Distributed Systems: Randomized Consensus

- Final project proposals
  - Sent feedback to some of you about concerns
  - The rest seemed great
    - ↳ Very excited about some of the proposals!
- Poll on guest lecture
  - Currently, we have a class on BFT scheduled in 2 weeks

- But goes over the same material as Daniel

- Two options
  - → COVER THE SAME MATERIAL
  - → USE THE TIME FOR OTHER TOPICS

- Exam observation

Commit: A persistence property
↳ THIS LOG ENTRY WILL NOT BE OVERWRITTEN/DELETED.
↳ Regardless of how the system evolves/future transitions (assuming failure model + correct impl).

What it is not
- A node (e.g., the leader) knowing this.

When are entries committed?

EASY CASE: MULTI PAXOS

Phase 1: Collect any entries replicated to a majority

Lamport'01

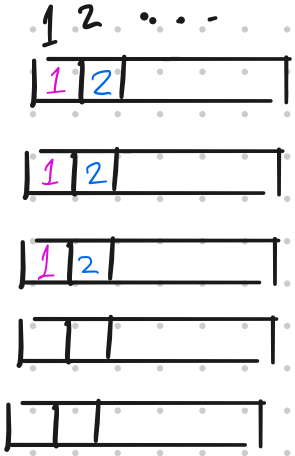
P2^b : If a proposal with value v is chosen then

EVERY HIGHER-NUMBERED PROPOSAL HAS VALUE v.

An entry is either committed (replicated to a quorum) or not.

HARD CASE : Raft

- Replicating to quorum is necessary but not sufficient
- But,



T=2 (a) Sufficiently replicating entries in the current term allows for commit

(b) For everything else apply the latest log rule.

Randomized Consensus Protocols

Why? Because of FLP [next class]

There is No deterministic, fault tolerant Consensus Protocol (Agreement, Validity, Termination) in the asynchronous model.

- But you have already looked at Two Consensus Protocols (and implemented one)...

We assumed a bit more than the asynch. model
↳ Timeouts for Heartbeats & Leader Election
Partial Synchrony [also next class]

There is ~~No~~ deterministic, fault tolerant Consensus Protocol (Agreement, Validity, Termination) in the ~~asynchronous model.~~

Today's goal: Relax a different constraint

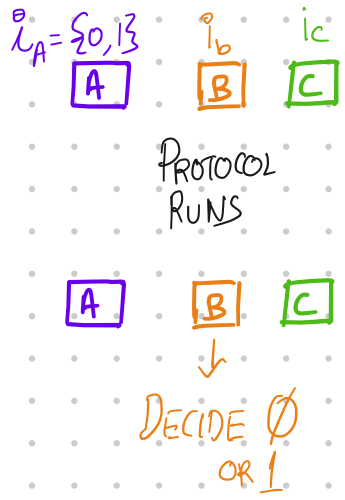
There is ? ~~No deterministic~~ Randomized, fault tolerant Consensus Protocol (Agreement, Validity, Termination) in the asynchronous model.

Why?

- Introduces binary consensus ∴ Used in FLP.
- May remove some assumptions about deployment — How To Set Timeouts
- Intellectually interesting
- From people here !!

Ben-Or Consensus

- Binary Consensus



- Agreement: If process p decides v and process q decides v' then $v = v'$
- Termination: Eventually some process p decides.
- Validity: The decided value v must have been some processes input
  - If all inputs are ∅ then protocol decides ∅

    ∅ ∅ ∅ ⇒ Decide ∅
  - If all inputs are 1 then protocol decides 1

    1 1 1 ⇒ Decide 1

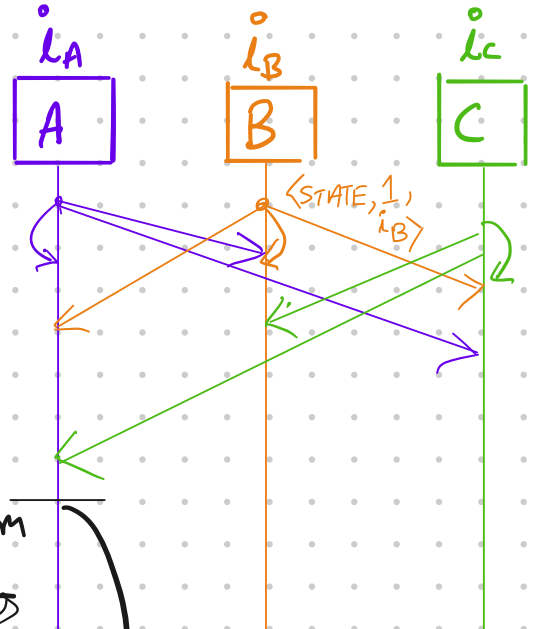
ϕ 1 ϕ ⇒ Decide ϕ } Both are OK!
ϕ 1 ϕ ⇒ Decide 1 }

Ben-Or: Binary consensus with
- Agreement
- Validity
- Termination w/ probability 1

Core Techniques
- Quorum Intersection
- Random Coin Flips To Break Conflicts
  ↳ Think back to Paxos discussion re: termination with competing proposers Section 2.4

Round 1

Round ↓
⟨State, 1, i_A⟩



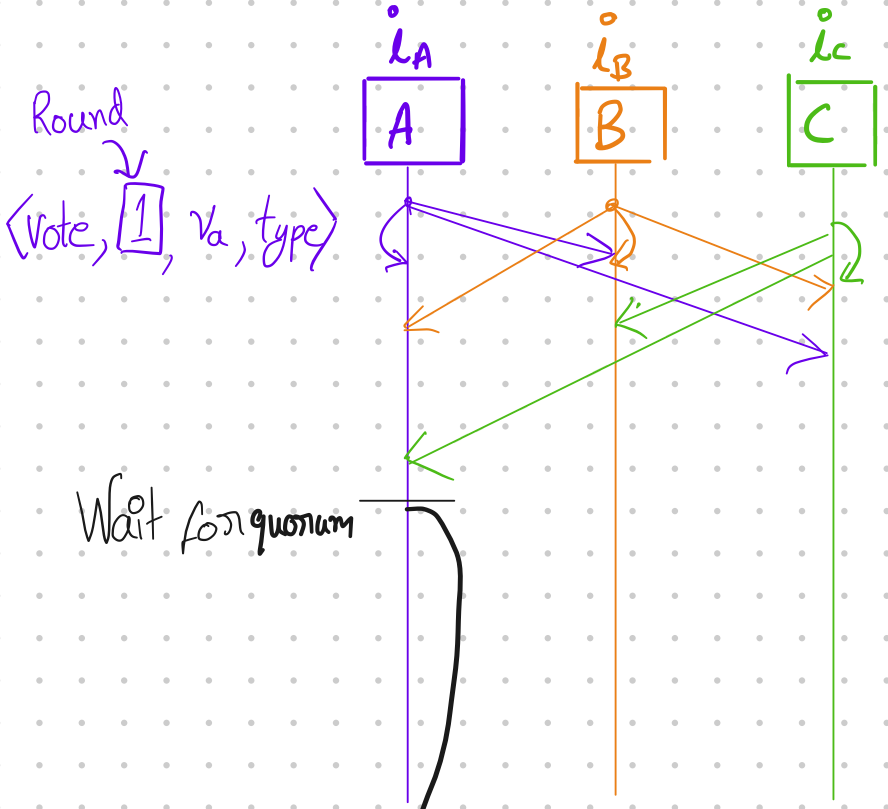
⟨1, - - -
⟨2, - - -

Paper Uses Messages Of Type 1 (1, r, ...) & Type 2 (2, r, ...)

Going to use
type 1 ≡ State
type 2 ≡ vote

→ Cases

- Quorum has the same value v
  - vote value $v_a = v$ type = D
- Else
  - vote value $v_a = ?$ type = L



→ Cases

① One ⟨vote, 1, v, D⟩ message } Some node saw evidence that a majority had the same value v

$i_A = v$

② $\geq f+1$ $\langle vote, 1, v, D \rangle$ message — A quorum saw that a majority had the same value v

Decide v

[Note, this also means for all future rounds $i_A = v$]

③ No message of type D

i_A = Randomly choose $\emptyset$ & 1

Go to Round 2

- State
- Vote

...

Why does it work

- What happens if $i_A = i_B = i_C = \emptyset$ (or 1) at round R?

Dec. $\emptyset$

- What happens if a quorum has the same input, e.g., $i_A = i_B = 1$?

1 or ?

- Claim: If node A decides in round R, then other nodes decide in round R+1. Why?

$\geq f+1$ <vote, R, v, D> message — A quorum saw that a majority had the same value v

Decide v

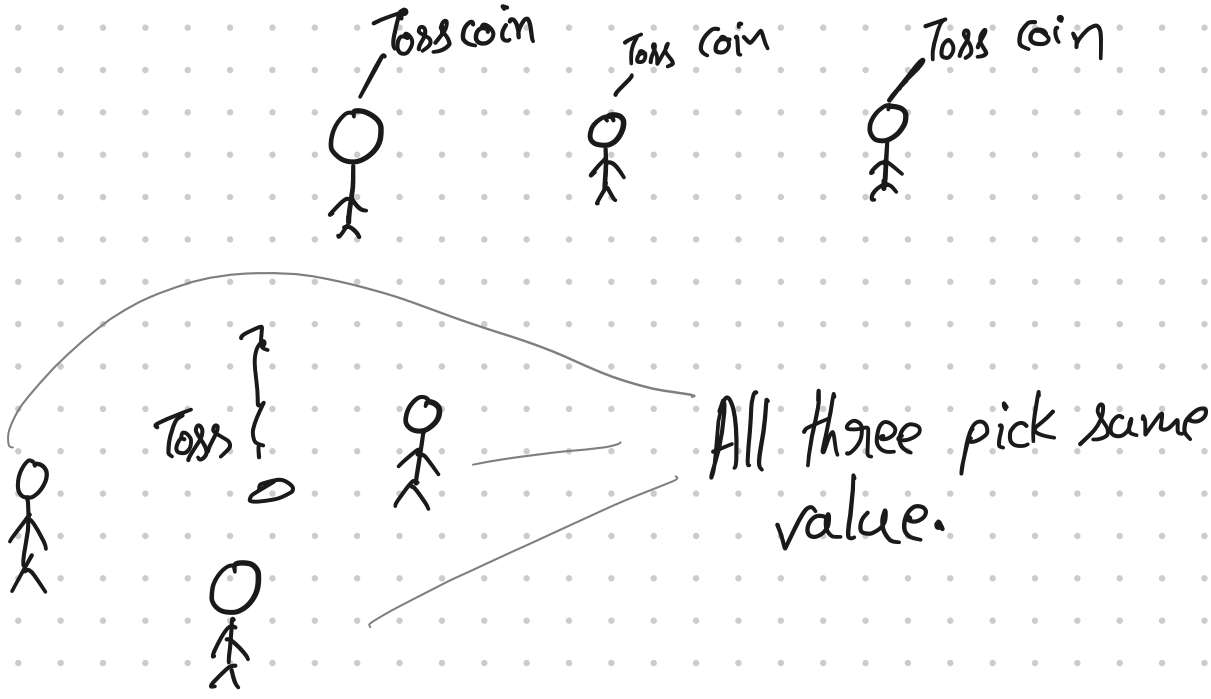
[Note, this also means for all future rounds $i_A = v$]

Optimization: Common Coin

③ No message of type D

i_A = Randomly choose 0 & 1

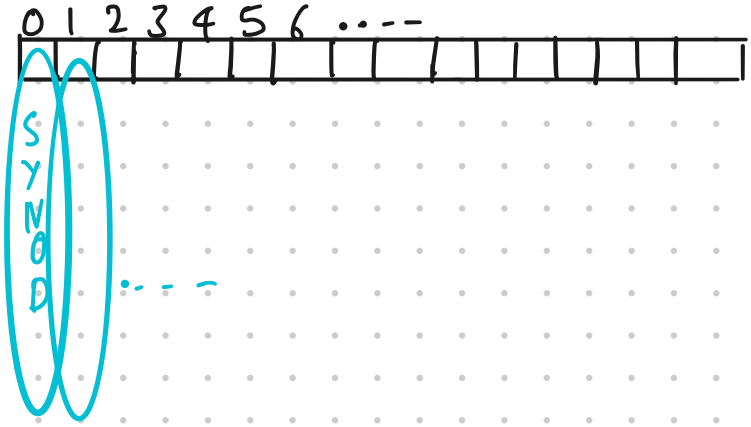
Remember: Want to get all nodes to propose the same value



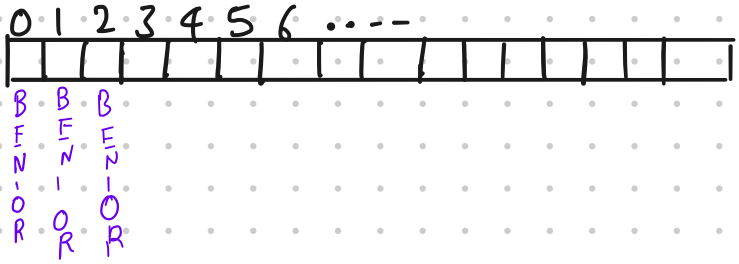
How?

Using Ben-Or In An RSM

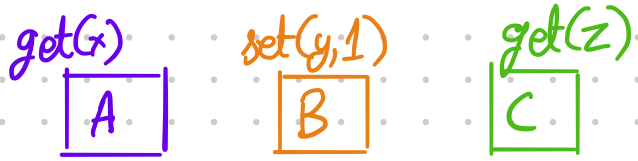
- Kind of saw this when building multipaxos



- Similar approach



Unique challenge: Binary consensus → Arbitrary commands



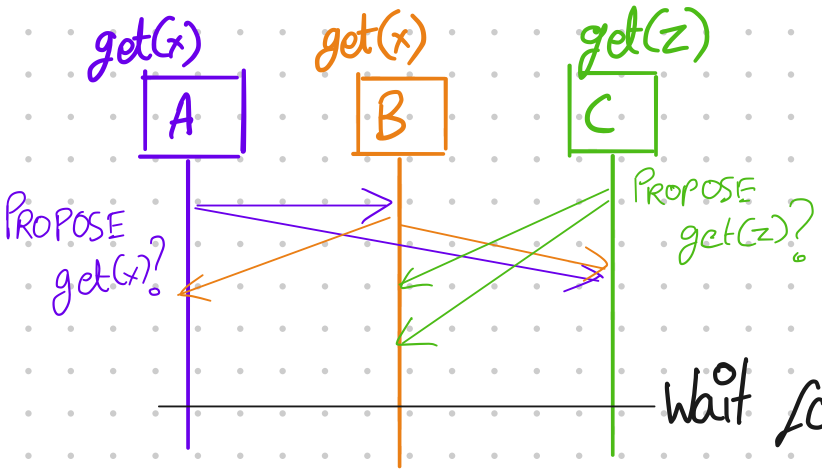
What do A, B & C use as input to Ben-Or?

Possibilities — Leader chooses command?

↳ That is what we did for multi-paxos.

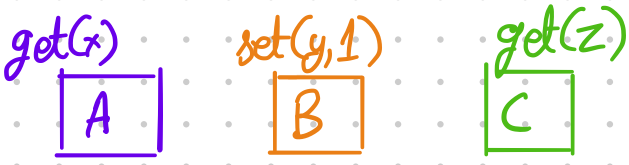
PAXIA

- USE A PROTOCOL TO CHOOSE ONE (OR TWO) PROPOSALS
  - ↳ Be wary: Why isn't this just consensus?



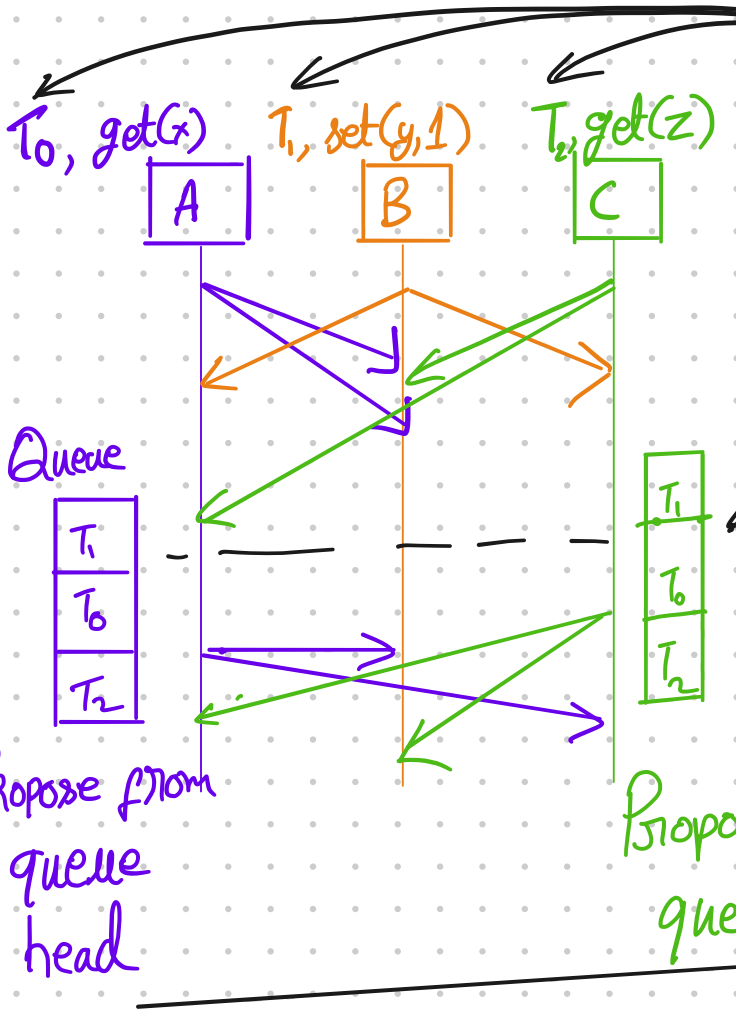
Wait for n-f (quorum)

- If quorum has some command
  - Ben Or input [for round 0] = C [or COMMON COMMAND]
- Else - Ben Or input = ⊥



How to deal with this situation?

Purpose of timestamps?



Priority queue with a consistent ordering

- Any node (A) that sees the same command C from a quorum enters Ben-Or with C.
- Other nodes use ⊥

What to do with coin toss?

Remember:

③ No message of type D

i_A = Randomly choose 0 & 1

Want coin to randomly choose b/w
c & ⊥

But, getting to this stage ⇒

(a) Some command c was agreed as a proposal by a quorum of nodes.

Why?

(b) ⩾1 node did not see a quorum support the command c

Why?

Might not know c?

Errata: Any node arriving at coin toss can recover c

Lab 4

An implementation of Rabia

↳ Mostly done.

→ Aim is to give you a chance to examine a running version

Without taking significant time from final project.

Rd ← Val ← Type

A B: vote, 1, ∅, D

C: vote, 1, 1, D

① Is it possible

A

state, R, ∅

A, B

vote, R, ∅, D

A, B,

decide(∅) t.

B

state, R, ∅

B, A

vote, R, ∅, D

B, A

decide(∅) $l_B = \emptyset$

C

state, R, 1

B ; C

vote, R, ?, 1

B, C

$l_C = \emptyset$