Abstract

The Paxos algorithm, when presented in plain English, is very simple.
**Three types of processes**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Proposers</strong></td>
<td>A proposer is a process that has a value to propose</td>
</tr>
<tr>
<td><strong>Acceptors (2f+1)</strong></td>
<td>Acceptors are the processes that ultimately choose which proposed value will be decided</td>
</tr>
<tr>
<td><strong>Learners</strong></td>
<td>A learner only cares about learning which value was decided</td>
</tr>
</tbody>
</table>
How the game is played

- **Election:** Proposers first try to get a majority of acceptors to follow them.

- **Legislating:** After acquiring a majority, a proposer can *try* to enforce her value, by getting acceptors to accept it, **but...**

- **Playing nice:** If an elected proposer finds that some previous value has been proposed, she proposes that value instead.

- **Winning the game:** once a majority of acceptors have accepted a value, the value is *chosen/decided*
Greetings, peasants! I am your fearless leader! Grant me your blessing!

We are with you, oh mighty leader!

We are with you, oh wise leader!

My first decree is:

The value should be 12

Sounds good to me!
How it is supposed to work
Greetings, peasants! I am your fearless leader! Grant me your blessing!

We are with you, oh mighty leader!

We are with you, oh wise leader!

Greetings, peasants! I am your fearless leader #2! Grant me your blessing!

We are with you, oh wise leader #2!

Proposer

Acceptors
DEALING WITH MULTIPLE PROPOSERS

- I swear I won’t follow an earlier leader!
- And, btw, here is my current accepted value (if any) by leader x.

Proposer

Acceptors

Learner

IAmLeader #1  YouAreLeader  Decree

Accept
DEALING WITH MULTIPLE PROPOSERS

I swear I won’t follow an earlier leader!
And, btw, here is my current accepted value (if any) by leader x.
DEALING WITH MULTIPLE PROPOSERS

- I swear I won’t follow an earlier leader!
- And, btw, here is my current accepted value (if any) by leader x.
THE CRUCIAL *YouAreLeader* MESSAGE

Proposer #1

- I swear I won’t follow an earlier leader!
- And, btw, here is my current accepted value (if any) by leader x.

Acceptors

1. Wait for a majority of *YouAreLeader* messages before proceeding.
2. If none of them contain a previously accepted value, propose your own. Otherwise, propose the value of the most recent leader.
THE CRUCIAL YouAreLeader MESSAGE

1. Wait for a majority of YouAreLeader messages before proceeding.
2. If none of them contain a previously accepted value, propose your own. Otherwise, propose the value of the most recent leader.

Important
By consulting a majority, the new leader makes sure she cannot have missed a chosen value (a value must be accepted by a majority to be chosen, and any two majorities overlap!)
**EXAMPLES OF ACCEPTOR STATES**
(as leader #50 comes to power)

<table>
<thead>
<tr>
<th>Acceptors</th>
<th>Value</th>
<th>By leader</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="circle.png" alt="Acceptors Icon" /></td>
<td>•</td>
<td>x 37</td>
</tr>
<tr>
<td><img src="circle.png" alt="Acceptors Icon" /></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><img src="circle.png" alt="Acceptors Icon" /></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><img src="circle.png" alt="Acceptors Icon" /></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><img src="circle.png" alt="Acceptors Icon" /></td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
# Examples of Acceptor States

(as leader #50 comes to power)

<table>
<thead>
<tr>
<th>Acceptors</th>
<th>Value</th>
<th>By leader</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>y</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>y</td>
<td>41</td>
</tr>
</tbody>
</table>
Examples of acceptor states
(as leader #50 comes to power)

<table>
<thead>
<tr>
<th>Acceptors</th>
<th>Value</th>
<th>By leader</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td>37</td>
</tr>
<tr>
<td>y</td>
<td>y</td>
<td>42</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>37</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>41</td>
</tr>
</tbody>
</table>
Overview of Paxos

**Proposer**
- Send `IAmLeader(n)` to all
- Wait for a majority of responses

**Acceptor**
- If \( n \) is the highest leader # I have seen: respond with `YouAreLeader(Value, LeaderWhoProposedValue)`

Once majority is received, send `Propose(n, V)` where \( V \) is the highest-leader proposal among the responses (or my own value, if none of the responses had a value)

If \( n \) is the highest leader # I have seen, send `Accept(n, V)` to the learner
TOLERATING $f$ failures

Safety
- There are $2f + 1$ acceptors
- A value is only chosen if accepted by a majority ($f + 1$)
- So, even if $f$ of those acceptors fail, one will remain and will be part of any future majority

Liveness
- The leader always waits for $f + 1$ responses. So, even if $f$ replicas fail, it will not block
THE THREAT TO LIVENESS: DUELING PROPOSERS

Greetings, peasants! I am your fearless leader #1! Grant me your blessing!

Greetings, peasants! I am your fearless leader #3! Grant me your blessing!

Greetings, peasants! I am your fearless leader #5! Grant me your blessing!

Greetings, peasants! I am your fearless leader #7! Grant me your blessing!

Greetings, peasants! I am your fearless leader #2! Grant me your blessing!

Greetings, peasants! I am your fearless leader #4! Grant me your blessing!

Greetings, peasants! I am your fearless leader #6! Grant me your blessing!

Greetings, peasants! I am your fearless leader #8! Grant me your blessing!

...
The threat to liveness: duel ing proposers

This problem can be avoided during synchrony (proposer faults can be detected accurately using timeouts)

It’s impossible to avoid during asynchrony!

Well, we kind of knew that already…
Paxos cannot be both safe and live during asynchrony! (that would violate FLP)

So it’s doing the next best thing: staying safe all the time and achieving liveness when the system starts behaving synchronously.
Using (multi) Paxos to Implement State Machine Replication

The original Paxos algorithm achieves agreement on one value.

SMR required replicas to agree on the sequence of commands that will be executed.

3. Ensure that all replicas go through the same sequence of state transitions.

MultiPaxos: Run an instance of Paxos for each slot in the sequence.

Important: we don’t need to run phase 1 (election) every time!