Commit Protocols

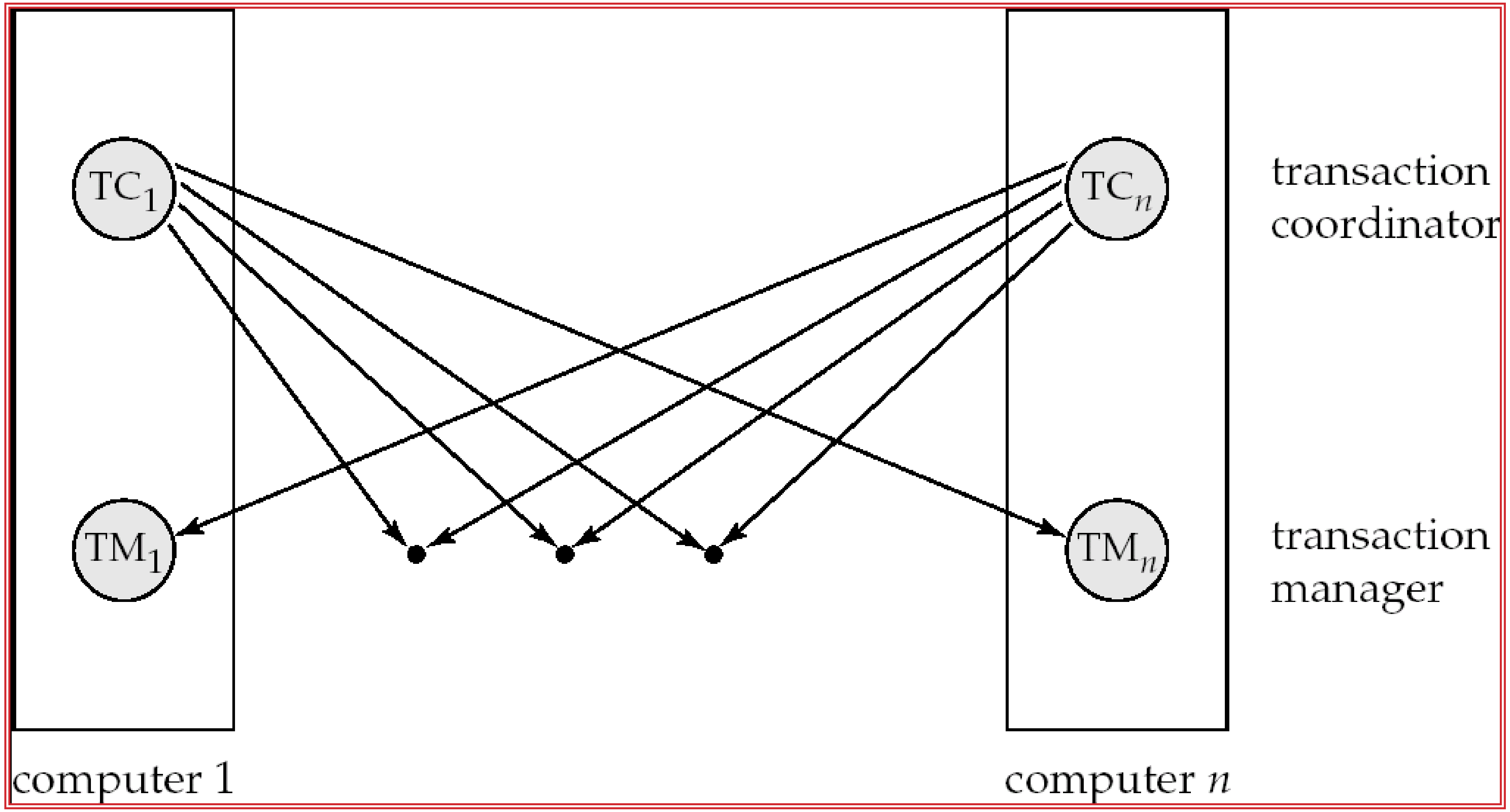
CS60002: Distributed Systems

Pallab Dasgupta
Professor,
Dept. of Computer Sc. & Engg.,
Indian Institute of Technology Kharagpur

Distributed Transactions

- Transaction may access data at several sites.
- Each site has a local transaction manager responsible for:
  - Maintaining a log for recovery purposes
  - Participating in coordinating the concurrent execution of the transactions executing at that site.
- Each site has a transaction coordinator, which is responsible for:
  - Starting the execution of transactions that originate at the site.
  - Distributing subtransactions at appropriate sites for execution.
  - Coordinating the termination of each transaction that originates at the site, which may result in the transaction being committed at all sites or aborted at all sites.

Transaction System Architecture



System Failure Modes

- Failures unique to distributed systems:
  - Failure of a site.
  - Loss of massages
    - Handled by network transmission control protocols such as TCP-IP
  - Failure of a communication link
  - Handled by network protocols, by routing messages via alternative links
  - Network partition
    - A network is said to be partitioned when it has been split into two or more subsystems that lack any connection between them
      - Note: a subsystem may consist of a single node
- Network partitioning and site failures are generally indistinguishable.

Commit Protocols

- Commit protocols are used to ensure atomicity across sites
  - a transaction which executes at multiple sites must either be committed at all the sites, or aborted at all the sites.
  - not acceptable to have a transaction committed at one site and aborted at another
- The *two-phase commit* (2PC) protocol is widely used
- The *three-phase commit* (3PC) protocol is more complicated and more expensive, but avoids some drawbacks of two-phase commit protocol. This protocol is not used in practice.

Two Phase Commit Protocol (2PC)

- Assumes fail-stop model – failed sites simply stop working, and do not cause any other harm, such as sending incorrect messages to other sites.
- Execution of the protocol is initiated by the coordinator after the last step of the transaction has been reached.
- The protocol involves all the local sites at which the transaction executed
- Let *T* be a transaction initiated at site S_i, and let the transaction coordinator at S_i be C_i

Phase 1: Obtaining a Decision

- Coordinator asks all participants to *prepare* to commit transaction T_i.
  - C_i adds the records <prepare *T*> to the log and forces log to stable storage
  - sends prepare *T* messages to all sites at which *T* executed
- Upon receiving message, transaction manager at site determines if it can commit the transaction
  - if not, add a record <no *T*> to the log and send abort *T* message to C_i
  - if the transaction can be committed, then:
  - add the record <ready *T*> to the log
  - force *all records* for *T* to stable storage
  - send ready *T* message to C_i

Phase 2: Recording the Decision

- *T* can be committed of C_i received a ready *T* message from all the participating sites: otherwise *T* must be aborted.

- Coordinator adds a decision record, <commit *T*> or <abort *T*>, to the log and forces record onto stable storage. Once the record stable storage it is irrevocable (even if failures occur)

- Coordinator sends a message to each participant informing it of the decision (commit or abort)

- Participants take appropriate action locally.

Handling of Failures - Site Failure

When site S_i recovers, it examines its log to determine the fate of transactions active at the time of the failure.

- **Log contain <**commit *T***> record: site executes** redo **(*T*)**
- **Log contains <**abort *T***> record: site executes** undo **(*T*)**
- **Log contains <**ready *T***> record: site must consult C_i to determine the fate of *T*.**
  - **If *T* committed, redo (*T*)**
  - **If *T* aborted, undo (*T*)**
- **The log contains no control records concerning *T* replies that S_k failed before responding to the** prepare ***T* message from C_i**
  - **since the failure of S_k precludes the sending of such a response C_1 must abort *T***
  - **S_k must execute** undo **(*T*)**

Handling of Failures- Coordinator Failure

- **If coordinator fails while the commit protocol for *T* is executing then participating sites must decide on *T*'s fate:**
  1. **If an active site contains a <**commit ***T*>** **record in its log, then *T* must be committed.**
  2. **If an active site contains an <**abort ***T*>** **record in its log, then *T* must be aborted.**
  3. **If some active participating site does not contain a <**ready ***T*>** **record in its log, then the failed coordinator C_i cannot have decided to commit *T*. Can therefore abort *T*.**
  4. **If none of the above cases holds, then all active sites must have a <**ready ***T*>** **record in their logs, but no additional control records (such as <**abort ***T*>** **of <**commit ***T*>**). In this case active sites must wait for C_i to recover, to find decision.**
- **Blocking problem : active sites may have to wait for failed coordinator to recover.**

Handling of Failures - Network Partition

- If the coordinator and all its participants remain in one partition, the failure has no effect on the commit protocol.
- If the coordinator and its participants belong to several partitions:
  - Sites that are not in the partition containing the coordinator think the coordinator has failed, and execute the protocol to deal with failure of the coordinator.
    - No harm results, but sites may still have to wait for decision from coordinator.
- The coordinator and the sites are in the same partition as the coordinator think that the sites in the other partition have failed, and follow the usual commit protocol.
    - Again, no harm results

Recovery and Concurrency Control

- **In-doubt transactions have a <ready *T*>, but neither a <commit *T*>, nor an <abort *T*> log record.**
- **The recovering site must determine the commit-abort status of such transactions by contacting other sites; this can slow and potentially block recovery.**
- **Recovery algorithms can note lock information in the log.**
  - **Instead of <ready *T*>, write out <ready *T*, *L*> *L* = list of locks held by *T* when the log is written (read locks can be omitted).**
  - **For every in-doubt transaction *T*, all the locks noted in the <ready *T*, *L*> log record are reacquired.**
- **After lock reacquisition, transaction processing can resume; the commit or rollback of in-doubt transactions is performed concurrently with the execution of new transactions.**

Three Phase Commit (3PC)

- **Assumptions:**
  - No network partitioning
  - At any point, at least one site must be up.
  - At most K sites (participants as well as coordinator) can fail

- **Phase 1: Obtaining Preliminary Decision: Identical to 2PC Phase 1.**
  - Every site is ready to commit if instructed to do so

Three Phase Commit (3PC)

- Phase 2 of 2PC is split into 2 phases, Phase 2 and Phase 3 of 3PC
  - In phase 2 coordinator makes a decision as in 2PC (called the pre-commit decision) and records it in multiple (at least K) sites
  - In phase 3, coordinator sends commit/abort message to all participating sites,
- Under 3PC, knowledge of pre-commit decision can be used to commit despite coordinator failure
  - Avoids blocking problem as long as < K sites fail
- Drawbacks:
  - higher overheads
  - assumptions may not be satisfied in practice