Failure Detectors

Concurrency Trilogy Part IV
Announcements

• Project proposals are due tonight, unless you got an extension.
Announcements

• Project proposals are due tonight, unless you got an extension.

• Only a few hours left to submit something or seek an extension.
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.
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
Revisiting RSMs

Application
Ordering

Application
Ordering

Application
Ordering

Application
Ordering

Client
Client
Client
Client
Revisiting RSMs

![Diagram of a process flow with Application and Ordering boxes connected by arrows, leading to Client boxes.]

Ordering
Application
Application
Application
Application
Ordering
Ordering
Ordering
Ordering
Client
Client
Client
Client
Revisiting RSMs

Client

Application

Ordering

Application

Ordering

Application

Ordering

Application

Ordering

Client

Client

Client

Client
Revisiting RSMs

- KVStore
- Raft
- Client
Revisiting RSMs

Raft

Raft

Raft

Raft

Client

Client

Client

Client
Revisiting RSMs
Revisiting RSMs

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

Client  

M0

KVStore

Raft

M1

KVStore

Raft

M2

KVStore

Raft

M3

KVStore

Raft

① set(x, 5)
Revisiting RSMs

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

Process:
1. `set(x, 5)`
2. AppendEntries
Revisiting RSMs

1. Client
2. Raft
3. KVStore

- Client
- Raft
- KVStore

M0, M1, M2, M3
Revisiting RSMs

M0

KVStore

Raft

set(x, 5)

M1

KVStore

Raft

AppendEntries

M2

KVStore

Raft

M3

KVStore

Raft

Client

set(x, 5)

Client

Client

Client

Client
Revisiting RSMs

The diagram shows the interaction between clients and Raft nodes in a distributed system. The sequence of events includes:

1. Client initiates a `set(x, 5)` operation.
2. The request is sent to the Raft node on M0, and `AppendEntries` messages are exchanged between Raft nodes to ensure consistency across the replicas.
3. Once the operation is committed, a `success` message is sent back to the client.

The diagram illustrates the consensus process in a Raft-based distributed system, where multiple nodes (M0, M1, M2, M3) participate in maintaining a consistent view of the key-value store.
Revisiting RSMs

Client

set(x, 5), 0, 1

M0

set(x, 5), 0, 1

M1

set(x, 5), 0, 1

M2

set(x, 5), 0, 1

M3

set(x, 5), 0, 1

Raft

KVStore

Client

Raft

KVStore

AppendEntries

set(x, 5)

success
Revisiting RSMs

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

When?
Revisiting RSMs

M0
KVStore
Raft

M1
KVStore
Raft

M2
KVStore
Raft

M3
KVStore
Raft

Term = 2

set(x, 5), 0, 1

set(x, 5), 0, 1

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

M0
KVStore
Raft

M1
KVStore
Raft

M2
KVStore
Raft

M3
KVStore
Raft

Client

M0
set(x, 5), 0, 1

M1
set(x, 5), 0, 1

M2
Term = 2
set(x, 5), 0, 1

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

M0
- KVStore
- Raft

M1
- KVStore
- Raft

M2
- KVStore
- Raft

M3
- KVStore
- Raft

Client

set(x, 5), 0, 1

set(x, 5), 0, 1

set(x, 5), 0, 1

set(x, 5), 0, 1

Term = 2
Revisiting RSMs

Is this safe?
Revisiting RSMs

Is this safe?
Revisiting RSMs

M0
KVStore
Raft
Client

M1
KVStore
Raft
Client

M2
KVStore
Raft
get(x)

M3
KVStore
Raft
Client

Term = 2
leaderCommit = -1

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

**M0**
- KVStore
- Raft

**M1**
- KVStore
- Raft

**M2**
- KVStore
- Raft

**M3**
- KVStore
- Raft

Client

---

- set(x, 5), 0, 1

- Term = 2
- leaderCommit = -1

- get(x)

---

- set(x, 5), 0, 1

- get(x), 1, 2
Revisiting RSMs

- **KVStore**
  - **Raft**
- **Client**
- **M0**
  - set(x, 5), 0, 1
  - get(x), 1, 2
- **M1**
  - set(x, 5), 0, 1
  - get(x), 1, 2
- **M2**
  - Term = 2
  - leaderCommit = -1
- **M3**
  - set(x, 5), 0, 1
  - get(x), 1, 2

Diagram:
- M0: KVStore, Raft
- M1: KVStore, Raft
- M2: KVStore, Raft, Get(x)
- M3: KVStore, Raft
- Client

Actions:
- M0: get(x)
- M1: AppendEntries
- M2: leaderCommit = -1
- M3: get(x)
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

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

Is this correct?
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

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

M1

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

M2

Term = 2
leaderCommit = 1

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

M3

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

M0
KVStore
Raft
Client

M1
KVStore
Raft
Client

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

M2
KVStore
Raft
Client

Term = 2
leaderCommit = 1

M3
KVStore
Raft
Client

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

M0
KVStore
Raft
Client

M1
KVStore
Raft

M2
KVStore
Raft
cas(x, 5, 4)

M3
KVStore
Raft

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

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

M2
Term = 2
leaderCommit = 2
set(x, 5), 0, 1
g(x), 1, 2

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

KVStore

Raft

Client

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

Term = 2
leaderCommit = 2

M0

M1

M2

M3

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

cas(x, 5, 4), 2, 2

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

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

M0

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

M1

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

M2

Term = 2
leaderCommit = 2
set(x, 5), 0, 1
cas(x, 5, 4), 2, 2
get(x), 1, 2

M3

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

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

cas(x,5,4), 2, 2

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

cas(x,5,4), 2, 2

M2
Term = 2

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

cas(x,5,4), 2, 2
Revisiting RSMs

Is this correct?
Revisiting RSMs

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

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

M2
Term = 2

M3
set(x, 5), 0, 1
get(x), 1, 2
cas(x, 5, 4), 2, 2

Configuration Change
Why?

- Want to be able to change the set of servers.
Why?

- Want to be able to change the set of servers.
- Take down servers for maintenance.
Why?

- Want to be able to change the set of servers.
- Take down servers for maintenance.
- Add new servers to replace failed ones.
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.
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>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</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>3</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Diagram:**
- Panel 1: `set(x, 5)`
- Panel 2: `set(x, 6)`
- Panel 3: `set(x, 6)`
## 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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<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>
<td>c</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</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>set(x, 5)</td>
<td>set(x, 6)</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>c</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</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>set(x, 5)</td>
<td>set(x, 6)</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>c</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</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>set(x, 5)</td>
<td>set(x, 6)</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>c</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Protocol

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

0 1
0 0
set(x, 5) set(x, 6)

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

0 1 2
0 0 0
set(x, 5) set(x, 6) ...
Protocol

\[
\begin{array}{ccc}
0 & 1 & 2 \\
0 & 0 & 3 \\
\text{set}(x, 5) & \text{set}(x, 6) & 4 \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
0 & 0 & 0 & 0 \\
\text{set}(x, 5) & \text{set}(x, 6) & \ldots & \ldots \\
\end{array}
\]

\[
\begin{array}{ccc}
0 & 1 & 2 \\
0 & 0 & 3 \\
\text{set}(x, 5) & \text{set}(x, 6) & \ldots \\
\end{array}
\]

\[
\begin{array}{c}
0 \\
\text{set}(x, 5) \\
\end{array}
\]

\[
\begin{array}{c}
0 \\
\text{set}(x, 6) \\
\end{array}
\]

\[
\begin{array}{c}
\ldots \\
\ldots \\
\end{array}
\]
### Protocol

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</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>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</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>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Note: The table represents the protocol actions and states over time.
### 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>0</td>
<td>set(x, 5)</td>
<td>set(x, 6)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>set(x, 6)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>set(x, 5)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>set(x, 5)</td>
</tr>
</tbody>
</table>

...
Protocol

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

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

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

What happens now?
Protocol

```
set(x, 5)  set(x, 6)
0          0

set(x, 5)  set(x, 6)
0          0

set(x, 5)  set(x, 6)
0          0
```

```
Protocol

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

---


**Protocol**

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<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>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<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>...</td>
<td>C-all</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</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>
<td>0</td>
<td>C-new</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</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>
<td>0</td>
<td>C-all</td>
</tr>
<tr>
<td>set(x, 5)</td>
<td>set(x, 6)</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Failure Detectors
What Problem?

- We have been depending on random timeouts, etc. to build consensus.
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.
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).
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 $p_0$ is failed.
Failure Detector

suspect p0 is failed.
suspect p0, p1 are failed.
Failure Detector

Application

suspect p0 is failed.
suspect p0, p1 are failed.
suspect p1, p2 are failed.

Failure Detector
Failure Detector

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

Completeness

Accuracy
Reasoning about Detectors

<table>
<thead>
<tr>
<th>Completeness</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failed nodes:</td>
<td></td>
</tr>
<tr>
<td>• When are they detected?</td>
<td></td>
</tr>
<tr>
<td>• Who detects them?</td>
<td></td>
</tr>
</tbody>
</table>
Reasoning about Detectors

**Completeness**

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

**Accuracy**

Live nodes:
- When can they be suspected?
Reasoning about Detectors

<table>
<thead>
<tr>
<th>Completeness</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td>Weak</td>
<td></td>
</tr>
</tbody>
</table>
### Reasoning about Detectors

<table>
<thead>
<tr>
<th>Completeness</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every failed node is eventually detected by <strong>all</strong> correct nodes.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Strong</th>
<th>Weak</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Reasoning about Detectors</td>
<td>Completeness</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Every failed node is eventually detected by <strong>all</strong> correct nodes.</td>
<td></td>
</tr>
<tr>
<td>Strong</td>
<td></td>
</tr>
<tr>
<td>Weak</td>
<td>Every failed node is eventually detected by <strong>some</strong> correct nodes.</td>
</tr>
<tr>
<td>Reasoning about Detectors</td>
<td>Completeness</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td>Every failed node is eventually detected by <strong>all</strong> correct nodes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Strong</th>
<th>Weak</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every failed node is eventually detected by <strong>some</strong> correct nodes.</td>
<td></td>
</tr>
</tbody>
</table>
## Reasoning about Detectors

<table>
<thead>
<tr>
<th>Completeness</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Strong</th>
<th>Weak</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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>
<td></td>
</tr>
</tbody>
</table>
## Reasoning about Detectors

<table>
<thead>
<tr>
<th></th>
<th>Eventual</th>
<th>Accuracy</th>
<th>Not Eventual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weak</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Reasoning about Detectors

<table>
<thead>
<tr>
<th></th>
<th>Eventual</th>
<th>Not Eventual</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Strong</strong></td>
<td>Eventually <strong>No correct node</strong> is ever suspected.</td>
<td></td>
</tr>
<tr>
<td><strong>Weak</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Reasoning about Detectors

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>Eventual</th>
<th>Not Eventual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eventual</td>
<td>Eventually <strong>No correct node</strong> is ever suspected.</td>
<td></td>
</tr>
<tr>
<td>Not Eventual</td>
<td><strong>No correct node</strong> is ever suspected.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Strong</th>
<th>Weak</th>
</tr>
</thead>
</table>

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

• Strong completeness, strong accuracy: **Perfect detector (P)**
Types of Detectors

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

- Strong completeness, strong accuracy: **Perfect detector (P)**
- Strong completeness, weak accuracy: **Strong detector (S)**
- Strong completeness, eventual strong strong accuracy: ◊P
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
Types of Detectors

- Weak completeness, strong accuracy: Q
Types of Detectors

- Weak completeness, strong accuracy: Q
- Weak completeness, weak accuracy: Weak Detector (W)
Types of Detectors

- Weak completeness, strong accuracy: $Q$
- Weak completeness, weak accuracy: Weak Detector (W)
- Weak completeness, eventual strong accuracy: $\diamondsuit Q$
Types of Detectors

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