

Self-Stabilizing Spanning Tree Algorithm for Large Scale Systems *

Thomas Herault, Pierre Lemarinier, Olivier Peres,
Laurence Pilard, and Joffroy Beauquier

LRI bat 490
Universite Paris-Sud
91405 Orsay Cedex
France

{herault,lemarini,peres,pilard,jb}@lri.fr

Abstract. We introduce a self-stabilizing algorithm that builds and maintains a spanning tree topology on any large scale system. We assume that the existing topology is a complete graph and that nodes may arrive or leave at any time. To cope with the large number of processes of a grid or a peer to peer system, we limit the memory usage of each process to a small constant number of variables, combining this with previous results concerning failure detectors and resource discovery. **Keywords:** Distributed Algorithm, Large Scale Systems, Self-Stabilization, Spanning Tree Construction, Failure Detectors.

1 Introduction

Peer to peer networks and grids are emerging large scale systems that gather thousands of nodes. These networks usually rely on IP to communicate: each node has a unique address used by other nodes to communicate with it and every node can communicate with every other node provided it knows its address. In such a system, it is not practical or even not possible for any one node to know the whole list of its neighbors because of its size and also because of the occurrence of failures.

Classical distributed applications, however, need a notion of neighborhood. In a large scale systems, it is generally given by an overlay network built by a specific algorithm. To account for this, we propose to abstract out these requirements using theoretical devices that have to be implemented in a system-specific way.

An algorithm for such a system also needs to tolerate failures. We demonstrated how to use self-stabilization [2] in order to build a bounded-degree spanning tree [3]. We claim that self-stabilization is appropriate for the purpose of building an overlay network because it allows the system to recover from any perturbation affecting either a link or a local variable. It then verifies its specification until the next failure.

* This work is partially funded by the PCRI/INRIA Futurs - Project Grand-Large and ACI Grid (French incentive)

2 Contributions

We introduce [3] a self-stabilizing algorithm that builds a bounded-degree spanning tree over a virtual complete graph. The nodes only store the identifiers of their neighbors and only rely on the devices provided by our model to establish and maintain the overlay network. Each node only has a constant number of local variables.

Our first contribution is a new model for distributed algorithms. The main advantage that we claim for it is that it allows to run distributed algorithms in real-world large scale systems. To achieve this, we abandon the notion of a system-provided, automatically updated neighbor list found in most existing works and replace it with two theoretical devices: an oracle for resource discovery and a failure detector to deal with possible identifiers of stopped (crashed) processes.

The oracle, when queried, replies with a valid process identifier which may, or not, be that of a process in the system. For our spanning tree algorithm, the only requirement is to give the identifier of the highest process an infinite number of times over any infinite number of queries.

The failure detector follows Chandra and Toueg's definition [1]. We proved that in our case, we need a $\diamond\mathcal{P}$ detector, i.e. one that is eventually perfect.

Our second contribution is the algorithm itself. Each process has δ neighbor fields, where δ is a user-provided integer constant. The algorithm is given [3] as a set of guarded rules that eliminate inconsistent configurations, build the tree and maintain it. We provide a formal proof of its correctness.

To guarantee that the tree is correctly built, the algorithm enforces a global invariant: each process only accepts as a child a process whose identifier is lower than its own identifier. Only the roots attempt to connect the topology, thus only them query their oracles, which allows to design an efficient implementation. Eventually there is a single root, so only one process queries its oracle. Finally, when the system is converged, the algorithm only induces a very low overhead.

We implemented the algorithm and the two devices on which it depends and measured its performances on the Grid Explorer high-performance experimental cluster, comparing several approaches in places where the specification leaves room for choices. This allowed us to show that the system displays the expected scalability.

References

1. T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43, March 1996.
2. E. Dijkstra. Self stabilizing systems in spite of distributed control. *Communications of the Association of the Computing Machinery*, 17(11):643–644, 1974.
3. T. Herault, P. Lemarinier, O. Peres, L. Pilard, and J. Beauquier. Self-stabilizing spanning tree algorithm for large scale systems. Technical Report 1457, LRI, 2006.