Weak Ordering Oracles for Failure Detection-Free Systems

Fernando Pedone[†] André Schiper[‡] Péter Urbán[‡] David Cavin[‡*]

[†]Hewlett-Packard Laboratories — Software Technology Laboratory
Palo Alto, CA 94304, USA
fernando_pedone@hp.com

[‡]Ecole Polytechnique Fédérale de Lausanne (EPFL) — Faculté I&C
CH-1015 Lausanne, Switzerland
{andre.schiper, peter.urban, david.cavin}@epfl.ch

1 Motivation

Agreement abstractions, such as consensus, atomic broadcast, and generic broadcast [2], are important building blocks in distributed systems subject to processor failures. Atomic broadcast, for example, has been used to build many fault-tolerant systems, such as highly-available databases. Atomic broadcast guarantees that if a message is broadcast to a group of processors and one of these processors delivers the message, then all processors also deliver the message — a property known as *agreement*; and if two processors deliver the same two messages, they do so in the same order — a property known as *total order*.

Asynchronous systems. A protocol implementation is as general as the underlying assumptions about its system model. In this sense, the asynchronous model of computation is a very general way to formalize distributed systems. It basically makes no assumptions about the time it takes to transmit messages over network links and for processors to perform their computations. The asynchronous model is very appealing as any solution that can be applied in it can also be applied in a stronger model (e.g., the synchronous model, where processors can estimate the time it takes for other processors to execute their programs).

Nevertheless, one fundamental result about distributed systems subject to processor failures states that atomic broadcast cannot be solved in a pure asynchronous model. The basic reason stems from the fact that in pure asynchronous systems, processors cannot tell a very slow processor (i.e., a processor that takes a long time to reply to the others) from a processor that has failed.

To circumvent this impossibility result, current proposals have extended the asynchronous model with further assumptions, such as failure detectors.

Failure detectors. Failure detectors provide processors with information about processors failures [1]. Failure detectors can make mistakes, that is, they can suspect processors that have not failed and not suspect processors that have failed. Several classes of failure detectors have been proposed [1].

Failure detectors are usually implemented with some timeout mechanism and even the weakest failure detector requires some timeout calibration to satisfy its properties. Moreover, even though protocols based on failure detectors can cope with wrong processor suspicions, their performance is hurt when suspicions are wrong. But more importantly, failure detectors have to deal with the following dilemma:

> THE FAILURE DETECTOR DILEMMA. On the one hand, timeout values should be large in order to maximize performance (by avoiding wrong suspicions). On the other hand, timeout values should be small in order to ensure a fast reaction to failures.

2 Weak Ordering Oracles

Instead of failure detectors, we propose to extend the asynchronous system with *weak ordering oracles*. Weak ordering oracles capture the behavior of network multicast in state-of-the-art local-area networks: if processors send multicast messages in

*Research supported by the Swiss National Science Foundation NCCR MICS project and the CSEM Swiss Center for Electronics and Microtechnology, Inc., Neuchâtel.

 Appeared in *Proc. of the 2002 International Conference on Dependable Systems and Networks (DSN), supplemental volume, pages B-32 and B-33.*

a local-area network, there is a good chance that some of the messages will be received by all processors in the same order. To illustrate this fact, Figure 1 shows the results of measurements performed in a cluster of 12 PCs connected by a 100 Mbit/s Ethernet. Each host broadcasts messages to all other hosts using IP multicast. When messages are broadcast with a period greater than ≈0.14 milliseconds, very few messages are received out of order (only about 5%).

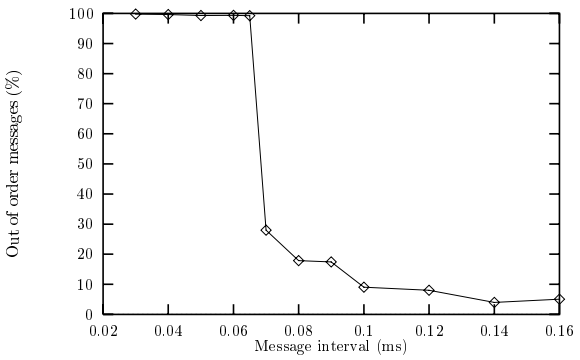

Figure 1: Spontaneous total order property

Briefly (see [3] for more details), our oracle is defined by the primitives W-ABroadcast(r, m) (query to broadcast message m) and W-ADeliver(r, m) (delivery of m). The integer parameter r groups messages with the same r value. The weak ordering property holds for r if there exists a message m such that each process delivers W-ADeliver(r, m) first (among all W-ADeliver$(r, -)$).

To illustrate this property, consider three processes, p_1, p_2, and p_3, executing the following queries to the oracle (for brevity, we denote next W-ABroadcast(r, m) by B(r, m)):

- p_1 executes B$(0, m_1)$; B$(1, m_2)$; B$(2, m_3)$,
- p_2 executes B$(0, m_4)$; B$(1, m_5)$; B$(2, m_6)$,
- p_3 executes B$(0, m_7)$; B$(1, m_8)$; B$(2, m_9)$,

and assume the following sequences of W-ADeliver(r, m) (for brevity, we denote next W-ADeliver(r, m) by D(r, m)):

- on p_1: D$(0, m_1)$; D$(1, m_2)$; D$(0, m_4)$; D$(2, m_3)$; D$(0, m_7)$; etc.
- on p_2: D$(0, m_4)$; D$(0, m_1)$; D$(1, m_5)$; D$(0, m_7)$; D$(2, m_3)$; etc.
- on p_3: D$(0, m_4)$; D$(0, m_7)$; D$(2, m_3)$; D$(1, m_8)$; etc.

The weak ordering property holds for $r = 2$ (m_3 is the first message with $r = 2$ delivered by p_1, p_2, and p_3), but does not hold for either $r = 0$ or $r = 1$.

We define the Weak Atomic Broadcast (WAB) oracle as the oracle that fulfills the weak ordering property infinitely often.

Consensus and atomic broadcast can be solved using the WAB oracle [3]. Actually consensus can be solved with an oracle weaker than the WAB oracle [3]. When using weak ordering oracles, atomic broadcast algorithms and other agreement-related algorithms do not suffer from the failure detector dilemma. Since they do not rely on failure detectors, there is no tuning of timeouts involved in the execution of an atomic broadcast. As a result, there is no notion of *reaction time to failures* with these protocols. The performance of algorithms using weak ordering oracles is as good in the presence of failures as in the absence of failures.

3 Experiments

We have implemented and conducted several experiments using our atomic broadcast algorithm. In the setup used to obtain Figure 1, and for a range between 10 and 250 messages broadcast per second, we have found that our atomic broadcast implementation using weak ordering oracles outperforms failure detector-based implementations of atomic broadcast using a timeout of 4 milliseconds on approximately 30%. For timeout values of 10 milliseconds failure detector-based implementations outperform weak ordering oracle based implementations by less than 30%. We believe that the performances of our weak ordering oracle implementation may further be improved.

Acknowledgments

We thank Matthias Wiesmann for providing us with Figure 1.

References

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

[2] F. Pedone and A. Schiper. Handling message semantics with generic broadcast protocols. *Distributed Computing*, 15(2):97–107, 2002.

[3] F. Pedone, A. Schiper, P. Urbán, and D. Cavin. Solving agreement problems with weak ordering oracles. Technical Report HPL-2002-44, Hewlett Packard Laboratories, 2002. Also appears as EPFL Technical Report IC/2002/010, March 2002.