A Sufficient Condition to Transform $\diamond\mathcal{S}$ into $\diamond\mathcal{P}$ in Asynchronous Systems*

Antonio Fernández
Universidad Rey Juan Carlos
28933 Móstoles, Spain
afernandez@acm.org

Mikel Larrea
Universidad del País Vasco
20018 San Sebastián, Spain
mikel.larrea@si.ehu.es

Sergio Arévalo
Universidad Rey Juan Carlos
28933 Móstoles, Spain
s.arevalo@escet.urjc.es

Abstract

Chandra and Toueg stated in [J. ACM 43 (2)(1996)] that $\diamond\mathcal{P}$ is strictly stronger than $\diamond\mathcal{S}$. This implies that no algorithm can transform *every* failure detector of class $\diamond\mathcal{S}$ into a failure detector of class $\diamond\mathcal{P}$ in an asynchronous system. In this paper, we present a subclass of $\diamond\mathcal{S}$, named *Ordered*-$\diamond\mathcal{S}$, whose failure detectors are in $\diamond\mathcal{S}$ and satisfy an additional property. We show how to use n (where n is the number of processes in the system) *Ordered*-$\diamond\mathcal{S}$ failure detectors to obtain a failure detector of class $\diamond\mathcal{P}$ in an asynchronous system. Interesting enough, several implementations of $\diamond\mathcal{S}$ proposed in the literature in fact implement *Ordered*-$\diamond\mathcal{S}$ detectors.

1 Introduction

The concept of *unreliable failure detector* was introduced by Chandra and Toueg [1] as a mechanism that provides (possibly incorrect) information about process failures. This mechanism has been used to solve several problems in asynchronous distributed systems, in particular the Consensus problem.

*Research partially supported by the Spanish Research Council (CICYT), under grant TIC2001-1586-C03-01.

In [1], failure detectors were characterized in terms of two properties: *completeness* and *accuracy*. Completeness characterizes the failure detector capability of suspecting every incorrect process (processes that actually crash), while accuracy characterizes the failure detector capability of not suspecting correct processes. In this paper, we focus on the following completeness and accuracy properties, from those defined in [1]:

- *Strong Completeness.* Eventually every process that crashes is permanently suspected by *every* correct process.
- *Eventual Strong Accuracy.* There is a time after which correct processes are not suspected by any correct process.
- *Eventual Weak Accuracy.* There is a time after which some correct process is never suspected by any correct process.

Combining strong completeness with each one of the two accuracy properties, we obtain two different failure detector classes:

- The class of *Eventually Perfect* failure detectors, $\diamond\mathcal{P}$, satisfying strong completeness and eventual strong accuracy.
- The class of *Eventually Strong* failure detectors, $\diamond\mathcal{S}$, satisfying strong completeness and eventual weak accuracy.

Chandra and Toueg stated in [1] that $\diamond\mathcal{P}$ is strictly stronger than $\diamond\mathcal{S}$. Since by definition $\diamond\mathcal{P}$ is a subclass of $\diamond\mathcal{S}$ (i.e., any algorithm implementing $\diamond\mathcal{P}$ implements also $\diamond\mathcal{S}$), this means that no algorithm can transform *every* failure detector of class $\diamond\mathcal{S}$ into a failure detector of class $\diamond\mathcal{P}$ in an asynchronous system.

In this paper, we present a property that can be satisfied by failure detectors of class $\diamond\mathcal{S}$. This property is related to the ability of determining both the identity and the order of the correct process that, by means of the eventual weak accuracy property of $\diamond\mathcal{S}$, is eventually *trusted* (i.e., not suspected by any correct processes). We call *Ordered*-$\diamond\mathcal{S}$ the class of $\diamond\mathcal{S}$ failure detectors that satisfy this property. Interesting enough, the property is satisfied by several implementations of $\diamond\mathcal{S}$ detectors proposed in the literature.

Then, we show how to use n *Ordered*-$\diamond\mathcal{S}$ failure detectors to obtain a failure detector of class $\diamond\mathcal{P}$ in an asynchronous system. This result is interesting since Fínez [2] has recently shown that there is no algorithm that transforms every *Ordered*-$\diamond\mathcal{S}$ detector into a detector of class $\diamond\mathcal{P}$ in

an asynchronous system. This means that the number of available *Ordered-*$\diamond\mathcal{S}$ detectors really matters in order to obtain a $\diamond\mathcal{P}$ detector.

The rest of the paper is organized as follows. In Section 2, we establish the model of the system we use in the paper. In Section 3, we present the failure detector class *Ordered-*$\diamond\mathcal{S}$. Finally, in Section 4, we propose an algorithm that transforms n failure detectors of class *Ordered-*$\diamond\mathcal{S}$ into a failure detector of class $\diamond\mathcal{P}$ in an asynchronous system.

2 System Model

We consider a distributed system consisting of a finite totally ordered set Π of n processes, $\Pi = \{p_1, p_2, \ldots, p_n\}$. The system is *asynchronous*, i.e., there are no timing assumptions about either the relative speeds of the processes or the delay of possible messages they could exchange. Processes can fail by *crashing*, that is, by prematurely halting. Crashes are permanent, i.e., crashed processes do not recover.

A *distributed failure detector* can be viewed as a set of n failure detection modules, each one attached to a different process in the system. These modules cooperate to satisfy the required properties of the failure detector. Upon request, each module provides its attached process with a set of processes it suspects to have crashed. These sets can differ from one module to another at a given time. Let us denote by $\mathcal{D}_i$ the set of suspected processes returned by a failure detector $\mathcal{D}$ to a given process p_i. We also denote by $\mathcal{T}_i$ the set of trusted (non-suspected) processes of the failure detection module attached to process p_i, i.e., $\mathcal{T}_i = \Pi - \mathcal{D}_i$. We assume that a process interacts only with its local failure detection module in order to get the current set of suspected processes.

3 The Failure Detector Class *Ordered-*$\diamond\mathcal{S}$

In this section, we present the failure detector class *Ordered-*$\diamond\mathcal{S}$. Since this is a subclass of $\diamond\mathcal{S}$, the detectors in this class must satisfy the strong completeness and eventual weak accuracy properties like any failure detector in $\diamond\mathcal{S}$. But, on top of that, they must also satisfy that all the correct processes eventually converge to the same trusted correct process (by means of a deterministic *leader* function applied to the set $\mathcal{T}_i$ of processes trusted by the failure detector). Moreover, this trusted correct process must be the first correct process in some order defined on the processes by the user, and that is provided to the failure detector.

There are several algorithms in the literature that implement $\diamond\mathcal{S}$ failure detectors that are in *Ordered*-$\diamond\mathcal{S}$. In the ring-based algorithm implementing $\diamond\mathcal{S}$ proposed in [3], the set of non-suspected processes can be different in different processes, but the algorithm guarantees that, eventually, the first (starting from the *initial candidate to leader* and following the order defined by the ring) correct process is not suspected by any correct process. From this, it is easy to derive a deterministic function *leader* that returns the first non-suspected process. Hence, the ring-based algorithm implementing $\diamond\mathcal{S}$ of [3] implements also a failure detector of class *Ordered*-$\diamond\mathcal{S}$.

In [4], a more efficient algorithm implementing a failure detector of class $\diamond\mathcal{S}$ is proposed. In this algorithm, the set of non-suspected processes contains only one process, that will eventually and permanently be the same correct process for all correct processes. Moreover, the algorithm guarantees that this *leader* process is the first correct process. This trivially gives a possible *leader* function. Again, this algorithm implements also a failure detector of class *Ordered*-$\diamond\mathcal{S}$.

4 Transforming *Ordered*-$\diamond\mathcal{S}$ into $\diamond\mathcal{P}$ in an Asynchronous System

In this section, we present an algorithm that uses n *Ordered*-$\diamond\mathcal{S}$ failure detectors to obtain a failure detector of class $\diamond\mathcal{P}$ in an asynchronous system. The approach followed is to execute concurrently the n *Ordered*-$\diamond\mathcal{S}$ failure detectors, each with the processes ordered differently, and combine their outputs to build a list of suspected processes that satisfies the properties of $\diamond\mathcal{P}$.

Figure 1 presents the algorithm in detail, which works as follows. We start concurrently the n *Ordered*-$\diamond\mathcal{S}$ failure detectors $\mathcal{D}^{(1)}$ to $\mathcal{D}^{(n)}$, each one using the order defined among processes in a cyclic fashion and considering as first process a different process (p_1 for $\mathcal{D}^{(1)}$, p_2 for $\mathcal{D}^{(2)}$, and so on). This way, the $leader^{(j)}$ function associated to detector $\mathcal{D}^{(j)}$, for $j = 1, ..., n$, will eventually return forever:

- The first correct process in $p_1, p_2, \ldots, p_n$ in the case of $\mathcal{D}^{(1)}$ $(j = 1)$.
- The first correct process in $p_2, p_3, \ldots, p_n, p_1$ in the case of $\mathcal{D}^{(2)}$ $(j = 2)$.
- ...
- The first correct process in $p_n, p_1, \ldots, p_{n-1}$ in the case of $\mathcal{D}^{(n)}$ $(j = n)$.

Every process p_i executes the following:

$suspected_i \leftarrow \emptyset$ $\quad$ $\{suspected_p$ *provides the properties of* $\diamond\mathcal{P}\}$

cobegin

|| *Task* j, $j = 1, ..., n$:
$\quad \mathcal{D}^{(j)}(p_j, ..., p_n, p_1, ..., p_{j-1})$
|| *Task* $n+1$: **repeat periodically**
$\quad suspected_i \leftarrow \Pi - \{leader^{(1)}(\mathcal{T}_i^{(1)}), leader^{(2)}(\mathcal{T}_i^{(2)}), \ldots, leader^{(n)}(\mathcal{T}_i^{(n)})\}$

coend

Figure 1: Transforming n *Ordered-*$\diamond\mathcal{S}$ detectors into $\diamond\mathcal{P}$ in an asynchronous system.

With this algorithm, the set of trusted processes in the system is the union of the outputs of all the *leader* functions. Clearly, the set of suspected processes is just the complementary set. It is simple to see that this set satisfies the properties of $\diamond\mathcal{P}$. Interesting enough, note that the construction of the set of suspected processes does not require any communication (besides that of the failure detectors).

Theorem 1 *Given* n Ordered-$\diamond\mathcal{S}$ *failure detectors,* $\mathcal{D}^{(1)}, \mathcal{D}^{(2)}, ..., \mathcal{D}^{(n)}$*, the algorithm of Figure 1 implements a failure detector of class* $\diamond\mathcal{P}$ *in an asynchronous system.*

Proof: We only give a sketch of the proof here. The proof relies on the property satisfied by the *Ordered-*$\diamond\mathcal{S}$ failure detectors: given a predefined arrangement of processes, there is a time after which all the correct processes permanently *trust*, by means of the *leader* function, the *first* correct process following that arrangement.

(1) By Task 1, eventually $leader^{(1)}(\mathcal{T}_i^{(1)})$ will return forever the first correct process in $p_1, p_2, \ldots, p_n$. By Task 2, eventually $leader^{(2)}(\mathcal{T}_i^{(2)})$ will return forever the first correct process in $p_2, p_3, \ldots, p_n, p_1$. In general, by Task j, $j = 1, ..., n$, eventually $leader^{(j)}(\mathcal{T}_i^{(j)})$ will return forever the first correct process in $p_j, ..., p_n, p_1, \ldots, p_{j-1}$.

(2) Let p_k be any correct process, with $k \in \{1, \ldots, n\}$. Clearly, p_k is the first correct process in $p_k, p_{k+1}, \ldots, p_n, p_1, p_2, \ldots, p_{k-1}$. Thus, by (1) eventually $leader^{(k)}(\mathcal{T}_i^{(k)})$ will return forever p_k.

(3) By (1) and (2), eventually $\{leader^{(1)}(\mathcal{T}_i^{(1)}), leader^{(2)}(\mathcal{T}_i^{(2)}), \ldots, leader^{(n)}(\mathcal{T}_i^{(n)})\}$ will be the set of correct processes in the system. Hence, the complementary set $\Pi - \{leader^{(1)}(\mathcal{T}_i^{(1)}), leader^{(2)}(\mathcal{T}_i^{(2)}), \ldots, leader^{(n)}(\mathcal{T}_i^{(n)})\}$ will be the set of incorrect processes in the system. Clearly, the set of suspected processes built in Task $n+1$ satisfies the properties of $\diamond\mathcal{P}$: eventually every process that crashes is permanently suspected by every correct process (strong completeness), and there is a time after which correct processes are not suspected by any correct process (eventual strong accuracy).

■

Acknowledgments

We would like to thank Michel Raynal for asking the right questions.

References

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

[2] Germán Fínez. Why strength is weaker than perfection: On the proof that $\diamond\mathcal{S} < \diamond\mathcal{P}$. 2002.

[3] M. Larrea, S. Arévalo, and A. Fernández. Efficient algorithms to implement unreliable failure detectors in partially synchronous systems. In *Proceedings of the 13th International Symposium on DIstributed Computing (DISC'99)*, pages 34–48. LNCS, Springer-Verlag, September 1999.

[4] M. Larrea, A. Fernández, and S. Arévalo. Optimal implementation of the weakest failure detector for solving consensus. In *Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems (SRDS'2000)*, pages 52–59, Nurenberg, Germany, October 2000.