Paxos week!

Doug Woos

Logistics notes

Next Monday: International Workers' Day

- No in-class lecture
- Will record a video lecture
- Please watch by next Wednesday!

Lab 2b due Wednesday

Problem Set 2 due Friday

- *Typeset, short* answers, please!

Lab 1, logical clocks discussion grades are out

Paxos

(deck based on slides from Lorenzo Alvisi)

Safe Replication?

- Suppose using primary/hot standby replication
- How can we tell if primary has failed versus is slow? (if slow, might end up with two primaries!)
- FLP: impossible for a deterministic protocol to guarantee consensus in bounded time in an asynchronous distributed system (even if no failures actually occur and all messages are delivered)

2PC vs. Paxos?

- Two phase commit: blocks if coordinator fails after the prepare message is sent, until the coordinator recovers
- Paxos: non-blocking as long as a majority of participants are alive, provided there is a sufficiently long period without further failures

The Part-Time Parliament

- Parliament determines laws by passing sequence of numbered decrees
- Legislators can leave and enter the chamber at arbitrary times
- No centralized record of approved decrees- instead, each legislator carries a ledger

Government 101

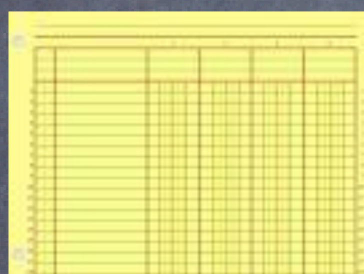
- No two ledgers contain contradictory information
- If a majority of legislators were in the Chamber and no one entered or left the Chamber for a sufficiently long time, then
  - any decree proposed by a legislator would eventually be passed
  - any passed decree would appear on the ledger of every legislator

Government 102

- Paxos legislature is non-partisan, progressive, and well-intentioned
- Legislators only care that something is agreed to, not what is agreed to
- To deal with Byzantine legislators, see Castro and Liskov, SOSP 99

Supplies

Each legislator receives



ledger

lots of messengers

scratch paper

pen with indelible ink

hourglass

Back to the future

- A set of processes that can propose values
- Processes can crash and recover
- Processes have access to stable storage
- Asynchronous communication via messages
- Messages can be lost and duplicated, but not corrupted

The Game: Consensus

SAFETY

- Only a value that has been proposed can be chosen
- Only a single value is chosen
- A process never learns that a value has been chosen unless it has been

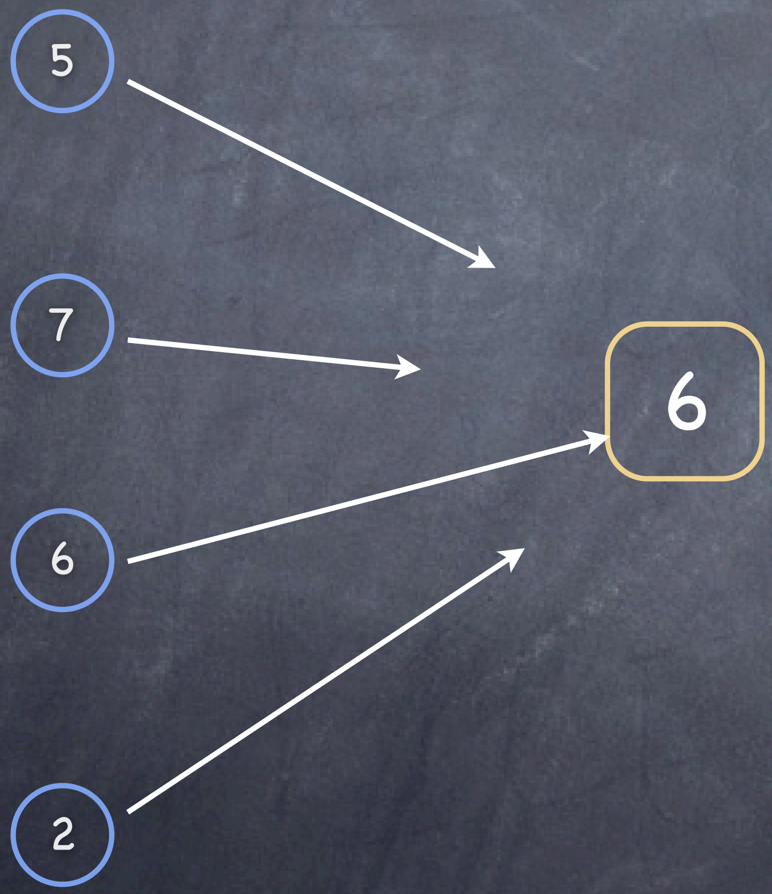
LIVENESS

- Some proposed value is eventually chosen
- If a value is chosen, a process eventually learns it

The Players

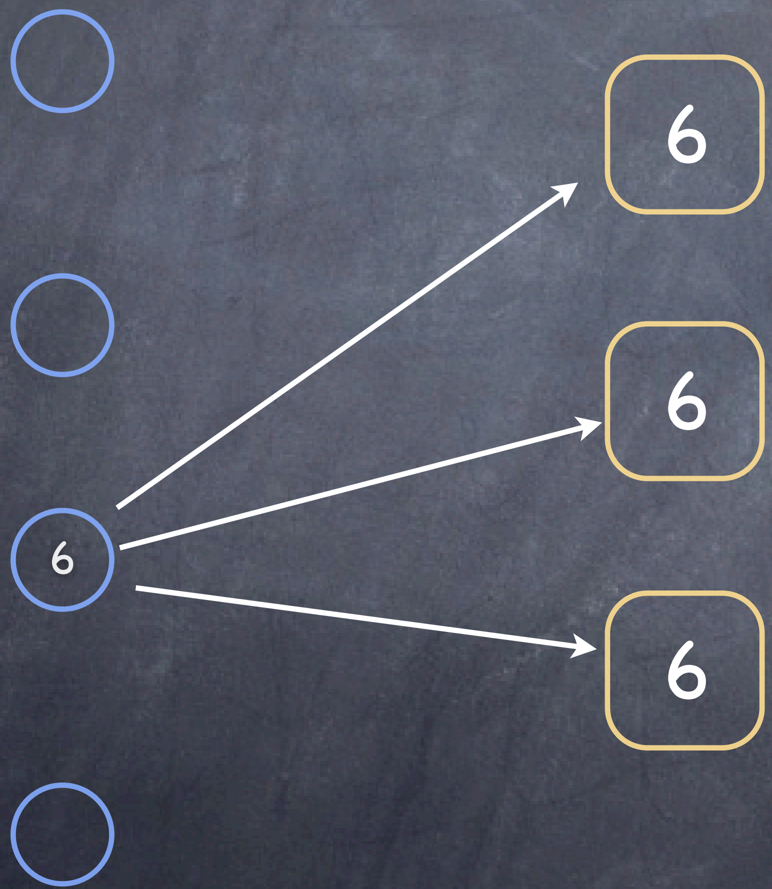
- Proposers
- Acceptors
- Learners

Choosing a value



Use a single acceptor

What if the acceptor fails?



6 is chosen!

- Choose only when a "large enough" set of acceptors accepts
- Using a majority set guarantees that at most one value is chosen

Accepting a value

- Suppose only one value is proposed by a single proposer.
- That value should be chosen!
- First requirement:

P1: An acceptor must accept the first proposal that it receives

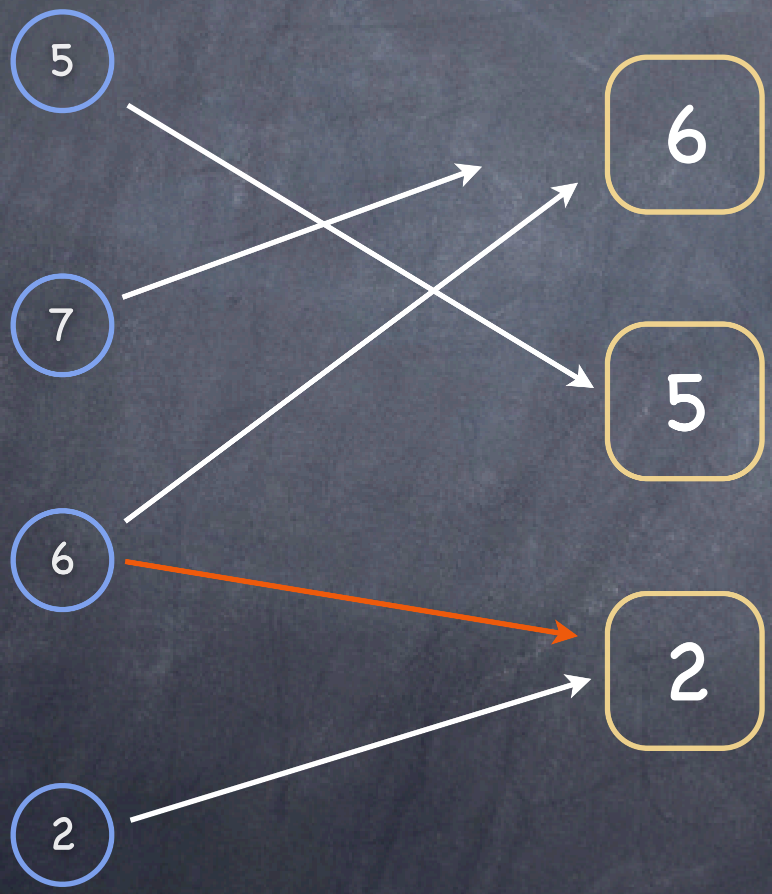
Accepting a value

- Suppose only one value is proposed by a single proposer.
- That value should be chosen!
- First requirement:

  P1: An acceptor must accept the first proposal that it receives

- ...but what if we have multiple proposers, each proposing a different value?

P1 + multiple proposers



No value is chosen!

Handling multiple proposals

- Acceptors must (be able to) accept more than one proposal
- To keep track of different proposals, assign a natural number to each proposal
  - A proposal is then a pair (psn, value)
  - Different proposals have different psn
  - A proposal is chosen when it has been accepted by a majority of acceptors
  - A value is chosen when a single proposal with that value has been chosen

Choosing a unique value

- We need to guarantee that all chosen proposals result in choosing the same value
- We introduce a second requirement (by induction on the proposal number):

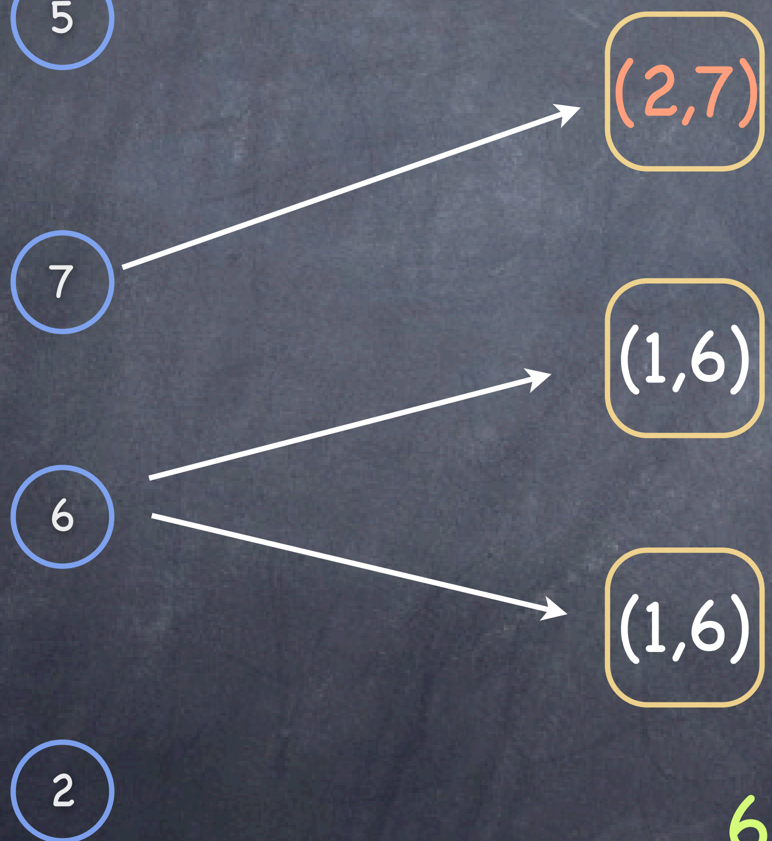
  P2. If a proposal with value v is chosen, then every higher-numbered proposal that is chosen has value v

  which can be satisfied by:

  P2a. If a proposal with value v is chosen, then every higher-numbered proposal accepted by any acceptor has value v

What about P1?

How does it know
it should not accept?



6 is chosen!

- Do we still need P1?

  YES, to ensure that *some* proposal is accepted

- How well do P1 and P2a play together?

  Asynchrony is a problem...

Another take on P2

- Recall P2a:

  If a proposal with value v is chosen, then every higher-numbered proposal accepted by any acceptor has value v

We strengthen it to:

  P2b: If a proposal with value v is chosen, then every higher-numbered proposal issued by any proposer has value v

Implementing P2 (I)

P2b: If a proposal with value v is chosen, then every higher-numbered proposal issued by any proposer has value v

Suppose a proposer p wants to issue a proposal numbered n. What value should p propose?

- If (n',v) with $n' < n$ is chosen, then in every majority set S of acceptors at least one acceptor has accepted (n',v)...
- ...so, if there is a majority set S where no acceptor has accepted (or will accept) a proposal with number less than n, then p can propose any value

Implementing P2 (II)

P2b: If a proposal with value v is chosen, then every higher-numbered proposal issued by any proposer has value v

What if for all S some acceptor ends up accepting a pair (n',v) with $n' < n$?

Claim: p should propose the value of the highest numbered proposal among all accepted proposals numbered less than n

Proof: By induction on the number of proposals issued after a proposal is chosen

Implementing P2 (III)

P2b: If a proposal with value v is chosen, then every higher-numbered proposal issued by any proposer has value v

Achieved by enforcing the following invariant

P2c: For any v and n, if a proposal with value v and number n is issued, then there is a set S consisting of a majority of acceptors such that either:

- no acceptor in S has accepted any proposal numbered less than n, or
- v is the value of the highest-numbered proposal among all proposals numbered less than n accepted by the acceptors in S