Empire of Colonies

Self-stabilizing and Self-organizing Distributed Algorithms

(Extended Abstract)

Shlomi Dolev and Nir Tzachar

Department of Computer Science, Ben-Gurion University of the Negev, Israel
dolev@cs.bgu.ac.il, tzachar@cs.bgu.ac.il

Abstract. Self-stabilization ensures automatic recovery from an arbitrary state; we define self-organization as a property of algorithms which display local attributes. More precisely, we say that an algorithm is self-organizing if (1) it converges in sublinear time and (2) reacts “fast” to topology changes. If \( s(n) \) is an upper bound on the convergence time and \( d(n) \) is an upper bound on the convergence time following a topology change, then \( s(n) \in o(n) \) and \( d(n) \in o(s(n)) \). The self-organization property can then be used for gaining, in sublinear time, global properties and reaction to changes. We present self-stabilizing and self-organizing algorithms for many distributed algorithms, including distributed snapshot and leader election.

We present a new randomized self-stabilizing distributed algorithm for cluster definition in communication graphs of bounded degree processors. These graphs reflect sensor networks deployment. The algorithm converges in \( O(\log n) \) expected number of rounds, handles dynamic changes locally and is, therefore, self-organizing. Applying the clustering algorithm to specific classes of communication graphs, in \( O(\log n) \) levels, using an overlay network abstraction, results in a self-stabilizing and self-organizing distributed algorithm for hierarchy definition.

Given the obtained hierarchy definition, we present an algorithm for hierarchical distributed snapshot. The algorithms are based on a new basic snap-stabilizing snapshot algorithm, designed for message passing systems in which a distributed spanning tree is defined and in which processors communicate using bounded links capacity. The combination of the self-stabilizing and self-organizing distributed hierarchy construction and the snapshot algorithm form an efficient self-stabilizer transformer. Given a distributed algorithm for a specific task, we are able to convert the algorithm into a self-stabilizing algorithm for the same task with an expected convergence time of \( O(\log^2 n) \) rounds.

1 Introduction

The availability and robustness, as well as the possibility for on-demand reconfiguration of large systems, are in many cases vital; be it clusters of servers that support commercial activity, a grid of computers that participate in a complicated computation or a

* Partially supported by IBM, Israeli ministry of science, Deutsche Telekom, Rita Altura Trust Chair in Computer Sciences, Israeli Grid Consortium and the Lynn and William Frankel Center for Computer Sciences.

© Springer-Verlag Berlin Heidelberg 2006
dynamic sensor network. In particular, an important aspect for large on-going systems is the ability to automatically recover from an inconsistent state, namely to be self-stabilizing \((11)\) or in other words, to have a system that can be started in an arbitrary state.

To capture the need of the industry in autonomic and self-* systems, we propose combining self-stabilization (in fact SuperStabilization \([12]\)) with self-organization. While self-stabilization is well defined, the self-organization property has no widely agreed upon definition. We propose to define self-organization as satisfying two main properties: locality and dynamicity. Namely, we require that (1) the algorithm stabilizes in sublinear time with regards to the number of processors and that (2) the addition and removal of processors influences a small number of other processors’ states. In other words, if \(s(n)\) represents the stabilization time and \(d(n)\) represents an upper bound on the stabilization time (and number of state changes) following a dynamic topology change, then: \(s(n) \in o(n)\) and \(d(n) \in o(s(n))\).

In this work, we enable algorithms to define (on the fly) and then use hyper communication links, which are overlay links that are constructed of communication links along a path. We regard the time that a message travels over such a link as one time unit, as (almost) no processing is involved in forwarding messages over these links (e.g., \([13, 26]\), MPLS \([6]\)).

**Main Contribution.** We define the self-organization property to capture locality and dynamicity. We present a clustering algorithm (in fact, a distributed maximal independent set algorithm) which is both self-stabilizing and self-organizing. To realize the clustering algorithm in an asynchronous system we present a scheme of local synchronization, achieved by using a local snapshot protocol. We employ the aforementioned clustering algorithm to define a graph hierarchy which can be used to convert any distributed task to be self-stabilizing incurring only a sublinear time overhead.

- **Self-Stabilizing and Self-Organizing hierarchy definition.** The hierarchy of subsystems is defined by partitioning the communication graph into small clusters, after which clusters are merged to form bigger clusters and so on. The partition can be done according to a designer’s input, using an automatic off-line clustering algorithm or even an on-line clustering algorithm that reflects the system’s current behavior. In particular, we suggest a randomized self-stabilizing and self-organizing partition that is based on periodical collection of local topology (up to a certain distance). The collected local topology supports a randomized local leader election, in which a non leader processor that does not identify a leader within a certain distance \(x\) tries to convert itself to a leader. Leaders within distance \(x\) from each other are eliminated, until there are no leaders that are within distance \(x\) or less from each other. Higher level partitions, using larger distances and overlay network abstraction between leaders, are constructed in a similar way.

In asynchronous systems, our clustering algorithm uses (for each processor) a (local) self-stabilizing snapshot algorithm for obtaining local synchronization of actions.

- **Self-Stabilizing snapshots.** We present a snap-stabilizing (e.g., \([7]\)) snapshot algorithm for distributed systems, that uses message passing with bounded link capacity, in which a spanning tree is distributively defined. Our snapshot algorithm is designed for