Welcome to CSCI 7000-001 Lec 7 (Feb 4)!

Recap: Formalization of Two Phase Commit

1. ✓ Representation of system state / configuration.
2. ✓ Initial configuration
3. Transition relation.

→ Configuration of a system executing 2PC can be captured as a Java tuple: P

$\langle$ tmState, rmState, tmProposed, msgs $\rangle$

- tmState: current state of the TM
- rmState: current state of every Resource Manager (RM).
- tmProposed: set of processes that have proposed.
- msgs: set of messages in flight

→ Initially, $\langle$ tmState = Init,

rmState = $[\forall p \in P.\ p \rightarrow \text{Working}]$

tmProposed = $\{\}$,

msgs = $\{\}\rangle$

→ Transition relation identifies all ways in which a system configuration can be mapped to a new configuration

i.e., All ways in which system can take a step.

<tmState, rmState, tmProposed, msgs> → <tmState', rmState', ...>

→ What is the transition relation for 2PC?

= What are all the steps can a 2PC system take?

Ans:

1. Some RM proposes commit } ∃(rm ∈ R). RMProposes(rm)
2. Some RM aborts } ∃(rm ∈ R). RMAborts(rm)
3. TM receives commit proposal from an RM
4. TM decides to commit } TMCommits
5. TM decides to Abort
6. Some RM receives Commit message
7. Some RM receives Abort message

→ How would you define RMProposes(rm), TMCommits, ...

<tmState, rmState, tmProposed, msgs>

↓

<tmState', rmState', tmProposed', msgs>

∧ tmState' = tmState,

∧ rmState' = rmState [rm ↦ Proposed]

∧ rmState[rm] = Working

"Enabling predicate"

$\wedge$ tmProposed' = tmProposed

$\wedge$ msgs' = msgs $\cup$ {<<"propose", rm>>}

TmCommit

$\wedge$ tmState = init
$\wedge$ tmProposed = P
} Enabling Predicate

$\wedge$ tmState' = Committed

$\wedge$ rmState' = rmState

$\wedge$ tmProposed' = tmProposed

$\wedge$ msgs' = msgs $\cup$ {"Commit"}

PAXOS

→ An "algorithm" for Consensus in an async. dist. sys.

Consensus: All the participants have to agree to chose Exactly one of the given finite set of values.

→ FLP: Chose two of three:

Paxos?

→ Paxos fixes & generalizes 2PC

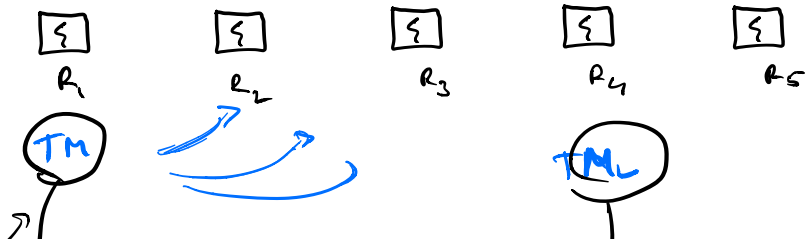
1. Transaction Manager (TM) is SPOF in 2PC
2. 2PC doesn't guarantee Validity

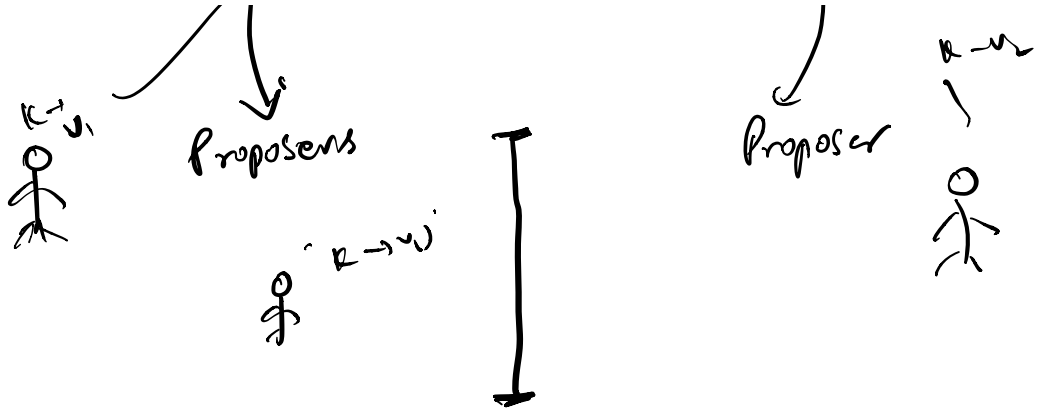
1. 2PC decides between Commit/Abort.

What if we want to decide on one among many unrelated values?

→ Let's start with generalization. Pick one among $\{v_1, v_2, \ldots, v_n\}$

$\{v_1 \ldots v_n\}$




Proposers
Proposer