A Model for Updating Real-Time Applications

JAMI MONTGOMERY  jamimontgomery@lucent.com and jmontgomery@cti.depaul.edu
Lucent Technologies, 2701 Lucent Lane, RM 7G506, Lisle, IL 60532, USA

Abstract. Updating application software is a common occurrence for modern computing systems. Software updates stem from the need to correct coding errors or to enhance the functionality of an application. Updating an application typically requires taking the current application offline and restarting a new application. This method of updating an application is perfectly acceptable for many general purpose-computing environments. However, in real-time environments that require high availability and have stringent timing constraints, taking a process offline for updates may be unacceptable or pose unnecessary risks. Some examples of these environments include telecommunications, air traffic control, railway control and medical patient monitoring. We present a new method to dynamically update a real-time application without having to take it offline. Our new method, which we call dynamic update for real-time systems, can be used to update real-time applications using rate-monotonic scheduling, while preserving the original deadline guarantees.

Keywords: dynamic update, code replacement, multithreaded, RT-Mach, software update

1. Introduction

Software applications typically require coding changes during their useful life span. The coding changes can stem from a desire to enhance an application or to correct a coding error. In such cases, applying code changes to an application usually requires taking the application off-line. Taking an application off-line to apply code changes (software updates) is an accepted practice for many applications. However, when the application is managing mission-critical or safety-critical applications, taking these systems off-line to apply software updates may pose unacceptable risks or jeopardize human life. An alternative method of updating an application is needed; one that does not require taking the running application off-line. We present a novel method of dynamically updating executing applications, one that is sensitive to the constraints of real-time environments. Our method, which we call dynamic update for real-time systems (DURTS), provides a method to update an executing process without having to take it off-line. DURTS does not require adopting new software development environments; it does not rely on custom execution environments or special operating system services; and it can be incorporated in new or existing applications. Using DURTS does require the user to add a user-written module to load and synchronize the new replacement modules and it also requires the use of our pseudo-linker to create the replacement module for the real-time application.

Hardware and software redundancy is one approach that has been used to reconfigure systems on-line. However, this approach significantly increases the cost of developing and deploying a real-time system. The increase in cost is attributed to the additional or redundant hardware and the increase in software complexity to hot-switch from a standby
to active mode. Examples of these systems can be found in Hecht (1976) and Weinstock and Green (1978).

There has been significant interest in the area of dynamically updating an executing process. The solutions presented so far rely on special operating systems services (Stewart et al., 1992; Goyeneche and Sousa, 1999), custom execution environments (Seto et al., 1998; Frieder and Segal, 1991; Hofmeister, 1994; Kramer and MacGee, 1985), non-standard development tools (Ssu and Jiau, 2000) or complex software development paradigms. All of these methods, while valuable, placed considerable burden on designers of real-time applications who sought dynamic update capabilities for real-time systems. A method to dynamically update real-time applications that does not rely on special operating system services, custom executing environments or software development tools is needed. In this article we present DURTS, a dynamic update system that can be used to dynamically update a real-time process without taking the process off-line. More importantly, DURTS does not require a custom execution environment or complex software development paradigms. DURTS is a “lightweight” update method that can be used with new applications or added to existing systems. For real-time applications that are scheduled using the rate-monotonic scheduling algorithm, a simple calculation can determine if a real-time process can be updated while preserving the original deadline guarantees. Our direct contributions include a pseudo-linker, two new operating system primitives and a method to dynamically load a new replacement module into an executing process.

The remainder of this paper is organized as follows. In Section 2, we provide a background on previous research in dynamic update systems. Section 3 describes our new dynamic update system for real-time processes, which builds on our previous research (Montgomery et al., 2001) and includes sections on how to create a replacement module (3.1), how to load a replacement module (3.2), and how to transfer control to a newly loaded replacement module (3.3). Section 4 discusses the timing implications of using DURTS within a real-time environment and gives some empirical timing results. Section 5 provides a summary and Section 6 discusses future directions.

2. Background

Dynamically updating a process is the act of changing the code executed by a process without taking the process off-line. The ability to dynamically update an executing process has enormous benefits in environments that have high availability requirements or long-running processes. These types of processes can be found in mission-critical and safety-critical systems. Examples of these systems include air traffic control systems (Davis et al., 1994), railroad train control systems (Egan et al., 1999), industrial control systems and patient monitoring systems (Kairouz et al., 1994). Providing a method to dynamically update a process is a favorable alternative to the traditional terminate-load-restart method (TLR), which violates the timing constraints of real-time systems.

Previous research into dynamic update systems has produced several notable models. CONIC, developed by Kramer and Magee (1985), was a comprehensive reconfiguration system used to dynamically reconfigure large or distributed applications. In CONIC,