Distributed Systems: Randomized Consensus

Plan for today

- Some notes on the exam
- Ben-Or + Rabia
- Some notes on Lab 4

Exam

- Asynchronous
  - No bounds on message delays
  - No bounds on processing delays
  - No global clock
    - → Processes P_1, P_2 might not agree on how much time has passed

Note:

- Does not mean that processes "have no clock"

clock

receive do

Logical

... → clock+1 } Clock?
end

Core problem: Cannot distinguish b/w delay & failure

- Partial synchrony
  After some time (Δ) message and/or processing delays are bounded.

  Before ⟷ still asynchronous.

  Observe: partially synchronous → asynchronous
  Cannot result in safety violations.
  Why?

Hopefully clearer after next week.

-
Π / Set of processes.

- Impossible to design protocols, in the async/part. sync model that assume knowledge of what processes are alive.

Why?

Generally $|\Pi|-1$ or $n-1$

↑ Number of processes in configuration

Back to Scheduled Programming

- Focus since RSM lecture: Consensus

Problem: FLP impossibility (next week!)

No deterministic, fault tolerant consensus protocol in the asynch. model

Consensus: Agreement Validity Termination

How have we gone about this so far?

Asynch → Partially synchronous

Today: Deterministic → Randomized.

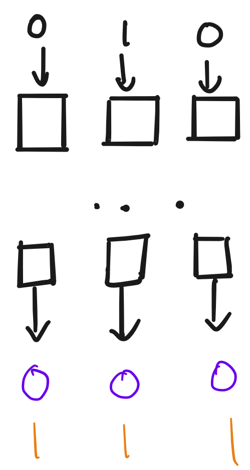
Why? ① Intellectually interesting

② Fewer assumptions

↳ Don't need to set H.B. timer; Election timer; etc.

Ben-Or

Binary consensus



Convenient problem to study
- Small set of possible values

Also features in FLP

- Validity

$\emptyset\ \emptyset\ \emptyset \dots \emptyset \longrightarrow \emptyset$

$1\ 1\ 1 \dots 1 \longrightarrow 1$

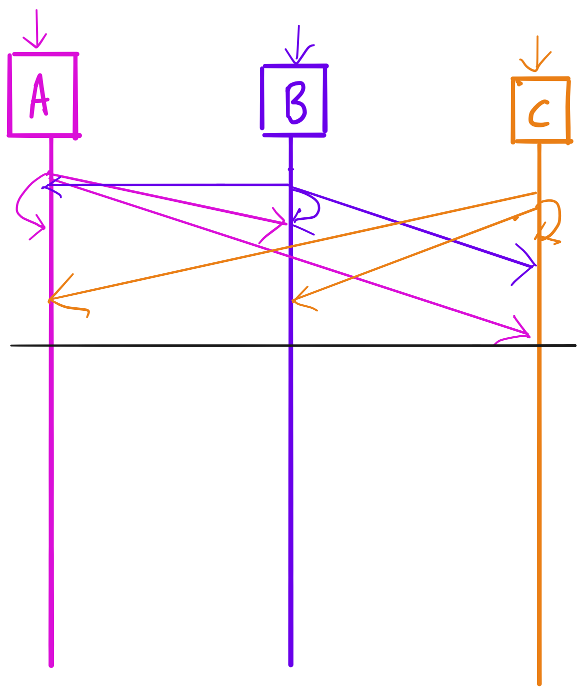
- Agreement

- Probabilistic Termination p=1 algorithm terminates

Core mechanism for ensuring agreement:

- Randomization to break ties

Round 1



⟨state, 1, p⟩ (⟨1, 1, p⟩)

wait for n-f messages.
if ~~≥~~ $\lfloor \frac{n}{2} \rfloor + 1$ messages agree on p
vote = p else vote = ?

Note

ⓐ If all nodes have the same input p then all nodes vote = p

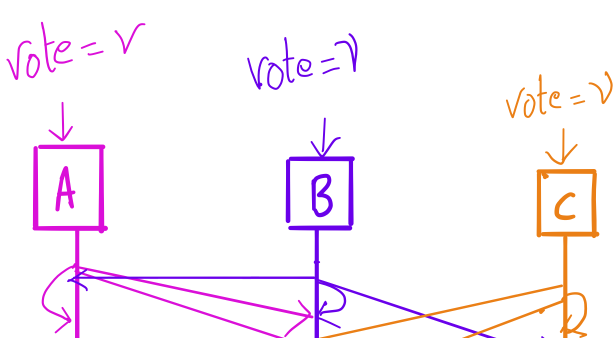
ⓑ What if $\lfloor \frac{n}{2} \rfloor + 1$ nodes have input p?

ⓒ ~~What if $\lfloor \frac{3n}{4} \rfloor + 1$ nodes have input p?~~

n → $\frac{n}{4} - 1$ not p
n → $\frac{3n}{4} + 1$ p
~~$\frac{1}{2} n + 1$~~

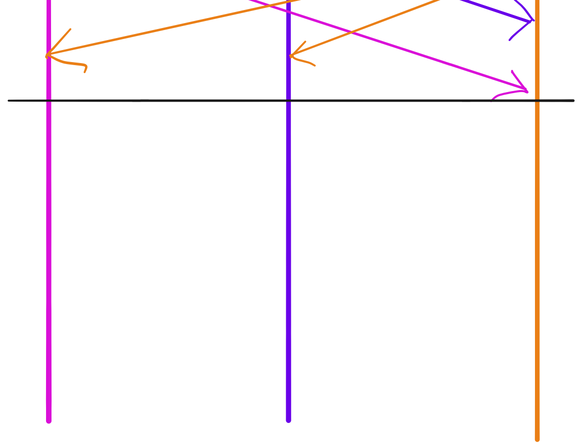
8 → 2 not p
8 → 6 p

Round 1



⟨vote, 1, v⟩ (⟨2, 1, v⟩)

- Reminder v = {1 or 0} or ?



If $v = 1$ for B
↪ Possible votes for A or C?

- Wait for $(n-f)$ votes.

If $\geq \lfloor \frac{n}{2} \rfloor + 1$ votes for non-? value v
- Decide v

If ≥ 1 vote for non-? value v
- Set proposal $= v$

If only ?-votes
- Set proposal = coin toss

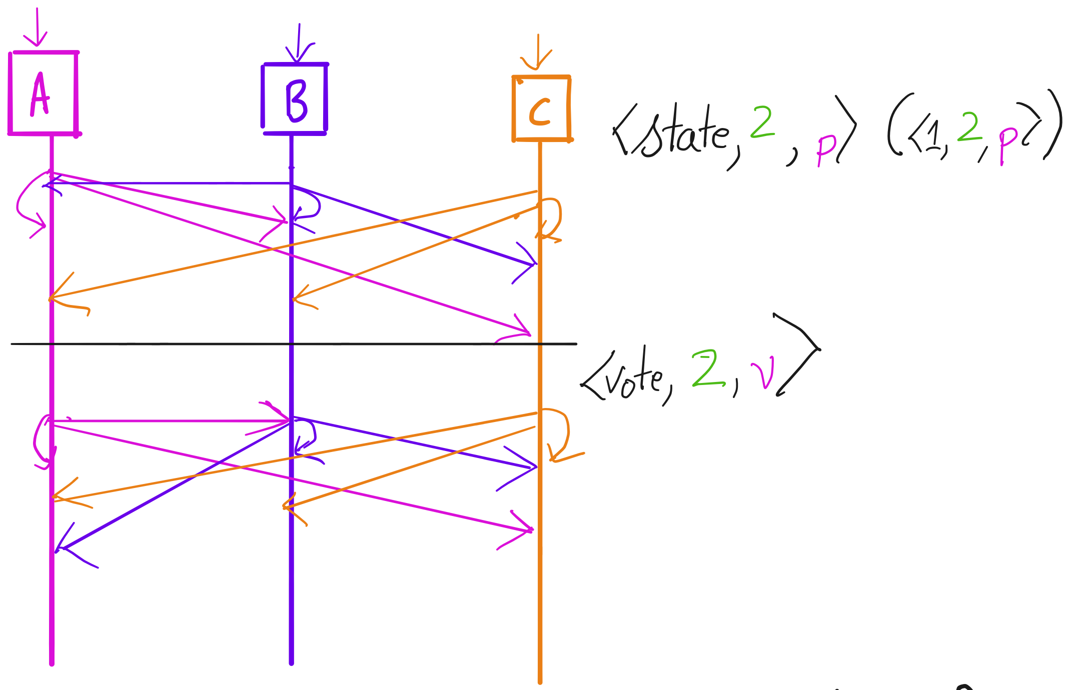
Run Ben-Or for round 2

(Q1.) What if all processes vote for non-? v?

(Q2) What if $\lfloor \frac{n}{2} \rfloor + 1$ processes vote for non-? v?

Decided the same value

Round 2



Agreement: Claim: If a process decides in round r all live processes decide by round $r+1$. Why?

How validity?

Q. What happens if a node gets only ? votes?

- Common coin?

So where we are

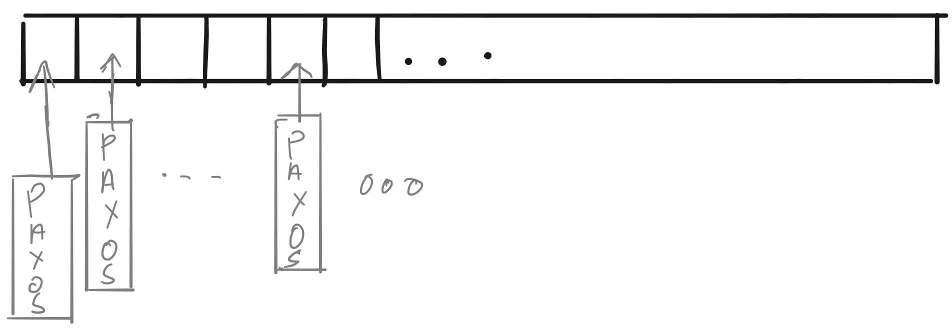
- Randomized consensus protocol for 0, 1

- Does terminate - See Aguilera & Toueg if interested.

Can we turn this into a RSM?

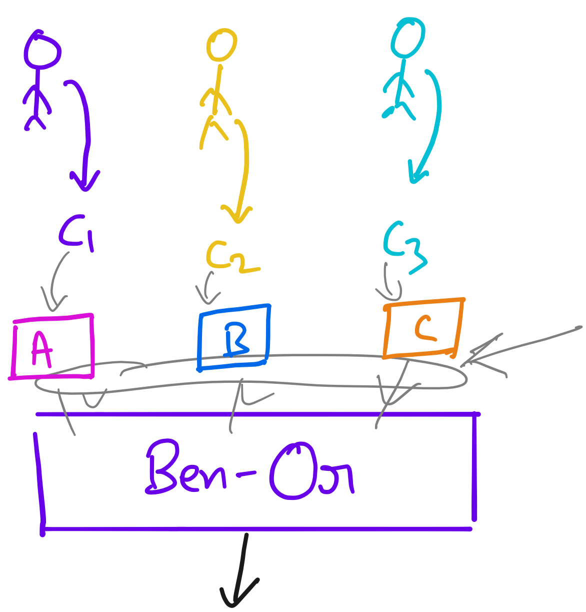
- Borrow an old idea:

Multi-Paxos



Run a Ben-Or round per slot.

New problem: Tyranny of choice



Slot 1: c_1 OR c_2 OR c_3

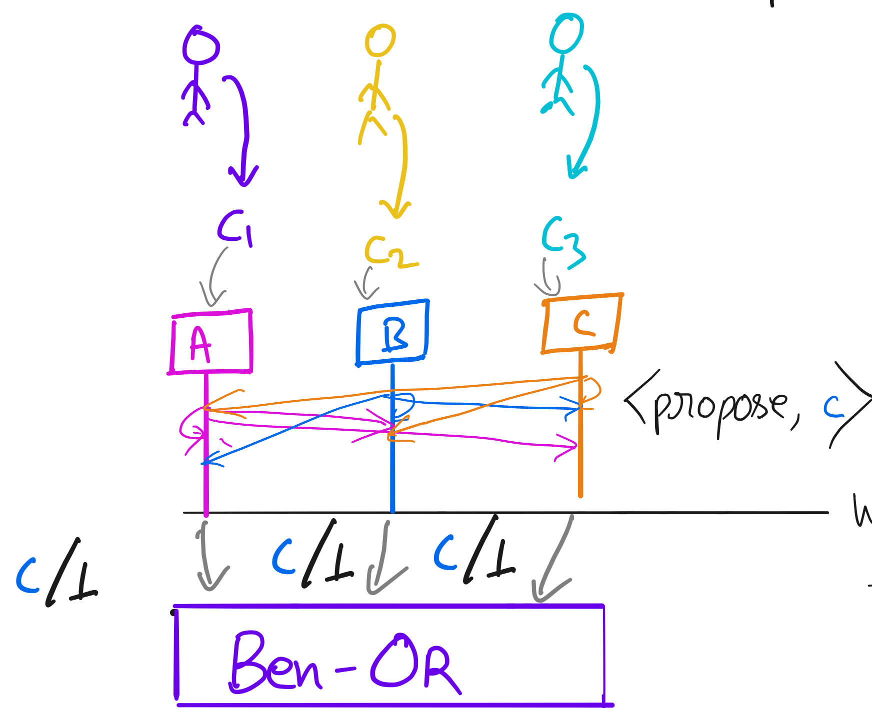
Expects Binary Input

- At most two values proposed (a)
- Possible proposals are known ahead of time (b)

Solving ⓐ: Need to reduce the number of choices

How? Quorum intersection

↳ At most one command can be proposed by a quorum



Wait for n-f proposals

- If $\geq \lfloor \frac{n}{2} \rfloor + 1$ proposal for comand c

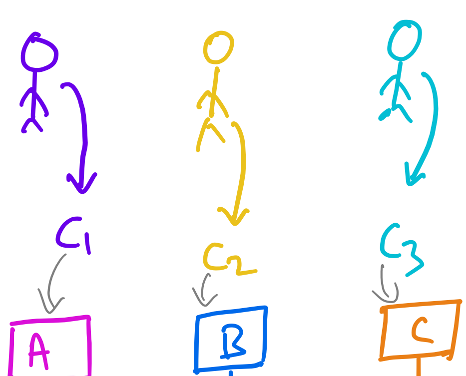
  Propose c

  Else

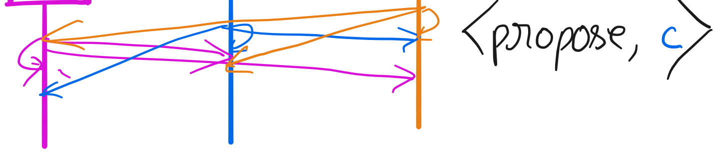
  Propose ⊥

Slot 1: c or ⊥

Sub Problem: How to make sure that nodes do not alway propose ⊥

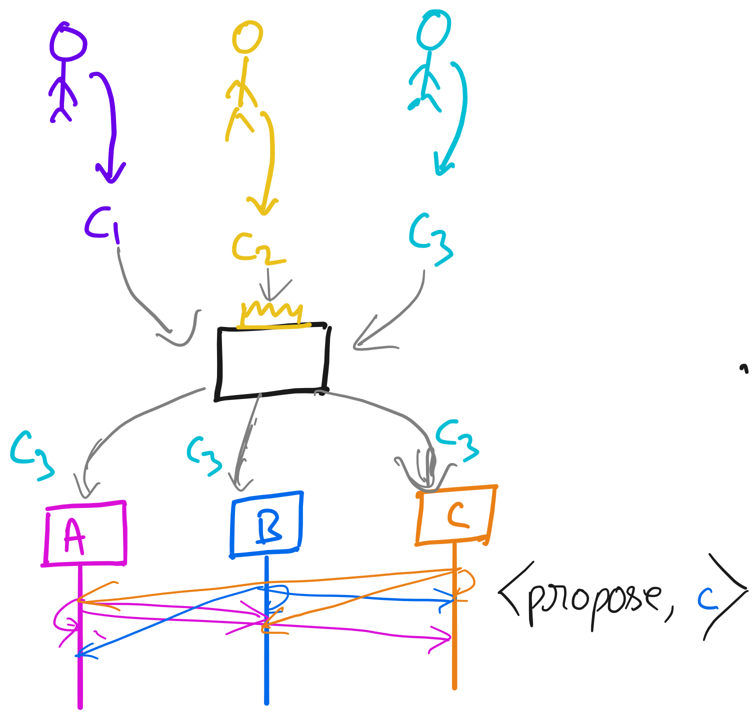


$c_1 \neq c_2 \neq c_3$



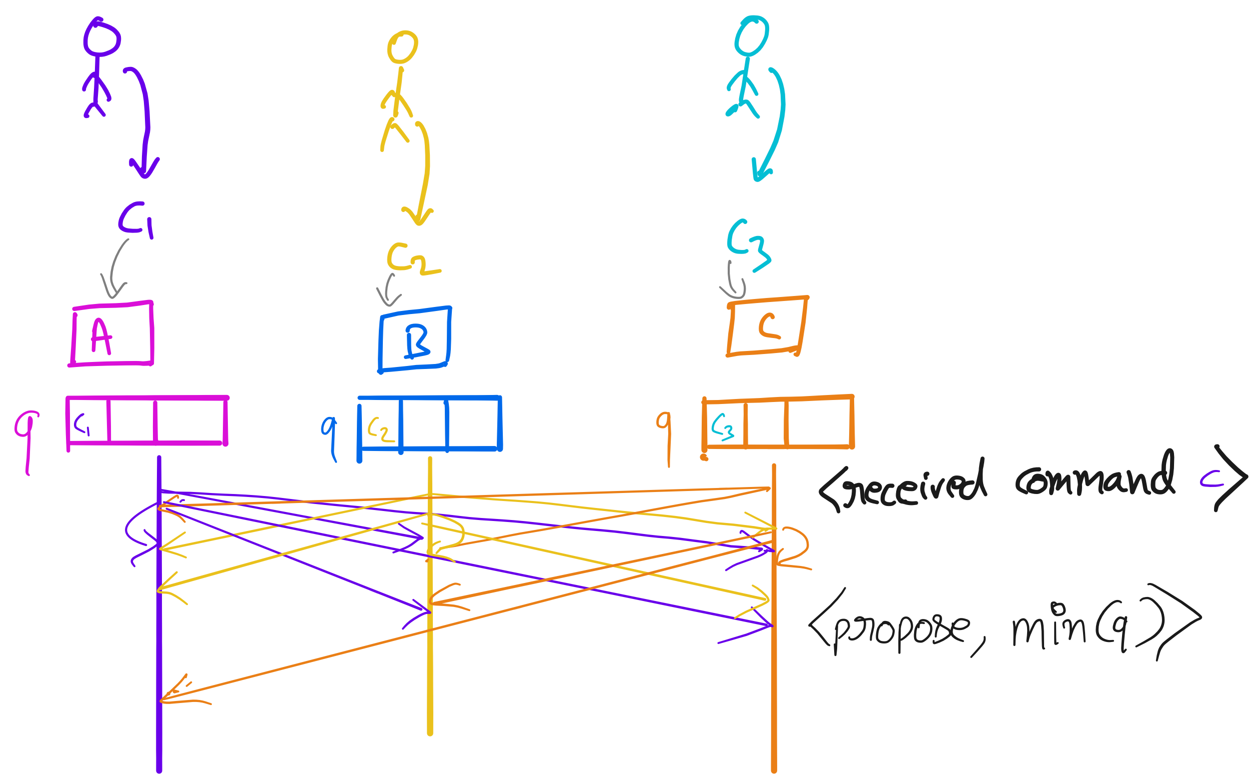
Two solutions

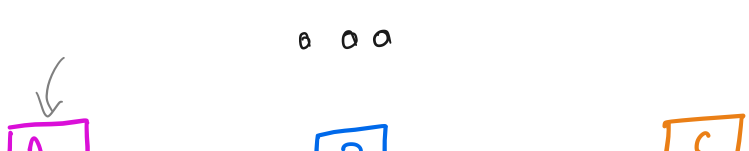
(i)

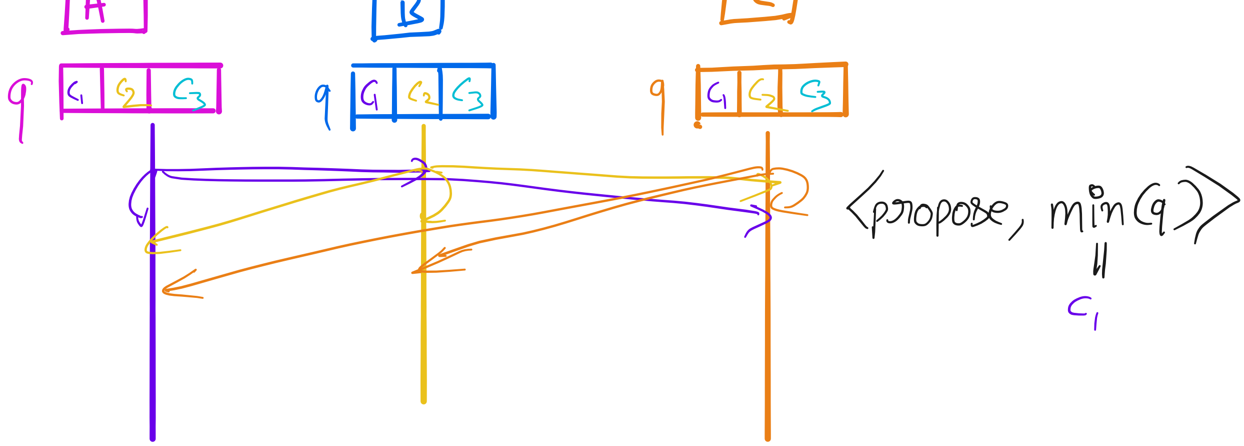


"Proxy"/
'Leader'/
...

(ii)

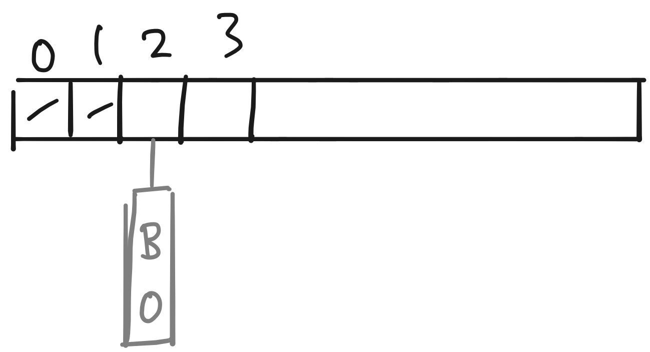




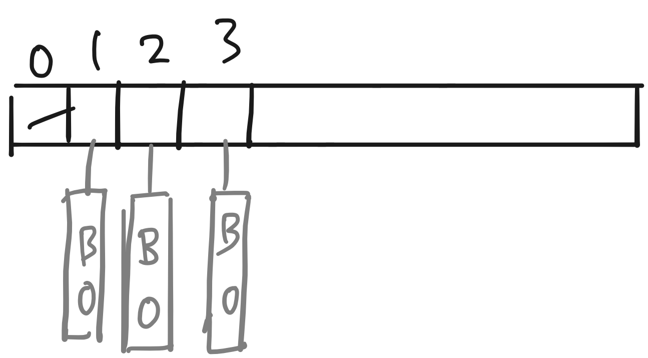


Does it work?

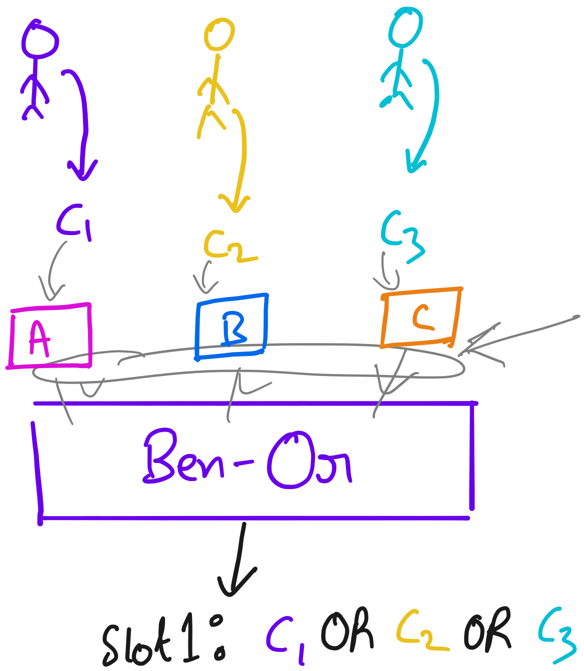
- No pipelining (1 active Ben-Or instance at a time)



- Pipelining?



Left as an exercise to the audience.



Expects Binary Input

- At most two values proposed (a) } ✓
- Possible proposals are known ahead of time (b)

(b) ← What the errata is about.

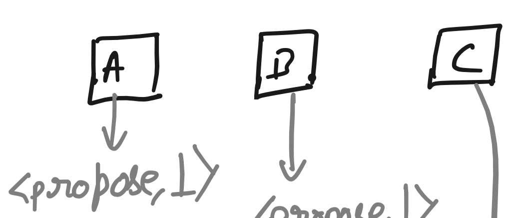
Original Idea →

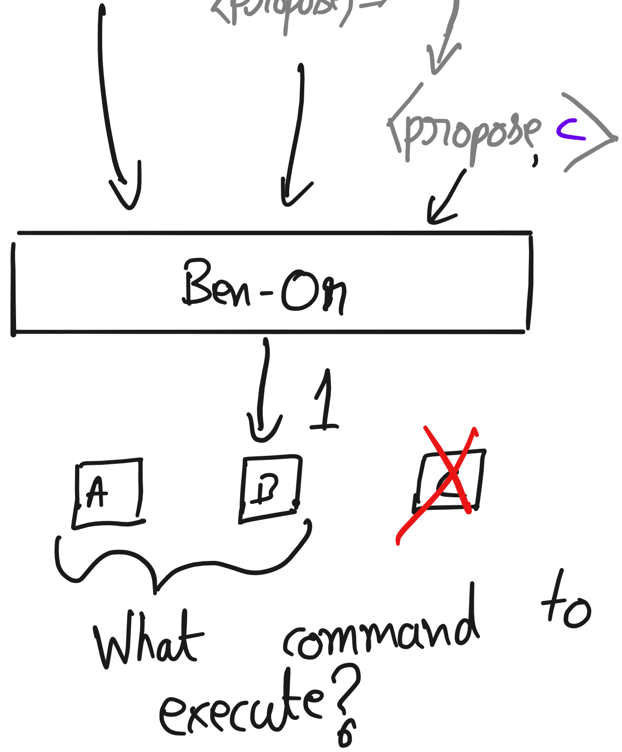
Use Ben-Or to decide 0/1

$0 \Rightarrow \bot \qquad 1 \Rightarrow c$

Problem: How to recover c?

- Quorum proposed c
  ⇒ At least one live node has c
- But no node might know that a quorum proposed c





Errata approach.

Ben-Or works on c or ⊥

↳ During coin toss

Coin must return

⊥ or c

How? Builds on the sequence of events requiring a coin toss (no votes other than ?; why nodes vote ?).

See errata.

Termination

One thing with Rabia: Weak-MVC (Ben-Or) terminates

```
wait until receiving ≥ n − f VOTE messages for round r
if non-? vote v′ appears ≥ f + 1 times in round-r VOTES then
    return v′                                                  ▷ Termination
else if non-? vote v′ appears at least once in round-r VOTE messages then
    state ← v′
```

Algorithm 2 Weak-MVC: Code for Replica i

```
When WEAK-MVC is invoked with input q and seq:
  // Exchange Stage: exchange proposals
  Send (PROPOSAL, q) to all                         ▷q is client request
  wait until receiving ≥ n − f PROPOSAL messages
  if request q appears ≥ ⌊n/2⌋ + 1 times in PROPOSALS then
      state ← 1
  else
      state ← 0
  // Randomized Binary Consensus Stage (Phase p ≥ 1)
  p ← 1                                             ▷Start with Phase 1
  while true do
      /* Round 1 */
      Send (STATE, p, state) to all                 ▷state can be 0 or 1
      wait until receiving ≥ n − f phase-p STATE messages
      if value v appears ≥ ⌊n/2⌋ + 1 times in STATES then
          vote ← v
      else
          vote ←?
      /* Round 2 */
      Send (VOTE, p, vote) to all                   ▷vote can be 0,1 or ?
      wait until receiving ≥ n − f phase-p VOTE messages
      if a non-? value v appears ≥ f + 1 times in VOTES then
          Return FINDRETURNVALUE(v)                 ▷Termination
      else if a non-? value v appears at least once in VOTES then
          state ← v
      else
          state ← COMMONCOINFLIP(p)                 ▷p-th coin flip
      p ← p + 1                                     ▷Proceed to next phase
```

What does termination mean?

ⓐ Stop participating in Ben-Or rounds?

Earlier in class →

> Claim: If a process decides in round r all live processes decide by round r+1. Why?

ⓑ Mark entry as committed?

- What should a node do if it receives propose, state or vote messages for committed slot?

Takeaways

- Please don't think of this class discussion as saying Rabia is complex, broken, or bad.

  I really like the protocol,
  found that one can implement a version in a day or so.

- Rather just a reflection on the complexity of making these systems practical.

Lab 4

- Original thought:

  Implement Rabia

Concerns

- Many pitfalls & corner cases, might require more than 2 weeks (Or seems unfair to give it less time than Raft)
- Cuts into final project time.

- For this year:
  - Lab 4 gives you a mostly complet Rabia implementation
  - Required task is to integrate it with a state machine (a lin. counter)

  - Extra Credit: Analyze a problem + propose fixes
    - ↳ Probably requires changes to the protocol

While the main lab is simple, the hope is that a Rabia implementation makes it easier to connect some of this to reality.