Distributed Systems & Randomized Consensus

Plan for today

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

Exam

- Asynchronous
  - No bounds on message delays
  - No bounds on processing delays
  - No global clock

  \[ \Rightarrow \text{Processes } p, q \text{ might not agree on how much time has passed} \]

Note:

- Does not mean that processes “have no clock”
  clock
  receive do
Core problem: Cannot distinguish b/w delay & failure

- Partial synchrony
  After some time ($\Delta$) message and/or processing delays are bounded.
  Before $\rightarrow$ still asynchronous.

Observe: partially synchronous $\rightarrow$ asynchronous

Cannot result in safety violations.

Why?

Hopefully clearer after next week.

- $\Pi$/Set of processes. in the async/part. sync model
  - Impossible to design protocols that assume knowledge of what processes are alive.
Why?

Generally \( \frac{n}{n-1} \) on \( n-1 \)

Number of processes in configuration

Back to scheduled Programming

- Focus since RSM lecture o Consensus

Problem: FLP impossibility (next week!)

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

Consensus o Agreement Validity Terminabon

How have we gone about this so far?

asynch \( \rightarrow \) Partially synchronous

Today: Deterministic \( \rightarrow \) Randomized

Why? 1) Intellectually interesting
(2) Fewer assumptions

\[ \Rightarrow \text{Don't need to set HoBo timer;}
\]
\[ \text{Election timer; etc.} \]

Ben-On

Binary consensus

\[
\setlength{
\arraycolsep}{1pt}
\begin{array}{c}
\begin{array}{ccc}
\square & \square & \square \\
\downarrow & \downarrow & \downarrow
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{ccc}
\square & \square & \square \\
\downarrow & \downarrow & \downarrow
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{c}
\square \\
\downarrow
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{c}
\square \\
\downarrow
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\begin{array}{c}
\square \\
\downarrow
\end{array}
\end{array}
\]

Convenient problem to study

- Small set of possible values

Also features in FLP

- Validity

\[ \emptyset \emptyset \emptyset \ldots \emptyset \rightarrow \emptyset \]

\[ 1 1 1 \ldots 1 \rightarrow 1 \]

- Agreement

- Probabilistic Termination \( p=1 \) algorithm terminates

Core mechanism for ensuring agreement.
Randomization to break ties

Round 1

\[\langle \text{state}, 1, p \rangle \text{ or } \langle 1, 1, p \rangle\]

Wait for \(n-f\) messages.
If \(\geq \lceil \frac{n}{2} \rceil + 1\) messages agree on \(p\), then vote = \(p\); else vote = ?

Note
(a) If all nodes have the same input \(p\) then all nodes vote = \(p\).
(b) What if \(\lceil \frac{n}{2} \rceil + 1\) nodes have input \(p\)?
(c) What if \(\lfloor \frac{3n}{4} \rfloor + 1\) nodes have input \(p\)?
If \( n = 1 \) for B
→ Possible votes for A or C?
→ Wait for \((n-f)\) votes.

If \( \geq \lceil \frac{n}{2} \rceil + 1 \) voted for non-
→ Decide 

If \( \geq 1 \) vote for non-
→ Set proposal = 
→ Set proposal = coin toss

Run Ben. On loss
Round 2

Q1. What if all processes vote for non-?

Q2. What if \( \lceil \frac{n}{2} \rceil + 1 \) processes vote for non-?
Round 2

\[ \langle \text{state}, 2, p \rangle (\langle 1, 2, p \rangle) \]

\[ \langle \text{vote}, 2, v \rangle \]

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 φ, 1

- Does terminate - see Aguilera & Toueg if interested.

Can we turn this into a RSM?
- Borrow an old idea:

**Multi-Paxos**

---

Run a Ben-Ori round per slot.

New problem: **Tyranny of choice**

---

**Ben-Ori**

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

---

Slot 1: \(c_1, c_2, c_3\)
Solving \( c \): Need to reduce the number of choices

How? Quorum intersection

\[ \implies \text{At most one command can be proposed by a quorum} \]

\[ \langle \text{propose, } c \rangle \]

\[
\begin{align*}
& \text{Wait for } n-f \text{ proposals,} \\
& \quad \text{If } \geq \left\lceil \frac{n}{2} \right\rfloor + 1 \text{ proposal for command } c \\
& \quad \text{Propose } c \\
& \quad \text{Else} \\
& \quad \text{Propose } \bot
\end{align*}
\]

Sub Problem: How to make sure that nodes do not always propose \( \bot \)

\[ C_1 \neq C_2 \neq C_3 \]
Two solutions

1. "Proxy" / Leader...

2. received command

<propose, c>
Does it work?

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

Pipelining?

Left as an exercise to the audience.
Expect's Binary Input
- At most two values proposed
- Possible proposals are known ahead of time

*(b)* What the errata is about.

**Original Idea**
Use Ben-On to decide 0/1

0 ⇒ 1  1 ⇒ 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-On works on $c$ on $I$.

During coin toss

Coin must return

$I$ 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's Weak-MVC (Ben-On) terminates

What does termination mean?
@ Stop participating in Ben-On rounds?

Earlier in class →

Claim: If a process decides in round \( n \) all live processes decide by round \( n+1 \). Why?

(a) 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 & common cases, might require more than 2 weeks (or seems easier to give it less time than last)

- Cuts into final project time.

- For this year's

  - Lab 4 gives you a mostly complete Rabin implementation

  - Required task is to integrate it with a state machine (a lin. counter)

- Extra Credit: Analyze a problem & propose fixes

  \[ \Rightarrow \text{ Probably requires changes to the protocol} \]
While the main lab is simple, the hope is that a Rabia implementation makes it easier to connect some of this to reality.