5-7-2007

# Schedulability Analysis of Hierarchical Real-Time Systems

Arvind Easwaran
*University of Pennsylvania*, arvinde@cis.upenn.edu

Insik Shin
*University of Pennsylvania*, ishin@cis.upenn.edu

Oleg Sokolsky
*University of Pennsylvania*, sokolsky@cis.upenn.edu

Insup Lee
*University of Pennsylvania*, lee@cis.upenn.edu



Copyright 2007 IEEE. Reprinted from:
Easwaran, A.; Insup Lee; Insik Shin; Sokolsky, O.; , "Compositional Schedulability Analysis of Hierarchical Real-Time Systems," Object and Component-Oriented Real-Time Distributed Computing, 2007. ISORC '07. 10th IEEE International Symposium on , vol., no., pp.274-281, 7-9 May 2007
doi: 10.1109/ISORC.2007.25
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4208854&isnumber=4208812

# Schedulability Analysis of Hierarchical Real-Time Systems

**Abstract**
Embedded systems are complex as a whole but consist of smaller independent modules interacting with each other. This structure makes embedded systems amenable to compositional design. Real-time embedded systems consist of real-time workloads having temporal deadlines. Compositional design of real-time embedded systems can be done using systems consisting of real-time components arranged in a scheduling hierarchy. Each component consists of some real-time workload and a scheduling policy for the workload. To simplify schedulability analysis for such systems, analysis can be done compositionally using interfaces that abstract the timing requirements of components. To facilitate analysis of dynam- ically changing real-time systems, the framework must support incremental analysis. In this paper, we summarize our work [19, 6] on schedulability analysis for hierarchical real-time systems. We describe a compositional analysis technique that abstracts resource requirements of components using periodic resource models. To support incremental analysis and resource bandwidth minimization, we describe an extension to this interface model. Each extended interface consists of multiple periodic resource models for different periods. This allows the selection of a periodic model that can schedule the system using minimum bandwidth. We also account for context switch overheads in these interfaces. We then describe an associative composition technique for such interfaces that supports incremental analysis.



Easwaran, A.; Insup Lee; Insik Shin; Sokolsky, O.; , "Compositional Schedulability Analysis of Hierarchical Real-Time Systems," Object and Component-Oriented Real-Time Distributed Computing, 2007. ISORC '07. 10th IEEE International Symposium on , vol., no., pp.274-281, 7-9 May 2007

doi: 10.1109/ISORC.2007.25

URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4208854&isnumber=4208812

# Schedulability Analysis of Hierarchical Real-Time Systems

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 interacting with each other. This structure makes embedded systems amenable to compositional design. Real-time embedded systems consist of real-time workloads having temporal deadlines. Compositional design of real-time embedded systems can be done using systems consisting of real-time components arranged in a scheduling hierarchy. Each component consists of some real-time workload and a scheduling policy for the workload. To simplify schedulability analysis for such systems, analysis can be done compositionally using interfaces that abstract the timing requirements of components. To facilitate analysis of dynamically changing real-time systems, the framework must support incremental analysis. In this paper, we summarize our work [19, 6] on schedulability analysis for hierarchical real-time systems. We describe a compositional analysis technique that abstracts resource requirements of components using periodic resource models. To support incremental analysis and resource bandwidth minimization, we describe an extension to this interface model. Each extended interface consists of multiple periodic resource models for different periods. This allows the selection of a periodic model that can schedule the system using minimum bandwidth. We also account for context switch overheads in these interfaces. We then describe an associative composition technique for such interfaces that supports incremental 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 the components using interfaces that abstract component complexities. 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. These abstractions must satisfy the timing requirements of components using minimum resource supply. *Incremental analysis* is also highly desirable for systems that

can be modified on the fly. Frameworks that support incremental analysis allow reuse of existing component interfaces for analysis of the modified system.

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. In this framework, resources are allocated from a parent node to its children. Many studies have been proposed on compositional schedulability analysis for component-based hierarchical scheduling frameworks [18, 12, 1, 2]. However, their approaches do not support incremental schedulability analysis. There have been recent studies [21, 22, 9, 20] that support incremental schedulability analysis using the interface theory [3, 4]. Assume-guarantee interface theories for real-time components with a generic real-time interface model [21, 22, 20] and with a bounded-delay resource partition interface model [9] have been proposed. However, they did not consider the problem of generating interfaces that use minimum resource supply to schedule components, taking into account context switch overheads. Also, when components use RM scheduler, the bounded-delay model based technique [9] does not support incremental analysis and the generic interface model based technique [21, 22, 20] does not support compositional analysis for dynamically changing systems.

In this paper we summarize our work [19, 6] on schedulability analysis of hierarchical real-time systems. Our interface model will be based on the periodic resource model [19, 12, 6], which can characterize the periodic behavior of resource allocations. This choice is implementation-oriented because many existing real-time schedulers support the periodic model. We describe our component interface model [19] that abstracts the resource demand of components in the form of a periodic resource model for a fixed period value. We develop schedulability conditions for the generation of such interfaces, when components comprise of periodic, independent tasks scheduled using RM or EDF schedulers. We then describe an interface composition technique that facilitates compositional analysis for such interfaces. However, this framework neither supports incremental analysis nor can it minimize the resource bandwidth required to schedule the system, taking into account context switch overheads. We then describe an extended interface model [6]

that allows component interfaces to consist of multiple periodic resource models for different periods. This interface allows the framework to select a resource model that can schedule the system using minimum bandwidth. We also describe how this model can account for context switch overheads. Finally, we describe a composition technique for such interfaces that is associative and hence can support incremental analysis.

**Related Work.** For real-time systems, there has been a growing attention to hierarchical scheduling frameworks [5, 10, 11, 7, 17, 18, 12, 1, 2, 15, 22, 20] that support hierarchical resource sharing under different scheduling algorithms.

Deng and Liu [5] proposed a two-level real-time scheduling framework for open systems, where the system-level scheduler schedules independently developed application components and each component has its own component-level scheduler for its internal tasks. Kuo and Li [10] presented an exact schedulability condition for such a two-level framework with the RM system scheduler and Lipari and Baruah [11, 13] 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 [16, 7]. In their framework, a parent component and its children are separated such that they interact with each other only through their resource partition model. However, they did not consider the component abstraction problem. The periodic resource model has been introduced to specify the periodic resource allocation guarantees provided to a component from its parent component [19, 12]. There have been studies [18, 12, 1, 2] on the component abstraction problem with periodic resource models. For a component with RM scheduler and a periodic resource model abstraction, Saewong *et al.* [18] introduced an exact schedulability condition based on worst-case response time analysis, and Lipari and Bini [12] presented a similar condition based on time demand calculations. Pedreira [1] and Davis and Burns [2] introduced worst-case response time analysis techniques under RM component-level scheduling, which enhance the previous work. All these techniques, however, do not support incremental analysis.

Matic and Henzinger [15] considered the issue of addressing the component abstraction problem in the presence of interacting tasks within a component. They considered two approaches, the RTW (Real-Time Workshop) [14] and the LET (Logical Execution Time) [8] semantics, for supporting tasks with intra-component and inter-component data dependencies. This technique also does not support incremental schedulability analysis.

There have been studies on the development of interface theory for supporting incremental design of component-based real-time systems, applying the interface theories [3, 4] into real-time context. These studies have proposed assume-guarantee interface theories for real-time components with a generic real-time interface model [21, 22, 20] and with a bounded-delay resource partition interface model [9]. For dynamically changing systems with RM scheduler based components, these frameworks do not support incremental and compositional analysis. The generic interface model theory [21, 22, 20] can be applied to periodic resource models if the periods of the resource models are fixed a priori for all the components in the system to one value. In our methodology, the value for the resource period can be chosen after the interfaces for all the components are generated. Hence the framework can select a period that minimizes the resource bandwidth required to schedule the system taking into account context switch overheads. This is not possible if the period values are fixed a priori.

The rest of the paper is organized as follows: Section 2 defines the incremental schedulability analysis problem that we address. Section 3 gives schedulability conditions for components scheduled using periodic resource models. Section 4 describes our work on compositional schedulability analysis for hierarchical real-time systems. Section 5 describes an extended framework that supports incremental analysis and also minimizes resource bandwidths. Section 6 concludes the paper and discusses future work.

## 2 System Model and 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 component 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 resource demand can be used in our framework. A real-time component consists of a real-time workload and a scheduling policy for the workload. The workload of a *simple component* comprises of periodic real-time tasks only. Whereas the workload of a *complex component* comprises of other simple and/or complex real-time components.

**Definition 1 (Simple Component)** *A* simple real-time component $C$ *is specified 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 RM or EDF scheduling policy.*

**Definition 2 (Complex Component)** *A complex real-time component* $C$ *is specified as,*

$C = \langle \{C_1, \cdots, C_n\}, RM/EDF \rangle$ *where each* $C_i$ *is a simple or complex component.*

The term component will be used to refer to both simple as well as complex components. The context should make the meaning clear and we will explicitly make a distinction wherever necessary. In this paper we address the schedulability analysis problem for a hierarchical real-time system. Figure 1 shows such a hierarchical system, where $CC_1$ 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 $CC_1$. In order to support compositional analysis, the resource demand of each simple component will be abstracted into an interface such that if the interface is schedulable then the component will also be schedulable. Interface for a complex component will be generated by composing the interfaces of components that form its workload. This composed interface must satisfy the property of compositionality for hierarchical real-time systems. *Compositionality* guarantees schedulability of component workload under the component scheduler, provided the composed interface of the component is schedulable. This property is given in Definition 3 where a dedicated uniprocessor means a processor that can supply $t$ units of computational resource every $t$ units of time. Also, in the definition by "interfaces $I_1, \ldots, I_n$ are *schedulable* under $A$ on a dedicated uniprocessor" we mean that resource requirements of the interfaces are met when they are scheduled under $A$ on a dedicated uniprocessor.

**Definition 3 (Compositionality of Interface)** *Let $C = \langle \{C_1, \ldots, C_n\}, A \rangle$ denote a complex component having interface $I$. Let $I$ be generated by composing interfaces $I_1, \ldots, I_n$ of components $C_1, \ldots, C_n$, respectively. $I$ satisfies the* compositionality *property if and only if whenever $I$ is* schedulable *on a dedicated uniprocessor, interfaces $I_1, \ldots, I_n$ are* schedulable *under $A$ on a dedicated uniprocessor.*

Since components in a hierarchical system can change dynamically, it is also desirable to support incremental analysis. One way to modify a hierarchical system is to add new components to it. In this paper we assume that this is the only modification that can be done to the system. In an incremental framework, the interface generated for a complex component will be independent of the order in which components that form its workload are added to the system. Hence the same interface will be generated for a component irrespective of the order in which its workload interfaces are composed.

**Definition 4 (Incremental Analysis)** *Let $C = \langle \{C_1, \ldots, C_n\}, A \rangle$ denote a component and $I_1, \ldots, I_n$ denote interfaces of components $C_1, \ldots, C_n$, respectively. Let $\mathcal{P}$ denote the set of all possible permutations of the set $[n]$. Also, let $I_\sigma$ denote an interface for $C$ generated by composing interfaces $I_1, \ldots, I_n$ where the order of composition is given by $\sigma \in \mathcal{P}$. Then, an analysis framework is* incremental *if and only if* $\forall(\sigma_1, \sigma_2 \in \mathcal{P}) \quad I_{\sigma_1} = I_{\sigma_2}$.

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. Since our focus in this work is on the component abstraction and composition problem, we ignore the context switch overheads incurred as a result of scheduling tasks within a simple component. We only consider context switch overheads incurred by components when they are scheduled among themselves. For example, in Figure 1 we will consider context switch overheads incurred when components $C_1$ and $C_2$ or $CC_1$ and $C_3$ are scheduled together. However, we will ignore context switch overheads incurred by tasks within components $C_1, C_2$ and $C_3$. The schedulability analysis problem that we address in this paper can be stated as follows.

**Definition 5** *Given a hierarchical real-time system,*

1. *Generate interface for each simple real-time component such that*

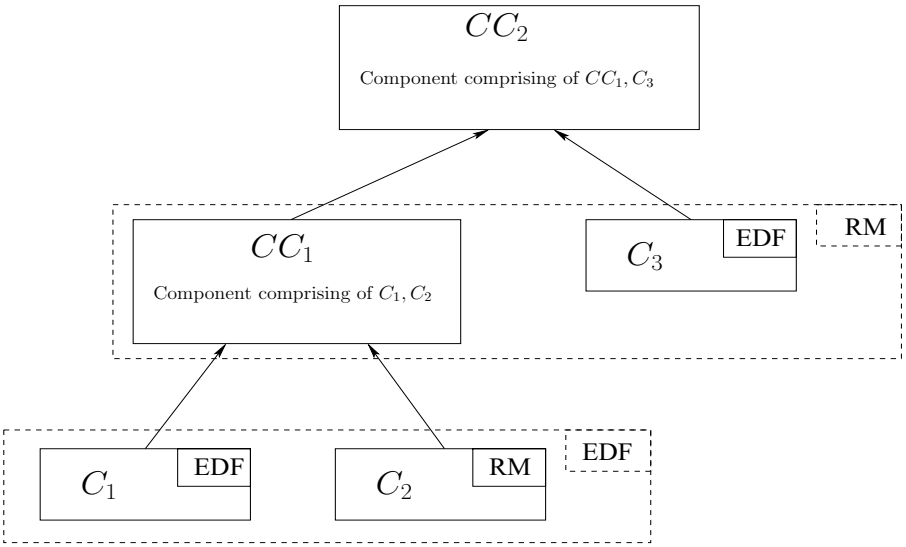



Figure 1: Hierarchical Real-Time System

- *if interface is schedulable then the component is also schedulable*
- *interface accounts for context switch overheads incurred by the component*

2. *Generate interface for each composed component by composing workload interfaces such that*
   - *composed interface satisfies* compositionality *property*
   - *composition supports incremental analysis*
   - *composed interface accounts for context switch overheads incurred by the complex component*
3. *Minimize the resource bandwidth that will be required to schedule the hierarchical system, when analysis is done using this interface model.*

In this paper we will use $\delta$ to denote the execution overhead incurred by the system for each context switch and $\text{LCM}_C$ to denote the least common multiple of periods of all the periodic tasks in a simple component $C$.

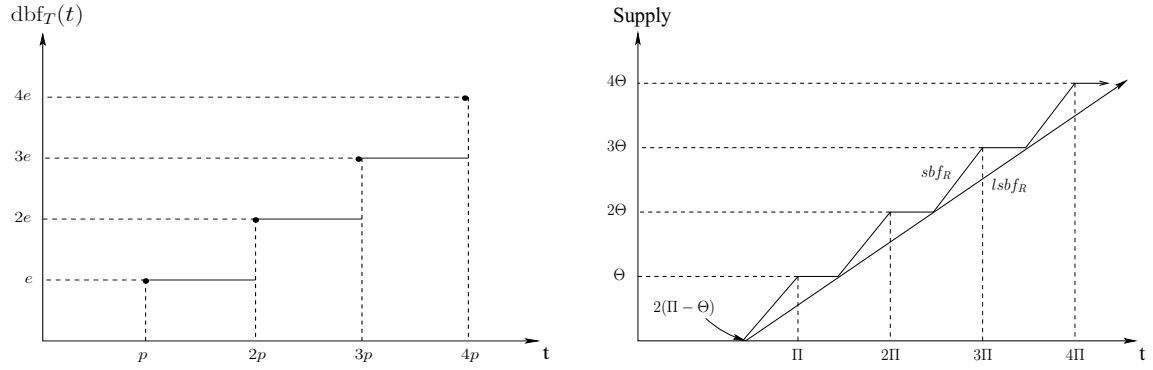
# 3 Schedulability Conditions for Periodic Resource Models

A real-time task consists of a set of real-time jobs that are required to meet temporal deadlines. The resource demand bound function ($dbf : \Re \rightarrow \Re$) of a real-time task upper bounds the amount of computational resource required to meet all its temporal deadlines. For a time interval length $t$, demand bound function gives the largest resource demand of the task in any time interval of length $t$. For example, Figure 2(a) shows the demand bound function $dbf_T$ of a periodic task $T = (p, e)$. As shown in the figure, the task requires $e$ units of computational resource every $p$ units of time in order to meet all its temporal deadlines. The demand bound function of a simple component is the worst-case resource requirements of tasks in the component. Given a component $C$ and time interval length $t$, $dbf_C(t)$ gives the largest resource demand of tasks in $C$ in any time interval of length $t$, when the tasks are scheduled using the scheduler in $C$. In our earlier work [19], we gave demand bound functions for simple components that use either RM or EDF scheduling policy. We reproduce these functions here for easy reference. Equation (1) gives the demand bound function for a component $C = \langle \{T_1 = (p_1, e_1), \cdots, T_n = (p_n, e_n)\}, EDF \rangle$. Similarly, Equation (2) gives the demand bound function for a task $T_i$ in a component $C = \langle \{T_1 = (p_1, e_1), \cdots, T_n = (p_n, e_n)\}, RM \rangle$ where $HP(T_i)$ denotes a set of tasks in $C$ having priority higher than $T_i$.

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

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

To satisfy the resource requirements of a real-time task or component, the system must supply sufficient computational resources. A resource model is a model for specifying the timing properties of this resource supply provided by the system. For example, a periodic resource supply that provides $\Theta$ units of resource every $\Pi$ units of time can be represented using the periodic resource model $R(\Pi, \Theta/\Pi)$. Here $\Theta/\Pi$ represents the resource bandwidth for model $R$. The supply bound function $\text{sbf} : \Re \rightarrow \Re$ of a resource model lower bounds the amount of resource that the model supplies. Given a resource model $R$ and interval length $t$, $\text{sbf}_R(t)$ gives the minimum amount of resource that model $R$ is guaranteed to supply in any time interval of length $t$. For a periodic resource model $R = (\Pi, \Theta/\Pi)$, Equations (3) and (4), first proposed by Shin and




(a) Periodic Demand Bound Function

(b) Periodic Supply Bound Function

Figure 2: Demand and Supply Bound Functions

Lee [19], give the supply bound function $sbf_R$ and its linear lower bound $lsbf_R$, respectively. In Equation (3), $k$ is equal to $max(1, \lceil (t - (\Pi - \Theta))/\Pi \rceil)$. Figure 2(b) shows the supply bound function $sbf_R$ and its linear lower bound $lsbf_R$ for model $R$.

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

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

In this paper we will abstract the resource requirements of components using periodic resource models. For this purpose, schedulability conditions must be defined for simple components over the resource models. A periodic resource model $R$ will satisfy the resource demand of a simple component $C$ if the maximum resource demand of $C$ is smaller than the minimum resource supply of $R$ in any time interval. Demand and supply bound functions of $C$ and $R$, respectively, can then be used to define these schedulability conditions. Theorems 1 and 2, proposed by Shin and Lee [19], gives schedulability conditions under EDF and RM schedulers, respectively.

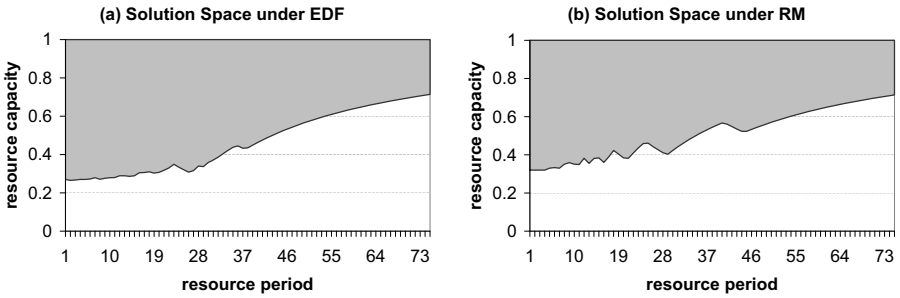



Figure 3: Schedulable region of periodic resource model $R(\Pi, \Theta/\Pi)$ in Example 1: (a) under EDF scheduling and (b) under RM scheduling.

**Theorem 1** *A component* $C = \langle \{T_1 = (p_1, e_1), \cdots, T_n = (p_n, e_n)\}, EDF \rangle$ *is schedulable over the worst-case resource supply of a periodic resource model* $R$*, if and only if*

$$\forall 0 < t \leq LCM_C \quad dbf_C(t) \leq lsbf_R(t) \tag{5}$$

**Theorem 2** *A component* $C = \langle \{T_1 = (p_1, e_1), \cdots, T_n = (p_n, e_n)\}, RM \rangle$ *is schedulable over the worst-case resource supply of a periodic resource model* $R$*, if and only if*

$$\forall T_i \quad \exists t_i \in [0, p_i] \quad dbf_{C,i}(t_i) \leq lsbf_R(t_i). \tag{6}$$

Interface generation algorithms described in this paper will use Equations (5) and (6) to compute resource models that guarantee component schedulability. These equations use linear supply bound functions instead of supply bound functions in order to make the algorithms tractable. If we used supply bound functions, the algorithms will be required to iterate over different values of resource bandwidth, making them intractable.

# 4 Compositional Schedulability Analysis

In our paper [19], we proposed an interface $R$ for a component $C = \langle \{W\}, A \rangle$ that abstracts the collective resource requirements of workload $W$ under scheduler $A$. This interface abstracts the resource requirements of the component as a periodic resource model $R = (\Pi, \Theta/\Pi)$, where $\Pi$ is user defined. The interface

does not reveal the internal information of the component such as the number of elements in its workload or its scheduling algorithm. Definition 6 gives the condition for a periodic resource model interface to be schedulable on a dedicated uniprocessor.

**Definition 6 (Periodic Resource Model Interface Schedulability)** *A periodic resource model interface,* $R = (\Pi, \Theta/\Pi)$*, is* schedulable *on a dedicated uniprocessor if and only if* $\Theta/\Pi \leq 1$*.*

We define the *(periodic) component abstraction* problem as the problem of deriving a periodic resource model interface for a real-time component. We formulate this problem as follows: given a component $C = \langle \{W\}, A \rangle$ and period $\Pi$, find an interface as an "optimal" periodic resource model $R = (\Pi, \Theta/\Pi)$ that can schedule $C$. Here optimality is with respect to minimizing the bandwidth of $R$.

**Example 1** *Let us consider a workload set* $\{T_1 = (50, 7), T_2 = (75, 9)\}$ *and a scheduling algorithm* $A =$ EDF/RM*. We now consider the problem of finding a periodic resource model* $R = (\Pi, \Theta/\Pi)$ *that can schedule component* $C = \langle \{T_1 = (50, 7), T_2 = (75, 9)\}, A \rangle$*. We can obtain a solution space to this problem by simulating Equation* (5) *under EDF scheduler or Equation* (6) *under RM scheduler. For any given resource period* $\Pi$*, we can find the smallest* $\Theta/\Pi$ *such that component* $C$ *is schedulable according to Theorem 1 or Theorem 2. Figure 3 shows such a solution space for EDF and RM schedulers, as the gray area, for resource periods in the range* 1 *to* 75*. For instance, when* $\Pi = 10$*, the minimum resource bandwidth that guarantees the schedulability of* $C$ *is 0.28 under EDF scheduling or 0.35 under RM scheduling.*

We define the *(periodic) component composition* problem as the problem of generating an interface for a complex component. This involves composing interfaces that abstract the component's workload. We formulate the component composition problem as follows: given a complex component $C = \langle \{C_1, \cdots, C_n\}, A \rangle$ and period $\Pi$, find an interface as an "optimal" periodic resource model $R = (\Pi, \Theta/\Pi)$ that can schedule $C$. Our approach is to develop "optimal" periodic resource model interfaces $R_1, \ldots, R_n$ that can schedule components $C_1, \ldots, C_n$, respectively, and to consider $C$ as consisting of $n$ periodic tasks, i.e., $C = \langle \{T_1 = (p_1, e_1), \cdots, T_n = (p_n, e_n)\}, A \rangle$, where $\forall i \quad 1 \leq i \leq n, R_i = (\Pi_i, \Theta_i/\Pi_i) \Rightarrow (p_i = \Pi_i, e_i = \Theta_i)$. We can now address the component composition problem because it is equivalent to the component abstraction

problem. Theorem 3 shows that this composition satisfies the property of *compositionality* of interfaces. In the theorem, by "interfaces $R_1, \ldots, R_n$ are schedulable under A" we mean that resource requirements of the interfaces, given by their linear supply bound functions, are met when they are scheduled under $A$.

**Theorem 3 (Compositionality of Periodic Resource Model Interface)** *Let $R = (\Pi, \Theta/\Pi)$ denote an interface of a complex component $C = \langle \{C_1, \cdots, C_n\}, A \rangle$. Let this interface be generated by composing interfaces $R_1 = (\Pi_1, \Theta_1, \Pi_1), \ldots, R_n = (\Pi_n, \Theta_n/\Pi_n)$ of components $C_1, \ldots, C_n$, respectively. For this composition, we assume that each interface $R_i = (\Pi_i, \Theta_i/\Pi_i)$ is mapped to a periodic task $T_i = (\Pi_i, \Theta_i)$. If $\Theta/\Pi \leq 1$, then interfaces $R_1, \ldots, R_n$ are* schedulable *under $A$ on a dedicated uniprocessor.*

# 5 Incremental Schedulability Analysis

In this section we will extend the framework described in Section 4 to support incremental analysis and also to minimize resource bandwidths of interfaces. We will allow component interfaces to comprise of multiple periodic resource models for different period values. Context switch overheads incurred by components will also be abstracted in these interfaces. The framework can then select a resource model that will minimize the overall resource bandwidth required to schedule the hierarchical system. To support incremental analysis, we also describe an associative technique for composition of such interfaces.

## 5.1 Component Interface Extension

Component interfaces described in Section 4 abstract the resource requirements of components using a single periodic resource model. The resource period for this model is fixed a priori and hence the framework is unable to select a period value that will minimize the resource bandwidth of the model taking into account context switch overheads. We will modify component interfaces by allowing them to abstract components using multiple periodic resource models [6]. Each interface will consist of a set of periodic resource models for different periods. For each period, a component interface will store the minimum bandwidth required from a resource model, having that period, to schedule the component.

**Definition 7 (Extended Component Interface)** *An extended interface for a real-time component $C$ is defined as,*

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

*such that* $\forall (R = (\Pi, \Theta/\Pi)) \in I \quad lsbf_R()$ *satisfies Equation* (5) *if $C$ uses EDF scheduling policy and Equation* (6) *if $C$ uses RM scheduling policy. $P^*$ is an user defined upper bound for the period.*

Schedulability of an extended interface can be defined using periodic resource model schedulability given in Definition 6.

**Definition 8 (Extended Interface Schedulability)** *On a dedicated uniprocessor system, interface $I$ is schedulable for period value $\Pi$ (denoted as $I \stackrel{S}{\equiv} \Pi$) if and only if $((\Pi, \Theta/\Pi) \in I) \wedge (\Theta/\Pi \leq 1)$. Interface $I$ is then schedulable on a dedicated uniprocessor if and only if $\exists \Pi \in [P^*] \quad I \stackrel{S}{\equiv} \Pi$.*

## 5.2 Compact Interface

Assuming each resource model uses a constant amount of storage, size of an extended interface given by Definition 7 is $O(P^*)$. In order to reduce this storage space, we will use a compact representation for these interfaces. Let $R_i$ represent the minimum bandwidth resource model, having period value $i$, that can schedule a component $C$. Then the compact interface for $C$ will store the time instant at which $lsbf_{R_i}()$ intersects $dbf_C()$ along with the value of $dbf_C()$ at that time instant. The compact interface will store such value pairs for each period in the range 1 to $P^*$. Whenever the intersection between supply and demand functions occurs at the same time instant for different periods, compact interfaces will require reduced storage space. We show using examples that, in practice, the storage requirements of a compact interface is much smaller than $O(P^*)$.

**Definition 9 (Compact Interface)** *Compact interface representation for a component $C$ is given as,*

$$CI = \{CI_j = \langle j_{min}, j_{max}, t_j, dbf(t_j)\rangle | \forall j \quad 1 \leq j \leq k, j_{min} \leq j_{max}, j_{min} = (j-1)_{max}+1, 1_{min} = 1, k_{max} = P^*\}$$

Converting a compact interface $CI$ to a regular interface $I$ involves computing the resource bandwidths for periods ranging from 1 to $P^*$. For example, consider an element $\langle j_{min}, j_{max}, t_j, dbf(t_j)\rangle$ of the compact interface for a component $C$. Periodic resource models $R = (i, \Theta_i/i)$, where $i$ ranges from $j_{min}$ to $j_{max}$, can then be obtained using either Equation (5) if $C$ uses EDF scheduler or Equation (6) if $C$ uses RM scheduler.

### 5.3 Interface Generation for Components that use EDF Scheduler

Algorithm 5.1 generates a compact interface $CI_k$ for a simple component $C_k$ that uses EDF scheduling policy. The algorithm uses schedulability conditions given in Equation (5) to generate the interface. It computes minimum bandwidth $b_i$, that a resource model with period $i$ must have in order to schedule component $C_k$. To generate interface $CI_k$, the algorithm computes minimum bandwidths for all resource models with periods in the range 1 to $P^*$. Let *relevant* time instants denote a set of time instants at which some task in the workload of $C_k$ has a deadline. Then, $dbf_{C_k}$ changes its value only at these time instants. Also, the linear supply bound function of any resource model that schedules component $C_k$ using minimum bandwidth must intersect $dbf_{C_k}$ at one of the *relevant* time instants. Since there are $O(\mathrm{LCM}_{C_k})$ *relevant* time instants in the time interval between 0 and $\mathrm{LCM}_{C_k}$, minimum bandwidth resource model for any period can be computed in time $O(\mathrm{LCM}_{C_k})$ using Equation (5). Algorithm 5.1 hence computes interface $CI_k$ in time $O(P^*\mathrm{LCM}_{C_k})$. In our earlier work [6] we have given a more efficient algorithm that will generate the compact interface in time $O(P^* + \mathrm{LCM}_{C_k} \ln \mathrm{LCM}_{C_k})$.

Algorithm to generate compact interfaces for simple components that use RM scheduling policy can be derived from Algorithm 5.1. Schedulability conditions in Algorithm 5.1 must use the conditions given in Equation (6). Given a simple component $C_k = \langle \{T_1, \cdots, T_n\}, RM\rangle$, these conditions check for schedulability of each task $T_l$ using demand function $dbf_{C_k,l}$. Condition in Line 7 of Algorithm 5.1 must then be modified to $(t \neq t_i)||(dbf \neq dbf_{C_k,l_i}(t_i))$. At any time instant, there can be multiple values for resource demand (one for each task in the component). $dbf_{C_k,l_i}(t_i)$ denotes the current value for demand obtained from Line 6 in the algorithm. For any task $T_l$ in $C_k$ there are $O(p_l)$ *relevant* time instants in $dbf_{C_k,l}$. Hence for each period, the minimum bandwidth resource model that can schedule task $T_l$ can be computed in time $O(p_l)$.

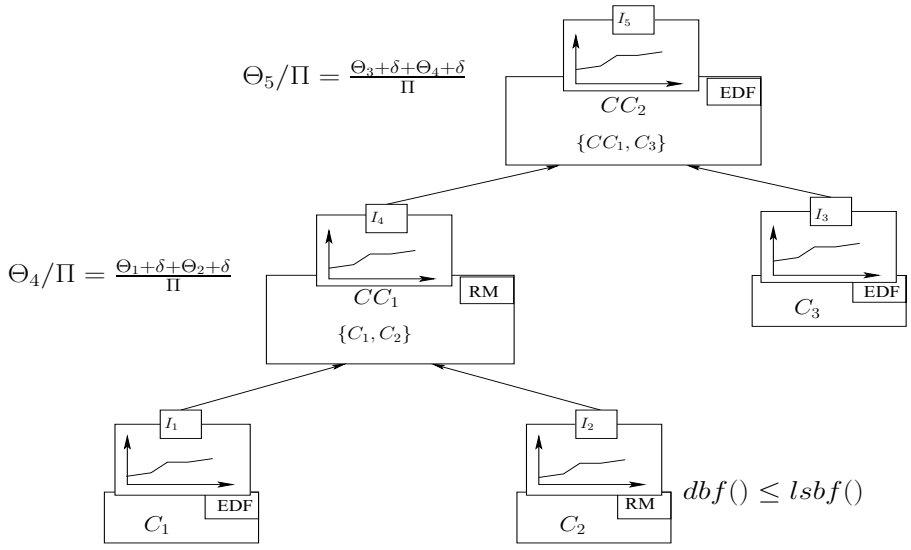
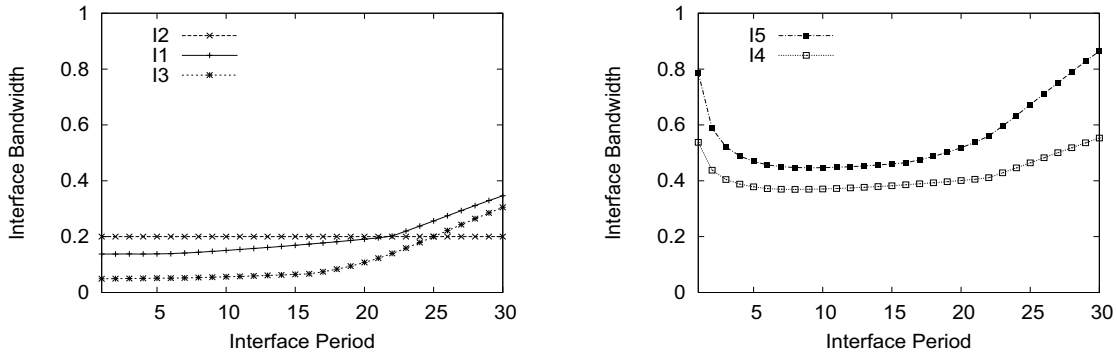



Figure 4: Extended Component Interface Model

**Algorithm 5.1** Interface Generation under EDF Scheduling Policy

**Input:** Component $C_k$ that uses EDF scheduling policy
**Output:** Compact interface $CI_k$

1: Solve Equation (5) with $\Pi = 1$ to compute minimum bandwidth $b_1$
2: Let $lsbf_{R_1}(t_1) = dbf_{C_k}(t_1)$ in Equation (5) where $R_1 = (1, b_1)$
3: Initialize $t = t_1, dbf = dbf_{C_k}(t_1), min = 1, max = 1, j = 1$
4: **for** $i = 2$ to $P^*$ **do**
5: Solve Equation (5) with $\Pi = i$ to compute minimum bandwidth $b_i$
6: Let $lsbf_{R_i}(t_i) = dbf_{C_k}(t_i)$ in Equation (5) where $R_i = (i, b_i)$
7: **if** $t \neq t_i$ **then**
8: Update $j_{min} = min, j_{max} = max$
9: Update $CI_k = CI_k \bigcup \{\langle j_{min}, j_{max}, t, dbf\rangle\}$
10: Update $min = max = max + 1, j = j + 1, t = t_i, dbf = dbf_{C_k}(t_i)$
11: **else**
12: $max = max + 1$
13: **end if**
14: **end for**




(a) Interfaces for Simple Components

(b) Interfaces for Complex Components

Figure 5: Interface Plots: Period vs. Bandwidth

Then the minimum bandwidth model with that period, which can schedule component $C_k$ can be computed in time $O(n \max_i\{p_i\})$ using Equation (6). Hence the interface generation algorithm for components that use RM scheduler can generate a compact interface in time $O(P^* n \max_i\{p_i\})$.

**Example 2** *Let component $C_1$ in Figure 4 consist of three tasks $T_1 = (45, 2), T_2 = (65, 3)$ and $T_3 = (85, 4)$, $C_2$ consist of three tasks $T_1 = \{35000, 2000\}, T_2 = \{55000, 3000\}$ and $T_3 = \{75000, 4000\}$, and $C_3$ consist of two tasks $T_1 = (45, 1)$ and $T_2 = (75, 2)$. Interfaces $I_1, I_2$ and $I_3$ are plotted in Figure 5(a) for period values in the range 1 to 30. Compact interfaces $CI_1, CI_2$ and $CI_3$ are given in Table 1 that assumes $P^* = 100000$. It is clear from Table 1 that the sizes of these compact interfaces are much smaller than $O(P^*)$. Given a period value, say $\Pi = 10$, the minimum bandwidths for the interfaces can be computed using Table 1 and Equations (5) and (6). For interface $I_1$, substituting $\Pi = 10, t = 90$ and $dbf(t) = 11$ in Equation (5) gives $b_{10} = 0.151$. Similarly, for interface $I_2$, substituting $\Pi = 10, t = 70000$ and $dbf(t) = 14000$ in Equation (6) gives $b_{10} = 0.20004$.*

## 5.4 Extended Interface Composition and Bandwidth Minimization

Extended interface for a complex component can be generated by composing the extended interfaces of components that form its workload. Interfaces generated using the algorithms given in Section 5.3 do not account for context switch overheads incurred by components. Context switch overhead for a component

| Interface $CI_1$ | | | | Interface $CI_2$ | | | | Interface $CI_3$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $j_{min}$ | $j_{max}$ | $t$ | $dbf$ | $j_{min}$ | $j_{max}$ | $t$ | $dbf$ | $j_{min}$ | $j_{max}$ | $t$ | $dbf$ |
| 1 | 1 | 9945 | 1369 | 1 | 22192 | 70000 | 14000 | 1 | 6 | 225 | 11 |
| 2 | 4 | 2210 | 304 | 22193 | 100000 | 35000 | 2000 | 7 | 16 | 90 | 4 |
| 5 | 5 | 855 | 117 | | | | | 17 | 100000 | 45 | 1 |
| 6 | 6 | 270 | 36 | | | | | | | | |
| 7 | 21 | 90 | 11 | | | | | | | | |
| 22 | 100000 | 45 | 2 | | | | | | | | |

Table 1: Compact Interfaces $CI_1, CI_2$ and $CI_3$

depends on the period of the resource model that will be used to schedule its workload. A smaller period will, in general, result in a larger number of context switches. In our framework once an interface for the root component is generated, the framework will select a value for resource period such that the corresponding resource model in the root interface has the least bandwidth among all the models in that interface. Further, all the components in the system will use resource models, having the same period value, from their respective interfaces. All the components in the workload of any component will then have the same priority under RM and EDF scheduling policies. Since within each period components will be assigned arbitrary but fixed priorities, each component will be context switched exactly once per period. The component will then incur a context switch overhead of $\delta$ in every period. Given a resource model $R = (\Pi, \Theta/\Pi)$ to schedule the component's workload, the actual bandwidth available for scheduling its workload will then be $(\Theta - \delta)/\Pi$. In other words, given a resource model $R = (\Pi, \Theta/\Pi)$ that schedules a component using minimum bandwidth, the resource model that can actually schedule the component taking into account context switch overheads is $R = (\Pi, (\Theta + \delta)/\Pi)$.

For each value of resource period, the resource model in the interface of a complex component can then be generated by adding the resource bandwidths of interfaces in its workload. Since addition is associative, this composition will support incremental analysis. Resource models of interfaces used in the composition will be modified to account for context switch overheads. For every resource model being composed we will add overhead $\delta/\Pi$ to its bandwidth, where $\Pi$ denotes the period of the model. For example, in Figure 4 interface $I_4$ for component $CC_1$ will be generated by adding the resource bandwidths of interfaces $I_1$ and $I_2$ along with appropriate context switch overheads.

**Definition 10 (Interface Composition)** *Let $I$ denote a component interface generated by composing interfaces $I_1, \cdots, I_n$. Then,*

$$I = \{(\Pi, \sum_{i=1}^{n}(\Theta_i + \delta)/\Pi) | \forall \Pi \quad 1 \leq \Pi \leq P^*, (\Pi, \Theta_i/\Pi) \in I_i\}$$

Any interface generated using Definition 10 will satisfy *compositionality* under resource period restriction, i.e., if the composed interface is schedulable using a resource model having some period value, then each of the interfaces that were used in the composition will also be schedulable as long as they use resource models with the same period value. We formalize this property in Theorem 4. This restriction forces the framework to use resource models, with identical periods, for all the interfaces in the system.

**Theorem 4 (Compositionality of Extended Interface)** *Let $I$ denote an extended interface, of a component with scheduler $A$, generated by composing extended interfaces $I_1, \cdots, I_n$ using Definition 10. If $I \overset{S}{\equiv} \Pi$, then resource models $(\Pi, \Theta_1/\Pi), \ldots, (\Pi, \Theta_n/\Pi)$ are* schedulable *under $A$ on a dedicated uniprocessor where $\forall i \quad 1 \leq i \leq n, (\Pi, \Theta_i/\Pi) \in I_i$.*

Interface composition can be applied iteratively at each level in the hierarchical system until an interface $I_r$ for the root component is generated. Then, the system is schedulable if there exists some period $\Pi$ in the range 1 to $P^*$ such that $I_r \overset{\mathrm{S}}{\equiv} \Pi$. Also, the framework will pick a value $i$ for resource period such that the corresponding resource model in $I_r$ has the least bandwidth among all resource models in $I_r$. For any other component in the system, resource model with period $i$ in the component's interface will then guarantee its schedulability (Theorem 4). For example, from Figure 5(b) which assumes $\delta = 0.1$, we get that the minimum resource bandwidth for interface $I_5$ is 0.447 and the corresponding period is 9.

# 6 Conclusion

In this paper we have summarized our work [19, 6] on schedulability analysis of hierarchical real-time systems. We abstracted components using periodic resource models and also defined composition for such

abstractions. To support incremental analysis and to minimize resource bandwidths in the presence of context switches, we extended component abstractions. Extended component interface comprised of a set of periodic resource models for different periods. This representation made it possible to determine a periodic model that minimizes the resource bandwidth for the interface taking into account context switch overheads. Composition for extended interfaces was achieved by addition of resource bandwidths of individual interfaces. This composition supports incremental analysis.

In order to make the composition associative, periodic resource models with identical periods are selected for all the interfaces in the system. It is an open problem whether incremental schedulability analysis can be achieved for hierarchical systems, using interfaces represented as periodic resource models with different periods.

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

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

[4] 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.

[5] 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.

[6] A. Easwaran, I. Shin, O. Sokolsky, and I. Lee. Incremental schedulability analysis of hierarchical real-time components. In *Proceedings of the 6th ACM International Conference on Embedded Software (EMSOFT '06)*, 2006.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

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

[15] 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.

[16] 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.

[17] J. Regehr and J. Stankovic. HLS: A framework for composing soft real-time schedulers. In *Proc. of IEEE Real-Time Systems Symposium*, pages 3–14, December 2001.

[18] 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.

[19] 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.

[20] L. Thiele, E. Wandeler, and N. Stoimenov. Real-time interfaces for composing real-time systems. In *Proceedings of the 6th ACM International Conference on Embedded Software (EMSOFT '06)*, pages 34–43, October 2006.

[21] 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.

[22] 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.