Paxos

Seif Haridi
Single Value Uniform Consensus

- **Validity**
  - Only proposed values may be *decided*

- **Uniform Agreement**
  - No two processes decide *different* values

- **Integrity**
  - Each process can decide a value at most *once*

- **Termination**
  - Every process *eventually* decides a value
Single Value Uniform Consensus

- (Uniform) Consensus is not solvable in the Fail-Silent model (asynchronous system model)

- Given a fixed set of deterministic processes there is no algorithm that solves consensus in the asynchronous model if one process may crash and stop

- There are some infinite executions that where processes are not able to decide on a single value

- Fischer, Lynch and Patterson FLP result
Assumptions

- Partially synchronous system
- Fail-noisy model
- Message duplication, loss, re-ordering
Importance

- Paxos is arguably the most important algorithm in distributed computing
- This presentation follows the paper “Paxos Made Simple” (Lamport, 2001)
High Level View of Paxos

• Elect a single proposer using $\Omega$
  • Proposer imposes its proposal to everyone
  • Everyone decides

• Problem with $\Omega$
  • Several processes might initially be proposers (contention)
High Level View of Paxos

- Elect a single proposer using $\Omega$
  - Proposer imposes its proposal to everyone
  - Everyone decides
- Problem with $\Omega$
  - Several processes might initially be proposers (contention)
- Solution is **Abortable Consensus**
  - Processes attempt to impose their proposals
  - Might abort if there is contention (safety) (multiple proposers)
  - $\Omega$ ensures eventually 1 proposer succeeds (liveness)
PAXOS ALGORITHM
Terminology

- **Proposers**
  - Will attempt imposing their *proposal* to set of acceptors

- **Acceptors**
  - May *accept* values issued by proposers

- **Learners**
  - Will *decide* depending on acceptors acceptances

- Each process plays all 3 roles in classic setting
Naïve Approach

- Centralized solution
  - Proposer sends value to a central acceptor
  - Acceptor decides first value it gets
- Problem
  - Acceptor is a single-point of failure
Abortable Consensus

- Decentralizes, i.e. proposers talks to set of acceptors

- Tolerate failures, i.e. acceptors might fail (needs only a majority of acceptors surviving)

- Proposers might fail to impose its proposal (aborts)
Decentralization & Fault-tolerance

- Quorum approach
  - Each proposer tries to impose its value $v$ on the set of acceptors
  - If majority of acceptors accept $v$, then $v$ is chosen
  - Learners try to decide the chosen value
**Ballot (round) Array (table)**

- Describes the **state of the acceptors** at various rounds
- Each raw describes one round
- Each acceptor’s state of $a_i$ initially $\bot$

<table>
<thead>
<tr>
<th>Round</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = 5$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n=2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n=1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n=0$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
</tr>
</tbody>
</table>
When to accept

- Ideally, there will be a single proposer
  - Should at least provide obstruction-free progress
    - Obstruction-free = if a single proposer executes without interference (contention) it makes progress

- Suggested invariant
  - P1. An acceptor **accepts** first proposal it receives
**Attempt**

- P1. An acceptor *accepts* first proposal it receives
- Problem
  - Impossible to later tell what was chosen
  - Forced to allow **restarting**! Let acceptors change their minds!

```
<table>
<thead>
<tr>
<th>p1</th>
<th>prop(red)</th>
<th>accept (red)</th>
</tr>
</thead>
<tbody>
<tr>
<td>p2</td>
<td></td>
<td>accept (red)</td>
</tr>
<tr>
<td>p3</td>
<td></td>
<td>accept (red)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>p4</td>
<td></td>
<td>accept (blue)</td>
</tr>
<tr>
<td>p5</td>
<td>prop(blue)</td>
<td>accept (blue)</td>
</tr>
</tbody>
</table>
```

**Red:** p₁, p₂, p₃

**Blue:** p₄, p₅

Any value chosen?
Ballot (round) Array (table)

- Two proposers $p_1$ and $p_2$ that propose red and blue
- But $a_3$ crashes

<table>
<thead>
<tr>
<th>Round</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = 5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n=2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n=1$</td>
<td>red</td>
<td>red</td>
<td>red</td>
<td>blue</td>
<td>blue</td>
</tr>
<tr>
<td>$n=0$</td>
<td>⊥</td>
<td></td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
</tr>
</tbody>
</table>
Ballot (round) Array (table)

- Two proposers p1 and p2 that propose red and blue
- But $a_3$ crashes

<table>
<thead>
<tr>
<th>Round</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>n = 5</td>
<td></td>
<td></td>
<td>red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n = 2</td>
<td></td>
<td></td>
<td></td>
<td>blue</td>
<td>blue</td>
</tr>
<tr>
<td>n = 1</td>
<td>red</td>
<td>red</td>
<td>blue</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n = 0</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
</tr>
</tbody>
</table>
Enabling Restarting

- Proposer can try to propose again
  - Distinguish proposals with unique sequence number
  - Often called ballot number
  - Monotonically increasing

- Implementation with n nodes
  - Process 1 uses seq: 1, n+1, 2n+1, 3n+1, ...
  - Process 2 uses seq: 2, n+2, 2n+2, 3n+2, ...
  - Process 3 uses seq: 3, n+3, 2n+3, 3n+3, ...

- Or...
  - Pair of values: (local clock or logical clock, local identifier)
  - Lexicographic order: if clock collides, choose highest pid
Problem with restart

Learners might decide red  Learners might decide blue
**Ballot (round) Array (table)**

- p1 proposes (1, red) and p2 proposes (3, blue)
- But a₁ and a₁ crashed

<table>
<thead>
<tr>
<th>Round</th>
<th>a₁</th>
<th>a₂</th>
<th>a₃</th>
<th>a₄</th>
<th>a₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>n = 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n = 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n = 3</td>
<td></td>
<td></td>
<td>blue</td>
<td>blue</td>
<td>blue</td>
</tr>
<tr>
<td>n=2</td>
<td>red</td>
<td>red</td>
<td>red</td>
<td>⊥</td>
<td>⊥</td>
</tr>
<tr>
<td>n=1</td>
<td>red</td>
<td>red</td>
<td>red</td>
<td>⊥</td>
<td>⊥</td>
</tr>
<tr>
<td>n=0</td>
<td>⊥</td>
<td></td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
</tr>
</tbody>
</table>
Ensuring Agreement

- Problem (previous slide):
  - If restarting allowed,
    - Majority may first accept red
    - Majority may later accept blue
- Solve it by enforcing:
  - P2. If proposal \((n,v)\) is chosen, every higher numbered proposal chosen has value \(v\)
Birds-eye View

- Abortable Consensus in a nutshell
  - P1. An acceptor accepts first proposal it receives
  - P2. If v is chosen, every higher proposal chosen has value v

- Handwaving
  - P1 ensures obstruction-free progress and validity
  - P2 ensures agreement
  - Integrity trivial to implement
    - Remember if chosen before, at most choose once
Attempt

- P2. If $v$ is chosen, every higher proposal chosen has value $v$
  - How to implement it?
- P2a. If $v$ is chosen, every higher proposal accepted has value $v$
- Lemma
  - P2a => P2
Problem

- Recall
  - P1. An acceptor accepts first proposal it receives
  - P2a. If v is chosen, every higher proposal accepted has value v
- Problem: we cannot prevent an acceptor from accepting higher value proposal

```
<table>
<thead>
<tr>
<th>Propose</th>
<th>Accept</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_1</td>
<td>accept(1, red)</td>
</tr>
<tr>
<td>p_2</td>
<td>accept(1, red)</td>
</tr>
<tr>
<td>p_3</td>
<td>accept(1, red)</td>
</tr>
<tr>
<td>p_4</td>
<td>Accept(3, blue)</td>
</tr>
<tr>
<td>p_5</td>
<td>Propose(3, blue) xor Accept(3, blue)</td>
</tr>
</tbody>
</table>
```

S. Haridi, KTHx ID2203.1x
Solution

- Strengthen P2a
  - P2b. If $v$ is chosen, every higher proposal issued has value $v$

- If obeyed, solves problem

```
Not allowed anymore.
p_1 propose(1, red) accept(1, red) red chosen
       accept(1, red)
       accept(1, red)

p_2

p_3

p_4 accept(5, blue)

p_5 propose(5, blue) accept(5, blue)
```
Ballot (round) Array (table)

- p1 proposes (1, red) and p2 proposes (3, blue)
- But a₂ and a₃ crashed before p2 proposes (3, blue)

<table>
<thead>
<tr>
<th>Round</th>
<th>a₁</th>
<th>a₂</th>
<th>a₃</th>
<th>a₄</th>
<th>a₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>n = 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n = 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n = 3</td>
<td></td>
<td></td>
<td>red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=2</td>
<td>red</td>
<td>red</td>
<td>red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=1</td>
<td>red</td>
<td>red</td>
<td>red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=0</td>
<td>⊥</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Ballot (round) Array (table)

- p1 proposes (1, red) and p2 proposes (3, blue)
- At round 3 p2 has to issue (1, red)

<table>
<thead>
<tr>
<th>Round</th>
<th>a₁</th>
<th>a₂</th>
<th>a₃</th>
<th>a₄</th>
<th>a₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>n = 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n = 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n = 3</td>
<td></td>
<td></td>
<td>red</td>
<td>red</td>
<td>red</td>
</tr>
<tr>
<td>n=2</td>
<td>red</td>
<td>red</td>
<td>red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=1</td>
<td>red</td>
<td>red</td>
<td>red</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
P2 Preserved

• P2. If v is chosen, every higher proposal chosen has value v
• P2a. If v is chosen, every higher proposal accepted has value v
• P2b. If v is chosen, every higher proposal issued has value v

• Lemma
  • P2b => P2a

• Recall P2a => P2.
  • Thus P2b => P2
Main Lemma

- P2c. If any proposal \((n,v)\) is issued, there is a majority set \(S\) of acceptors such that either
  - (a) no one in \(S\) has accepted any proposal numbered less than \(n\)
  - (b) \(v\) is the value of the highest proposal among all proposals less than \(n\) accepted by acceptors in \(S\)

- Lemma: P2c => P2b
Main lemma

- (a) no one in $S$ has *accepted* any proposal number > 3
- p2 issues (3, blue) at round 3

<table>
<thead>
<tr>
<th>Round</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = 5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n = 4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n = 3$</td>
<td>red</td>
<td>red</td>
<td>blue</td>
<td>blue</td>
<td>blue</td>
</tr>
<tr>
<td>$n = 2$</td>
<td>red</td>
<td>red</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
</tr>
<tr>
<td>$n = 1$</td>
<td>red</td>
<td>red</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
</tr>
<tr>
<td>$n = 0$</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
</tr>
</tbody>
</table>
Main lemma

- (b) $v$ is the value of the highest proposal among all proposals less than $n$ accepted by acceptors in $S$
- red is chosen at round 3, no proposer at round 4
- Proposer at round 5 will always get red querying any majority

<table>
<thead>
<tr>
<th>Round</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = 5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n = 4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n = 3$</td>
<td>red</td>
<td>red</td>
<td>red</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$n=2$</td>
<td>red</td>
<td>red</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$n=1$</td>
<td>red</td>
<td>red</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
</tr>
<tr>
<td>$n=0$</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
</tr>
</tbody>
</table>

31
Main lemma

- (b) $v$ is the value of the highest proposal among all proposals less than $n$ accepted by acceptors in $S$
- red is chosen at round 3, no proposer at round 4
- Proposer at round 5 will always get red querying any majority

<table>
<thead>
<tr>
<th>Round</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = 5$</td>
<td></td>
<td>red</td>
<td>red</td>
<td>red</td>
<td></td>
</tr>
<tr>
<td>$n = 4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n = 3$</td>
<td>red</td>
<td>red</td>
<td>red</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$n = 2$</td>
<td>red</td>
<td>red</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$n = 1$</td>
<td>red</td>
<td>red</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
</tr>
<tr>
<td>$n = 0$</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
</tr>
</tbody>
</table>
How to implement P2c

- A proposer at round n needs a query phase to get the value of highest round number + a promise that the state of S does not change until round n

<table>
<thead>
<tr>
<th>Round</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
<th>$a_4$</th>
<th>$a_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = 5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n = 4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n = 3$</td>
<td>red</td>
<td>red</td>
<td>red</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$n = 2$</td>
<td>red</td>
<td>red</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$n = 1$</td>
<td>red</td>
<td>red</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
</tr>
<tr>
<td>$n = 0$</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
</tr>
</tbody>
</table>
How to implement P2c

- A proposer issues prop(n, v)
- Guarantee?
  - v is the value of the highest proposal among all proposals less than n accepted by acceptors in S
- Need a prepare(n) phase Before issuing prop(n, v)
  - Extract a promise from a majority of acceptors not to accept a proposal less than n
  - Acceptor sends back its highest numbered accepted value
Abortable Consensus

**Proposer**
- Pick unique sequence $n$, send \textit{prepare}(n) to all acceptors

3) Proposer upon majority $S$ of promises:
   - Pick value $v$ of highest proposal number in $S$, or if none available pick $v$ freely
   - Issue \textit{accept}(n,v) to all acceptors

5) Proposer upon majority $S$ of responses:
   - If got majority of acks decide(v) and broadcast decide(v); Otherwise abort

**Acceptors**
- Upon \textit{prepare}(n):
  - Promise not accepting proposals numbered less than $n$
  - Send highest numbered proposal accepted with number less than $n$ (\textit{promise})

2) Upon \textit{prepare}(n):

5) Upon \textit{accept}(n,v):
   - If not responded to prepare $m>n$, accept proposal (\textit{ack}); otherwise reject (\textit{nack})

\textbf{Abortable consensus satisfies:}

P2c. If $(n,v)$ is \textit{issued}, there is a majority of acceptors $S$ such that:
   a) no one in $S$ has accepted any proposal numbered “$<$“ $n$, OR
   b) $v$ is value of highest proposal among all proposals “$<$“ $n$ accepted by acceptors in $S$
Paxos Correctness
• P2b. If \( v \) is chosen, every higher proposal issued has value \( v \)

• P2c. If any prop \((n,v)\) is issued, there is a set \( S \) of a majority of acceptors s.t. either
  • (a) no one in \( S \) has accepted any proposal numbered less than \( n \)
  • (b) \( v \) is the value of the highest proposal among all proposals less than \( n \) accepted by acceptors in \( S \)

• Lemma: P2c => P2b
  • Proof map:
    • Prove lemma by assuming P2c, prove P2b follows
      • Prove P2b follows by assuming \( v \) is chosen, prove every higher proposal issued has value \( v \)

• Thus: if P2c is true, and prop \((n,v)\) chosen
  • Show by induction every higher proposal issued has value \( v \)
• P2b. If \( v \) is chosen, every higher proposal issued has value \( v \).

• P2c. If any prop \((n,v)\) is issued, there is a set \( S \) of a majority of acceptors s.t. either:
  • (a) no one in \( S \) has accepted any proposal numbered less than \( n \).
  • (b) \( v \) is the value of the highest proposal among all proposals less than \( n \) accepted by acceptors in \( S \).

Thus: P2c is true, and prop \((n,v)\) chosen.

Show by induction on (on prop number) every higher proposal issued has value \( v \).

Need to show by induction that all proposals \((m,u)\), where \( m \geq n \), have value \( u = v \).
• P2b. If $v$ is chosen, every higher proposal issued has value $v$

• P2c. If any prop $(n,v)$ is issued, there is a set $S$ of a majority of acceptors s.t. either
  • (a) no one in $S$ has accepted any proposal numbered less than $n$
  • (b) $v$ is the value of the highest proposal among all proposals less than $n$ accepted by acceptors in $S$

• Thus: P2c is true, and prop $(n,v)$ chosen

  • Show by induction that all proposals $(m,u)$, where $m \geq n$, have value $u=v$

  • Induction base
    • Inspect proposal $(n,u)$.
    • Since $(n,v)$ chosen & proposals are unique, $u=v

<table>
<thead>
<tr>
<th>Round</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$v$</td>
<td>$v$</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$w$</td>
<td>⊥</td>
<td>⊥</td>
</tr>
<tr>
<td>0</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
</tr>
</tbody>
</table>
- Induction step
  - Assume proposals n, n+1, n+2, ..., m have value v (ind.hypothesis)
    - Show proposal (m+1,u) has u=v
  - P2c implies proposal (m+1,u) has a majority S that either
    - a) no one in S has accepted any proposal numbered less than m+1
    - b) u is the value of the highest proposal among all proposals less than m+1 accepted by acceptors in S
      - a) cannot be, as (n,v) accepted by a majority overlapping with S
      - b) must be true
  - Hence, u is the value of the highest proposal among all proposals less than m+1 accepted by acceptors in S
  - By the induction hypothesis, all proposals n, ..., m have value v. Majority of prop m+1 intersects with majority of prop n, thus u=v
### Induction step

- Assume proposals $n, n+1, n+2, \ldots, m$ have value $v$ (ind.hypothesis)
  - Show proposal $(m+1,u)$ has $u=v$
- $u$ is the value of the highest proposal among all proposals less than $m+1$ accepted by acceptors in $S$
- By the induction hypothesis, all proposals $n, \ldots, m$ have value $v$. Majority of prop $m+1$ intersects with majority of prop $n$, thus $u=v$

<table>
<thead>
<tr>
<th>Round</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$a_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>$v$</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>$v$</td>
</tr>
<tr>
<td>2</td>
<td>$v$</td>
<td>$v$</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$w$</td>
<td>$\perp$</td>
<td>$\perp$</td>
</tr>
<tr>
<td>0</td>
<td>$\perp$</td>
<td>$\perp$</td>
<td>$\perp$</td>
</tr>
</tbody>
</table>
Agreement Satisfied

- This algorithm satisfies P2c
  - accept(n,v) only **issued** if a majority S responded to prepare(n), s.t. for each $p_i$ in $S$:
    a) either: $p_i$ hadn’t accepted any prop less than n, or
    b) $v$ is value of highest proposal less than n accepted by $p_i$

- By their promise, a) and b) will not change

- prepare(n) often called **read(n)**
- accept(n,v) often called **write(n,v)**
Agreement

- P2c. If \((n,v)\) is issued, there is a majority of acceptors \(S\) s.t.
  - a) no one in \(S\) has accepted any proposal numbered less than \(n\), or
  - b) \(v\) is the value of the highest proposal among all proposals less than \(n\) accepted by acceptors in \(S\).

- P2. If \((n,v)\) is chosen, every higher proposal chosen has value \(v\).

- We proved that if P2c is satisfied, then P2 is satisfied
  - P2c \(\Rightarrow\) P2

- Thus the algorithm satisfies agreement (safety)
Obstruction Freedom and Validity

• P1. An acceptor accepts first “proposal” it receives

• P1 is satisfied because we accept
  • if prepare(n) & accept(n,v) received first

• Thus the algorithm satisfies obstruction-free progress (liveness)
Getting Familiar with Paxos
Abortable Consensus

**Proposer**

1) Pick unique sequence n, send prepare(n) to all acceptors

3) Proposer upon majority S of promises:
   - Pick value v of highest proposal number in S, or if none available pick v freely
   - Issue accept(n,v) to all acceptors

5) Proposer upon majority S of responses:
   - If got majority of acks decide(v) and broadcast decide(v);
   - Otherwise abort

**Acceptors**

2) Upon prepare(n):
   - Promise not accepting proposals numbered less than n
   - Send highest numbered proposal accepted with number less than n (promise)

4) Upon accept(n,v):
   - If not responded to prepare m>n, accept proposal (ack); otherwise reject (nack)
Message loss and failures

- Many sources of *abort*
  - Contention (multiple proposals competing)
  - Message loss (e.g. not getting an ack)
  - Process failure (e.g. proposer dies)

- So Proposers try Abortable Consensus again…
  - Prepare(5), Accept(5,v), prepare(15), …
  - Eventually the Paxos should terminate (FLP85?)
FLP ghost

 proposers a and b forever racing...
  • Eventual leader election ($\Omega$) ensures liveness
  • Eventually only one proposer => termination
Familiarizing with Paxos (1/4)

- Different processes accept different values, same process accepts different values
- Assume 4 proposers \{a,b,c,d\}, 7 acceptors \{p_1,...,p_7\}

```
p_1
  a.prep(1):ok  a.acpt(1,red):ok

p_2
  a.prep(1):ok

p_3
  a.prep(1):ok

p_4
  a.prep(1):ok

p_5

p_6

p_7
```

S. Haridi, KTHx ID2203.1x
Familiarizing with Paxos (2/4)

- Different nodes accept different values, same node accepts different values
- Assume 4 proposers \{a,b,c,d\}, 7 acceptors \{p_1,...,p_7\}

\[
\begin{align*}
p_1 & : \text{a.prep(1):ok a.acpt(1,\text{red}):ok} \\
p_2 & : \text{a.prep(1):ok b.prep(2):ok b.acpt(2,\text{blue}):ok} \\
p_3 & : \text{a.prep(1):ok b.prep(2):ok} \\
p_4 & : \text{a.prep(1):ok b.prep(2):ok} \\
p_5 & : \text{b.prep(2):ok} \\
p_6 \phantom{\text{b.prep(2):ok}} \\
p_7 \phantom{\text{b.prep(2):ok}}
\end{align*}
\]
Familiarizing with Paxos (3/4)

- Different nodes accept different values, same node accepts different values
- Assume 4 proposers \{a,b,c,d\}, 7 acceptors \{p_1,...,p_7\}

<table>
<thead>
<tr>
<th>Proposer</th>
<th>Acceptors</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>p_1, p_2, p_3, p_4, p_5, p_6, p_7</td>
</tr>
<tr>
<td>b</td>
<td>p_2, p_4, p_5, p_6, p_7</td>
</tr>
<tr>
<td>c</td>
<td>p_3, p_5, p_6, p_7</td>
</tr>
</tbody>
</table>

```

<table>
<thead>
<tr>
<th></th>
<th>p_1</th>
<th>p_2</th>
<th>p_3</th>
<th>p_4</th>
<th>p_5</th>
<th>p_6</th>
<th>p_7</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>acpt(1, red):ok</td>
<td>acpt(2, blue):ok</td>
<td>acpt(3, green):ok</td>
<td>acpt(4, green):ok</td>
<td>acpt(5, red):ok</td>
<td>acpt(6, blue):ok</td>
<td>acpt(7, green):ok</td>
</tr>
<tr>
<td>c</td>
<td>prep(3):ok</td>
<td>prep(3):ok</td>
<td>prep(3):ok</td>
<td>prep(3):ok</td>
<td>prep(3):ok</td>
<td>prep(3):ok</td>
<td>prep(3):ok</td>
</tr>
</tbody>
</table>
```

S. Haridi, KTHx ID2203.1x
Familiarizing with Paxos (4/4)

- Different nodes accept different values, same node accepts different values
- Assume 4 proposers \{a, b, c, d\}, 7 acceptors \{p_1, ..., p_7\}

```
p_1
a.prep(1):ok    a.acpt(1, red):ok
p_2
a.prep(1):ok    b.prep(2):ok    b.acpt(2, blue):ok
p_3
a.prep(1):ok    b.prep(2):ok    c.prep(3):ok    c.acpt(3, green):ok
p_4
a.prep(1):ok    b.prep(2):ok    c.prep(3):ok    d.prep(4):ok
p_5
b.prep(2):ok    c.prep(3):ok    d.prep(4):ok
p_6
p_7
```
Optimizations
Paxos (AC) in a nutshell

- Necessary
  - Reject accept(n,v) if answered prepare(m) : m>n
    - i.e. prepare extracts promise to reject lower accept
Possible scenario #1

- Caveat
  - Proposers \{a,b,c\}, acceptors \{p_1,p_2,p_3\}

<table>
<thead>
<tr>
<th></th>
<th>a.prep(80):ok</th>
<th>b.prep(10):ok</th>
<th>b.accept(10,\text{red}):fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p_2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p_3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- accept(10) will be rejected, why answer prepare(10)?
- No point answering prepare(n) if accept(n,v) will be rejected
Summary of Optimizations

• Necessary
  • Reject accept(n,v) if answered prepare(m) : m>n
    • i.e. prepare extracts promise to reject lower accept

• Optimizations
  • a) Reject prepare(n) if answered prepare(m) : m>n
    • i.e. prepare extracts promise to reject lower prepare
**Possible scenario #2**

- **Caveat**

<table>
<thead>
<tr>
<th></th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
<th>p5</th>
<th>p6</th>
<th>p7</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>prep(80):ok</td>
<td>prep(80):ok</td>
<td>prep(80):ok</td>
<td>prep(80):ok</td>
<td>prep(80):ok</td>
<td>prep(80):ok</td>
<td>prep(80):ok</td>
</tr>
<tr>
<td>b</td>
<td>prep(90):ok</td>
<td>prep(90):ok</td>
<td>prep(90):ok</td>
<td>acpt(90,red):ok</td>
<td>acpt(80,blue):fail</td>
<td>acpt(90,red):ok</td>
<td>acpt(80,blue):ok</td>
</tr>
</tbody>
</table>

- accept(80,blue) can anyway not get majority, as P2b guarantees every higher proposal issued would have same value!

S. Haridi, KTHx ID2203.1x
Summary of Optimizations (2)

- Necessary
  - Reject accept(n,v) if answered prepare(m) : m>n
    - i.e. prepare extracts promise to reject lower accept

- Optimizations
  - a) Reject prepare(n) if answered prepare(m) : m>n
    - i.e. prepare extracts promise to reject lower prepare
  - b) Reject accept(n,v) if answered accept(m,u) : m>n
    - i.e. accept extracts promise to reject lower accept
  - c) Reject prepare(n) if answered accept(m,u) : m>n
    - i.e. accept extracts promise to reject lower prepare
Possible scenario #3

- Caveat

\[
\begin{align*}
p1 & \quad \text{prep}(1) \quad \text{ok} \quad \text{acpt}(1, \text{red}) \quad \text{ok} \\
p2 & \quad \text{ok} \quad \text{ok} \\
p3 & \quad \text{ok} \\
p4 & \quad \text{ok} \\
p5 & \quad \text{ok} \\
\end{align*}
\]

Opt: ignore old responses
Summary of Optimizations (3)

- **Necessary**
  - Reject accept(n,v) if answered prepare(m) : m>n
    - i.e. prepare extracts promise to reject lower accept

- **Optimizations**
  - a) Reject prepare(n) if answered prepare(m) : m>n
    - i.e. prepare extracts promise to reject lower prepare
  - b) Reject accept(n,v) if answered accept(m,u) : m>n
    - i.e. accept extracts promise to reject lower accept
  - c) Reject prepare(n) if answered accept(m,u) : m>n
    - i.e. accept extracts promise to reject lower prepare
  - d) Ignore old messages to proposals that got majority
State to Remember

- Each acceptor remembers
  - **Highest proposal** \((n,v)\) accepted
    - Needed when proposers ask prepare\((m)\)
    - Lower prepares anyway ignored (optimization a & c)

  - **Highest prepare** it has promised
    - It has promised to ignore accept\((m)\) with lower number

- Can be saved to stable storage (recovery)
One more optimizations - 1

- Paxos requires 2 round-trips (with no contention)
  - Prepare(n) : prepare phase (read phase)
  - Accept(n, v): accept phase (write phase)

- P2. If v is chosen, every higher proposal chosen has value v

- Optimization 1
- Proposer skips the accept phase if a majority of acceptors return the same value v
Performance

- Paxos requires 4 messages delays (2 round-trips)
  - Prepare(n) needs 2 delays (Broadcast & Get Majority)
  - Accept(n, v) needs 2 delays (Broadcast & Get Majority)

- In many cases only accept phase is run
  - Paxos only needs **2 delays** to terminate
    - (Believed to be) optimal
Two more optimizations - 2

- Paxos requires 2 round-trips (with no contention)
  - Prepare(n) : prepare phase (read phase)
  - Accept(n, v): accept phase (write phase)

- We often need to run many consensus instances
  - Note that proposer needs not know value in prepare(n)
  - Initialize acceptors as if they accepted a prepare(1) of an initial leader $l_1$ among possible proposers
  - Initially $l_1$ runs only accept phase until suspected
  - Subsequent leaders can run prepare for many instances in advance (with higher ballot number)