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
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 Players

• Proposers

• Acceptors

• Learners
Overview

• Paxos is a protocol that enables replicated state machines
  • Consensus on command log (comprising of instances)

• Workflow terminology of a single Paxos instance:
  • Propose —> Accept —> Chosen —> Learnt
  • But have to “prepare” before issuing a proposal
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
Choosing a value

Use a single acceptor
What if the acceptor fails?

- 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
• ...but what if we have multiple proposers, each proposing a different value?
P1 + multiple proposers

No value is chosen!
Handling multiple proposals

- Acceptors must 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

P2. If a proposal with value $V$ is chosen, then every higher-numbered proposal that is chosen has value $V$. 
It’s up to the Acceptors!

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

We strengthen it to:

P2a. If a proposal with value $v$ is chosen, then every higher-numbered proposal accepted by any acceptor has value $v$
What about P1?

- 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...

How does $a_1$ know it should not accept?

- $a_1$ chooses (2,7)
- $a_2$ chooses (1,6)
- $a_3$ chooses (1,6)

6 is chosen!
It’s up to the Proposers!

Recall P2a:

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 $v$
What to propose

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

Suppose p wants to issue a proposal numbered n.

• If p can be certain that no proposal numbered n’ < n has been chosen then p can propose any value!

  • If a proposal numbered n’ < n has been chosen, then it has been accepted by a majority set S
  • Any majority set S’ must intersect S
  • If p can find one S’ in which no acceptors has accepted a proposal numbered n’ < n, then no such proposal can have yet been chosen!
  • If no such S’, a proposal numbered n’ < n may have been chosen...
  • Then what?
What to propose

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

Suppose p wants to issue a proposal numbered n.

• If p can be certain that no proposal numbered n’ < n has been chosen then p can propose any value!

• If not, p should propose the chosen value. But how?
  • Sometimes it cannot tell whether a proposal/value has been chosen
  • p should propose the highest numbered proposal among all proposals, numbered less than n, accepted by some majority set S
Example
It’s up to an invariant!

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 $S$
P2c in action

- $v$ is the value of the highest-numbered proposal among all proposals numbered less than $n$ and accepted by the acceptors in $S$
**P2c in action**

- $v$ is the value of the highest-numbered proposal among all proposals numbered less than $n$ and accepted by the acceptors in $S$

The invariant is violated
Future telling?

• $p$ must learn the highest-numbered proposal with number less than $n$, if any, that has been or will be accepted by each acceptor in some majority of acceptors.

• Avoid predicting the future by extracting a promise from a majority of acceptors not to subsequently accept any proposals numbered less than $n$. 
The proposer’s protocol (I)

- A proposer chooses a new proposal number $n$ and sends a request to each member of some set of acceptors, asking it to respond with:
  
a. A promise never again to accept a proposal numbered less than $n$, and  
b. The accepted proposal with highest number less than $n$ if any.

...call this a prepare request with number $n$
The proposer’s protocol (II)

• If the proposer receives a response from a majority of acceptors, then it can issue a proposal with number $n$ and value $v$, where $v$ is
  • the value of the highest-numbered proposal among the responses, or
  • is any value selected by the proposer if responders returned no proposals

A proposer issues a proposal by sending, to some set of acceptors, a request that the proposal be accepted. ...call this an accept request.
The acceptor’s protocol

- An acceptor receives prepare and accept requests from proposers.
  - It can always respond to a prepare request.
  - It can respond to an accept request, accepting the proposal, iff it has not promised not to, e.g.

P1a: An acceptor can accept a proposal numbered $n$ iff it has not responded to a prepare request having number greater than $n$

...which subsumes P1.
Small optimizations

• If an acceptor receives a prepare request $r$ numbered $n$ when it has already responded to a prepare request for $n’ > n$, then the acceptor can simply ignore $r$.

...so an acceptor needs only remember the highest numbered proposal it has accepted and the number of the highest-numbered prepare request to which it has responded.
Learning chosen values (I)

Once a value is chosen, learners should find out about it. Many strategies are possible:

i. Each acceptor informs each learner whenever it accepts a proposal.

ii. Acceptors inform a distinguished learner, who informs the other learners

iii. Something in between (a set of not-quite-as-distinguished learners)
Questions

• What are the liveness properties of Paxos? Why is Paxos not considered live?
Question

• What do you do when nodes fail? How is Paxos robust to failures?
Question

• Are there any advantages/disadvantages to having a designated leader?