Dependable Distributed Systems
Master of Science in Engineering in Computer Science

AA 2022/2023

LECTURE 6: FAILURE DETECTION

Recap on Timing Assumptions

Synchronous

- timing assumptions are explicit on
  - Upper bounds on process execution time
  - Upper bounds on communication time
  - Existence of a common global clock

Asynchronous

- there are no timing assumptions

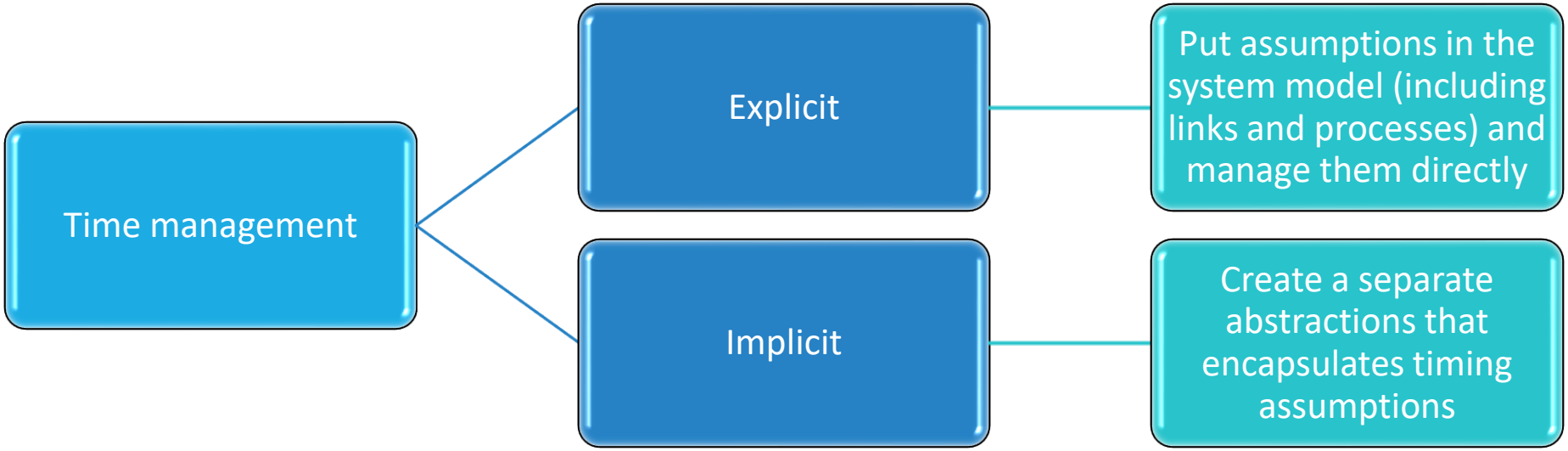
Partial synchrony

- requires abstract timing assumptions (after an unknown time t the system becomes synchronous)

Recap on Timing Assumptions

NOTE

- manipulating time inside a protocol/algorithm is complex and the correctness proof may become very involved and sometimes prone to errors



Failure Detector Abstraction

A Failure Detector is an oracle providing information about the failure state of a process

- it is a software module to be used together with process and link abstractions
- It encapsulates timing assumptions of either partially synchronous or fully synchronous system
  - Stronger are the timing assumption, more accurate the information provided by a failure detector will be

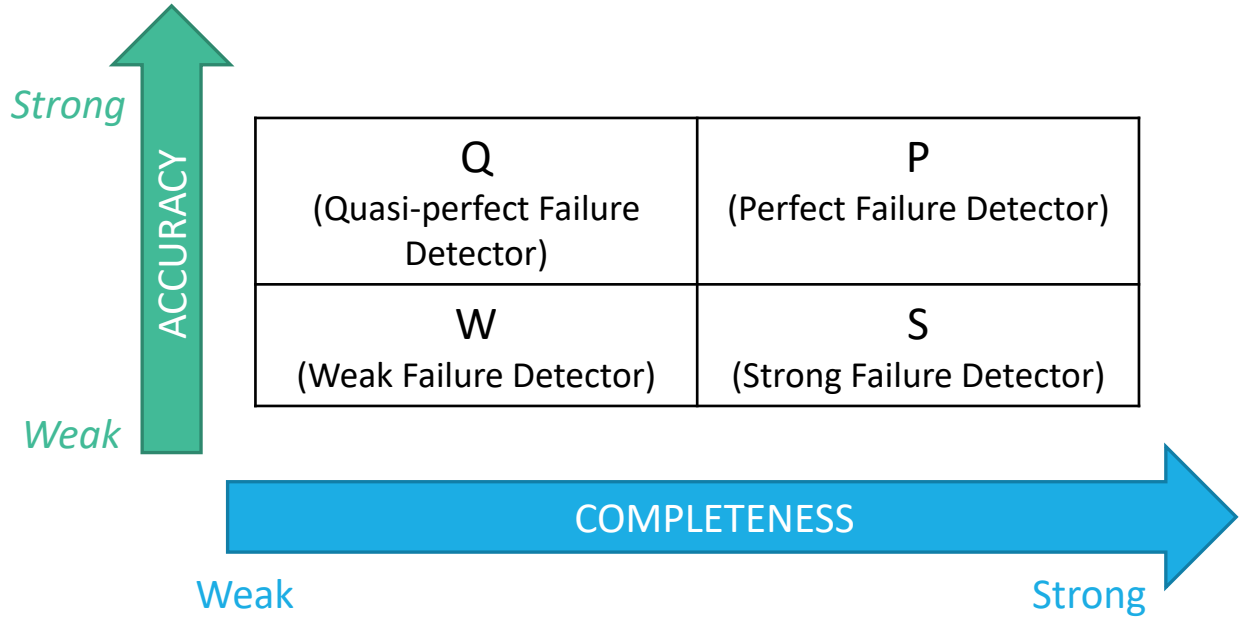
A failure detector is generally described by two properties:

- Accuracy (informally is the ability to avoid mistakes in the detection)
- Completeness (informally is ability to detect all failures)

Failure Detectors Classification

Correct processes are never suspected

Some correct process is never suspected



Eventually every process that crashes is permanently suspected by **some** correct process

Eventually every process that crashes is permanently suspected by **every** correct process

Failure Detectors Classification

OBSERVATION

- W guarantees that there is at least one correct process that is never suspected.
- Practically speaking, this could be difficult to achieve and thus it is worth to consider accuracy properties to hold eventually

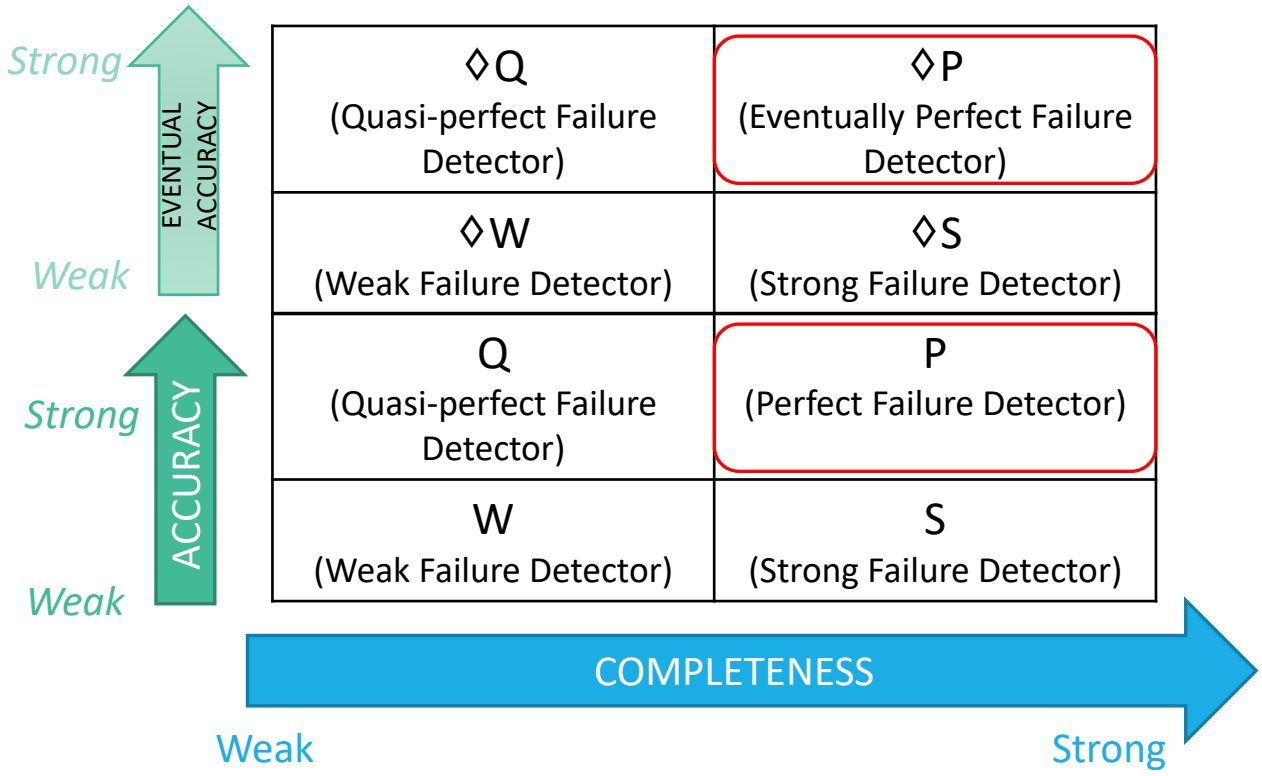
EVENTUAL STRONG ACCURACY

- There is a time after which correct processes are not suspected

EVENTUAL WEAK ACCURACY

- There is a time after which some correct process in not suspected

Failure Detectors Classification



Perfect Failure detectors (P)

System model

- synchronous system
- crash failures

Using its own clock and the bounds of the synchrony model, a process can infer if another process has crashed

Perfect failure detectors (P) Specification

Module 2.6: Interface and properties of the perfect failure detector

Module:

Name: PerfectFailureDetector, **instance** $\mathcal{P}$.

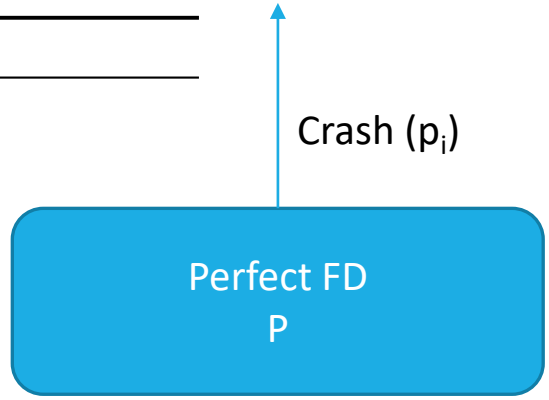
Events:

Indication: $\langle\, \mathcal{P}, \mathit{Crash} \mid p \,\rangle$: Detects that process p has crashed.

Properties:

PFD1: *Strong completeness:* Eventually, every process that crashes is permanently detected by every correct process.

PFD2: *Strong accuracy:* If a process p is detected by any process, then p has crashed.



Perfect failure detectors (P) Implementation

Algorithm 2.5: Exclude on Timeout

Implements:
PerfectFailureDetector, **instance** $\mathcal{P}$.

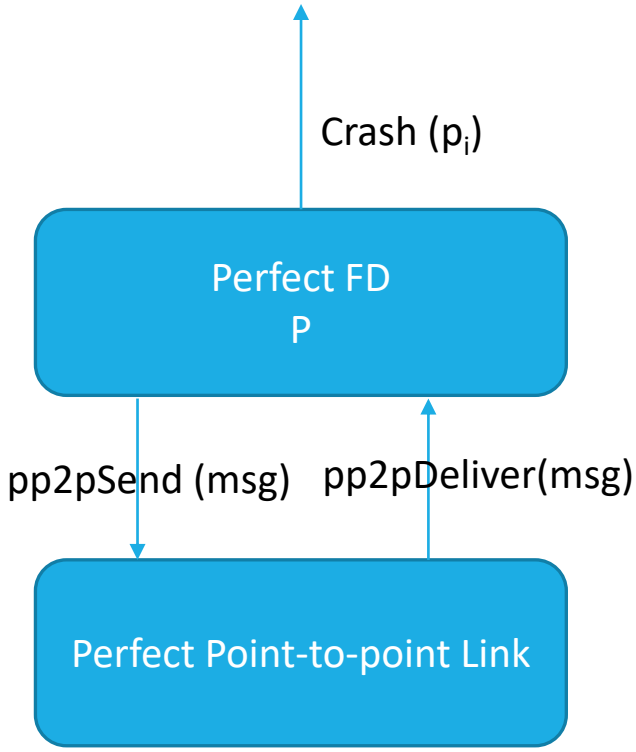
Uses:
PerfectPointToPointLinks, **instance** *pl*.

upon event $\langle\ \mathcal{P}, \mathit{Init}\ \rangle$ **do**
alive := Π;
detected := $\emptyset$;
starttimer(Δ);

upon event $\langle$ *Timeout* $\rangle$ **do**
forall $p \in \Pi$ **do**
if $(p \notin \mathit{alive}) \wedge (p \notin \mathit{detected})$ **then**
detected := *detected* $\cup \{p\}$;
trigger $\langle\ \mathcal{P}, \mathit{Crash} \mid p\ \rangle$;
trigger $\langle$ *pl*, *Send* | *p*, [HEARTBEATREQUEST] $\rangle$;
alive := $\emptyset$;
starttimer(Δ);

upon event $\langle$ *pl*, *Deliver* | *q*, [HEARTBEATREQUEST] $\rangle$ **do**
trigger $\langle$ *pl*, *Send* | *q*, [HEARTBEATREPLY] $\rangle$;

upon event $\langle$ *pl*, *Deliver* | *p*, [HEARTBEATREPLY] $\rangle$ **do**
alive := *alive* $\cup \{p\}$;



Correctness

- To prove the correctness, we must prove that both Strong Completeness and Strong Accuracy are satisfied

- What if links are fair loss?

- What if we select a timeout too long?

- What if we select a timeout too short?

Eventually perfect failure detectors (◊P)

System model

- partial synchrony
- Crash failures
- Perfect point-to-point links

Crashes can be accurately detected only after a (unknown) time *t*

- Before time *t* the systems behaves as an asynchronous one
- The failure detector may make mistake before time *t* considering correct processes as crashed.
- The notion of detection becomes suspicious

Eventually perfect failure detectors (◊P) Specification

Module 2.8: Interface and properties of the eventually perfect failure detector

Module:

Name: EventuallyPerfectFailureDetector, **instance** $\diamond\mathcal{P}$.

Events:

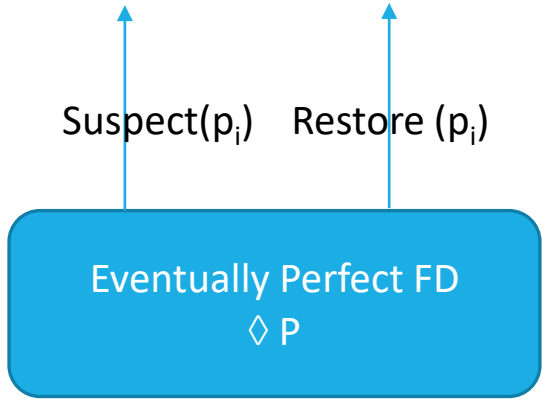
Indication: $\langle\ \diamond\mathcal{P}, \mathit{Suspect} \mid p\ \rangle$: Notifies that process p is suspected to have crashed.

Indication: $\langle\ \diamond\mathcal{P}, \mathit{Restore} \mid p\ \rangle$: Notifies that process p is not suspected anymore.

Properties:

EPFD1: *Strong completeness:* Eventually, every process that crashes is permanently suspected by every correct process.

EPFD2: *Eventual strong accuracy:* Eventually, no correct process is suspected by any correct process.



Basic constructions rules of an eventually perfect FD

Use timeouts to suspect processes that did not sent expected messages

A suspect may be wrong

- A process p_i may suspect another one p_j as the current timeout is too short

◊ P is ready to change its judgment as soon as it receives a message from p_j

- In this case, the timeout value is updated

If p_j has actually crashed, p_i does not change its judgment anymore

Eventually perfect failure detectors (◊P) Implementation

Algorithm 2.7: Increasing Timeout

Implements:
EventuallyPerfectFailureDetector, **instance** $\diamond\mathcal{P}$.

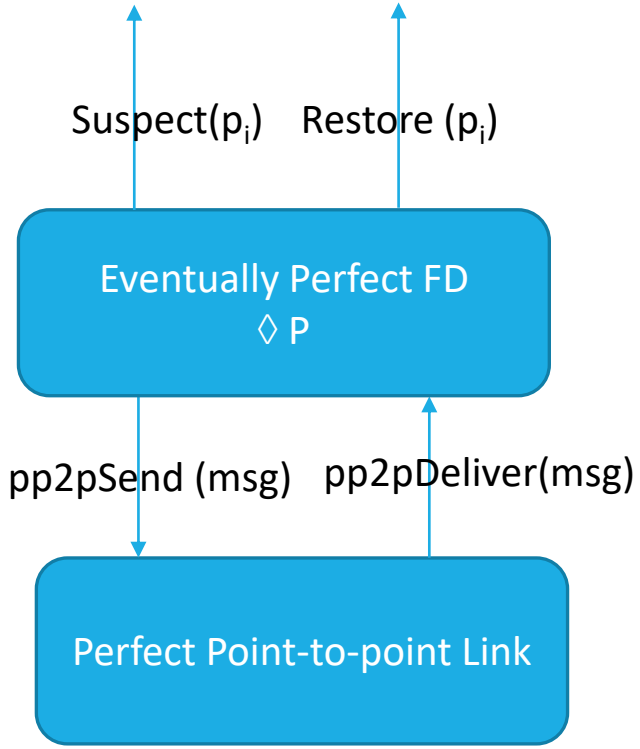
Uses:
PerfectPointToPointLinks, **instance** *pl*.

upon event $\langle\ \diamond\mathcal{P}, \mathit{Init}\ \rangle$ **do**
  $\mathit{alive} := \Pi$;
  $\mathit{suspected} := \emptyset$;
  $\mathit{delay} := \Delta$;
  starttimer(*delay*);

upon event $\langle\ \mathit{Timeout}\ \rangle$ **do**
  if $\mathit{alive} \cap \mathit{suspected} \neq \emptyset$ **then**
    $\mathit{delay} := \mathit{delay} + \Delta$;
  forall $p \in \Pi$ **do**
    if $(p \notin \mathit{alive}) \wedge (p \notin \mathit{suspected})$ **then**
      $\mathit{suspected} := \mathit{suspected} \cup \{p\}$;
      trigger $\langle\ \diamond\mathcal{P}, \mathit{Suspect} \mid p\ \rangle$;
    else if $(p \in \mathit{alive}) \wedge (p \in \mathit{suspected})$ **then**
      $\mathit{suspected} := \mathit{suspected} \setminus \{p\}$;
      trigger $\langle\ \diamond\mathcal{P}, \mathit{Restore} \mid p\ \rangle$;
    trigger $\langle\ pl, \mathit{Send} \mid p, [\text{HEARTBEATREQUEST}]\ \rangle$;
  $\mathit{alive} := \emptyset$;
  starttimer(*delay*);

upon event $\langle\ pl, \mathit{Deliver} \mid q, [\text{HEARTBEATREQUEST}]\ \rangle$ **do**
  trigger $\langle\ pl, \mathit{Send} \mid q, [\text{HEARTBEATREPLY}]\ \rangle$;

upon event $\langle\ pl, \mathit{Deliver} \mid p, [\text{HEARTBEATREPLY}]\ \rangle$ **do**
  $\mathit{alive} := \mathit{alive} \cup \{p\}$;



Correctness

Strong completeness

- If a process crashes, it will stop to send messages. Therefore the process will be suspected by any correct process and no process will revise the judgement.

Eventual strong accuracy

- After time T the system becomes synchronous. i.e., after that time a message sent by a correct process p to another one q will be delivered within a bounded time. If p was wrongly suspected by q, then q will revise its suspicious.

References

C. Cachin, R. Guerraoui and L. Rodrigues. Introduction to Reliable and Secure Distributed Programming, Springer, 2011

- Chapter 2 – from Section 2.6.1 to Section 2.6.5

T. Chandra, S. Toueg *Unreliable Failure Detectors for Reliable Distributed Systems*

- https://dl.acm.org/doi/pdf/10.1145/226643.226647