Incremental Schedulability Analysis of Hierarchical Real-Time Components*

Arvind Easwaran, Insik Shin, Oleg Sokolsky, and Insup Lee
Department of Computer and Information Science
University of Pennsylvania, Philadelphia, PA
{arvinde, ishin, sokolsky, lee}@cis.upenn.edu

ABSTRACT

Embedded systems are complex as a whole but consist of smaller independent modules minimally interacting with each other. This structure makes embedded systems amenable to compositional system design. Compositional design of real-time embedded systems can be done using hierarchical systems which consist of real-time components arranged in a scheduling hierarchy. Each component consists of a real-time workload and a scheduling policy for the workload. To simplify schedulability analysis of hierarchical systems, analysis can be done compositionally using interfaces that abstract the timing requirements of components. Associative composition will facilitate analysis of systems in which components are modified on the fly. In this paper, we propose efficient algorithms to abstract the resource requirements of components in the form of periodic resource models. Each component interface consists of a set of periodic resource models for different values of period, which allows the selection of a periodic interface that minimizes the collective real-time requirements of hierarchical components. We also describe an interface composition algorithm which accounts for context switch overheads incurred by components and is associative.

Categories and Subject Descriptors

C.3 [**Computer Systems Organization**]: Special-Purpose and Application based Systems—*Real-time and embedded systems*; D.2.2 [**Software Engineering**]: Design Tools and Techniques—*Modules and Interfaces*; D.4.1 [**Operating Systems**]: Process Management—*Scheduling*; I.1.2 [**Symbolic and Algebraic Manipulation**]: Algorithms—*Algebraic algorithms*

General Terms

Algorithms, Design, Performance, Theory

Keywords

Associative real-time interfaces, Compositional schedulability analysis, Hierarchical scheduling, Incremental schedulability analysis

1. INTRODUCTION

The increasing complexity of real-time embedded systems demands advanced design and analysis methods for the assurance of timing requirements. Component-based design has been widely accepted as an approach to facilitate the design of complex systems. It provides means for decomposing a complex system into simpler components and for composing them into a system using interfaces that abstract component complexity. To take advantage of component-based design for real-time embedded systems, schedulability analysis should be addressed for component-based real-time systems. It is desirable to achieve schedulability analysis *compositionally*, i.e., to achieve the system-level schedulability analysis by combining component interfaces that abstract component-level timing requirements. Further, the abstracted interfaces must minimize the resource demand of components. In this paper, we consider the periodic resource interface model since it is simple and practical in the sense that many existing real-time schedulers support the periodic model. It is also desirable to achieve incremental schedulability analysis, in particular, for systems with dynamically changing components.

Component-based real-time systems often involve hierarchical scheduling frameworks for supporting hierarchical resource sharing among components under different scheduling policies. The hierarchical framework can be generally represented as a tree of nodes, where each node represents a component consisting of some real-time workload and a scheduling policy, and resources are allocated from a parent to its children. Many studies have been proposed on compositional schedulability analysis for component-based hierarchical scheduling frameworks [22, 15, 23, 1, 4]. However, their approaches do not support incremental schedulability analysis. There have been recent studies [25, 26, 12] that support incremental schedulability analysis using the interface theory [5, 6]. However, these interface theory based techniques cannot be directly applied for the problem of finding the periodic interface, among all possible periodic interfaces with different periods, that minimizes the

*This research was supported in part by NSF CNS-0410662, NSF CNS-0509327, NSF CNS-0509143, ARO W911NF-05-1-0182, and ONR MURI N00014-04-1-0735

EMSOFT'06, October 22–25, 2006, Seoul, Korea.

resource demand of hierarchical components. In this paper, we address this problem taking context switch overhead into consideration.

This paper presents an approach that supports incremental schedulability analysis of component interfaces represented as periodic resource models by allowing associative composition of such interfaces. Our component interface abstracts the minimum resource demand of the underlying component, in the form of periodic resource models for different values of period. The composed interface is generated by adding together resource capacities of individual interfaces for each value of the period. This composition also takes into consideration the context switch overhead incurred by components. Once a single interface is generated for the entire system, the component at the topmost level can select a value for period that minimizes the resource demand of the system. The corresponding periodic resource model is exported to the operating system for scheduling and the chosen value for period is propagated to all the components in the system where resource capacities are given by the corresponding interfaces.

In addition to the interface composition scheme, we also propose efficient algorithms to generate interfaces for components that use rate monotonic (RM) or earliest deadline first (EDF) schedulers [17]. Furthermore, we develop an efficient representation for interfaces and show using examples that the size of these interfaces are small in practice.

Related Work. For real-time systems, there has been a growing attention to hierarchical scheduling frameworks that support hierarchical resource sharing under different scheduling algorithms.

Deng and Liu [7] proposed a two-level real-time scheduling framework for open systems. Kuo and Li [13] presented an exact schedulability condition for such a two level framework with the RM system scheduler and Lipari and Baruah [14, 16] presented similar conditions with the EDF system scheduler. The common assumption shared by these previous approaches is that the system scheduler has a (schedulable) utilization bound of 100%. In open systems, however, it is desirable to be more general since there could be more than two-levels and different schedulers may be used at different levels.

Mok and Feng proposed the bounded-delay resource partition model for a hierarchical scheduling framework [20, 10]. In their framework, a parent component and its child components are separated such that they interact with each other only through their resource partition model. However, they did not consider the component abstraction problem. Shin and Lee [24] addressed the component abstraction problem but their composition technique is not associative.

The periodic resource model has been introduced to specify the periodic resource allocation guarantees provided to a component from its parent component [23, 15]. There have been studies [22, 15, 23, 1, 4] on the component abstraction problem with periodic resource models. For a component with RM scheduler and a periodic resource model abstraction, Saewong *et. al.* [22] introduced an exact schedulability condition based on worst-case response time analysis, and Lipari and Bini [15] presented a similar condition based on time demand calculations. Shin and Lee [23] provided an exact schedulability condition under EDF scheduling. Pedreira [1] and Davis and Burns [4] introduced worst-case response time analysis techniques under RM component-level scheduling, which enhance the previous work. All these abstraction techniques produce periodic interfaces whose composition is not associative. Lipari and Bini [15] describe a technique to compute a set of periodic resource models, that satisfy the schedulability condition under RM scheduler. For each value of resource period, their technique evaluates the schedulability condition over a set of significant points on the time demand function. In this paper, we present an efficient algorithm to compute the periodic interfaces and also describe an associative technique for composing them. For each value of interface period greater than one, this algorithm evaluates the schedulability condition at only two significant points on the time demand function. Lipari and Bini [15] also describe a non-linear optimization technique to minimize the context switch overhead of the component, as well as the resource utilization of the interface. They consider a two level hierarchical framework where the optimization is done locally for each component. In our approach, we account for context switch overheads in the interface capacity and minimize the resource demand at the topmost level of a multi level hierarchical framework.

There have been studies on the development of interface theory for supporting incremental design of component-based real-time systems, applying the interface theories [5, 6] into real-time context. These studies have proposed assume-guarantee interface theories for real-time components with a generic real-time interface model [25, 26] and with a bounded-delay resource partition interface model [12]. These theories can be applied to periodic resource models if the period of the periodic resource abstraction is fixed a priori for all the components in the system to some arbitrary value. In our methodology, the value for the period can be chosen after the interfaces for all the components are generated. Hence the system can select a period that minimizes the resource demand of the system, which is not possible if the period value is fixed a priori.

Matic and Henzinger [19] considered the issue of addressing the component abstraction problem in the presence of task dependences, within and between components, using two approaches; the RTW (Real-Time Workshop) [18] and the LET (Logical Execution Time) [11] semantics. They showed that the previous results [15, 23] can be extended for supporting interacting tasks with data dependencies when one of those two approaches are employed, while both approaches produce a trade-off between the end-to-end latency and the overheads incurred in abstracting components.

The rest of the paper is organized as follows: Section 2 states the compositional schedulability analysis problem and Section 3 briefly describes the techniques developed in this paper. Section 4 develops a supply bound function for periodic resource models in the presence of context switch overheads. Section 5 gives the interface generation algorithm under EDF schedulers and Section 6 describes a similar algorithm for RM schedulers. Section 7 gives the algorithm to compose component interfaces and also describes a mechanism to account for context switch overheads. Section 8 compares the results of this paper with earlier work and Section 9 concludes the paper.

2. PROBLEM STATEMENT

In this paper we assume that each real-time task is an independent periodic task with deadline equal to the period. For schedulability analysis using our approach, the compo-

nent must export its worst case resource demand which depends on the task model and the scheduler. Any task model for which the component can compute its worst case resource demand can be used in our framework. Independent tasks are assumed because our focus in this paper is on component abstraction and composition. Matic and Henzinger [19] have addressed the issue of abstracting a set of interacting periodic tasks with data dependences using buffers, and their technique can be applied in our framework. A real-time component consists of a real-time workload and a scheduling policy for the workload.

DEFINITION 1 (SIMPLE COMPONENT). *A simple real-time component C is defined as,*

$C = \langle \{T_1, \cdots, T_n\}, RM/EDF \rangle$ where $T_i = (p_i, e_i)$ is a real-time task with period p_i and worst case execution time e_i. Deadline of T_i is assumed to be same as p_i. The tasks in the component are scheduled using either rate monotonic (RM) or earliest deadline first (EDF) scheduler [17].

DEFINITION 2 (COMPLEX COMPONENT). *Components in a hierarchical system that comprise of other components will be called complex components. A complex component C is defined as,*

$C = \langle \{C_1, \cdots, C_n\}, RM/EDF \rangle$ where each C_i is either a simple component or another complex component. These components are scheduled among themselves using either RM or EDF scheduler.

The term component will be used to refer to both simple as well as complex components in this paper. The context should make the meaning clear and we will explicitly make a distinction wherever necessary. The resource demand of a real-time component is the collective resource requirements that the tasks in the component request under the scheduling algorithm of the component. The demand bound function [23] gives the maximum resource demand of a component within a given time interval. Demand bound function for a component with tasks $\{T_1, \cdots, T_n\}$ and using EDF scheduler is given in Eq. (1). Demand bound function for a task T_i in a component with tasks $\{T_1, \cdots, T_n\}$ and using RM scheduler is given in Eq. (2) where $HP(T_i)$ represents a set of tasks in the component having priority higher than that of T_i.

$$dbf(t) = \sum_{i=1}^{n} (\lfloor t/p_i \rfloor e_i) \tag{1}$$

$$dbf_i(t) = \sum_{T_k \in HP(T_i)} (\lceil t/p_k \rceil e_k) + e_i \tag{2}$$

A resource model is a model for specifying the timing properties of a real-time resource. Supply bound function gives the minimum resource supply that the resource model is guaranteed to provide within a given time interval. A resource model R will satisfy the resource demand of a real-time component C if the maximum resource demand of C is less than the minimum resource supply of R for any time interval. In this paper, we consider the periodic resource model $R = (\Pi, \Theta)$ [23], which guarantees a minimum resource supply of Θ units in any time interval consisting of Π units. This resource supply can be provided in any arbitrary fashion within the time interval. The supply bound function $sbf_R(t)$ of the periodic resource model $R = (\Pi, \Theta)$ is given

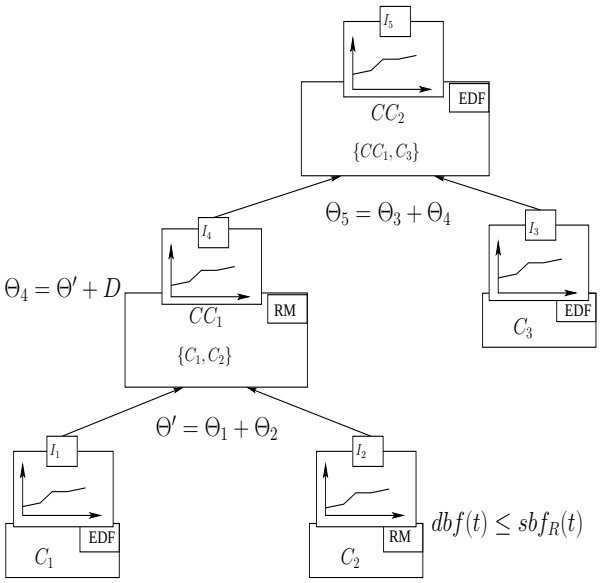

Figure 1: Hierarchical Real-Time System with Interfaces

in [23] for the time interval length t as

$$sbf_R(t) = \begin{cases} t - (k+1)(\Pi - \Theta) & \text{If } t \in [(k+1)\Pi - 2\Theta, \\ & (k+1)\Pi - \Theta] \\ (k-1)\Theta & \text{Otherwise} \end{cases} \tag{3}$$

where $k = max(1, \lceil (t - (\Pi - \Theta))/\Pi \rceil)$. A linear lower bound, $lsbf_R(t)$, for the supply bound function is given in [23] as

$$lsbf_R(t) = \Theta/\Pi[t - 2(\Pi - \Theta)]. \tag{4}$$

In this paper, we address the compositional schedulability analysis problem for a hierarchical real-time system. Figure 1 shows such a hierarchical system. At each level in the hierarchy, we have a set of real-time components and/or complex components along with a scheduler for scheduling them. In Figure 1, CC_1 is a complex component composed from components C_1 and C_2, and CC_2 is a complex component consisting of components C_3 and CC_1. For schedulability analysis, the resource demand of each component will be abstracted into a component interface such that if the interface is schedulable then the component is also schedulable. Interface for a complex component will be generated by composing the interfaces of individual components in the complex component. Since the components in the hierarchical system change dynamically, it is desirable to make this composition associative.

Context switches play an important role in schedulability analysis because they consume real-time resources. In a hierarchical system, context switches can occur at each level of the hierarchy. In this paper, context switches occurring within a simple component will be called task context switches and those occurring between components will be called component context switches. In Figure 1, context switches occurring within components C_1, C_2 and C_3 are task context switches and those occurring between components C_1 and C_2 or CC_1 and C_3 are component context switches. We assume that the task level context switch overhead is accounted for in the demand bound functions

of simple components. This can be achieved by bounding the number of preemptions within a component using equations given in [9]. The compositional schedulability analysis problem that we address can now be stated as follows.

Problem 1. Given a hierarchical real-time system,

1. Generate interfaces for each real-time component such that
   - schedulability of the interface guarantees schedulability of the component
   - interface takes into consideration component context switch overhead
2. Compose interfaces such that
   - schedulability of the composed interface guarantees schedulability of the individual interfaces
   - composition is associative
3. Minimize the abstracted resource demand for the hierarchical system

3. OVERVIEW OF OUR APPROACH

In this section we give an outline of our approach for generation and composition of component interfaces.

3.1 Component Interface Generation

A component interface must abstract the resource requirements of the component as a resource model. This will hide the complexity of the task set and the scheduler from higher level components in the system. If the abstracted resource model is schedulable then the underlying component is also schedulable. Schedulability conditions for a component C and resource model R can be given using demand and supply bound functions [23, 15]. Let $dbf(t)$ denote the demand bound function of a component C using EDF scheduler and $lsbf_R(t)$ denote the linear supply bound function of a periodic resource model $R = (\Pi, \Theta)$. Further, let LCM denote the least common multiple of the periods of all the tasks in component C. Component C is schedulable using resource model R if,

$$\forall t \in (0, LCM], dbf(t) \leq lsbf_R(t) \tag{5}$$

Similarly, for a component C using RM scheduler with demand bound function $dbf_i(t)$ for each task T_i, C is schedulable using resource model R if,

$$\forall i, \exists t \in (0, p_i], dbf_i(t) \leq lsbf_R(t) \tag{6}$$

We will use linear supply bound function in schedulability conditions to make the interface generation algorithms described later tractable, even though this linearization leads to only sufficient schedulability conditions. If instead we used supply bound functions, then the algorithms will be required to iterate over different values of Θ making them intractable. The interface generation procedure described in this paper uses schedulability conditions given in Eqs. (5) and (6) to compute resource models that guarantee component schedulability. We will abstract components as periodic resource models such that each component interface will consist of a set of periodic resource models with different values for the period. For each period, it will give the minimum resource capacity required from a periodic model to guarantee schedulability of the component.

Definition 3 (Component Interface). *An interface for a real-time component C is defined as,*

$$I = \{(\Pi, \Theta) | 1 \leq \Pi \leq P^*\}$$

such that $\forall (R = (\Pi, \Theta)) \in I, lsbf_R(t)$ satisfies schedulability conditions given by Eq. (5) or (6) where demand bound function is given by Eq. (1) or (2). P^* is the system designer defined upper bound for the period of resource models in component interfaces.

In this paper we assume that for values of Π greater than P^*, the periodic resource model in I is either (Π, Π) if $(P^*, \Theta) \in I$ satisfies $\Theta \leq P^*$ or $(\Pi, \Pi + 1)$ otherwise. Periodic resource models in interfaces are assumed to have at least 100% utilization for period values greater than P^*. In this paper we also assume that $P^* = LCM$ for simple components and have shown in our technical report [8] that such interfaces are good approximations under certain reasonable assumptions.

3.2 Compact Interface Representation

Under assumptions stated in Section 3.1, the size of an interface given by Definition 3 will be $O(P^*)$. In order to reduce the space requirements for an interface, we give below a compact representation that uses schedulability conditions given by Eqs. (5) and (6).

Definition 4 (Compact Interface Representation). *Compact interface representation for a component C is given as,*

$$RI = \{RI_j = \langle \Pi, t_j, dbf(t_j) \rangle | j_{min} \leq \Pi \leq j_{max}, 1 \leq j \leq k\}$$

where, $1_{min} = 1, k_{max} = P^*$ and $\forall j, j_{min} = (j-1)_{max} + 1$.

For a range of values of period Π, the compact representation gives a value for time instant and a value for the demand bound at that time instant. The resource capacities of periodic resource models for this range of period values can be obtained by substituting the time instant and demand bound function values in Eq. (5) or (6). Converting a compact interface representation RI to a component interface I then involves computing the resource capacities Θ for all values of the period from 1 to P^*. For example, a single interface representation $\langle [1, P^*], 75, 9 \rangle$ is sufficient to compute the entire component interface as given by Definition 3. For each value of period in the interval $[1, P^*]$, the corresponding periodic resource model for component interface can be computed by substituting the values $t = 75$ and $dbf(t) = 9$ in Eq. (5) for EDF scheduler or in Eq. (6) for RM scheduler.

3.3 Interface Composition

Interface composition aims to compose a set of interfaces representing a set of lower level components into a single interface for the complex component. This interface must account for the context switch overhead incurred by the complex component and also satisfy schedulability conditions. Interface composition will be ***associative*** if the same interface is generated for a complex component, irrespective of the order in which components are added to the complex component. Consider a situation where some underlying simple component is modified leading to a change in the workload of the complex component. If composition is associative then the new interface for the complex component can be directly computed using the old interface of

the complex component as well as old and new interfaces of the modified simple component. Associativity is essential for efficient schedulability analysis of systems in which components are added and deleted from the system dynamically. One way to compose interfaces maintaining associativity would be to add the corresponding minimum resource demand for each resource period. Schedulability of the composed interface then guarantees schedulability of each of the individual interfaces. Since the result of this composition is another interface, this technique can be applied iteratively. After each composition, the composed interface will be modified to account for the context switch overhead incurred by it when scheduled with other interfaces in the system.

As shown in Figure 1, component C_1 is abstracted as interface I_1, C_2 as I_2 and C_3 as I_3. Component CC_1 comprises of components C_1 and C_2. Hence interface I_4 is generated by addition of interfaces I_1 and I_2 along with component context switch overhead D. For period values greater than P^*, the periodic models in interface I_4 are assumed to be equal to $(\Pi, \Pi + 1)$.

$$I_4 = \{(\Pi, \Theta_1 + \Theta_2 + D) | (\Pi, \Theta_1) \in I_1, (\Pi, \Theta_2) \in I_2, 1 \leq \Pi \leq P^*\} \bigcup \{(\Pi, \Pi + 1) | \Pi > P^*\}$$

3.4 Resource Demand Minimization

Once a single interface is generated for the top-level component in the hierarchical system, the top-level component will pick one value for the resource period which will be used by all the interfaces in the system. Our composition technique dictates that in order for the analysis to be correct, all the interfaces in the system must use the same resource period. This value is picked such that the corresponding periodic model in the interface at the topmost level has the least resource demand as compared to other periodic models in that interface. This periodic resource model will then be exported to the operating system for scheduling purposes. The periodic models for each of the components in the system can then be obtained from their corresponding interfaces using the same value for period. Using a single value for the period means that all the interfaces at any level in the system have the same priority under RM or EDF scheduler. We assume that the scheduler at any level will assign arbitrary priorities to the interfaces at that level in each period of execution.

As shown in Figure 1, interfaces I_3 and I_4 are composed to generate interface I_5. Since there is only one interface at this level (topmost level), no component context switch overhead is added to I_5. The entire system is schedulable if there exists some (Π, Θ) in interface I_5 such that $\Theta \leq \Pi$. The system can then select a value for Π that minimizes the resource demand (least Θ/Π) for interface I_5.

4. SUPPLY BOUND FUNCTION WITH CONTEXT SWITCH OVERHEAD

Component level context switch overhead depends on the period of the resource model chosen for the component interface. A smaller period will, in general, result in a larger number of context switches. We would like to select a resource period that minimizes both the context switch overhead as well as the resource demand. For this reason, we would like to account for context switch overheads in the interface resource demand itself. In this section we will derive a supply bound function for periodic resource models that accounts for this context switch overhead. This supply bound function will be used in schedulability conditions that will be used for generation of component interfaces.

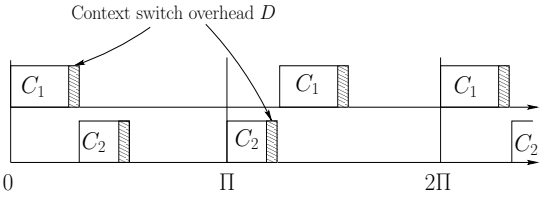

Figure 2: Component Level Context Switch Overhead

Since context switching will use some resource capacity, given the linear supply bound function in Eq. (4), the actual supply that will be available to the component in any time interval of length Π will be less than Θ. Let D denote an upper bound on the overhead associated with each context switch as shown in Figure 2. We assume without loss of generality that this overhead is incurred by the component at the end of its execution. Since the periods of all the interfaces in the hierarchical system will be fixed to one value by the top-level component, interfaces will have the same priority under both RM and EDF scheduler. We assume that all the interfaces at any level in the system will be assigned arbitrary priorities by the scheduler at that level, within each period of execution. The supply bound function must then account for one context switch overhead in every time interval of length Π. Since every component executes exactly once (begins execution only if it is the highest priority component among all components that have not yet executed) in each period, there will be exactly one context switch associated with that component. If Θ is the resource supplied, $(\Theta - D)$ will be the actual resource available to the component. Also, the worst case delay in the supply of this resource will be at most $2(\Pi - \Theta + D)$.

DEFINITION 5. *Let D be the execution overhead associated with every component context switch. Linear supply bound function with context switch overhead is,*

$$lsbf_R(t) = (\Theta - D)/\Pi[t - 2(\Pi - (\Theta - D))] \quad (7)$$

Schedulability conditions stated in Eqs. (5) and (6) can be modified to account for component context switching overhead by using the linear supply bound function given in Eq. (7).

5. INTERFACE GENERATION FOR COMPONENTS WITH EDF SCHEDULER

In this section we give an algorithm to generate an interface for components that use EDF scheduler. The algorithm uses the schedulability condition given in Eq. (5) to generate a periodic resource set over varying resource periods representing the component interface. The algorithm generates a compact representation for this interface which is then exported to higher levels for composition.

5.1 Interface Generation Algorithm

Algorithm 1 takes as input a real-time component C using EDF scheduler and outputs a compact interface representation RI for C that satisfies schedulability condition given in

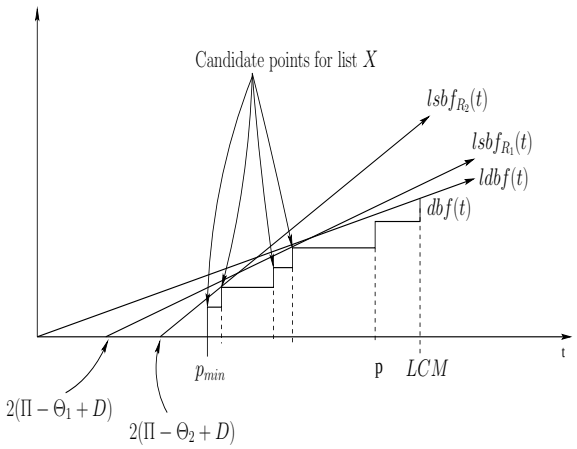

Figure 3: Schedulability Condition under EDF

Eq. (5). Computation of Θ^+ in line 1 of Algorithm 1 can be done in time $O(LCM)$ because we only need to look at time instants at which some task has its deadline. These are time instants at which $lsbf_R(t)$ can intersect $dbf(t)$ guaranteeing schedulability. As shown in Figure 3 the linear demand bound function $ldbf(t)$ intersects with the demand bound function $dbf(t)$ at $t = LCM$. Since any linear supply bound function $lsbf_R(t)$ must always be greater than $dbf(t)$ to ensure schedulability, optimal $lsbf_R(t)$ must intersect $dbf(t)$ at some point earlier than or equal to $t = LCM$. If it does not intersect $dbf(t)$, then $lsbf_R(t)$ will not be optimal and Θ can be decreased further. As shown in Figure 3, for period Π, $lsbf_{R_1}(t)$ is the optimal supply bound function for the demand bound function $dbf(t)$.

Algorithm 1 initially finds a point p of intersection between $dbf(t)$ and optimal $lsbf_R(t)$ for $\Pi = 1$. It then determines candidate points of intersection between $dbf(t)$ and $lsbf_R(t)$ for higher values of Π using procedure RelevantPointsEDF(X) (line 7). Procedure RelevantPointsEDF(X) given in Algorithm 2 takes as input a list X consisting of a single time instant representing intersection of optimal $lsbf_R(t)$ with $dbf(t)$ for period value 1. List Q in procedure RelevantPointsEDF(X) only consists of time instants at which some task has its deadline. Assuming $(\Pi - \Theta + D)$ is a non-decreasing function of Π for optimal Θ, we can show that the point of intersection of $lsbf_R(t)$ with the time axis is non-decreasing. Further, if optimal $lsbf_R(t)$ for period Π intersects $dbf(t)$ at some point t_1, then we can show that optimal $lsbf_R(t)$ for period $(\Pi+1)$ will intersect $dbf(t)$ either at t_1 or at a point on $dbf(t)$ which is smaller than t_1 and has the least slope with t_1. Procedure RelevantPointsEDF(X) updates list X with such a sequence of relevant time instants. Due to lack of space, the proof of correctness of Algorithm 1 is given in the technical report [8]. Running time of Algorithm 1 is $O(LCM) + O(LCM \ln LCM) = O(LCM \ln LCM)$ where procedure RelevantPointsEDF(X) runs in time $O(LCM \ln LCM)$.

Example 1. Let component C_1 in Figure 1 consist of three tasks $T_1 = (45, 2), T_2 = (65, 3)$ and $T_3 = (85, 4)$ and component C_3 in Figure 1 consist of two tasks $T_1 = (45, 1)$ and $T_2 = (75, 2)$. Interface I_1 for component C_1 and I_3 for C_3 are plotted in Figure 4 for values of period between 1 and 30. Compact representations for interfaces I_1 and I_3 generated

Algorithm 1 Interface Generation under EDF Scheduler

1: Solve $\Theta^+ = \max_\Theta[\forall t \in (0, LCM]\{$ Eq. (5) $\}]$ with $\Pi = 1$
2: **if** Θ is maximized at $t = p_{min} = \min_i\{p_i\}$ **then**
3: Set $RI = \{RI_1 = \langle \Pi, p_{min}, dbf(p_{min})\rangle | (1 = 1_{min}) \leq \Pi \leq (1_{max} = LCM)\}$
4: Terminate
5: **else**
6: Initialize list X with p where Θ is maximized at $t = p$
7: Perform $RelevantPointsEDF(X)$
8: Initialize $j = 1, Z = 1$
9: **for** $\Pi = 2$ to LCM **do**
10: **if** X has only one element **then**
11: $RI = RI \bigcup \{RI_j = \langle \Pi', t, dbf(t)\rangle | j_{min} = Z, j_{max} = LCM\}$ where t is first element in X
12: Terminate
13: **else**
14: Solve Eq. (5) to compute Θ_1 with t equal to first element in X
15: Solve Eq. (5) to compute Θ_2 with t equal to the next element in X
16: **if** $\Theta_2 > \Theta_1$ **then**
17: $RI = RI \bigcup \{RI_j = \langle \Pi', t, dbf(t)\rangle | j_{min} = Z, j_{max} = (\Pi - 1)\}$ where t is first element in X
18: Update $Z = \Pi, j = j + 1$
19: Remove the first element of X
20: **end if**
21: **end if**
22: **if** $\Pi = LCM$ **then**
23: $RI = RI \bigcup \{RI_j = \langle \Pi', t, dbf(t)\rangle | j_{min} = Z, j_{max} = LCM\}$ where t is first element in X
24: **end if**
25: **end for**
26: **end if**

Algorithm 2 $RelevantPointsEDF(X)$

1: Initialize t' to the first element of X, s' to 0
2: Let Q be a list of tuples $\langle t, s\rangle$ where t is a point on the time axis and s is the slope of a line segment connecting $dbf(t)$ to $dbf(t')$ such that $t < t'$
3: Sort Q such that the resulting list has non-decreasing slope
4: **for** Each element $\langle t, s\rangle$ of Q taken in sorted order **do**
5: **if** $t < t'$ and $s > s'$ **then**
6: Append t to X
7: Update $s' = s, t' = t$
8: **end if**
9: **end for**

using Algorithm 1 are given in Table 1. The size of these compact representations is much smaller than the LCM of the periods of tasks in the components.

Given a period value, say $\Pi = 10$, the optimum resource capacities for the interfaces can be computed using the compact representations given in Table 1 and schedulability condition given in Eq. (5). For interface I_1, substituting $\Pi = 10, t = 90$ and $dbf(t) = 11$ in Eq. (5) gives a resource capacity of $\Theta = 1.607$. Similarly, for interface I_3, putting $\Pi = 10, t = 90$ and $dbf(t) = 4$ in Eq. (5) gives a resource capacity of $\Theta = 0.662$.

Interface I_1			Interface I_3		
Π	t	dbf(t)	Π	t	dbf(t)
[1,1]	9945	1369	[1,1]	675	33
[2,4]	2210	304	[2,6]	225	11
[5, 6]	270	117	[7,16]	90	4
[7,21]	90	11	[17,∞]	45	1
[22,∞]	45	2			

Table 1: Representation for Interfaces I_1 and I_3

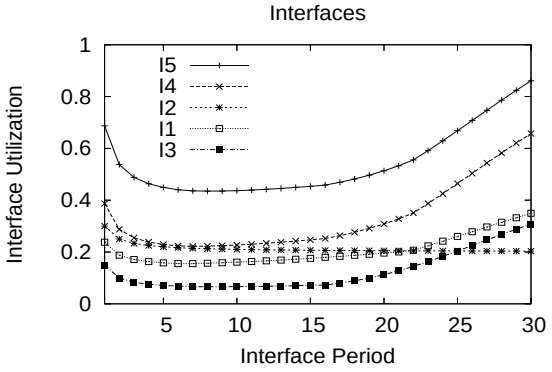

Figure 4: Interface Plots: Period vs. Utilization

6. INTERFACE GENERATION FOR COMPONENTS WITH RM SCHEDULER

In this section we describe algorithms to generate an interface for a real-time component that uses RM scheduler. Schedulability conditions given in Eq. (6) are used by the algorithm to generate the interface. We will first give an algorithm to generate interfaces for each task in the component separately. We later show how to combine interfaces of individual tasks to generate an interface for the entire component.

6.1 Interface Generation for a Single Task

Algorithm 3 computes interface RI_{k_i} for a task T_i in component C_k that uses RM scheduler. In line 1 of Algorithm 3 we take the minimum value of Θ because that gives a periodic resource model which minimizes the resource demand for task T_i. Schedulability condition for task T_i requires that the linear supply bound function $lsbf_R(t)$ intersect the demand bound function $dbf_i(t)$ at some point within $(0, p_i]$. For each period Π, Algorithm 3 computes the minimal Θ that guarantees schedulability of the component. As shown in Figure 5, $lsbf_{R_1}(t)$ is the optimal supply bound function for $dbf_i(t)$ with period Π. Procedure RelevantPointsRM(X) in line 7 of Algorithm 3 is similar to Algorithm 2 given in Section 5.1 with the condition in line 5 replaced with $[(t > t') \wedge (s > s')]$. Also, list Q only contains time instants t such that some task is released at instant $t + \delta$ for a very small fixed δ as shown in Figure 5 and $p < t \leq p_i$ where p is the first element in X. These are time instants at which the resource demand for the interface can be minimized still guaranteeing schedulability of the component. If t_1 represents a time instant where optimal $lsbf_R(t)$ for period Π intersects $dbf_i(t)$, then we can show that the optimal $lsbf_R(t)$ for period $(\Pi + 1)$ will intersect $dbf_i(t)$ either at t_1 or at a point greater than t_1 and having least slope with t_1. Procedure RelevantPointsRM(X) computes such a list of relevant time instants and stores it in X. Procedure RelevantPointsRM(X) can be executed in time $O(p_i \ln p_i)$ giving an overall running time of $O(p_i) + O(p_i \ln p_i) + O(LCM)$ to Algorithm 3. Correctness of Algorithm 3 has been proved in [8].

Algorithm 3 Interface Generation under RM Scheduler

1: Solve $\Theta^- = \min_\Theta[\forall t \in (0, p_i]\{$ Eq.6 $\}]$ with $\Pi = 1$
2: **if** Θ is minimized at $t = p_i$ **then**
3: Set $RI_{k_i} = \{RI^1_{k_i} = \langle \Pi, p_i, dbf_i(p_i)\rangle | (1 = 1_{min}) \leq \Pi \leq (1_{max} = LCM)\}$
4: Terminate
5: **else**
6: Initialize list X with p where Θ is minimized at $t = p$
7: Perform $RelevantPointsRM(X)$
8: Initialize $j = 1, Z = 1$
9: **for** $\Pi = 2$ to LCM **do**
10: **if** X has only one element **then**
11: $RI_{k_i} = RI_{k_i} \bigcup \{RI^j_{k_i} = \langle \Pi', t, dbf_i(t)\rangle | j_{min} = Z, j_{max} = LCM\}$ where t is first element in X
12: Terminate
13: **else**
14: Solve Eq. (6) to compute Θ_1 with t equal to first element in X
15: Solve Eq. (6) to compute Θ_2 with t equal to next element in X
16: **if** $\Theta_2 < \Theta_1$ **then**
17: $RI_{k_i} = RI_{k_i} \bigcup \{RI^j_{k_i} = \langle \Pi', t, dbf_i(t)\rangle | j_{min} = Z, j_{max} = (\Pi - 1)\}$ where t is first element in X
18: Update $Z = \Pi, j = j + 1$
19: Remove the first element in X
20: **end if**
21: **end if**
22: **if** $\Pi = LCM$ **then**
23: $RI_{k_i} = RI_{k_i} \bigcup \{RI^j_{k_i} = \langle \Pi', t, dbf_i(t)\rangle | j_{min} = Z, j_{max} = LCM\}$ where t is first element in X
24: **end if**
25: **end for**
26: **end if**

As a passing note, Algorithm 3 can also be used to compute the optimal CPU execution frequency for a task set under RM scheduler. Optimal execution frequency is the minimum execution frequency of the CPU that guarantees

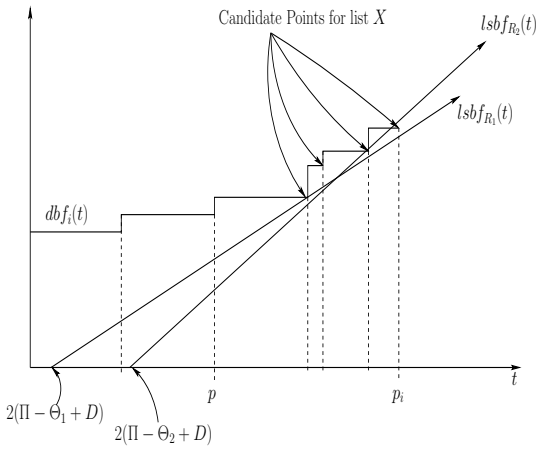

Figure 5: Schedulability Condition under RM

schedulability of the component. This problem has already been solved for components that use EDF scheduler [3, 2] or RM scheduler [21]. Our algorithm provides an alternate solution to this problem, details of which can be obtained from our technical report [8].

6.2 Combining Task Interfaces to Generate Component Interface

Let $RI_{k_1}, \cdots, RI_{k_n}$ denote interface representations for tasks $T_1, \cdots, T_n$ respectively generated by Algorithm 3 and let $I_{k_1}, \cdots, I_{k_n}$ denote the corresponding task interfaces. In this section we describe a procedure for combining these interfaces to generate an interface I_k for component C_k that satisfies the schedulability condition for the component. Let $GiveTheta(I_{k_i}, \Pi)$ return the value of Θ such that $(\Pi, \Theta) \in I_{k_i}$. Also, let $GiveModel(I_{k_i}, \Pi)$ return the periodic resource model (Π, Θ) such that $(\Pi, \Theta) \in I_{k_i}$. Algorithm 4 generates the component interface I_k for component C_k given task interfaces $I_{k_1}, \cdots, I_{k_n}$. For each period Π, Algorithm 4 computes the maximum required resource supply from among interfaces of all the tasks in the component. This is the minimum resource supply that must be provided by a periodic resource with period Π to satisfy schedulability condition of component C_k.

Algorithm 4 Composing Task Interfaces into a Component Interface

1: **for** $\Pi = 1$ to LCM **do**
2: Compute $l = \arg\max_i(GiveTheta(I_{k_i}, \Pi))$
3: $I_k = I_k \bigcup \{GiveModel(I_{k_l}, \Pi)\}$
4: $RI_k = RI_k \bigcup \{\langle \Pi, t, dbf(t) \rangle \in RI_{k_l}^j | j_{min} \leq \Pi \leq j_{max}\}$
5: **end for**
6: $I_k = I_k \bigcup \{(\Pi, \Pi + 1) | \Pi > LCM\}$

Example 2. Let component C_2 in Figure 1 consist of three tasks $T_1 = \{35000, 2000\}, T_2 = \{55000, 3000\}$ and $T_3 = \{75000, 4000\}$. Its interface I_2 for period values between 1 and 30 is given in Figure 4 and the compact representation for I_2 generated using Algorithm 3 is given in Table 2. Again, the size of the compact representation is much smaller than the LCM of component.

Given a period value, say $\Pi = 10$, the corresponding resource capacity for interface I_2 can be computed by substituting the values $\Pi = 10, t = 70000$ and $dbf(t) = 14000$ from Table 2 in Eq. (6). Then, $\Theta = 2.0004$ is the optimum resource capacity for interface I_2 with period $\Pi = 10$.

Interface I_2		
Π	t	dbf(t)
[1,22192]	70000	14000
[22193,∞]	35000	2000

Table 2: Representation for Interface I_2

7. INTERFACE COMPOSITION

Interface generated from a simple component already accounts for the context switch overhead incurred by that component. Hence these interfaces can be composed directly. But, as will be seen later in this section, interfaces generated by composing other interfaces do not account for the context switch overhead. In this section we will first describe interface composition that guarantees schedulability and is associative. We will then show how to modify a composed interface so that it accounts for the context switch overhead incurred by that complex component.

7.1 Associative Composition

As described in Section 3, we compose interfaces by simply adding the individual resource demands for each value of period. Since addition is an associative operation, this technique will make the composition associative.

DEFINITION 6 (COMPOSED INTERFACE). *Let $I_1, \cdots, I_n$ denote a set of interfaces for the components of a complex component CC with $LCM_1, \cdots, LCM_n$ denoting the least common multiples of the corresponding task periods. Let P^* for the composed interface be either $LCM_{min} = \min\{LCM_1, \cdots, LCM_n\}$ or $U \times p$ for some $p > 1$ where U is the utilization of component CC. Composed interface I is given by,*

$$I = \{(\Pi, \sum_{i=1}^{n} \Theta_i) | \Pi \in [1, P^*]\} \bigcup \{(\Pi, \Pi + 1) | \Pi > P^*\}$$

where $\forall 1 \leq i \leq n, (\Pi, \Theta_i) \in I_i$

Interface composition can be repeated at each level of the hierarchical system which will result in a single interface at the topmost level. Composition and interface guarantees at each level then ensure that the entire system is schedulable if the interface at the topmost level is schedulable. An interface I is schedulable if and only if,

$$\exists \Pi \geq 1, (\Theta, \Pi) \in I \text{ and } \Theta \leq \Pi \tag{8}$$

If the interface at the topmost level is schedulable, then the top-level component will pick a value Π^* for period Π that minimizes the resource demand (Θ/Π) of the system. Every component in the system will then receive a resource supply corresponding to the periodic resource model for period Π^* given in the interface for that component.

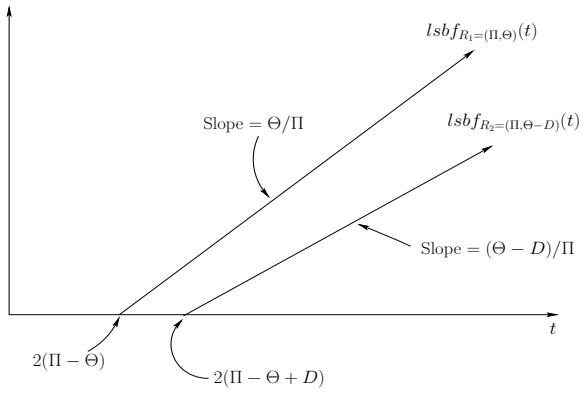

Figure 6: Supply Functions with Context Switch Overhead

7.2 Interface Modification for Context Switch Overhead

The schedulability condition used to generate an interface for a simple component uses linear supply bound function given in Eq. (7). Hence interfaces generated for simple components using Algorithm 1 or Algorithm 3 account for the context switch overhead incurred by those components. But interfaces composed from other interfaces using resource capacity addition as described in Section 7.1 do not account for this overhead. Consider a higher level component providing a minimum supply of resource to an interface composed as in Section 7.1. Since a portion of this supply will be required for context switching, the interface will not be schedulable with this minimum supply. This scenario is shown in Figure 6 where $lsbf_{R_1=(\Pi,\Theta)}(t)$ is the supply provided by a higher level interface and the actual supply available after discounting the context switch overhead D is $lsbf_{R_2=(\Pi,\Theta-D)}(t)$. This means that if the supply bound function of the composed interface (interface generated in Section 7.1) for a specific period is $lsbf_{(\Pi,\Theta-D)}(t)$, then the supply bound function of the exported interface (component interface that will be exported to higher levels) for the same period must be $lsbf_{(\Pi,\Theta)}(t)$.

Definition 7 (Schedulability Condition). *A composed interface is schedulable by an exported interface if and only if for each period* Π,

$$\forall t \in (0,\Pi], (\Theta - D)/\Pi(t - 2(\Pi - \Theta + D)) \geq \Theta'/\Pi(t - 2(\Pi - \Theta')) \quad (9)$$

where $lsbf_{CI}(t) = \Theta'/\Pi(t - 2(\Pi - \Theta'))$ *is the linear supply bound function of the composed interface and* $lsbf_{EI}(t) = \Theta/\Pi(t - 2(\Pi - \Theta))$ *is the linear supply bound function of the exported interface.*

Setting $\Theta = \Theta' + D$ satisfies Eq. (9) exactly. Thus given a composed interface, we can generate an exported interface by adding the context switch overhead D to the minimum resource demand for each value of period.

In Figure 4, interface I_4 is obtained by composing interfaces I_1 and I_2 with context switch overhead $D = 0.1$ and interface I_5 is obtained by composing interfaces I_4 and I_3. The minimum resource demand for the composed interface I_5 occurs when period $\Pi = 8$ as shown in Figure 4. The corresponding utilization value is 0.435 and is shown in Table 3.

8. COMPARISON TO EARLIER WORK

The interface generated at the topmost level of a hierarchical system is used by the top-level component to choose a period for the periodic resource model such that the resulting model has the least resource demand among all models in that interface. This resource model is then used for scheduling the lower level interfaces and components. The same period is also used to determine the periodic resource models to be used for scheduling all other components in the system. This is in contrast to the compositional framework developed by Shin and Lee [23] in which each real-time component itself chooses the period for its periodic resource abstraction. Since the periods of resource abstractions are chosen independently, they are all not necessarily the same. Hence to compose these abstractions, their supply bound functions are first transformed to suitable demand bound functions by treating each periodic abstraction (Π, Θ) as a periodic task with period Π and worst case execution time Θ. Schedulability conditions are then used to generate a periodic resource abstraction for the demand bound functions. This results in the composition being not associative and also induces an overhead in the composed abstraction due to transformation of supply bound functions to demand bound functions.

Example 3. Let I'_1, I'_2, I'_3, I'_4 and I'_5 denote interfaces generated by the compositional framework in [23] corresponding to interfaces I_1, I_2, I_3, I_4 and I_5 in Figure 1. We assume that each component determines its interface period such that resource demand is minimized locally. Then from Figure 4, $I'_1 = (6, 0.932)$, $I'_2 = (148, 29.8)$ and $I'_3 = (9, 0.593)$. Using schedulability conditions given in [23], interface $I'_4 = (4, 1.563)$ and $I'_5 = (2, 1.100)$. If we compose all three interfaces I'_1, I'_2 and I'_3 at a time, like $I''_5 = I'_1||I'_2||I'_3$, then $I''_5 = (5, 2.248)$. As shown in Table 3, in both cases, the minimum resource demand generated is greater than the minimum resource demand of interface I_5 generated by our approach.

The single period restriction can force a component interface to use a much smaller period than required, thereby increasing its context switching overhead. As shown in Table 3, interface I_2 will be forced to use a period value of 8 even though component C_2 can be scheduled using an interface with a much larger period. This can increase the number of context switches incurred by component C_2, as compared to when it is abstracted using the approach given in [23]. Our on-going work aims to compare the increase in context switching overhead as a result of the single period restriction versus the decrease in collective resource demand as a result of the associative composition.

9. CONCLUSION

In this paper we have described algorithms for abstracting components that use RM or EDF as the real-time scheduler. The algorithms run in time polynomial in the least common multiple of the periods of tasks in the component and generate efficient interfaces in practice. Interfaces account for task level context switching overhead incurred by the component and represent the resource requirements as periodic resource models. Each interface is represented as a set of periodic resource models for different values of period. This representation makes it possible to determine a periodic model

	Π	Θ	Θ/Π
$I_5 = (I_1 \| I_2) \| I_3$	8	3.4808	0.435
I_1'	6	0.932	0.155
I_2'	148	29.8	0.201
I_3'	9	0.593	0.066
$I_4' = I_1' \|\|' I_2'$	4	1.563	0.391
$I_5' = I_3' \|\|' I_4'$	2	1.100	0.550
$I_5'' = I_1' \|\|' I_2' \|\|' I_3'$	5	2.248	0.450

Table 3: Interface Models with Minimum Utilization

that minimizes the resource demand of the interface among all periodic models in that interface. Interface composition is achieved by addition of resource demands of individual interfaces. This composition takes into consideration the component context switch overhead incurred by the system. The composition is associative thereby simplifying analysis of systems with dynamically changing components.

In order to make the composition associative, periodic resource models with the same period are selected for all the interfaces in the system. It is an open problem whether incremental schedulability analysis can be achieved using interfaces represented as periodic resource models with varying period values.

10. ACKNOWLEDGMENTS

The authors would like to thank all the anonymous reviewers for their valuable inputs.

11. REFERENCES

[1] L. Almeida and P. Pedreiras. Scheduling within temporal partitions: response-time analysis and server design. In *Proc. of the Fourth ACM International Conference on Embedded Software*, September 2004.

[2] H. Aydin, R. Melhem, D. Mosse, and P.M. Alvarez. Power-aware scheduling for periodic real-time tasks. *IEEE Transactions on Computers*, 53(5):584–600, 2004.

[3] Hakan Aydin, Rami Melhem, Daniel Mosse, and Pedro Mejia-Alvarez. Determining optimal processor speeds for periodic real-time tasks with different power characteristics. In *13th Euromicro Conference on Real-Time Systems (ECRTS'01)*, page 225, 2001.

[4] R. I. Davis and A. Burns. Hierarchical fixed priority pre-emptive scheduling. In *Proc. of IEEE Real-Time Systems Symposium*, December 2005.

[5] L. de Alfaro and T. A. Henzinger. Interface automata. In *Proceedings of the Ninth Annual Symposium on Foundations of Software Engineering*. ACM Press, 2001.

[6] L. de Alfaro and T. A. Henzinger. Interface theories for component-based design. In *Proceedings of the First International Workshop on Embedded Software*, pages pp. 148–165. Lecture Notes in Computer Science 2211, Springer-Verlag, 2001.

[7] Z. Deng and J. W.-S. Liu. Scheduling real-time applications in an open environment. In *Proc. of IEEE Real-Time Systems Symposium*, pages 308–319, December 1997.

[8] Arvind Easwaran, Insik Shin, Insup Lee, and Oleg Sokolsky. Associative composition of hierarchical real-time systems. Technical Report MS–CIS–06–06, University of Pennsylvania, 2006.

[9] Arvind Easwaran, Insik Shin, Insup Lee, and Oleg Sokolsky. Bounding preemptions under EDF and RM schedulers. Technical Report MS–CIS–06–07, University of Pennsylvania, 2006.

[10] X. Feng and A. Mok. A model of hierarchical real-time virtual resources. In *Proc. of IEEE Real-Time Systems Symposium*, pages 26–35, December 2002.

[11] T. A. Henzinger, B. Horowitz, and C. M. Kirsch. Giotto: A time-triggered language for embedded programming. *Proceedings of IEEE*, 91:84–99, 2003.

[12] T. A. Henzinger and S. Matic. An interface algebra for real-time components. In *Proc. of IEEE Real-Time Technology and Applications Symposium*, pages 253–263, April 2006.

[13] T.-W. Kuo and C.H. Li. A fixed-priority-driven open environment for real-time applications. In *Proc. of IEEE Real-Time Systems Symposium*, pages 256–267, December 1999.

[14] G. Lipari and S. Baruah. Efficient scheduling of real-time multi-task applications in dynamic systems. In *Proc. of IEEE Real-Time Technology and Applications Symposium*, pages 166–175, May 2000.

[15] G. Lipari and E. Bini. Resource partitioning among real-time applications. In *Proc. of Euromicro Conference on Real-Time Systems*, July 2003.

[16] G. Lipari, J. Carpenter, and S. Baruah. A framework for achieving inter-application isolation in multiprogrammed hard-real-time environments. In *Proc. of IEEE Real-Time Systems Symposium*, December 2000.

[17] C.L. Liu and J.W. Layland. Scheduling algorithms for multi-programming in a hard-real-time environment. *Journal of the ACM*, 20(1):46 – 61, 1973.

[18] Mathworks. Models with multiple sample rates. In *Real-Time Workshop User Guide*, pages 1–34, The MathWorks Inc, 2005.

[19] S. Matic and T. A. Henzinger. Trading end-to-end latency for composability. In *Proc. of IEEE Real-Time Systems Symposium*, pages 99–110, December 2005.

[20] A. Mok, X. Feng, and D. Chen. Resource partition for real-time systems. In *Proc. of IEEE Real-Time Technology and Applications Symposium*, pages 75–84, May 2001.

[21] S. Saewong and R. Rajkumar. Practical voltage-scaling for fixed-priority RT-systems. In *Proc. of IEEE Real-Time Technology and Applications Symposium*, pages 106–115, May 2003.

[22] S. Saewong, R. Rajkumar, J.P. Lehoczky, and M.H. Klein. Analysis of hierarchical fixed-priority scheduling. In *Proc. of Euromicro Conference on Real-Time Systems*, June 2002.

[23] I. Shin and I. Lee. Periodic resource model for compositional real-time guarantees. In *Proc. of IEEE Real-Time Systems Symposium*, pages 2–13, December 2003.

[24] I. Shin and I. Lee. Compositional real-time scheduling framework. In *Proc. of IEEE Real-Time Systems Symposium*, December 2004.

[25] E. Wandeler and L. Thiele. Real-time interface for interface-based design of real-time systems with fixed priority scheduling. In *Proceedings of the 5th ACM International Conference on Embedded Software (EMSOFT '05)*, pages 80–89, October 2005.

[26] E. Wandeler and L. Thiele. Interface-based design of real-time systems with hierarchical scheduling. In *Proc. of IEEE Real-Time Technology and Applications Symposium*, pages 243–252, April 2006.