Brief Announcement: Sharing Memory in a Self-stabilizing Manner*

Noga Alon[1], Hagit Attiya[2], Shlomi Dolev[3], Swan Dubois[4], Maria Gradinariu[4], and Sébastien Tixeuil[4]

[1] Sackler School of Mathematics and Blavatnik School of Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv, 69978, Israel
nogaa@tau.ac.il

[2] Department of Computer Science, Technion, 32000, Israel
hagit@cs.technion.ac.il

[3] Contact author. Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, 84105, Israel
dolev@cs.bgu.ac.il

[4] LIP6, Universite Pierre et Marie Curie, Paris 6/INRIA, 7606, France

Introduction. A core abstraction for many distributed algorithms simulates shared memory [3]; this abstraction allows to take algorithms designed for shared memory, and port them to asynchronous message-passing systems, even in the presence of failures. There has been significant work on creating such simulations, under various types of permanent failures, as well as on exploiting this abstraction in order to derive algorithms for message-passing systems (see [2].)

All these works, however, only consider permanent failures, neglecting to incorporate mechanisms for handling *transient* failures. Such failures may result from incorrect initialization of the system, or from temporary violations of the assumptions made by the system designer, for example the assumption that a corrupted message is always identified by an error detection code. The ability to automatically resume normal operation following transient failures, namely to be *self-stabilizing* [4], is an essential property that should be integrated into the design and implementation of systems.

This paper presents the first practically self-stabilizing simulation of shared memory that tolerates crashes. Specifically, we simulate a single-writer multi-reader (SWMR) *atomic* register in asynchronous message-passing systems where less than a majority of processors may crash. The simulation is based on reads and writes to a (majority) quorum in a system with a fully connected graph

* More details can be found in [1]. Noga Alon is supported in part by an ERC advanced grant, by a USA-Israeli BSF grant, by the Israel Science Foundation. Hagit Attiya is supported in part by the *Israel Science Foundation* (grant number 953/06). The work started while Shlomi Dolev was a visiting professor at LIP6 supported in part by the ICT Programme of the European Union under contract number FP7-215270 (FRONTS), Microsoft, Deutsche Telekom, US Air-Force and Rita Altura Trust Chair in Computer Sciences.

N.A. Lynch and A.A. Shvartsman (Eds.): DISC 2010, LNCS 6343, pp. 525–527, 2010.

topology[1]. A key component of the simulation is a new bounded labeling scheme that needs no initialization, as well as a method for using it when communication links and processes are started at an arbitrary state.

Overview of our simulation. A simulation of a SWMR atomic register in a message-passing system, supports two procedures, read and write, for accessing the register. The ABD simulation [3] is based on a quorum approach: In a write operation, the writer makes sure that a quorum of processors (consisting of a majority of the processors, in its simplest variant) store its latest value. In a read operation, a reader contacts a quorum of processors, and obtains the latest values they store for the register; in order to ensure that other readers do not miss this value, the reader also makes sure that a quorum stores its return value.

A key ingredient of this scheme is the ability to distinguish between older and newer values of the register; this is achieved by attaching a *sequence number* to each register value. In its simplest form, the sequence number is an unbounded integer, which is increased whenever the writer generates a new value. This solution could be appropriate for a an *initialized* system, which starts in a consistent configuration, in which all sequence numbers are zero, and are only incremented by the writer or forwarded as is by readers. In this manner, a 64-bit sequence number will not wrap around for a number of writes that is practically infinite, certainly longer than the life-span of any reasonable system.

However, when there are transient failures in the system, as is the case in the context of self-stabilization, the simulation starts at an uninitialized state, where sequence numbers are not necessarily all zero. It is possible that, due to a transient failure, the sequence numbers might hold the maximal value when the simulation starts running, and thus, will wrap around very quickly.

Our solution is to partition the execution of the simulation into *epochs*, namely periods during which the sequence numbers are supposed not to wrap around. Whenever a "corrupted" sequence number is discovered, a new epoch is started, overriding all previous epochs; this repeats until no more corrupted sequence numbers are hidden in the system, and the system stabilizes. Ideally, in this steady state, after the system stabilizes, it will remain in the same epoch (at least until all sequence numbers wrap around, which is unlikely to happen).

This raises, naturally, the question of how to label epochs. The natural idea, of using integers, is bound to run into the same problems as for the sequence numbers. Instead, we capitalize on another idea from [3], of using a bounded labeling scheme for the epochs. A *bounded labeling scheme* [6,5] provides a function for generating labels (in a bounded domain), and guarantees that two labels can be compared to determine the largest among them.

Existing labeling schemes assume that initially, labels have specific initial values, and that new labels are introduced only by means of the label generation function. However, transient failures, of the kind the self-stabilizing simulation must withstand, can create incomparable labels, so it is impossible to tell which is the largest among them or to pick a new label that is bigger than all of them.

[1] Standard end-to-end schemes can be used to implement the quorum operation in the case of general communication graph.

To address this difficulty, we present a constructive bounded labeling scheme that allows to define a label larger than *any set* of labels, provided that its size is bounded. We assume links have bounded capacity, and hence the number of epochs initially hidden in the system is bounded.

The writer tracks the set of epochs it has seen recently; whenever the writer discovers that its current epoch label is not the largest, or is incomparable to some existing epoch, the writer generates a new epoch lable that is larger than all the labels it has. The number of bits required to represent an epoch label depends on m, the maximal size of the set, and it is in $O(m \log m)$. We ensure that the size of the set is proportional to the total capacity of the communication links, namely, $O(cn^2)$, where c is the bound on the capacity of each link, and hence, each epoch label requires $O((cn^2(\log n + \log c))$ bits.

It is possible to reduce this complexity, making c essentially constant, by employing a data-link protocol for communication among the processors.

We show that, after a bounded number of write operations, the results of reads and writes can be linearized in a manner that satisfies the semantics of a SWMR register. This holds until the sequence numbers wrap around, as can happen in a realistic version of the unbounded ABD simulation.

The labeling scheme. Let $k > 1$ be an integer, and let X be the set $\{1, 2, .., k^2{+}1\}$. $\mathcal{L}$ (the set of labels) is the set of all ordered pairs, (s, A), such that $s \in X$ and $A \subseteq X$ $|\mathcal{L}| = \binom{k^2+1}{k}k^2 + 1 = k^{(1+o(1))k}$. A label (s_i, A_i) is smaller ($\prec$) a label (s_j, A_j) if and only if $(s_j \in A_i)$ and $s_i \notin A_j$.

Given a subset S of at most k labels $(s_1, A_1), (s_2, A_2), \ldots$ in $\mathcal{L}$, we compute a new label (s_i, A_i) which is greater (with respect to $\prec$) than every label of S:

- s_i is an element of X that is not in the union $A_1 \cup A_2 \cup \ldots \cup A_k$, and
- A is a subset of size k of X containing all values $(s_1, s_2, \ldots, s_k)$.

It can be proved [1] that this element exists and that this yields a bounded labeling scheme, even with uninitialized values.

References

1. Alon, N., Attiya, H., Dolev, S., Dubois, S., Gradinariu, M., Tixeuil, S.: Practically Stabilizing Atomic Memory, arXiv 1007.1802 (2010)
2. Attiya, H.: Robust Simulation of Shared Memory: 20 Years After. EATCS Distributed Computing Column (2010)
3. Attiya, H., Bar-Noy, A., Dolev, D.: Sharing Memory Robustly in Message-Passing Systems. J. ACM 42(1), 124–142 (1995)
4. Dolev, S.: Self-Stabilization. MIT Press, Cambridge (2000)
5. Dolev, D., Shavit, N.: Bounded Concurrent timestamping. SIAM J. on Computing 26(2), 418–455 (1997)
6. Israeli, A., Li, M.: Bonded timestamps. Distr. Comp. 6(4), 205–209 (1993)