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>Strong</th>
<th>Weak</th>
<th>Eventual strong</th>
<th>Eventual weak</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong</td>
<td>Perfect $P$</td>
<td>Strong $S$</td>
<td>$\diamond P$</td>
<td>$\diamond S$</td>
</tr>
<tr>
<td>Weak</td>
<td>Quasi $Q$</td>
<td>Weak $W$</td>
<td>$\diamond Q$</td>
<td>$\diamond W$</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

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

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:

\[ \text{output}_p := 0 \]

\[ \text{cobegin} \]

\[ \text{|| Task 1:} \quad \text{repeat forever} \]

\[ \{ \text{\( p \) queries its local failure detector module} \; D_p \} \]

\[ \text{suspects}_p := D_p \]

\[ \text{send} \; \{ p, \text{suspects}_p \} \; \text{to all} \]

\[ \text{|| Task 2:} \quad \text{when receive}(q, \text{suspects}_q) \; \text{from some} \; q \]

\[ \text{output}_p := (\text{output}_p \cup \text{suspects}_p) - \{ q \} \]

\[ \text{coend} \]

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 $f \leq n - 1$

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

- $v \oplus \perp = \perp \oplus v = v$
- $v \otimes \perp = \perp \otimes v = \perp$
- $v \triangleleft \perp = \perp \triangleleft v = \perp$
- $v \oplus v = \perp \triangleleft v = \perp$
- $v \otimes v = \perp \otimes v = \perp$
- $v \triangleleft v = \perp \triangleleft v = \perp$
- $v \otimes v = \perp \otimes v = \perp$
- $v \triangleleft v = \perp \triangleleft v = \perp$
- $v \otimes v = \perp \otimes v = \perp$

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: $V_p := (\ldots, V_p, \ldots, 1)$ (p's estimate of the proposed values)
2: $\Delta_p := (\ldots, \Delta_p, \ldots, 1)$ (asynchronous rounds $r_i \leq t \leq n-1$)
3: (phase 1)
   4: for $r_p := 1$ to $n-1$
   5:   send $(r_p, \Delta_p, p)$ to all
   6:   wait until $[V_q : received (r_p, \Delta_p, q) or q \in D_p]$ (query the failure detector)
   7:   $O_p := V_p$
   8:   $V_p := V_p \otimes (\exists q \in D_p)$
   9: $\Delta_p := V_p \otimes O_p$ (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 $[V_q : received (r_p, V_q, q) or q \in D_p]$ (compiles the "intersection", including $V_p$)
13: $V_p := \otimes q \in D_p$ 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, $V_c \leq V_p$ for all processes $p$ that complete phase 1

A useful Lemma

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

Proof

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

$\n - r \leq n-2$
$\n - c$ will send to all $\Delta_i$ with $v_i$ in round $r$
$\n - By weak accuracy, all correct processes receive $v_i$ by next round
$\n - r = n-1$
$\n - $v_i$ has been forwarded $n-1$ times:
- every other process has seen $v_i$
Two additional cool lemmas

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 \( 1 \)-vector,
\[
V_c[i] \neq \bot \Rightarrow V_p[i] \neq \bot \quad \forall p
\]

By the definition of \( \otimes \)
\[
V_c[i] = \bot \Rightarrow V_p[i] = \bot \quad \forall p
\]

after phase 2.

Lemma 3. \( V_c \neq (\bot, \bot, \bot, \ldots) \)

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 \( \Diamond P \) 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 \( P_2 \).
A lower bound - II

Consider three executions:

- \( P_1 \leftarrow 0; P_2 \leftarrow 0 \)
  - All processes in \( P_1 \) crash before they can propose
  - Detectors work perfectly
  
\[ P_1 \text{ decides } 0 \]
  after \( t_1 \)

- \( P_1 \leftarrow 1; P_2 \leftarrow 1 \)
  - All processes in \( P_1 \) crash before they can propose
  - Detectors work perfectly

- \( P_1 \leftarrow 0; P_2 \leftarrow 0 \)
  - All processes in \( P_2 \) crash before they can propose
  - Detectors work perfectly
  
\[ P_2 \text{ decides } 1 \]
  after \( t_2 \)
### A lower bound - II

Consider three executions:

<table>
<thead>
<tr>
<th>Execution</th>
<th>Processes</th>
<th>Event</th>
<th>Detectors</th>
<th>Opinion</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1 \leftarrow 0; P_2 \leftarrow 0$</td>
<td>All processes in $P_2$ crash before they can propose</td>
<td>No process crashes</td>
<td>Detectors work perfectly</td>
<td>$P_1$ decides 0 after $t_1$</td>
</tr>
<tr>
<td>$P_1 \leftarrow 0; P_2 \leftarrow 1$</td>
<td>All processes in $P_1$ crash before they can propose</td>
<td>Detectors make mistakes until $\max(t_1, t_2)$</td>
<td>$P_1$ believes $P_2$ crashed, and vice versa</td>
<td>$P_2$ decides 1 after $t_2$</td>
</tr>
<tr>
<td>$P_1 \leftarrow 1; P_2 \leftarrow 1$</td>
<td>All processes in $P_1$ crash before they can propose</td>
<td>Detectors work perfectly</td>
<td>$P_1$ decides 0</td>
<td>$P_2$ decides 1</td>
</tr>
</tbody>
</table>

### 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 $c$
Phase 1
Each process, including $r$, sends its opinion timestamped $r$ to $c$.

Phase 2
$c$ waits for first $\lceil n/2+1 \rceil$ 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 $\lceil n/2+1 \rceil$ 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 $\Diamond S$

\begin{align*}
\gamma_c & := \text{input bit}; \quad \rho_c := 0; \quad \delta_{\text{def}} := \text{undecided} \\
\text{while } p \text{ undecided do} & \\
\quad \text{ce} & := r+1 \\
\quad \text{ce} & := (r \mod n) + 1 \quad \text{(phase 3: all processes send opinion to current coordinator)} \\
\quad p & \text{ sends } (p, r, v_p, t_p) \text{ to } c \\
\quad (\text{phase 2: current coordinator gathers a majority of opinions}) & \\
\quad c & \text{ waits for first } \lceil n/2+1 \rceil \text{ opinions } (q, r, v_q, t_q) \\
\quad c & \text{ selects among them the value } v_q \text{ with the largest } t_q \\
\quad c & \text{ sends } (c, r, v_q) \text{ to all} \\
\quad (\text{phase 3: all processes wait for new suggestion from the current coordinator}) & \\
\quad p & \text{ waits until suggestion } (x, r, v_x) \text{ arrives or } c \in G_S. \\
\quad \text{if suggestion is received then} & \left( v := x; \quad \text{ce} := r; \quad p \text{ sends } (r, \text{ACK}) \text{ to } c \right) \\
\quad \text{else } & \quad p \text{ sends } (r, \text{NACK}) \text{ to } c \\
\quad (\text{phase 4: coordinator waits for majority of replies. If majority adopted the coordinator's suggestion, then coordinator sends request to decide}) & \\
\quad c & \text{ waits for first } \lceil n/2+1 \rceil \text{ } (c, \text{ACK}) \text{ or } (r, \text{NACK}) \\
\quad c & \text{ if } \text{receives } \lceil n/2+1 \rceil \text{ ACKs, then } c \text{ sends } (r, \text{DECIDE}) \text{ to all} \\
\quad \text{when } p \text{ delivers } (r, \text{DECIDE}) & \quad \{ p \text{ decides } v; \quad \delta_{\text{def}} := \text{decided} \}
\end{align*}
### Validity

The value decided upon must have been suggested by the coordinator in some round:

- A coordinator suggests a value only by selecting among the participants' opinions.

From the algorithm, it is clear that each opinion corresponds to a value proposed by some process.

### Agreement

Strong Agreement

All processes that decide, decide the same value.

**Proof**

- Trivially true if no process decides.
- If some process decides, it has delivered (-, DECIDE, -) to a coordinator.
- The coordinator has received a majority of (-, ACK) in a Phase 3...
- Let \( c \) be the earliest round in which a majority of (-, ACK) have been sent to the coordinator on \( \neq r \).
- Let \( r' \) be the value suggested by \( c \) in Phase 2 of round \( r \).
- Enter the Locking Lemma!

### The Locking Lemma - I

**Locking Lemma**

For all rounds \( r' \):

- \( r' \geq r \):
  - If a coordinator \( c' \) sends \( v_{c',r'} \) in Phase 2 of \( r' \), then \( v_{c,r} = v_{c,r'} \).

**Proof**

- Trivially holds for \( r' = r \).
- Assume it holds for all \( r' : r \leq r' < k \).
- Let \( c \) be the coordinator for round \( k \).
- If \( c \) suggests \( v_{c} \), it must have received opinions from a majority of processes.
- There exists some \( p \) that sent an ACK in Phase 3 of round \( r \) whose opinion has been received by \( c \).
- Consider the time of adoption \( t_{p} \).
  - In Phase 3 of round \( r \), \( t_{p} = r \).
  - In Phase 2 of round \( k \), \( t_{p} \geq r \).
  - For any \( t_{q} \) collected in round \( k \), \( t_{q} < k \).

### The Locking Lemma - II

Consider \( t \), the largest time of adoption collected by \( c \).

Clearly, \( t \leq t \).

- \( c \) adopted its suggestion from \( q \), where \( q \) is the process that sent \( (q, k, t_{q}, t) \).
- The coordinator of round \( t \) sent its suggestion in Phase 2 of round \( t \), where \( r \leq t < k \).
- By the Induction Hypothesis, that coordinator sent \( v_{c} \).
- Then, \( c \) sets \( v_{c} \) to \( v_{c} \).

Been there, done that?
Agreement

All processes that decide, decide $v_c$.

**Proof**
- Suppose $p$ delivers $(r, DECIDE, v_p)$.
- The coordinator $c^*$ for round $r$ has sent $(r, DECIDE, v_{c^*})$ in Phase 4 of round $r$.
- To do so, $c^*$ must have received a majority of $(r, ACK)$ in Phase 4 of $r$.
- $r$ is the earliest round in which a majority of $(r, ACK)$ have been sent to a round's coordinator.
- Clearly, $r \leq r^*$.
- By the locking Lemma, $c^*$ must have suggested the locked value: $v_{c^*} = v_c$.

Termination

No correct process is blocked forever at a wait statement.

By eventual weak accuracy, there is a correct process $c$ and a time $t$ such that no process suspects $c$ after $t$.

There is a round $r$ such that:
- all correct processes reach $r$ after time $t$ (no one suspects $c$)
- $c$ is the coordinator for round $r$.
- If some correct process decides, eventually all do on the same value by Agreement.

◊ 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.