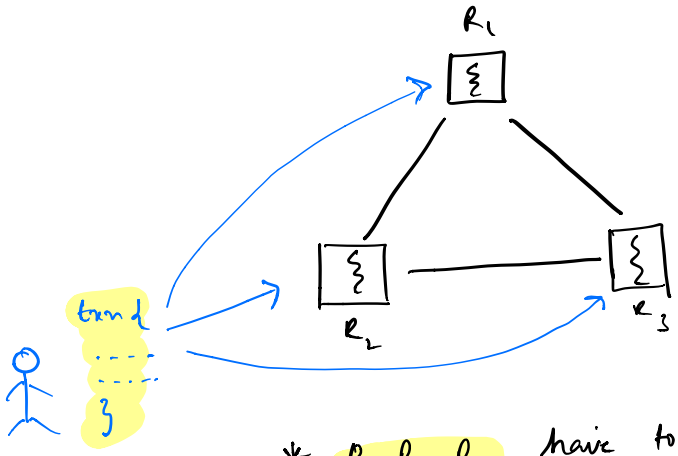
Welcome to CSCI 7000-001 Lec6 (Feb 2)!

Topics:

1. Two Phase Commit (2PC) Consensus Algorithm
2. TLA+ & PlusCal Specification Languages

Recall: Distributed txn commit problem



* R_1, R_2, R_3 have to agree on either committing the txn or aborting it.
* A subset of nodes cannot decide to commit while other subset aborts

⇒ Distributed transaction commit is an instance of the consensus problem.

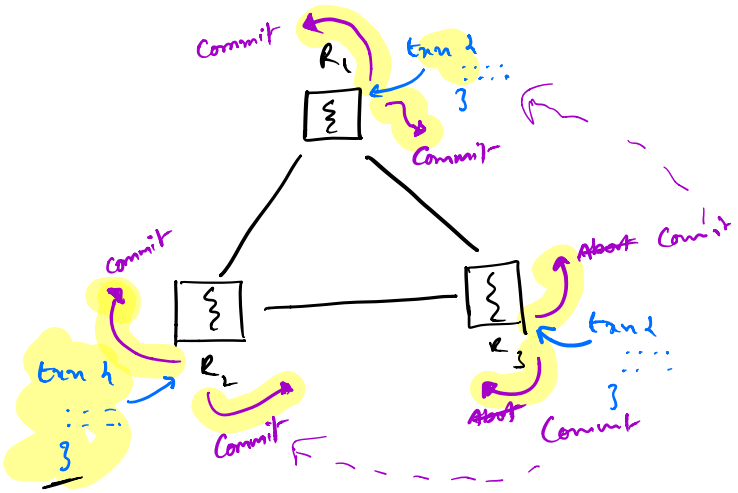
Qn: What does FLP Theorem say about distributed consensus?

→ Two Phase Commit (2PC) is an algorithm to achieve Consensus about a txn's fate (i.e Commit or Abort) in an asynchronous distributed system.

Watch out: How does 2PC "beat" FLP?

1. Agreement 2. Validity 3. Termination

→ Consider a naive attempt at distributed txn Commit



* Txn Executes on all nodes (R_1, R_2, R_3).
* Each node Sends its proposal (Commit/Abort) to all other nodes.
* If atleast one proposal is "Abort", all nodes Abort.
* Else all nodes Commit.

Qn: What is wrong with this algorithm?

Asymmetry ⇒ no Agreement

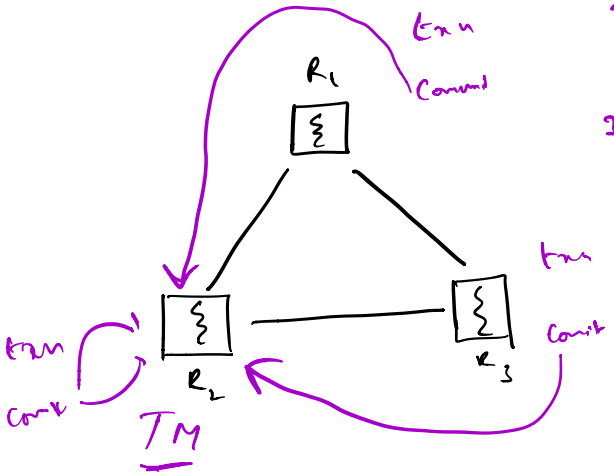
→ Diverging decisions (Lack of Agreement) can be fixed by making 1 node responsible for collecting proposals of all nodes, and then make a decisions.

→ Diverging decisions (Lack of Agreement) can be fixed by making 1 node responsible for collecting proposals of all nodes, and then make a decision.

"Transaction Manager (TM)"

"Resource Managers (RMs)"

* How will this work?

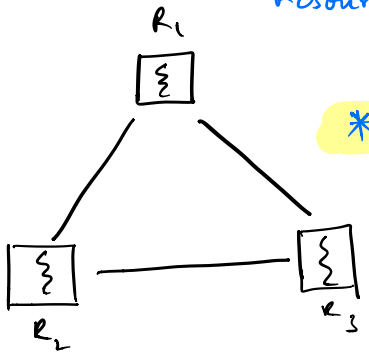


1. Execution
2. Communication of proposals
3. Making decision based on proposals
4. Communication of decision

→ Diverging decisions (Lack of Agreement) can be fixed by making 1 node responsible for collecting proposals of all nodes, and then make a decision.

"Transaction Manager (TM)"

"Resource Managers (RMs)"



TM

* How will this work?

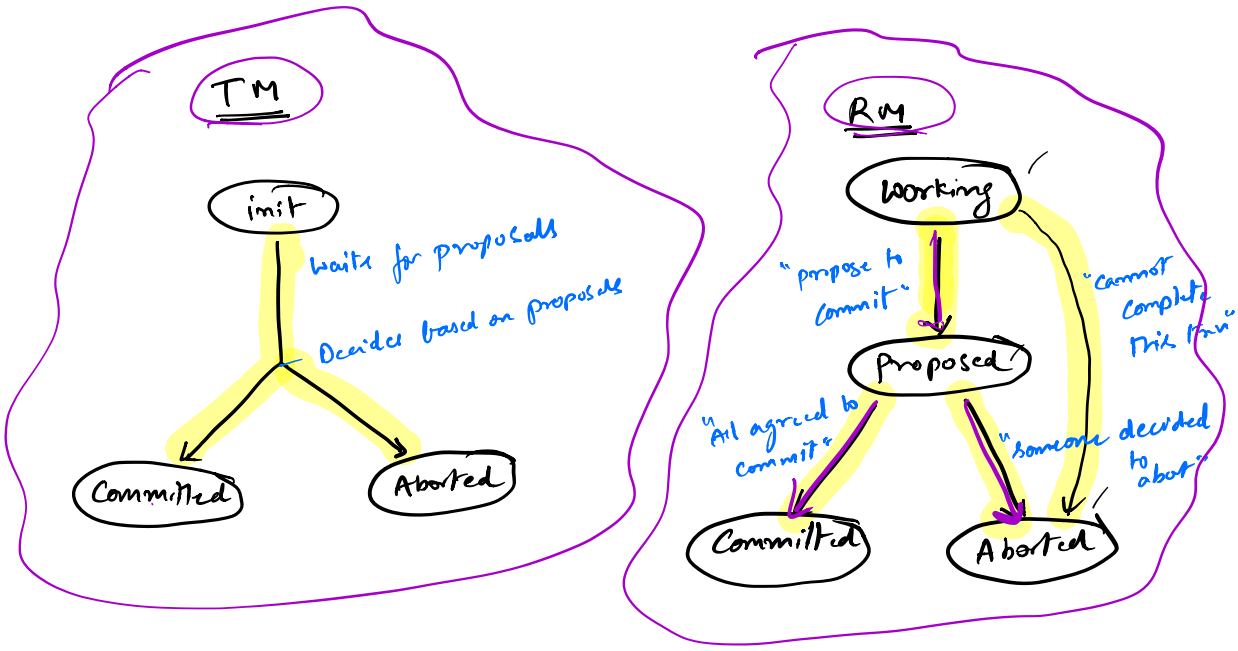
1. Each RM works on the txn independently. Depending on how the Execution goes, proposes to Commit or Abort. Only TM knows its proposal.

2. TM waits for proposals from all RMs. If all RMs propose to Commit, then TM decides to Commit. Else, TM decides to Abort.

3. TM communicates its decision to all RMs.

What should we call this algorithm?

Two-phase Commit

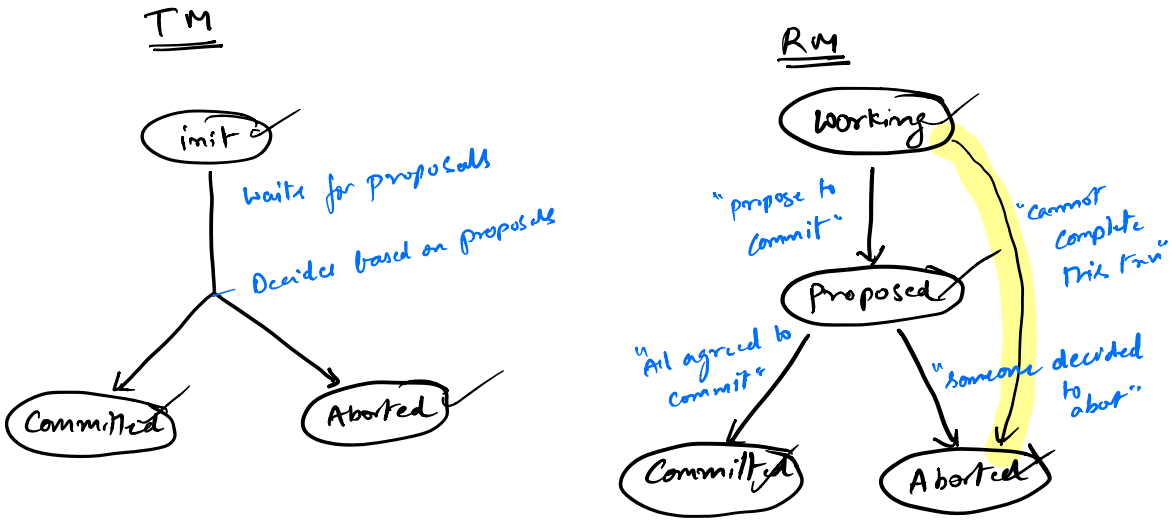


2PC Formalization

Elements of formalization:

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

P (tmState, rmState, msgBag, tmProposed)



2PC Formalization

Elements of formalization:

1. Representation of system state / configuration.
2. Initial configuration → ("init", [rm → "working"], {}, {})
3. Transition relation.

(tmState, rmState, tmProposed, msgs)

tmState ∈ {Init, Committed, Aborted}

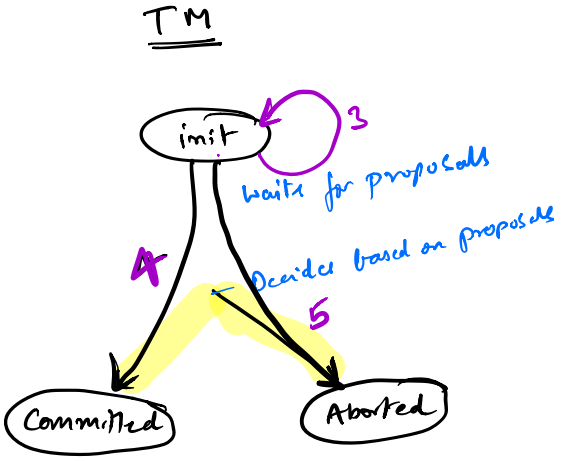
rmState ∈ [RM → {Working, Proposed, Committed, Aborted}]

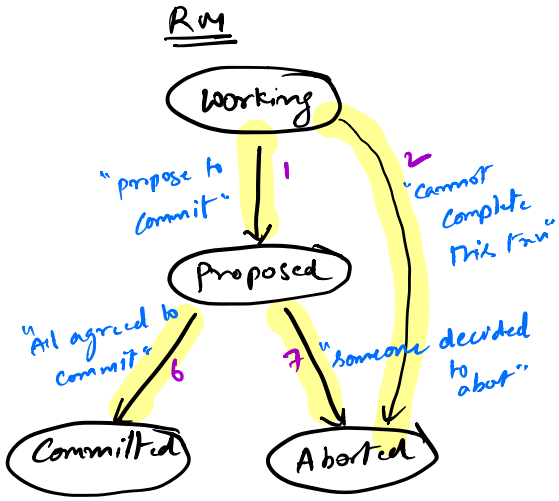
tmProposed ⊆ RM

msgs ∈ SUBSET ([type : {"Prepare"}, rm : RM] ∪ [type : {"Commit", "Abort"}])

SUBSET (RM × {"Prepare", "Commit", "Abort"})

RM × RM × {"Prepare", ~~"Commit"~~, "Abort", "DecideCommit", "DecideAbort"}





2PC Formalization

Elements of formalization :

1. Representation of system state / configuration.
2. Initial configuration — <tmState, rmState, tmPrepared msgs>
3. Transition relation. — <tmState', rmState', tmPrepared' msgs'>

Steps :

* Some RM proposes commit
* Some RM aborts
* TM receives commit proposal from one RM
* Some RM receives commit msg
* Some RM receives Abort msg
* TM decides to commit
* TM decides to Abort