Quorum-Based Replication in Asynchronous Crash-Recovery Distributed Systems

Luís Rodrigues¹ and Michel Raynal²

¹ Universidade de Lisboa
ler@di.fc.ul.pt,
² IRISA
raynal@irisa.fr

Abstract. This paper describes a solution to the replica management problem in asynchronous distributed systems in which processes can crash and recover. Our solution is based on an Atomic Broadcast primitive which, in turn, is based on an underlying Consensus algorithm. The proposed technique makes a bridge between established results on Weighted Voting and recent results on the Consensus problem.

1 Introduction

Replication is a well known technique to increase the reliability and availability of data [8]. Replication involves coordination among replicas. For instance, replicas may need to agree on a common state after a given action or on the order by which requests will be processed. Several of these coordination activities are instances of the Consensus problem, that can be defined in the following way: each process proposes an initial value to the others, and, despite failures, all correct processes have to agree on a common value (called decision value), which has to be one of the proposed values. Unfortunalty, this problem has no deterministic solution in asynchronous systems where processes may fail, a result known as the Fischer-Lynch-Paterson’s (FLP) impossibility result [5].

The FLP impossibility result has motivated researchers to find a set of minimal assumptions that, when satisfied by a distributed system, makes consensus solvable in this system. The concept of unreliable failure detector introduced by Chandra and Toueg constitutes an answer to this challenge [4]. From a practical point of view, an unreliable failure detector can be seen as a set of oracles: each oracle is attached to a process and provides it with information regarding the status of other processes. An oracle can make mistakes, for instance, by not suspects a failed process or by suspecting a not failed one. The concept has also been extended to the crash-recovery model [1,9,11].

Weighted voting [6] is a well known technique to manage replication in the crash-recovery model. The technique consists in assigning votes to each replica and define quorums for read and write operations. Quorums for conflicting operations, namely read/write and write/write must overlap such that conflicts can be detected. Typically, voting algorithms are applied in the same context of transactions [7]: quorums ensure one-copy equivalence for each replica, concurrency
control techniques ensure mutual consistency of data and atomic commitment protocols ensure updates persistence (write operations are performed in the write quorum). It should be noted that, in asynchronous systems, these solutions must also rely on variants of Consensus to decide the outcome of transactions [12].

This paper explores an alternative path to the implementation of quorum-based replication that relies on the use of an Atomic Broadcast primitive. An Atomic Broadcast primitive allows processes to broadcast and deliver messages in such a way that processes agree not only on the set of messages they deliver but also on the order of message deliveries. By employing this primitive to disseminate updates, all correct copies of a service are delivered the same set of updates in the same order, and consequently the state of the service is kept consistent. The proposed technique makes: i) a bridge between established results on Weighted Voting and recent results on the Consensus problem; ii) a bridge between the active replication model in the synchronous crash (no-recovery) model and the asynchronous crash-recovery model.

2 System Model and Building Blocks

We consider a system consisting of a finite set of processes. At a given time, a process is either up or down. When it is up, a process progresses at its own speed behaving according to its specification. While being up, a process can fail by crashing: it then stops working and becomes down. A down process can later recover: it then becomes up again and restarts by invoking a recovery procedure. The model is augmented with a failure detector so that the Consensus can be solved [9,11,11]. Each process is equipped with two local memories: a volatile memory and a stable storage. When it crashes, a process definitely loses the content of its volatile memory; the content of a stable storage is not affected by crashes.

Processes communicate and synchronize by sending and receiving messages through channels. The quorum-based replica management algorithm requires the use of an unreliable transport protocol and of an atomic broadcast protocol. It has been shown that the atomic broadcast problem is equivalent to Consensus in asynchronous crash-recovery systems [13]. By resourcing to the atomic broadcast protocol our algorithm does not use a Consensus protocol explicitly (the Consensus is encapsulated by the atomic broadcast primitive).

3 Quorum-Based Replica Management

Weighted voting [6] is a popular technique to increase the availability of replicated data in networks subject to node crashes or network partitions. The technique consists in assigning votes to each replica and define quorums for read read and write operations. Quorums for conflicting operations, namely read/write and write/write must overlap such that conflicts can be detected. Typically, voting algorithms are applied in the context of transactions [7]: quorums ensure one-copy equivalence for each replica, concurrency control techniques ensure mutual