Journal of Automata, Languages and Combinatorics **14** (2009) 1, 63–73
© Otto-von-Guericke-Universität Magdeburg

SOFTWARE REJUVENATION IN EMBEDDED SYSTEMS [1]

CHANDRA M. KINTALA
Yahoo! Labs
Bangalore, India
e-mail: kintala@yahoo-inc.com

ABSTRACT

Mobile communication devices have multitasking embedded software running in their operating systems (OS) as well as applications. Both the OS modules and the application components are assigned predetermined memory in those devices due to near-realtime performance requirements and lack of virtual memory. Memory (stack and heap) overflow problems occur in such software components because of programmer's inability to accurately estimate the size of the needed memory and also because of Mandelbugs due to their complexity. A new software rejuvenation variation called Opportunistic Micro Rejuvenation (OMR) is proposed where a task that "misbehaves" is identified and rejuvenated at an opportune instant. Following the traditional model for software rejuvenation, a Stochastic Area Network model is used to simulate and study the availability of such embedded systems with OMR. It is shown that OMR alone will increase the availability only slightly but when combined with other memory management techniques, such as shared supplementary memory, OMR will increase the availability significantly.

Keywords: Stack overflow, heap overflow, embedded systems, software rejuvenation, resource constrained fault-tolerance

1. Introduction

Mobile communication devices and other similar systems are becoming increasingly ubiquitous giving rise to large number of applications and services embedded in them. Such applications have near-real-time performance needs and place high demands on the networking and multi-tasking capabilities. Fast market growth of those devices, short time-to-market pressures and high performance needs have led to increased software complexity. Most embedded systems, as much as 95 % according to Middha et al. [9], do not use virtual memory and hence programmers are expected to estimate their run-time memory (stack and heap) needs for their operating systems as well as applications. Estimating run-time memory requirements accurately is significantly hard [1] due to increased complexity of the OS and application software in embedded

[1] This paper is based on research done with colleagues when the author was at Motorola Labs in Bangalore. That research was first reported in a conference paper [13] containing some other results as well. This is an expanded version of the relevant portions of that conference paper.