Brief Announcement: Dynamic FTSS in Asynchronous Systems: The Case of Unison*

Swan Dubois, Maria Gradinariu Potop-Butucaru, and Sébastien Tixeuil

LIP6 - UMR 7606, Université Pierre et Marie Curie - Paris 6/INRIA

Context. The advent of ubiquitous large-scale distributed systems advocates that tolerance to various kinds of faults and hazards must be included from the very early design of such systems. *Self-stabilization* [1] is a versatile technique that permits forward recovery from any kind of *transient* fault, while *Fault-tolerance* [2] is traditionally used to mask the effect of a limited number of *permanent* faults. The seminal works of [3,4] define *FTSS* protocols as protocols that are both *Fault Tolerant and Self-Stabilizing*, *i.e.* able to tolerate a few crash faults as well as arbitrary initial memory corruption. In [3], some impossibility results in asynchronous systems are presented. In [4], a general transformer is presented for synchronous systems. The transformer of [4] was proved impossible to transpose to asynchronous systems in [5] due to the impossibility of tight synchronization in the FTSS context. It turns out that FTSS possibility results in fully *asynchronous* systems known to date are restricted to *static* tasks, *i.e.* tasks that require eventual convergence to some global fixed point (tasks such as naming or vertex coloring fall in this category).

In this work, we consider the more challenging problem of *dynamic* tasks, *i.e.* tasks that require both eventual safety and liveness properties (examples of such tasks are clock synchronization and token passing). Due to the aforementioned impossibility of tight clock synchronization, we consider the *unison* problem, that can bee seen as a *local* clock synchronization problem. In the unison problem [6], each node is expected to keep its digital clock value within one time unit of every of its neighbors' clock values (weak synchronization), and increment its clock value infinitely often. Note that in synchronous completely connected systems where clocks have discrete time unit values, unison induces tight clock synchronization. Several self-stabilizing solutions exist for this problem, both in synchronous and asynchronous systems, yet none of those can tolerate crash faults. As a matter of fact, there exists a number of FTSS results for *dynamic* tasks in *synchronous* systems. In particular, clock synchronization is well-studied. The reader can find more references in [7].

Contributions. In this work, we tackle the open issue of FTSS solutions to *dynamic* tasks in *asynchronous* systems, using the unison problem as a case study.

Our first negative results show that whenever two or more crash faults may occur, FTSS unison is impossible in any asynchronous setting. The remaining case of one crash fault drives the most interesting results.

* This work was funded in part by ANR project SHAMAN.

I. Keidar (Ed.): DISC 2009, LNCS 5805, pp. 291–293, 2009.

We first extract two key properties satisfied by all previous self-stabilizing asynchronous unison protocols: *minimality* and *priority*. Minimality means that nodes maintain no extra variables but the digital clock value. Priority means that whenever incrementing the clock value does not break the local safety predicate between neighbors, the clock value is actually incremented in a finite number of activations, even when no neighbor modifies its clock value.

Then, depending on the fairness properties of the scheduling of nodes, we provide various results with respect to the possibility or impossibility of unison. When the scheduling is *unfair* (only global progress is guaranteed), FTSS unison is impossible. When the scheduling is *weakly fair* (a processor that is continuously enabled is eventually activated), then it is impossible to solve FTSS unison by a protocol that satisfies either minimality or priority. The case of *strongly fair* scheduling (a processor that is activated infinitely often is eventually activated) is similar whenever the maximum degree of the graph is at least three. Our negative results still apply when the clock variable is unbounded and the scheduling is central (*i.e.* a single processor is activated at any time).

On the positive side, we propose a FTSS protocol for connected networks of maximum degree at most two (*i.e.* rings and chains), that satisfies both minimality and priority properties. This protocol makes minimal system hypotheses with respect to the aforementioned impossibility results (maximum degree, scheduling, etc.) and is optimal with respect to the containment radius that is achieved (*no* correct processor is *ever* prevented from incrementing its clock).

The table above provides a summary of the main results of the work. More details about this work are available in [7].

	Unfair	Weakly fair		Strongly fair		
		Minimal	Priority	$\Delta \geq 3$		$\Delta \leq 2$
				Minimal	Priority	
$f = 1$	Imp.	Imp.	Imp.	Imp.	Imp.	Pos.
$f \geq 2$	Imp.					

Perspectives. Future works follow: fix the remaining open cases, give results with bounded clocks, and deal with malicious nodes instead of crashes.

References

1. Dijkstra, E.W.: Self-stabilizing systems in spite of distributed control. Commun. ACM 17, 643–644 (1974)
2. Fischer, M.J., Lynch, N.A., Paterson, M.: Impossibility of distributed consensus with one faulty process. J. ACM 32(2), 374–382 (1985)
3. Anagnostou, E., Hadzilacos, V.: Tolerating transient and permanent failures (extended abstract). In: WDAG, pp. 174–188 (1993)

4. Gopal, A.S., Perry, K.J.: Unifying self-stabilization and fault-tolerance (preliminary version). In: PODC, pp. 195–206 (1993)
5. Beauquier, J., Kekkonen-Moneta, S.: Fault-tolerance and self stabilization: impossibility results and solutions using self-stabilizing failure detectors. Int. J. Systems Science 28, 1177–1187 (1997)
6. Misra, J.: Phase synchronization. Inf. Process. Lett. 38(2), 101–105 (1991)
7. Dubois, S., Potop Butucaru, M., Tixeuil, S.: Dynamic FTSS in Asynchronous Systems: the Case of Unison. Research Report arXiv:0904.4615, INRIA (April 2009)