Announcements

• Project proposals are due tonight, unless you got an extension.
  • Only a few hours left to submit something or seek an extension.
• No quiz next week.
• Should have gotten results for last week's quiz.
RSMs All Over Again
Revisiting RSMs

Application -> Ordering

Client

Application -> Ordering

Client

Application -> Ordering

Client

Application -> Ordering

Client
Revisiting RSMs
Revisiting RSMs

Client  
Raft  
Client  
Raft  
Client  
Raft  
Client
Revisiting RSMs

Diagram showing multiple Raft nodes connected in a loop, with arrows indicating communication between them. At the bottom, there are four Client icons.
The act of executing a command at the application is **destructive**. Cannot undo a command.
Revisiting RSMs

Requirement: All application replicas end up in the same state.
Revisiting RSMs

M0
KVStore
Raft

M1
KVStore
Raft

M2
KVStore
Raft

M3
KVStore
Raft

Client

① set(x, 5)
Revisiting RSMs

M0

KVStore

Raft

M1

KVStore

Raft

②AppendEntries

M2

KVStore

Raft

M3

KVStore

Raft

①set(x, 5)

Client

Client

Client

Client
Revisiting RSMs

M0

KVStore

Raft

③ set(x, 5) ④

M1

KVStore

Raft

② AppendEntries

M2

KVStore

Raft

②

M3

KVStore

Raft

②

① set(x, 5)

⑤ success

Client

Client

Client

Client
Revisiting RSMs

KVStore

Raft

set(x, 5)

1

2

2

2

AppendEntries

M0

M1

M2

M3

set(x, 5), 0, 1

set(x, 5), 0, 1

set(x, 5), 0, 1

set(x, 5), 0, 1

Client

Client

Client

Client

success

③set(x, 5) ④

②AppendEntries

①set(x, 5)
Revisiting RSMs

For which replicas is $x=5$?
Revisiting RSMs

When?
Revisiting RSMs

Is this safe?
Revisiting RSMs

- M0
  - KVStore
  - Raft
  - Client
- M1
  - KVStore
  - Raft
  - Client
- M2
  - KVStore
  - Raft
  - Client
- M3
  - KVStore
  - Raft
  - Client

set(x, 5), 0, 1

Term = 2
leaderCommit = -1
Revisiting RSMs

Client

M0 KVStore Raft

M1 KVStore Raft

M2 KVStore Raft

M3 KVStore

Term = 2
leaderCommit = -1

set(x, 5), 0, 1
get(x), 1, 2

set(x, 5), 0, 1
Revisiting RSMs

M0
KVStore
Raft
Client

M1
KVStore
Raft
Client

M2
KVStore
Raft
Client

M3
KVStore
Raft
Client

set(x, 5), 0, 1
get(x), 1, 2

Term = 2
leaderCommit = -1

set(x, 5), 0, 1
get(x), 1, 2

set(x, 5), 0, 1
get(x), 1, 2
Revisiting RSMs

Is this correct?
Revisiting RSMs

M0
KVStore
Raft
Client

M1
KVStore
Raft
Client

M2
KVStore
Raft
Client

M3
KVStore
Raft
Client

set(x, 5), 0, 1
get(x), 1, 2

Term = 2
leaderCommit = 0

set(x, 5), 0, 1
get(x), 1, 2
Revisiting RSMs

M0
KVStore
Raft
Client

M1
KVStore
Raft
Client

M2
KVStore
Raft
Client

M3
KVStore
Raft
Client

Term = 2
leaderCommit = 1

M0
set(x, 5), 0, 1
get(x), 1, 2

M1
set(x, 5), 0, 1
get(x), 1, 2

M2
set(x, 5), 0, 1
get(x), 1, 2

M3
set(x, 5), 0, 1
get(x), 1, 2
Revisiting RSMs

Is this correct?
Revisiting RSMs

M0
set(x, 5), 0, 1
get(x), 1, 2
cas(x, 5, 4), 2, 2

M1
set(x, 5), 0, 1
cas(x, 5, 4), 2, 2

M2
Term = 2
leaderCommit = 2

M3
set(x, 5), 0, 1
cas(x, 5, 4), 2, 2
Configuration Change
Why?

• Want to be able to change the set of servers.
  • Take down servers for maintenance.
  • Add new servers to replace failed ones.
• Other reasons.
How?

• Use a **special** log message which contains the set of servers.

• Use Raft to replicate this to everyone.
How Special?

• All peers use configuration as soon as logged.
• Why safe?
  • We know how to revert this change.
Protocol

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>set(x, 5)</td>
<td>set(x, 6)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>set(x, 5)</td>
<td>set(x, 6)</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

\[ C \]
## Protocol

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>set(x, 5)</td>
<td>set(x, 6)</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>c</td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>set(x, 5)</td>
<td>set(x, 6)</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>c</td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>set(x, 5)</td>
<td>set(x, 6)</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>c</td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>set(x, 5)</td>
<td>set(x, 6)</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>c</td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>set(x, 5)</td>
<td>set(x, 6)</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>c</td>
</tr>
</tbody>
</table>
### Protocol

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>set(x, 5)</td>
<td>set(x, 6)</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>set(x, 5)</td>
<td>set(x, 6)</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>set(x, 5)</td>
<td>set(x, 6)</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

What happens now?
Protocol

\[
\begin{array}{ccc}
0 & 1 \\
0 & 0 \\
\text{set}(x, 5) & \text{set}(x, 6)
\end{array}
\]

\[
\begin{array}{ccccccc}
0 & 1 & 2 & 3 & 4 & 5 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\text{set}(x, 5) & \text{set}(x, 6) & \ldots & \ldots & \ldots & \text{C-all}
\end{array}
\]

\[
\begin{array}{ccc}
0 & 1 & 2 \\
0 & 0 & 0 \\
\text{set}(x, 5) & \text{set}(x, 6) & \ldots
\end{array}
\]
Protocol

0 1 2 3 4 5
0 0 0 0 0 0
set(x, 5) set(x, 6) … … … … 5
0 0 0 0 0 0
C-all

0 1 2 3 4 5
0 0 0 0 0 0
set(x, 5) set(x, 6) … … … … 5
0 0 0 0 0 0
C-all

0 1 2 3 4
0 0 0 0 0
set(x, 5) set(x, 6) … … … …

0 1 2 3 4 5
0 0 0 0 0 0
set(x, 5) set(x, 6) … … … … 5
0 0 0 0 0 0
C-all
### Protocol

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>set(x, 5)</strong></td>
<td><strong>set(x, 6)</strong></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td><strong>C-all</strong></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>set(x, 5)</strong></td>
<td><strong>set(x, 6)</strong></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td><strong>C-all</strong></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>set(x, 5)</strong></td>
<td><strong>set(x, 6)</strong></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td><strong>C-all</strong></td>
</tr>
</tbody>
</table>

---

**Note:** The protocol seems to involve setting values of `x` and `y` and comparing them to determine if certain conditions are met, possibly related to a decision-making process or a game scenario. The values `C-all` and `C-new` might indicate different states or outcomes in the protocol.
Failure Detectors
What Problem?

• We have been depending on random timeouts, etc. to build consensus.
  • Based on partial synchrony: the network is not always behaving at its worse.
  • Tedious to model (for proofs) and tune (for deployment).
  • Abstract them away with failure detectors.
Failure Detector

suspect p0 is failed.
suspect p0, p1 are failed.
suspect p1, p2 are failed.
suspect p1 is failed.
Reasoning about Detectors

Completeness

Failed nodes:
• When are they detected?
• Who detects them?

Accuracy

Live nodes:
• When can they be suspected?
<table>
<thead>
<tr>
<th>Reasoning about Detectors</th>
<th>Completeness</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong</td>
<td>Every failed node is eventually detected by <strong>all</strong> correct nodes.</td>
<td><strong>No correct node</strong> is ever suspected.</td>
</tr>
<tr>
<td>Weak</td>
<td>Every failed node is eventually detected by <strong>some</strong> correct nodes.</td>
<td><strong>Some correct node</strong> is never suspected by any node.</td>
</tr>
</tbody>
</table>
### Reasoning about Detectors

<table>
<thead>
<tr>
<th></th>
<th><strong>Accuracy</strong></th>
<th><strong>Not Eventual</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Eventual</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Strong</td>
<td>Eventually <strong>No correct node</strong> is ever suspected.</td>
<td><strong>No correct node</strong> is ever suspected.</td>
</tr>
<tr>
<td>Weak</td>
<td>Eventually <strong>some correct node</strong> is never suspected by any node.</td>
<td><strong>Some correct node</strong> is never suspected by any node.</td>
</tr>
</tbody>
</table>
Types of Detectors

- Strong completeness, strong accuracy: Perfect detector (P)
- Strong completeness, weak accuracy: Strong detector (S)
- Strong completeness, eventual strong accuracy: ♦P
- Strong completeness, eventual weak accuracy: ♦S or Ω
Types of Detectors

- Weak completeness, strong accuracy: $Q$
- Weak completeness, weak accuracy: Weak Detector ($W$)
- Weak completeness, eventual strong accuracy: $\Diamond Q$
- Weak completeness, eventual weak accuracy: $\Diamond W$
How to use Failure Detectors?
How to build failure detectors?