

Department of Computer Science

PURDUE
UNIVERSITY

CS505: Distributed Systems

Lecture 7: Failure Detectors

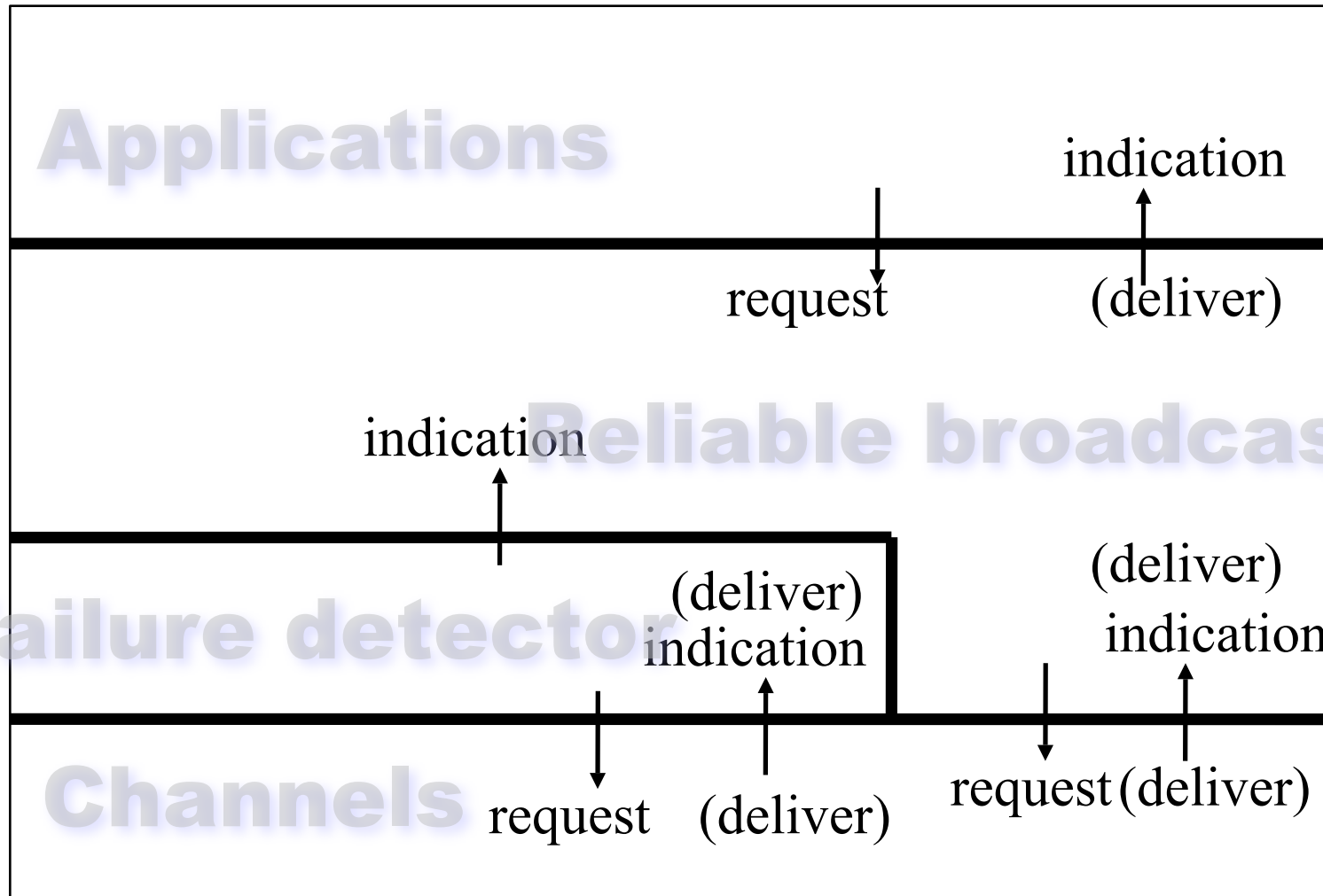
Outline

- ▶ **Definitions**
- ▶ **Failure detector classification and comparison**
- ▶ **An alternative definition**
- ▶ **Implementation**

Concept

- ▶ **A failure detector is a distributed module that provides processes with *suspicious* about crashed processes**
 - Outputs a *list of suspected processes*
- ▶ **It is a module implemented using (i.e., it encapsulates) timing assumptions**
 - Assumptions are confined within single module
 - Decisions throughout algorithm are based on same module
 - E.g., point-to-point channels, broadcast
- ▶ **According to the timing assumptions, the suspicions can be accurate or not**

Example



Properties

▶ **Traditionally [Chandra&Toueg'96] define two properties for failure detectors**

1. *Completeness*

- The degree to which failures (crashes) are indeed detected
- Reflects liveness

2. *Accuracy*

- The degree to which false suspicions are limited
- Reflects safety

▶ **Focus on process failures**

- No information on communication channels

Formally

- ▶ **Processes** $\Pi = \{p_1, p_2, \dots\}$
 - Processes fail by halting
- ▶ **Clock ticks range over** T
- ▶ **Failure pattern** F is function $T \rightarrow 2^\Pi$
 - Represents set of processes failed by time t
 - No recovery: $F(t) \subseteq F(t+1)$
 - $crashed(F) = \bigcup_{t \in T} F(t)$
 - $correct(F) = \Pi - crashed(F)$

Failure Detector D

- ▶ Failure detector history H is function $\prod \Pi \times T \rightarrow 2^{\Pi}$
 - Outputs for each process a set of suspected processes
 - $H(p, t)$ is value of H for process p at time t
 - Thus $H(p_1, t)$ and $H(p_2, t)$ can diverge at time t

- ▶ A failure detector D provides information on F in an execution
 - When queried by p at t , the failure detector module outputs $D_p = H(p, t)$

- ▶ A failure detector D maps F to sets of possible failure detector histories

Completeness

- ▶ **Bounds the amount of false negatives**
- ▶ ***Strong Completeness***
 - There is a time after which every process that crashes is suspected by every correct process
- ▶ ***Weak Completeness***
 - There is a time after which every process that crashes is permanently suspected by *some* correct process
- ▶ **“Eventual” is inherent**
 - No bounds on transmission delays...
- ▶ **Not sufficient on its own**
 - A failure detector which invariably suspects all processes is strongly complete

Accuracy

- ▶ **Bounds the amount of false positives**
- ▶ ***(Perpetual) Strong Accuracy***
 - No process is suspected before it crashes
- ▶ ***(Perpetual) Weak Accuracy***
 - Some correct process is never suspected (at least one correct process is never suspected)
- ▶ ***Eventual Strong Accuracy***
 - There is a time after which correct processes are not suspected by any correct process
- ▶ ***Eventual Weak Accuracy***
 - There is a time after which some correct process is never suspected by any correct process
- ▶ **Eventual properties reflect alternating unstable and stable phases**
 - Ensure safety together with majority assumption in unstable phases
- ▶ **Not sufficient on its own**
 - A failure detector which never suspects any process is strongly accurate

Some Failure Detectors

▶ Perfect P

- Strong Completeness
- Strong Accuracy

Corresponds to synchronous system

▶ Eventually perfect $\diamond P$ (“diamond” P)

- Strong Completeness
- Eventual Strong Accuracy

Corresponds to partially synchronous system

Perfect Failure Detector

- ▶ A perfect failure detector has strong accuracy and strong completeness
- ▶ This is an abstraction
- ▶ It is impossible to have a perfect failure detector
- ▶ We have to live with *unreliable failure detectors*
 - I.e., failure detectors which can make mistakes

Overview

Accuracy

Completeness

	Strong	Weak	Eventual Strong	Eventual Weak
Strong	<i>Perfect</i> P	<i>Strong</i> S	<i>Eventually Perfect</i> $\diamond P$	<i>Eventually Strong</i> $\diamond S$
Weak	Q	<i>Weak</i> W	$\diamond Q$	<i>Eventually Weak</i> $\diamond W$

Let D and D' be Two Failure Detectors

- ▶ D implements D' if there exists an algorithm that implements/emulates D' using D
- ▶ D is weaker than D' if D' implements D ($D \leq D'$)
 - D' is stronger than D
- ▶ D and D' are equivalent if each is weaker than the other ($D \cong D'$)
- ▶ D is strictly weaker than D' if D is weaker than D' but not inversely ($D < D'$)
- ▶ Immediately

$$X \geq \diamond X, P \geq Q, S \geq W, \diamond P \geq \diamond Q, \diamond S \geq \diamond W$$

Completeness Revisited

► Suppose the following algorithm $T_{D \rightarrow D'}$

periodically process p :

`send(D_p)` to every process q

when `receive(D_q)` from q :

`output` \leftarrow `output` $\cup D_q \setminus \{q\}$

▶ Claim

- The simple algorithm $T_{D \rightarrow D'}$ transforms any detector with weak completeness into strong completeness

▶ Implementable?

- Channel?
- Reliable broadcast?

▶ Does it preserve accuracy?

▶ (Completeness)

- Let p be any process that crashes. If eventually some correct process permanently suspects p in H_D , then eventually all correct processes permanently suspect p in `output`.

▶ (Perpetual Accuracy)

- Let p be any process. If no process suspects p in H_D before time t , then no process suspects p in `output` before time t

▶ (Eventual Accuracy)

- Let p be any correct process. If there is a time after which no correct process suspects p in H_D , then there is a time after which no correct process suspects p in `output`

Thus

- ▶ $P \cong Q, S \cong W, \diamond P \cong \diamond Q, \diamond S \cong \diamond W$
- ▶ Now we mainly need to care about
 - Eventual vs perpetual accuracy (weak or strong)
 - $\diamond S$ vs $S, \diamond P$ vs P

The Weakest Failure Detector

- ▶ A failure detector D is said to be the *weakest* to implement an abstraction A if
 - D implements A
(one can exhibit an algorithm implementing A using D only)
 - Any failure detector D' that implements A implements D
(D' is stronger than D)

An Alternative Proposition

▶ Failure detector Ω [CHT'96]

- Outputs a process q at every process p (we say that p trusts q)
- Ensures the following property

▶ Eventual Unique Leader

Eventually the same correct process is permanently trusted by every correct process

▶ The process that is trusted can be changing until some eventual time

Ω vs S/P?

▶ Can one implement Ω with any of the “traditional” failure detectors, e.g., $\diamond S$?

▶ Consider the following algorithm:

`output` $\leftarrow \Pi \setminus \Omega_p$

▶ Thus we have that $\diamond S \leq \Omega$

▶ How about implementing Ω with $\diamond S$?

From $\diamond S$ to Ω

- ▶ Every process p maintains a counter per process q , incremented whenever p suspects q
- ▶ Every process periodically broadcasts its counters
- ▶ A process that receives a counter for process q sets its counter to the max of the new counter and its own for q
- ▶ The process that is put in `output` (i.e., trusted) is the one with the smallest counter (and index)
- ▶ Do we need Reliable Broadcast or will multisend do?

Implementation

- ▶ **We still need to implement the failure detector**
- ▶ **Need to separate assumptions on the system from the mechanisms**
 - Usually links considered eventually synchronous
- ▶ **E.g. [ADGFT'03]**
 - Any number of processes may crash
 - Only the output links of an unknown correct process are eventually timely (all other links can be asynchronous and/or lossy)

Heartbeat Failure Detector

- 1. Processes periodically exchange heartbeat messages**
- 2. A process sets a timeout based on some heuristic**
- 3. A process suspects another process if it timeouts that process**
- 4. A process that delivers a message from a suspected process revises its suspicion and increases its timeout**

In Perspective

▶ Push

- Processes keep sending heartbeats “I am alive”

▶ Pull

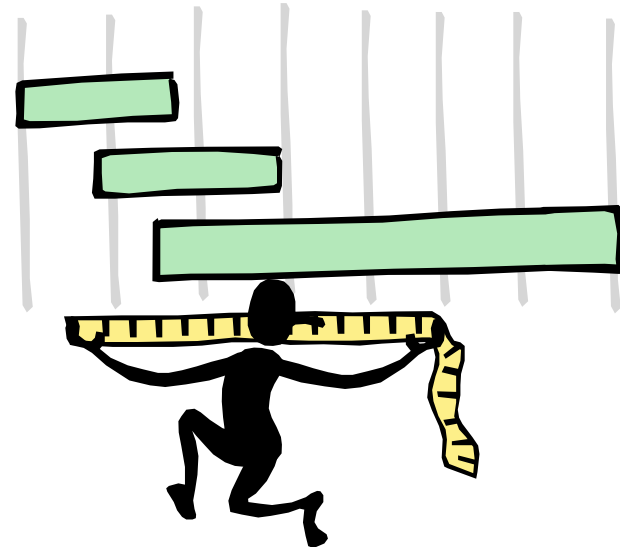
- Failure detector asks individual processes “are you alive?”
- If no answer “I am alive” is received from some process after a given time, the process is suspected

▶ What are advantages and disadvantages of these two approaches?



Metrics for Failure Detectors

- ▶ **Detection time**
- ▶ **Mistake recurrence time**
- ▶ **Mistake duration**
- ▶ **Average mistake rate**
- ▶ **Query accuracy probability**
- ▶ **Good period duration**
- ▶ **Network load**



References

- ▶ ***Unreliable Failure Detectors for Reliable Distributed Systems.*** T.D. Chandra and S. Toueg, JACM, 43(2): 225-267, 1996.
- ▶ ***The Weakest Failure Detector for Solving Consensus.*** T.D. Chandra, V. Hadzilacos, and S. Toueg. JACM 43(4):685-722, 1996.
- ▶ ***A Short Introduction to Failure Detectors for Asynchronous Distributed Systems.*** M. Raynal, SIGACT News, 36(1): 53-70, 2005.
- ▶ ***On Implementing Omega with Weak Reliability and Synchrony Assumptions.*** M.K. Aguilera, C. Delporte-Gallet, H. Fauconnier and S. Toueg, PODC '03, 306-314.