

Toward Self-Stabilizing Operating Systems*

Shlomi Dolev Reuven Yagel

The robustness of an operating system is, in some cases, more important than its performance [4, 3]. The experience with existing operating systems, and in fact with every large on-going software package, is that it almost has its own independent behavior. The behavior is tuned up and modified by system administrators that constantly and continuously monitor it. The system is usually complicated to monitor. The system administrators use human behavior and character terms, as if the system is an entity with its own will, to refer to its input output scenarios. The importance of a design that is based on well understood theoretical paradigms, and give us control over the resulting system cannot be exaggerated. In particular in the case of the operating system, robustness is a must, as the operating system forms a basic infrastructure in almost every computing system.

Designing robust operating system is a complicated and challenging task. The system designer makes several probabilistic assumptions that may not hold in a long enough execution. For example, soft errors [2] may cause an arbitrary change in memory bits that the error correcting schemes used will not identify. Another example, is that the communication between the system components can be made reliable, say by use of error correcting codes this assumption is also based on probability (where the life length of the system is a parameter). Once the probabilistic assumptions do not hold the designer can no longer guarantee much. In this work we propose several approaches for designing automatic recovering operating system that is based on the well defined and well understood self-stabilization paradigm [1]. Roughly speaking, a system is *self-stabilizing* if it can be started in any possible state and converge to a desired behavior. A *state* of a system is an assignment of arbitrary values to the systems variables.

A self-stabilizing algorithm/system makes the obvious assumption that it is executed. This assumption is not simple to achieve since both the microprocessor [2] and the operating system should be self-stabilizing, ensuring that even-

tually the (self-stabilizing) applications programs are executed. An elegant composition technique of self-stabilizing algorithms [1] is used here to show that once the underlying microprocessor stabilizes the self-stabilizing operating system (which can be started in arbitrary state) stabilizes, and then the self-stabilizing algorithms that implement the applications stabilize. In this work we consider the important layer of the operating system. Operating systems are essential parts of most computer systems. The operating system manages the hardware resources, and form an abstract (virtual) machine that is convenient to program by higher level applications developers.

One approach in designing self-stabilizing operating system is to consider an existing operating system (e.g., Microsoft Windows, Linux) as a black-box and add components to monitor its activity and take actions accordingly, such that automatic recovery is achieved. We call this approach the black-box based approach. The other extreme approach is to write a self-stabilizing operating system from scratch. We call this approach the tailored solution approach. We present three design solutions in the scale of the black-box to the tailored solutions. The first simplest technique we propose for automatically recovery of an operating system is based on repeatedly reinstalling the operating system and then re-execution. The second technique is to repeatedly reinstall only the executable portion, monitoring the state (variables content) of the operating system and assigning a legitimate state whenever required. Then we present a tailored very tiny self-stabilizing design for components of an operating system.

References

- [1] Shlomi Dolev, *Self-Stabilization*, The MIT Press, 2000.
- [2] Shlomi Dolev and Yinnon Haviv, "Self-Stabilizing Soft-Errors Resilient Microprocessor", IEEE DSN 2003, pp. B18-B19, 2003.
- [3] A. Fox and D. Patterson. "Self-Repairing Computers", *Scientific American*, June, 2003
- [4] J. O. Kephart and D. M. Chess, "The Vision of Autonomic Computing", *IEEE Computer*, 41-50, January, 2003.

* Department of Computer Science, Ben-Gurion University, Beer-Sheva, 84105, Israel, {dolev,yagel}@cs.bgu.ac.il. Reuven is also with Rafael WTEC, Mizpe-Ramon. Partially supported by Microsoft, MFAT, IBM, NSF, STRIMM, and Rita Altura Trust Chair in Computer Sciences.