Self-Stabilization by Local Checking and Correction

by

George Varghese

B.Tech, Electrical Engineering
Indian Institute of Technology, Bombay
(1981)
M.S., Computer Studies
North Carolina State University
(1983)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

October 1992

© Massachusetts Institute of Technology 1992

Signature of Author .....
Department of Electrical Engineering and Computer Science
October 20, 1992

Certified by .....
Baruch Awerbuch
Associate Professor
Thesis Supervisor

Certified by .....
Nancy A. Lynch
Professor
Thesis Co-supervisor

Accepted by .....
Campbell L. Searle
Chairman, Departmental Committee on Graduate Students

Self-Stabilization by Local Checking and Correction

by

George Varghese

Submitted to the Department of Electrical Engineering and Computer Science
on October 20, 1992, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

A *self-stabilizing protocol* is one that begins to behave correctly in bounded time, no matter what state the protocol is started in. Self-stabilization abstracts the ability of a protocol to tolerate *arbitrary* faults that stop. We investigate the power and applicability of *local checking and correction* for the design of stabilizing network protocols.

A link subsystem is a pair of neighboring nodes and the two links between them. Intuitively, a protocol P is *locally checkable* if whenever P is in a bad state, some link subsystem is also in a bad state. A protocol P is *locally correctable* if P can be corrected to a good state by locally correcting link subsystems.

We present four general techniques for designing stabilizing protocols. We first show that every locally checkable and correctable protocol can be stabilized in time proportional to the height of an underlying partial order. Second, we show that every locally checkable protocol on a tree can be stabilized in time proportional to the height of the tree. Third, we show that every locally checkable protocol can be stabilized in time proportional to the number of network nodes. The third result shows that we can dispense with the need for local correctability or the need for the underlying topology to be a tree as long as we are willing to pay a higher price in stabilization time. Fourth, we show that any deterministic synchronous protocol π can be converted to an asynchronous, stabilizing version of π. The fourth technique is useful because there are network tasks for which a synchronous protocol exists but for which no asynchronous, locally checkable solution is known.

We also present two useful heuristics. The first heuristic, that of *removing unexpected packet transitions*, can often be used to transform a protocol into a locally checkable equivalent. A number of existing protocols work in a *dynamic network* model where links can fail and recover. The second heuristic states that locally checkable protocols for dynamic networks can sometimes be made locally correctable. The basic idea is to use the link failure and recovery actions of the original protocol to locally correct link subsystems.

Together our techniques cover a broad range of networking tasks. We use our general techniques to construct new or improved stabilizing solutions to many specific for Mutual Exclusion, Network Resets, Spanning Trees, Topology Update, Min Cost Flows etc. Many of our solutions are practical and can be applied to real networks without appreciable loss in efficiency. For example, the messages required for local checking can easily be piggybacked on the "keep-alive" traffic sent between neighbors in real networks.

Our techniques also help in succinctly understanding existing stabilizing protocols. We

define a special case of local checking called *one-way checking*. We show that many existing protocols implicitly use one-way checking together with two other methods that we call *counter flushing* and *timer flushing*.

In the past, papers on stabilization have avoided message passing models of communication because of the problems caused by unbounded storage Data links. In a stabilizing setting, such links can be initialized with an unbounded number of fictitious packets. Thus almost any non-trivial network task is impossible in a stabilizing setting in which the links have unbounded storage and the nodes are restricted to be finite state machines. We avoid this problem by using the standard asynchronous message passing model of a computer network except that each link is what we call a Unit Storage Data Link (UDL) that can store at most one packet. Our UDL model can be implemented over real physical channels. Our UDL model also generalizes easily to a Bounded Storage Data Link which can store a constant number of packets.

We introduce a new definition of stabilization in terms of the external behavior of a system. The definition allows us to define that an automaton A stabilizes to another automaton B even though A and B have different state sets. The definition also allows a clean statement of a useful *Modularity Theorem*. This theorem allows us to prove that a large system is stabilizing by proving that each of its pieces is stabilizing.

Keywords: Self-Stabilization, Fault-Tolerance, Network Protocols, Distributed Algorithms, Local Checking and Correction.

Thesis Supervisor: Baruch Awerbuch
Title: Associate Professor

Thesis Co-supervisor: Nancy A. Lynch
Title: Professor

Acknowledgments

I have been fortunate to be helped by a huge number of talented people. Basically, I have had two sets of advisors – one set from DEC where I work, and another set from MIT.

I thank Radia Perlman for being my advisor at DEC. I owe so much to her. My interest in the field of self-stabilization began after watching Radia design two elegant and widely used stabilizing protocols. After Radia went back to MIT to complete her doctorate, she encouraged me to do the same. She plotted my course load, provided feedback, directed my application process, cajoled, consoled, and occasionally scolded. Recently, she was selected by Data Communications Magazine as one of twenty people who have changed the field of networking. But to me she is something much better, a good and loyal friend.

I thank Alan Kirby of DEC for being a wonderful manager and boss. By trusting and believing in me, he has helped me to do things that I did not think I was capable of. Besides being a fine manager, Alan also has a great deal of technical vision. All the projects he has assigned me to work on have provided seed ideas for this thesis.

I thank Baruch Awerbuch of MIT who supervised my thesis together with Nancy Lynch. Baruch is a restless visionary who is constantly conquering new areas. This thesis began with Baruch's conviction that the so called end-to-end problem had a stabilizing solution. That problem led us to invent local checking and correction. Still later, Baruch was convinced that it was "easy" to construct an efficient and stabilizing reset protocol. It is typical of his approach that he began with problems that other people (especially me!) considered hard.

I thank Nancy Lynch of MIT for being a superb thesis supervisor and advisor. Nancy had to struggle with initial drafts of this thesis that were truly terrible. Under her guidance, the formalism gradually emerged. Her idea of using a specification of stabilization based on external behavior has been particularly fruitful. Working with her has been a true education, as I have begun to learn a little of what scholarship and rigor means.

I thank Robert Gallager of MIT for his inspiration, and his patience in reading this thesis. His insistence on clarity has shamed me into rewriting parts of the thesis. I will always remember the kindness which he, a great and brilliant man, shows to his students.

I thank Mark Tuttle of DEC Cambridge Research Lab for his painstaking reading of this thesis. Time and again, Mark spotted errors and helped clean up important definitions. Even

his most critical comments were leavened with humor and encouragement. If this thesis is reasonably free of errors, it is because of the hard work of Nancy and Mark. Mark also helped me to understand the big picture; the introduction and Chapter 10 have improved because of his efforts.

I thank Boaz Patt who worked together with Baruch and myself on many of the ideas in this thesis. I thank Boaz especially for his hard work on an initial version of the reset protocol, and for initiating the use of an external behavior specification for the reset protocol. I owe Boaz many things, including the succinct title of this thesis.

I thank the DEC Graduate Engineering Program (GEEP) for their incredible generosity in allowing me to go to MIT on full salary for two years. I especially thank Shirley Stahl and Terry Sarandrea of GEEP for their ability to cut through administrative procedures and to help me when I needed help. I also thank Joanne Talbot at MIT for her administrative support at MIT.

This thesis is dedicated to my father, who would have loved to leaf through it if he were alive; and to my mother and sister, for their patience and love.

Finally, I'd like to thank God and his son Jesus, from whom I believe all good gifts come – even this wonderful set of people that it has been such a joy to live and work with.