On Accrual Failure Detectors*

Xavier Défago *,†, Péter Urbán*, Naohiro Hayashibara*, Takuya Katayama*
*School of Information Science, JAIST
1-1 Asahidai, Tatsunokuchi, Nomi, Ishikawa 923-1292, Japan.
†PRESTO, Japan Science and Technology Agency (JST).
Email: {defago,urban,nao-haya,katayama}@jaist.ac.jp

May 14, 2004

Abstract

Traditionally, failure detectors have considered a binary model whereby a given process can be either trusted or suspected. This paper defines a family of failure detectors, called accrual failure detectors, that revisits this interaction model. Accrual failure detectors associate to each process a real value representing a *suspicion level*. An important advantage of accrual failure detectors over binary ones is to allow distributed applications to trigger different actions depending on the suspicion level. For instance, an application can take precautionary measures when the suspicion level reaches a given level, and then take more drastic actions after it raises above a second (much higher) level. The paper defines accrual failure detectors and their basic properties. Four classes of accrual failure detectors are discussed, each of which is proved equivalent to a class of binary unreliable failure detectors ($\mathcal{P}$, $\mathcal{S}$, $\Diamond\mathcal{P}$, and $\Diamond\mathcal{S}$).

1 Introduction

Context of the study. Failure detection is a fundamental issue in fault-tolerant distributed computing, from both a practical and a theoretical standpoint.

The concept of unreliable failure detectors was introduced and formalized by Chandra et al. [3, 2], who, among other things, identify the minimal conditions to solve the Consensus problem [16] in asynchronous systems. Equally important, they also show that failure detectors constitute a fundamental *abstraction* for distributed systems and algorithms.

Except for strictly synchronous distributed systems, failure detection is inherently uncertain. Unreliable failure detectors account for this uncertainty by allowing mistakes, thus introducing the notion of "suspicion" as opposed to detection. A *detection* occurs when a faulty process is recognized as such. In contrast, a *suspicion* occurs when a process is considered as faulty, although this may not actually be the case. As an analogy, a similar difference exists between "knowing something" and "believing in something."

The distinction between detection and suspicion is essential, but it is also often desirable to know more than just "process p is suspect." More specifically, one wants to know the *degree of confidence* associated with each suspicion (also called *suspicion level* here). For instance, a distributed application may want to assign primary roles to processes that are "not suspect at all" and relegate to secondary roles those that are "a little suspect."

*Research supported by the Japan Society for the Promotion of Science, a Grant-in-Aid for JSPS Fellows from the Japanese Ministry of Education, Culture, Sports, Science and Technology, and the Swiss National Science Foundation.

Illustration. Let us illustrate this with a simple example. Consider a distributed application with one master process and a collection of worker processes. The master holds a list of jobs that needs to be computed, dispatches these jobs to the available workers, and gathers results. Dependencies existing between jobs, some are more urgent than others. To simplify the discussion, assume that the master never crashes but that some of the workers may crash. Obviously, the master must be able to detect when a worker has crashed and take appropriate actions, otherwise some tasks will never complete.

With a confidence level associated to suspicions, this could be realized as follows. Urgent jobs are dispatched to workers that have the lowest suspicion level, while less urgent jobs are sent to workers with a higher suspicion level. When the suspicion level associated with a worker reaches a moderately high threshold, the master replicates the computation by sending a new instance of the same job to another worker. Finally, when the suspicion level goes beyond some very high threshold, the worker is removed from the list of workers and all corresponding resources are released. Note that the binary model normally considered (i.e., trust vs. suspect) does not allow for such differentiated actions to be taken.

Contribution of the paper. In this paper, we define a family of failure detectors, called accrual failure detectors, whereby each monitoring process associates, to each of the monitored process, a real value that changes over time.[1] The value represents the *suspicion level*, where zero means that the process is not suspected, and positive values mean that the process is suspected; the larger the value, the stronger the suspicion. Roughly speaking, accrual failure detectors ensure that the suspicion level associated with a monitored process p (1) accrues toward infinity if p is faulty, and (2) is bounded if p is correct. We define four classes of accrual failure detectors, called $\mathcal{P}_{ac}$, $\mathcal{S}_{ac}$, $\Diamond\mathcal{P}_{ac}$, and $\Diamond\mathcal{S}_{ac}$, depending whether the bound is known or not, and whether the properties hold for all pairs of processes or for just some of them. We prove that each class of accrual failure detectors is equivalent to a class of binary unreliable failure detectors (namely, $\mathcal{P}$, $\mathcal{S}$, $\Diamond\mathcal{P}$, and $\Diamond\mathcal{S}$), thus showing that accrual failure detectors are neither more nor less powerful than binary ones.

Practical considerations. One important advantage of an accrual failure detector over a binary one is that the former provides a suitable abstraction for implementing a generic failure detection service, one that suits the needs of multiple applications with different quality of service requirements. The reason is that an accrual failure detector leaves the task of interpreting the suspicion level to applications, and hence different applications can set different thresholds to suspect processes according to their needs. In contrast, a binary failure detector provides a value already interpreted (suspect or not), and is thus poorly designed to serving more than a single application at a time.

Related work. We present existing work that, just like our approach, uses numeric and sometimes accruing values for failure detection or similar purposes.

Cosquer et al. [5] describe a group membership service that allows the tuning of its failure detection by monitoring various system parameters that are combined internally into a single value. By exposing this value to processes, this could provide a strong basis for implementing an accrual failure detector.

Aguilera et al. [1] propose the failure detector called $\mathcal{HB}$ (Heartbeat) that can be used together with an unreliable failure detector to solve Consensus in partitionable systems. Roughly speaking, the failure detector associates to each process an integer value that increases as long as the process remains reachable. In contrast, the output of an accrual failure detector increases if the process is *not* reachable.

[1] An implementation for such a failure detector, called the φ failure detector, was outlined at a workshop during DISC 2003 [12]. A revised version with a performance comparison is also available as a technical report [14].

More recently, Friedman [8] outlined in a position paper the idea of a *fuzzy group membership*, where a value called fuzziness level would be associated with each process to determine the extent to which the process belongs to the group. Technical issues were developed later by Friedman and Tcharny [9, 11, 10]. Although the papers address different issues, the authors rely on some fuzzy failure detector that outputs some integer value and uses two thresholds to define three suspicion levels (*trusted*, *fuzzy*, or *suspected*). There are no details, however, because this is not the main focus of their work. In particular, they give no definition nor implementation of fuzzy failure detectors. We believe that, although developed independently, our works could in fact complement each other.

Sampaio et al. [17] define slowness oracles as being some failure detector oracle that outputs a list of processes ordered according to the perceived responsiveness of each process. Accrual failure detectors also quantify responsiveness, hence their output values could be used to establish (or estimate) this order.

Mostefaoui et al. [15] propose an algorithm to implement a leader oracle (also called Ω failure detector) that relies internally on integer counters that are incremented each time the corresponding process is suspected to have crashed. If the process has crashed, the counter never stops increasing. This part of their protocol is in fact similar to an accrual failure detector and an adaptation of their protocol might possibly be used to implement a leader oracle based on accrual failure detectors. A major difference is that counters never decrease, even for correct processes.

Structure of the paper. The remainder of the paper is structured as follows. Section 2 presents the system model and important definitions. Section 3 defines the general notion of accrual failure detectors, as well as four main classes of accrual failure detectors, called $\mathcal{P}_{ac}$, $\Diamond\mathcal{P}_{ac}$, $\mathcal{S}_{ac}$, and $\Diamond\mathcal{S}_{ac}$ respectively. Section 4 outlines a simple heartbeat-based algorithm to implement $\Diamond\mathcal{P}_{ac}$ in a partially synchronous model. Section 5 proves simple yet useful properties about using accrual failure detectors with multiple suspicion thresholds. Section 6 proves the equivalence between the four classes of accrual failure detectors and their binary counterparts. Finally, Section 7 concludes the paper.

2 System Model & Definitions

Basic system model. We consider a distributed system consisting of a set of processes $\Pi = \{p_1, \ldots, p_n\}$. Notice that, at this stage, we make no specific assumption regarding how processes communicate, such as whether this is done using messages or some shared memory.[2]

We assume the existence of some global time, unbeknown to processes, the domain of which, denoted by $\mathbb{T}$, is an infinitely countable subset of real numbers with no upper bound. We assume that processes may access some local clock by calling the pseudo-function now; values can be provided by a real-time clock or by simply counting the number of steps the process takes. now returns strictly monotonically increasing values from $\mathbb{T}$. Nothing is assumed, however, regarding the synchronization of clocks between processes. We assume that processes always make progress, and that at least $\delta > 0$ time units elapse between consecutive steps, both in local and global time.

Failures. The failure model considered in this paper is based on the model of Chandra and Toueg [3]. A process can be correct or faulty. A process is *faulty* if its behavior deviates from its specification, and a process is *correct* if it is not faulty. We say that a process *fails* when its behavior starts deviating from its specification.[3]

A failure pattern is a function $F : \mathbb{T} \mapsto 2^{\Pi}$, where $F(t)$ is the set of processes that have failed before or at time t. The function $correct(F)$ denotes the set of correct processes (processes that never belong

[2] Interprocess communication is indeed irrelevant to the specification of the failure detectors.

[3] We think of failures as permanent. In particular, we do not define predicates related to recovery.

to failure pattern F) while $faulty(F)$ denotes the set of faulty processes (complement of $correct(F)$ with respect to Π).

Failure detectors. Chandra and Toueg [2] define failure detectors as a collection of failure detector modules, one attached to each process, that output information on the failure pattern that occurs in an execution.[4]

A failure detector module outputs information from a range $\mathcal{R}$ of values. A failure detector history H with range $\mathcal{R}$ is a function $H : \Pi \times \mathbb{T} \mapsto \mathcal{R}$, where $H(p, t)$ is the value output by the failure detector module of process p at time t. A failure detector $\mathcal{D}$ is a function that maps every failure pattern F to a set of failure detector histories with range $\mathcal{R}_\mathcal{D}$ (where $\mathcal{R}_\mathcal{D}$ is the range of the information output by the failure detector modules of $\mathcal{D}$). $\mathcal{D}(F)$ is the set of failure detector histories that failure detector $\mathcal{D}$ permits for failure pattern F.

Binary failure detectors, such as the failure detectors defined in [3], output values from the range $\mathcal{R} = 2^\Pi$, the power set of Π. If a process is part of the output set, it is *suspected* to have failed, otherwise it is *trusted*. An *S-transition* occurs when a trusted process becomes suspected and a *T-transition* occurs when a suspected process becomes trusted.

Chandra and Toueg [3] define a class hierarchy of unreliable binary failure detectors, of which we present only four here, called $\mathcal{P}$ (perfect), $\mathcal{S}$ (strong), $\Diamond\mathcal{P}$ (eventually perfect), and $\Diamond\mathcal{S}$ (eventually strong). The four classes differ by the set of failure detector histories permitted by each of them. This set is defined by two properties of *completeness* and *accuracy*. All four failure detectors mentioned above share the same property of completeness, and only differ by their accuracy property:

(STRONG COMPLETENESS) Eventually every faulty process is permanently suspected by all correct processes. [classes $\mathcal{P}$, $\Diamond\mathcal{P}$, $\mathcal{S}$, $\Diamond\mathcal{S}$]

(STRONG ACCURACY) Correct processes are never suspected. [class $\mathcal{P}$]

(EVENTUAL STRONG ACCURACY) There is a time after which correct processes are never suspected by any correct process. [class $\Diamond\mathcal{P}$]

(WEAK ACCURACY) Some correct process is never suspected. [class $\mathcal{S}$]

(EVENTUAL WEAK ACCURACY) There is a time after which some correct process is never suspected by any correct process. [class $\Diamond\mathcal{S}$]

3 Accrual Failure Detectors

We first define the notion of suspicion level between a pair of processes. Then, we define the notion of accrual failure detector for a distributed system with n processes. Finally, we define four classes of accrual failure detectors that are of particular interest.

3.1 Suspicion level

Consider two distinct processes p and q, with q monitoring p. Let $\mathbb{R}_0^+$ denote the real positive numbers and zero. The suspicion level of process q monitoring process p expresses the confidence of p in the statement that q is faulty. It is defined as follows.

[4]The definition of failure detectors of Chandra and Toueg [3] restricts the output to a set of suspected processes, but the definition of Chandra et al. [2] allows from an arbitrary range. Accrual failure detectors are based on the latter and more general definition.

Definition 1 (Suspicion level) *The suspicion level of process q with respect to process p is the function $susp\_level_{q \to p} : \mathbb{T} \mapsto \mathbb{R}_0^+$.*

We require that the function $susp\_level_{q \to p}$ satisfies the following two properties.

Property 1 (Accruement) *If process p is faulty, the suspicion level $susp\_level_{q \to p}(t)$ tends to infinity as time goes to infinity.*

$$p \in faulty(F) \Rightarrow \lim_{t \to +\infty} susp\_level_{q \to p}(t) = +\infty$$

Property 2 (Reset) *If process p is correct, then for any time t_0, $susp\_level_{q \to p}(t) = 0$ for some time $t \geq t_0$.*

$$p \in correct(F) \Rightarrow \big(\forall t_0 \in \mathbb{T}, \exists t \geq t_0 : susp\_level_{q \to p}(t) = 0\big)$$

Depending on the class of failure detector and the pair of processes, one of the two properties below may be satisfied by the function $susp\_level_{q \to p}$.

Property 3 (Unknown upper bound) *If process p is correct, then $susp\_level_{q \to p}(t)$ is bounded.*

$$p \in correct(F) \Rightarrow \big(\exists SL_{max} \in \mathbb{R}_0^+, \forall t \in \mathbb{T} : susp\_level_{q \to p}(t) \leq SL_{max}\big)$$

Property 4 (Known upper bound) *If process p is correct, then $susp\_level_{q \to p}(t)$ is bounded by a known value $SL_{max} \in \mathbb{R}_0^+$.*

$$p \in correct(F) \Rightarrow \big(\forall t \in \mathbb{T} : susp\_level_{q \to p}(t) \leq SL_{max}\big)$$

3.2 Accrual failure detector: definition

An accrual failure detector $\mathcal{D}_{ac}$ is a failure detector with range $(\mathbb{R}_0^+)^\Pi$, the set of all functions that map processes to non-negative real numbers (note the analogy to binary failure detectors whose range is 2^Π). In other words, failure detector modules output non-negative real values, with each value corresponding to a process and representing the current suspicion level of that process. More precisely, the history of failure detector $\mathcal{D}_{ac}$ is defined as follows.

$$H(q,t)(p) = \begin{cases} susp\_level_{q \to p}(t) & \text{if } p \neq q \\ 0 & \text{otherwise} \end{cases}$$

Concrete classes of accrual failure detectors put some restrictions on the suspicion level functions. An accrual failure detector must satisfy the following property.[5]

Property 5 *For all pairs of distinct processes p and q, the property of Accruement (Prop. 1) holds for $susp\_level_{q \to p}$.*

We define four classes of accrual failure detectors, depending on how the suspicion level satisfies the properties on an upper bound. The choice of names is not arbitrary. Indeed, as we prove later (Sect. 6), the four classes are equivalent to their respective binary counterparts.

$\mathcal{P}_{ac}$ For *all* pairs of distinct processes p and q, the properties of *Known* Upper Bound (Prop. 4) and Reset (Prop. 2) are satisfied.

[5] In this paper, we require that the Accruement property holds for all pairs of processes. However, it is possible to weaken this property, making it equivalent to the property of Weak Completeness. For simplicity and because Chandra and Toueg [3] have shown Weak and Strong Completeness to be equivalent, we do not consider the issue here.

$\Diamond\mathcal{P}_{ac}$ For *all* pairs of distinct processes p and q, the properties of *Unknown* Upper Bound (Prop. 3) and Reset (Prop. 2) are satisfied.

$\mathcal{S}_{ac}$ For *some* correct process p and any other process $q \neq p$, the properties of *Known* Upper Bound (Prop. 4) and Reset (Prop. 2) are both satisfied for $susp\_level_{q \to p}$.

$\Diamond\mathcal{S}_{ac}$ For *some* correct process p and any other process $q \neq p$, the properties of *Unknown* Upper Bound (Prop. 3) and Reset (Prop. 2) are both satisfied for $susp\_level_{q \to p}$.

4 Simple Implementation

In this section, we propose a simple heartbeat-based algorithm to implement an accrual failure detector. First, we present the system model assumed by the algorithm. Second, we describe the algorithm and prove that it implements an accrual failure detector of class $\Diamond\mathcal{P}_{ac}$. Notice that the algorithm described in this paper is intended as a simple illustration. Stochastic implementations of adaptive accrual failure detectors have been proposed [12, 13, 14]. Finally, we present the transformation of a well-known adaptive failure detector into an accrual failure detector.

Partially synchronous system model. We extend the model described in Section 2, by considering that processes communicate only by message-passing. In particular, we assume that processes have their own memory space.[6] Also, channels are reliable, and we consider only crash failures of processes.

We assume a partially synchronous model, as defined by Chandra and Toueg [3], where some *unknown* bounds on process speed and message delays hold after some *unknown* time called GST (for global stabilization time).[7]

Algorithm The algorithm (Algorithm 1) is based on heartbeats and is actually quite simple. The code of the algorithm, identical for all processes, is expressed for some arbitrary process $q \in \Pi$. A monitored process sends heartbeat messages on a regular basis (according to its own local clock). Heartbeats are sequence numbered, so that a heartbeat message with higher sequence number is considered more recent. A monitoring process q keeps track of the time of arrival $T_{last}(p)$ (according to its own local clock) of the most recent heartbeat message from a monitored process p. The value of the function $susp\_level_{q \to p}(t)$ is given by the time elapsed since the arrival of the most recent heartbeat (according to the local clock of the monitoring process).

Lemma 1 *Algorithm 1 satisfies Prop. 1 (Accruement) for $susp\_level_{q \to p}$, where p and q are two distinct processes in Π, and p is faulty.*

PROOF. To prove the property, we show that $susp\_level_{q \to p}(t)$ tends toward infinity, given that p crashes.

Since p crashes, it can send only a finite number of heartbeat messages. Let the heartbeat with the greatest sequence number arrive at time t_0. The algorithm updates $T_{last}(p)$ to t_0 at this time, and will never update $T_{last}(p)$ again. It follows that, for any time greater than t_0, the function $susp\_level_{q \to p}(t) = t - t_0$. This function obviously tends toward infinity, thus completing the proof. $\square_{\text{Lemma 1}}$

[6] In particular, this means that variables are *not* shared between processes. Although the same variable name (say, T_{last}) may be employed by two different processes (say, p and q), this always refers to two *distinct* variables (that is, T_{last} of p and T_{last} of q).

[7] This model is in fact a simple variation over the definitions of partial synchrony due to Dwork et al. [6].

Algorithm 1 Simple implementation of an accrual failure detector.

code of some process $q \in \Pi$:

```
1:  Initialization:
2:     start := now
3:     next_sn := 1                                   {Sequence number for the next heartbeat}
4:     forall p in Π − {q} do
5:        T_last(p) := start                          {Arrival time of the last heartbeat from each process}
6:        SN_last(p) := 0                             {Seq. number of the last heartbeat received}
7:  when receive (heartbeat, sn) from p               {receive heartbeat with sequence number sn}
8:     if sn > SN_last(p) then
9:        T_last(p) := now
10:       SN_last(p) := sn
11: periodically do
12:    broadcast (heartbeat, next_sn)
13:    next_sn := next_sn + 1
14: when queried about process p at time t
15:    susp_level_{q→p}(t) := if p ≠ q then t − T_last(p) else 0
```

Lemma 2 *Algorithm 1 satisfies Prop. 2 (Reset) for $susp\_level_{q \to p}$, where p and q are two distinct processes in Π, and p is correct.*

PROOF. To prove the property, we show that for any time t_0, $susp\_level_{q \to p}(t) = 0$ for some time $t \geq t_0$, given that p is correct.

As p is correct, it will send a heartbeat after t_0. Let this heartbeat be received at t; obviously, $t > t_0$. The algorithm sets $T_{last}(t) = t$ at time t, and hence a query at time t returns $susp\_level_{q \to p}(t) = 0$, thus completing the proof of the lemma. $\square_{\text{Lemma 2}}$

Lemma 3 *Algorithm 1 satisfies Prop. 3 (unknown upper bound) for $susp\_level_{q \to p}$, where p and q are two distinct processes in Π, and p is correct.*

PROOF. Let t_1 be the arrival time of the first heartbeat message H_1 sent after GST. Assume as a worst case that all messages sent prior to GST arrive after t_1. Those messages are ignored because of their lower sequence number. Hence, until t_1, $susp\_level_{q \to p}(t)$ is bounded by $t_1 - start$.

After t_1, only heartbeat messages with a higher sequence number, hence sent *after* H_1, are considered by the algorithm. It follows that they are subject to the synchrony assumptions of the model. Let Δ be the end-to-end upper bound on transmission time. Let Δ' be the maximal interval between the sending of two consecutive heartbeats.[8] It follows that the largest interval elapsed between receiving two consecutive heartbeats is $\Delta + \Delta'$.

Combining the two parts, we obtain that $susp\_level_{q \to p}(t)$ is bounded by $\max(t_1 - start, \Delta + \Delta')$.

$$\forall t \in \mathbb{T} : susp\_level_{q \to p}(t) \leq \max\left(t_1 - start, \Delta + \Delta'\right)$$

This completes the proof of the lemma. $\square_{\text{Lemma 3}}$

Theorem 4 *Algorithm 1 implements an accrual failure detector of class $\Diamond\mathcal{P}_{ac}$.*

PROOF. The proof follows directly from Lemma 1, Lemma 2, and Lemma 3, as these lemmas hold for an arbitrarily chosen pair of processes. $\square_{\text{Theorem 4}}$

[8] The exact values of Δ and Δ' depend on the synchrony assumptions on process speeds, transmission times and drift rates of local clocks with respect to global time.

4.1 Converting Chen's failure detector to an accrual one

Chen et al. [4] have proposed a well-known implementation for a network adaptive binary failure detector. Briefly speaking, their failure detector, based on heartbeats, monitors heartbeat arrivals to estimate the time EA when the next heartbeat should be expected to arrive. The algorithm sets a timeout by taking this arrival time and adding a constant safety margin α, initially computed from some QoS requirements.

There is a simple way to transform their algorithm to implement an accrual failure detector. Roughly speaking, it works as follows. When the expected arrival EA is reached (and the heartbeat is not yet received), the suspicion level begins to increase linearly over time.Now, if a process sets a constant suspicion threshold to α, the resulting failure detector is identical to Chen's original implementation. Nevertheless, the accrual failure detector can serve multiple applications with various qualities of service or applications with multiple thresholds or even more general adaptation policies.

5 Using Multiple Thresholds with Accrual Failure Detectors

In this section, we take a look at useful properties when an accrual failure detector is converted into a binary one by means of some threshold. In particular, we look at the special case where there are two failure detectors defined by two thresholds, where one threshold is always lower than the other.

Let p and q be two processes, and $susp\_level_{q \to p}$ the suspicion level function of q with respect to p. Let $T_1, T_2 : \mathbb{T} \mapsto \mathbb{R}^+$ be two threshold functions of q. Let $suspect^{T_1}_{q \to p}$ and $suspect^{T_2}_{q \to p}$ be two predicates defined as follows,

$$\forall t \in \mathbb{T}, \forall T \in \{T_1, T_2\} : suspect^{T}_{q \to p}(t) \Leftrightarrow susp\_level_{q \to p}(t) > T(t)$$

When $suspect^{T_1}_{q \to p}(t)$ (resp. $suspect^{T_2}_{q \to p}(t)$) is true, this means that process p if suspected at time t by the failure detector $\mathcal{D}_{T_1}$ (resp. $\mathcal{D}_{T_2}$) defined by threshold T_1 (resp. threshold T_2). We consider the case where $T_1(t)$ is strictly lower than $T_2(t)$ for any time t. We have the following simple theorem.

Theorem 5 *At all time, failure detector $\mathcal{D}_{T_2}$ suspects p only if failure detector $\mathcal{D}_{T_1}$ suspects p.*

$$\forall t \in \mathbb{T} : suspect^{T_2}_{q \to p}(t) \Rightarrow suspect^{T_1}_{q \to p}(t)$$

PROOF. The proof is straightforward.

$$\begin{aligned} & suspect^{T_2}_{q \to p}(t) \\ \Rightarrow \quad & susp\_level_{q \to p}(t) > T_2(t) \\ \Rightarrow \quad & susp\_level_{q \to p}(t) > T_1(t) \\ \Rightarrow \quad & suspect^{T_1}_{q \to p}(t) \end{aligned}$$

$\square_{\text{Theorem 5}}$

Chen et al. [4] propose a set of metrics to evaluate the quality of service (QoS) of failure detectors. In terms of such metrics, we can state the following corollaries when comparing $\mathcal{D}_{T_1}$ and $\mathcal{D}_{T_2}$.

Corollary 6 *Failure detector $\mathcal{D}_{T_1}$ detects failures as fast as or faster than failure detector $\mathcal{D}_{T_2}$. In other words, $T_D(\mathcal{D}_{T_1}) \leq T_D(\mathcal{D}_{T_2})$ where $T_D(\mathcal{D})$, called* detection time*, is the time that elapses from the failure of p until the failure detector module of $\mathcal{D}$ at q begins to suspect p permanently.*

Corollary 7 *Failure detector $\mathcal{D}_{T_2}$ generates wrong suspicions at most as frequently as failure detector $\mathcal{D}_{T_1}$. In other words, $T_{MR}(\mathcal{D}_{T_1}) \geq T_{MR}(\mathcal{D}_{T_2})$ where $T_{MR}(\mathcal{D})$, called* mistake recurrence time*, measures the time between two consecutive wrong suspicions made by $\mathcal{D}$.*

Corollary 8 *Failure detector $\mathcal{D}_{T_2}$ wrongly suspects a process for a duration at most as long as failure detector $\mathcal{D}_{T_1}$. In other words, $T_M(\mathcal{D}_{T_1}) \geq T_M(\mathcal{D}_{T_2})$ where $T_M(\mathcal{D})$, called* mistake duration*, measures the time that elapses from the beginning of a wrong suspicion until its end (i.e., until the mistake is corrected).*

We can also say that failure detector $\mathcal{D}_{T_1}$ is more *aggressive* than failure detector $\mathcal{D}_{T_2}$, and conversely, that $\mathcal{D}_{T_2}$ is more *conservative* than $\mathcal{D}_{T_1}$.

6 Equivalence between failure detectors

This section proves that accrual failure detector classes are equivalent to their binary counterparts. The equivalence between an accrual failure detector class $\mathcal{C}_{ac}$ and its binary equivalent $\mathcal{C}$ is important for two reasons. First, it shows that the assumptions underlying the class $\mathcal{C}_{ac}$ are not stronger than those underlying the class $\mathcal{C}$. Second, it shows that any problem that can be solved with a binary failure detector of class $\mathcal{C}$ can also be solved with an accrual failure detector of class $\mathcal{C}_{ac}$. In other words, the oracles hidden in the two equivalent failure detector classes are equally powerful.

6.1 From accrual to binary

We now prove that an accrual failure detector $\mathcal{D}_{ac}$ (where $\mathcal{D}_{ac}$ belongs to one of $\mathcal{P}_{ac}$, $\Diamond\mathcal{P}_{ac}$, $\mathcal{S}_{ac}$, or $\Diamond\mathcal{S}_{ac}$) can be reduced to a binary failure detector $\mathcal{D}$ (where $\mathcal{D}$ belongs to the respective class $\mathcal{P}$, $\Diamond\mathcal{P}$, $\mathcal{S}$, or $\Diamond\mathcal{S}$). The reduction implies, among other things, that the Consensus problem can be solved in asynchronous systems with an accrual failure detector of any of the four classes mentioned above.

We focus on the reduction from $\Diamond\mathcal{P}_{ac}$ into $\Diamond\mathcal{P}$, and then explain how to adapt the proof for the three other reductions ($\Diamond\mathcal{S}_{ac}$ into $\Diamond\mathcal{S}$, $\mathcal{P}_{ac}$ into $\mathcal{P}$, and $\mathcal{S}_{ac}$ into $\mathcal{S}$).

Algorithm 2 Transforming an accrual failure detector with unknown bound into a binary failure detector.

```
1: Initialization:
2:     T := 1                                          {threshold to suspect}
3:     suspect_{q→p} := susp_level_{q→p} > T           {true if q suspects p}
4: when suspect_{q→p} = false and susp_level_{q→p} > T
5:     suspect_{q→p} := true
6: when suspect_{q→p} = true and susp_level_{q→p} = 0
7:     suspect_{q→p} := false
8:     T := T + 1                                      {increase threshold if suspicion was wrong}
```

Consider two processes p and q, where q monitors p. Algorithm 2 uses an accrual failure detector $\mathcal{D}_{ac}$, the output of which is given by the function $susp\_level_{q\rightarrow p}$. The output of Algorithm 2 is given by the value of the boolean variable $suspect_{q\rightarrow p}$. Process p is suspected when and only when the variable $suspect_{q\rightarrow p}$is true.

The algorithm uses a dynamic threshold T to trigger suspicions. Whenever $susp\_level_{q\rightarrow p}$ rises beyond the threshold T, q begins to suspect p (or continues to suspect p). Whenever the value of $susp\_level_{q\rightarrow p}$ falls to zero, q stops suspecting p and the threshold T is increased. Very similar algorithms have been discussed by Dwork et al. [6], Chandra et al. [2], and Fetzer et al. [7], expressed with timeouts rather than an abstract threshold.

The above algorithm implements a failure detector of class $\Diamond\mathcal{P}$. To see this, it is enough to show that the Strong Completeness and the Eventual Strong Accuracy properties of the accrual failure detector yield the corresponding properties of the binary failure detector.

Lemma 9 (Strong Completeness) *Given an accrual failure detector $\mathcal{D}_{ac}$ of class $\Diamond\mathcal{P}_{ac}$, Algorithm 2 satisfies the property of Strong Completeness.*

PROOF. Consider a faulty process p and a correct process q. By assumption, $susp\_level_{q \to p}$ satisfies the Accruement property. We show that this implies that q eventually suspects p forever.

As $susp\_level_{q \to p}$ goes to infinity, there is a time t_1 after which $susp\_level_{q \to p}$ is always strictly positive. Therefore, no T-transitions occur after t_1, and thus the current threshold T_1 at time t_1 no longer changes. Again, as $susp\_level_{q \to p}$ goes to infinity, there is a time t_2 after which it is forever greater than T_1. It follows that p is permanently suspected after this time t_2. $\square_{\text{Lemma 9}}$

Lemma 10 *Given an accrual failure detector $\mathcal{D}_{ac}$ of class $\Diamond\mathcal{P}_{ac}$, Algorithm 2 satisfies the property of Eventual Strong Accuracy.*

PROOF. Let p and q be two distinct correct processes. By assumption, $susp\_level_{q \to p}$ satisfies the Unknown Upper Bound and Reset properties. Given this, we show that there is a time after which q no longer suspects p. Let SL_{max} denote the unknown bound for $susp\_level_{q \to p}$. We consider two cases.

- *Case 1.* The threshold T rises above SL_{max} during the execution of the algorithm (at some time t). It results that the transition at time t is a T-transition (S-transitions do not increase T) and no more S-transitions will occur after time t, as $susp\_level_{q \to p}$ will never reach T.
- *Case 2.* The threshold never rises above SL_{max}, hence we know that only a finite number of T-transitions occur. Hence only a finite number of S-transitions occur. Let the last S-transition happen at time t_1. The Reset property ensures that $susp\_level_{q \to p}$ reaches zero at some time $t_2 > t_1$. This triggers a T-transition that is followed by no S-transition.

Therefore, there is a time after which q no longer suspects p. $\square_{\text{Lemma 10}}$

Theorem 11 *Algorithm 2 transforms an accrual failure detector of class $\Diamond\mathcal{P}_{ac}$ into one of class $\Diamond\mathcal{P}$.*

PROOF. The proof follows directly from Lemma 9 (Strong completeness) and Lemma 10 (Eventual strong accuracy). $\square_{\text{Theorem 11}}$

We have proved the transformation from $\Diamond\mathcal{P}_{ac}$ into $\Diamond\mathcal{P}$. The proofs for the other three transformations is very straightforward and only outlined here (details are in the appendix).

Consider first the transformation from $\Diamond\mathcal{S}_{ac}$ into $\Diamond\mathcal{S}$. Algorithm 2 as it stands does this transformation. The argument is simple. $\Diamond\mathcal{S}_{ac}$ ensures that the Unknown Upper Bound and Reset properties are met for some correct process p and all other processes. The proof of Lemma 10 can easily be adapted to show that there is a time after which p is never suspected, thus proving Eventual Weak Completeness.

Now, consider the transformation from $\mathcal{P}_{ac}$ into $\mathcal{P}$ (same argument for $\mathcal{S}_{ac}$). $\mathcal{P}_{ac}$ ensures that there is a *known* upper bound, say B, on the suspicion level between any pair of correct processes. This bound is known, so Algorithm 2 is modified so that the threshold T is initially set to B. This ensures that a correct process is never suspected.

6.2 From binary to accrual

We now prove that a binary failure detector $\mathcal{D}$ (where $\mathcal{D}$ belongs to one of $\mathcal{P}$, $\Diamond\mathcal{P}$, $\mathcal{S}$, or $\Diamond\mathcal{S}$) can be transformed into an accrual failure detector $\mathcal{D}_{ac}$ (where $\mathcal{D}_{ac}$ belongs to the respective class $\mathcal{P}_{ac}$, $\Diamond\mathcal{P}_{ac}$,

$\mathcal{S}_{ac}$, or $\Diamond\mathcal{S}_{ac}$). For all four classes of failure detectors, the transformation from $\mathcal{D}$ to $\mathcal{D}_{ac}$ is done by the same algorithm (Algorithm 3).

The algorithm is expressed between two processes p and q, where q monitors p. Whenever the binary failure detector $\mathcal{D}$ trusts the monitored process p, the suspicion level is set to 0. Whenever $\mathcal{D}$ suspects p, the suspicion level is set to 2 plus the time that elapsed since the beginning of the suspicion

Algorithm 3 Transforming a binary failure detector into an accrual failure detector.

```
1: Initialization:
2:     timestamp := now
3: when suspect_{q→p} becomes true                          {S-transition}
4:     timestamp := now
5: when queried
6:     if suspect_{q→p} then
7:         susp_level_{q→p} := 2 + now − timestamp          {always at least 2}
8:     else
9:         susp_level_{q→p} := 0
```

Lemma 12 (Accruement) *Algorithm 3 satisfies the property of Accruement (Prop. 1) for some binary failure detector $\mathcal{D}$ of class $\mathcal{P}$, $\mathcal{S}$, $\Diamond\mathcal{P}$, or $\Diamond\mathcal{S}$.*

PROOF. Consider a faulty process p and a correct process q. By definition, $\mathcal{D}$ ensures that p is eventually suspected permanently (Strong Completeness; $\mathcal{D}$ is of class $\mathcal{P}$, $\mathcal{S}$, $\Diamond\mathcal{P}$, or $\Diamond\mathcal{S}$). Let t_1 be the time when the last S-transition occurs (or the starting time of the algorithm if no S-transition occurs). The resulting suspicion level function is $susp\_level_{q\to p}(t) = 2 + t - t_1$ after t_1. This function clearly goes to infinity with time, thus proving the Accruement property. $\square_{\text{Lemma 12}}$

Theorem 13 *Algorithm 3 transforms $\mathcal{P}$ into $\mathcal{P}_{ac}$ and $\mathcal{S}$ into $\mathcal{S}_{ac}$.*

PROOF. From Lemma 12, Algorithm 3 satisfies the Accruement property.

Consider the transformation from $\mathcal{S}$ into $\mathcal{S}_{ac}$ (the proof for the transformation of $\mathcal{P}$ into $\mathcal{P}_{ac}$ is nearly identical). Let p be a correct process that is never suspected by $\mathcal{D}$ (Weak Accuracy). Let q be any other correct process. Since p is never suspected, $susp\_level_{q\to p} = 0$ always. Known Upper Bound (Prop. 4) and Reset (Prop. 2) are both trivially satisfied. $\square_{\text{Theorem 13}}$

Theorem 14 *Algorithm 3 transforms $\Diamond\mathcal{P}$ into $\Diamond\mathcal{P}_{ac}$ and $\Diamond\mathcal{S}$ into $\Diamond\mathcal{S}_{ac}$.*

PROOF. From Lemma 12, Algorithm 3 satisfies the Accruement property.

Consider the transformation from $\Diamond\mathcal{S}$ into $\Diamond\mathcal{S}_{ac}$ (the proof for the transformation of $\Diamond\mathcal{P}$ into $\Diamond\mathcal{P}_{ac}$ is nearly identical). Let p be the correct process that is never suspected after some time t by any other correct process q (Eventual Weak Accuracy). We prove Unknown Upper Bound and Reset for $susp\_level_{q\to p}$.

Again, the proof is easy: $susp\_level_{q\to p}$ is constantly equal to 0 after t, hence Reset is trivially satisfied. The highest suspicion level occurs if there is only one suspicion lasting from the starting time t_0 of the algorithm until time t: this level is $2 + t - t_0$. Therefore Unknown Upper Bound is satisfied.[9]

[9] Note that the time t is not known a priori, hence Known Upper Bound does not hold.

□Theorem 14

Finally, note that Algorithm 3 has an interesting property. Consider any binary failure detector $\mathcal{D}$. If $\mathcal{D}$ is transformed into an accrual failure detector $\mathcal{D}_{ac}$ by Algorithm 3 and back into a binary failure detector by setting a threshold of 1, the result is $\mathcal{D}$ itself. In other words, using the accrual failure detector $\mathcal{D}_{ac}$ with suspicion threshold 1 results in the same sequence of suspicions as generated with $\mathcal{D}$ (with transitions occurring at the same times).

7 Conclusion

Failure detectors constitute a fundamental abstraction for fault-tolerant distributed systems. However, from a more practical perspective, the binary model of these failure detectors is limited by the fact that they combine monitoring and interpretation. The accrual failure detectors presented in this paper decouple these two issues by outputting a suspicion level rather than a binary value, and leaving it to processes to interpret that value. Ideally, the monitoring can be done by a single service running on each machine, while the interpretation is left to each application process. Such a service can be implemented as a daemon, a shared library or a kernel service, depending on the desired tradeoff between intrusiveness and performance.

The equivalence results presented in this paper show that accrual failure detectors do not hide any additional synchrony assumptions with respect to their binary counterpart. In addition, equivalence with other failure detectors can be deduced for other results (e.g., between the eventual leader Ω and $\Diamond\mathcal{S}_{ac}$ by transitivity from equivalence proved by Chandra et al. [2] and Chandra and Toueg [3]).

References

[1] M. K. Aguilera, W. Chen, and S. Toueg. Using the heartbeat failure detector for quiescent reliable communication and consensus in partitionable networks. *Theor. Comput. Science*, 220(1):3–30, June 1999.

[2] T. D. Chandra, V. Hadzilacos, and S. Toueg. The weakest failure detector for solving consensus. *J. ACM*, 43(4):685–722, July 1996.

[3] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267, 1996.

[4] W. Chen, S. Toueg, and M. K. Aguilera. On the quality of service of failure detectors. *IEEE Trans. on Computers*, 51(5):561–580, May 2002.

[5] F. J. N. Cosquer, L. T. Rodrigues, and P. Veríssimo. Using tailored failure suspectors to support distributed cooperative applications. In *Proc. 7th IASTED Intl. Conf. on Parallel and Distributed Computing and Systems (PDCS'95)*, pages 352–356, Washington, DC, USA, October 1995.

[6] C. Dwork, N. A. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, April 1988.

[7] C. Fetzer, M. Raynal, and F. Tronel. An adaptive failure detection protocol. In *Proc. 8th IEEE Pacific Rim Symp. on Dependable Computing (PRDC'01)*, pages 146–153, Seoul, Korea, December 2001.

[8] R. Friedman. Fuzzy group membership. In A. Schiper, A. A. Shvartsman, H. Weatherspoon, and B. Y. Zhao, editors, *Future Directions in Distributed Computing*, number 2584 in LNCS, pages 114–118. Springer-Verlag, January 2003. Position paper.

[9] R. Friedman and G. Tcharny. Evaluating failure detection in mobile ad-hoc networks. TR CS-2003-06, Technion, Israel, October 2003.

[10] R. Friedman and G. Tcharny. Fuzzy membership based reliable delivery for mobile ad-hoc networks. TR CS-2003-14, Technion, Israel, December 2003.

[11] R. Friedman and G. Tcharny. Stability detection in mobile ad-hoc networks. TR CS-2003-12, Technion, Israel, November 2003.

[12] N. Hayashibara, X. Défago, and T. Katayama. Two-ways adaptive failure detection with the φ-failure detector. In *Proc. Workshop on Adaptive Distributed Systems (WADiS'03)*, pages 22–27, Sorrento, Italy, October 2003.

[13] N. Hayashibara, X. Défago, and T. Katayama. Flexible failure detection with κ-fd. RR IS-RR-2004-006, Japan Advanced Institute of Science and Technology, Ishikawa, Japan, February 2004.

[14] N. Hayashibara, X. Défago, R. Yared, and T. Katayama. The φ accrual failure detector. RR IS-RR-2004-010, Japan Advanced Institute of Science and Technology, Ishikawa, Japan, May 2004.

[15] A. Mostéfaoui, M. Raynal, and C. Travers. Crash-resilient time-free eventual leadership. TR, IRISA, Rennes, France, April 2004.

[16] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, April 1980.

[17] L. M. R. Sampaio, F. V. Brasileiro, W. Cirne, and J. C. A. Figueiredo. How bad are wrong suspicions? towards adaptive distributed protocols. In *Proc. IEEE Intl. Conf. on Dependable Systems and Networks (DSN'03)*, pages 551–560, San Francisco, CA, USA, June 2003.

A Appendix: Proofs

We present here some of the proofs that were only summarized in the main body of the paper.

A.1 Reduction: from accrual to binary

Algorithm 4 transforms $\mathcal{P}_{ac}$ to $\mathcal{P}$ and $\mathcal{S}_{ac}$ to $\mathcal{S}$. The algorithm uses the known upper bound on the suspicion level as a threshold to both suspect and trust the monitored process p. Unlike in Algorithm 2, there is no need to dynamically modify the threshold used to suspect p.

Algorithm 4 Transforming an accrual failure detector with known bound into a binary failure detector.

1: Initialization:
2: $T :=$ bound on the accrual value {*threshold to both suspect and trust*}
3: **when** queried
4: $suspect_{q \to p} := susp\_level_{q \to p} > T$

Lemma 15 (Strong Completeness) *Given an accrual failure detector that satisfies Accruement, Algorithm 4 satisfies Strong Completeness.*

PROOF. Consider a faulty process p and a correct process q. We show that the Accruement property of $susp\_level_{q \to p}$ implies that q will eventually suspect p forever. This holds because $susp\_level_{q \to p}$ goes to infinity, and thus there is a time t at which the suspicion level rises above the threshold T. Hence the failure detector permanently suspects p after this time t. $\square_{\text{Lemma 15}}$

Lemma 16 (Accuracy) *Given an accrual failure detector of class $\mathcal{P}_{ac}$ (resp. $\mathcal{S}_{ac}$), Algorithm 4 satisfies Strong Accuracy (resp. Weak Accuracy).*

PROOF. Consider any two correct processes p and q ($p \neq q$) when proving Strong Accuracy (for the transformation $\mathcal{P}_{ac}$ to $\mathcal{P}$) and the correct process p to which Weak Accuracy refers, and any other correct process q, when proving Weak Accuracy (for the transformation $\mathcal{S}_{ac}$ to $\mathcal{S}$). For these two processes p and q, we show that the Known Upper Bound and Reset properties imply that q never suspects p. The proof is simple: Known Upper Bound implies that the suspicion level will never rise above the threshold T. $\square_{\text{Lemma 16}}$

Lemma 17 (Weak Accuracy) *Given an accrual failure detector of class $\mathcal{S}_{ac}$, Algorithm 4 satisfies Weak Accuracy.*

PROOF. The proof is a simple adaptation of that of Lemma 16 with the difference that the correct process p for which the Known Upper Bound holds is the correct process that is never suspected. $\square_{\text{Lemma 17}}$

Theorem 18 *Algorithm 4 transforms an accrual failure detector of class $\mathcal{P}_{ac}$ into a binary one of class $\mathcal{P}$.*

PROOF. The proof follows directly from Lemma 15 (Strong Completeness) and Lemma 16 (Strong Accuracy). $\square_{\text{Theorem 18}}$

Theorem 19 *Algorithm 4 transforms an accrual failure detector of class $\mathcal{S}_{ac}$ into a binary one of class $\mathcal{S}$.*

PROOF. Follows directly from Lemma 15 (Strong Completeness) and Lemma 17 (Weak Accuracy). $\square_{\text{Theorem 19}}$