On Failure Detectors and Type Boosters

Rachid Guerraoui and Petr Kouznetsov

Distributed Programming Lab

Swiss Federal Institute of Technology
in Lausanne (EPFL)

Motivation

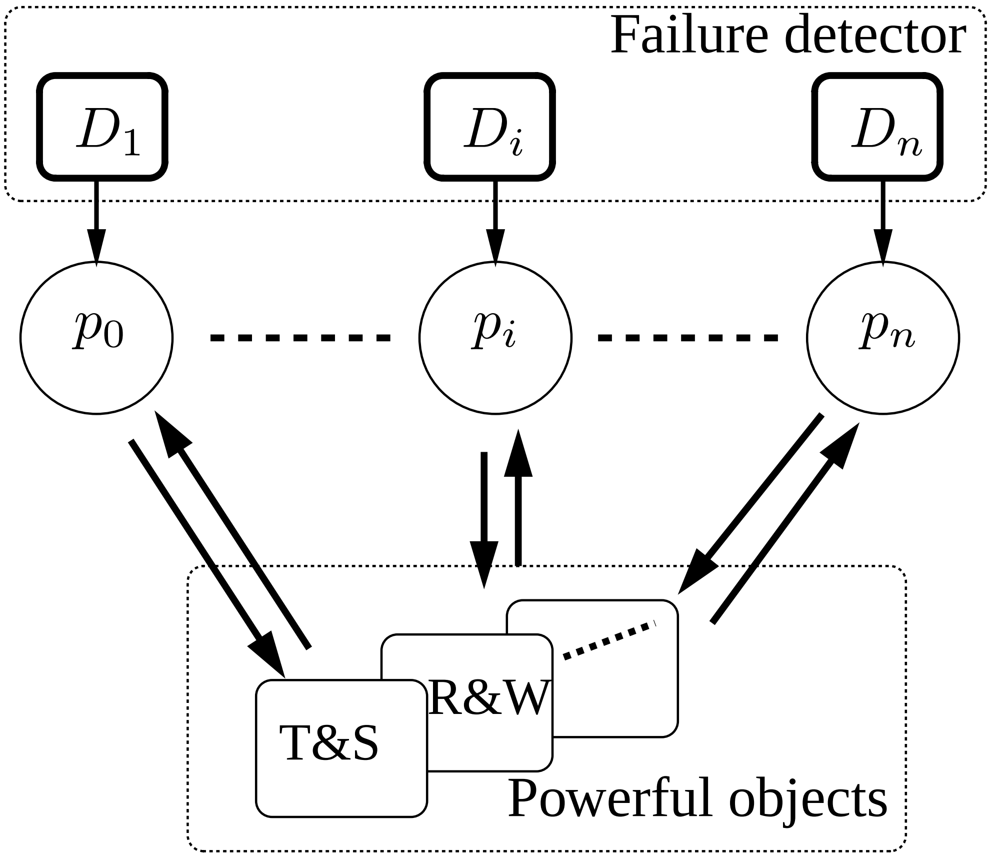
Registers are weak [FLP85, LAA87].

(1) Stronger types: queue, T&S, C&S, etc.

$cons(\mathcal{S})$ - *consensus number* of a set of types $\mathcal{S}$ [Her91, Jay97].
E.g., $cons(\mathsf{T\&S}) = 2$, $cons(\mathsf{C\&S}) = \infty$.

(2) *Failure detectors* [CT96].
Ω is the weakest failure detector for consensus [CHT96, LH94].

What if we combine the trends?



The question

- $n + 1$ processes
- Read-write memory
- Shared objects of types in $\mathcal{S}$: $cons(\mathcal{S}) = n$

What is the weakest failure detector $\mathcal{D}$ to wait-free solve consensus?

Background

[Nei95]: Ω_k outputs a set of at most k processes so that, eventually, all correct processes detect the same set that includes at least one correct process.

- $\Omega_1 \equiv \Omega$, $\Omega_{k+1} \prec \Omega_k$
- Ω_n is sufficient to solve $(n+1)$-process consensus using $\mathcal{S}$ and registers.

Is Ω_n *necessary?*

Contribution

Theorem. Ω_n *is necessary to implement wait-free* $(n+1)$*-process consensus with registers and objects of* *one-shot deterministic* *types in* $\mathcal{S}$ *such that* $cons(\mathcal{S}) \leq n$.

Corollary. Ω_n *is necessary to implement* $(n+1)$*-process wait-free consensus using registers and* $(n-1)$*-resilient objects of* *any* *types.*

A hint of the proofs

1. System model
2. Boosting consensus power
3. Boosting resilience

System model

- $n + 1$ asynchronous processes: $p_0, \ldots, p_n$
- MWMR registers
- Wait-free linearizable objects of one-shot deterministic types in $\mathcal{S}$, $cons(\mathcal{S}) \leq n$.
- A failure detector $\mathcal{D}$

Failure detectors and reducibility

- A failure detector $\mathcal{D}$ is defined as a map of each *failure pattern* F to a set of *failure detector histories* $\mathcal{D}(F)$ [CHT96]

- $\mathcal{D}$ is *weaker than* $\mathcal{D}'$ if there exists $T_{\mathcal{D}' \rightarrow \mathcal{D}}$ (a *reduction algorithm*) that emulates $\mathcal{D}$ out of $\mathcal{D}'$

Team Consensus

- Processes are partitioned (a priori) into non-empty *teams* Π_1 and Π_2.

- Agreement is ensured only if each team proposes at most one value.

$$\text{Consensus} \Leftrightarrow \text{Team Consensus}$$

Proof strategy

Assume that a failure detector $\mathcal{D}$ implements $(n+1)$-process consensus using $\mathcal{S}$ and registers. (Let A be the corresponding algorithm.)

The goal: to show that Ω_n is *weaker* than $\mathcal{D}$, i.e., to construct a reduction algorithm $T_{\mathcal{D}\rightarrow\Omega_n}$ that emulates the output of Ω_n.

Simulation tree construction (as in [CHT96])

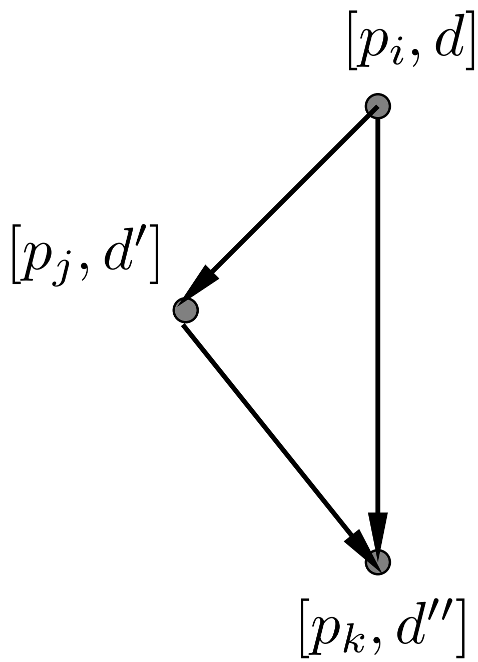
Every process p_i maintains (using registers):

(1) an ever-growing sample of the current failure detector history in the form of DAG G_i

(2) an ever-growing simulation tree Υ_i: each path in G_i induces a simulated run of A

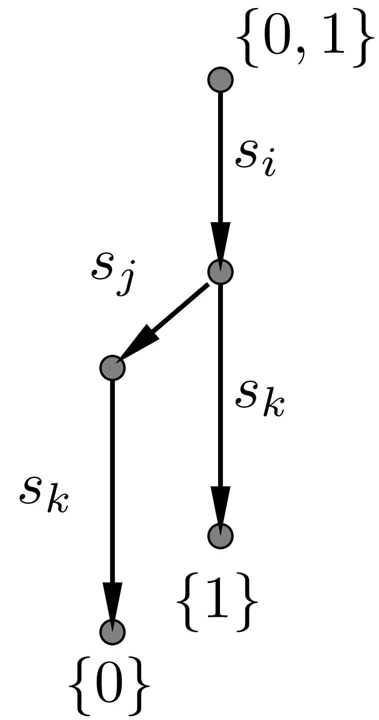
$$\exists \Upsilon : \forall p_i, \ \Upsilon_i(t) \rightarrow_{t \rightarrow +\infty} \Upsilon$$

Tagged simulation tree (as in [CHT96])



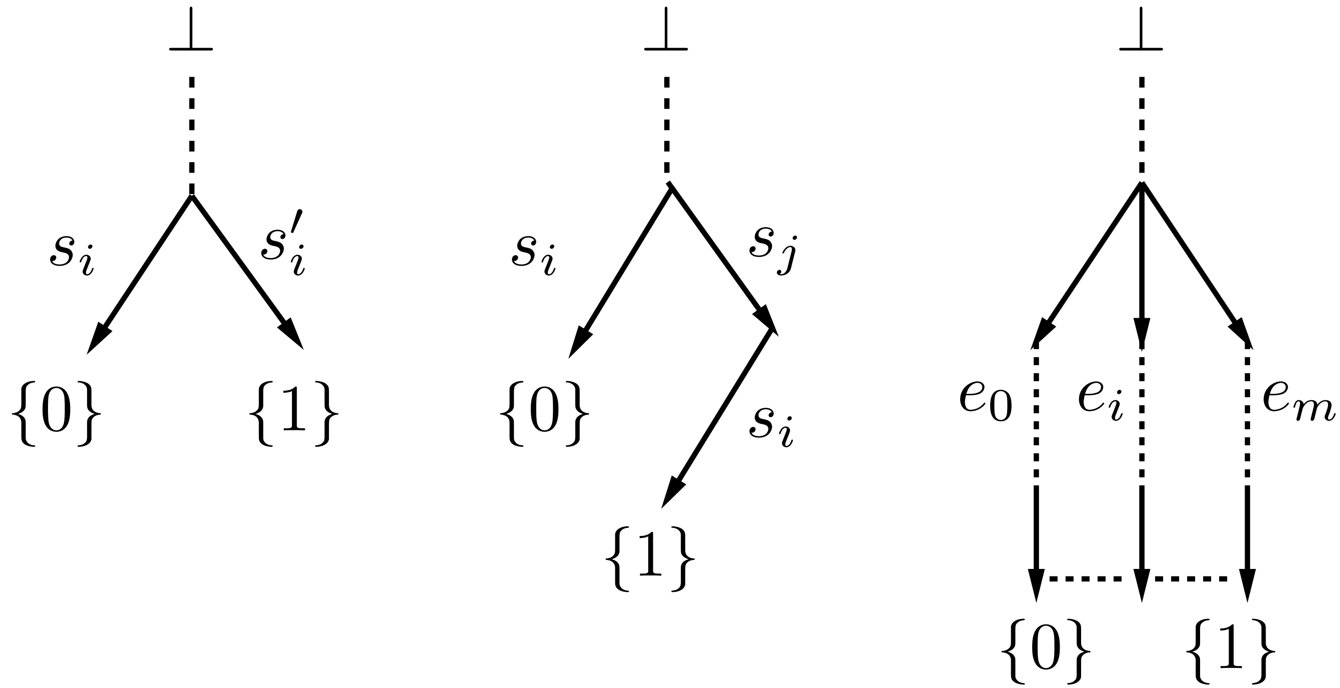
DAG G_i

$\Longrightarrow$



Simulation tree Υ_i

Finite *critical* subtrees of Υ



(a) fork (b) hook (c) rake

Deciding sets

Each critical subtree ε defines a set of at most n processes, a ***deciding set*** of ε.

Claim: The deciding set of ε includes at least one correct process. Suppose not.

(a),(b): [CHT96, LH94];

(c): $\mathcal{S}$ and registers solve $(n+1)$-process team consensus $\Longrightarrow$ $cons(\mathcal{S}) > n$ — a contradiction.

The reduction algorithm $T_{\mathcal{D}\rightarrow\Omega_n}$

Every process p_i periodically:

1. Updates G_i and Υ_i
2. Locates *the first* critical subtree ε in Υ_i
3. Outputs the deciding set of ε

Ω_n is emulated!

Corollary: boosting resilience with Ω_n

- a set $\mathcal{K}$ of $(n-1)$-resilient linearizable objects
- registers and $\mathcal{K}$ solve $(n-1)$-resilient $(n+1)$-process consensus

Then Ω_n is the weakest failure detector to implement *wait-free* $(n+1)$-process consensus using $\mathcal{K}$ and registers.

Wait-freedom vs. t-resilience [CHJT94]

For any $t < k$ and any set of types S, t-*resilient* k-process consensus can be implemented out of S and registers

if and only if

wait-free $(t + 1)$-process consensus can be implemented out of S and registers.

Corollary proof: sufficient part

- $(n-1)$-resilient objects in $\mathcal{K}$ and registers implement wait-free n-process consensus [CHJT94].
- wait-free t-process consensus objects and Ω_n implement wait-free $(n+1)$-process consensus [Nei95].

Corollary proof: necessary part

- $\mathcal{K}$ can be implemented out of wait-free n-process consensus objects [Her91, CHJT94]

- n-process consensus is a one-shot deterministic type, $cons(n\text{-process consensus}) = n$ [Her91].

- Ω_n is necessary to implement wait-free $(n+1)$-process consensus out of $\{n\text{-process consensus}, \text{register}\}$.

Open questions

- No deterministic one-shot assumption [BGA94].
- Boosting $\mathcal{S}$ ($cons(\mathcal{S}) = n$) to the higher levels (than $n+1$) of the consensus hierarchy.

(Ref: Technical report IC-EPFL ID:200348)

Questions?

References

[BGA94] Elizabeth Borowsky, Eli Gafni, and Yehuda Afek. Consensus power makes (some) sense! In *Proceedings of the 13th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 363–372, August 1994.

[CHJT94] Tushar Chandra, Vassos Hadzilacos, Prasad Jayanti, and Sam Toueg. Wait-freedom vs. t-resiliency and the robustness of wait-free hierarchies. In *Proceedings of the 13th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 334–343. ACM Press, 1994.

[CHT96] Tushar D. Chandra, Vassos Hadzilacos, and Sam Toueg. The weakest failure detector for solving consensus. *Journal of the ACM (JACM)*, 43(4):685–722, July 1996.

[CT96] Tushar D. Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM (JACM)*, 43(2):225–267, March 1996.

[FLP85] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(3):374–382, April 1985.

[Her91] Maurice Herlihy. Wait-free synchronization. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 13(1):124–149, January 1991.

[Jay97] Prasad Jayanti. Robust wait-free hierarchies. *Journal of the ACM (JACM)*, 44(4):592–614, 1997.

[LAA87] Michael C. Loui and Hosame H. Abu-Amara. Memory requirements for agreement among unreliable asynchronous processes. *Advances in Computing Research*, pages 163–183, 1987.

[LH94] Wai-Kau Lo and Vassos Hadzilacos. Using failure detectors to solve consensus in asynchronous shared-memory systems. In *Proceedings of the 8th International Workshop on Distributed Algorithms (WDAG)*, volume 857 of *LNCS*, pages 280–295. Springer Verlag, 1994.

[Nei95] Gil Neiger. Failure detectors and the wait-free hierarchy. In *Proceedings of the 14th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 100–109, August 1995.