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

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

• Simple illustration
  • Set of tasks
  • Dispatch tasks; wait for results

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

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

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
  - \( C = \{c_{ij} : \text{channel from } p_i \text{ to } p_j\} \)

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.
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

Outline: Part 1

- 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

- **Application**
  - propose
  - decide
  - Consensus

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
- **Validity**
  - If a process decides \( v \), then \( v \) was proposed by some process.
**Consensus (specification)**

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

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

---

**Impossibility**

- **Model**
  - Asynchronous
- **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]
- **Leader Election**
  - Elect one correct leader
- **Group Membership**
  - Agree on group composition
  - See [CKV01]
- **Atomic Commit**
  - Decide on issue of transaction: Commit / Abort

---

**Outline: Part I**

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

---

**Failure Detectors (concept)**

- **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 $p_1$ and $p_2$ has not crashed (wrong suspicion)
  - trust $p_1$ and $p_2$ has crashed

- **Properties**
  - Restrict allowed mistakes
  - Completeness; Accuracy

Complements

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

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

- **Eventual**
  - There is a time after which correct processes are not suspected by any correct process.

- **Regular**
  - There is a time after which some correct process is never suspected

Strong Failure Detector Classes

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>Perpetual</th>
<th>Eventual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong</td>
<td>$\mathcal{P}$ (Perfect)</td>
<td>$\mathcal{P}$ (Eventually Perfect)</td>
</tr>
<tr>
<td>Weak</td>
<td>$\mathcal{S}$ (Eventually Strong)</td>
<td>$\mathcal{S}$ (Strong)</td>
</tr>
</tbody>
</table>

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

Attention: counter example

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

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

- **Eventual Accuracy**
  - There is a time after which correct processes are not suspected by any correct process.

- **Chaotic Period**
  - Good Period

<table>
<thead>
<tr>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>$P_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{P}$</td>
<td>$\mathcal{P}$</td>
<td>$\mathcal{P}$</td>
<td>$\mathcal{P}$</td>
</tr>
</tbody>
</table>

Failure Detector Classes

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>Perpetual</th>
<th>Eventual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong</td>
<td>$\mathcal{P}$ (Perfect)</td>
<td>$\mathcal{P}$ (Eventually Perfect)</td>
</tr>
<tr>
<td>Weak</td>
<td>$\mathcal{S}$ (Eventually Strong)</td>
<td>$\mathcal{S}$ (Strong)</td>
</tr>
</tbody>
</table>
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 $FD \in ◇S$

- **Assumptions**
  - At least majority of processes are correct
    $$t = \left\lfloor \frac{n-1}{2} \right\rfloor$$
    where $t$ is max. faulty processes

---

Solving Consensus with ◇S

- **Selection of coordinator**
  - One coordinator/round
    $$\text{round } i : e = p(i \text{ mod } n)$$

- **Suspicion**
  - Suspect coordinator $\Rightarrow$ reject round $\Rightarrow$ 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)

**Phase 1**

**Phase 2**

```v := v_1``
Solving Consensus with $\diamond S$

(crash-free case)

Phase 1

estimate

propose

v := v_1

Phase 2

BOOM

Phase 3

ack

Phase 4

decide

Solving Consensus with $\diamond S$

(one-crash case)

Phase 1

Round 1

Solving Consensus with $\diamond S$

(one-crash case)

Phase 1

Round 1

BOOM

Solving Consensus with $\diamond S$

(one-crash case)

Phase 1

Round 1

BOOM

Solving Consensus with $\diamond S$

(one-crash case)

Phase 1

Round 1

BOOM

Solving Consensus with $\diamond S$

(one-crash case)
Solving Consensus with \(\diamond S\) (one-crash case)

Event. Accuracy in Practice

- Observations
  - Bad period: stagnation (progress possible)
  - Good period: progress guaranteed
  - Termination \(\Rightarrow\) finished

- In practice
  - Good period "often enough" \(\Rightarrow\) Termination
  - FD: must ensure accuracy "often enough"

Weakest Failure Detector for Consensus
from [CHT96]

- Properties: \(\diamond W\)
  - Weak Completeness:
    Crash detected by some correct process.
  - Eventual Weak Accuracy:
    Eventually, some correct process never suspected.

- Equivalence
  \(\diamond W = \diamond S\)

\(\Omega\): Eventual Leadership
from [Lam98]

- Definition
  - Eventually, one correct process becomes leader for all.

- Application
  - Solve Consensus: e.g., PAXOS algorithm of Lamport.

- Equivalence
  \(\diamond W = \Omega\)
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 Tradeoff

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

Requirements vs. Guarantees

- FD QoS
  - 2(d,a): effect. detection time, effect. mistakes
- Application requirements
  - s(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

---

Chen’s FD: Mechanism

- **Suspicion when...**
  - Freshness point / past
  - No heartbeat at 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
  - no loss

Part IV

Accrual FDs

Outline: Part IV

- Motivation
- Definition (accrual FDs)
- Equivalence: /P
- Relation w/QoS

Motivation

- Objective (long term)
  - Offer failure detection as generic service
  - E.g., NTP for clock synchronization

- Open issues
  - Accomodate various usage patterns and QoS requirements simultaneously
  - interact
  - notification
  - self-configuration, deployment
Different Patterns

- **Dispatch. – Worker**
  - Action: release resources
  - Needs stability
  - => conservative FD

- **Dispatch, replica**
  - Action: failover
  - Needs quick reaction (see Consensus)
  - => mid. aggressive FD

Variable Suspection Costs

- **Case 1:**
  - Cost varies with time:
    - amount work completed
    - available resources
  - => conservative FD

Decoupling

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

Accrual Failure Detectors

- **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
  \[ s_{lp} : \mathbb{T} \rightarrow \mathbb{R}_0^+ \]
  - means “confidence that \( p \) is faulty”: (0 = trust \( p \))

Property 1 (Upper bound)

- **If \( p \) is correct**
  - \( s_{lp}(t) \) is bounded
  - bound \( SL_{max} \) is unknown
  \[ p \in \text{correct}(F) \Rightarrow \exists SL_{max} : \forall t (s_{lp}(t) \leq SL_{max}) \]
Property 2 (Accruement)

- If $p$ is faulty
- $slqp(t)$ is eventually monotonic
- (unknown) minimum increase rate $\frac{d}{dt} slqp(t) \leq K \Rightarrow$ eventually monotonic

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

- Class $\mathcal{P}_{ac}$
- for all pairs of processes:
  - Upper bound holds
  - Accrual holds

**Equivalence:** $\mathcal{P}_{ac} \times \mathcal{P}$

- can solve same set of problems
- implementable in same systems

**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 $T : T \rightarrow \mathbb{R}^+$
  - triggers suspicion
  - defines binary FD: $DT_i$

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

- Hypotheses
  - Hyp.1: At all time, $T_1(t) < T_2(t)$

Theorem

- $DT_2$ suspects $p$ only if $DT_1$ suspects $p$

Detection time

- detect. time w/$DT_1$ ≤ detect. time w/$DT_2$

Query accuracy prob.

- prob. trust w/$DT_1$ ≤ prob. trust w/$DT_2$
**Threshold function**
- function of time $T_i: T \mapsto \mathbb{R}^+$
- triggers suspicion
- defines binary FD: $D_{T_i}$

**QoS w/multiple thresholds**

- **Hypotheses**
  - Hyp. 1: At all time, $T_i(t) \leq T_j(t)$
  - Hyp. 2: $D_{T_1}$ & $D_{T_2}$ use same threshold
- **Theorem**
  - $D_{T_2}$ has T-transition $\Rightarrow$ $D_{T_1}$ has T-transition
- **Mistake rate**
  - mistake rate w/$D_{T_1}$ $\geq$ mistake rate w/$D_{T_2}$
- **Good period duration**
  - good period duration w/$D_{T_1}$ $<$ w/$D_{T_2}$
- **mistake duration: cannot say**

**Implementations**

- **Chen-based adaptation** [CTA02]
  - After “expected arrival” point, increase with time
  - Reset when receive heartbeat
  - Safety margin $\alpha$ set with threshold

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

<table>
<thead>
<tr>
<th></th>
<th>suspicion level related to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple implementation</td>
<td>detection time</td>
</tr>
<tr>
<td>Chen-based adaptation</td>
<td>accrual FD</td>
</tr>
<tr>
<td></td>
<td>accuracy</td>
</tr>
</tbody>
</table>

- 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