Adding Dynamicity to the Uncertainty that Characterizes Distributed Systems: Challenges Ahead

Raimundo Macêdo[1] **and Michel Raynal**[2]

[1]Distributed Systems Laboratory (LaSiD/DCC)
Federal University of Bahia
Campus de Ondina – Salvador – BA – Brazil

[2]IRISA/INRIA
University of Rennes
Campus de Beulieu – Rennes – France

macedo@ufba.br, raynal@irisa.fr

Abstract. *The uncertainties inherent to distributed systems, such as unpredictable message delays and process failures, gave place to a great research effort in the past, where numerous solutions to fault-tolerance mechanisms have been proposed with a variety of guarantees and underlying system assumptions. The advent of new classes of distributed system applications (such as social networks, security, smart objects sharing etc) and technologies (VANET, WiMax, Airborn Networks, DoD Global Information Grid, P2P) are radically changing the way in which distributed systems are perceived. Such emerging systems have a composition, in terms of processes participating to the system, that is self-defined at run time depending, for example, on their will to belong to such a system, on the geographical distribution of processes etc. In this paper, we point to some of the challenges that have to be addressed by fault-tolerance solutions in the light of such a new dynamicity dimension, and that can motivate future collaborative work.*

1. Introduction

A distributed system is usually characterized by a set $\Pi = \{p_1, p_2, ..., p_n\}$ of processes sited on possibly distinct networked computers and a set $\chi = \{c_1, c_2, ..., c_m\}$ of communication channels. Networked computers form arbitrary network topologies and processes communicate by using a communication protocol that implements process-to-process communication. Such process-to-process communication defines communication channels that may include several intermediate network level communication links. Therefore, a communication channel c_i connecting processes p_i and p_j defines a "is able to communicate"relation between p_i and p_j, rather than a network level link connecting the machines that host p_i and p_j. The actual behavior or properties observed in the distributed system, which defines a distributed system model, results from the way processes and channels are implemented by the underlying operating and communication systems. Whether processes and channels can operate within certain known time bounds, and how and when processes and channels can fail, are the usual uncertainties that designers have to face when solving application problems in distinct models. That is why being able to solve such problems despite failures (i.e., being Fault-Tolerant) has been considered a major research challenge. In this context, the abstractions of distributed consensus

and concurrent registers are important building blocks from which fault-tolerance mechanisms such as state machine replication can be constructed. Basically, a consensus protocol guarantees that distributed processes can unanimously agree on a proposed output value, whereas concurrent registers provide distributed processes with certain guarantees on read/write operations. However, the possibility of solving these problems depends on the uncertainty level assumed in a given system model. For instance, let us consider the synchronous (or time-based) and asynchronous (or time-free) models. Under process crashes and reliable channels, the reliable multicast problem is solvable in both models [Lynch 1996, Hadzilacos and Toueg 1993], whereas distributed consensus can be solved in the synchronous model, but not in the asynchronous model [Fisher et al. 1985].

The advent of new classes of distributed system applications (such as social networks, security, smart objects sharing etc) and technologies (VANET, WiMax, Airborn Networks, DoD Global Information Grid, P2P) are radically changing the way in which distributed systems are perceived, adding even more uncertainties that lead to more dynamic distributed system models.

Such emerging systems have a composition, in terms of processes participating to the system, that is self-defined at run time depending, for example, on their will to belong to such a system, on the geographical distribution of processes etc. Therefore, a common denominator of such emerging systems is the dynamicity dimension related to the processes that actually make the system at a given time, which introduces a new source of unpredictability inside a distributed system. Such a dynamicity reflects also on the available system resources that dynamically changes following system compositions. This in turn requires applications to be adaptive, for instance, to less network bandwidth or degraded Quality-of-Service (QoS). Ideally, in these highly dynamic scenarios, adaptiveness characteristics of applications should be self-managing or autonomic.

In this paper, we point to some of the challenges that need to be addressed by distributed consensus and concurrent register solutions in the light of such a new dynamicity dimension, and should motivate future collaborative work of the Franco-Brazilian community working in distributed computing. The reminder of this extended abstract is structured as follows. In section 2 is presented related work with emphasis on our previous collaborative work on this field. Challenges that should be of primary importance for the Franco-Brazilian research community interested in distributed computing are presented in section 3.

2. Related and Previous Work

Among the faul-tolerant problems, distributed consensus has received a great deal of attention because it can be used as a basic building block to solve several class of problems that includes group membership, atomic commitment, atomic broadcast, among others. Motivated by the consensus impossibility result in asynchronous systems, researchers have proposed a number of partially synchronous distributed system models that introduce different levels of synchrony into the asynchronous system, where the consensus problem is solvable [Dwork et al. 1988, Dolev et al. 1987, Cristian and Fetzer 1999, Lamport 1998, Chandra and Toueg 1996]. Among them, the failure detectors mechanism proposed by Chandra and Toueg[Chandra and Toueg 1996] has received special attention because of its simplicity, encapsulating the synchrony needed to achieve consensus by

defining axiomatic properties associated with different classes of failures detectors.

In [Hurfin et al. 1999] we have defined a general framework that simplifies the system designer work. Such framework can be used to solve a variety of classes of distributed agreement problems based on failure detectors. Following the same paradigm of failure detectors, we defined consensus protocols based on a decentralized communication pattern and observed that a tradeoff has to be found between the number of communication steps and the number of exchanged messages, by automatically switching between a centralized and a decentralized communication pattern [Greve et al. 2000]. The notion of round is at the core of failure detector based Protocols. To be more tolerant with regard to the message transfer delays, we introduced the notion of cycle of K rounds [Hurfin et al. 2001].

In [Gorender et al. 2007] we have explored the notion of a hybrid model to solve the uniform consensus problem, where we assume that the underlying system is capable of providing distinct QoS communication guarantees. Because timely and untimely channels may exist in parallel, the underlying system model can be hybrid in space (in this sense, similar to TCB [Veríssimo and Casimiro 2002]), but, differently from TCB, the nature of such hybridism does not require that all processes are interconnected by timely channels (which would characterize a synchronous wormhole).

In recent work, we have investigated the implementation of regular registers in asynchronous dynamic message-passing systems, where the dynamicity rate (i.e., frequency of joins and leaves of the processes) is constant [Baldoni et al. 2009]. In another recent work, we focused on the support for self-management behavior, that allows an application to dynamically adapt to a given system configuration [Andrade and de Araújo Macêdo 2009]. These research efforts represent a preliminary endeavor towards models and mechanisms that fulfill the requirements of modern dynamic distributed systems.

3. New Challenges

The new classes of highly dynamic distributed system applications and technologies pose a tremendous challenge to fault-tolerant systems. The main difficulty stems from the fact that in distributed systems the implementation of fault-tolerance mechanisms (such as replication) depends on coordinated actions from the system processes. As the system composition changes dynamically, it is difficult to assure that the necessary resources will be kept long enough in order to guarantee the prescribed fault-tolerance properties (for instance, that redundancy level is kept satisfactory).

So, the main challenges consists in defining appropriate *abstractions suited to dynamic systems*, and related self-management mechanisms. Those abstractions should be *locality-based* and should take into account notions such a *presence detector* (similar to failure detectors). From a computational point of view, an insight into population protocols [Angluin et al. 2007] should be of primary importance and could be the starting point of promising research inside the realm of distributed computing.

Referências

Andrade, S. S. and de Araújo Macêdo, R. J. (2009). A non-intrusive component-based approach for deploying unanticipated self-management behaviour. In *Proceedings of*

IEEE ICSE 2009 Workshop Software Engineering for Adaptive and Self-Managing Systems.

Angluin, D., Aspnes, J., Eisenstat, D., and RuppertDwork, E. (2007). The computational power of population protocols. *Distributed Computing*, 20(4):279–304.

Baldoni, R., Bonomi, S., Kermarrec, A.-M., and Raynal, M. (2009). Implementing a register in a dynamic distributed system. In *International Conference on Distributed Computing Systems (ICDCS2́009)*.

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

Cristian, F. and Fetzer, C. (1999). The timed asynchronous distributed system model. *IEEE Transactions on Parallel and Distributed Systems*, 10(6):642–657.

Dolev, D., Dwork, C., and Stockmeyer, L. (1987). On the minimal synchronism needed for distributed consensus. *Journal of the ACM*, 34(1):77–97.

Dwork, C., Lynch, N., and Stockmeyer, L. (1988). Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2):288–323.

Fisher, M. J., Lynch, N., and Paterson, M. S. (1985). Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382.

Gorender, S., de Araújo Macêdo, R., and Raynal, M. (2007). An Adaptive Programming Model for Fault-Tolerant Distributed Computing. *IEEE Transactions on Dependable and Secure Computing*, pages 18–31.

Greve, F., Hurfin, M., Macêdo, R., and Raynal, M. (2000). Consensus based on strong failure detectors : A time and message-efficient protocol. In *IEEE International Workshop on Fault-Tolerant Parallel and Distributed Systems. Lecture Notes in Computer Science 1800. Springer Verlag,pg 1258-1267.*

Hadzilacos, V. and Toueg, S. (1993). *Distributed Systems (Edited by Sape Mullender). Chapter 5. Fault-Tolerant Broadcasts and Related Problems*. Addison-Wesley Pub Co.

Hurfin, M., Macêdo, R., Mostefaoui, A., and Raynal, M. (2001). A consensus protocol based on a weak failure detector and a sliding round window. In *20th IEEE Int. Symposium on Reliable Distributed Systems (SRDS0́1).*

Hurfin, M., Macêdo, R., Raynal, M., and Tronel, F. (1999). A general framework to solve agreement problems. In *18th IEEE Symposium on Reliable Distributed Systems (SRDS9́9).*

Lamport, L. (1998). The part time parliament. *ACM Trans. on Computer Systems*, 16(2):133–169.

Lynch, N. A. (1996). *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc.

Veríssimo, P. and Casimiro, A. (2002). The timely computing base model and architecture. *IEEE Transactions on Computers*, 51(8):916–930.