Failure Detector Properties: 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.

Hard to implement?

Failure Detector Properties: Accuracy

**Strong Accuracy**  
No correct process is ever suspected.

**Weak Accuracy**  
Some correct process is never suspected.

Even weak accuracy hard to realize. So:

**Eventual Strong Accuracy**  
There is a time after which no correct processes is ever suspected.

**Eventual Weak Accuracy**  
There is a time after which some correct processes is never suspected.

Failure Detector Classes

<table>
<thead>
<tr>
<th>Completeness</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong</td>
<td>Weak</td>
</tr>
<tr>
<td>Perfect P</td>
<td>Strong S</td>
</tr>
<tr>
<td>Quasi Q</td>
<td>Weak W</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Eventual</td>
<td></td>
</tr>
<tr>
<td>Strong P</td>
<td>Strong S</td>
</tr>
<tr>
<td>Weak Q</td>
<td>Weak W</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 8 classes
- Perfect failure detectors:
  - strong completeness
  - strong accuracy

Reducibility

$T_{\odot \mp \Delta}$ transforms failure detector $\odot$ into failure detector $\odot'$
- uses $\odot$ to keep variable $output_p$ at every process $p$
- $output_p$ emulates output of $\odot'$
- If we can transform $\odot$ into $\odot'$ then:
  - $\odot \geq \odot'$
  - $\odot$ is stronger than $\odot'$, $\odot'$ is weaker than $\odot$, or $\odot$ is reducible to $\odot'$
- If $\odot \geq \odot'$ and $\odot' \geq \odot$ then we say that $\odot$ and $\odot'$ are equivalent:
  - $\odot \equiv \odot'$
Simplify, Simplify!

- All weakly complete failure detectors are reducible to strongly complete failure detectors
  \[ P \geq Q, \quad S \geq W, \quad \diamond P \geq \diamond Q, \quad \diamond S \geq \diamond W \]

- All strongly complete failure detectors are reducible to weakly complete failure detectors (!
  \[ Q \geq P, \quad W \geq S, \quad \diamond Q \geq \diamond P, \quad \diamond W \geq \diamond S \]

Weakly and strongly complete failure detectors are equivalent!

From Weak Completeness to Strong Completeness

Every process \( p \) executes the following:

\begin{verbatim}
output\_p := 0
cobegin
  || Task 1: repeat forever
    { p queries its local failure detector module \( D\_p \) }
    suspects\_p := D\_p
    send (p, suspects\_p) to all
  || Task 2: when receive(q, suspects\_q) from some q
    output\_p := (output\_p : suspects\_q) - {q}
\end{verbatim}

The Theorems

**Theorem 1.** In asynchronous systems in which processes can use \( W \), Consensus can be solved as long as \( f \leq n - 1 \)

**Theorem 2.** There is no \( f \)-resilient Consensus protocol using \( \diamond P \) for \( f = n/2 \)

**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**. It follows that \( \diamond W \) is the weakest failure detector that can solve Consensus.

Solving Consensus Using \( S \)

- \( S \): Strong Completeness, Weak Accuracy
  - at least one correct process \( c \) is never suspected
  - Each process \( p \) has its own failure detector \( D\_p \)
  - Input values are chosen from the set \( \{0, 1\} \)
Notation

- We introduce the operators $\cdot, \ast, \circ$
  - They operate element-wise on vectors whose entries have values from the set $\{0, 1, \bot\}$

\[
\begin{align*}
  v \cdot v &= v \\
  v \ast v &= v \\
  v \circ v &= v \\
  v \cdot w &= v \\
  v \ast w &= v \\
  v \circ w &= v \\
  v \cdot w &= v \\
  v \ast w &= v \\
  v \circ w &= v
\end{align*}
\]

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

A Useful Lemma

**Lemma 1.** After Phase 1 is complete, $V_r \preceq V_p$ for all processes $p$ which complete Phase 1.

**Proof.**

We show that $F(V_r) = v_r \preceq v_p \preceq v_\bot \implies V_p \preceq V_r$.

1. Let $v$ be the first round when $v$ sees $v_r$
2. If $v = -\bot$, then $v_r \preceq v_\bot$
3. $v$ will send to all $v_p$ in round $r$
4. By round accuracy, all correct processes will receive $v_r$ in the next round
5. $v$ has been forwarded $n-1$ times every other process has seen $v_r$

Solving Consensus using any $D \in S$

**Step 1.** $V_p(\bot, \ldots, \bot, v_p, \ldots, \bot)$ (a vector of the proposed values)

**Step 2.** $A_p(\bot, \ldots, \bot, v_p, \ldots, \bot)$

**Step 3.** (phase 1)

4. for $r_p = 1$ to $n$.
5. send $(r_p, A_p)$ to all
6. wait until $V_p$ received $(r_p, A_p)$ or $q \neq \bot$.
7. $q^* = V_p$
8. $V_p = V_p \circ (D \text{ received } A_p)$
9. $A_p = V_p \ast A_p$

**Step 2**

10. send $(r, V_r, \bot)$ to all
11. wait until $V_p$ received $(r, V_r, \bot)$ or $q \neq \bot$
12. $V_r = V_r \circ (D \text{ received } V_r)$

**Step 3**

13. $V_r = V_r \circ (D \text{ received } V_r)$
14. $V_r = V_r \circ (D \text{ received } V_r)$
15. decide on leftmost non-$\bot$ coordinate of $V_r$

Two Additional Useful Lemmas

**Lemma 2.** After Phase 2 is complete, $V_r = V_p$ for all processes $p$ that complete Phase 1.

**Proof.**

- All processes that have completed Phase 2 have received $V_r$. Since $V_r$ is the smallest vector, it follows that $V_r(p) = \bot \implies V_r[p] = \bot$, $\forall p$.

- By the definition of $V_r$, $V_r(\bot) = \bot \implies V_r[p] = \bot$, $\forall p$ after Phase 2.

**Lemma 3.** $V_r(\bot, \ldots, \bot, \bot, \ldots, \bot, 1)$

**Proof.** $V_r$ contains its initial value.
Solving Consensus

Theorem. The protocol to the left satisfies Validity, Agreement, and Termination

Proof.
Left as an exercise

A Lower Bound - I

Theorem. Consensus with $\frac{f}{n}P$ needs $f < n/2$

Proof.
- Suppose there exist such a protocol, and $n$ is even
- Divide set of processes in two sets $P_1$ and $P_2$ of size $n/2$

A Lower Bound - II

Consider three executions:

1. $P_1 \leftarrow 0; P_2 \leftarrow 1$
   - Detectors work perfectly
   - All processes in $P_1$ are crashed before they can propose
   - $P_1$ decides 0 after $t_1$
2. $P_1 \leftarrow 0; P_2 \leftarrow 1$
   - Detectors make errors
   - All processes in $P_1$ are crashed before they can propose
   - $P_1$ decides 0 after time $t_2$
3. $P_1 \leftarrow 0; P_2 \leftarrow 1$
   - Detectors work perfectly
   - All processes in $P_1$ are crashed before they can propose
   - $P_1$ decides 0 after $t_1$