k-Stabilization of Reactive Tasks

Joffroy Beauquier* Christophe Genolini† Shay Kutten ‡

Intuitively speaking, traditional fault tolerance methods were global in nature. For example, the *reset* approach (e.g. [1, 2, 3]) is to bring all the nodes into some predefined state. Such an approach is becoming less and less reasonable in modern networks, since these are much larger than traditional ones, and are growing fast. Thus it was suggested in [4] that protocols can scale to handle larger networks if the smaller is the number of faults, the shorter is the recovery time. Such protocols are called *fault local* [6]. In [4, 5, 6] it is shown how to do that for various cases of *non-reactive* problems

We study the scenario where transient faults hit up to k (for a given k) nodes in a *reactive asynchronous* distributed system by corrupting their state undetectably. (The exact number of nodes, the specific nodes the faults hit, and the time they occur, if at all, are *not* known.) We concentrate on the standard benchmark problem for reactive systems- *token passing*, and we treat the more realistic case, where a node P that holds the token must finish some task (often termed the *critical section* of its program, a section that is outside of our algorithm) before forwarding the token. Thus no other node can guess the duration of the time that P holds the token.

We present two algorithms that stabilize into a legitimate configuration (in which exactly one node has a token) in time that depends only on k, and not on n (the number of nodes). One of the algorithms stabilizes in $O(k)$ time, and is, thus, time optimal. The other stabilizes in $O(k^2)$ time, but uses only a constant number of (logarithmic size) variables per node. In terms of the number of individual nodes' steps the stabilization takes $O(kn)$ steps, and it is shown that any 1-stabilizing algorithm (that is, when $k = 1$) must use at least $n - 3$ steps.

The protocols are similar (one uses memory to speed up the other). Both assume that k is smaller than $\sqrt{n}$. For the case that k is larger they have a simple extension that makes them self stabilize.

*LRI- Universite Paris Sud, Batiment 490, F91405 ORSAY Cedex, France, Joffroy.Beauquier@lri.fr

†LRI- Universite Paris Sud, Batiment 490, F91405 ORSAY Cedex, France, Christophe.Genolini@lri.fr

‡Dept. of Industrial Engineering, The Technion, and IBM T.J. Watson Research Center, kutten@ie.technion.ac.il

References

[1] Y. Afek, S. Kutten and M. Yung. Local Detection for Global Self Stabilization, *Theoretical Computer Science*, No 186, pp. 199-229. 1997.

[2] B. Awerbuch, B. Patt-Shamir, G. Varghese, and S. Dolev. Self-stabilization by local checking and global reset. In *Proc. 8th International Workshop on Distributed Algorithms*,

[3] S. Dolev and T. Herman. Superstabilizing protocols for dynamic distributed systems. In *Proc. of the Second Workshop on Self-Stabilizing Systems*, pages 3.1–3.15, May 1995.

[4] S. Kutten and D. Peleg. Fault-local distributed mending. In *Proceedings of the 14th Annual ACM Symposium on Principles of Distributed Computing*, Aug. 1995.

[5] S. Kutten and D. peleg. Tight Fault Locality. In *36th Annual IEEE Symposium on Foundations of Computer Science*. Milwaukee, WI, USA, October 1995.

[6] S. Kutten and B. Patt-Shamir. Time-adaptive self-stabilization. In *Proceedings of the 16th Annual ACM Symposium on Principles of Distributed Computing*, pages 149–158, Aug. 1997.