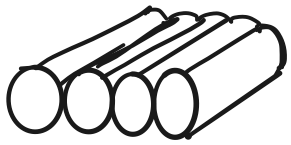
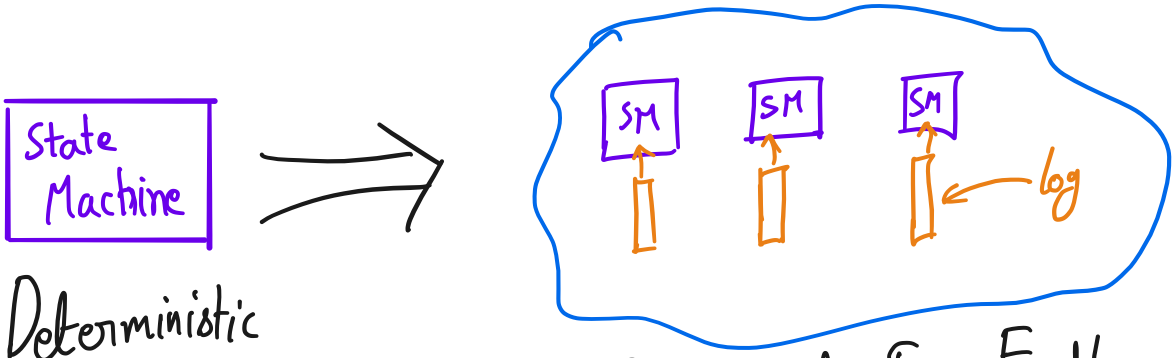
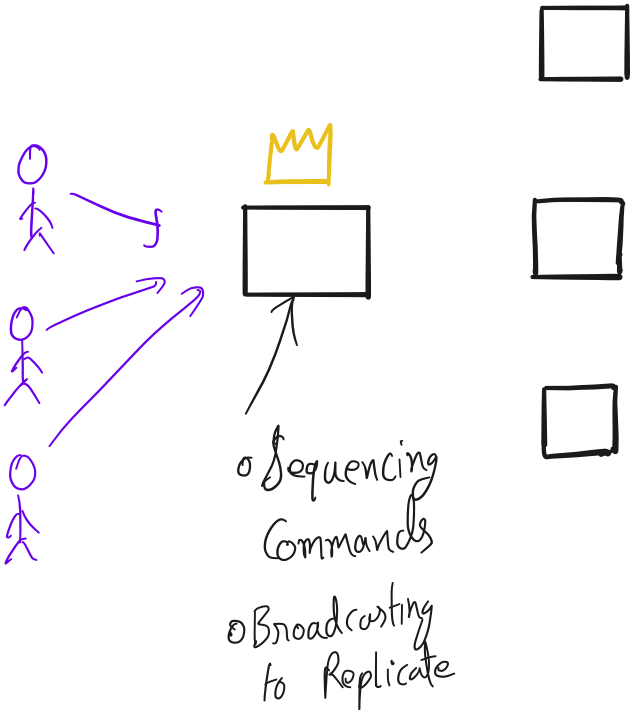
Raft



Last week: RSMs: A framework for building fault tolerant distributed application



Raft is one such protocol.

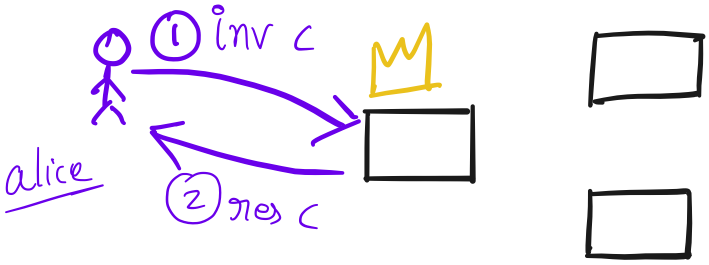


- Sequencing Commands
- Broadcasting to Replicate

Assumptions:
- Partial Synchrony

Safety Guarantees:
- Agreement
- Validity

Useful to Understand Requirements



① alice invokes command c

Commit Point

② System responds to alice

↳ c's effects are guaranteed to persist despite permissible failures

Commit Point in Rayt

- After command c is committed at slot i, slot i always contains c

  } Common to nearly all consensus protocols!

- All slots 0..i-1 contain committed commands

  } Log Completeness

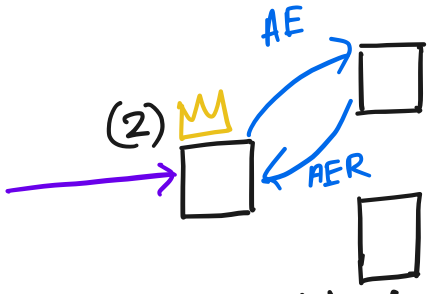
  Rayt-Only

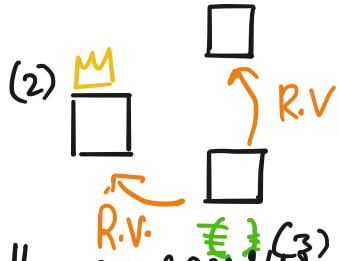
All of the portions of the Protocol are designed to maintain these properties.

- Replication (AE, AER)
- Leader Election (R.V0, R.V.R0)
- Reconfiguration

Core Approach: Quorum Intersection

Any two changes to Raft state (term, log, configuration) must have been signed off by a common node





Common building block for all consensus protocols

Parameter: f := # of nodes that can fail.

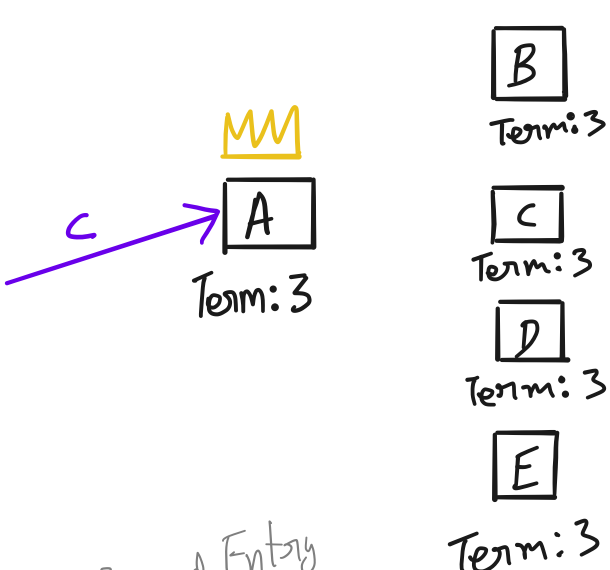
Min quorum size?

n - f n - f + 1 3 20

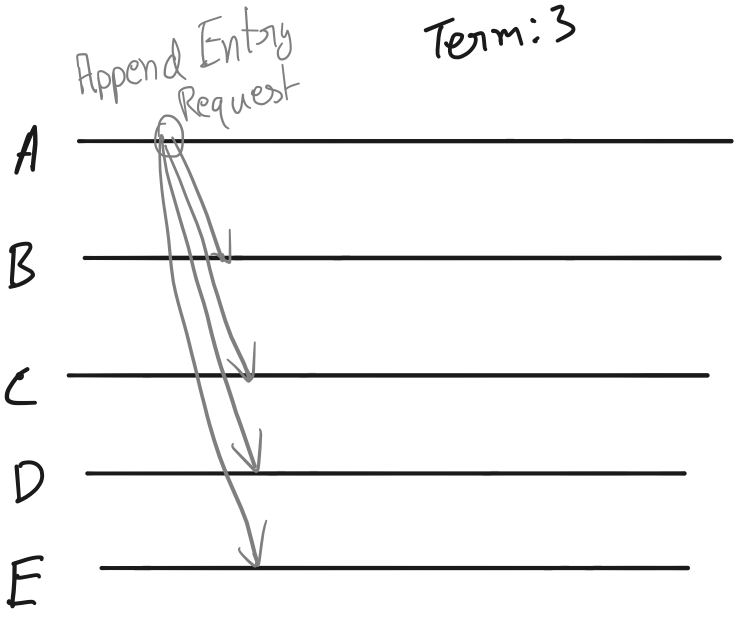
Invariants

- Term: $\mathbb{Z}$: $\emptyset$ or 1 leader. (Leader election ensures this)
- (term, index) $\longleftrightarrow$ command or $\perp$
- In steady state, leader's log is authoritative.

Replication/Log Synchronization



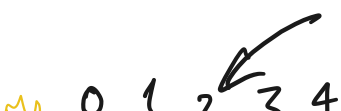
	0	1	2	3	4	5	6	7	8
A	T0	I_3'							
B	T0	I_3							
C	T0								
D	T0								
E									



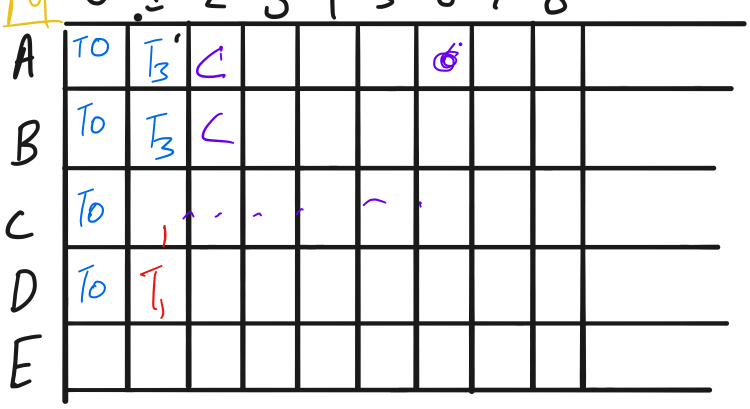
Append Entry Contents
- Term 3
- Command(s) c

Entries
- Last log index } Why?
- Last log term } 3
- CI

[<3, 1>, c]



Processing at B



Processing at C

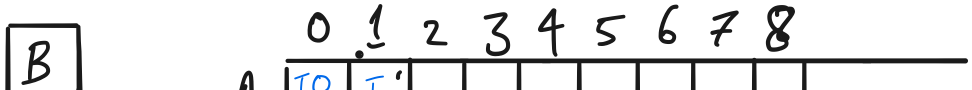
Processing responses at A (👑)

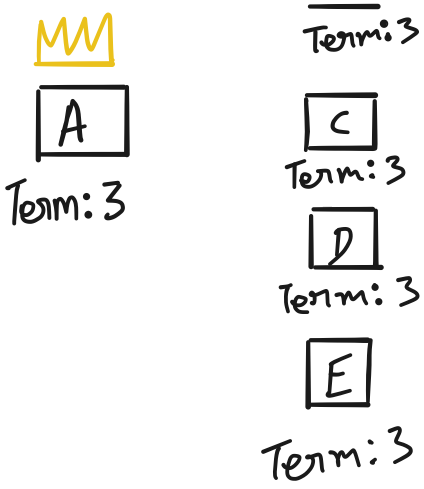
- When is a command committed?

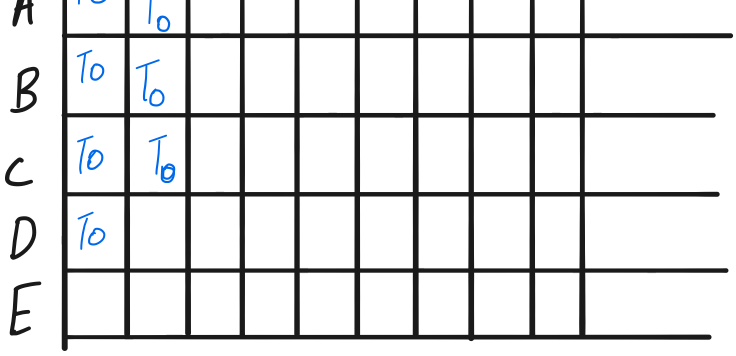
$n = 2f+1$?

- When can A execute the command?

- When can D execute the command?

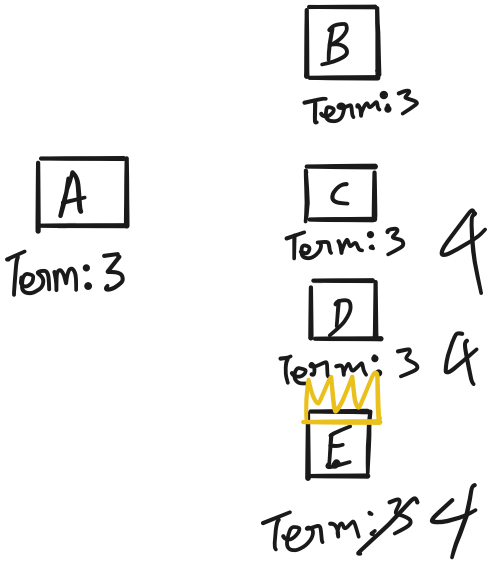


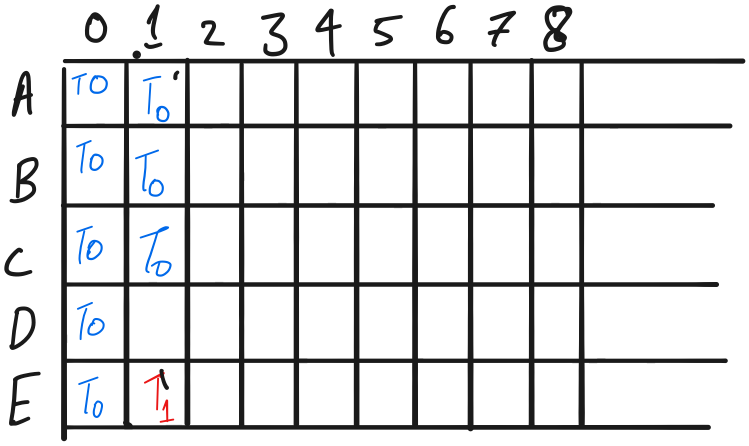




When can A execute command at index 1?

Why?





Addendum: When is a command commited?

Leader Election

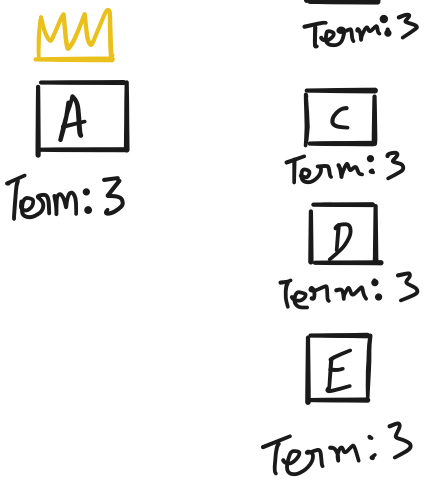
Goal: Allow progress if leader fails

Conflict

- Stable leadership is necessary for progress. Why?

Avoid leader election when unnecessary

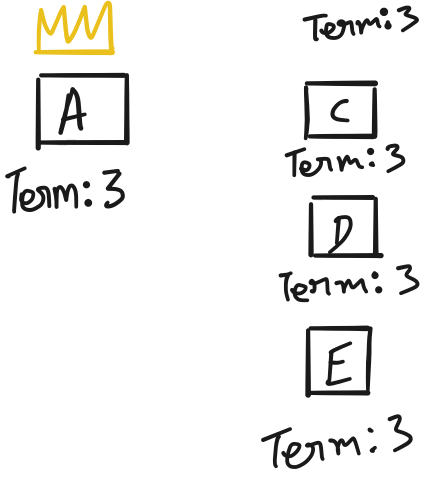
- Want to detect that a leader has failed soon. Why?



- Solution
  - Heartbeats: no-op append entry requests
- Handling at Leader
- Handling at follower.

Election Timeout

- Relation to Heartbeats?



Leader Election

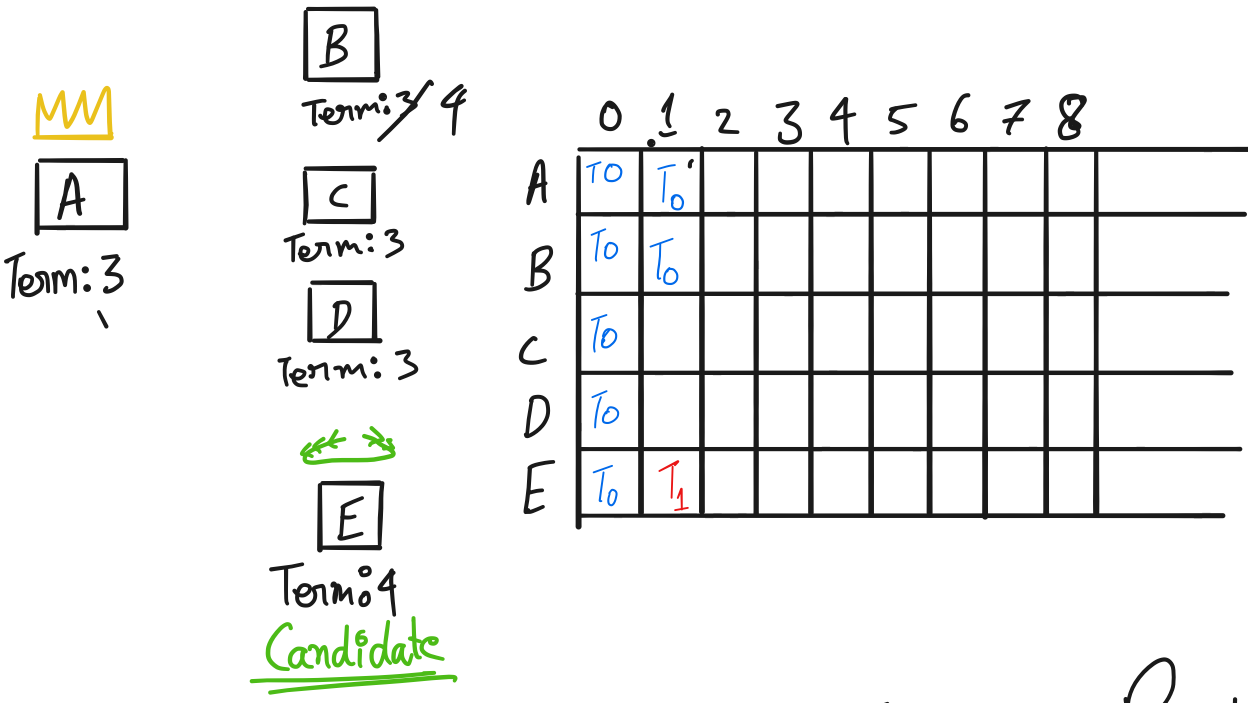
Requirement 1°: New leader's log should contain all commited log entries

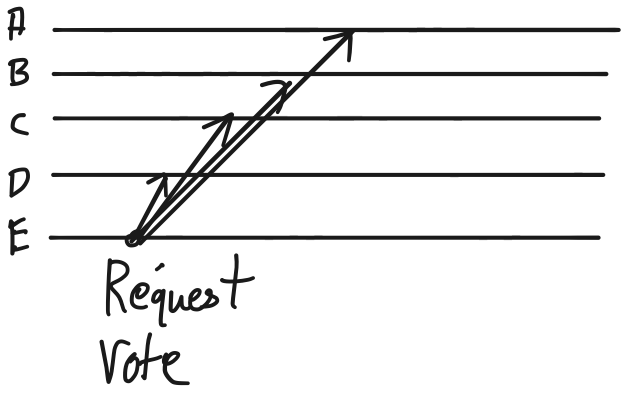
Slight refinement of

> After command c is committed at slot i, slot i always contains c

Requirement 2°: At most one leader at a time.

Protocol assumes there is no preference for what node becomes leader





Information Req Vote Req

term: 4

ID: E

Last Log Index: 1

Last Log Term: 1

What does a node (B) do on receiving R.V

B agrees to at most leader

(a) Update term (if received term is higher)

↳ Consequence

(b) Check if B has already voted in term

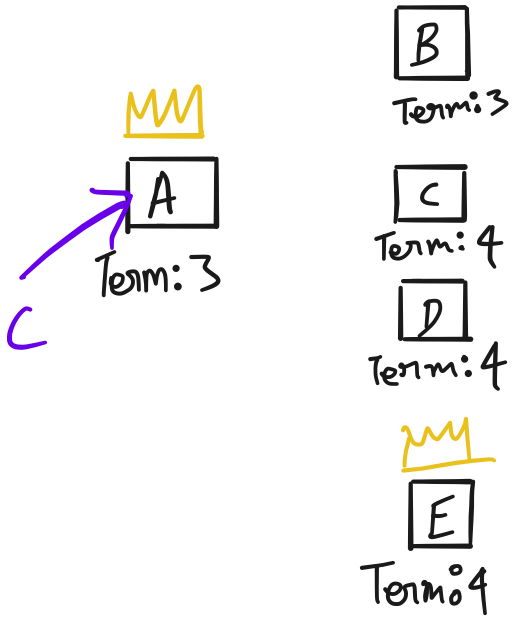
(c) Check who has more up-to-date log

Longer

Commit from a later term.

When does a candidate → leader?

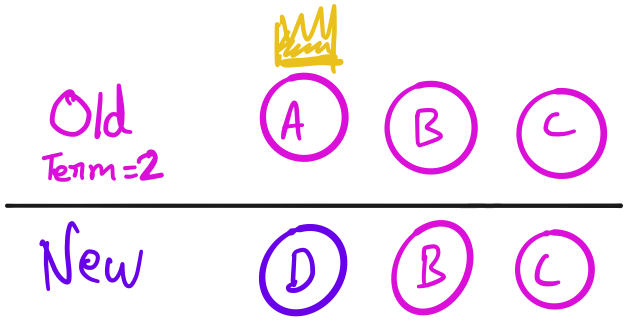
When does leader in term(t-1) → FOLLOWER (in term t)?

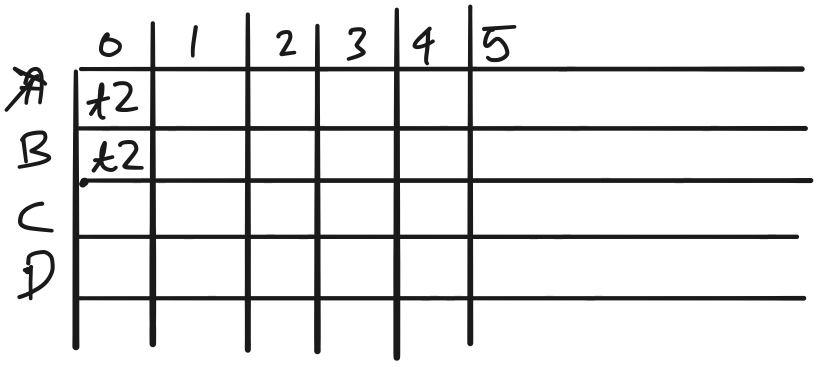


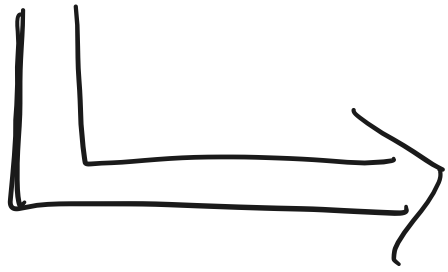
Re-configuration

- Goal: Change the set of nodes participating

- Why hard? Need to pick a slightly complex setting







Must ensure that D or C do not get elected. Why?

But A cannot continue as leader either. Why?

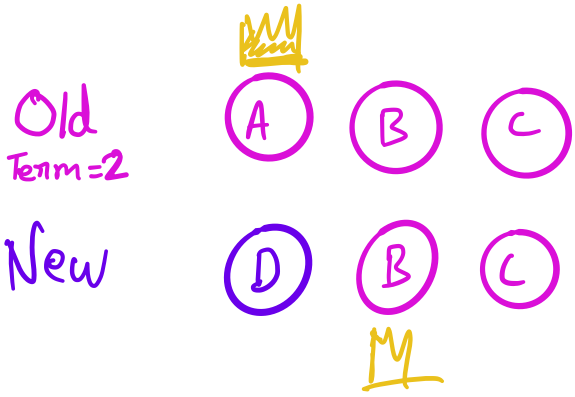
Things we need to fix

ⓐ Synchronize logs so that a quorum of nodes in the new config contain committed entries.

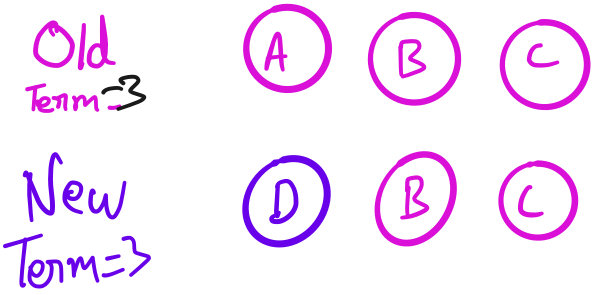
Observation: AE/AER synchronizes logs for a quorum of nodes so they match the leaders log.

Idea: ① Run leader election to find a node that

Joint Quorum
- (a) Can commit to nodes in the new configuration
- (b) Has all committed entries



② Have the new leader commit an entry



	0	1	2	3	4	5
A	t2					
B	t2					
C						
D						

③ Project?

Notes on practice

- Leader election

- Reconfiguration