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

**Accuracy**

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

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

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

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

**Failure detectors**

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

**Reducibility**

$T_D \rightarrow D'$ transforms failure detector $D$ into failure detector $D'$

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: $D \equiv D'$
Simplify, Simplify!

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

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

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:

\[
\text{output}_p := 0 \\
\text{cobegin} \\
\quad \text{|| Task 1: repeat forever} \\
\quad \quad \{ p \text{ queries its local failure detector module } \mathcal{D}_p \} \\
\quad \quad \text{suspects}_p := \mathcal{D}_p \\
\quad \quad \text{send } \langle p, \text{suspects}_p \rangle \text{ to all} \\
\quad \text{|| Task 2: when receive}(q, \text{suspects}_q) \text{ from some } q \\
\quad \quad \text{output}_p := (\text{output}_p \cup \text{suspects}_p) - \{q\} \\
\text{coend}
\]

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 $f < n/2$

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—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 $\odot, \star, \oplus$

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

$\mathbf{v} \odot \perp = \perp \odot \mathbf{v} = \mathbf{v} = \mathbf{v} \odot \mathbf{v} = \mathbf{v}$

$\mathbf{v} \star \perp = \perp \star \mathbf{v} = \mathbf{v} = \mathbf{v} \star \mathbf{v} = \mathbf{v}$

$\mathbf{v} \oplus \perp = \perp \oplus \mathbf{v} = \mathbf{v} = \mathbf{v} \oplus \mathbf{v} = \mathbf{v}$

$\mathbf{v} \& \perp = \perp \& \mathbf{v} = \mathbf{v} = \mathbf{v} \& \mathbf{v} = \mathbf{v}$

$\mathbf{v} \oplus \mathbf{v} = \mathbf{v} \perp \& \mathbf{v} = \mathbf{v} \& \mathbf{v} = \mathbf{v}$

$\mathbf{v} \oplus \mathbf{v} = \mathbf{v} \& \mathbf{v} = \mathbf{v} \& \mathbf{v} = \mathbf{v}$

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

Solving Consensus using any $D \in S$

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

A useful Lemma

Lemma 1 After phase 1 is complete, $V_p \leq V_p$ for all processes $p$ that complete phase 1

Proof We show that $V_p[i] = v_i \& \perp \neq \perp \Rightarrow \forall p: V_p[i] = v_i$

Let $r$ be the first round when $c$ sees $v_i$

\[ r \leq n - 2 \]

- $c$ will send to all $\Delta_p$ with $v_i$ in round $r$

- By weak accuracy, all correct processes receive $v_i$ in round $r$

\[ r = n - 1 \]

- $v_i$ has been forwarded $n-1$ times: every other process has seen $v_i$

A useful Lemma

Lemma 1 After phase 1 is complete, $V_p \leq V_p$ for all processes $p$ that complete phase 1

Proof We show that $V_p[i] = v_i \& \perp \neq \perp \Rightarrow \forall p: V_p[i] = v_i$

Let $r$ be the first round when $c$ sees $v_i$

\[ r \leq n - 2 \]

- $c$ will send to all $\Delta_p$ with $v_i$ in round $r$

- By weak accuracy, all correct processes receive $v_i$ in round $r$

\[ r = n - 1 \]

- $v_i$ has been forwarded $n-1$ times: every other process has seen $v_i$
Two additional cool lemmas

1. $V_p = (\perp, \ldots, \perp, V_p, \ldots, \perp)$ (p's estimate of the proposed values)
2. $V_p = (\perp, \ldots, \perp, V_p, \ldots, \perp)$
3. (Phase 1) $V_p = (\perp, \ldots, \perp, V_p, \ldots, \perp)$
4. for $r_p = 1$ to $n-1$
5. send $(r_p, V_p, p)$ to all
6. wait until $(V_p, V_q, q)$ or $q \in D_p$
7. $O_p = V_p$
8. $V_p \leftarrow O_p = (\oplus q$ received $A_p$
9. $V_p \leftarrow V_p \oplus O_p$ (value is only echoed first time it is seen)
10. (Phase 2)
11. send $(r_p, V_p, p)$ to all
12. wait until $(V_p, V_q, q)$ or $q \in D_p$
13. $V_p \leftarrow V_p \otimes q$ received $V_q$ (computes the "intersection", including $V_p$
14. (Phase 3)
15. decide on leftmost non-$\perp$ coordinate of $V_p$

Lemma 2. After Phase 2 is complete, $V_c = V_p$ for each $p$ that completes phase 2

Proof

All processes that completed phase 2 have received $V_c$. Since $V_c$ is the smallest $V$ vector,
$V_c[i] \neq \perp \Rightarrow V_p[i] \neq \perp \forall p$

By the definition of $\otimes$
$V_c[i] = \perp \Rightarrow V_p[i] = \perp \forall p$

after phase 2

Lemma 3. $V_c \neq (\perp, \perp, \ldots, \perp)$

Solving consensus

1. $V_p = (\perp, \ldots, \perp, V_p, \ldots, \perp)$ (p's estimate of the proposed values)
2. $V_p = (\perp, \ldots, \perp, V_p, \ldots, \perp)$
3. (Phase 1)
4. for $r_p = 1$ to $n-1$
5. send $(r_p, V_p, p)$ to all
6. wait until $(V_p, V_q, q)$ or $q \in D_p$
7. $O_p = V_p$
8. $V_p \leftarrow V_p \oplus (\oplus q$ received $A_q$
9. $V_p \leftarrow V_p \otimes O_p$ (value is only echoed first time it is seen)
10. (Phase 2)
11. send $(r_p, V_p, p)$ to all
12. wait until $(V_p, V_q, q)$ or $q \in D_p$
13. $V_p \leftarrow V_p \otimes q$ received $V_q$ (computes the "intersection", including $V_p$
14. (Phase 3)
15. decide on leftmost non-$\perp$ coordinate of $V_p$

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

Proof

Left as an exercise

A lower bound - I

Theorem. Consensus with $\Diamond P$ requires $f < n/2$

Proof

Suppose $n$ is even, and a protocol exists that solves consensus when $f = n/2$
Divide the set of processes in two sets of size $n/2$, $P_1$ and $P_2$
Consider three executions:

- All processes in $P_1$ crash before they can propose.
- Detectors work perfectly.
- $P_1 \leftarrow 0$; $P_2 \leftarrow 0$

- All processes in $P_2$ crash before they can propose.
- Detectors work perfectly.
- $P_1 \leftarrow 0$; $P_2 \leftarrow 0$

- $P_1$ decides 0 after $t_1$

- $P_1 \leftarrow 1$; $P_2 \leftarrow 1$

- All processes in $P_2$ crash before they can propose.
- Detectors work perfectly.
- $P_1 \leftarrow 0$; $P_2 \leftarrow 0$

- $P_1$ decides 0 after $t_1$

- $P_1 \leftarrow 1$; $P_2 \leftarrow 1$

- All processes in $P_1$ crash before they can propose.
- Detectors work perfectly.
- $P_1 \leftarrow 0$; $P_2 \leftarrow 0$

- $P_2$ decides 1 after $t_2$
A lower bound - II

Consider three executions:

1. All processes in $P_2$ crash before they can propose. Detectors work perfectly.
   - $P_1$ decides 0 after $t_1$
   - $P_2$ decides 1 after $t_2$

2. No process crashes. All processes in $P_1$ crash before they can propose. Detectors make mistakes: until $\max(t_1, t_2)$, $P_1$ believes $P_2$ crashed, and vice versa.
   - $P_1$ decides 0
   - $P_2$ decides 1

3. No process crashes. All processes in $P_1$ crash before they can propose. Detectors make mistakes: until $\max(t_1, t_2)$, $P_1$ believes $P_2$ crashed, and vice versa.
   - $P_1$ decides 0
   - $P_2$ decides 1

The case of the Rotating Coordinator

Solving consensus with $\Diamond W$ (actually, $\Diamond S$)

- Asynchronous rounds
- Each round has a coordinator $c$
- $c_{id} = (r \ mod \ n) + 1$
- Each process $p$ has an opinion $v_p \in \{0, 1\}$ (with a time of adoption $t_p$)
- Coordinator collects opinions to form a suggestion
- If they believe $c$ to be correct, processes adopt its suggestion and make it their own opinion
- A suggestion adopted by a majority of processes is "locked"

One round, four phases

Phase 1

Each process, including $c$, sends its opinion timestamped $r$ to $c$. 

...
One round, four phases

Phase 1
Each process, including \(c\), sends its opinion timestamped \(r\) to \(c\).

Phase 2
\(c\) waits for first \([n/2+1]\) opinions with timestamp \(r\).
\(c\) selects \(v\), one of the most recently adopted opinions.
\(v\) becomes \(c\)'s suggestion for round \(r\).
\(c\) sends its suggestion to all.

Phase 3
Each \(p\) waits for a suggestion, or for failure detector to signal \(c\) is faulty.
If \(p\) receives a suggestion, \(p\) adopts it as its new opinion and ACKs to \(c\).
Otherwise, \(p\) NACKs to \(c\).

Phase 4
\(c\) waits for first \([n/2+1]\) responses.
If all ACKs, then \(c\) decides on \(v\) and sends DECIDE to all.
If \(p\) receives DECIDE, then \(p\) decides on \(v\).

Consensus using \(\diamond S\)

\(v_p\) := input bit; \(r_p := 0\); \(t_p := 0\); \(\text{state}_p := \text{undecided}\)
while \(p\) undecided do
\(c := r + 1\)
\(c := (r \mod n + 1)\)
\(p\) sends \((p, r, v_p, t_p)\) to \(c\)
\(c\) waits for first \([n/2+1]\) opinions \((q, r, v_q, t_q)\)
\(c\) selects among them the value \(v_q\) with the largest \(t_q\)
\(c\) sends \((c, r, v_q)\) to all
\(c\) waits for suggestions from the current coordinator\)
\(p\) waits until suggestion \((c, r, v)\) arrives or \(c \in 0.S_p\)
if suggestion is received then \(v_p := v\); \(t_p := t\); \(p\) sends \((r, \text{ACK})\) to \(c\)
else \(p\) sends \((r, \text{NACK})\) to \(c\)
\(c\) waits for first \([n/2+1]\) \((r, \text{ACK})\) or \((r, \text{NACK})\)
if \(c\) receives \([n/2+1]\) \((r, \text{ACK})\) or \((r, \text{NACK})\)
to all
when \(p\) delivers \((r, \text{DECIDE}, v)\) then \(p\) decides \(v\); \(\text{state}_p := \text{decided}\).
\( S \) Consensus as Paxos

- All processes are acceptors
- In round \( r \), node \((r \mod n) + 1\) serves both as a distinguished proposer and as a distinguished learner
- The round structure guarantees a unique proposal number
- The value that a proposer proposes when no value is chosen is not determined