Brief Announcement: A Framework for Semantic Reasoning about Byzantine Quorum Systems

Evelyn Pierce
Département de Systèmes de Communication
Ecole Polytechnique Federale de Lausanne
1015 Lausanne, Switzerland
evelyn.pierce@epfl.ch

Lorenzo Alvisi
Department of Computer Sciences
University of Texas at Austin
Austin, TX USA
lorenzo@cs.utexas.edu

ABSTRACT
We have defined a class of shared variables called *TS-variables* that includes those implemented by the various Byzantine quorum system constructions of Malkhi and Reiter, and developed a number of definitions and theorems enabling us to reason about these variables abstractly. Using these tools, we have reduced the problem of Lamport's atomic semantics for such variables to the simpler problem of regular semantics. We discuss the fact that both these problems have remained stubbornly difficult to solve for some types of Byzantine quorum system variables (notably *masking quorum system variables*) by showing that they are not solvable by traditional approaches in an asynchronous environment. Finally, for such variables we define the notion of pseudoregular and pseudoatomic semantics, and state briefly that a similar reduction holds for these concepts.

Categories and Subject Descriptors
B.8.1 [**Performance and Reliability**]: Reliability, Testing, and Fault-Tolerance; C.2.4 [**Computer Systems Organization**]: Distributed Systems—*client/server*

General Terms
Design, Reliability, Theory

Keywords
atomic variable semantics, Byzantine fault tolerance, quorum systems, distributed data services

1. INTRODUCTION
Analyzing the semantics of shared variables implemented by quorum systems, particularly Byzantine quorum systems, can be quite challenging. Heretofore, such analysis has been limited to individual protocols; there has been no framework for reasoning about the semantics of quorum variables as a family. Thus, while linearizable operations have been straightforwardly achieved for some types of quorum systems (notably the dissemination quorum systems of [4]), there have been no tools available for generalizing these solutions or for exploring the limits of their applicability to other types of quorum systems for which such operations have been significantly more difficult to achieve (e.g., masking quorum systems, [3]).

Addressing this need was one of the primary motivations for this work. We have developed a set of definitions and theorems that allow us to reason about the *class* of shared variables implemented by quorum systems. We have adapted Lamport's formal definitions of the concepts of *safe*, *regular*, and *atomic* semantics [2], which have traditionally been used to describe the semantics of Byzantine quorum systems, so that they can be applied in a direct and formal manner to such variables.

Using these formalisms, we have proved that the writeback mechanism used in [4] to produce atomic semantics for dissemination quorum systems can in fact be used to implement an atomic protocol *from any regular TS-variable protocol*. The correctness of the atomic protocol of [4] can thus be viewed as a corollary of our result.

We have also used our formalism to explore the question of why similar approaches have consistently failed to solve the problem of atomic semantics for masking quorum systems. As a preliminary result of this analysis, we have shown that for a significant subclass of TS-variables including masking quorum systems, traditional approaches to protocol design inherently permit the existence of failed read queries (aborted, retried or incorrect) in an asynchronous environment. We have also shown that the somewhat weaker notions of *pseudoregular* and *pseudoatomic* semantics, in which aborted reads are tolerated, can be straightforwardly implemented for such systems.

2. TS-VARIABLES AND THE REDUCTION PROBLEM

The basic tool used in this work is an abstraction called a *TS-variable*. Informally, a TS-variable is a distributed shared variable whose value is read and written in conjunction with an associated timestamp, and which has the following additional property:

PROPERTY 1. *All write operations on a TS-variable are linearizable by timestamp.*

In other words, writes to a TS-variable are totally ordered by timestamp, and this total order is consistent with the partial order in which the operations are actually processed.

Following the example of [3], we discuss the semantics of these variables using a slightly generalized version of Lamport's definitions of *regular* and *atomic* semantics ([2]). We omit our formal specification of the generalized properties for reasons of space. Informally, however:

DEFINITION 1. *A TS-variable v is* regular *if any read operation on v returns either the value of the most recently completed write operation or the value of some write operation that is concurrent with the read.*

DEFINITION 2. *A TS-variable v is* atomic *if all read and write operations on v are linearizable, i.e., they behave as though performed in some linear order that is an extension of the partial order in which they are actually processed.*

2.1 The reduction

We extend an arbitrary regular TS-variable read/write protocol as follows: for read operation $read(val, ts)$ on a TS-variable v, let $writeback(val, ts)$ be the operation of rewriting the resulting value and timestamp back to v. We do not specify how this operation is to be implemented, except that we require that the extended protocol remain regular (as a property of reads and writes, disregarding writebacks) and that the following additional property be observed:

PROPERTY 2. *If a read operation r follows a write operation w and a writeback operation wb, then the timestamp of the value returned by r is at least as great as the greater of the timestamps used by w and wb.*

Our contention is that, given a regular TS-variable with read protocol $read(val, ts)$ and write protocol $write(val)$, the protocol

$$\mathbf{Write}_{atom}(val) \stackrel{def}{=} write(val)$$
$$\mathbf{Read}_{atom}(val, ts) \stackrel{def}{=} read(val, ts); writeback(val, ts)$$

defines an atomic TS-variable protocol. We omit the full proof for reasons of space; however, the structure of our argument is as follows:

1. We observe that the new protocol also implements a TS-variable.[1]
2. We specify an order relation ts' for the new protocol that extends both the partial order of operations by timestamp and the partial order of operations by order of processing.
3. We prove that ts' is in fact a total order on the operations of the new protocol.

These steps suffice to show that the new protocol is, in fact, atomic. Thus any regular TS-variable protocol can be strengthened to an atomic protocol by means of this simple enhancement.

3. APPLICATION TO MASKING QUORUM SYSTEMS

Variables implemented by masking quorum systems, as originally proposed in [3], are easily seen to be TS-variables. They also enjoy the following attractive features:

1. No ordering of operations is required
2. No server intercommunication is required in the absence of faults.
3. No locks or other devices for mutual exclusion are required.
4. A low constant number of communiation rounds suffices for each operation.
5. A single version of the variable is maintained at each server.

Unfortunately, unlike the dissemination quorum systems presented in the same paper, masking quorum systems also have the following limitation:

- A successful read requires identical responses from multiple servers.

This limitation, in combination with the features listed above, is inconsistent with regular semantics in an asynchronous environment. To see this, note that (4) and (5), above, imply that the system returns a bounded number of versions of the variable in response to any given read operation.[2] Call this bound k. By (3), it is possible for a given read operation to be concurrent with any number p of write operations in an asynchronous environment. And by (1) and (2), if $p \geq k-1$, then it is possible for every version of the variable received by the read operation to be different, with the result that the read operation cannot successfully determine a value; it must therefore return a possibly incorrect (i.e., old or nonsensical) value or abort. Such a result is not permitted under regular semantics.

[1]This is very straightforward, as the write operation of the new protocol is exactly the same as that of the underlying regular TS-variable protocol.

[2]Note that the same is true if each server maintains a bounded history of the variable.

It is for precisely this reason that the problem of regular (and thus atomic) semantics has proved significantly more difficult for masking quorum systems than for dissemination (or indeed benign-fault-tolerant) quorum systems.

3.1 Pseudo-Atomic Semantics

Fortunately, for applications that can tolerate some probability of aborted operations, especially those for which writes are significantly less frequent than reads, it is possible to implement an approximation of regular semantics for masking quorum systems while maintaining the advantages listed above. We call this approximation *pseudo-regular semantics*:

Definition 3. *A TS-variable protocol is pseudo-regular if any read operation that is not concurrent with any write operations returns the value of the most recently completed write, while a read that is concurrent with one or more write operations* either *returns the value of one of those operations or aborts.*

(An example of such a protocol for a masking quorum system variable, and an appropriate application, are given in [1].)

The writeback enhancement described above can also be used to promote such a protocol to a *pseudo-atomic* protocol, where:

Definition 4. *A TS-variable protocol is* pseudo-atomic *if all non-aborted read and write operations are linearizable, and the only aborted operations are reads concurrent with one or more writes.*

The proof of this reduction is almost identical to the one described in the previous section.

4. CONCLUSION

The purpose of the work summarized here is to provide tools for formally analyzing the semantics of shared variables implemented by various types of quorum systems, and to improve our understanding of the relative difficulty of implementing various levels of semantics for various types of quorum system variables. We hope that these results will help to facilitate and focus future research in this area.

5. ACKNOWLEDGEMENTS

We would like to express our sincere thanks to Dahlia Malkhi and Michael Reiter for numerous thought-provoking discussions, and to Jennifer Welch, HyunYoung Lee, Pete Manolios and Phoebe Weidmann for their helpful comments on this work.

6. REFERENCES

[1] L. Alvisi, D. Malkhi, E. Pierce, M. Reiter, and R. Wright. Dynamic byzantine quorum systems. In *Proceedings of the 1st International Conference on Dependable Systems and Networks*, pages 283–292, June 2000.

[2] L. Lamport. On interprocess communications. *Distributed Computing*, 1:77–101, 1986.

[3] D. Malkhi and M. Reiter. Byzantine quorum systems. *Distributed Computing*, 11(4):203–213, 1998.

[4] D. Malkhi and M. Reiter. Secure and scalable replication in phalanx. In *Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems*, pages 51–58, October 1998.