Revisiting Failure Detection for Grid Systems

Xavier DÉFAGO

[1] *School of Information Science, Japan Adv. Inst. of Science & Tech.* ***(JAIST)***

[2] *PRESTO, Japan Science & Tech. Agency* ***(JST)***

IFIP WG 10.4 – Summer 2005 meeting – July 2005. Hakone, Japan.

Acknowledgements

- **Naohiro HAYASHIBARA**
  - now at Tokyo Denki University
- **Péter URBÁN**
- **Rami YARED**
- **Takuya KATAYAMA**

- **... and many people through enlightening discussions**

JAIST

JST

Related Projects

- **COE program "Trustworthy e-Society"**
- **PRESTO, JST "Information & Systems"**
- **Jinzai Yosei "Dependable Internet"**

- **OBIGrid**
  - Bioinformatics Grid; RIKEN & AIST
- **StarBED Internet Emulator**
- **OurGrid, PlanetLab.**

JAIST

JST

Grid Systems

- **What Grid?**
  - Data-G, computational-G, domain-G, ..., *-Grid
- **What is the/a Grid?**
  - Structured Internet?
  - Loosely coupled global / enterprise network?
  - Decentralized distributed OS?
- **Key point**
  - Virtualizing of resources, ...
  - “Glue” between resources: i.e., distributed system

JAIST

JST

Grid Systems & Fault-Tolerance

- **Needs**
  - 24/7 operation,
  - reliability & availability,
  - self-managing, auto-configuration,...
  - security, accountability,...
- **Current Reality**
  - ... a LOOOONG way to go!

JAIST

JST

Failure Detection in Grid

- **Failure detection**
  - ability to detect failed components
  - prevents blocking forever
  - basic mechanism for fault-tolerance
- **Failure detection as service**
  - E.g., [Stelling et al. 1998], [van Renesse et al. 1998],...
  - E.g., NTP for clock synchronization

JAIST

JST

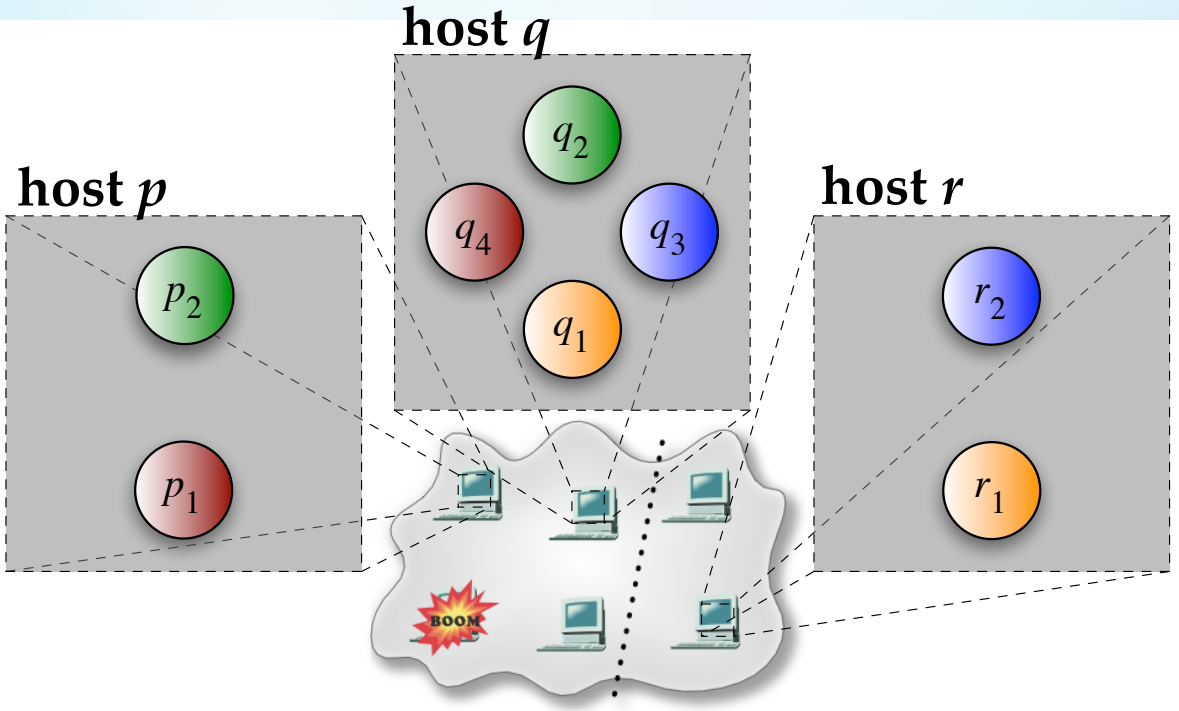
Failure Detection as Service

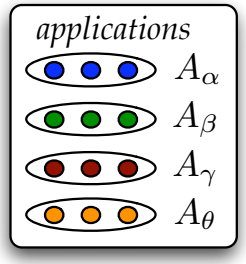
- **Current situation**
  - ad hoc detection rather than service
  - hardcoded timeouts in programs
  - hidden behind heavy abstractions
  - “proprietary” mechanisms
- **Open challenges (highly opiniated)**
  - proper abstractions, QoS negotiation
  - unattended management
  - reduction of overhead, scalability

JAIST

JST

Simult. Indep. Requirements

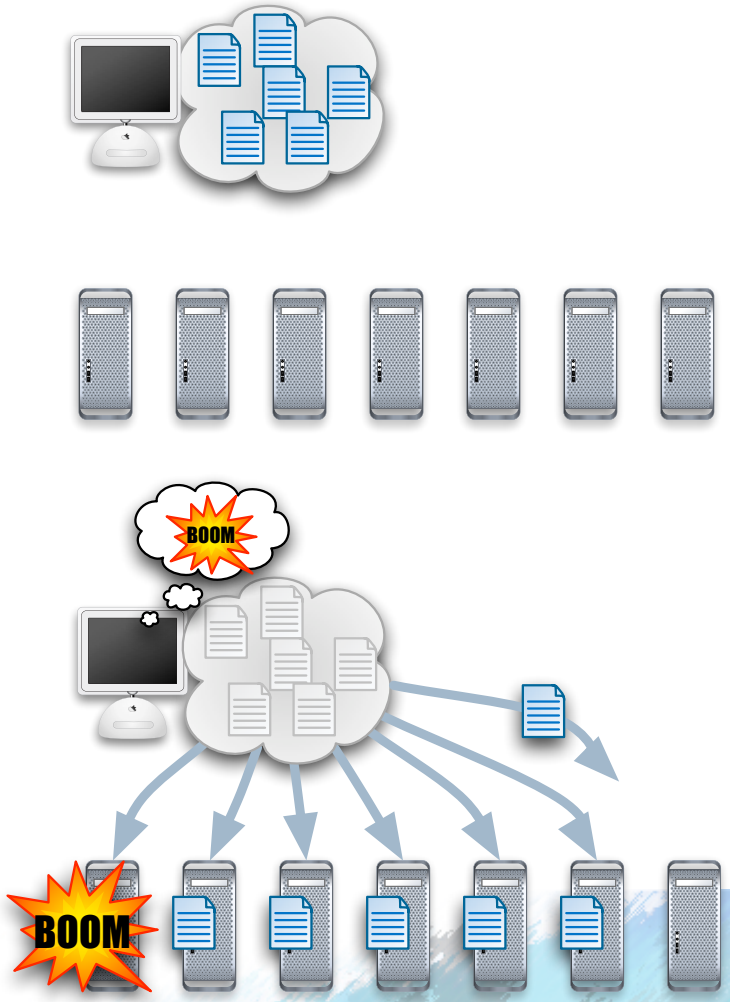




- **Large-scale systems**
  - Many distributed applications simultaneously
  - Different requirements

Example / Motivation

- **Simple case**
  - "Bag-of-Tasks" computations
  - Dispatch tasks
  - Wait for results
- **Environment**
  - **Partial failures**
  - Heterogeneous
  - Unpredictable comm.

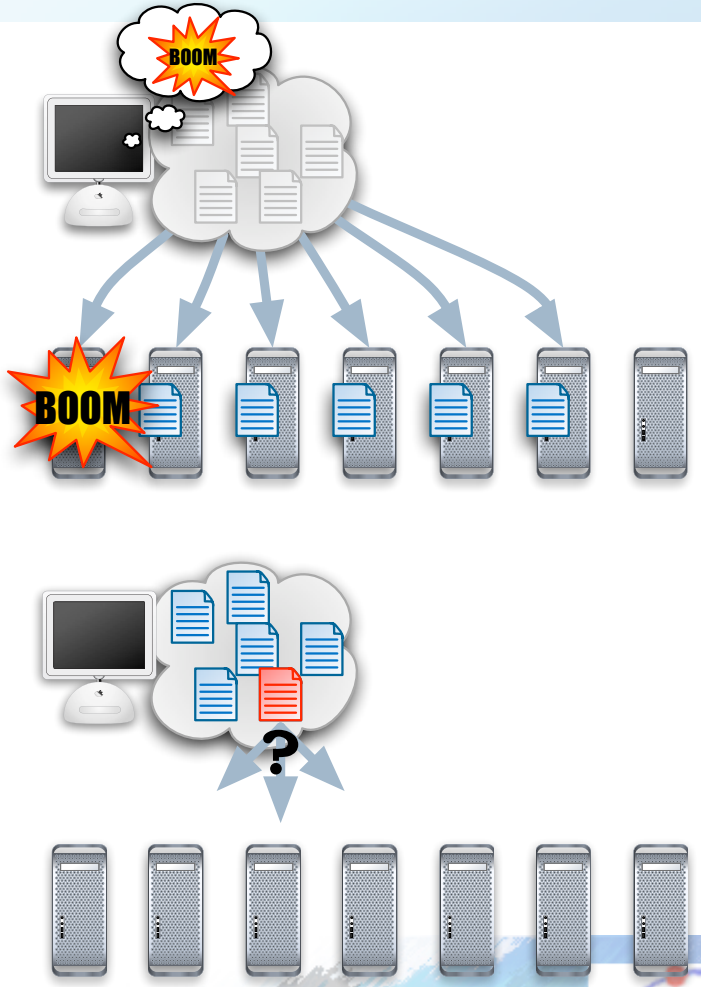


JAIST

JST

Usage Patterns

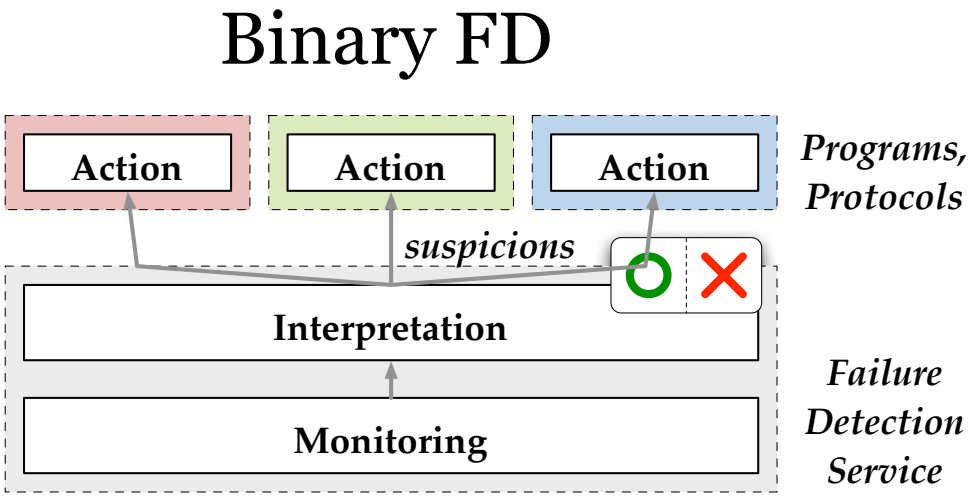
- **Case 1:**
  - Cost varies with time:
    - amount work completed
    - available resources



JAIST

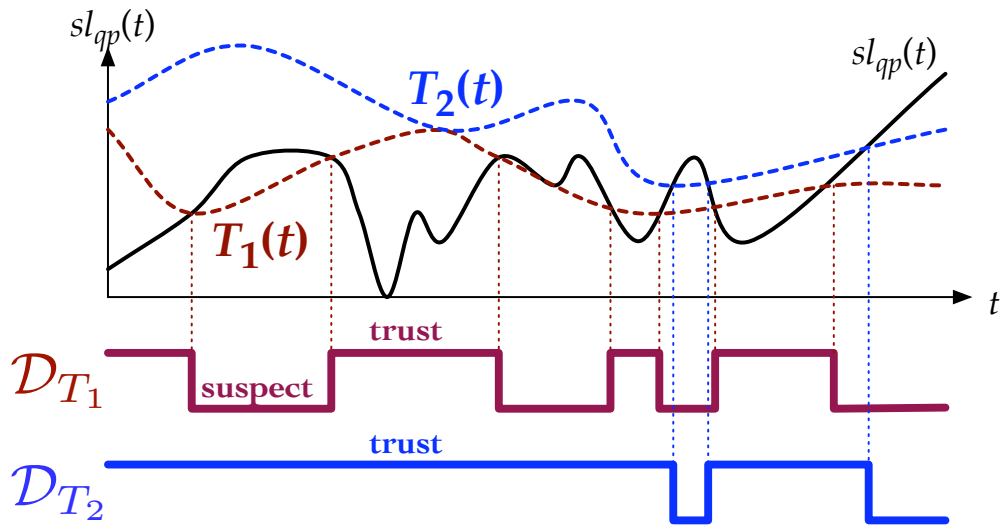
Abstractions

Accrual Failure Detectors



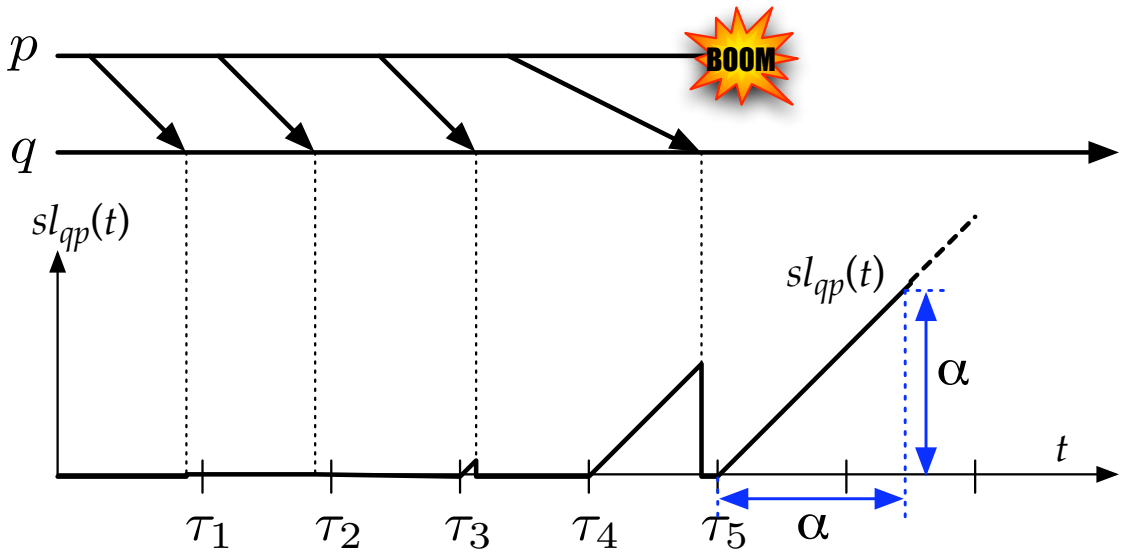
- **Accrual failure detection** **[Hayashibara; PhD 2004]**
  - 2 roles: *monitoring, interpretation*
  - interpretation –> QoS
  - => **decoupling**

Accrual Failure Detectors



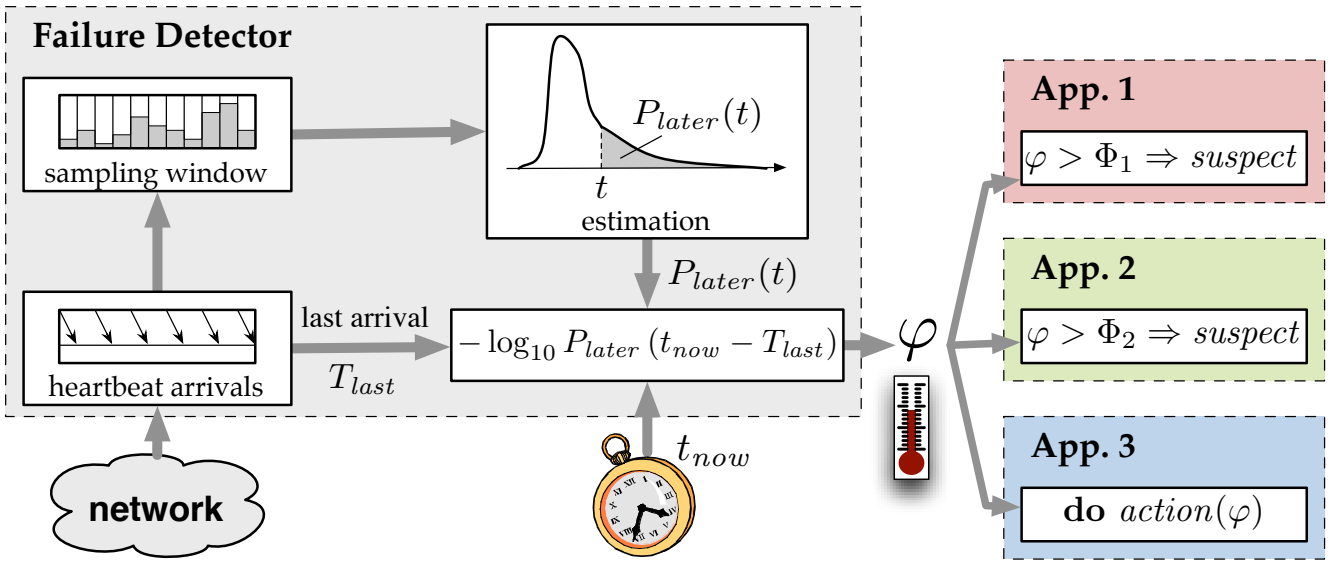
- **Accrual FD abstraction** **[Défago et al.; DSN 2005]**
  - combine different QoS
  - properties; relation w/FD theory

Chen FD as Accrual



- **Chen-based adaptation** **[Chen et al.; TC 2002]**
  - After freshness point, increase with time
  - **Reset** when receive heartbeat
  - Safety margin α set with threshold

φ Accrual FD



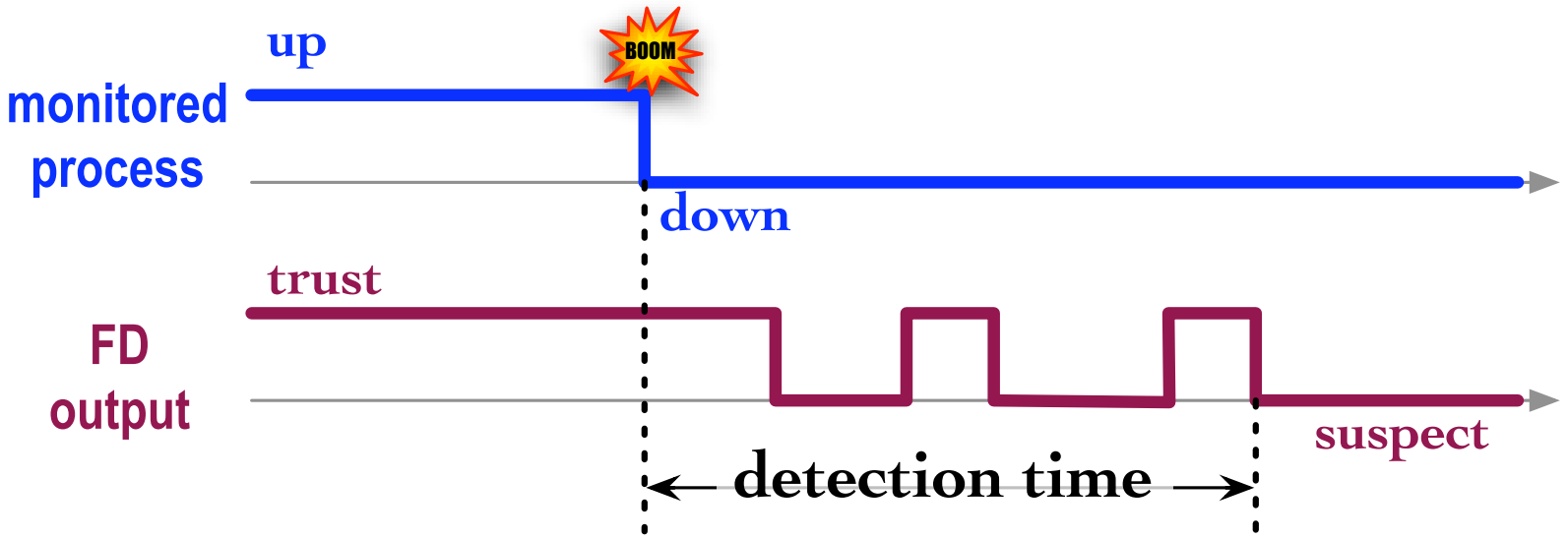
- **φ failure detector** **[Hayashibara et al.; SRDS 2004]**
  - Heartbeat based, estimate arrival distribution

QoS of Failure Detectors

JAIST

JST

QoS of Failure Detectors

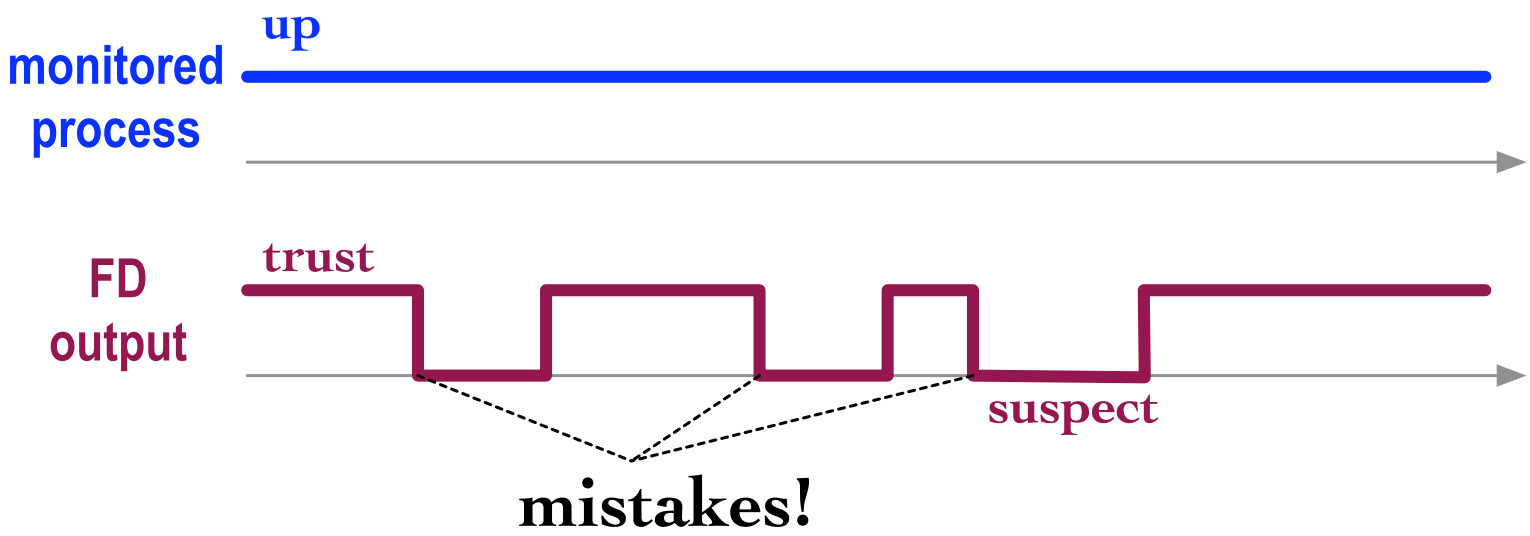


- **Metrics**

  when p faulty:

  - Detection time

QoS of Failure Detectors

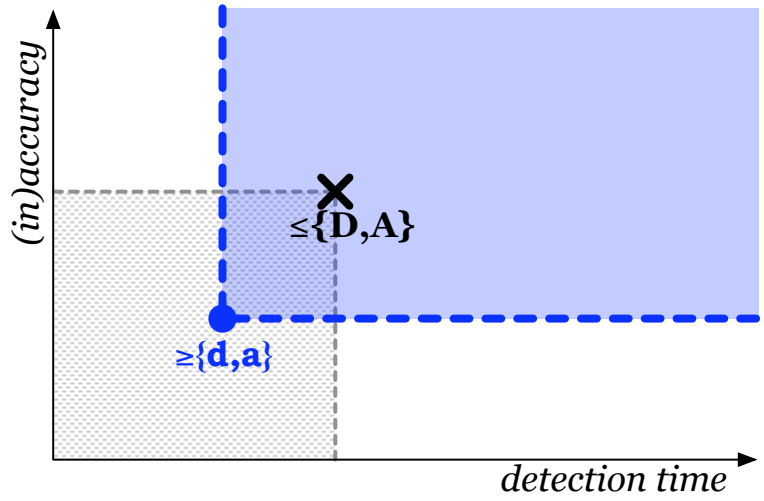


- **Metrics (accuracy)**

  when p correct:

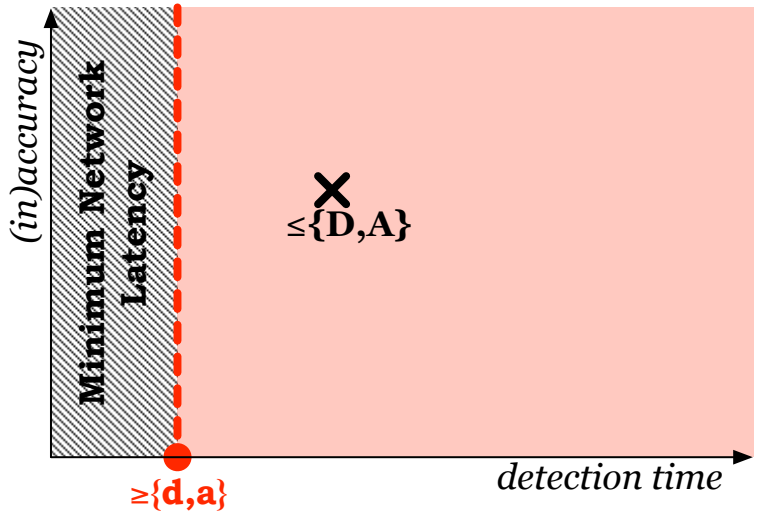
  - average mistake rate
  - query accuracy prob.
  - good period duration

Requirements vs. Guarantees



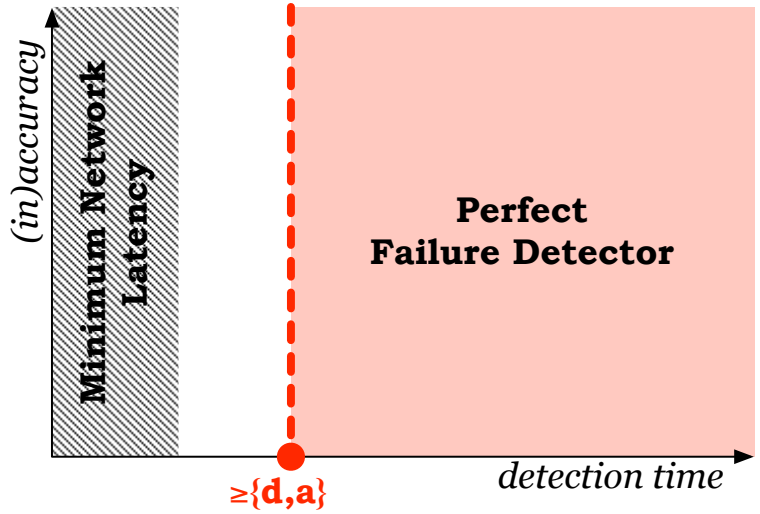
- **Application requirements**
  - ≤{D,A} : max. detect. time, max. mistakes
- **FD QoS**
  - ≥{d,a} : effect. detection time, effect. mistakes

In a Perfect World



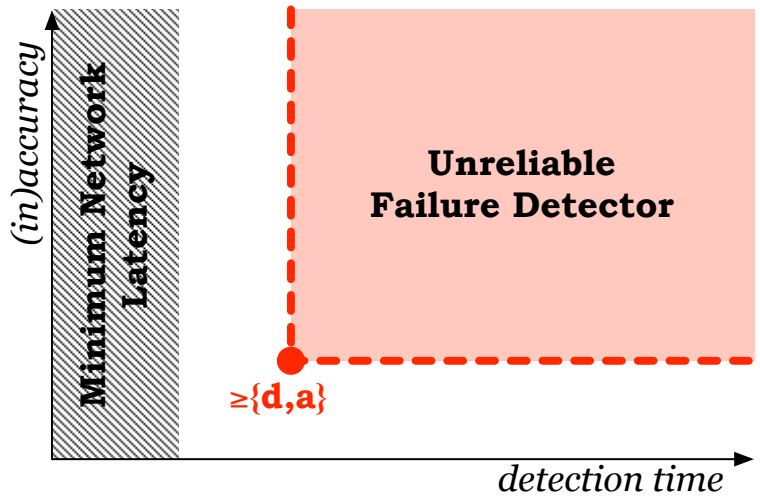
- **Ideal**
  - FD limited by min. network latency
  - "acceptable" network/system load

In a Perfect World



- **Perfect FD**
  - "realistic" detection time
  - absolute accuracy (no mistakes)
  - (some failure types **can** be detected perfectly)

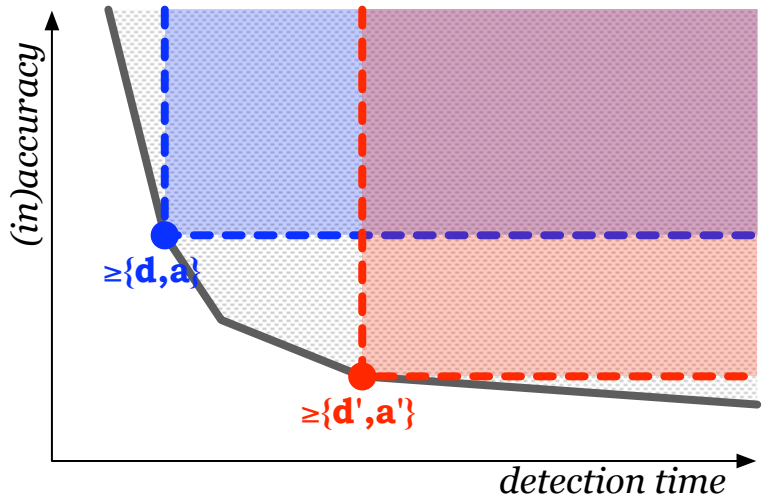
In a Less Perfect World



- **Unreliable FDs**
  - "realistic" detection latency
  - imperfect accuracy

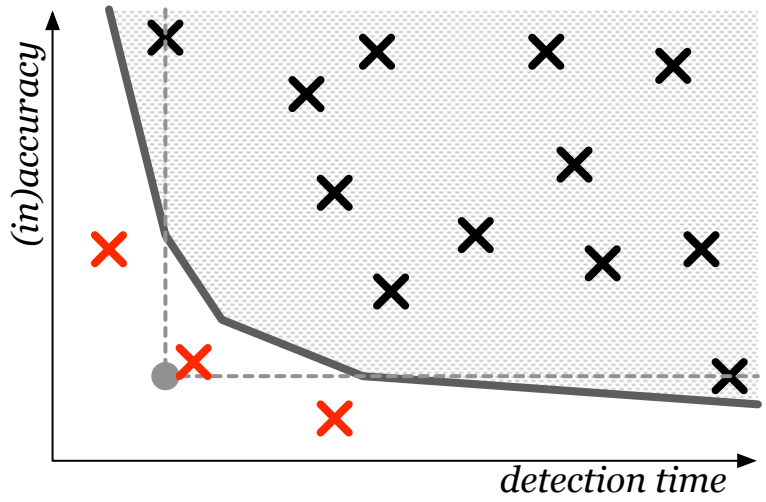
JAIST

Parametric Failure Detector



- **Parametric FDs**
  - Parameter value defines FD best QoS
  - E.g., Chen FD,...
  - Tradeoff: accuracy <-> detection latency

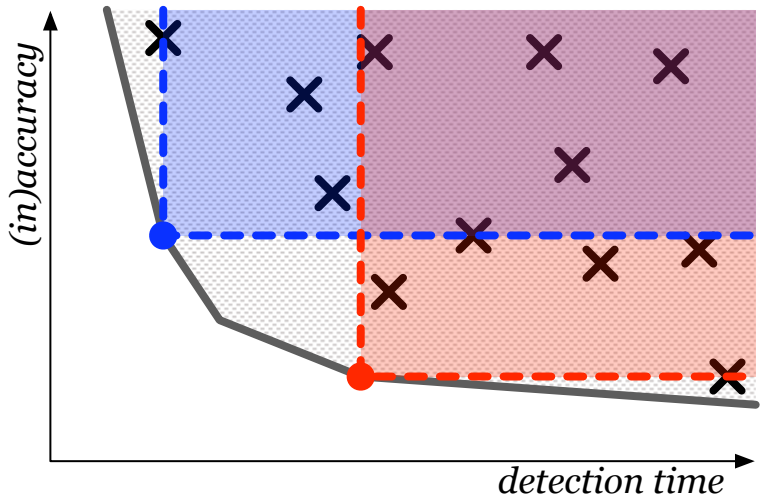
QoS Coverage



- **Coverage of FD**
  - FD **could** be tuned to support app. req.
  - Measure of FD

JAIST

Dynamic QoS Coverage



- **Approximate coverage**
  - Instantiate several QoS sets
  - Find minimal set; minimal change

JAIST

Experimentation

Comparative Analyses

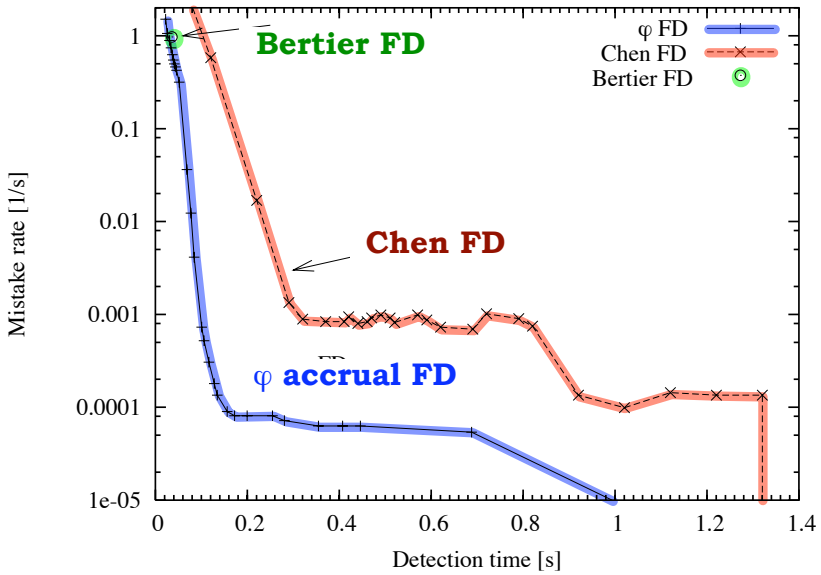
- **3 FD implementations**
  - Chen FD ; [Chen et al.] (FTCS 2000; TC 2002)
  - Bertier FD ; [Bertier et al.] (DSN 2002)
  - PHI accrual FD ; [Hayashibara et al.] (SRDS 2004)
- **Goal**
  - “Realistic” executions (e.g., LAN, WAN)
  - Identify QoS coverage

JAIST

JST

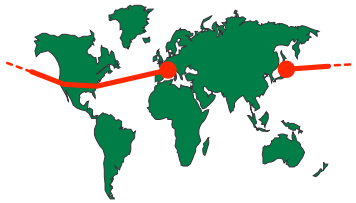
Experimentation: LAN

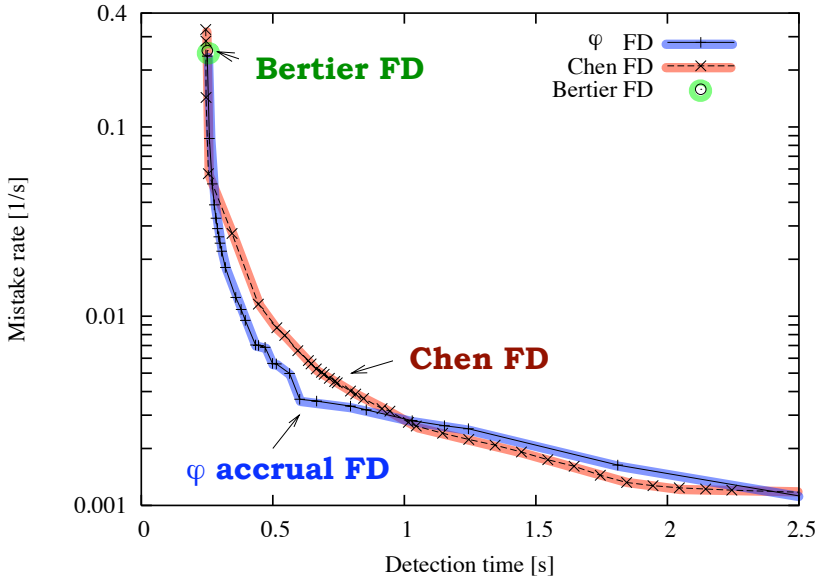
- **LAN**
  - single FastEther hub
- **Parameters**
  - HB interval: 20 ms
  - Duration: 5½ hour
  - Total HB: 1’000’000
  - no loss



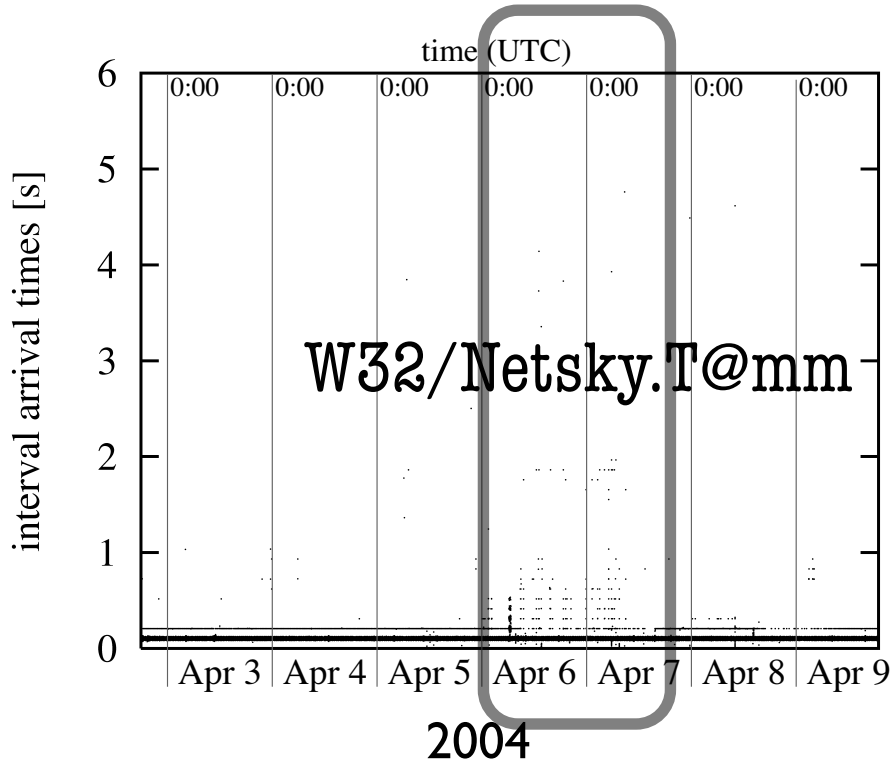
Experimentation: WAN

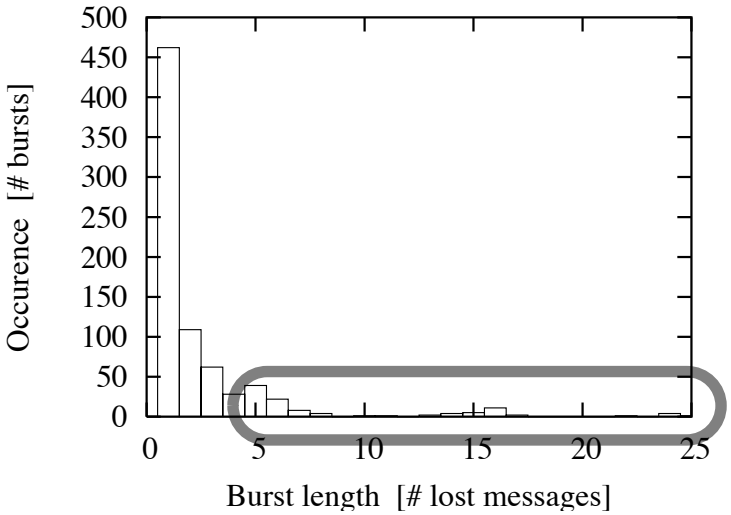
- **WAN**
  - JAIST (JP) – EPFL (CH)
- **Parameters**
  - HB interval: 100 ms
  - Duration: 1 week
  - Total HB: ~ 6’000’000





Experimentation: WAN





Wrapping Up

Conclusion

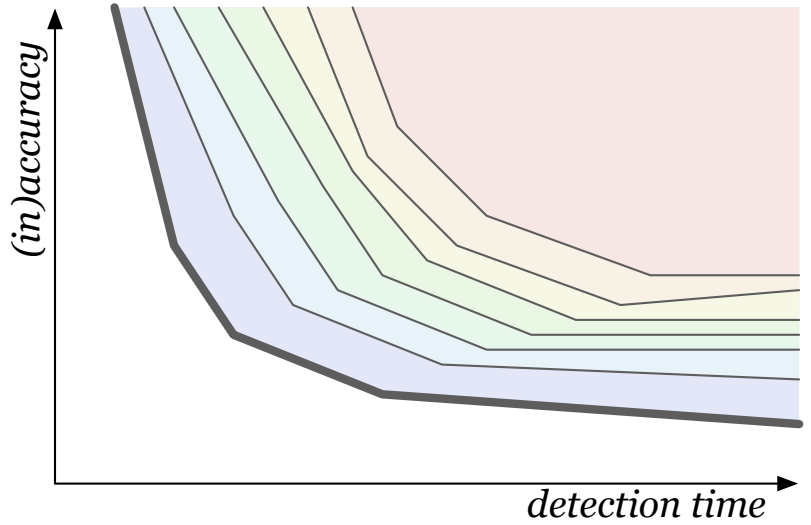
- **Ongoing work**
  - Translucent abstractions
  - Improved implementations
  - Wider experimentation
  - QoS negotiation
- **Much work to do...**
  - Self-configuration
  - Low-overhead protocols
  - Notification mechanisms

JAIST

JST

Future Directions

- **QoS Coverage**
  - stricter definition
  - gradients (uncertainty)
- **QoS negotiation**
  - dynamic (re-)negotiation
  - prob./best-effort negotiation
  - fail-safe enforcement



JAIST

JST

Future Directions

- **Other environments**
  - E.g., wireless, dial-up,...
- **Characterize traffic**
  - metrics
  - clustering
  - "benchmarking" sets