Journal of Computer and System Sciences 81 (2015) 383–397

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Communication-optimal eventually perfect failure detection in partially synchronous systems ☆

Alberto Lafuente *, Mikel Larrea, Iratxe Soraluze, Roberto Cortiñas

University of the Basque Country UPV/EHU, Faculty of Computer Science, Lardizabal 1, 20018 San Sebastián, Spain

ARTICLE INFO

Article history:
Received 11 March 2009
Received in revised form 18 July 2013
Accepted 19 May 2014
Available online 2 July 2014

Keywords:
Distributed algorithms
Fault tolerance
Consensus
Partial synchrony
Unreliable failure detectors
Communication optimality

ABSTRACT

Since Chandra and Toueg introduced the failure detector abstraction for crash-prone systems, several algorithms implementing failure detectors in partially synchronous systems have been proposed. Their performance can be measured by their *Communication efficiency*, defined as the number of links used forever. In this regard, in a *communication-efficient* algorithm only n links are used forever, n being the number of processes in the system. In this paper, we present *communication optimality*, a communication efficiency degree reached when only c links are used forever, c being the number of correct processes. We show that c is the minimum number of links used forever required to implement $\diamond\mathcal{P}$ and that c is also optimal for $\diamond\mathcal{S}$ and Ω when $c < n$. Finally, we propose two communication-optimal $\diamond\mathcal{P}$ algorithms following respectively one-to-all and one-to-one communication patterns to manage suspicions, showing that there is a trade-off between detection latency and sporadic communication overhead.

1. Introduction

1.1. Background and related work

Unreliable failure detectors, proposed by Chandra and Toueg in [1], are a mechanism providing (possibly incorrect) information about process failures. This mechanism has been used to solve several problems in asynchronous distributed systems where processes may crash by prematurely halting, in particular the *consensus* problem [2]. The specific classes of failure detectors proposed in [1] are defined by a *completeness* property, which characterizes the failure detector's capability of suspecting incorrect processes, and by an *accuracy* property, which restricts the mistakes the failure detector can make. More specifically, Chandra and Toueg defined, among others, the following two completeness properties and two accuracy properties that a failure detector may satisfy:

- *Strong Completeness:* eventually every process that crashes is permanently suspected by *every* correct process.
- *Weak Completeness:* eventually every process that crashes is permanently suspected by *some* 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.

☆ Research partially supported by the Spanish Research Council, under grant TIN2013-41123-P, the Basque Government, under grants IT395-10 and S-PE12UN109, and the University of the Basque Country UPV/EHU, under grant UFI11/45.

* Corresponding author. Fax: +34 943015590.
E-mail addresses: alberto.lafuente@ehu.es (A. Lafuente), mikel.larrea@ehu.es (M. Larrea), iratxe.soraluze@ehu.es (I. Soraluze), roberto.cortinas@ehu.es (R. Cortiñas).

http://dx.doi.org/10.1016/j.jcss.2014.06.010
0022-0000/

Table 1
Four classes of failure detectors.

	Eventual strong accuracy	Eventual weak accuracy
Strong completeness	Eventually perfect ($\diamond\mathcal{P}$)	Eventually strong ($\diamond\mathcal{S}$)
Weak completeness	Eventually quasi-perfect ($\diamond\mathcal{Q}$)	Eventually weak ($\diamond\mathcal{W}$)

The combination of these properties gives us four classes of failure detectors, which are shown in Table 1. Consensus can be solved using a failure detector of any of those four classes. In particular, there is another failure detector class denoted by Ω, which is equivalent[1] to $\diamond\mathcal{S}$ and $\diamond\mathcal{W}$, and which has been proved to be sufficient and necessary, i.e., weakest, for solving consensus [3]. The Ω failure detector class provides eventual agreement on a common and correct *leader* among all non-faulty processes in a system.

Specific algorithms for implementing Ω and/or $\diamond\mathcal{S}$ have been proposed, e.g. [4–11]. Observe that $\diamond\mathcal{P}$ trivially satisfies the properties of $\diamond\mathcal{S}$. Also, $\diamond\mathcal{P}$ can be easily transformed into Ω, e.g., by choosing as leader the non-suspected process with the lowest identifier.

Failure detectors of the class $\diamond\mathcal{P}$, being strictly stronger than Ω and $\diamond\mathcal{S}$,[2] can also be used to solve consensus. Several algorithms implementing $\diamond\mathcal{P}$ have been proposed in the literature. The algorithm proposed by Chandra and Toueg in [1] uses a heartbeat mechanism and all-to-all communication to detect faulty processes. The algorithms proposed by Aguilera et al. in [12] and by Larrea et al. in [13] use heartbeats too, and rely on a leader-based approach. On the other hand, the algorithms proposed by Larrea et al. in [14,15] use a polling—or query/reply—mechanism on a ring arrangement of processes. Roughly speaking, the leader-based and the ring-based algorithms are more efficient than the all-to-all algorithm regarding the number of sent messages (linear vs. quadratic). Observe also that, compared to polling, the heartbeat mechanism reduces the number of messages to the half. Moreover, heartbeats inherently provide a certain level of communication reliability in a system with fair lossy links, while polling usually requires reliable communication.

Chandra and Toueg showed in [1] that $\diamond\mathcal{Q}$ can also be used to solve consensus. To do so, they first showed that classes $\diamond\mathcal{Q}$ and $\diamond\mathcal{P}$ are equivalent from a problem solvability point of view, i.e., a problem which is solvable with $\diamond\mathcal{P}$ is also solvable with $\diamond\mathcal{Q}$ and vice versa. It is worth noting that the equivalence of $\diamond\mathcal{Q}$ and $\diamond\mathcal{P}$ does not come for free, i.e., not all failure detectors in $\diamond\mathcal{Q}$ are in $\diamond\mathcal{P}$. Instead, it means that any failure detector in $\diamond\mathcal{Q}$ can be extended with a simple distributed algorithm to obtain a failure detector in $\diamond\mathcal{P}$.

Since consensus cannot be solved in crash-prone, pure asynchronous systems [2], algorithms that implement failure detectors make some weak timing assumptions. Specifically, a partially synchronous model [1,16] is considered in this work. In such a model, in every run of the system, there are bounds on relative process speeds and on message transmission times, but these bounds are not known and they hold only after some unknown but finite time. In practice, the bounds depend on parameters such as the network speed or the size of the system, and hold easier in, for example, local area networks than in wide area networks. Actually, the bounds must exist and hold only for the links that connect correct processes. Hence, it is important to design failure detection algorithms that use a low number of links, e.g., by arranging the processes in a logical ring topology.

As shown recently in [17], algorithms for the class $\diamond\mathcal{P}$ combining a heartbeat-based detection mechanism on a logical ring arrangement of processes outperform the aforementioned ones in terms of the number of links permanently used, while preserving good quality-of-service. In this regard, algorithms in [17] are communication-efficient following Aguilera et al. [12], i.e., eventually only n unidirectional links are used forever. With regard to this performance measure, heartbeat-based ring algorithms outperform other ring algorithms based on polling [14,15] or algorithms using a centralized communication pattern [12,13]. By all means, algorithms using an all-to-all communication pattern, such as Chandra–Toueg's algorithm [1], are far from being communication-efficient.

Other failure detection algorithms specifically designed for wide area networks, e.g. [18], are based on local failure detectors that provide their properties in a neighborhood of processes (using all-to-all communication inside each neighborhood), and propagate the information about suspicions among neighborhoods. The advantage of a ring based approach is that, once the logical ring is defined, neighborhoods are implicit, and eventually involve just two processes for every correct process, i.e., its correct predecessor and correct successor in the ring.

1.2. Implementing failure detectors

Obviously, an algorithm implementing a failure detector of a given class must satisfy the properties defined for that class. When implementing a failure detector, besides satisfying the properties of the class it belongs to, performance should also be taken into account. In this work, we focus on the following two performance issues:

[1] Two failure detectors are equivalent if they are reducible to each other [1]. Informally, a failure detector D is reducible to a failure detector D' if there is a distributed algorithm that can transform D into D'. The concepts of reducibility and equivalence can be naturally extended to classes of failure detectors.

[2] While $\diamond\mathcal{P}$ can be transformed into Ω or $\diamond\mathcal{S}$ *asynchronously*, i.e, without any additional synchrony assumption, neither Ω nor $\diamond\mathcal{S}$ can be transformed into $\diamond\mathcal{P}$.