Theoretical Computer Science 709 (2018) 80–97

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Randomized k-set agreement in crash-prone and Byzantine asynchronous systems[☆]

Achour Mostéfaoui[a], Hamouma Moumen[b], Michel Raynal[c,d,*]

[a] LINA, Université de Nantes, 44322 Nantes, France
[b] University of Batna, Algeria
[c] Institut Universitaire de France, France
[d] IRISA, Université de Rennes, 35042 Rennes, France

ARTICLE INFO

Article history:
Received 21 April 2016
Received in revised form 17 March 2017
Accepted 23 March 2017
Available online 25 March 2017

Keywords:
Asynchronous system
Broadcast abstraction
Byzantine process
Coin
Crash failure
Distributed algorithm
k-Set agreement
Message-passing system
Randomized algorithm
Signature-free algorithm

ABSTRACT

k-Set agreement is a central problem of fault-tolerant distributed computing. Considering a set of n processes, where up to t may commit failures, let us assume that each process proposes a value. The problem consists in defining an algorithm such that each non-faulty process decides a value, at most k different values are decided, and the decided values satisfy some context-depending validity condition. Algorithms solving k-set agreement in synchronous message-passing systems have been proposed for different failure models (mainly process crashes, and process Byzantine failures). Differently, k-set agreement cannot be solved in failure-prone asynchronous message-passing systems when $t \geq k$. To circumvent this impossibility an asynchronous system must be enriched with additional computational power.

Assuming $t \geq k$, this paper presents two distributed algorithms that solve k-set agreement in asynchronous message-passing systems where up to t processes may commit crash failures (first algorithm) or more severe Byzantine failures (second algorithm). To circumvent k-set agreement impossibility, this article considers that the underlying system is enriched with the computability power provided by randomization. Interestingly, the algorithm that copes with Byzantine failures is signature-free, and ensures that no value proposed only by Byzantine processes can be decided by a non-faulty process. Both algorithms share basic design principles.

1. Introduction

Distributed agreement in the presence of process failures. The world is distributed and more and more applications are now distributed. Moreover, when considering the core of non-trivial distributed applications, it appears that the computing entities (processes) have to agree in one way or another, for example to take a common decision, execute specific actions, or validate some commitment. Said another way, agreement problems lie at the core of distributed computing.

The most famous distributed agreement problem is the *consensus* problem. Let us consider a set of processes, where some of them may commit failures. Assuming each process proposes a value, the consensus problem is defined by the

[☆] Parts of the work presented in this paper appeared in the ACM conferences SPAA 2001 and ICDCN 2016 [24,27].
[*] Corresponding author.
E-mail address: raynal@irisa.fr (M. Raynal).

http://dx.doi.org/10.1016/j.tcs.2017.03.018
0304-3975/© 2017 Elsevier B.V. All rights reserved.

following properties: each non-faulty process must decide a value (termination), such that the same value is decided by the non-faulty processes (agreement), and this value satisfies some validity condition, which depends on the proposed values and the considered failure model [13,29].

The k-set agreement problem is a natural weakening of consensus [10]. It allows the non-faulty processes to decide different values, as long as no more than k values are decided (the problem parameter k can be seen as the coordination degree imposed to processes). Hence, consensus is 1-set agreement. Let us notice that k-set agreement can be easily solved in crash-prone systems where k (the maximal number of different values that can be decided) is greater than t (the maximal number of processes that may be faulty). The k-set agreement problem has applications, e.g., to compute a common subset of wavelengths (each process proposes a wavelength and at most k of them are selected), or to duplicate k state machines where at most one is required to progress forever [16,35].

Crash and Byzantine failures. A process crash failure occurs when a process stops prematurely. After it crashed, a process never recovers; moreover it behaves correctly (i.e., according to its code) before crashing. A crash failure can be seen as a benign failure, as a crashed process did not pollute the computation before crashing (e.g., by disseminating fake values).

The situation is different with Byzantine failures. This failure type has been introduced in the context of synchronous distributed systems [21,29,33], and then investigated in the context of asynchronous distributed systems [2,22,34]. A process has a *Byzantine* behavior when it arbitrarily deviates from its intended behavior. We then say that it "commits a Byzantine failure" (otherwise we say the process is *non-faulty* or *correct*). This bad behavior can be intentional (malicious) or simply the result of a transient fault that altered the local state of a process, thereby modifying its behavior in an unpredictable way. Let us notice that, from a failure hierarchy point of view, process crashes (unexpected halting) constitute a strict subset of Byzantine failures. As asynchronous message-passing systems are more and more pervasive, the assumption "no process has a bad behavior" is no longer sensible. Hence, agreement in asynchronous Byzantine message-passing systems is becoming a more and more important issue of fault-tolerance.

An impossibility result and how to cope with it. Let us consider a system made up of n processes, where up to t may be faulty. Whatever the value of k (with respect to t), k-set agreement can always be solved if the system is synchronous [33]. The situation is different in asynchronous systems where k-set agreement is impossible to solve in the process crash failure model when $k \leq t$ [5,19,37]. As Byzantine failures are more severe than crash failures, this impossibility remains true in asynchronous Byzantine systems.

It follows from this impossibility that, when $k \leq t$, either the space of values that can be proposed must be restricted [14,25], or the underlying asynchronous distributed system must be enriched with additional computational power for k-set agreement to be solved. Such an additional computational power can be provided with partial synchrony assumptions (e.g., [11,39] which consider $k = 1$), minimal synchrony assumptions (e.g., [6] which considers $k = 1$ and Byzantine failures), appropriate failure detectors (e.g., [9,26] which consider $k = 1$ and crash failures, and [15] which considers $k = 1$ and Byzantine failures), or randomization (e.g., [3] which considers $k = 1$ and crash failures, [31] which considers $k = 1$ and Byzantine failures, [8] which considers $k \leq t$ and crash failures in read/write shared memory systems, and [27] which considers $k \leq t$ and crash failures in message-passing systems).

Intrusion-tolerant agreement with respect to Byzantine processes. The validity property associated with a distributed agreement problem relates its outputs to its inputs. As no process creates fake values in a crash-prone system, the k-set agreement validity property is easy to state, namely, a decided value must be a value proposed by a process. In a system where processes may commit Byzantine failures, there is no way to direct a Byzantine process to decide some specific value. Consequently the k-set agreement validity property can only be on the values decided by the correct processes. Moreover, the notion of a "value proposed by a faulty process" is dubious.

A classical validity property for Byzantine consensus (see, e.g., [22]) states that, if all the non-faulty processes propose the same value, they must decide it. Hence, as soon as two non-faulty processes propose different values, any value can be decided by the correct processes, even a value "proposed" by a Byzantine process. (Let us observe that a Byzantine process can appear as proposing different values to different correct processes.) More generally, and as noticed and deeply investigated in [30], it follows that the solvability of Byzantine k-set agreement is sensitive to the particular validity property that is considered.

This paper considers the following validity property (introduced in [28] where it is called *intrusion-tolerance*): no value proposed only by Byzantine processes can be decided by a non-faulty process. One way to be able to design a k-set algorithm providing this property, consists in allowing a non-faulty process to decide a default value $\perp$, except (to prevent triviality) when the non-faulty processes propose the same value. (The $\perp$ decision at some non-faulty processes can occur for example in the adversary scenario where the non-faulty processes propose different values, while the Byzantine processes propose the same value.) Another way to design a k-set algorithm providing intrusion-tolerance consists in adding a constraint on the total number of different values that can be proposed by the non-faulty processes. Let $m \geq 2$ be this number. It is shown in [18] that, in an n-process system where up to t processes may commit Byzantine failures, such a constraint is $n - t > mt$ (i.e., there is a value proposed by at least $(t + 1)$ non-faulty processes).

Content of the paper. This paper is on k-set agreement in n-process asynchronous message-passing systems, where $k \leq t$. It presents two algorithms. The first is a k-set agreement algorithm for asynchronous message-passing systems where up to