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>
</tr>
</thead>
<tbody>
<tr>
<td>Strong</td>
<td>Perfect $P$</td>
</tr>
<tr>
<td>Weak</td>
<td>Quasi $Q$</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
  \[ P \geq Q, \quad S \geq W, \quad \Diamond P \geq \Diamond Q, \quad \Diamond S \geq \Diamond W \]

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

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

- $\square$ at least some correct process $c$ is never suspected
- $\heartsuit$ Each process $p$ has its own failure detector
- $\spadesuit$ Input values are chosen from the set $\{0, 1\}$
**Notation**

We introduce the operators $\circ, \ast, \otimes$

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

- $\bot \ast \bot = \bot$
- $\bot \ast v = v$
- $v \ast \bot = \bot$
- $v \ast v = v$
- $\bot \otimes \bot = \bot$
- $\bot \otimes v = \bot$
- $v \otimes \bot = \bot$
- $v \otimes v = v$
- $\bot \oplus \bot \bot$
- $\bot \oplus v = \bot$
- $v \oplus \bot = \bot$
- $v \oplus v = v$
- $\bot \oplus v = v$
- $\bot \oplus \bot = \bot$
- $\bot \oplus v = v$
- $v \oplus \bot = \bot$
- $v \oplus v = v$
- $\bot \oplus v = v$
- $\bot \oplus \bot = \bot$
- $\bot \oplus v = v$

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

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

1. $V_p = (\ldots, v_p, \ldots)$ (p's estimate of the proposed value)
2. $\Delta_p = (\ldots, \Delta_p, \ldots)$ (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 $[V_q \circ (r_p, \Delta_p, q) \text{ or } q \in D_p]$ (query the failure detector)
7: $O_p : V_p$
8: $V_p \leftarrow V_p \circ O_p$
9: $\Delta_p : V_p \oplus O_p$
10: (phase 2)
11: send $(r_p, V_p, p)$ to all
12: wait until $[V_q \circ (r_p, V_p, q) \text{ or } q \in D_p]$ (computes the "intersection" including $V_p$)
13: $V_p \leftarrow \Delta_p \circ V_q$
14: (phase 3)
15: decide on leftmost non-$\bot$ coordinate of $V_p$

**A useful Lemma**

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

Proof: We show that $V_c[i] = v_i \land v_i \neq \bot = \forall p: V_p[i] = v_i$

- 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 $v_i$ by next round
- $r = n - 1$
- $v_i$ 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, $V_c[i] \neq \perp \Rightarrow V_p[i] \neq \perp$ for all $p$.
- By the definition of intersection, $V_c = \bigotimes V_p$ after phase 2.

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

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 into two sets of size $n/2$, $P_1$ and $P_2$. 

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 into 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.
- 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 $t_2$. 

Before: $P_1 \leftarrow 0; P_2 \leftarrow 0$.
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
  - $P_1$ decides 0 after $t_1$

- $P_1 \leftarrow 0; P_2 \leftarrow 1$
  - No process crashes
  - Detectors make mistakes: until $\max(t_1, t_2)$, $P_1$ believes $P_2$ crashed, and vice versa
  - $P_2$ decides 1 after $t_2$

- $P_1 \leftarrow 1; P_2 \leftarrow 1$
  - All processes in $P_1$ crash before they can propose
  - Detectors work perfectly
  - $P_1$ decides 1 after $t_1$

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 suggestions and make them 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 $r$, 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.
- $r$ 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 ♦$S$**

$v_p := \text{input bit}; r := 0; t := 0; \text{state}_p := \text{undecided}$

while $p$ undecided do

$\ell := r + 1$

(phase 1: all processes send opinion to current coordinator)
$p$ sends $(p, v_p, t_p)$ to $c$

(phase 2: current coordinator gathers a majority of opinions)
$c$ waits for first $\lceil n/2 + 1 \rceil$ opinions $(q, r, v_q, t_q)$
$c$ selects among them the value $v_q$ with the largest $t_q$
$c$ sends $(c, v_q)$ to all

(phase 3: all processes wait for new suggestions from the current coordinator)
$p$ waits until suggestion $(c, v_q)$ arrives or $v_c \in G_S$
if suggestion is received then \$\{ v_c := v_q \}; p$ sends $(c, \text{ACK})$ to $c$
else $p$ sends $(c, \text{NACK})$ to $c$

(phase 4: coordinator waits for majority of replies. If majority adopted the coordinator's suggestion, then coordinator sends request to decide)
$c$ waits for first $\lceil n/2 + 1 \rceil$ $(c, \text{ACK})$ or $(c, \text{NACK})$
$c$ receives $\lceil n/2 + 1 \rceil$ ACKs, then $c$ sends $(c, \text{DECIDE}, v)$ to all
when $p$ delivers $(c, \text{DECIDE}, v)$ then \$p$ decides $v$; $\text{state}_p := \text{decided}$
Validity

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

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

From the algorithm, it is clear that each opinion correspond 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, -) from a coordinator.
- The coordinator has received a majority of (-, ACK) in a Phase 3...
- Let $p_r$ be the earliest round in which a majority of (-, ACK) have been sent to the coordinator $c_r$ of $r$.
- Let $v_{r_0}$ be the value suggested by $c_r$ in Phase 2 of round $r$.
- Enter the Locking Lemma!

The Locking Lemma – I

Consider $t$, the largest time of adoption collected by $c_{r_0}$.

Clearly, $r \geq t < k$.

Let $c_k$ adopt its suggestion from $q_i$, where $q_i$ is the process that sent $(q_i, k, t_{i}, t)$.

The coordinator of round 1 sent its suggestion in Phase 2 of round $t$, where $r \leq t < k$.

By the Induction Hypothesis, that coordinator sent $v_{r_0}$!

Then, $c_k$ sets $v_{r_0}$ to $v_{r_0}$.

Been there, done that!

The Locking Lemma – II

Consider $t$, the largest time of adoption collected by $c_{r_0}$.

Clearly, $r \geq t < k$.

Let $c_k$ adopt its suggestion from $q_i$, where $q_i$ is the process that sent $(q_i, k, t_{i}, t)$.

The coordinator of round 1 sent its suggestion in Phase 2 of round $t$, where $r \leq t < k$.

By the Induction Hypothesis, that coordinator sent $v_{r_0}$!

Then, $c_k$ sets $v_{r_0}$ to $v_{r_0}$.

Been there, done that!
Agreement

All processes that decide, decide $v_c$.

Proof

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

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:

1. All correct processes reach $r$ after time $t$ (no one suspects $c$).
2. $c$ is the coordinator for round $r$.
3. 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.