 1 receives transaction that updates relation P
- Node 3 receives transaction that updates P
- Node 4 receives transactions that reads P

How do we ensure isolation ?

Distributed Locking

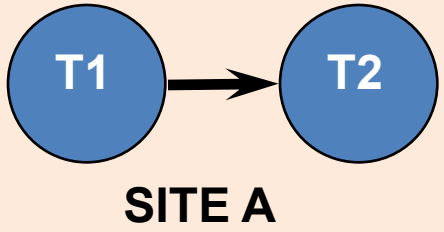
How do we manage locks for objects across many sites?

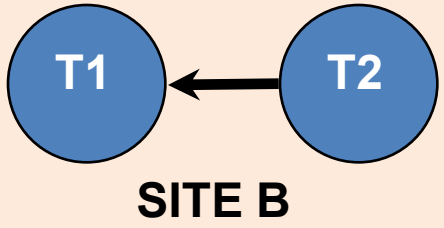
- Centralized: One site does all locking.
  - Vulnerable to single site failure.
- Primary Copy: All locking for an object done at the primary copy site for this object.
  - Reading requires access to locking site as well as site where the object is stored.
- Fully Distributed: Locking for a copy done at site where the copy is stored.
  - Locks at all sites while writing an object.

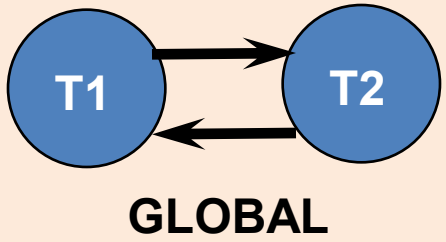
What about deadlocks?

Distributed Deadlock Detection

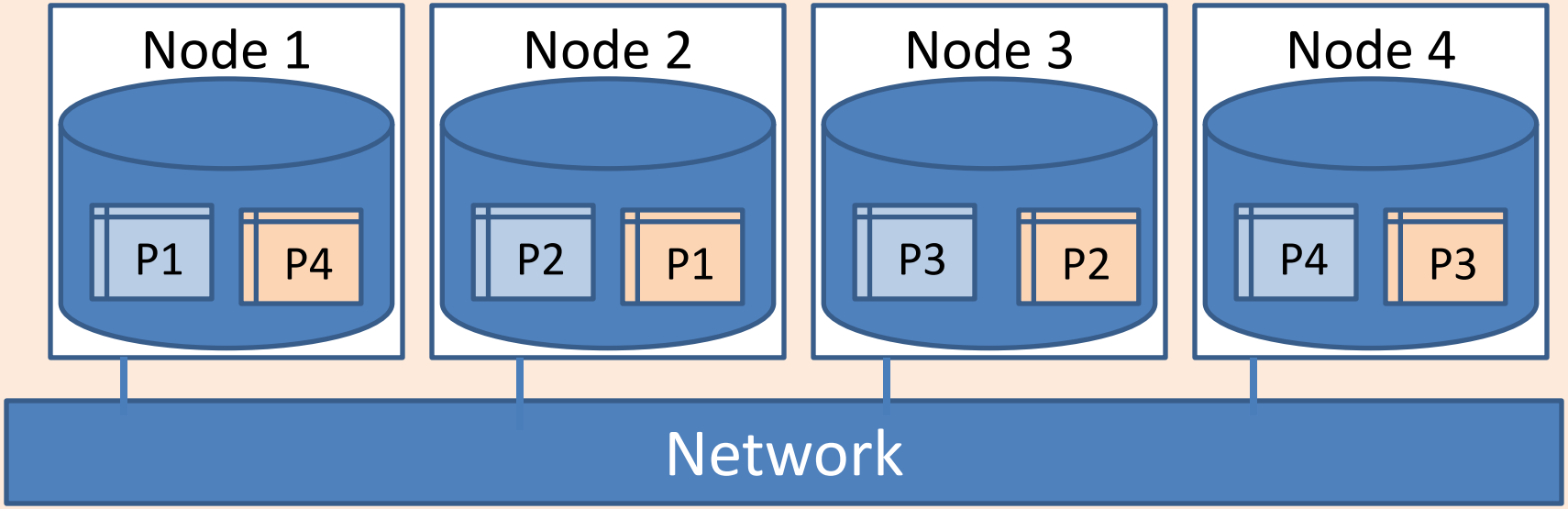
- Each site maintains a local waits-for graph.
- A global deadlock might exist even if the local graphs contain no cycles
- Three solutions:
  - Centralized (send all local graphs to one site);
  - Hierarchical (organize sites into a hierarchy and send local graphs to parent in the hierarchy);
  - Timeout (abort Xact if it waits too long).







Atomicity & Durability: what is different ?



Consider shared nothing parallel/distributed DB

- Node 1 receives transaction that updates relation P
- Node 3 receives transaction that updates P
- Node 4 receives transactions that reads P

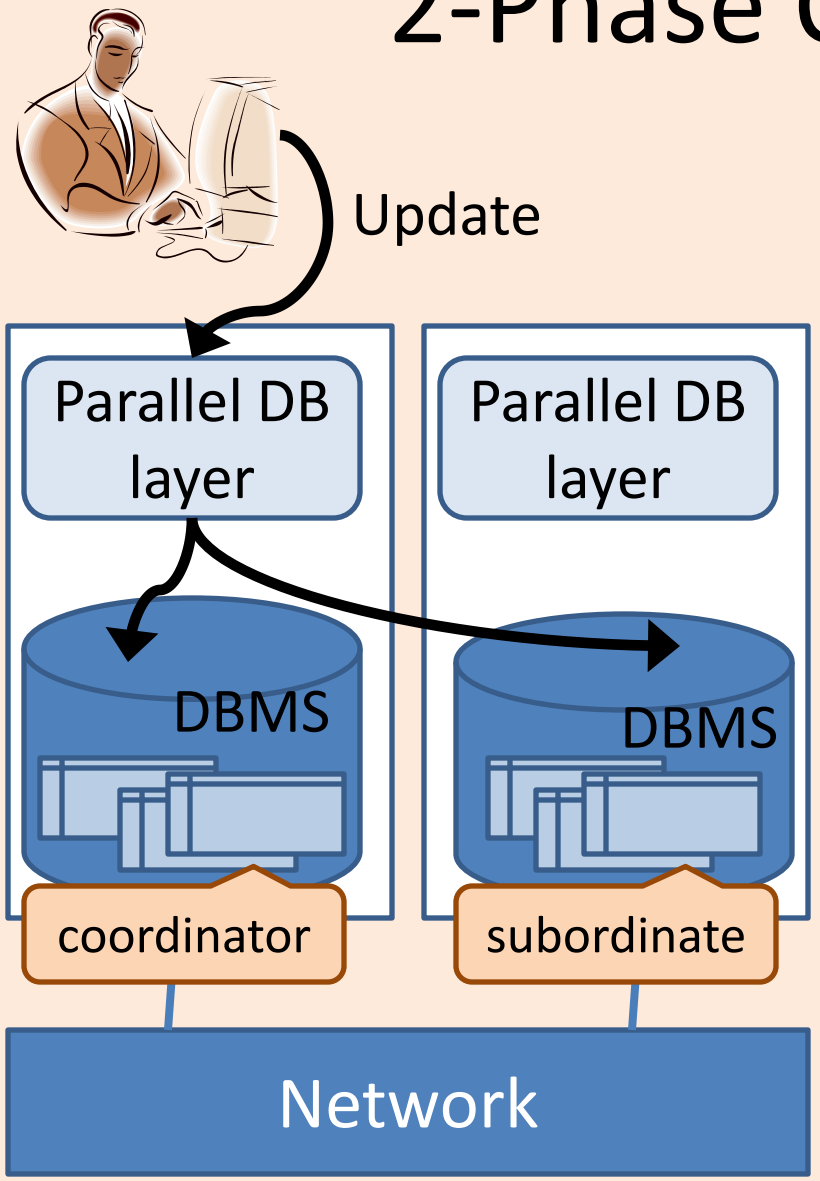
What if one node crashes ?

What if network does down ?

Distributed Recovery

- Two new issues:
  - New kinds of failure, e.g., links and remote sites.
  - If "sub-transactions" of an Xact execute at different sites, all or none must commit. Need a commit protocol to achieve this!
- A log is maintained at each site, as in a centralized DBMS, and commit protocol actions are additionally logged.

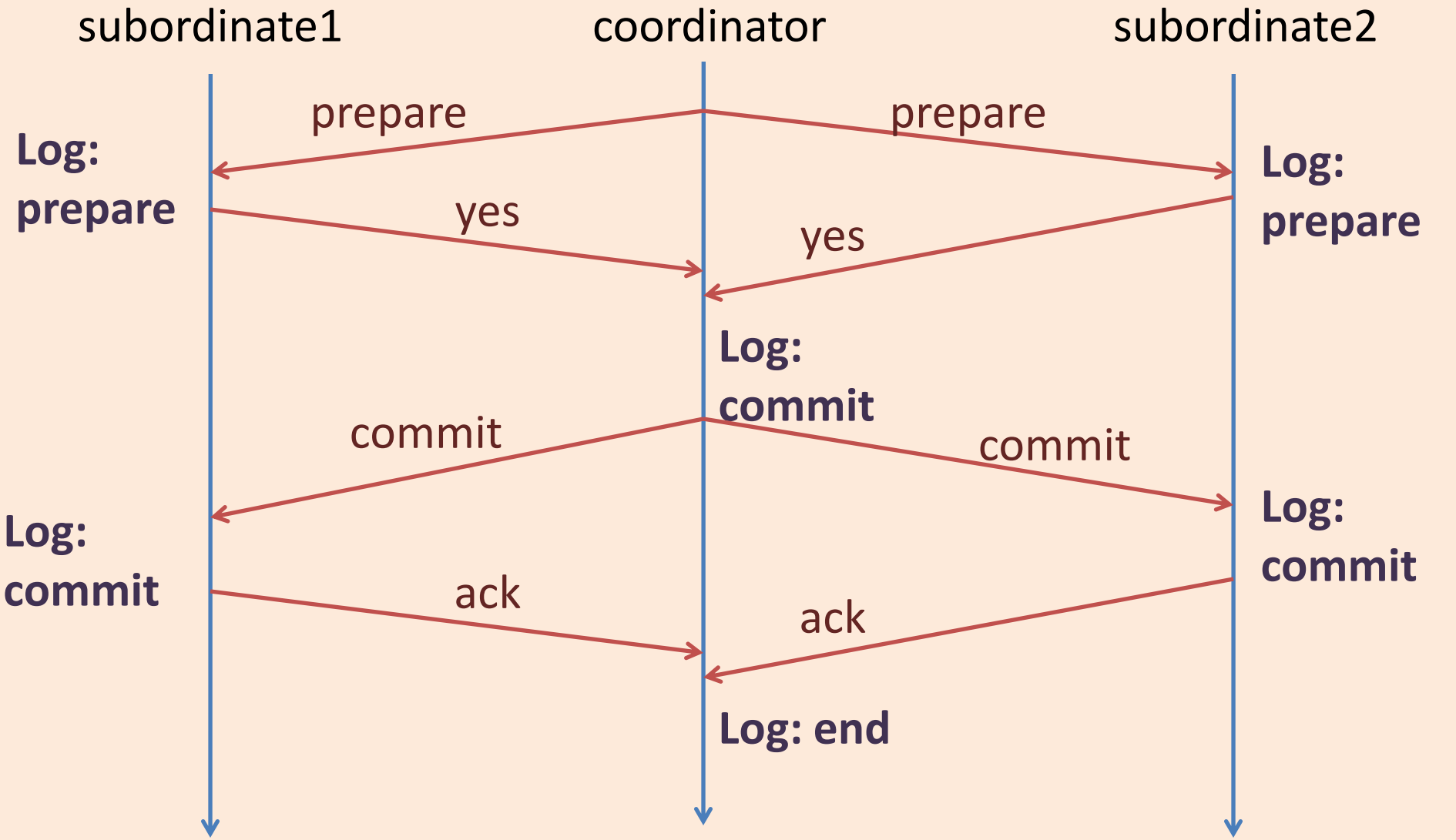
2-Phase Commit (2PC)



When an Xact wants to commit:

- Coordinator sends **prepare** msg to each subordinate.
- Subordinate force-writes an **abort** or **prepare** log record and then sends a **no** or **yes** msg to coordinator.
- If coordinator gets unanimous yes votes, force-writes a **commit** log record and sends **commit** msg to all subs. Else, force-writes **abort** log rec, and sends **abort** msg.
- Subordinates force-write **abort/commit** log rec based on msg they get, then send **ack** msg to coordinator.
- Coordinator writes **end** log rec after getting all acks.

Example: 2PC (commit success)



Comments on 2PC

- Two rounds of communication: first, voting; then, termination. Both initiated by coordinator.
- Any site can decide to abort an Xact.
- Every msg reflects a decision by the sender; to ensure that this decision survives failures, it is first recorded in the local log.
- All commit protocol log recs for an Xact contain Xactid and Coordinatorid. The coordinator's abort/commit record also includes ids of all subordinates.

Restart after a failure at a site

- If we have a commit or abort log rec for Xact T, but not an end rec, must redo/undo T.
  - If this site is the coordinator for T, keep sending **commit/abort** msgs to subs until **acks** received.
- If we have a prepare log rec for Xact T, but not commit/abort, this site is a subordinate for T.
  - Repeatedly contact the coordinator to find status of T, then write **commit/abort** log rec; redo/undo T; and write **end** log rec.
- If we don't have even a prepare log rec for T, unilaterally abort and undo T.
  - This site may be coordinator! If so, subs may send msgs.

Coordinator Failures: Blocking

- If coordinator for Xact T fails, subordinates who have voted yes cannot decide whether to commit or abort T until coordinator recovers.
  - T is blocked.
  - Even if all subordinates know each other (extra overhead in prepare msg) they are blocked unless one of them voted no.

Link and Remote Site Failures

- If a remote site does not respond during the commit protocol for Xact T, either because the site failed or the link failed:
  - If the current site is the coordinator for T, should abort T.
  - If the current site is a subordinate, and has not yet voted yes, it should abort T.
  - If the current site is a subordinate and has voted yes, it is blocked until the coordinator responds.

Observations on 2PC

- Ack msgs used to let coordinator know when it can "forget" an Xact; until it receives all acks, it must keep T in the Xact Table.
- If coordinator fails after sending prepare msgs but before writing commit/abort log recs, when it comes back up it aborts the Xact.
- If a subtransaction does no updates, its commit or abort status is irrelevant.

2PC with Presumed Abort

- When coordinator aborts T, it undoes T and removes it from the Xact Table immediately.
  - Doesn't wait for **acks**; "presumes abort" if Xact not in Xact Table. Names of subs not recorded in **abort** log rec.
- Subordinates do not send acks on abort.
- If subxact does not do updates, it responds to prepare msg with reader instead of yes/no.
- Coordinator subsequently ignores readers.
- If all subxacts are readers, 2nd phase not needed.