Atomicity

- Either all the operations associated with a program unit are executed to completion, or none are performed

- Ensuring **atomicity** in a distributed system requires a **transaction coordinator**, which is responsible for the following:
  - Starting the execution of the transaction
  - Breaking the transaction into a number of subtransactions, and distribution these subtransactions to the appropriate sites for execution
  - Coordinating the termination of the transaction, which may result in the transaction being committed at all sites or aborted at all sites

Two-Phase Commit Protocol (2PC)

- Assumes fail-stop model
- Execution of the protocol is initiated by the coordinator after the last step of the transaction has been reached
- When the protocol is initiated, the transaction may still be executing at some of the local sites
- The protocol involves all the local sites at which the transaction executed
- Example: 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

- C_i adds <prepare *T*> record to the log
- C_i sends <prepare *T*> message to all sites
- When a site receives a <prepare *T*> message, the transaction manager determines if it can commit the transaction
  - If no: add <no *T*> record to the log and respond to C_i with <abort *T*>
  - If yes:
    - add <ready *T*> record to the log
    - force *all log records* for *T* onto stable storage
    - send <ready *T*> message to C_i

Phase 1 (Cont.)

- Coordinator collects responses
  - All respond “ready”, decision is *commit*
  - At least one response is “abort”, decision is *abort*
  - At least one participant fails to respond within time out period, decision is *abort*

Phase 2: Recording Decision in the Database

- Coordinator adds a decision record

  <abort *T*> or <commit *T*>

  to its log and forces record onto stable storage
- Once that record reaches 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

Failure Handling in 2PC – Site Failure

- The log contains a <commit T> record
  - In this case, the site executes **redo**(T)
- The log contains an <abort T> record
  - In this case, the site executes **undo**(T)
- The contains a <ready T> record; consult C_i
  - If C_i is down, site sends **query-status** T message to the other sites
- The log contains no control records concerning T
  - In this case, the site executes **undo**(T)

Failure Handling in 2PC – Coordinator C_i Failure

- If an active site contains a <commit T> record in its log, the T must be committed
- If an active site contains an <abort T> record in its log, then T must be aborted
- If some active site does *not* contain the record <ready T> in its log then the failed coordinator C_i cannot have decided to commit T
  - Rather than wait for C_i to recover, it is preferable to abort T
- All active sites have a <ready T> record in their logs, but no additional control records
  - In this case we must wait for the coordinator to recover
  - Blocking problem – T is blocked pending the recovery of site S_i

Concurrency Control

- Modify the centralized concurrency schemes to accommodate the distribution of transactions
- Transaction manager coordinates execution of transactions (or subtransactions) that access data at local sites
- Local transaction only executes at that site
- Global transaction executes at several sites