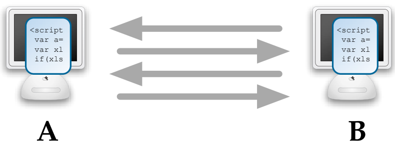
Failure Detection in Distributed Systems:

Retrospective and recent advances

Xavier DÉFAGO

School of Information Science,
Japan Adv. Inst. of Science & Tech. ***(JAIST)***
PRESTO, Japan Science & Tech. Agency ***(JST)***

Failure Detection



- **Context**
  - Two computer programs
  - Exchange messages

Failure Detection

- **Context**
  - A crashes
  - B must detect crash

Simple Approach

- **Solution**
  - Use time & timeouts
- **Problem**
  - How to set timeout?
  - What implications,
    - when too long.
    - when too short.

Simple Approach

- **Problems**
  - Depends on application needs.
  - Depends on system behavior.
  - Needs global time?
- **Outcome**
  - Long failover / reconfiguration time
  - Unstable applications
  - Many side-effects: difficult to maintain

Question

Can we do better?
How?

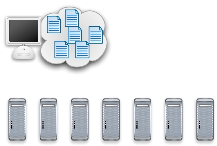
Answer

- **Can we do better?**
  - YES!
- **How?**
  - Depends on context

Illustration: Case 1

Job dispatcher

CPU farm

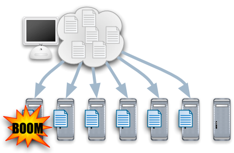


- **Simple illustration**
  - Set of tasks

Illustration: Case 1

Job dispatcher

CPU farm

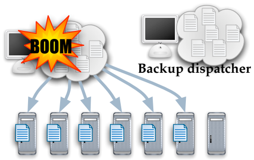


- **Simple illustration**
  - Set of tasks
  - Dispatch tasks; wait for results

Illustration: Case 2

Job dispatcher

CPU farm



- **Dispatcher**
  - Failover
  - Must keep consistency

Outline

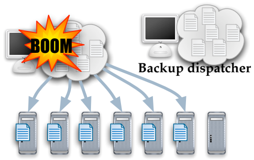
- **I: Theory**
  - Agreement problems, Unreliable failure detectors
- **II: QoS**
  - QoS metrics, Comparison of FDs
- **III: Implementation**
  - Basic FDs, Adaptive FDs
- **IV: Accrual FDs**
  - Novel concept
- **V: Conclusion**

Part I
Theory

Context: Case 2

Job dispatcher

CPU farm



- **Dispatcher**
  - Failover
  - Must keep consistency
  - Requires **agreement** between dispatchers

Outline: Part I

- **System Model**
- **Consensus & Impossibility**
- **Unreliable FDs**
- **Solving Consensus w/FD**

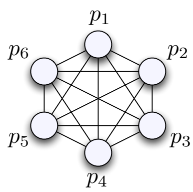
Outline: Part I

- **System Model**
- **Consensus & Impossibility**
- **Unreliable FDs**
- **Solving Consensus w/FD**

System Model

- **Processes**
  - represents running program (or state machine)

$$P = \{p_1, p_2, \ldots, p_n\}$$



- **Communication**
  - message driven
  - fully connected

$$C = \{c_{ij} | c_{ij} : \text{channel from } p_i \text{ to } p_j\}$$

Failure Models

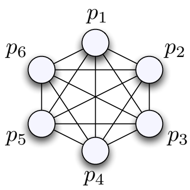
- **Crash failures**
  - Failed process stops executing any event.
- **Omission failures**
  - Failed process omits executing some events.
- **Arbitrary failures (Byzantine)**
  - Failed process can do anything.

Failure Models

- **Crash failures**
  - Failed process stops executing any event.
- **Omission failures**
  - Failed process omits executing some events.
- **Arbitrary failures (Byzantine)**
  - Failed process can do anything.

Synchrony

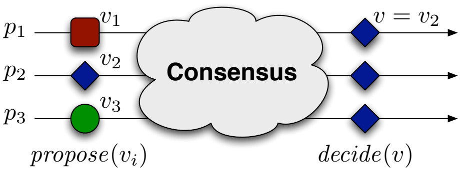
- **Synchronous**
  - Bound on comm. delays
  - Bound on process speed
- **Asynchronous**
  - No bounds
- **Semi-synchronous (e.g.)**
  - GST: global stabilization time
  - Unknown bounds after GST



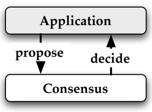
Outline: Part I

- **System Model**
- **Consensus & Impossibility**
- **Unreliable FDs**
- **Solving Consensus w/FD**

Consensus



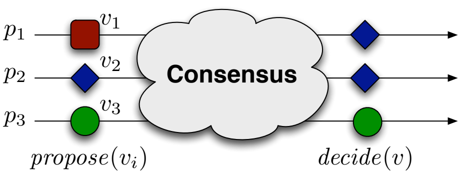
- **Problem**
  - All (correct) processes decide
  - Decision value is same for all processes
  - Decision value is one of proposed values



Consensus (specification)

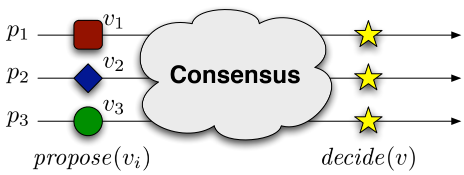
- **Integrity**
  - Every process decides at most once.
- **Validity**
  - If a process decides v, then v was proposed by some process.
- **Agreement**
  - Two correct processes do not decide differently
- **Termination**
  - Every correct process eventually decides.

Consensus (specification)



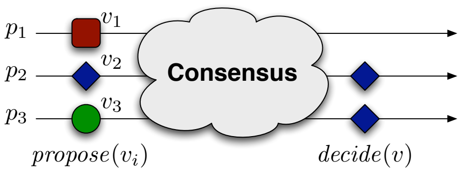
- **Agreement**
  - Two correct processes do not decide differently

Consensus (specification)



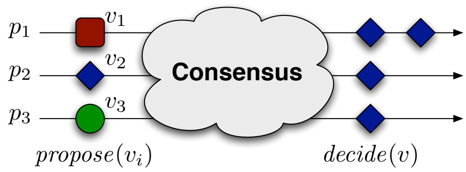
- **Validity**
  - If a process decides v, then v was proposed by some process.

Consensus (specification)



- **Termination**
  - Every correct process eventually decides.

Consensus (specification)



- **Integrity**
  - Every process decides at most once.

Impossibility

from [FLP85]

- **Model**
  - **Asynchronous**
  - Some process **may crash**
- **Result**
  - Consensus has **no deterministic solution**
- **Reason**
  - Impossibility to distinguish **crashed** ***vs.*** **slow** process
- **NB**
  - Cannot guarantee ***Termination*** in **all** cases.
  - **Probabilistic Termination** possible *(w/Prob.=1)*

Related Agreement Problems

- **Total Order Broadcast**
  - Deliver messages in same sequence
  - See [DSU04]

Impossible

- **Leader Election**
  - Elect one correct leader

Impossible

- **Group Membership**
  - Agree on group composition
  - See [CKV01]

Impossible

- **Atomic Commit**
  - Decide on issue of transaction: Commit / Abort

Impossible

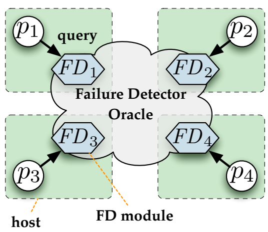
Outline: Part I

- **System Model**
- **Consensus & Impossibility**
- **Unreliable FDs**
- **Solving Consensus w/FD**

Failure Detectors (concept)

from [CT96]

- **FD module**
  - Attached to each process
  - Queried locally
  - Outputs list of suspected processes

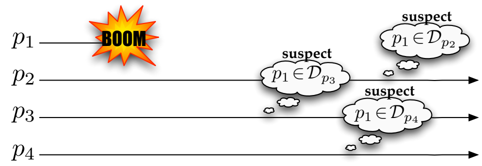


- **Failure Detector**
  - Distributed entity
  - Federation of FD modules
  - Global properties

Unreliable Failure Detectors

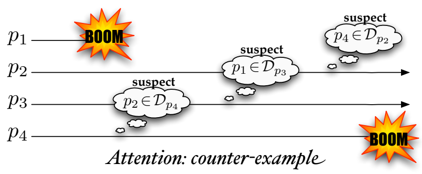
- **FD Unreliable**
  - Can make **mistakes**; can change its mind
- **Possible Mistakes**
  - suspect *p* and *p* has **not** crashed (wrong suspicion)
  - trust *p* and *p* has crashed
- **Properties**
  - Restrict allowed mistakes
  - Completeness; Accuracy

Completeness



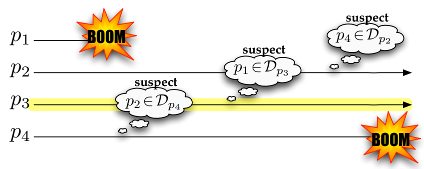
- **Strong Completeness**
  - Eventually every process that crashes is permanently suspected by all processes.

Accuracy



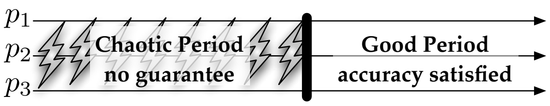
- **Strong Accuracy**
  - No process is suspected before it crashes.

Accuracy



- **Weak Accuracy**
  - Some correct process is never suspected

Eventual Accuracy



- **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

Failure Detector Classes

Accuracy	*Perpetual*	*Eventual*
Strong	$\mathcal{P}$ (*Perfect*)	$\Diamond\mathcal{P}$ (*Eventually Perfect*)
Weak	$\mathcal{S}$ (*Eventually Strong*)	$\Diamond\mathcal{S}$ (*Strong*)

- **Remark**
  - Other classes of failure detectors exist.

Outline: Part I

- **System Model**
- **Consensus & Impossibility**
- **Unreliable FDs**
- **Solving Consensus w/FD**

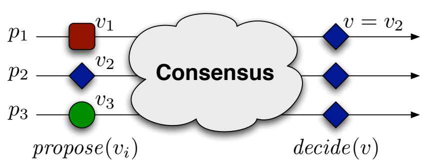
Solving Consensus with ◊S

from [CT96]

- **System Model**
  - Asynchronous System
  - Crash permanent failures
  - Quasi-Reliable channels
  - Failure detector $FD \in \Diamond\mathcal{S}$
- **Assumptions**
  - At least majority of processes are correct

$$t = \left\lceil \frac{n-1}{2} \right\rceil \text{ where } t \text{ is max. faulty processes}$$

Solving Consensus with ◊S



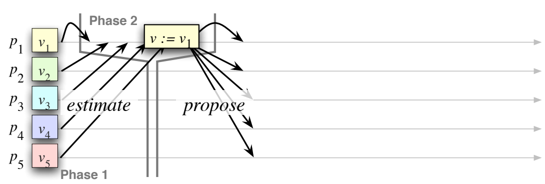
- **Requirements**
  - Reliable Broadcast, failure detector $\in \Diamond\mathcal{S}$
- **Concept**
  - Use rotating coordinator
  - Asynchronous rounds, 4 phases

Solving Consensus with ◊S

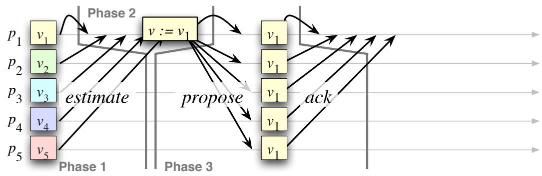
- **Selection of coordinator**
  - One coordinator/round round $i : c = p_{(1+i \bmod n)}$
- **Suspicion**
  - Suspect coordinator => reject round => go to next one.
- **Estimate**
  - Processes keep estimate (of decision value)
  - Initialized with initial value
  - Modified in round *i* by coordinator of *i*
- **Timestamp**
  - Estimate modified in what round

Solving Consensus with ◊S
(crash-free case)

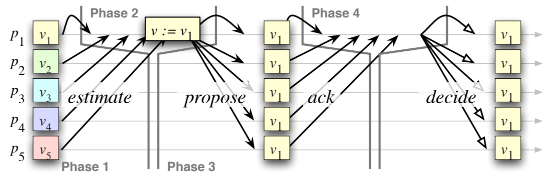
Solving Consensus with ◊S
(crash-free case)



Solving Consensus with ◊S
(crash-free case)



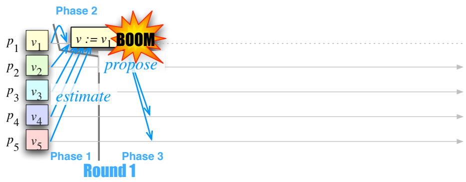
Solving Consensus with ◊S
(crash-free case)



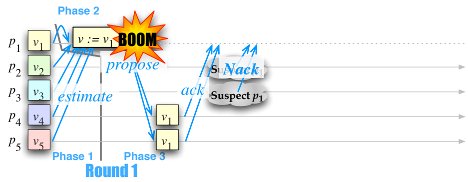
Solving Consensus with ◊S
(one-crash case)

Solving Consensus with ◊S
(one-crash case)

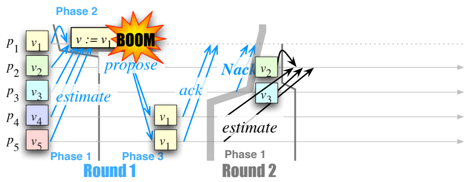
Solving Consensus with ◊S
(one-crash case)



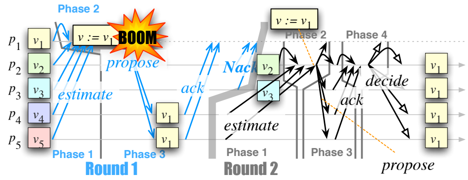
Solving Consensus with ◊S
(one-crash case)



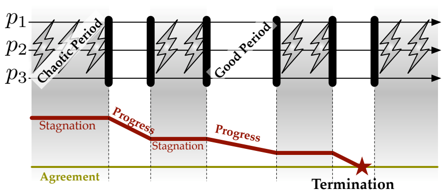
Solving Consensus with ◊S
(one-crash case)



Solving Consensus with ◊*S*
(one-crash case)

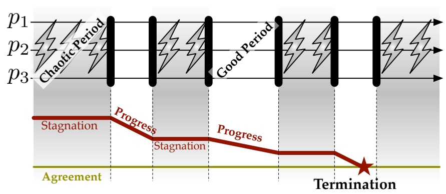


Event. Accuracy in Practice



- **Observations**
  - Bad period: stagnation (progress possible)
  - Good period: progress guaranteed
  - Termination => finished

Event. Accuracy in Practice



- **In practice**
  - Good period *"often enough"* => Termination
  - FD: must ensure accuracy *"often enough"*

Weakest Failure Detector
for Consensus
from [CHT96]

- **Properties: ◊*W***
  - **Weak Completeness:**
    Crash detected by **some** correct process.
  - **Eventual Weak Accuracy:**
    Eventually, **some correct** process never suspected.
- **Equivalence**
  - $\Diamond W \asymp \Diamond S$

Ω: Eventual Leadership
from [Lam98]

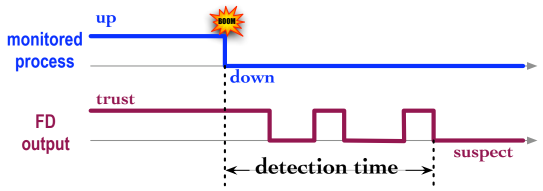
- **Definition**
  - **Eventually**, **one** correct process becomes leader **for all**.
- **Application**
  - Solve Consensus: e.g., PAXOS algorithm of Lamport.
- **Equivalence**
  - $\Diamond W \asymp \Omega$

Part II
Quality of Service

Outline: Part II

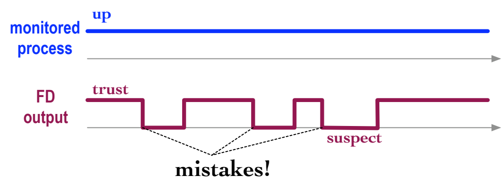
- **Quality of Service Metrics**
- **Comparison**

QoS of Failure Detectors



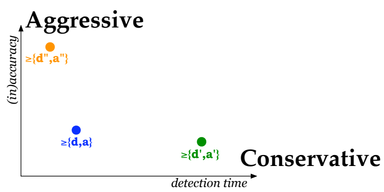
- **Latency Metric**
  when *p* faulty:
  - *Detection time*
  - *"How long to detect?"* The shorter the better.

QoS of Failure Detectors



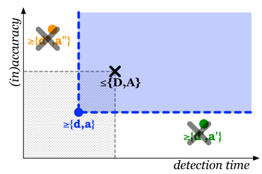
- **Accuracy Metrics**
  when *p* correct:
  - *Average mistake rate*
  - *Query accuracy prob.*
  - *Good period duration*

QoS Tradeoff



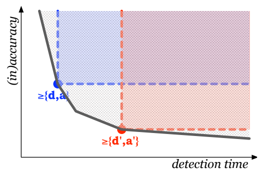
- **Aggressive FD**
  - Short latency; low accuracy
- **Conservative FD**
  - Long latency; high accuracy

Requirements vs. Guarantees



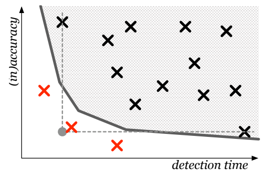
- **FD QoS**
  - ≥{d,a} : effect. detection time, effect. mistakes
- **Application requirements**
  - ≤{D,A} : max. detect. time, max. mistakes

Parametric Failure Detector



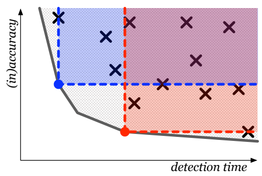
- **Parametric FD protocol**
  - Parameter value defines FD best QoS
  - Tradeoff: accuracy <-> detection latency

QoS Coverage



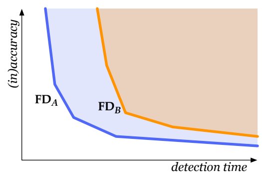
- **Coverage of FD**
  - FD **could** be tuned to support app. req.
  - Measure of FD

Dynamic QoS Coverage



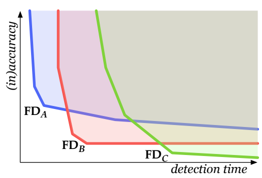
- **Approximate coverage**
  - Instantiate several QoS sets
  - Find minimal set; minimal change

Comparing Parametric FDs



- **Simple case**
  - 2 FDs
  - A includes B
  - So, A better than B

Comparing Parametric FDs



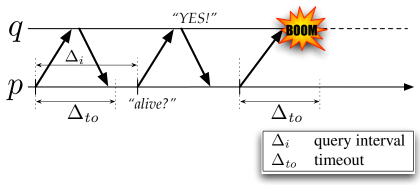
- **Complex case**
  - 3 FDs: aggressive, intermediate, conservative
  - Which one is better?

Part III Implementations

Outline: Part III

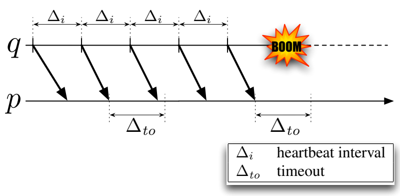
- **Basic FDs**
- **Adaptive FDs**
- **Experimental Results**

Interrogation



- **Advantage**
  - Simplest
- **Drawback**
  - Longest detection time

Heartbeat



- **Advantage**
  - Good with broadcast medium
- **Drawback**
  - *q* takes active role

Parameters & QoS

- **Detection Time**
  - **Heartbeat interval**
  - **Timeout**
  - **Transmission delays**
- **Accuracy**
  - **Timeout**
  - **Variations** in transmission delays

Heartbeat Interval

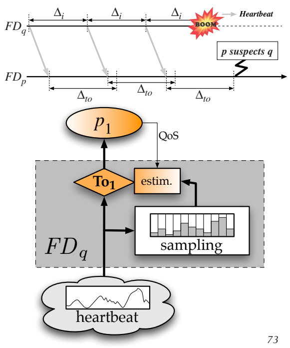
- **Observation**
  - Small influence on accuracy
  - Limited by network admin.
  - Detection time ≈ transmission + interval
- **Rule-of-thumb**
  - Same order as average transmission delay

Timeout

- **Observation**
  - Influence on detection time
  - Influence on accuracy
  - Too short = instability
- **Problem**
  - Depends on system behavior
  - System behavior changes
  - **=> need adaptive timeout**

Adaptive Approach

- **Concept**
  - Periodic heartbeat
  - Timeout => suspicion
  - Timeout adjusted dynamically
- **Parameters**
  - Period
  - Safety margin

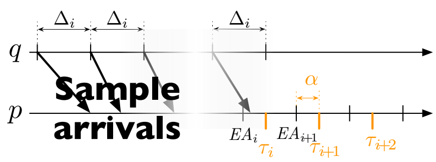


Chen's Failure Detector

from [CTA02]

- **Context**
  - Heartbeat failure detector
  - Adaptive
  - Several variants
- **Adaptation**
  - Freshness points
- **Parameters**
  - Known heartbeat interval
  - Safety margin based on QoS

Chen'FD: Freshness Points



- **Freshness points**
  - Samples heartbeat arrivals
  - Compute normalized distribution
  - Estimates future arrivals
  - Adds safety margin α

Δ_i :	heartbeat interval
HB_i :	i^{th} heartbeat msg
EA_i :	expected arrival for HB_i
α :	safety margin
τ_i :	freshness point for HB_i

Chen's FD: Mechanism

- **Suspicion when...**
  - Freshness point *i* past
  - No heartbeat $k \geq i$ received
- **QoS tuning**
  - Adjust safety margin based on QoS requirements
- **Other**
  - variants based on other assumptions (e.g., synchronized clocks, etc.)

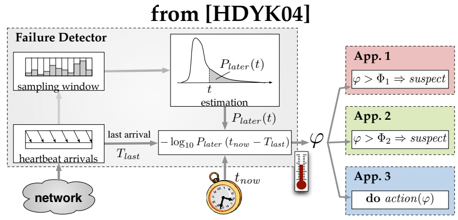
Bertier's Failure Detector

from [BMS02]

- **Principle**
  - Based on Chen's failure detector
  - Adaptive, but not tunable
- **Safety margin**
  - Safety margin adjusted dynamically
  - Use Van Jacobson's estimation
  - Very aggressive failure detector

PHI-FD

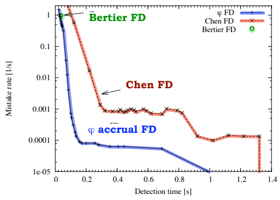
from [HDYK04]



- **PHI-FD**
  - Estimates arrival distribution probability
  - Increase level based on probability
  - Sets threshold for suspicions (based on QoS)

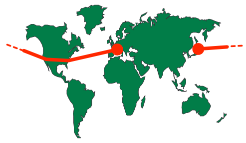
Experimentation: LAN

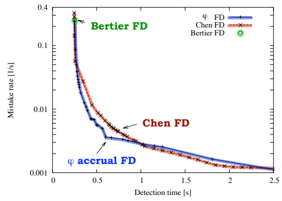
- **LAN**
  - single FastEther hub
- **Parameters**
  - HB interval: 20 ms
  - Duration: 5½ hour
  - Total HB: 1’000’000
  - no loss



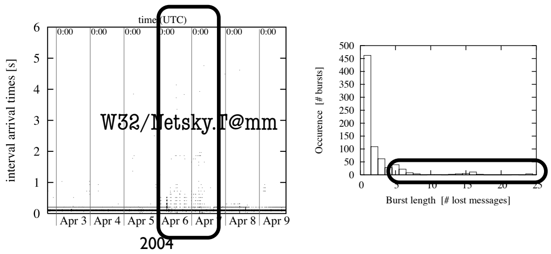
Experimentation: WAN

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





Experimentation: WAN



Part IV
Accrual FDs

Outline: Part IV

- **Motivation**
- **Definition (accrual FDs)**
- **Equivalence: $\Diamond P$**
- **Relation w/QoS**

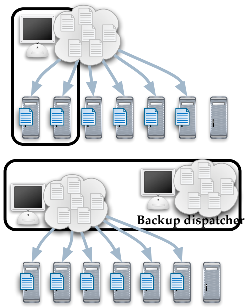
Motivation

- **Objective (long term)**
  - Offer failure detection as generic service
  - E.g., NTP for clock synchronization
- **Open issues**
  - interac
  - notifica
  - self-configuration, deployment

Accomodate various **usage patterns** and **QoS requirements simultaneously**

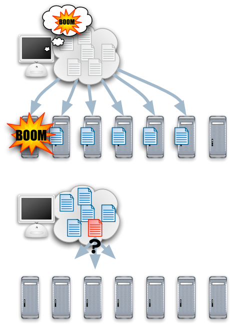
Different Patterns

- **Dispatch. – Worker**
  - Action: release resources
  - Needs stability
  - => conservative FD
- **Dispatch. replica**
  - Action: failover
  - Needs quick reaction (see Consensus)
  - => mid. aggressive FD



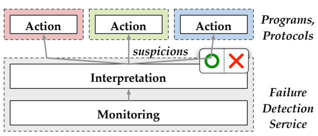
Variable Suspicion Costs

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



Decoupling

Binary FD



- **Failure detection**
  - 2 roles: *monitoring, interpretation*
  - interpretation –> QoS
  - => **decoupling**

Accrual Failure Detectors

from [DUHK05]

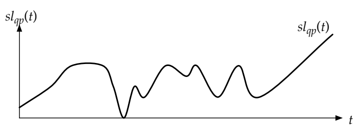
- **Abstraction**
  - Provides **suspicion level**
  - Separates monitoring / interpretation
  - QoS handled locally (e.g., by thresholds)
- **Formally**
  - Well-defined behavior
  - Preserves theoretical characteristics
  - Relation between threshold & QoS
- **Practically**
  - Many implementations possible

Suspicion Level

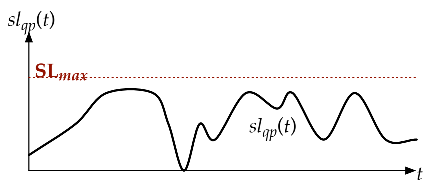
- **Suspicion Level**
  - function of **time** to **non-negative**

$$sl_{qp} : \mathbb{T} \mapsto \mathbb{R}_0^+$$

  - means "confidence that *p* is faulty": (0 = trust *p*)



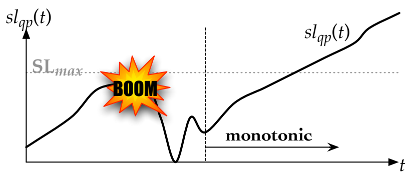
Property 1 (Upper bound)



- **If p is correct**
  - $sl_{qp}(t)$ is **bounded**
  - bound SL_{max} is **unknown**

$$p \in correct(F) \Rightarrow \exists SL_{max} : \forall t\,(sl_q p(t) \leq SL_{max})$$

Property 2 (Accruement)



- **If *p* is faulty**
  - $sl_{qp}(t)$ is eventually **monotonic**
  - **(unknown) minimum** increase rate

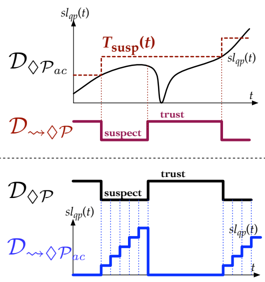
$$p \in faulty(F) \Rightarrow \exists K \exists Q \forall k \geq K \begin{pmatrix} sl_{qp}(t_q^{qry}(k)) \leq sl_{qp}(t_q^{qry}(k+1)) \\ \wedge sl_{qp}(t_q^{qry}(k)) < sl_{qp}(t_q^{qry}(k+Q)) \end{pmatrix}$$

Class $\Diamond\mathcal{P}_{ac}$

- **Class** $\Diamond\mathcal{P}_{ac}$
  - for all pairs of processes:
    - Upper bound holds
    - Accruement holds
- **Equivalence:** $\Diamond\mathcal{P}_{ac} \asymp \Diamond\mathcal{P}$
  - can solve **same set of problems**
  - implementable in **same systems**

Equivalence

- **Accrual to Binary**
  - dynamic thresholds
  - *suspect*: threshold & trust
  - *trust*: rate insufficient

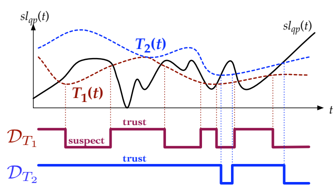


Equivalence

$$\Diamond\mathcal{P}_{ac} \asymp \Diamond\mathcal{P}$$

- **Means...**
  - same computational power
    - can solve same set of problems
    - can be implemented over same set of systems
- **but...**
  - **loss of information**
  - **overall performance can be different**

QoS w/multiple thresholds

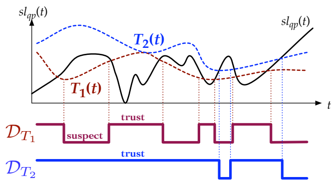


- **Threshold function**
  - function of time $T_i : \mathbb{T} \mapsto \mathbb{R}^+$
  - triggers suspicion
  - defines binary FD: $\mathcal{D}_{T_i}$

QoS w/multiple thresholds

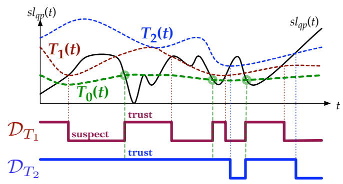
- **Hypotheses**
  - Hyp.1: At all time, $T_1(t) \leq T_2(t)$
- **Theorem**
  - $\mathcal{D}T_2$ suspects p only if $\mathcal{D}T_1$ suspects p
- **Detection time**
  - detect. time w/ $\mathcal{D}T_1 \leq$ detect. time w/ $\mathcal{D}T_2$
- **Query accuracy prob.**
  - prob. trust w/ $\mathcal{D}T_1 \leq$ prob. trust w/ $\mathcal{D}T_2$

QoS w/multiple thresholds



- **Threshold function**
  - function of time $T_i : \mathbb{T} \mapsto \mathbb{R}^+$
  - triggers suspicion
  - defines binary FD: $\mathcal{D}_{T_i}$

QoS w/multiple thresholds

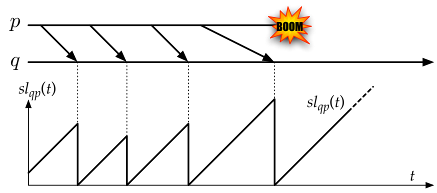


- **Trust threshold function**
  - shared by all detectors

QoS w/multiple thresholds

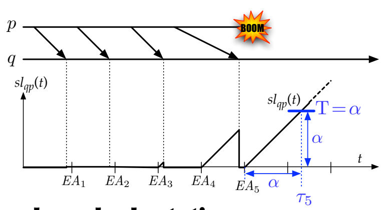
- **Hypotheses**
  - Hyp.1: At all time, $T_1(t) \leq T_2(t)$
  - Hyp.2: $\mathcal{D}T_1$ & $\mathcal{D}T_2$ use **same trust threshold**
- **Theorem**
  - $\acute{\mathcal{D}}T_2$ has T-transition => $\mathcal{D}T_1$ has T-transition
- **Mistake rate**
  - mistake rate w/ $\mathcal{D}T_1$ >= mistake rate w/ $\mathcal{D}T_2$
- **Good period duration**
  - good period duration w/ $\mathcal{D}T_1$ <= w/ $\mathcal{D}T_2$
- **mistake duration: cannot say**

Implementations



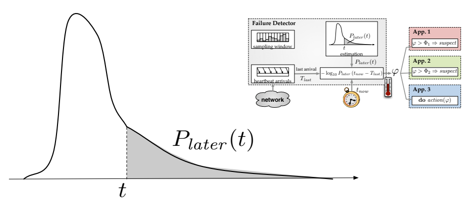
- **Simple implementation**
  - Partially synchronous model
  - Susp. level increase with time
  - **Reset** when receive heartbeat

Implementations



- **Chen-based adaptation** **[CTA02]**
  - After "expected arrival" point, increase with time
  - **Reset** when receive heartbeat
  - Safety margin α set with threshold

Implementations



- **PHI accrual FD** **[HDYK04]**
  - Samples arrival probability
  - Estimate prob. receive HB later
  - **Reset** when receive heartbeat

Implementations

	suspicion level *related to...*
Simple implementation	**detection time**
Chen-based adaptation	
φ accrual FD	**accuracy**

- **Difference**
  - **Dominant metric** for suspicion level

Part V Conclusion

Other Important Issues

- **Notification**
  - Gossiping
  - Hierarchy, spanning tree
- **Interface**
  - QoS negotiation
- **Evaluation**
  - Representative environments
  - Benchmarks

Conclusions

- **Failure Detection**
  - Active area of research
  - Theory well-developed
  - Practice lagging
- **Open questions...**
  - Better abstractions / interfaces
  - Better performance (QoS)
  - Lower overhead
  - Self-configuration