Scheduling Real-Time Tasks in the Presence of Dynamic Software Updates

Dong Kwan Kim

Department of Multimedia Engineering,
Catholic University of Pusan,
Pusan, 609-757, Korea
dongkwan@cup.ac.kr

Abstract. Although dynamic software updating has addressed various problems on software maintenance, there are no noticeable outcomes in applying it to real-time systems. In this paper, a novel EDF-based scheduling algorithm is presented to update real-time systems dynamically while preserving the schedulability of tasks and satisfying precedence constraints intrinsically imposed between the updated and updating tasks.

Keywords: Dynamic Software Updates, Real-time Systems, Earliest Deadline First (EDF), Total Bandwidth Server (TBS).

1 Introduction

Dynamic Software Update (DSU) [1] changes the code of a computer program while it runs, thus saving the programmer's time and using computing resources more productively. This paper applies DSU to real-time applications-a computing domain characterized by long-running operations [2] and high reliability.

For safe updates, an object to be updated needs to reach a quiescent point where updating systems block all other objects from accessing it during updates [1]. Since the updated object remains suspended during updates, it cannot restart the execution until the dynamic update task has completed. Therefore, a scheduling algorithm considering dynamic updates should ensure precedence constraints between an updating task and an updated task. Fig. 1. shows Earliest Deadline First (EDF) schedule *without* considering the precedence constraint. τ_1 and τ_2 are periodic tasks and a dynamic update task, *dut*, updates τ_2 dynamically. In this example, the EDF schedule violates the following precedence constraint: τ_2 depends on *dut*: $dut \rightarrow \tau_2$, which implies that the pure EDF algorithm is not suitable to the DSU for real-time systems.

To the best of my knowledge, most of dynamic updating systems have been applied to non-real-time systems. A novel EDF-based scheduling algorithm is described and demonstrated to update real-time tasks at runtime.

In this paper, the TB(N) algorithm proposed in [3] is modified to assign the optimal deadline of dynamic update tasks. Furthermore, a real-time scheduling algorithm for DSU is introduced. The algorithm guarantees the schedulability of

G. Lee et al. (Eds.): ICHIT 2012, LNCS 7425, pp. 650–656, 2012.

periodic tasks and satisfies precedence constraints intrinsically imposed between the updated and updating tasks. The performance of the proposed scheduling algorithm is evaluated by comparing with the Rate Monotonic (RM)-based scheduling algorithm for DSU [4] in terms of the response time of a dynamic update task and hard real-time guarantees of all running tasks.

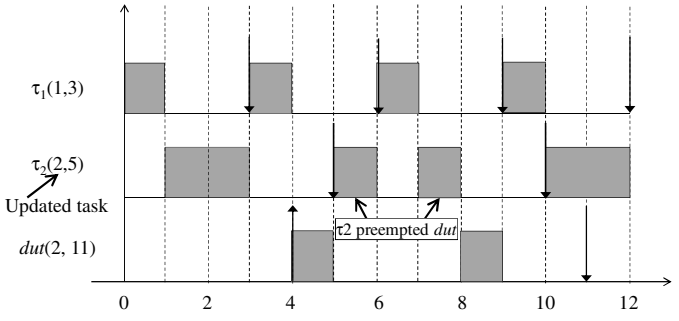

Fig. 1. EDF schedule disregarding precedence constraints

2 System Model

The algorithm considers a set of n preemptable periodic tasks, $\{ \tau_1, \tau_2, \ldots , \tau_n\}$ and m preemptable dynamic update tasks, $\{ J_1, J_2 , \ldots , J_m\}$. The dynamic update tasks can update the periodic tasks at runtime. Each τ_i has a number of invocations, that is, jobs, which are released periodically with hard deadlines and known execution times. Each J_k has a known execution time but does not have an explicit deadline. The instances of the dynamic update task are triggered aperiodically when requested.

The paper introduces a deadline assignment algorithm for dynamic update tasks and an EDF-based preemptable scheduling algorithm for periodic tasks in the presence of dynamic update tasks. The proposed algorithms are based on the following assumptions:

1. All tasks are executed on a uniprocessor system.
2. At $t = 0$, all periodic tasks are activated.
3. Deadline ties are always broken in favor of the dynamic update task.
4. The dynamic update task consists of three steps: suspending an old version of an updated task, replacing the old version with new one, and restarting the new version of the updated task.
5. Dynamic update tasks are processed in a First-In-First-Out order, thus a dynamic update task cannot preempt another dynamic update task.

There are three ways to assign the processor utilization for a dynamic update task, *dut* in order to determine its deadline. In the first approach, the scheduling algorithm assigns the server utilization factor, U_s for *dut*. U_s satisfies the condition $U_p + U_s \leq 1$ where U_p is the utilization factor of the periodic task set.