Solving agreement problems with failure detectors: a survey *

Jean-Michel HELARY **
Michel HURFIN **

Abstract

This paper surveys problems related to achieving agreement in distributed systems. Various agreement problems can be specified as a variant of the basic consensus problem. Unfortunately, this fundamental problem cannot be solved in asynchronous systems if crash failures can occur. In order to overcome this impossibility result, Chandra and Toueg have augmented the asynchronous system model with the notion of failure detectors. A failure detector is associated with each process of the distributed computation and is responsible for detecting external failures. Suspicions are essentially implemented using time-out mechanisms, which means that (1) the detection of a real failure is usually delayed and (2) a failure detector can make mistakes by incorrectly suspecting a process to have crashed. In this paper, various classes of failure detectors are presented. All are specified by a completeness property and an accuracy property. A completeness property puts a condition on the detection of crashed processes, while an accuracy property restricts the possible mistakes made by a failure detector. After reviewing the hierarchy of failure detector classes, two particular solutions to the consensus problem are examined. In each case, a different class of failure detectors is used. The proposed solutions have liveness properties and deliver their expected results provided that a minimal set of well-defined conditions is satisfied. After a detailed presentation of the consensus problem, various agreement problems are reviewed and their relationship with the consensus problem are underlined. Finally, for each agreement problem, we indicate under which minimum assumptions a solution can be found.

Key words: Computer theory, Distributed system, Distributed processing, Failure detection, Fault tolerant system, Information exchange, Reliability, Time-out, Review.

---

RÉSoudRE DES PROBLÈMES D’ACCORD À L’AIDE DE DÉTECTEURS DE DÉFAILANCES: UN TOUR D’HORIZON

Résumé

Cet article fait un tour d’horizon des problèmes nécessitant l’obtention d’un accord dans un système distribué. Différents problèmes d’accord peuvent être spécifiés comme une variante d’un problème élémentaire : le consensus. Malheureusement, ce problème fondamental ne peut pas être résolu dans des systèmes asynchrones dès lors que les processus sont susceptibles de stopper prématurément leur exécution. Afin de surmonter ce résultat d’impossibilité, Chandra et Toueg ont augmenté le modèle asynchrone en introduisant la notion de détecteur de défaillances. Un détecteur de défaillances est associé à chaque processus du calcul distribué et est chargé de détecter les défaillances externes. La mise en œuvre des suspicions se fait principalement via des mécanismes de temporisation. Cela signifie que (1) la détection d’une défaillance réelle est généralement différée et que (2) un détecteur de défaillances peut commettre des erreurs en suspectant, à tort, un processus d’avoir stoppé son exécution. Dans cet article, diverses classes de détecteurs de défaillances sont présentées. Toutes sont spécifiées par une propriété de complétude et une propriété d’exactitude. Une propriété de complétude définit les contraintes concernant la détection des processus réellement arrêtés tandis que la propriété d’exactitude vise à limiter les erreurs que peut commettre un détecteur de défaillances. Après avoir passé en revue la hiérarchie des classes de détecteurs de défaillances, deux solutions particulières au problème

* This work has been partially supported by a CNET-FRANCE TELECOM grant 95 IB 122.
** IRISA, Campus de Beaulieu, F-35042 Rennes Cedex, France, helary@irisa.fr.
Problem of agreement, we indicate under which hypotheses the consensus are examined. Finally, for each problem of agreement, different problems of accordance are examined and their relations with the problem of consensus are outlined. Finally, for each problem of agreement, we indicate under which hypotheses minimal solutions can be found.

Mots clés : Informatique théorique, Système réparti, Traitement réparti, Détection défaillance, Système tolérant les pannes, Echange d'information, Fiabilité, Temporisation, Article synthèse.

Contents

I. Introduction.
II. Unreliable asynchronous model of distributed computation.
III. Asynchronous models with failure detectors.
IV. Solving consensus.
V. Conclusion.
References (25 ref.).

I. INTRODUCTION

A distributed system consists of a set of communicating processes. Each process is the execution of a local program. Two broad categories of systems have been considered, according to the way used by processes to communicate. In shared-memory systems, processes communicate by accessing shared objects. In message-passing systems, processes communicate only by exchanging messages through communication channels. In this paper, we study only message-passing systems. In that context, several models of computation can be considered, depending on the nature and properties of system’s components (processes and communication channels). Among others, assumptions concerning synchrony and reliability of components must be precisely stated in order to define specific models. Considered as one of the most attractive is the category of asynchronous message-passing models, where no timing assumption can be made either on the relative speeds of processes or on the communication delays: processes run at their own pace (as long as they are not waiting for messages) and communication delays are unbounded. These models are attractive and have gained much currency for several reasons: they have simple semantics; applications programmed on the basis of this model are easier to port than those based upon specific timing assumptions; and, practically, variable or unexpected workloads are sources of asynchrony. Nevertheless, distributed systems are prone to failures. Different types of process and channel failures can be considered. They can be benign, e.g. omission failures whereby processes or channels fail to send, to deliver, or to transport expected messages; or crash failures, whereby a process fails by halting prematurely. They can be arbitrary (or malicious or Byzantine), when results produced by a process deviate arbitrarily from the program specification.

Several important paradigms have been identified to simplify the task of designing fault-tolerant distributed applications on the top of distributed systems. Among them, the class of agreement problems, containing problems such as consensus, reliable broadcasts, group memberships, commitments, etc., is of primary importance because of its wide applicability. But it was shown, in 1985, that the consensus problem cannot be solved deterministically in an asynchronous message-passing model if even a single process crash-failure occurs; this result is known as the Fisher-Lynch-Paterson impossibility result (or, in brief, FLP impossibility result [13]). Due to their similarity with consensus, most agreement problems suffer from the same impossibility result. Consequently, the consensus problem has been frequently used as a standard to compare computability in different asynchronous fault-tolerant distributed systems.

To circumvent such impossibility results, several approaches have been proposed: randomization (agreement is reached with a certain probability), weakening of agreement problems, introduction of some degree of synchrony into the computation model, etc. The failure detectors, introduced by Chandra and Toueg, are related to the latter approach. In fact, they are presented as an abstraction to hide partial synchrony assumptions. Therefore, they are specified by a set of abstract properties they satisfy as opposed to giving specific implementations. Applications using failure detector’s services can be designed, on top of unreliable asynchronous distributed systems, and this design relies solely on the properties of the failure detector’s services. In particular, failure detectors allow to define conditions under which some agreement problems, such as consensus, become solvable.

The aim of this paper is to present, on the one hand, the concept and a hierarchy of failure detector classes, and on the other hand, various agreement problems (using an unified formalism). The rest of this paper is organized as follows: in Section II, the model of computation is presented and some impossibility results are outlined. Section III presents the hierarchy of failure detector classes and addresses implementation concerns. Two solutions of the consensus problem are detailed in Section IV, and Section VI is devoted to the presentation and the solvability of agreement problems. Section VI concludes this survey.