Failure Detection Sequencers: Necessary and Sufficient Information about Failures to Solve Predicate Detection

Felix Gärtner

Laboratoire de Programmation Distribuée (LPD)
École Polytechnique Fédérale de Lausanne (EPFL)
Switzerland

fgaertner@lpdmail.epfl.ch

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

joint work with Stefan Pleisch
IBM Research, Zurich Research Laboratory, Switzerland

Some Important Results

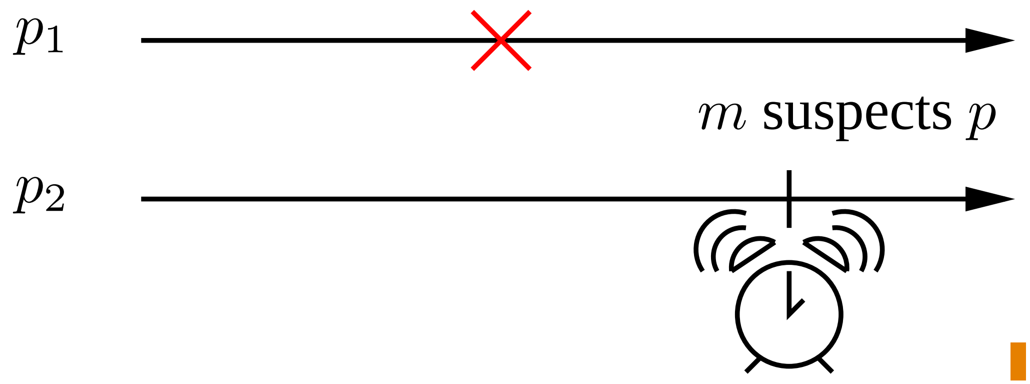
UEFA Champions League

- Deportivo La Corogne - FC Bayern München 2:1
- Bayer Leverkusen - Olympiakos Piräus 2:0
- RC Lens - AC Milan 2:1
- Maccabi Haifa - Manchester United 3:0

What ist the weakest failure detector
for solving the predicate detection problem?

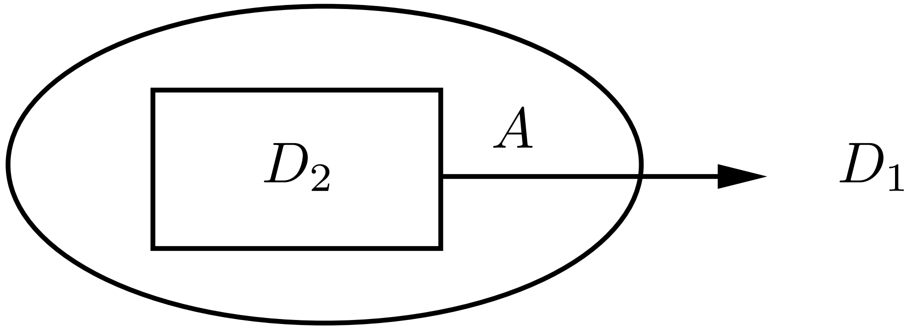
Failure Detectors [Chandra and Toueg 1996]

- Asynchronous system model with crash failures.



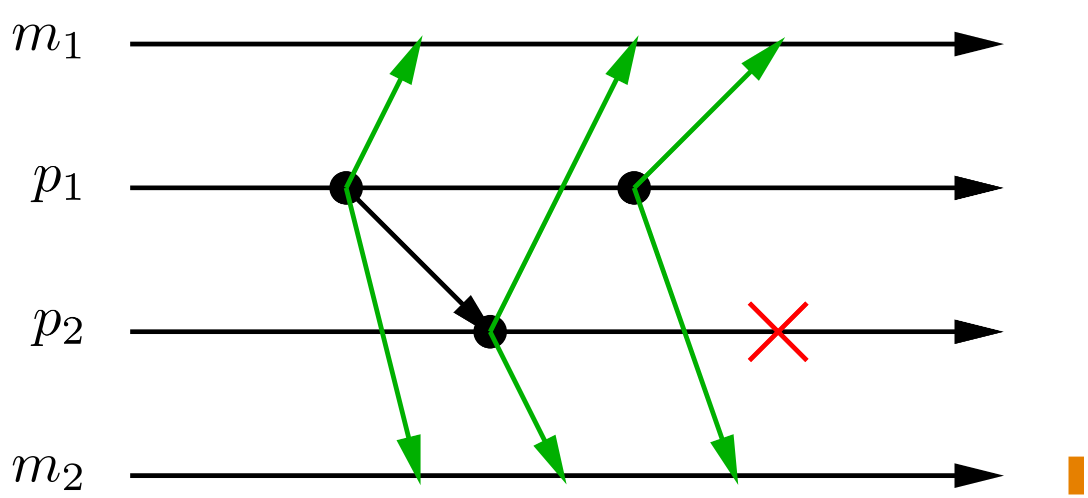
- Process p is not suspected before it crashes.
- If p crashes, it will eventually be suspected.
- Class of perfect failure detectors $\mathcal{P}$.

Weakest Failure Detectors



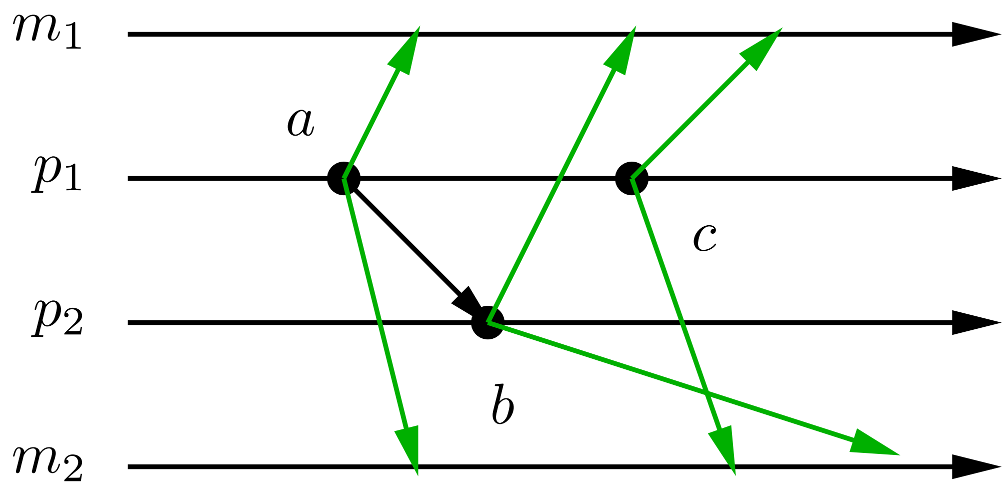
- Set $\mathcal{D}$ of all failure detectors, take $D_1, D_2 \in \mathcal{D}$.
- D_1 weaker than D_2 ($D_1 \le D_2$) if there exists an algorithm A which transforms output of D_1 into output of D_2.
- Failure detector D is weakest to solve a problem P:
  1. D allows to solve P
  2. Every failure detector D' which allows to solve P is at least as strong as D ($D' \ge D$).

Predicate Detection



- Does a global predicate φ hold throughout the computation?
  - If algorithm issues detection then φ held in computation.
  - If φ holds in computation, then eventually algorithm issues detection.

Example



- Control message is sent with every (relevant) event to all observers.
  - Global state = vector of local states = cut through space/time diagram.
  - Observation = sequence of global states.
  - φ holds = φ is true in one state during the observation.
    - Examples: "a has happened", b happened before c.
  - For simplicity just look at observer-independent predicates [Charron-Bost et al. 1995].

Outline

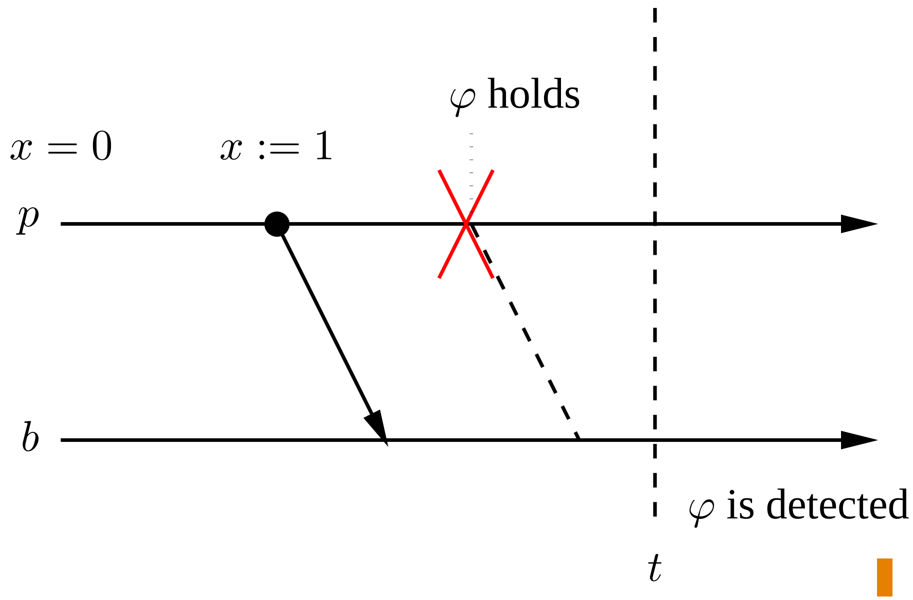
Question: What is the weakest failure detector for predicate detection?

- Bad news: There is none! Generalization of a proof by Charron-Bost, Guerraoui, and Schiper [2000].
- Define in analogy to failure detectors a “slightly” stronger device: failure detection sequencer
- A particular sequencer Σ is equivalent to predicate detection.
- Implementation of Σ in synchronous systems.
- Σ and causality.

Impossibility Proof

- Assume there is a failure detector-based algorithm A solving predicate detection.

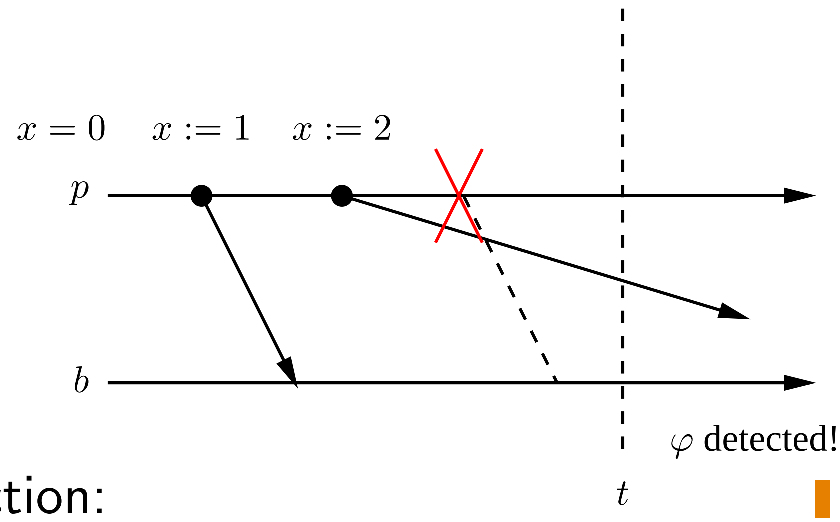
- Consider predicate $\varphi \equiv crashed \wedge x = 1$ in the following computation:



- Since A is correct and φ holds, A eventually issues detection (at time t).

Impossibility Proof (cont.)

- Consider similar scenario:



- Path to a contradiction:
  - Another event $x := 2$ occurs $x := 1$ and crash event.
  - System is asynchronous $\Rightarrow$ defer control message for some arbitrary but finite time.
  - For b at time t this scenario is indistinguishable from the previous scenario (failure detector does not help, no matter how strong it is).
  - A is deterministic: A issues detection of φ at t.
  - But: φ never held, A is not correct, a contradiction.

Failure Detection Sequencer

- Failure detector is a function of failures [Chandra and Toueg 1996]:

$$D : F \rightarrow H$$

- Failure detection sequencer is a function of failures and the computation history:

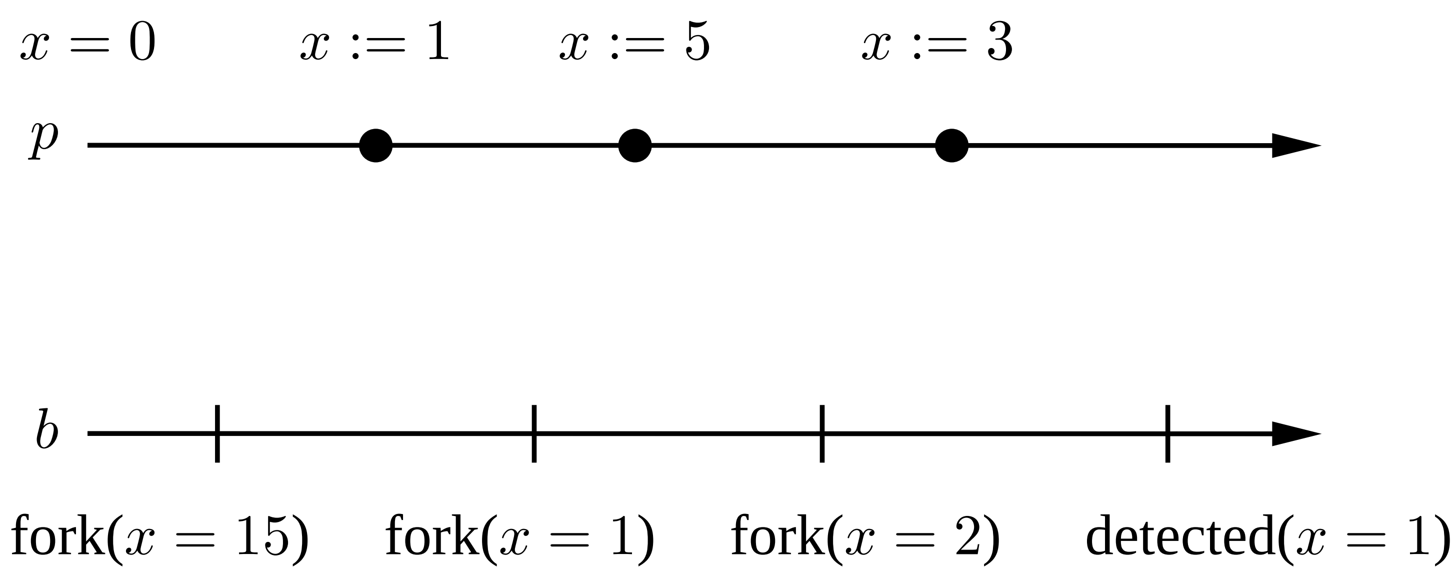
$$S : F \times C \rightarrow H$$

- Every failure detector is a failure detection sequencer (but not vice versa).
- Define a particular sequencer Σ:
  - If p crashes, it will eventually be suspected.
  - Process p is not suspected before it crashes.
  - If a process is suspected, Σ yields the final state of that process.

Σ is Equivalent to Predicate Detection

- Look at adaptive predicate detection:
  - Can issue new detection predicates at runtime via $fork(\varphi)$.
  - Detects disjunction of all given predicates.
  - Upon detection, issues $detected(\varphi)$.
- If you have an adaptive predicate detection algorithm, you can build Σ:
  - For any new state occurring in the computation, fork an appropriate new instance of predicate detection.
- Can use Σ to solve adaptive predicate detection. . .

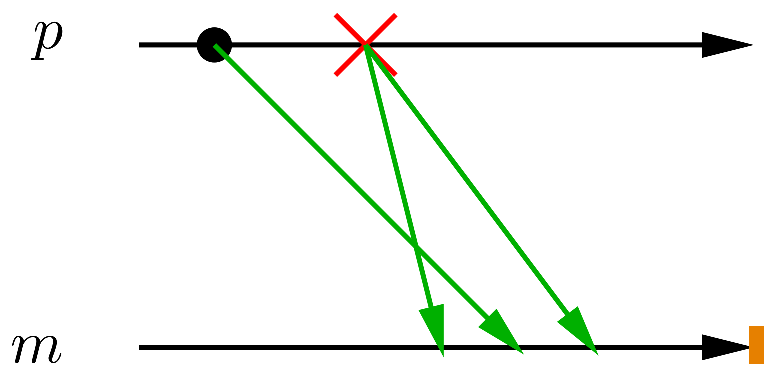
Example



- Valid behavior of adaptive predicate detection.

Σ and Causality

- Idea of solving predicate detection using Σ:
  - Use standard predicate detection techniques
  - Use Σ to "embed" crash events consistently into the causality relation
- Can "sequence" events consistently.



Implementing Σ

- Σ can be implemented in synchronous systems:
  - Piggyback most recent state on heartbeats.
  - If timeout on heartbeats runs out, no messages are in transit: return most recent state.
- Σ can be implemented using $\mathcal{P}$ and synchronous communication:
  - Send state after every step, keep most recent state.
  - If $\mathcal{P}$ suspects p, wait for communication timeout and return state.

Example: Implementation using $\mathcal{P}$ and Synchronous Communication

On every process p_j:

  with every step FIFOsend "alive in state s" *to all*

On every process p_i:

  variables:

    $D_i[1..n]$ **init** $(\bot, \ldots, \bot)$ {* sequencer output *}

    $r_i[1..n]$ **init** $(\delta, \ldots, \delta)$ {* timers *}

    $S_i[1..n]$ **init** $\langle$initial states of $p_1, \ldots, p_n\rangle$

  algorithm:

    upon FIFOreceive "alive in state s" from p_j **do**

      $\langle$reset timer $r_i[j]$ to $\delta\rangle$

      $S_i[j] := s$

    upon $\langle\mathcal{D}$ suspects $p_j\rangle$ **do**

      $\langle$reset timer $r_i[j]$ to $\delta\rangle$

    upon $\langle$expiry of timer $r_i[j]\rangle$ **do**

      if $\langle\mathcal{D}$ suspects $p_j\rangle$ **then**

        $D_i[j] := S_i[j]$

      endif

Σ and Synchronous Systems

- Synchronizer [Awerbuch 1985]:
  - Generates a sequence of rounds $r_1, r_2, \ldots$
  - At beginning of every round, a surviving process sends exactly one message to every other process.
  - Synchronizer guarantees that all messages from round r_i are received before round r_{i+1} starts.
- Synchronizers in crash-affected systems cannot be implemented even with $\mathcal{P}$.
- Using Σ you can implement a synchronizer for crash-affected systems.
- Synchronizer emulates a form of global time which is available in synchronous systems.
- Σ offers "full synchrony" without referring to a global clock.

Summary

- What is the weakest failure detector for predicate detection in crash-affected asynchronous systems?

- The mechanism which has enough power to solve predicate detection cannot be a failure detector in the formal sense of Chandra and Toueg [1996].

- One way of defining such a mechanism: Failure detection sequencer.
  - Gives final state of crashed process.
  - In practice: detect a crash only if there are no messages from that process in transit.

- Every failure detector is a sequencer, but every algorithm "is" a sequencer too:
  - The "weakest sequencer" is the problem itself.
  - Difficulty: Finding a problem which characterizes the problem in terms of failure information.

Acknowledgements

- Slides produced using pdfLaTeX and Klaus Guntermann's PPower4.

References

Awerbuch, B. 1985. Complexity of network synchronization. *Journal of the ACM 32*, 4 (Oct.), 804–823.

Chandra, T. D. and Toueg, S. 1996. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM 43*, 2 (March), 225–267.

Charron-Bost, B., Delporte-Gallet, C., and Fauconnier, H. 1995. Local and temporal predicates in distributed systems. *ACM Transactions on Programming Languages and Systems 17*, 1 (Jan.), 157–179.

Charron-Bost, B., Guerraoui, R., and Schiper, A. 2000. Synchronous system and perfect failure detector: Solvability and efficiency issues. In *International Conference on Dependable Systems and Networks (IEEE Computer Society)* (2000).