Raft
Consensus for Rubyists
@vanstee
Big Nerd Ranch
Coffee Snob Scenario
Want to meet for coffee?
Totally!

I’m in.

Totally!

zzzzz...
Ok, let’s meet at Octane.
Matt

Amy

I'll be there.

Sounds good.

Kim

Dan

...
Who wants coffee? *wakes up
Who wants coffee?
Oh, we’re going to Octane.
Delicious Coffee Consensus
What did we just do?
• Leader election
• State replication
• Partition tolerance
Now write an algorithm.
Humans > Computers > Congress
Why Ruby
Multiple db servers?
Multiple app servers?
Multiple clients?
Multiple db servers?  Distributed
Multiple app servers?
Multiple clients?
Multiple db servers? Distributed
Multiple app servers? Distributed
Multiple clients?
Multiple db servers?  Distributed
Multiple app servers?  Distributed
Multiple clients?  Distributed
If you build webapps, you build distributed systems.
Raft
@ongardie
Diego Ongaro
and
John Ousterhout
consensus
/kən'sensəs/
Agreeing upon state across distributed processes even in the presence of failures.
When should I use it?
• Distributed locks
• Configuration
• Background jobs
Why not Paxos?
Laying the Groundwork
Leader Election
Log Replication
Safety
Follower

Candidate

Leader

Times out,
Starts election
Follower

Times out, Starts election

Candidate

Wins election

Leader
Follower

Times out, 
Restarts election

Candidate

Times out, 
Starts election

Leader

Wins election
Follower

Times out, Starts election

Candidate

Times out, Restarts election

Wins election

Discovers current or new leader, Steps down

Leader
RequestVote

Candidate → Follower

Vote for me please.

AppendEntries

Leader → Follower

Here are some entries for your log.

Oh, and btw I’m still the leader.
Higher numbers are used to determine leaders and check log entries. The term is incremented each time an election is started. Any command with an old term is ignored.

Term

Term 1  Term 2  Term 3

Election  Normal Operation  Split vote
Candidate

Times out, starts election

Votes

Follower

Follower

Follower
Candidate

Votes for self

Votes

Follower

Follower

Follower
Candidate

Sends out RequestVotes

Follower

Votes

Follower

Follower
Candidate

Follower

Votes

Responds with success

Follower
Yay! I won!
Leader Election
Log Replication
Safety
Log Entries Explained
<table>
<thead>
<tr>
<th>Leader</th>
<th>Follower</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>1</strong> SET X = 1</td>
<td><strong>1</strong> SET X = 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>1</strong> SET Y = 2</td>
<td><strong>1</strong> SET Y = 2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>1</strong> SET Z = 3</td>
<td><strong>1</strong> SET Z = 3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td><strong>2</strong> SET X = 4</td>
<td><strong>2</strong> SET X = 4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td><strong>2</strong> SET X = 5</td>
<td><strong>2</strong> SET X = 4</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>
APPEND ENTRIES EXAMPLE

{
  :entries => [{ 4 => 'SET X = 4' }],
  :term => 1,
  :prev_log_term => 1,
  :prev_log_index => 3,
  :leader_commit => 3,
  :leader_id => '192.168.1.101/7238'
}
Before

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SET X = 1
SET Y = 2
SET Z = 3

After

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SET X = 1
SET Y = 2
SET Z = 3
SET X = 4
Log entries are always committed in the same order and are never uncommitted.
Happy Log Entry
Leader

Follower

Follower
Sad Log Entry
Facepalm Log Entry
How do we guard against losing log entries?
Leader Election

Log Replication

Safety
Only cast votes for nodes with logs that contain at least as many entries as your own.
New leaders must commit a log entry form their new term before committing old entries.
Bonus Round
• Cluster changes
• Log compaction
• Client specifics
Why Ruby
Ruby is great for expressing complex problems.
Then why doesn’t the academic community love it?
• Community
• Understandability
• Learning Material
• Read Papers
• Go To Conferences
• Talk To People
Question Time
Make sure I repeat your questions.