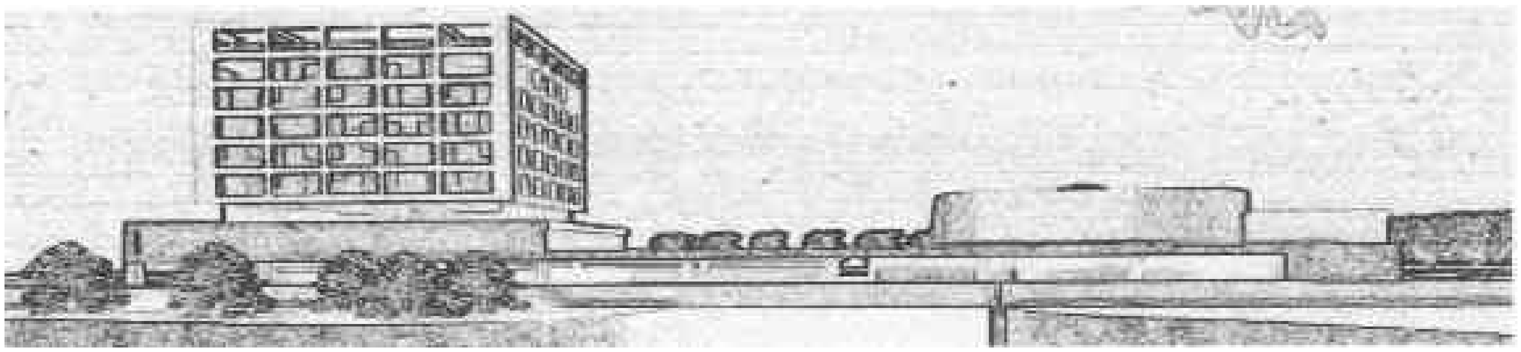
VOLUME V, NUMBER 3



REPORTS ON SYSTEMS AND COMMUNICATIONS

Móstoles (Madrid), January 2005
Depósito Legal: M-50653-2004
ISSN: 1698-7489

Table of Contents

Minimal System Conditions to Implement Unreliable Failure Detectors [⋆]

Antonio Fernández, Ernesto Jiménez, and Sergio Arévalo

[1] LADyR, GSyC, Universidad Rey Juan Carlos, 28933 Móstoles, Spain
[2] EUI, Universidad Politécnica de Madrid, 28031 Madrid, Spain
[3] LADyR, GSyC, Universidad Rey Juan Carlos, 28933 Móstoles, Spain

Abstract. In this paper we explore the minimal system requirements to implement unreliable failure detectors. We first consider systems formed by lossy asynchronous and eventually timely links. On these systems we define two properties, the *Weak Property* and the *Strong Property*, depending on whether all correct processes can be reached with links that are not lossy asynchronous from one or from all correct processes, respectively. We present necessary conditions based on these properties. We show that there is no algorithm that implements $\diamond\mathcal{S}$, Ω, nor $\mathcal{S}$ (resp. $\diamond\mathcal{P}$ nor $\mathcal{P}$) if we allow one single failure in a system that, when all processes are correct, does not satisfy the Weak (resp. Strong) Property. Then, we propose an algorithm that implements $\diamond\mathcal{P}$ if the Strong Property is satisfied, and $\diamond\mathcal{S}$ (and Ω with an additional assumption) if only the Weak Property is satisfied. For systems formed by synchronous and lossy asynchronous links only, we propose another algorithm that implements detector class $\mathcal{P}_4$ if the Strong Property is satisfied, and implements a new detector class $\mathcal{S}'$ (and Ω with an additional assumption) if only the Weak Property is satisfied.

1 Introduction

Unreliable failure detectors were proposed by Chandra and Toueg [1] as devices that can be used to solve consensus in asynchronous systems with crash failures. These devices allow to circumvent the seminal result from Fischer et al [2] that shows the impossibility of solving consensus in these systems even with one single failure. Beyond consensus, it is also known that unreliable failure detectors are useful to solve other fundamental problems in distributed computing. For these reasons there is a growing interest in deriving practical efficient algorithms to implement failure detectors [3–10].

In a tutorial at PODC 2002, Keidar and Rajsbaum [11] asked, among other questions, for the weakest requirements on systems that allow to implement the different classes of failure detectors. In their recent works Aguilera et al. [5, 4] have partially answered this question for the class of failure detectors Ω

[⋆] Revised: June 2005; Revised: September 2006. Partially supported by the Spanish MEC under grants TIN2005-09198-C02-01, TIN2004-07474-C02-02, and TIN2004-07474-C02-01, and the Comunidad de Madrid under grant S-0505/TIC/0285.

[12]. As usual, Aguilera et al. consider that all the processes are connected by directed links. They consider several classes of links. A *timely* link is a reliable link with a known upper bound on the message delivery time. An *eventually timely* link is a link that behaves timely (but the bound may not be known) after some unknown stabilization time. Before this time messages can be lost. A *fair lossy* link is one in which if infinite messages are sent, infinite messages are received (but infinite messages can also be lost). Finally, a *lossy asynchronous* link is one in which messages can be lost and there is no bound on the message delivery time. In [6] they present an algorithm that implements Ω in a system with all links lossy asynchronous except the (input and output) links of an (unknown) correct process. Later in [5] they show that a failure detector of class Ω can be implemented in a system with lossy asynchronous links as long as there is one (unknown) correct process whose *output* links are eventually timely. However, they prove that in this case any algorithm will have runs in which $\Omega(n^2)$ links are permanently carrying messages. They also show that if, additionally, some (unknown) correct process has all its (input and output) links fair lossy, Ω can be implemented efficiently, i.e. eventually only one process (the leader) sends messages. In [13], it is shown that Ω can be implemented as long as all correct processes can be reached from a (unknown) correct process with eventually timely paths (paths of eventually timely links and correct processes), even if initially processes only know their own identity. It is also shown there that none of the classes originally proposed in [1] can be implemented if the identity of (at least) one process is unknown to the rest of processes.

In [4], it is shown that Ω can be implemented in a system with fair lossy links, in which at most t processes can crash and t is known, if some correct process has t eventually timely output links. If additionally $n > 2t$, then consensus can be solved. If the links are reliable, they also have an algorithm in which eventually only t links carry messages. They also show that even if all processes have $t-1$ timely output links and all links are reliable neither Ω can be implemented nor consensus can be solved. In [14] it is shown that if initially processes only know their own identity and t, Ω can be implemented if all correct processes are connected via fair lossy paths (paths of fair lossy links and correct processes) and there is one process that can reach $t-f$ (f is the actual number of failures in the run) other correct processes via eventually timely paths. Additionally, if all correct processes are fully connected with fair lossy links and one process has all output links eventually timely, Ω can be implemented efficiently.

Regarding other classes of failure detectors, Larrea et al. [8] consider the implementation of the eight original classes proposed by Chandra and Toueg (namely, the *perpetual* classes $\mathcal{P}$, $\mathcal{S}$, $\mathcal{Q}$, and $\mathcal{W}$, and the *eventual* classes $\diamond\mathcal{P}$, $\diamond\mathcal{S}$, $\diamond\mathcal{Q}$, and $\diamond\mathcal{W}$) in systems with all links eventually timely. They show that no perpetual detector can be implemented in these systems even if only one process can fail. Then, they show that all eventual detectors can be implemented in these systems. Implicitly, all it is required for this possibility result is that all the links in a ring formed by the correct processes are eventually timely. Only these links carry messages forever. Another related work is [9], in which an algorithm that implements $\diamond\mathcal{S}$ and Ω in a system with all links eventually timely is proposed. With this algorithm, eventually only the output links of one process (the correct process with smallest identifier) carry messages. In fact, the synchrony requirement they impose could be relaxed so that only these links are required to be eventually timely.

A different approach to limit the system requirements to implement failure detectors has been proposed by Mostefaoui et al. [15, 16]. In these works the condition on the system has to do with its behavior. They show that their algorithms implement a $\diamond\mathcal{S}$ detector if messages are received by the processes in the appropriate order. They do a probabilistic analysis and show that these requirements are met with high probability for one single failure.

Contributions. In this paper we continue the exploration of the system limits to implement unreliable failure detectors. We study five of the traditional classes of failure detectors: Ω, $\mathcal{P}$, $\mathcal{S}$, $\diamond\mathcal{P}$, and $\diamond\mathcal{S}$, and two additional (perpetual) classes, $\mathcal{P}_4$ and $\mathcal{S}'$, which are weak versions (they have weaker accuracy) of $\mathcal{P}$ and $\mathcal{S}$, respectively. As far as we know, the class $\mathcal{S}'$ has never been previously proposed and seems an interesting class to study further. The class $\mathcal{P}_4$ was first proposed in [17].

We consider systems formed either by lossy asynchronous and timely links (class Ψ), or lossy asynchronous and eventually timely links (class $\mathcal{E}$). In the complete directed graph formed by the processes and the links, we call a path that does not contain lossy asynchronous links nor faulty processes a *good* path. We are interested on the set R of correct processes that can reach all correct processes via good paths. We explore the implementability of failure detectors depending on whether the set R has at least one correct process (*Weak Property)* or R contains all correct processes (*Strong Property*). Additionally, we say that we have the *Min Property* if R contains the correct processor with smallest identifier. Observe that the Strong Property implies the Min Property, and the latter implies the Weak Property.

First, we show that there is no algorithm that implements $\diamond\mathcal{P}$ (and hence $\mathcal{P}$ and $\mathcal{P}_4$) with single failures in a system that, when all processes are correct, does not satisfy the Strong Property. Similarly, we show that $\diamond\mathcal{S}$ (and hence Ω, $\mathcal{S}$, and $\mathcal{S}'$) cannot be implemented if we allow one single failure in a system that, when all processes are correct, does not satisfy the Weak Property.

Then, we present algorithms that work on minimal system conditions. First, we propose an algorithm for the systems in Ψ that implements $\mathcal{P}_4$ if the Strong Property is satisfied and implements $\mathcal{S}'$ if only the Weak Property is satisfied. We propose a second algorithm for the systems in $\mathcal{E}$ that implements $\diamond\mathcal{P}$ if the Strong Property is satisfied and $\diamond\mathcal{S}$ if only the Weak Property is satisfied. If the Min Property is satisfied, both algorithms implement Ω.

For the systems in $\mathcal{E}$, the Weak Property is the minimal requirement imposed above to implement Ω (with f unknown) [13]. The Min Property is also strictly weaker than the requirements in [9]. However it cannot be compared with those in [5], since they have no requirement regarding the correct process with smallest identifier. Note as well that the Strong Property is weaker than the property for

$\Diamond\mathcal{P}$ in [8], which requires all links connecting the correct processes in a ring to be eventually timely.

The rest of the paper is structured as follows. In Section 2 we present the model and notations we use in the rest of the paper. In Section 3 we show necessary conditions to implement the classes of failure detectors we consider here. Section 4 presents an algorithm that implements perpetual failure detectors in systems with weak synchrony. Similarly, Section 5 presents an algorithm that implements eventual failure detectors in systems with weak synchrony. Finally, Section 7 contains some concluding remarks.

2 The model

We consider systems formed by a finite set Π of $n > 1$ processes. Processes have unique and totally ordered identifiers, known to all the processes. We assume that processes can communicate with each other only by sending and receiving messages, and that every pair of processes is connected by a pair of directed links (with opposite directions). Let $\Lambda = \{(p, q) : p, q \in \Pi; p \neq q\}$ denote the set of directed links of a system. Clearly, if we see the system as a graph $G = (\Pi, \Lambda)$, G is a complete directed graph.

Failure-prone processes. A process can fail by permanently crashing. We say that a process is correct if it does not fail. We denote by *correct* the set of correct processes. The complementary set $\Pi - \mathit{correct}$ is denoted by *crashed*. We assume that the algorithms have no a priori knowledge of the number of failures that can occur.

To add generality, for the impossibility results (necessary conditions) of Section 3 we assume that the execution of processes is synchronous, and that their clocks are synchronized. However, we do not need such strong assumptions in our algorithms. In them, we only assume that each line of the algorithm takes no more than σ time to be executed (σ can be a very loose bound). For the *perpetual detectors algorithm* (Section 4) we additionally assume that σ is known to the algorithm. In the algorithms we also assume the availability of timers at each process. We use $\tau_p(T)$ to denote the time it takes a timer of process p started with a value T to expire. We first require that, for all p, $\tau_p(T)$ is finite if T is finite. Then, for the *perpetual detectors algorithm* we also require that for all p and all T, $\tau_p(T) \geq T$, while for the *eventual detectors algorithm* (Section 5) we only require that $\tau_p(T)$ is non-decreasing with T and $\lim_{T\to\infty} \tau_p(T) = \infty$. Note that processes do not use global clocks.

Unreliable failure detector classes. Chandra and Toueg defined several classes of unreliable failure detectors by specifying their corresponding *completeness* and *accuracy.* These properties are defined on the lists of suspected processes maintained by (the failure detector modules of) the processes. The completeness property requires that every process that actually crashes is eventually suspected, while the accuracy property restricts the mistakes (i.e., the false suspicions) that a failure detector can make. Chandra and Toueg defined two completeness and four accuracy properties in [1]. We only consider here one completeness property:

– *Strong Completeness:* Eventually every process that crashes is permanently suspected by *every* correct process.

Regarding accuracy, we consider the four properties proposed by Chandra and Toueg:

– *(Perpetual) Strong Accuracy:* No process is suspected before it crashes.
– *(Perpetual) Weak Accuracy:* Some correct process is never suspected.
– *Eventual Strong Accuracy:* There is a time after which no correct process is ever suspected by any correct process.
– *Eventual Weak Accuracy:* There is a time after which some correct process is never suspected by any correct process.

We are going to consider here two more accuracy properties, which are slightly weaker than Strong Accuracy and Weak Accuracy, respectively, as defined above. The first property has been stated in [18, 17]. The second has never been stated as far as we know.

– *(Perpetual) Quasi-Strong Accuracy.* No correct process is ever suspected by any correct process.
– *(Perpetual) Quasi-Weak Accuracy.* Some correct process is never suspected by any correct process.

Failure detectors with eventual accuracy may suspect *every* process at one time or another, while failure detectors with perpetual accuracy require that at least one correct process is never suspected. Combining the completeness property with one of the accuracy properties we obtain one class of failure detectors. We consider here six different classes, which are presented in Figure 1. Quasi-Strong Accuracy combined with Strong Completeness yields a weak version of the class $\mathcal{P}$, which has been denoted in [17] by $\mathcal{P}_4$. Quasi-Weak Accuracy combined with Strong Completeness yields a weak version of the class $\mathcal{S}$, which we are going to denote here by $\mathcal{S}'$.

Completeness	**Accuracy**					
	Strong	Weak	Quasi-Strong	Quasi-Weak	Eventual Strong	Eventual Weak
Strong	$\mathcal{P}$	$\mathcal{S}$	$\mathcal{P}_4$	$\mathcal{S}'$	$\diamond\mathcal{P}$	$\diamond\mathcal{S}$

Fig. 1. Classes of failure detectors we consider, defined in terms of completeness and accuracy.

Finally, we define the Ω failure detector. In an Ω failure detector, each process chooses some process in the system as *leader*. The detector must guarantee that all correct processes eventually agree on a single correct process as leader. More formally, Ω failure detectors must satisfy the following property.

Property 1. There is a time after which every process $p \in \mathit{correct}$ permanently has the same process $l \in \mathit{correct}$ as leader.

We will consider in this paper that an Ω failure detector also implements $\diamond\mathcal{S}$, because the set $\Pi \setminus \{\mathit{leader}\}$ guarantees the properties of $\diamond\mathcal{S}$.

Types of links. We consider the following three types of links[4].

(i) *Lossy asynchronous (LA):* A message sent across a lossy asynchronous link can be lost or arbitrarily delayed; however, a message that is not lost will eventually be delivered. (Note that all messages sent using a lossy asynchronous link may be lost.)
(ii) *Timely (T):* The link is reliable and there is a *known* bound Δ on the maximum message delay. (Hence, a message that is sent at time t is received by time $t + \Delta$.)
(iii) *Eventually timely (ET):* There is a *possibly unknown* global stabilization time GST such that until GST the link behaves like lossy asynchronous; after GST the link is reliable and there is a *possibly unknown* bound Δ on the maximum message delay. (Hence, a message that is sent at time $t > GST$ is received by time $t + \Delta$.)

We assume that links do not modify the messages they carry nor they generate spontaneous messages. In order to make the negative results stronger, we assume for the impossibility results that links do not replicate messages and that they deliver them in order. However, in our algorithms we allow messages to be replicated and received out of order. In fact, our algorithms explicitly resend messages, creating replicated messages. We finally assume that messages are unique, in the sense that an algorithm can determine whether a message received is a duplicate (either generated by the link or resent by some other process) of a previously received message. This can be easily implemented using a message identifier formed by the unique identifier of the sending process and a sequence number, unique for that process.

Classes of systems. In this paper we consider two large classes of systems. A system belongs to the class Ψ if each of its links[5] is either lossy asynchronous or timely. A system belongs to the class $\mathcal{E}$ if each of its links is either lossy asynchronous or eventually timely. Since timely links are special cases of eventually timely links (with $GST = 0$ and known Δ), we have that $\Psi \subset \mathcal{E}$. From now on, when we assume a system in $\mathcal{E}$ we also include those in Ψ.

Then, we consider a system $S \in \mathcal{E}$ characterized by the pair $(L_S, \mathit{correct}_S)$, where $L_S : \Lambda \rightarrow \{LA, T, ET\}$, and $\mathit{correct}_S \subseteq \Pi$[6]. We denote this by $S = (L_S, \mathit{correct}_S)$. L_S is a function that determines for each link in S its type. Of

[4] Observe that the type of a link depends on its observed behavior. This behavior can be satisfied in all or in a specific set of runs of the system.
[5] I.e., the behavior of its links in each run.
[6] Intuitively, a system as defined is the set of all runs in which all processes in $\mathit{correct}_S$ are correct and the links behave as implied by L_S.

course L_S must be consistent with the class to which S belongs. Then if $S \in \Psi$, L_S can take the values LA and T only. The set $correct_S$ is the set of correct processes in S. When convenient we will use the complementary set $crashed_S$, and when clear from the context we will remove the subindex.

For any system $S \in \mathcal{E}$ we are going to define an associated graph $G(S)$ which is derived from the attributes of S. We mentioned above that S can be seen as the complete directed graph (Π, Λ). Then, $G(S)$ is the directed subgraph induced in this graph by the set $correct_S$, from which all the lossy asynchronous links (links with $L_S(p, q) = LA$) have been removed. Then, $G(S)$ only contains correct processes as vertices and directed timely links (if $S \in \Psi$) or directed eventually timely links (if $S \in \mathcal{E}$).

Given two correct processes p and q in S, we say that q can be reached from p if either $p = q$ or there is a directed path from p to q in $G(S)$. The set of processes that can be reached from a process p is denoted by $reach(p)$. It can be trivially observed that

Observation 1 *If $q \in reach(p)$, then $reach(q) \subseteq reach(p)$.*

We define now the following properties that can be satisfied by a system $S \in \mathcal{E}$.

Property 2 (Weak). There is some process $p \in correct$ such that $reach(p) = correct$ in $G(S)$.

Property 3 (Min). $reach(\min(correct)) = correct$ in $G(S)$.

Property 4 (Strong). For all process $p \in correct$, $reach(p) = correct$ in $G(S)$.

Observe that the Strong Property implies the Min Property, and the Min Property implies the Weak Property.

Finally, we will use the following notation. Given a system $S = (L_S, correct_S) \in \mathcal{E}$, we denote by $S(p)$, for $p \in correct_S$, the system obtained from S by removing p from the set $correct_S$. That is, $S(p) = (L_S, correct_S \setminus \{p\})$. Then, we denote by $\Phi(S)$ the set that contains S and all systems $S(p)$, $p \in correct_S$, i.e. $\Phi(S) = \{S\} \cup \{S(p) : p \in correct_S\}$.

Algorithms. We study here algorithms that implement unreliable failure detectors of the above classes. These algorithms are implemented as one local module for each process of the system. A module exists as long as its local process has not crashed. Modules exchange messages among each other to provide the required completeness and accuracy properties. They also, upon request by their local process, provide it with the current list of suspected processes or the current leader.

An algorithm $\mathcal{A}$ implements a failure detector of a given class C for a set of systems Σ. If it is claimed that $\mathcal{A}$ implements the detector for a system set Σ, and $S \in \Sigma$, then *every* run of $\mathcal{A}$ in S must guarantee that the implemented detector belongs to C, independently of when the processes in $crashed_S$ fail and the behavior of the links (as long as they are consistent with their type). The sets of systems we consider here are subsets of the class $\mathcal{E}$. We will make clear the subset of systems for which an algorithm implements the detector in each case.

3 Necessary conditions to implement failure detectors

In this section we obtain minimal conditions that systems in $\mathcal{E}$ must satisfy to be able to implement a failure detector of the classes of interest.

3.1 Conditions for detectors with weak accuracy

We consider first the detector classes $\diamond\mathcal{S}$, Ω, $\mathcal{S}'$, and $\mathcal{S}$. We show that it is not possible to implement a $\diamond\mathcal{S}$ detector (and hence Ω, $\mathcal{S}$, and $\mathcal{S}'$ detectors) for a set of systems that contains one system S that does not satisfy the Weak Property and those obtained from it with one more failure (denoted $S(p)$). Recall that we denote this set by $\Phi(S)$. The following theorem shows this.

Theorem 1. *Let $S \in \mathcal{E}$ be a system that does not satisfy the Weak Property, $S(p)$ be the system obtained from S by removing process $p \in correct_S$ from the set of correct processes, and $\Phi(S) = \{S\} \cup \{S(p) : p \in correct_S\}$. There is no algorithm that implements a $\diamond\mathcal{S}$ failure detector for the set of systems $\Phi(S)$.*

Proof. For the sake of contradiction, let us assume there is such an algorithm $\mathcal{A}$. Let us consider a run R of $\mathcal{A}$ in S in which all the messages sent across lossy asynchronous links are lost. By Eventual Weak Accuracy, there will be a process $p \in correct_S$ and a time t such that p is never suspected by any correct process after t. Now, note that the set of processes $Q = correct_S \setminus reach(p)$ is not empty, since the Weak Property does not hold. Furthermore, from Observation 1, no message from the processes in $reach(p)$ ever reaches any process in Q.

Let us consider now system $S(p)$. By assumption, $\mathcal{A}$ should also implement a $\diamond\mathcal{S}$ detector in $S(p)$. Let us consider a run R' of $\mathcal{A}$ in $S(p)$ in which all the messages sent across lossy asynchronous links are lost and all processes behave as closely as possible to their behavior in R. Note that the processes in Q do not have any way of distinguishing R' from R, since like before they receive no message from $reach(p)$, which are the processes that noticed the failure of p. Hence, they can in fact behave exactly like in R, and they will never suspect p after time t. This violates the Strong Completeness property, and hence $\mathcal{A}$ cannot exist.

Clearly, if there is no algorithm for a given set, there is no algorithm for any of its supersets. Also, since all detectors in Ω, $\mathcal{S}$, and $\mathcal{S}'$ are also in $\diamond\mathcal{S}$, we have the following corollary.

Corollary 1. *Let $S \in \mathcal{E}$ be a system that does not satisfy the Weak Property, and $\Sigma \supseteq \Phi(S)$ be a set of systems. There is no algorithm that implements a $\diamond\mathcal{S}$, Ω, $\mathcal{S}'$, or $\mathcal{S}$ failure detector for the set of systems Σ.*

Note that this holds even if S has no failures.

Corollary 2. *Let $S \in \mathcal{E}$ be a system* without failures *that does not satisfy the Weak Property, and Σ be a set of systems that include S and the systems obtained from S with one single failure (i.e., $\Sigma \supseteq \Phi(S)$). There is no algorithm that implements a $\diamond\mathcal{S}$, Ω, $\mathcal{S}'$, or $\mathcal{S}$ failure detector for the set of systems Σ.*

3.2 Conditions for detectors with strong accuracy

Let us now look at detectors with some form of strong accuracy. Like before, we will show first that there is no algorithm to implement a detector in $\diamond\mathcal{P}$ for a set of systems that contains a system that does not satisfy the Strong Property and those obtained from it with one additional failure. The following theorem shows this.

Theorem 2. *Let $S \in \mathcal{E}$ be a system that does not satisfy the Strong Property, $S(p)$ be the system obtained from S by removing process $p \in correct_S$ from the set of correct processes, and $\Phi(S) = \{S\} \cup \{S(p) : p \in correct_S\}$. There is no algorithm that implements a $\diamond\mathcal{P}$ failure detector for the set of systems $\Phi(S)$.*

Proof. For the sake of contradiction, let us assume there is such an algorithm $\mathcal{A}$. Let us consider a run R of $\mathcal{A}$ in S in which all the messages sent across lossy asynchronous links are lost. By Eventual Strong Accuracy, there will be a time t such that no correct process is ever suspected by any correct process after t. Since S does not satisfy the Strong Property, there is some process $p \in correct_S$ such that $reach(p) \neq correct_S$. Then, the set of processes $Q = correct_S \setminus reach(p)$ is not empty (note that $reach(p)$ only contains correct processes). Furthermore, from Observation 1, no message from the processes in $reach(p)$ ever reaches any process in Q.

Let us consider now system $S(p)$. By assumption, $\mathcal{A}$ should also implement a $\diamond\mathcal{P}$ detector in $S(p)$. Let us consider a run R' of $\mathcal{A}$ in $S(p)$ in which all the messages sent across lossy asynchronous links are lost and all processes behave as closely as possible to their behavior in R. Note that the processes in Q do not have any way of distinguishing R' from R, since like before they receive no message from $reach(p)$, which are the processes that noticed the failure of p. Hence, they can in fact behave exactly like in R, and they will never suspect p after time t. This violates the Strong Completeness property, and hence $\mathcal{A}$ cannot exist.

Again, if there is no algorithm for a given set, there is no algorithm for any of its supersets. Also, since the detectors in $\mathcal{P}$ and $\mathcal{P}_4$ are also in $\diamond\mathcal{P}$, we have the following corollary.

Corollary 3. *Let $S \in \mathcal{E}$ be a system that does not satisfy the Strong Property, and $\Sigma \supseteq \Phi(S)$ be a set of systems. There is no algorithm that implements a $\diamond\mathcal{P}$, $\mathcal{P}_4$, or $\mathcal{P}$ failure detector for the set of systems Σ.*

Corollary 4. *Let $S \in \mathcal{E}$ be a system* without failures *that does not satisfy the Strong Property, and Σ be a set of systems that include S and the systems obtained from S with one single failure (i.e., $\Sigma \supseteq \Phi(S)$). There is no algorithm that implements a $\Diamond\mathcal{P}$, $\mathcal{P}_4$, or $\mathcal{P}$ failure detector for the set of systems Σ.*

4 Algorithm for perpetual failure detectors

In this section we present an algorithm that implements a failure detector for systems in the class Ψ. For all the systems in Ψ that satisfy the Weak Property the algorithm implements a detector of class $\mathcal{S}'$. If additionally they satisfy the Min Property the algorithm implements a detector of class Ω. For all the systems that satisfy the Strong Property the algorithm implements a detector of class $\mathcal{P}_4$. Figure 2 presents the algorithm in detail. For all $p \in \Pi$, the sets $suspected_p$ provide the required completeness and accuracy properties for $\mathcal{S}'$ and $\mathcal{P}_4$, while the values $leader_p$ satisfy Property 1 for Ω.

```
Algorithm Perpetual
init:
 (1)  $suspected_p \leftarrow \emptyset$
 (2)  $leader_p \leftarrow \min(\Pi)$
 (3)  $Timeout_p \leftarrow \eta + (n-1)(\Delta + 4\sigma)$
 (4)  reset $timer_p(q)$ to $Timeout_p$, for each process $q \neq p$
 (5)  start tasks 1 and 2
Task 1:
 (6)  loop forever
 (7)      send $(ALIVE, p)$ to every process except $p$ every $\eta$ time
Task 2:
 (8)  upon reception of message $(ALIVE, q)$ do
 (9)      if [message $(ALIVE, q)$ was not previously received] then
 (10)         reset $timer_p(q)$ to $Timeout_p$
 (11)         resend message $(ALIVE, q)$ to every process except $p$
 (12) upon expiration of $timer_p(q)$ do
 (13)     $suspected_p \leftarrow suspected_p \cup \{q\}$
 (14)     $leader_p \leftarrow \min(\Pi \setminus suspected_p)$
```

Fig. 2. Algorithm to implement perpetual failure detectors in systems of class Ψ. The code is for process p.

The proof of the following theorem has been moved to the appendix due to space limitations.

Theorem 3. *Let $S \in \Psi$ be a system in which the Algorithm Perpetual (Figure 2) is executed. Then,*
(i) if S satisfies the Weak Property, Algorithm Perpetual implements a failure

detector of class $\mathcal{S}'$,
(ii) if, additionally, S satisfies the Min Property, Algorithm Perpetual implements a failure detector of class Ω, and
(ii) if, additionally, S satisfies the Strong Property, Algorithm Perpetual implements a failure detector of class $\mathcal{P}_4$.

5 Algorithm for eventual failure detectors

In this section we present an algorithm that implements failure detectors for systems in class $\mathcal{E}$. We show that, for all systems in $\mathcal{E}$ that satisfy the Weak Property the algorithm implements $\Diamond\mathcal{S}$, if they satisfy the Min Property it implements Ω, and if they satisfy the Strong Property it implements $\Diamond\mathcal{P}$. Figure 3 presents the algorithm in detail.

```
Algorithm Eventual
init:
 (1)  $suspected_p \leftarrow \emptyset$
 (2)  $leader_p \leftarrow \min(\Pi)$
 (3)  $Timeout_p(q) \leftarrow \eta + 1$, for each process $q \neq p$
 (4)  reset $timer_p(q)$ to $Timeout_p(q)$, for each process $q \neq p$
 (5)  start tasks 1 and 2
Task 1:
 (6)  loop forever
 (7)      send $(ALIVE, p)$ to every process except $p$ every $\eta$ time
Task 2:
 (8)  upon reception of message $(ALIVE, q)$ do
 (9)      if [message $(ALIVE, q)$ was not previously received] then
 (10)         $suspected_p \leftarrow suspected_p \setminus \{q\}$
 (11)         $leader_p \leftarrow \min(\Pi \setminus suspected_p)$
 (12)         reset $timer_p(q)$ to $Timeout_p(q)$
 (13)         resend message $(ALIVE, q)$ to every process except $p$
 (14) upon expiration of $timer_p(q)$ do
 (15)     $Timeout_p(q) \leftarrow Timeout_p(q) + 1$
 (16)     $suspected_p \leftarrow suspected_p \cup \{q\}$
 (17)     $leader_p \leftarrow \min(\Pi \setminus suspected_p)$
```

Fig. 3. Algorithm to implement eventual failure detectors in systems of class $\mathcal{E}$. The code is for process p.

We have moved the proof of the following theorem to the appendix due to space limitation.

Theorem 4. *Let $S \in \mathcal{E}$ be a system in which the Algorithm Eventual (Figure 3) is executed. Then,*

(i) if S satisfies the Weak Property, Algorithm Eventual implements a failure detector of class $\diamond\mathcal{S}$,
(ii) if, additionally, S satisfies the Min Property, Algorithm Eventual implements a failure detector of class Ω, and
(ii) if, additionally, S satisfies the Strong Property, Algorithm Eventual implements a failure detector of class $\diamond\mathcal{P}$.

6 The Failure Detector Class $\mathcal{S}'$

In this section we explore the new class $\mathcal{S}'$ of unreliable failure detectors proposed in this paper. We first show that this class is strictly weaker than the class $\mathcal{S}$. To do so, we show that while any detector in $\mathcal{S}$ can be used to solve uniform consensus, the same is not so for any detector in $\mathcal{S}'$. Then, we show that any detector in $\mathcal{S}'$ can be used to solve non-uniform consensus in an asynchronous system with any number of failures. We show this by showing how to transform a failure detector in $\mathcal{S}'$ into a failure detector of class (Ω, Σ^{ν}), which can be used to solve nonuniform consensus [19].

Theorem 5. *It is not enough to have a failure detector of class $\mathcal{S}'$ to solve uniform consensus in a crash-prone asynchronous system with single crashes.*

Proof. (Sketch) By way of contradiction, let $\mathcal{A}$ be an algorithm that solves uniform consensus with any detector in $\mathcal{S}'$. Consider a system with two processes p and q and a detector $D \in \mathcal{S}'$. Let R_0 be a run of $\mathcal{A}$ in which both processes propose 0 and p fails before sending any message. D always returns $\{p\}$ to q as the set of suspected processes. Then, q must decide 0 at some time t_0. Similarly, let run R_1 be a run of $\mathcal{A}$ in which both processes propose 1 and q fails before sending any message; D always returns $\{q\}$ to p as the list of suspected processes; and p decides 1 at time t_1. Finally, consider a run R of $\mathcal{A}$ in which all messages sent are delayed until a time $t > \max(t_0, t_1)$, and p fails at this time. D always returns $\{p\}$ to q and $\{q\}$ to p as their respective lists of suspected processes. Until time t process p behaves exactly like in R_1, and hence decides 1 at time t_1. Similarly, process q behaves exactly like in R_0 until time t, and hence decides 0 at time t_0. Since they decide different values, $\mathcal{A}$ does not solve uniform consensus.

Now we show that any failure detector $D \in \mathcal{S}'$ can be transformed into a failure detector of class (Ω, Σ^{ν}). A failure detector of class Ω can be obtained from D by using, for instance, the algorithm proposed by Chu [20] for $\diamond\mathcal{W}$[7]. A detector of class Σ^{ν} is trivially obtained from D by returning as quorum at each process the complement of the list of suspected processes returned by D. Clearly, all the quorums returned at the correct processes contain at least the correct process that is never suspected by them. Eventually these quorums only contain correct processes by the strong completeness of $\mathcal{S}'$.

[7] Any detector in $\mathcal{S}'$ is trivially in $\diamond\mathcal{W}$.

7 Conclusions

In this paper we explore the minimal system synchrony to implement unreliable failure detectors. We present algorithms that implement detectors in systems with weak synchrony requirements and show that these requirements are in fact needed.

There are still a number of open problems related with this work. For instance, we present algorithms that implement detectors of classes $\mathcal{P}_4$ and $\mathcal{S}'$, which are weaker than $\mathcal{P}$ and $\mathcal{S}$, in systems with limited synchrony. It would be nice to have algorithms that implement $\mathcal{P}$ and $\mathcal{S}$ in systems with the same synchrony.

Finally, observe that our algorithms have a quadratic number of links carrying messages forever in the worse case. We believe this is the best we can hope for $\diamond\mathcal{S}$ (since there are similar bounds for Ω in a system with one correct process whose output links are eventually timely [5]). However, we would like to have a proof of that.

References

1. Chandra, T.D., Toueg, S.: Unreliable failure detectors for reliable distributed systems. Journal of the ACM **43** (1996) 225–267
2. Fischer, M., Lynch, N., Paterson, M.: Impossibility of distributed consensus with one faulty process. Journal of the ACM **32** (1985) 374–382
3. Bertier, M., Marin, O., Sens, P.: Implementation and performance evaluation of an adaptable failure detector. In: Proceedings of the 2002 International Conference on Dependable Systems and Networks. (2002)
4. Aguilera, M., Delporte-Gallet, C., Fauconnier, H., Toueg, S.: Communication-efficient leader election and consensus with limited link synchrony. In: Proceedings of the 23rd Annual Symposium on Principles of Distributed Computing (PODC'2004). (2004)
5. Aguilera, M., Delporte-Gallet, C., Fauconnier, H., Toueg, S.: On implementing Ω with weak reliability and synchrony assumptions. In: Proceedings of the 22nd Annual Symposium on Principles of Distributed Computing (PODC'2003), Boston, Massachusetts (2003)
6. Aguilera, M., Delporte-Gallet, C., Fauconnier, H., Toueg, S.: Stable leader election. In: Proceedings of the 15th International Symposium on DIstributed Computing (DISC'2001), Lisbon, Portugal, LNCS 2180, Springer-Verlag (2001) 108–122
7. Chen, W., Toueg, S., Aguilera, M.K.: On the quality of service of failure detectors. IEEE Transactions on Computers **51** (2002) 13–32
8. Larrea, M., Fernández, A., Arévalo, S.: On the implementation of unreliable failure detectors in partially synchronous systems. IEEE Transactions on Computers **53** (2004) 815–828
9. Larrea, M., Fernández, A., Arévalo, S.: Optimal implementation of the weakest failure detector for solving consensus. In: Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems (SRDS'2000), Nurenberg, Germany (2000) 52–59
10. Larrea, M., Fernández, A., Arévalo, S.: Eventually consistent failure detectors. Journal of Parallel and Distributed Computing **65** (2005) 361–373

11. Keidar, I., Rajsbaum, S.: On the cost of fault-tolerant consensus when there are no faults. In: Tutorial on the 21st Annual Symposium on Principles of Distributed Computing (PODC'2002), Monterey, California (2002)
12. Chandra, T.D., Hadzilacos, V., Toueg, S.: The weakest failure detector for solving consensus. Journal of the ACM **43** (1996) 685–722
13. Jiménez, E., Arévalo, S., Fernández, A.: Implementing unreliable failure detectors with unknown membership. Information Processing Letters **100** (2006) 60–63
14. Fernández, A., Jiménez, E., Raynal, M.: Eventual leader election with weak assumptions on initial knowledge, communication reliability, and synchrony. In: Proceedings of the International Conference on Dependable Systems and Networks (DSN-2006), Philadelphia, PA, USA (2006)
15. Mostéfaoui, A., Mourgaya, E., Raynal, M.: Asynchronous implementation of failure detectors. In: 2003 International Conference on Dependable Systems and Networks (DSN 2003). (2003) 351–360
16. Mostéfaoui, A., Powell, D., Raynal, M.: A hybrid approach for building eventually accurate failure detectors. In: 10th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2004). (2004) 57–65
17. Larrea, M.: Brief announcement: Understanding perfect failure detectors. In: Proceedings of the 21st Annual ACM Symposium on Principles of Distributed Computing, PODC 2002. (2002) 257
18. Défago, X., Schiper, A., Urbán, P.: Total order broadcast and multicast algorithms: Taxonomy and survey. ACM Comput. Surv. **36** (2004) 372–421
19. Eisler, J., Hadzilacos, V., Toueg, S.: The weakest failure detector to solve nonuniform consensus. In: Proceedings of the 24rd Annual Symposium on Principles of Distributed Computing (PODC'2005), Las Vegas, NV (2005) 189–196
20. Chu, F.: Reducing Ω to $\diamond\mathcal{W}$. Information Processing Letters **67** (1998) 289–293

A Proof of Theorem 3

Let us assume we have a system $S \in \Psi$ and run the algorithm in this system. We start by proving the strong completeness property. For simplicity we do not differentiate between a message and its replicas, and consider them the same message. We first need to show that messages from faulty processes eventually disappear from the system.

Lemma 1. *Any message $(ALIVE, p)$ sent by a process $p \in \Pi$ eventually disappears from the system*

Proof. First, we note that a message cannot remain forever in a link, since it remains at most Δ time in a timely link and is either lost or eventually delivered in a lossy asynchronous link. Similarly, a message cannot remain forever in a process, since once a message is received either the process crashes or the message is processed in Lines 8-11. In this latter case this processing ends in at most 4σ time units with the message either dropped or resent. Finally, the message cannot loop forever since a process never resends the same message twice.

Then, we can show the strong completeness of the system in the next lemma.

Lemma 2. *Every process $p \in$ correct will eventually and permanently suspect every process in crashed.*

Proof. Let us consider we are at a time t at which all processes in *crashed* have in fact crashed and all their packets have disappeared from the system (which eventually happens from the above lemma). Then a process $p \in$ *correct* will have every process $q \in$ *crashed* in its list $suspected_p$ by time $t + \tau_p(\eta + (n-1)(\Delta + 4\sigma)) + 2\sigma$, since by then $timer_p(q)$ will have expired and the Lines 12-13 would have included q in $suspected_p$. Note that once q is inserted in $suspected_p$ it is never removed.

The following lemmas are used to prove the accuracy of the system.

Lemma 3. *The (first copy of a) message $(ALIVE, p)$ sent by a correct process p at time t is received at process $q \in reach(p)$, $q \neq p$, by time $t+\Delta(n-1)+4\sigma(n-2)$.*

Proof. Let us consider a shortest path from p to q in $G(S)$. Such a path must exist by definition of $reach(p)$, and by definition of $G(S)$ contains only correct processes such that two consecutive processes are connected by a directed timely link. We prove that a process at distance i from p in this path receives the (first copy of the) message by time $t + \Delta i + 4\sigma(i-1)$. This will show the claim since q is at most at distance $n-1$ from p in this path.

Let us consider first process r at distance 1 from p. Since the link from p to r is timely, the message will be received at r by time $t + \Delta$. Now if process u is at distance $i > 1$ in the path, by induction hypothesis the process s at distance $i-1$ will receive the (first copy of the) message by time $t + \Delta(i-1) + 4\sigma(i-2)$. Then, it will process it in Lines 8-11, which takes at most 4σ time units and will resend the message to u (among others). Since the link from s to u is timely, u will receive this copy of the message at most Δ time later. (Observe that this may not be the first copy received by u.) Then, the message is received at u by time $t + \Delta i + 4\sigma(i-1)$.

Lemma 4. *A process $q \in reach(p)$, $q \neq p$, permanently receives new $(ALIVE, p)$ messages sent by a correct process p within intervals of at most $\eta + \Delta(n-1) + 4\sigma(n-2)$ time. Hence, q never suspects p.*

Proof. The first part follows for the previous lemma and the fact that p sends a new $(ALIVE, p)$ message every η time. Then since processing a new message and resetting the timer in Lines 8-10 takes at most 3σ time units, the timer $timer_q(p)$ will be reset at least once in each interval of $\eta + (n-1)(\Delta + 4\sigma) - \sigma$ time. Since the timer is always reset to $\eta + (n-1)(\Delta + 4\sigma)$, and for all T $\tau_q(T) \geq T$, it will never expire and q never adds p to its set $suspected_q$ of suspected processes.

Proof (Theorem 3). (i) Lemma 2 shows that the algorithm enforces Strong Completeness. Then, by the Weak Property, there is some correct process p such that $reach(p) = correct$. Then, from Lemma 4, no correct process $q \neq p$ ever suspects p. Since p never suspects itself, the algorithm guarantees Quasi-Weak Accuracy.

(ii) From Lemma 4 and the Min Property, $\min(correct)$ will never be suspected by a correct process. From Lemma 2 we have that all correct processes will eventually include every process $q < \min(correct)$ in their list of suspected processes and choose as leader the not suspected process with smallest identifier (Line 14). Hence eventually Property 1 will be satisfied with $\min(correct)$ as leader.

(iii) Lemma 2 shows that the algorithm enforces Strong Completeness. Then, by the Strong Property every correct process p has that $reach(p) = correct$. From Lemma 4, no correct process $q \neq p$ ever suspects p, and since p never suspects itself, p in never suspected by any correct process. This is true for all $p \in correct$, and hence the algorithm guarantees Quasi-Strong Accuracy.

B Proof of Theorem 4

Let S be a system of class $\mathcal{E}$. Again we start the proof of correctness for the algorithm by showing that it guarantees Strong Completeness. As before we first show that any message eventually disappears from the system.

Lemma 5. *Any message $(ALIVE, p)$ sent by a process $p \in \Pi$ eventually disappears from the system.*

Proof. First, we note that a message cannot remain forever in a link. If the link is eventually timely and the message was sent up to time GST it is either lost or eventually delivered. If the link is eventually timely and the message is sent after GST it remains in the link at most Δ time. Finally, if the link is lossy asynchronous, the message is either lost or eventually delivered. Similarly, a message cannot remain forever in a process, since once a message is received either the process crashes or the message is processed in Lines 8-13. In this latter case this processing ends in at most 6σ time units with the message either dropped or resent. Finally, the message cannot loop forever since a process never resends the same message twice.

Then, we can show the strong completeness of the system in the next lemma.

Lemma 6. *Every process $p \in$ correct will eventually and permanently suspect every process in crashed.*

Proof. Let us consider we are at a time t at which all processes in *crashed* have in fact crashed and all their packets have disappeared from the system (which eventually happens from the above lemma). Consider a process $p \in \mathit{correct}$ and a process $q \in \mathit{crashed}$, and assume that $\mathit{Timeout}_p(q) = T$ at time t (note that $\mathit{Timeout}_p(q)$ never decreases). Then, p will include q in its list $\mathit{suspected}_p$ by time $t + \tau_p(T) + 4\sigma$, since by then the timer $\mathit{timer}_p(q)$ will have expired (it was reset for the last time before time t with a value of at most T, and $\tau_p(T)$ is non-decreasing with T) and the Lines 14-17 will have been executed. Since no new message from q will ever be received by p, it will never be removed from $\mathit{suspected}_p$ at Line 10.

We show now the accuracy of the detector implemented by the algorithm. For simplicity we consider all described events to occur after time GST.

Lemma 7. *The (first copy of a) message $(ALIVE, p)$ sent by a correct process p at time $t > GST$ is received at process $q \in \mathit{reach}(p)$, $q \neq p$, by time $t + \Delta(n-1) + 6\sigma(n-2)$.*

Proof. The proof is similar to that of Lemma 3.

Lemma 8. *After GST, process $q \in \mathit{reach}(p)$, $q \neq p$, permanently receives new $(ALIVE, p)$ messages sent by a correct process p within intervals of at most $\eta + \Delta(n-1) + 6\sigma(n-2)$ time. Hence, there is a time after which q never suspects p.*

Proof. The first claim follows for the previous lemma and the fact that p sends a new $(ALIVE, p)$ message every η time. We show now that p is added to $\mathit{suspected}_q$ a finite number of times. Then the second claim follows, since from the first claim, Line 10 will be executed periodically and p will be removed from $\mathit{suspected}_q$ if it was there.

Let us assume by way of contradiction that p is added an infinite number of times to $\mathit{suspected}_q$ (in Line 16). Then, $\mathit{Timeout}_q(p)$ grows to infinity, since it is increased every time this happens (in Line 15). However, once time GST has passed and $\tau_q(\mathit{Timeout}_q(p)) > \eta + \Delta(n-1) + 6\sigma(n-2) + 5\sigma$, from the first claim Lines 8-12 are executed (and hence $\mathit{timer}_q(p)$ reset) always before $\mathit{timer}_q(p)$ expires, and then p is never again inserted in $\mathit{suspected}_q$.

We have now the proof of the theorem.

Proof (Theorem 4). (i) Lemma 6 shows that the algorithm enforces Strong Completeness. Then, by the Weak Property, there is some correct process p such that $\mathit{reach}(p) = \mathit{correct}$. Then, from Lemma 8, every correct process $q \neq p$ eventually and permanently stops suspecting p. Since p never suspects itself, the algorithm guarantees Eventual Weak Accuracy.

(ii) From Lemma 6 we have that all correct processes will eventually include every process $q < \min(\mathit{correct})$ in their list of suspected processes. From Lemma 8 and the Min Property, there is a time after which $\min(\mathit{correct})$ is never suspected by a correct process. Each time the set of suspected processes is modified, the leader is updated with the process not suspected with smallest identifier (Lines 11 and 17). Hence eventually Property 1 will be satisfied with $\min(\mathit{correct})$ as leader.

(iii) Lemma 6 shows that the algorithm enforces Strong Completeness. Then, by the Strong Property every correct process p has that $\mathit{reach}(p) = \mathit{correct}$. From Lemma 8, there is a time after which no correct process $q \neq p$ suspects p, and since p never suspects itself, there is a time after which p in never suspected by any correct process. This is true for all $p \in \mathit{correct}$, and hence the algorithm guarantees Eventual Strong Accuracy.