Distributed Commit

- **Goal**: Either all members of a group decide to perform an operation, or none of them perform the operation.

Assumptions

- **Failures**: 
  - Crash failures that can be recovered
  - Communication failures detectable by timeouts

- **Notes**: 
  - Commit requires a set of processes to agree...
  - …similar to the Byzantine general problem…
  - …but the solution much simpler because stronger assumptions

Two Phase Commit (2PC)

2PC State Machine

- **2PC: Crash Recovery Protocol**
  - *Stable storage* is persistent memory that supports writes that are atomic with respect to failures
  - Log actions:
    - c sends VOTE_REQ write start
    - p votes YES write yes
    - p votes NO write abort
    - c decides commit write commit
    - c decides abort write abort
    - p receives decision write decision
2PC: Crash Recovery Protocol

Upon recovery a process \( r \) starts reading the values logged to stable storage.
- If there is a \textit{start} then \( r \) was the coordinator:
  - If there is a subsequent \textit{abort} or \textit{commit} then decision was made; otherwise decide \textit{abort}.
- Otherwise, \( r \) was a participant:
  - If there is \textit{abort} or \textit{commit} then the decision was made;
  - If there is no \textit{yes} then decide \textit{abort}.
  - Otherwise (i.e., there is an \textit{yes} record) run termination protocol.

... when can these records be garbage collected?

Example: Reliable Communication

- Backward recovery: retransmit packet if lost
- Forward recovery: use erasure coding
  - Instead of sending \( k \) packets, send \( n > k \) using erasure coding
  - As long as receiver gets at least \( k \) packets out of \( n \), it can reconstruct the original \( k \) packets

Recovery Techniques: Checkpoints

- Goal: recover a process from error
- Backward recovery: checkpoint the state of the process periodically
  - Go to previous checkpoint, if error
  - Problem: same failure may repeat
- Forward recovery: go to a known good state if error
  - Problem: need to know in advance which error may occur

Recovery Techniques: Message Logging

- Sender based: sender logs message before sending it out
- Receiver based: receiver logs message before delivering it
- Replay log messages between checkpoints \( \rightarrow \) restore state beyond most recent checkpoint

Distributed Checkpointing: Recovery Line

- Recovery line: most recent snapshot
  - If a process \( P \) has recorded the receipt of message \( m \) there should be a process \( Q \) that recorded sending of message \( m \)
- How do you find a recovery line?

Independent Checkpointing: The Domino Effect

- Domino effect: cascaded rollback to find a recovery line
- Solutions:
  - Coordinate checkpointing: use two-phase non-blocking protocol (see the book)
  - Logging and replaying messages
Message Logging and Checkpointing

- Incorrect replay of messages after recovery, leading to an orphan process

Stable Storage

- Storage designed to survive anything except major calamities

- Use two disks to record identical information
  1) Write and verify sector on disk 1
  2) Write and verify sector on disk 2

- Recovery
  - Verify all sectors
  - If two corresponding sectors differ, copy sector from disk 1 to disk

Stable Storage Recovery

- Sector has different value

a) Stable Storage
b) Crash after drive 1 is updated
c) Bad spot