EECS 591
DISTRIBUTED SYSTEMS

Manos Kaprisos
Fall 2019
PAXOS

An asynchronous consensus protocol
Abstract

The Paxos algorithm, when presented in plain English, is very simple.
THREE TYPES OF PROCESSES

Proposers  A proposer is a process that has a value to propose

Acceptors (2f+1)  Acceptors are the processes that ultimately choose which proposed value will be decided

Learners  A learner only cares about learning which value was decided
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!

My first decree is: The value should be 12

We are with you, oh wise leader!

Sounds good to me!

How it is supposed to work
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

IAmLeader #1  YouAreLeader  Decree

Acceptors

Learner

- I swear I won’t follow an earlier leader!
- And, btw, here is my current accepted value (if any) by leader x.
Proposer #1

IAmLeader #1 YouAreLeader Decree

Acceptors

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

Proposer #2

IAmLeader #2 YouAreLeader Decree

Learner

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.
THE CRUCIAL YouAreLeader MESSAGE

Proposer #1

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.

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

Problem set #2
• PS2 is out; due on 10/16
  • Individual work only

Implementation project
• The implementation project is out; due on 10/30
  • Teams of 2

Research project
• Topic declarations due Tuesday 10/8

Midterm exam
• In class, 3-5pm, Monday 10/21
PRESENTATIONS

• Motivation, motivation, motivation!
• Keep it simple
  • Give the high-level intuition
• Avoid the “wall of text”
• Speak normally, with changes to your inflection
• Practice, practice, practice!
**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>×</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></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></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>x</td>
<td>41</td>
</tr>
</tbody>
</table>
Overview of Paxos

**Proposer**

Send \texttt{IAmLeader}(n) to all

Wait for a majority of responses

Once majority is received, send \texttt{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 \texttt{n} is the highest leader # I have seen, respond with \texttt{YouAreLeader}(Value, LeaderWhoProposedValue)

**Acceptor**

Wait for a majority of responses

If \texttt{n} is the highest leader # I have seen, send \texttt{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
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: DUELING 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