RAFT CONTINUED

(& Maybe Multipaxos)

Announcements: - Midterm on 03/12
- Class: 03/19

Last class

- Quorum intersection
- Walk through replication
  - When does a leader in term t know that an entry (at idx i) in term t is committed?
  - When does a follower know (i, t) is committed.
- Kind of walk through leader election

Safety Requirements

[ELECTION SAFETY] At most 1 active leader — [Leader election uses quorum

intersection so that

[STATE MACHINE SAFETY] - If node A applies (delivers) command c at index i to its state machine, then no other node will apply c' (c' ≠ c) at index i

↳ Only committed indices are applied

+ A committed index is never overwritten

[LEADER COMPLETENESS]

↳ Log entries committed in term t are present in the log for all leaders in terms > t

[LEADER APPEND-ONLY] + LEADERS DO NOT OVERWRITE ENTRIES IN THEIR LOG, THEY ONLY APPEND

MIA: LOG MATCHING?

Log Matching: if two logs (two nodes) contain an entry with the same index (I) and term (T) then both have identical entries at index ∅..I.

Why? Leader completeness requires ensuring that the term T leader's log contains all entries committed in terms < T.

↳ Need a way to check log contents before voting for leader.

- Makes this cheaper.

How? Append Entries Req

- Term

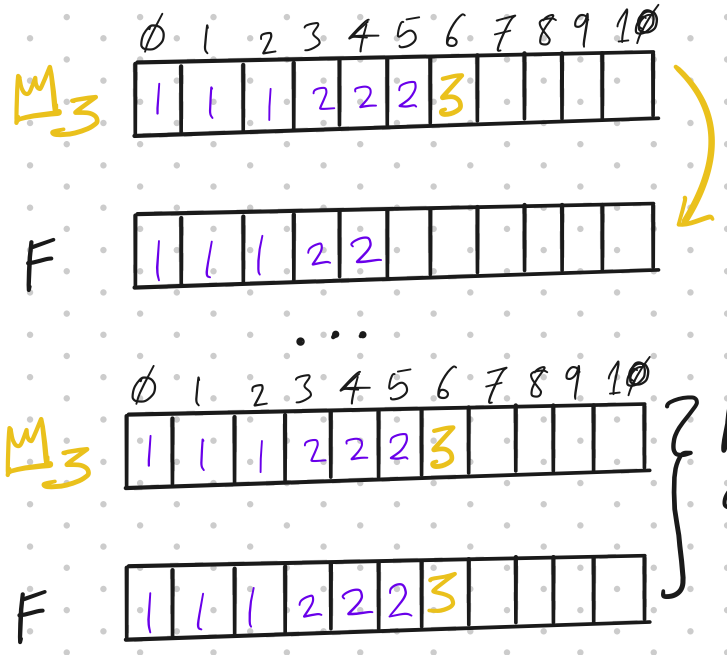
- entries
- prevLogIndex
- prevLog Term
- leaderCommit
- leader ID

if AER.term >= currentTerm

ONLY CALLED BY LEADER +

if not log.contains(prevLogIndex) OR log[prevLogIndex].term ≠ prevLog Index

SEND FALSE RESPONSE



Adding entry at index 6 of follower's log succeeds iff leader and follower have same entry at index 5

Induction { @ ⇒ Entry at index n determines entry at n-1

| @ Index 0, term 1

(b) Index 0, term 1 ≡ [illegible]

Trivial

→ Log matching!

ELECTION SAFETY
LEADER COMPLETENESS } Leader Election
- Quorum intersection + Vote criterion

LEADER APPEND ONLY } Protocol logic at lead
LOG MATCHING } Append entry mechanism

STATE MACHINE SAFETY } LAO + LEADER COMPLETENESS + ...

LEADER COMPLETENESS

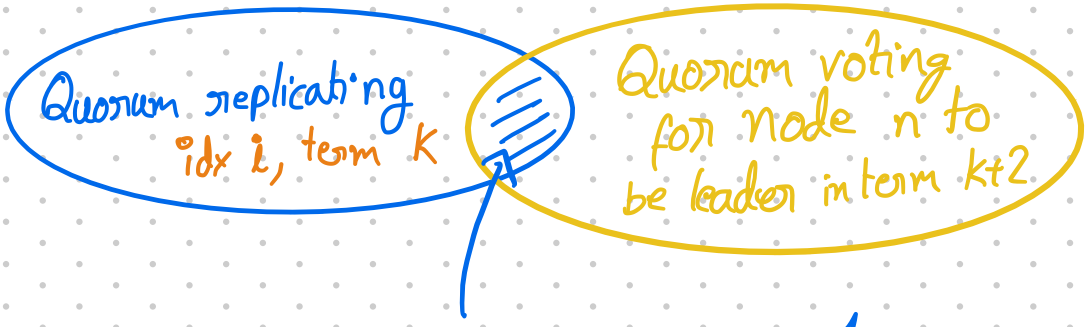
Log entries committed in term t are present in the log for all leaders in terms $> t$

① Any committed entry is replicated by a quorum

↳ Remember
- Necessary
- Not sufficient

↳ An entry replicated by a quorum might not be committed.

② Quorum intersection



Not empty. Must contain 1 or more nodes.

③ Most "up-to-date" log in a quorum contains all committed entries.

Defⁿ Up-to-date [Note: concerns raised last week]

Log A (node A) more up to date than Log B

if

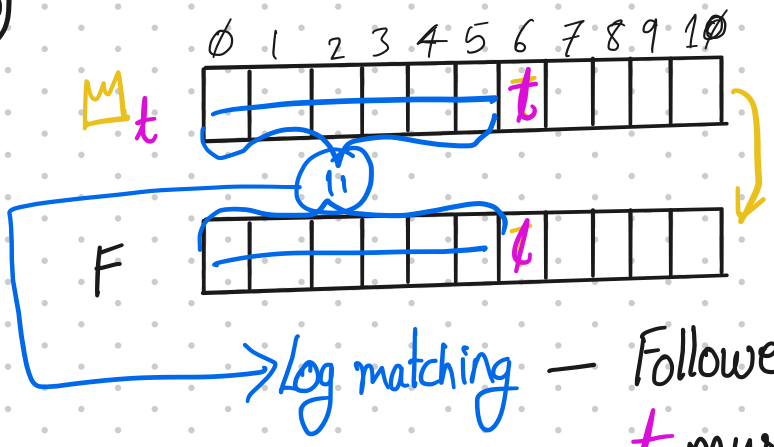
- Last index in A has higher term (was written in higher term) than last index in B
- Last index in A & B have same term and A's last log index > B's last log index.

Why? ① All log updates driven by leader.

② If leader completeness* holds up to term t

↳ t has all entries committed in $0 \ldots (t-1)$

ⓐ

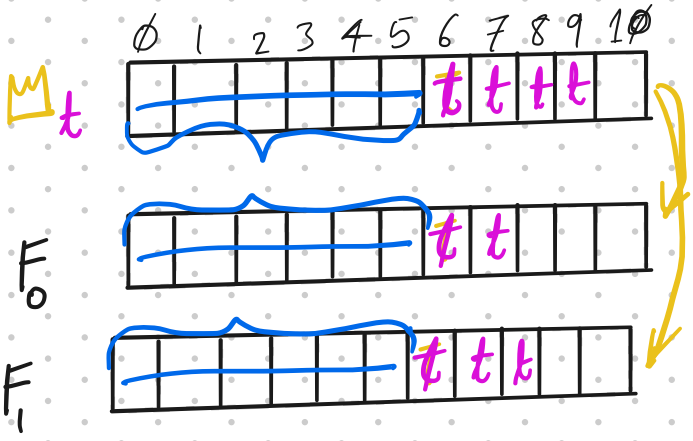


F t

F $t-2$

Log matching — Follower with entry in term t must contain entries committed in $0 \ldots t-1$

ⓑ



In a quorum where all last log index for all logs is t, follower with longest log must contain all entries committed in t.

④ Leader election protocol chooses most up-to-date node in quorum as leader.

	LI	LT
A	3	7

Max committed index + term

$\langle 4,7 \rangle$ is not sufficiently

	LI	LT
B	5	2
C	4	7
D	2	2
E	3	7

Index: 3
Term: 7
} replicated

Note: <3,7> Might not be committed.
But it could be!

	LI	LT
A	3	7
B	5	2
C	4	7
D	2	2
E	3	7

	Can get votes from	Can Become Ldr
A	A, B, D, E	✓
B	B, D	X
C	A, B, C, D, E	✓
D	D	X
E	A, B, D, E	✓

	LI	LT
A	8	7
B	5	2
C	5	2
D	5	2
E	6	3

Max possibly committed

Term: 5

Index: 2

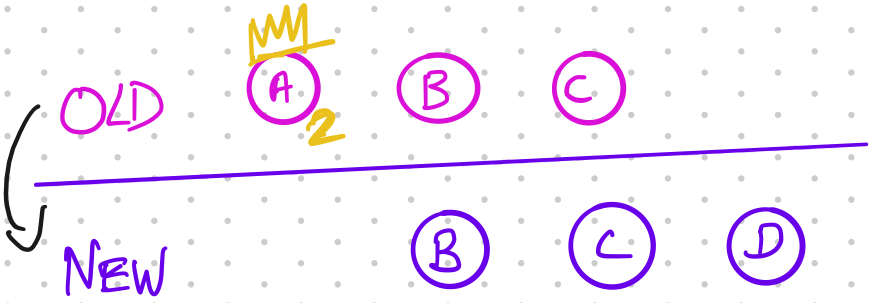
	Can be leader?	Quorum?
A	✓	
B	✓	

C
D
E

Note: Not quite a proof (assumed leader completeness before) but should be able to modify into proof by induction [Term of leader trivially has all previously committed entries]

Election safety? 1-vote each term

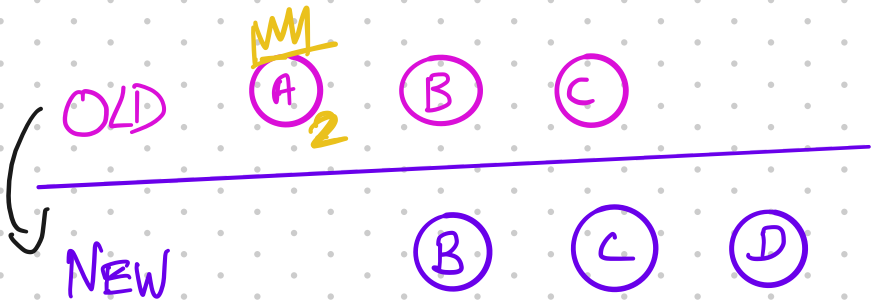
Reconfiguration
- Want to add and/or remove nodes
  ↳ While continuing to process requests & handle failures



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

Intersect: Who in the new

Leader completeness & configuration (B, C, D) can be leader?



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

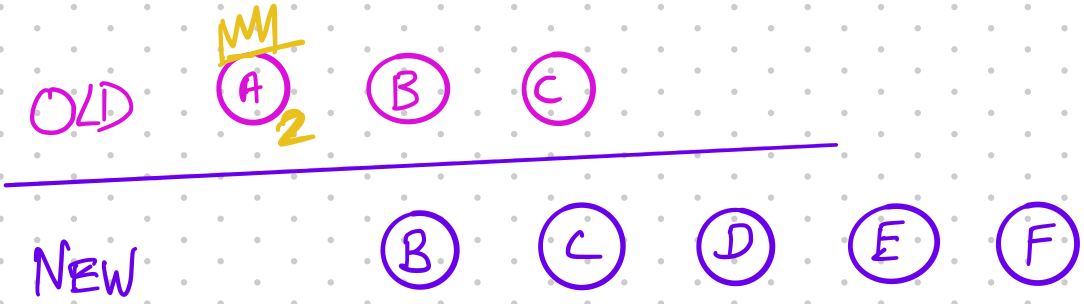
Leader election protocol:

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

Note: needs to get quorum

	LI	LT
B	3	2
C	1	1
D	-	-

Can make this worse



	LI	LT
B	3	2

C 1 1
D - -
E - -
F - -

Need to guarantee LEADER COMPLETENESS during configuration change.

Joint quorums.

Building Blocks

① Configuration stored as log entry

ⓐ Usual rules for appending.

⇒ Node A: config Cat index 5 in term 2

Node B: configCat 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.

⇒ Reconfiguration change COMPLETE

→ But configuration ... only when entry is committed

② Joint quorum

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

OLD (A)$_2$ B C

OLD + NEW (A B C) (B C D)

NEW B C D

OLD + NEW (A B C) OLD (B C D) NEW

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

① Leader Election

OLD

	LI	LT
A	1	1
B	1	1
C	1	
D		

NEW

	LI	LT
B	5	2
C	5	2
D	5	2

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

	LI	LT
A	5	2
B	5	2
C	1	1

	LI	LT
B	5	2
C	1	1
D	–	–

② Any entry must be committed to

	LI	LT
B	5	2
C	1	1
D	–	–

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?

NEW

	LI	LT
A	5	2
B	5	2
C	1	1

	LI	LT
B	5	2
C	5	2
D	-	-

	1	2	3	4	5	6
A	1	1	2	2	C^2_{0+N}	
B	1	1	2	2	C^2_{0+N}	C^3_N
C	1	1	2	2	C^2_{0+N}	
D						

- Where does B replicate C^3_N?

- Is {A, B} a quorum for C^3_N?

Paxos: A Greek island in the Ionian Sea

↳ Core: Synod protocol

↳ Agreement on a single log index

- A committed entry is never overwritten

PROPOSER

PROPOSAL ID

↓

Monotonically increasing number

ACCEPTOR — Last Proposal ID seen

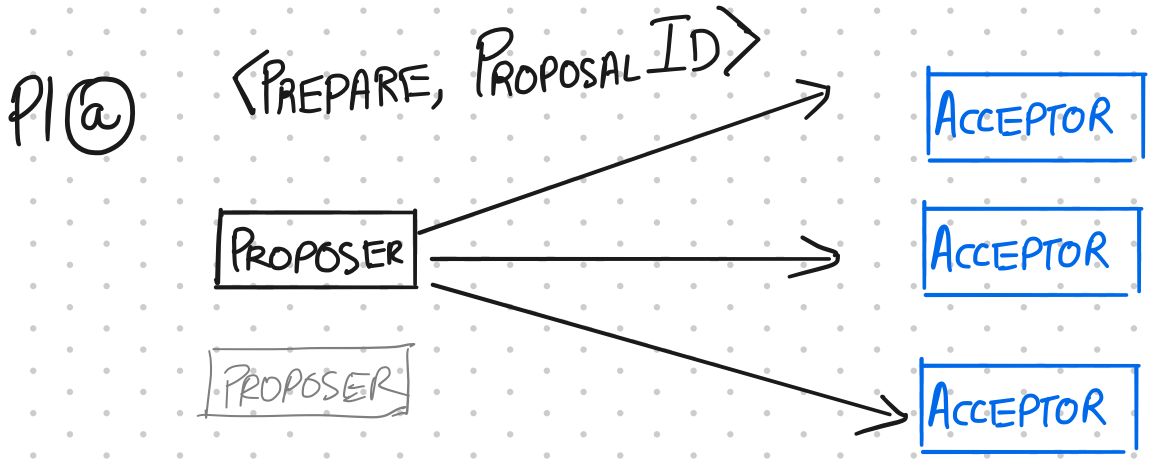
ACCEPTOR

ACCEPTOR

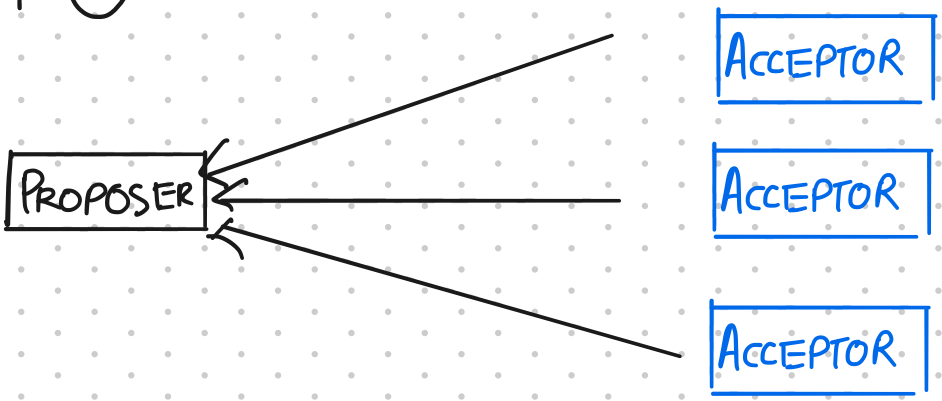
Value committed when "ACCEPTED" By QUORUM $(N/2+1)$

(think of term)

P1@



P1(b)



If ProposalID > last PID then

last PID = PID

send( )

<PROMISE,
PROPOSAL ID,
PREV. ACCEPTED VALUE (or 1),
PROPOSAL ID WHEN ACCEPTED (or 1)>

PROPOSER

↳ Wait for promise from quorum.

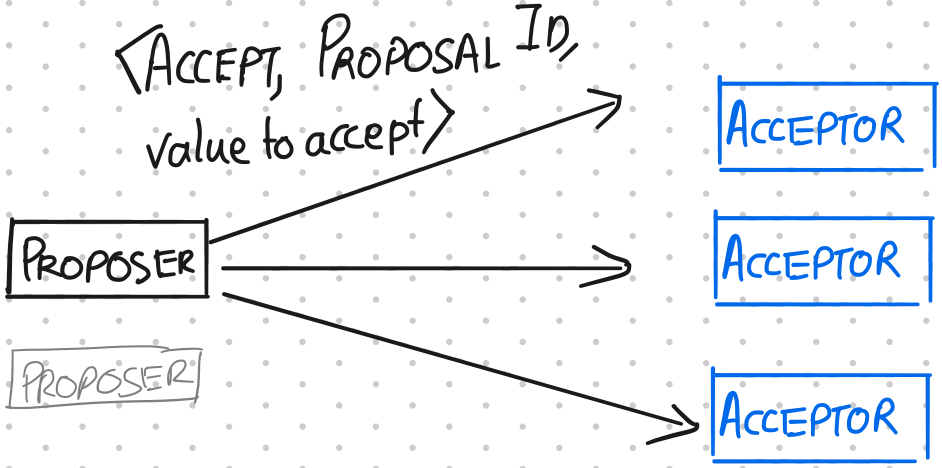
If no prev. accepted value:

value to accept = input

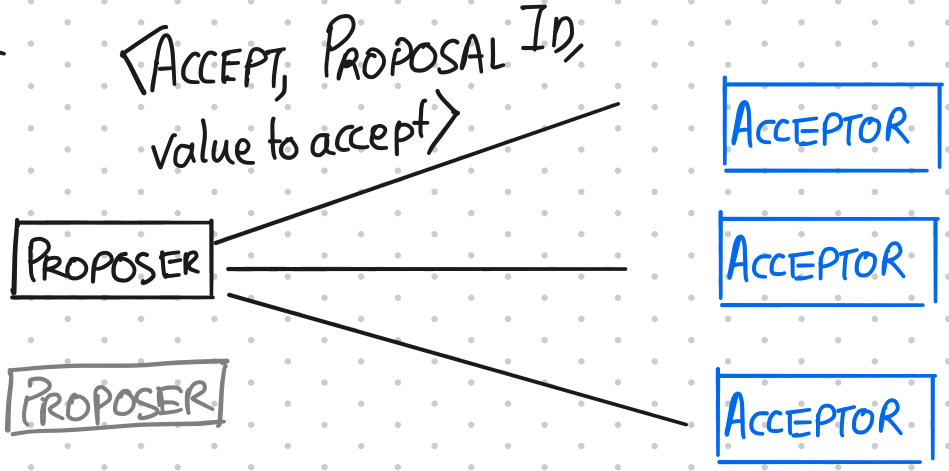
else

value to accept = Prev. accepted value w/ highest proposal ID

P2a



P2b



if PROPOSAL ID = last PID then accept val.

send(...)

Observation: Proposer can recover a previously committed value

Looking ahead

Leader -> Proposer

Relax leader completeness but require P1a/b on the log

	LI	LT	0	1
A			1	2
B			1	2
C			1	2
D			1	
E			1	

3

LI LT	1	2	3	4	5	6	7
A 8,7					2	3	
B 5,2					2		
C 5,2					2		
D 5,2					2		
E 6,3					2	3	