Raft (Continued)

Reminder: Project proposal — Next week

Reminder from last class (or two)

RSM :- Replicate commands + order in which they are executed

- Execute command once it (and its order) has been sufficiently replicated.
  [Not going to be lost due to failures]

Raft - A protocol for this.

Core inv: - Leader based - one leader at a time. Term lockout

- Leader's log is authoritative

  ↳ Protocol is designed to replicate the Leader's log to all other replica's

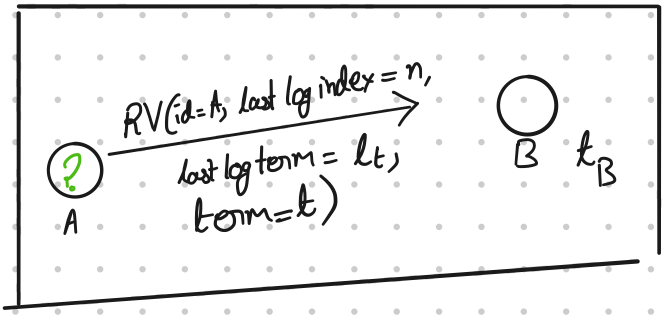
[Designed to do so efficiently]

AE + Log matching

- Node n becomes leader in term t
  ⇒ n's log contains all commands committed before term t
  [Leader Completeness]

How? Restrictions on granting votes + becoming leader

- To become leader in term t, node A must get votes from quorum.

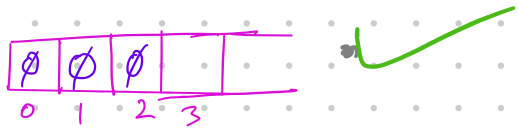


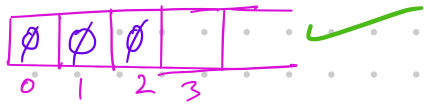
- B grants vote if
  - $t_B \leq t$
  - Not voted for anyone in term t.
  - l_t > term of last log entry at B

    OR [l_t = term of last log entry @ B AND

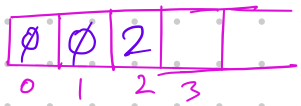
    n ≥ index of last log entry @ B]

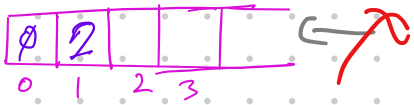
"A's log is at least as up to date as B"

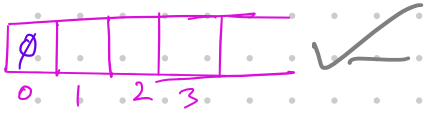




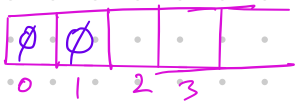
Term=3 CANDIDATE

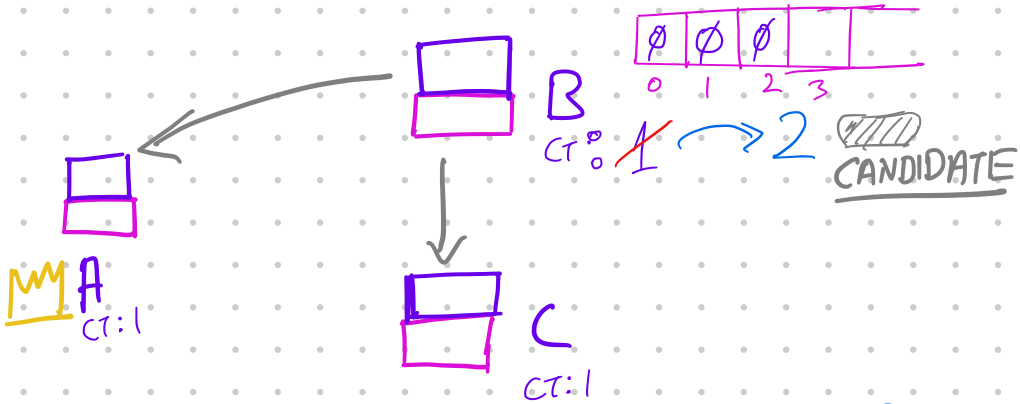




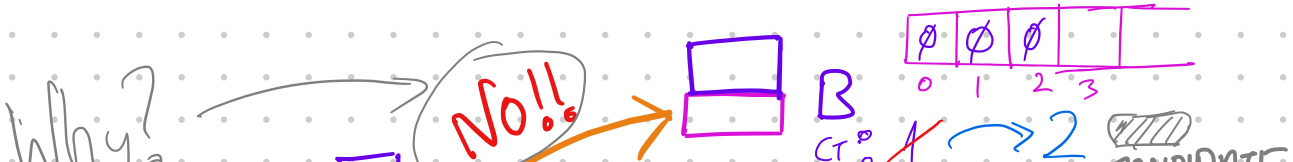


Term=3 CANDIDATE

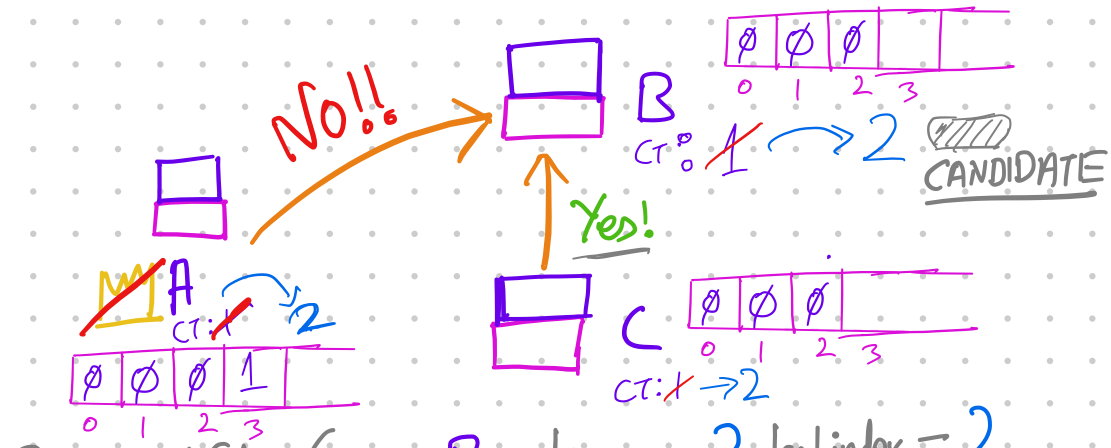




RequestVote (ID=B, term=2, lastIndex=2, lastIndexTerm=∅)

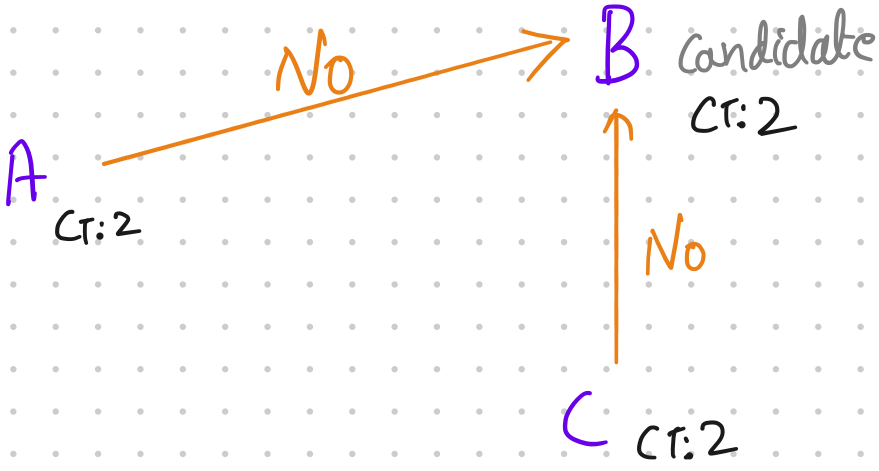


RequestVote (ID = B, term = 2, lastIndex = 2, lastIndexTerm = ∅)



RequestVote (ID = B, term = 2, lastIndex = 2, lastIndexTerm = ∅)

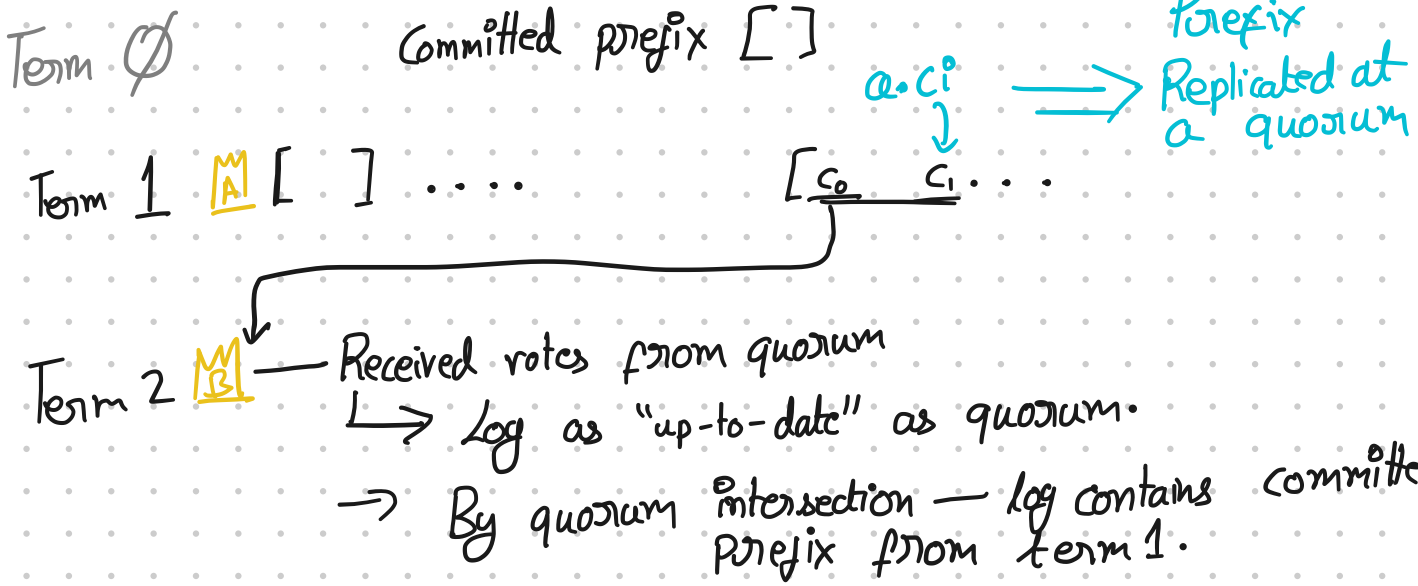
Observe: Candidate might not become leader



What now? [Talked about this last class]

- Node n becomes leader in term t
  ⇒ n's log contains all commands committed before term t } How?

Observe — Protocol structure ⇒ Prefix committed



More generally

Easy to show: Term 1 leader's log contains committed prefix

↳ Trivial → Committed prefix is empty

Assume term t leader's log contains committed prefix from term $t-1$, $t-1$ leader had $t-2$ prefix, ...

Show any node n that can become leader in term $t+1$ must contain committed prefix from term t

↳ Let us say last command committed in term t is at index l.

Equivalently ⟶ $[0 \dots i]$ prefix committed in term i

$\Rightarrow$ $[0 \dots i]$ replicated at a quorum

[Protocol requirement to commit]

Node n can become leader in $t+1$ if it can get votes from a quorum

$\Rightarrow \exists$ a quorum Q s.t.

$\exists$ a quorum Q, s.t.

$\forall$ $q \in Q$

either

last log entry in q was committed in a term BEFORE n's last last log entry

OR q has fewer entries than n.

But $\exists\, q \in Q$, s.t. q's log contains $[0 \ldots i]$ log prefix [Quorum intersection]

Say q's log is $[0 \ldots i \ldots j]$

If command j added at term $> t$
↳ Q cannot exist [why]

If command j added in term t
$\Rightarrow$ n's log contains

[0...i...j]

If command j added in term t' $[t'<t]$ & ; Committed in t'' $[t'' \leq t']$

↳ $t''=t'$ → Trivial
↳ q log [0...i...j]

↳ $t''<t'$ → Inductive hypothesis

$t''+1$ leader log had [0...i] prefix

$t''+2$

⋮

t' had [0..i] prefix q log [0..i..j]

$t+1$ ← Term @ n leader

t ←

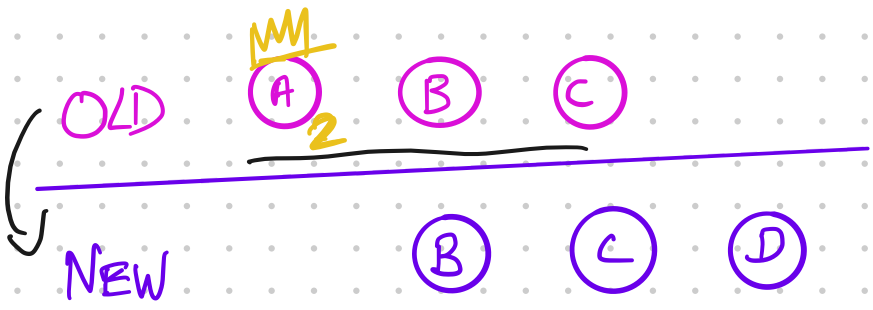
t' ← Term @ j app. q

t'' ← Term @ i comm

Reconfiguration — Joint quorums

- Disclaimer :

- Want to add and/or remove nodes
  - ↳ While continuing to process requests & handle failures



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

Leader completeness: Who in the new configuration (B, C, D) can be leader?

Leader election protocol:
Who in the new configuration (B, C, D) can be leader?

	LI	LI
B	3	2
C	1	1
D	-	-

Can be worse.

NEW (B) (C) (D) (E) (F)

Observation: Need some way to maintain leader completeness during reconfig.
- JOINT QUORUM

Building Blocks

① Configuration stored as log entry

ⓐ Usual rules for appending.

⇒ Node A: config Cat index 5 in term 2

Node B: config Cat index 5 in term 2

What can you say about A's log & B's log?

ⓑ Unusual Rules for applying

↳ Node uses last configuration in its log: does not wait for commit.

→ But configuration change COMPLETE

only when

② Joint quorum

Old → New implemented as old → old+new → new

Commit

Configurations

OLD: (A) (B) (C) — A: WW, 2

OLD+NEW: [(A) (B) (C)] [(B) (C) (D)]

NEW: (B) (C) (D)

OLD+NEW: [(A) (B) (C)] OLD [(B) (C) (D)] NEW

↳ Same node needs to be leader in both old & new

Leader Election

OLD	LI	LT
A	5	2
B	5	2
C	1	1

NEW	LI	LT
B	5	2
C	1	1
D	–	–

	1	2	3	4	5
A	1	1	2	2	C^2_{OtN}
B	1	1	2	2	C^2_{OtN}
C	1				
D					

old to

— Any entry must be committed to both old & new.

OLD	LI	LT
A	5	2
B	5	2
C	1	1

NEW	LI	LT
B	5	2
C	1	1
D	–	–

TRICKINESS: When to go from

Old+new → new?

After old+new committed. Why?

OLD	LI	LT
A	5	2
B	5	2
C	1	1

NEW	LI	LT
B	5	2
C	5	2
D	–	–

– Where does B replicate C^3_N?

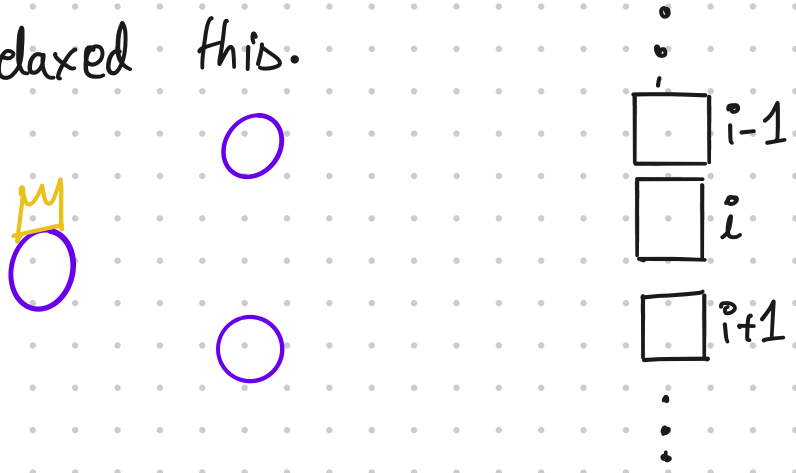
- Is {A, B} a quorum for C_{No}^{3}?

Paxos

Above Raft: Observe — Protocol structure ⇒ Prefix committed

[- - - - c]

What if we relaxed this.



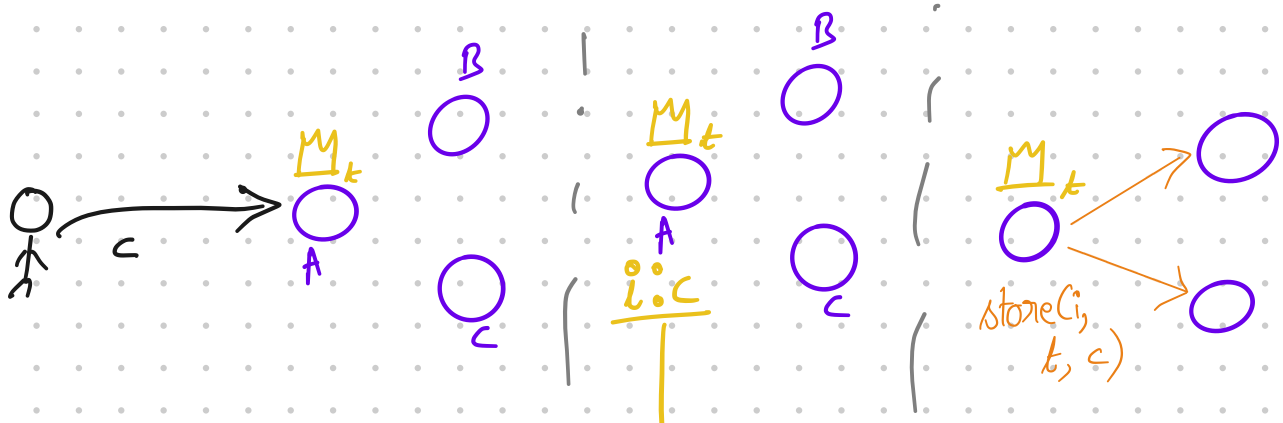
Big Requirements

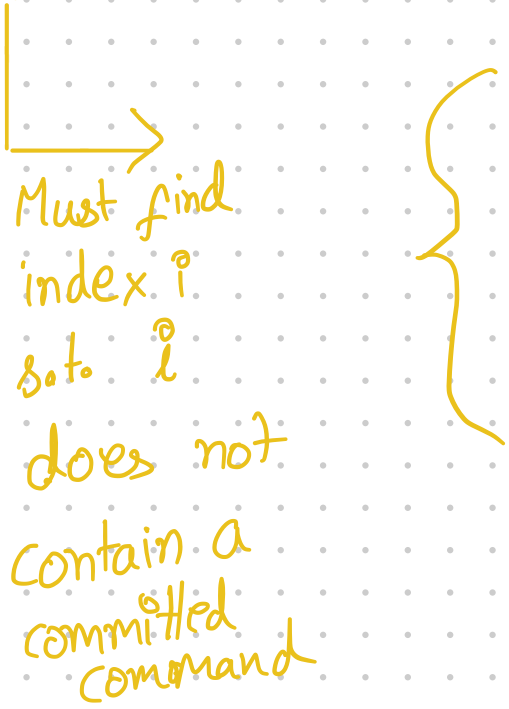
@ Make sure leader does not overwrite a committed command.

If command c committed at index i then index i should neve contain command c'

$c' \neq c$

Last class :- Load & store





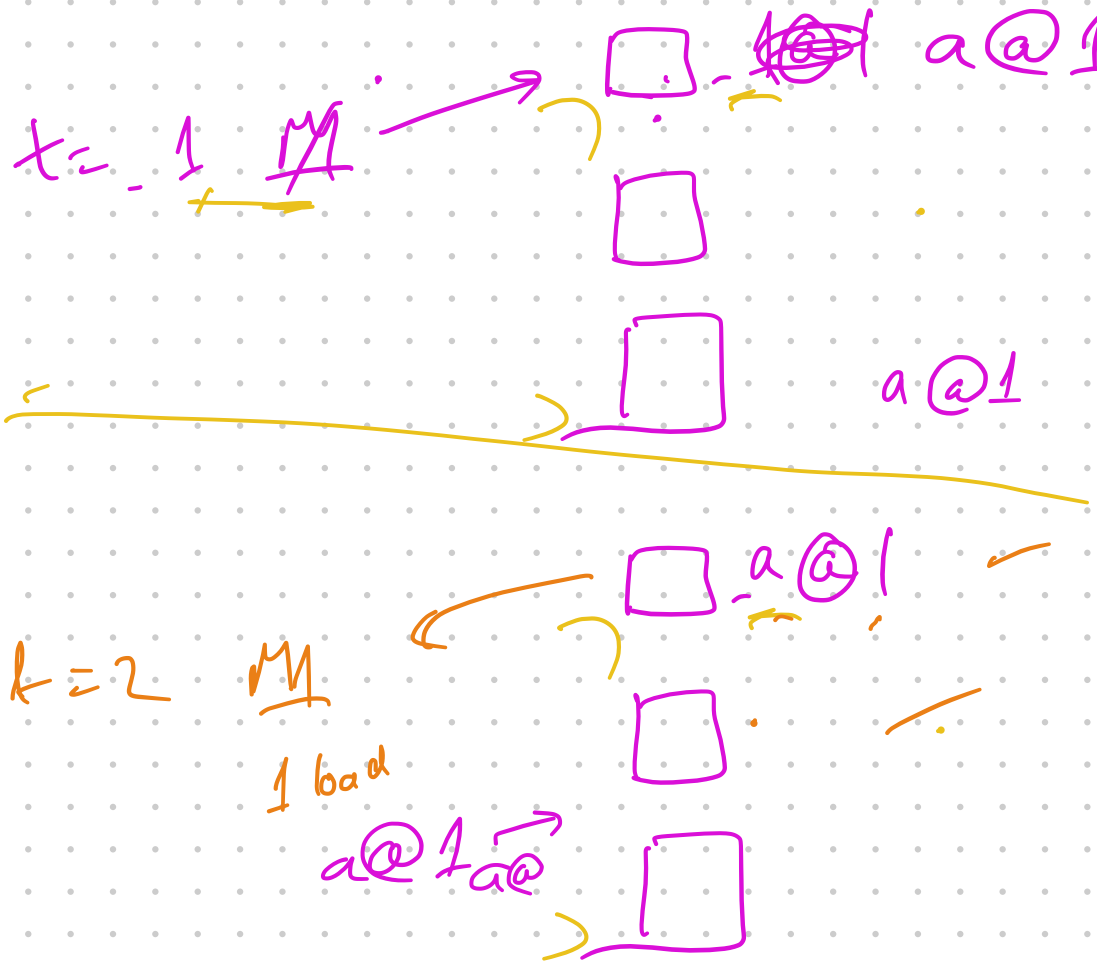
Synod Protocol

Round/term t

Phase 1 :- Load to find value at slot/index

Phase 2. If phase 1 found
values $\{v_1@t_1, v_2@t_2, \ldots\}$
pick value v_i at max t_i | Why
store (v_i, t)

else
store $(\perp, t)$



A 2

k=7 B 1

k=3 C 1.3

k=3 D 1 3

k=3 E 1 3