Unreliable Failure Detectors for Reliable Distributed Systems

A different approach

Augment the asynchronous model with an unreliable failure detector for crash failures

Define failure detectors in terms of abstract properties, not specific implementations

Identify classes of failure detectors that allow to solve Consensus

The Model

General

- asynchronous system
- processes fail by crashing
- a failed process does not recover

Failure Detectors

- outputs set of processes that it currently suspects to have crashed
- the set may be different for different processes

Completeness

Strong Completeness  Eventually every process that crashes is permanently suspected by every correct process

Weak Completeness  Eventually every process that crashes is permanently suspected by some correct process
Accuracy

Strong Accuracy
No correct process is ever suspected

Weak Accuracy
Some correct process is never suspected

Reducibility

If we can transform $D$ into $D'$ then we say that $D$ is stronger than $D'$ ($D \geq D'$) and that $D'$ is reducible to $D$.

If $D \geq D'$ and $D' \geq D$ then we say that $D$ and $D'$ are equivalent:

$D \equiv D'$

Failure detectors

<table>
<thead>
<tr>
<th>Completeness</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Strong</td>
</tr>
<tr>
<td></td>
<td>Weak</td>
</tr>
<tr>
<td></td>
<td>Eventual strong</td>
</tr>
<tr>
<td></td>
<td>Eventual weak</td>
</tr>
<tr>
<td>Strong</td>
<td>Perfect $P$</td>
</tr>
<tr>
<td>Weak</td>
<td>Quasi $Q$</td>
</tr>
</tbody>
</table>

$T_{D \rightarrow D'}$ transforms failure detector $D$ into failure detector $D'$.
Simplify, Simplify!

All weakly complete failure detectors are reducible to strongly complete failure detectors

\[
P \geq Q, \; S \geq W, \; \diamond P \geq \diamond Q, \; \diamond S \geq \diamond W
\]

All strongly complete failure detectors are reducible to weakly complete failure detectors (!)

\[
Q \geq P, \; W \geq S, \; \diamond Q \geq \diamond P, \; \diamond W \geq \diamond S
\]

Weakly and strongly complete failure detectors are equivalent!

From Weak Completeness to Strong Completeness

Every process \(p\) executes the following:

\[
\begin{align*}
\text{output}_p \leftarrow 0 \\
\text{cobegin} \\
\text{|| Task 1:} & \text{ repeat forever} \\
& \{ p \text{ queries its local failure detector module } D_p \} \\
& \text{suspects}_p \leftarrow D_p \\
& \text{send} (p, \text{suspects}_p) \text{ to all} \\
\text{|| Task 2:} & \text{ when receive} (q, \text{suspects}_q) \text{ from some } q \\
& \text{output}_p \leftarrow (\text{output}_p \cup \text{suspects}_p) = \{q\} \\
\text{coend}
\end{align*}
\]

The Theorems

Theorem 1: In an asynchronous system with \(W\), consensus can be solved as long as \(f \leq n - 1\)
The Theorems

Theorem 1  In an asynchronous system with $W$, consensus can be solved as long as $f \leq n - 1$

Theorem 2  There is no $f$-resilient consensus protocol using $\Diamond P$ for $f \geq n/2$

Theorem 3  In asynchronous systems in which processes can use $\Diamond W$, consensus can be solved as long as $f < n/2$

Theorem 4  A failure detector can solve consensus only if it satisfies weak completeness and eventual weak accuracy—i.e. $\Diamond W$ is the weakest failure detector that can solve consensus.

Solving consensus using $S$

$S$: Strong Completeness, Weak Accuracy

- at least some correct process $c$ is never suspected

- Each process $p$ has its own failure detector

- Input values are chosen from the set $\{0,1\}$
**Notation**

We introduce the operators $\oplus$, $\otimes$, $!$

They operate element-wise on vectors whose entries have values from the set \{0, 1, $\perp$\}

$v \otimes \perp = v$
$v \otimes v = v$
$v \otimes \perp = \perp$
$v \otimes v = \perp$
$v \oplus v = v$
$v \oplus \perp = \perp$
$v \oplus v = \perp$
$v \oplus \perp = \perp$

Given two vectors A and B, we write $A \preceq B$ if $A[i] \neq \perp$ implies $B[i] \neq \perp$

**Solving Consensus using any $D \in S$**

1: $V_p := (\ldots, 1, v^*_p, \ldots, \perp)$ (p's estimate of the proposed values)
2: $\Delta_p := (\ldots, 1, \Delta_p, \perp, \ldots, \perp)$ (asynchronous rounds $r_p, 1 \leq r_p \leq n-1$)
3: $\{\text{phase 1}\}$
4: for $r_p := 1$ to $n-1$
5: send $(r_p, \Delta_p, p)$ to all
6: wait until $[V_p : \text{received } (r_q, \Delta_q, q) \text{ or } q \in D_p]$, (query the failure detector)
7: $O_p := V_p$
8: $V_p := V_p \oplus (\otimes q \text{ received } \Delta_q)$
9: $\Delta_p := \Delta_p \otimes O_p$, (value is only echoed the first time it is seen)
10: $\{\text{phase 2}\}$
11: send $(r_p, V_p, p)$ to all
12: wait until $[V_q : \text{received } (r_q, V_q, q) \text{ or } q \in D_p]$
13: $V_p := \otimes q \text{ received } V_q$ (computes the "intersection", including $V_p$)
14: $\{\text{phase 3}\}$
15: decide on leftmost non-$\perp$ coordinate of $V_p$