The Weakest Failure Detector for Message Passing Set-Agreement

Carole Delporte-Gallet[1], Hugues Fauconnier[1],
Rachid Guerraoui[2], Andreas Tielmann[1]

[1]Laboratoire d'Informatique Algorithmique, Fondements et Applications (LIAFA),
University Paris VII, France

[2]School of Computer and Communication Sciences,
EPFL, Switzerland

Schedule:

- What is set-agreement?
- What is a failure detector?
- What is a weakest failure detector?
- What is the weakest failure detector for set-agreement?
- How does this failure detector relate to other failure detectors?

- n processes $\Pi = \{p_1, \ldots, p_n\}$
- Processes communicate by message passing
- Fully connected asynchronous network
- Reliable channels
- Processes may crash
  (processes that do not crash are called correct)
- The system is enhanced with failure detectors

- Introduced by Chaudhuri (1993)
- Also called $(n-1)$-set-agreement
- All processes try to agree (decide) on some set of proposed values:

  Agreement: At most $n-1$ values are decided.
  Validity: Every value decided must have been proposed.
  Termination: Eventually, every correct process decides.
- Generalization of consensus (at most 1 value is decided)
- Trivially solvable, if less than $n-1$ processes may crash
- Not solvable, if any number of processes may crash (Saks/Zaharoglou, Borowsky/Gafni, Herlihy/Shavit) (even in shared memory)
- But: solvable with additional information about failures ($\Rightarrow$ failure detectors)

- Introduced by Chandra and Toueg (1996)
- Allow to circumvent some impossibility results (e.g. for Consensus)
- Modeled as distributed oracles
- Provide information about failures that occur in an execution
- Provide no otherwise information
  - Can be specified as a function of a failure pattern
  - Cannot provide any information about messages sent or the state of the processes

Ω

- Outputs single processes
- Eventually, the same correct process is output at all correct processes

Σ (the quorum failure detector)

- Outputs lists of trusted processes
- Eventually, only correct processes are trusted
- For every process, for every time, every pair of output-lists intersects

anti-Ω

- Outputs single processes
- At least one id of a correct process is only finitely often output

A weakest failure detector $\mathcal{D}$ for a problem P has to be ...

Sufficient: with $\mathcal{D}$ it is possible to solve P

Necessary: every other sufficient failure detector is stronger than ($\equiv$ can emulate) $\mathcal{D}$

- Notion was introduced by Chandra et al. (CHT result)
- Every problem has a weakest failure detector (Jayanti & Toueg, PODC'08)

Some weakest failure detector results:

- Consensus with a correct majority and message passing: Ω (Chandra, Hadzilacos and Toueg, 1992)
- Consensus in shared memory: Ω (Hadzilacos and Lo, 1994)
- Generalization for Consensus in message-passing: (Ω, Σ) and the emulation of registers in message-passing: Σ (Delporte, Fauconnier and Guerraoui, 2003)
- In shared-memory for set-agreement: anti-Ω (Zieliński, 2008)
- Σ ($\equiv (\text{anti-}\Omega, \Sigma)$) is sufficient, but not necessary for set-agreement in message-passing (Delporte, Fauconnier and Guerraoui, 2008)
- In shared-memory for k-set-agreement: k-anti-Ω (Gafni/Kouznetsov, Fernandez-Anta/Rajsbaum/Travers, Delporte-Gallet/Fauconnier/Guerraoui/Tielmann, 2009)
- ...

Can we derive a failure detector from the shared-memory proof?

- Every message passing algorithm can also be executed in shared memory
- $\Rightarrow$ anti-Ω is necessary in message passing
- But is anti-Ω also sufficient?
- No. (Intuition: anti-Ω may behave arbitrary for any finite amount of time)
- $\Rightarrow$ Our failure detector has to be strictly stronger than anti-Ω
- Zieliński conjectured that Σ $(\equiv (\text{anti-}\Omega, \Sigma))$ is the weakest failure detector for message passing (TR 2007)
- Delporte et al. showed that Σ is sufficient, but not necessary (PODC 2008)

Our Result:

The weakest failure detector for message passing set-agreement is $\mathcal{L}$.

The Loneliness Detector $\mathcal{L}$:

- outputs “true” or “false”
- **Prop. 1:** At least one process never outputs “true”
- **Prop. 2:** If only one process is correct, then it eventually outputs “true” forever

Note that at some processes, the output may be unstable forever.

$\mathcal{L}$ is sufficient for Set-Agreement

Algorithm for process p_i:

```
to propose(v):
    initially:
        send ⟨v⟩ to all p_j with j > i;
    on receive ⟨v′⟩ do:
        send ⟨v′⟩ to all;
        decide v′; halt;                    (∗ decision D1 ∗)
    on L = "true" do:
        send ⟨v⟩ to all;
        decide v; halt;                     (∗ decision D2 ∗)
```

Properties of $\mathcal{L}$:

Prop. 1: At least one process never outputs "true"

Prop. 2: If only one process is correct, then it eventually outputs "true" forever

Proof:

Validity: ✓

Termination: ✓

Agreement: ✓

$\mathcal{L}$ is necessary for Set-Agreement

- To show: every other sufficient failure detector is stronger than $\mathcal{L}$
- Assume some failure detector $\mathcal{D}$ is sufficient for set-agreement
- Let A be an algorithm s.t. A using $\mathcal{D}$ implements set-agreement

Algorithm for process p_i:

$output$:= "false";
execute A with value i and detector $\mathcal{D}$, but omit sending messages to others;
if p_i has decided in A, then $output$:= "true";

Proof:

- At least one process never outputs "true" ✓
- If only one process is correct, then it eventually outputs "true" ✓

Thus, $\mathcal{L}$ is the weakest failure detector for message passing set-agreement.

How does $\mathcal{L}$ compare to anti-Ω and Σ ?

Implementation of anti-Ω using $\mathcal{L}$

Recall anti-Ω:

- Outputs single processes
- At least one id of a correct process is only finitely often output

Algorithm for process p_i:

initially:

$\quad$ $lonely := \emptyset$; $\qquad$ (∗ processes where $\mathcal{L} =$ "*true*" ∗)

$\quad$ $output := \{1\}$; $\qquad$ (∗ anti-Ω-output ∗)

on $\mathcal{L} =$ "*true*" **do:**

$\quad$ $lonely := lonely \cup \{i\}$;

$\quad$ send $\langle lonely \rangle$ to all;

$\quad$ $output := \min(\{1, \ldots, n\} \setminus lonely)$;

on receive $\langle lonely' \rangle$ **do:**

$\quad$ **if** $lonely \neq lonely'$ **then** send $\langle lonely \cup lonely' \rangle$ to all;

$\quad$ $lonely := lonely \cup lonely'$;

$\quad$ $output := \min(\{1, \ldots, n\} \setminus lonely)$;

Proof idea:

- For all i: if correct $= \{p_i\}$, then eventually $\mathcal{L}$-output $=$ "true" at p_i (say at time t_i).
- Anti-Ω can behave arbitrarily for any finite amount of time.
- For all i: let the output at p_i up to time t_i be the same as if correct $= \{p_i\}$.
- Let all messages to all p_i be delayed after time t_i.
- $\Rightarrow$ Eventually, $\mathcal{L}$-output $=$ "true" at all p_i.
- $\Rightarrow$ A violation of the specification of $\mathcal{L}$.

Implementation of $\mathcal{L}$ using Σ

Recall the properties of Σ:

- Outputs lists of trusted processes
- **Completeness:** Eventually, only correct processes are trusted
- **Intersection:** For every process, for every time, every pair of output-lists intersects.

Implementation of $\mathcal{L}$ using Σ at process p_i:

$output :=$ "false";
If Σ-output $= \{p_i\}$, then $output :=$ "true";

Proof idea ($\Pi = \{p_1, p_2, p_3\}$):

- If correct $= \{p_1\}$ or $\{p_1, p_2\}$, then $\mathcal{L}$ can output "true" at p_1 and p_2.
- $\Rightarrow$ The failure patterns are indistinguishable for p_1 (for any finite amount of time).
- If correct $= \{p_1\}$, then Σ has to output $\{p_1\}$.
- $\Rightarrow$ If correct $= \{p_1, p_2\}$, then Σ outputs also $\{p_1\}$ (and equivalently for p_2).
- $\Rightarrow$ Contradiction to intersection.

Relations between the Failure Detector Classes

$$
\begin{tikzcd}
\text{emulating registers in message-passing} \arrow[d] & \text{set-agreement in message-passing} \arrow[d] & \text{set-agreement in shared-memory} \arrow[d] \\
\Sigma \arrow[r, "\text{is strictly stronger } (n > 2)"'] & \mathcal{L} \arrow[r, "\text{is strictly stronger}"'] & \text{anti-}\Omega
\end{tikzcd}
$$

Conclusions:

- $\mathcal{L}$ is the weakest failure detector for message passing set-agreement.
- The proof is surprisingly simple (especially compared to the shared memory proof).
- Sometimes results in message passing are easier to prove than in shared-memory.

Outlook:

- Can this approach be extended to k-set-agreement?
- Is there some distinct type of a weak register that "belongs" to (k-)set-agreement?