Boosting Dependable Ubiquitous Computing: A Case Study

Christian Fernández-Campusano, Roberto Cortiñas, and Mikel Larrea

University of the Basque Country UPV/EHU, Spain
{christianrobert.fernandez,roberto.cortinas,mikel.larrea}@ehu.es

Abstract. Ubiquitous computing has inherent features, e.g., a number of elements in the system with restricted communication and computation capabilities, which make harder to achieve dependability. In this work we explore this research line through the study of TrustedPals, a smartcard-based framework which allows implementing security policies in ubiquitous systems and applications in a decentralized way. The current architecture of TrustedPals uses a consensus algorithm adapted to the omission failure model. In this work, we propose to alternatively use the Paxos algorithm, in order to address more ubiquitous environments.

1 Introduction

Dependability is composed of several aspects, such as availability and reliability, which imply fault-tolerance. In this regard, reaching agreement is a key topic to achieve dependability. Consensus [1] is one of the most important problems in fault-tolerant distributed computing, and constitutes a paradigm that represents a family of agreement problems. Roughly speaking, in consensus processes propose an initial value and have to decide on one of the proposed values.

Although many solutions have been proposed to solve consensus in synchronous systems, Fischer et al. [2] showed that it is impossible to solve consensus deterministically in asynchronous systems where at least one process may crash. In order to circumvent this impossibility, Chandra and Toueg [3] proposed the failure detector abstraction. Roughly speaking, a failure detector is an abstract module located at each process of the system that provides information about (the operational state of) other processes in the system. Failure detectors offer a modular approach that allows other applications such as consensus to use them as a building block. Additionally, the failure detector abstraction allows to encapsulate the synchrony assumptions of the system, so applications that make use of failure detectors can be designed as if they run in an asynchronous system.

Recently, Cortiñas et al. [4] proposed a modular architecture which combines failure detectors with a tamper-proof smartcard-based secure platform named *TrustedPals* [5] in order to solve consensus in a partially synchronous system prone to Byzantine failures. They also showed how to solve a security problem called *Secure Multiparty Computation* [6] through consensus. Secure Multiparty Computation is a general security problem that can be used to solve various real-life problems such as distributed voting, private bidding and online auctions.

G. Urzaiz et al. (Eds.): UCAmI 2013, LNCS 8276, pp. 42–45, 2013.

2 Current and Proposed Architecture

Figure 1 (left) presents the current architecture of TrustedPals [4], which is composed of the following elements:

- The TrustedPals platform allows to transform every failure into a process crash or a message omission, which implies that the initial Byzantine failure model is turned into a more benign one, namely the omission failure model.
- An Eventually Perfect ($\diamond\mathcal{P}$) failure detector adapted to the omission failure model ($\diamond\mathcal{P}_{om}$) allows to detect *well connected* processes, i.e., processes that can actively participate in the consensus.
- Finally, a consensus algorithm adapted from [3] allows to reach agreement by using the previous $\diamond\mathcal{P}_{om}$ failure detector.

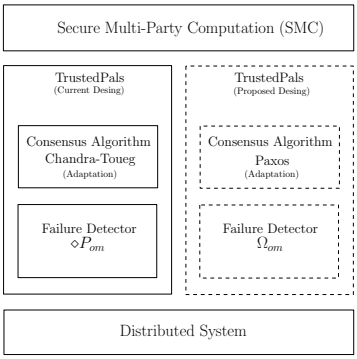

Fig. 1. Current architecture (left) vs proposed one (right)

Although the proposed solution is suitable in the security context presented of [4], it presents some drawbacks that could be improved in order to be applied in other scenarios. For example, it does not consider processes which crash and later recover. Also, the consensus algorithm requires a high degree of reliability on communication, i.e., non-omissive processes and quasi-reliable channels.

Figure 1 (right) presents the architecture we propose in this work. Note that it keeps the modular approach of the previous one, but with two main differences:

- The previous consensus algorithm is replaced by the Paxos algorithm [7].
- The failure detector class considered now is Ω [8].

The combination of those two new elements in the architecture provides the system with several interesting features. On the one side, Paxos allows to reach consensus tolerating a high degree of omissive behaviour (loss of messages at processes or channels). Also, Paxos allows to cope with processes which crash and later recover. On the other hand, the Omega failure detector, Ω, provides Paxos with the eventually stable leader required to guarantee termination.