University of Pennsylvania
ScholarlyCommons

Departmental Papers (CIS) | Department of Computer & Information Science

5-7-2007

Compositional Schedulability Analysis of Hierarchical Real-Time Systems

Arvind Easwaran
University of Pennsylvania

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

Insik Shin
University of Pennsylvania

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

Copyright 2007 IEEE. Reprinted from *Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, ISORC '07*, May 2007, pages 274-281.

This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of the University of Pennsylvania's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org. By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

This paper is posted at ScholarlyCommons. http://repository.upenn.edu/cis_papers/355
For more information, please contact repository@pobox.upenn.edu.

Compositional 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 them amenable to compositional design. Real-time embedded systems consist of realtime workloads having deadlines. Compositional design of such systems can be done using 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 should be done compositionally using interfaces that abstract timing requirement of components. To facilitate analysis of dynamically changing systems, the framework should also support incremental analysis. In this paper, we overview our approach to compositional and incremental schedulability analysis of hierarchical real-time systems. We describe a compositional analysis technique that abstracts resource requirement 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 overhead of components in these extended interfaces. We then describe an associative composition technique for such interfaces, that supports incremental analysis.

Comments
Copyright 2007 IEEE. Reprinted from *Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, ISORC '07*, May 2007, pages 274-281.

This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of the University of Pennsylvania's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by writing to pubs-permissions@ieee.org. By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

This conference paper is available at ScholarlyCommons: http://repository.upenn.edu/cis_papers/355

Compositional Schedulability Analysis of Hierarchical Real-Time Systems

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

Abstract

Embedded systems are complex as a whole but consist of smaller independent modules interacting with each other. This structure makes them amenable to compositional design. Real-time embedded systems consist of real-time workloads having deadlines. Compositional design of such systems can be done using 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 should be done compositionally using interfaces that abstract timing requirement of components. To facilitate analysis of dynamically changing systems, the framework should also support incremental analysis. In this paper, we overview our approach to compositional and incremental schedulability analysis of hierarchical real-time systems. We describe a compositional analysis technique that abstracts resource requirement 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 overhead of components in these extended 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 this component-based design, schedulability analysis should be addressed for component-based real-time systems. It is desirable to achieve schedulability analysis *compositionally* by combining component interfaces that abstract component-level timing requirement. Ideally, these abstractions must satisfy the timing requirement of components using minimum resource supply.

Component-based real-time systems often involve hierarchical scheduling frameworks for supporting hierarchical resource sharing among components having varied priorities and under different schedulers. 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 for the workload. In this framework, resources are allocated from a parent node to its children. Also, in such frameworks a component is typically an open system, i.e., it can be an element in the workload of another component. System may then comprise of components with partially specified workload, i.e., components that are not yet closed. The unspecified elements in the workload may be added to the system, on the fly, in some arbitrary order. Interface generated for a component must then be independent of the order in which elements are added to its workload. Analysis frameworks that possess this property are said to be *incremental*.

In this paper we summarize our work [16, 7] on schedulability analysis of hierarchical real-time systems. Our interface model is based on the periodic resource model [16, 12, 7], 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 [16] 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 and independent tasks scheduled under RM or EDF scheduler. We then describe an interface composition technique that facilitates compositional analysis for such in-

Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE

terfaces. However, this framework neither supports incremental analysis nor can it minimize the resource bandwidth required to schedule the system taking into account context switch overhead of components. We then describe an extended interface model [7] 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, 8, 15, 12, 1, 2, 13, 20, 18] 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] 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 [14, 8]. 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 [16, 12]. There have been studies [15, 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.* [15] 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 [13] considered the issue of addressing the component abstraction problem in the presence of interacting tasks within a component. Davis and Burns [3] presented the Hierarchical Stack Resource Policy (HSRP) for supporting mutually exclusive resource sharing between components in a hierarchical scheduling framework. These techniques as well, do not support incremental schedulability analysis.

There have been studies on the development of interface theory for supporting incremental analysis of component-based real-time systems, applying the interface theory [4] and network calculus [17] into real-time context. These studies have proposed assume-guarantee interfaces for real-time components, with a generic real-time interface model [19, 20, 18] and with a bounded-delay resource partition interface model [9], to support interfaces towards compositional and incremental schedulability analysis. Their techniques support compositional analysis for components with EDF or RM scheduler. However, their techniques for supporting incremental analysis have been restricted to components with EDF scheduler [20, 18, 9], i.e., they are not applicable to components with RM scheduler. Also, these approaches do not address the problem of minimizing the resource bandwidth requirement of interfaces taking into account context switch overhead of components.

The rest of the paper is organized as follows: Section 2 defines our system model and the schedulability analysis problem that we address. Section 3 gives schedulability conditions for components scheduled using periodic resource models. Sections 4 and 5 describe our work on compositional and incremental schedulability analyses, respectively, of component-based hierarchical real-time systems. 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 period. For schedulability analysis using our approach, the component must export its worst case resource demand which depends on the task model and 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 component C is specified as $C = \langle\{T_1 = (p_1, e_1), \ldots, T_n = (p_n, e_n)\}, A\rangle$, where each T_i is a real-time task with period p_i and worst case execution time e_i, and A is a scheduler.*

Definition 2 (Complex Component) *A complex component C is specified as $C = \langle\{C_1, \ldots, C_n\}, A\rangle$, where each*

Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE
IEEE COMPUTER SOCIETY

C_i is a simple/complex component different from C, and A is a scheduler.

We use the term component to refer to both simple as well as complex components. The context should make the meaning clear and we 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 system, where C_4 is a complex component composed from components C_1 and C_2, and C_5 is a complex component consisting of components C_3 and C_4. For compositional schedulability analysis of such a system, resource demand of components in the system must be abstracted into interfaces. These interfaces must be such that if a component interface is schedulable then the corresponding component must also be schedulable. Here, an interface or a component is schedulable means that its resource requirement can be satisfied on a dedicated uniprocessor. Furthermore, 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* which can be defined as follows.

Definition 3 (Compositionality of Interface) *Let $C = \langle \{C_1, \ldots, C_n\}, A \rangle$ denote a complex component with 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 property of* compositionality *if and only if whenever I is* schedulable*, interfaces $I_1, \ldots, I_n$ are also* schedulable *under A.*

Real-time components in a hierarchical system can be modified on the fly. These modifications could happen either due to desired customizations, or due to changes in the operating environment of the system. A hierarchical system can be modified either by adding new components to it or by removing existing components from it. In this paper, we assume that these are the only modifications that can be done. Analysis frameworks for such a dynamically changing system must then support incremental schedulability analysis. In an incremental framework, interface generated for a complex component is independent of the order in which interfaces of components in its workload are composed. If a framework is not incremental, then it can generate two different interfaces for the same complex component, and this is undesirable.

Definition 4 (Incremental Analysis) *Let $C = \langle \{C_1, \ldots, C_n\}, A \rangle$ denote a complex 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 $\{1, \ldots, n\}$. Also, for each $\sigma \in \mathcal{P}$, let I_σ denote an interface for C generated by composing interfaces $I_1, \ldots, I_n$ such that the order of composition is given by σ. Then, this composition supports incremental analysis if and only if $I_{\sigma_1} = I_{\sigma_2}$, for all $\sigma_1, \sigma_2 \in \mathcal{P}$.*

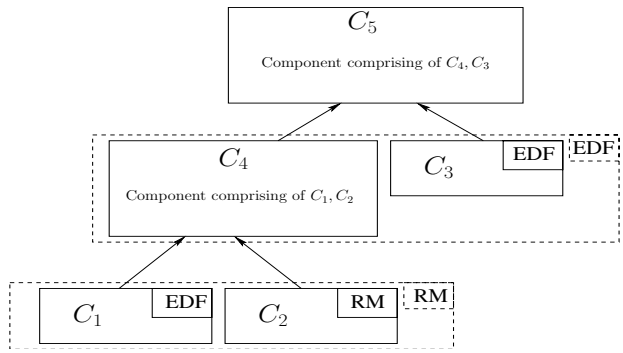


Figure 1. Hierarchical Real-Time System

Context switches play an important role in schedulability analysis because they consume real-time resources. Therefore, in our analysis framework we must account for the context switch overhead incurred by components. In a hierarchical system, context switches can occur at each level of the hierarchy. Since our focus in this work is on component abstraction and composition problems, we ignore context switch overhead incurred as a result of scheduling tasks within a simple component. Existing techniques [6] can be used in our framework to upper bound this overhead. Hence, we only consider context switch overhead incurred by components when they are scheduled among themselves. For example, in Figure 1 we only consider context switches that occur when components C_1 and C_2 or C_4 and C_3 are scheduled together.

The schedulability analysis problem that we address in this paper can be stated as follows: Given a hierarchical real-time system,

1. Generate an interface for each simple component such that if the interface is schedulable, then the component is also schedulable. This interface must account for the context switch overhead incurred by the component.

2. Generate an interface for each complex component by composing interfaces of its workload, such that the composed interface satisfies *compositionality*. This composed interface must account for the context switch overhead incurred by the component and the composition must support incremental analysis.

Finally, the framework must minimize resource bandwidth required to schedule the hierarchical system, when analysis is done using this interface model.

Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE

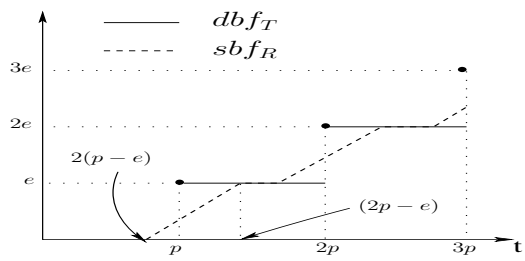


Figure 2. Periodic Demand and Supply: $T = (p, e)$**,** $R = (p, e/p)$

3. Schedulability Conditions

A real-time task consists of a set of real-time jobs that are required to meet temporal constraints. The resource demand bound function (dbf : Real $\rightarrow$ Real) of a real-time task upper bounds the amount of computational resource required to meet all of its temporal constraints. For a time interval length t, the demand bound function of a task gives the largest resource demand of the task in any time interval of length t. For example, Figure 2 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 its deadlines. Similar to the demand bound function of a task, 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. Note that for a component, we need to consider its scheduling policy in computing its demand bound function.

In our earlier work [16], 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 underlying system. For example, a periodic resource supply that provides Θ units of resource every Π units of time can be represented using the periodic resource model $R = (\Pi, \Theta/\Pi)$. Here, Θ/Π represents the resource bandwidth for model R. The supply bound function (sbf : Real $\rightarrow$ Real) of a resource model lower bounds the amount of resource that the model supplies. That is, given a resource model R and interval length t, $sbf_R(t)$ 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), proposed by Shin and Lee [16], 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 shows the supply bound function sbf_R for a periodic model $R = (p, e/p)$.

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

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

In this paper, we limit to periodic resource models as abstractions for the resource requirement of components. 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. The demand and supply bound functions of C and R, respectively, can then be used to define these schedulability conditions. Theorems 1 and 2 give schedulability conditions under EDF and RM schedulers, respectively [16]. Here, LCM_C denotes the least common multiple of periods of all the periodic tasks in component C.

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*

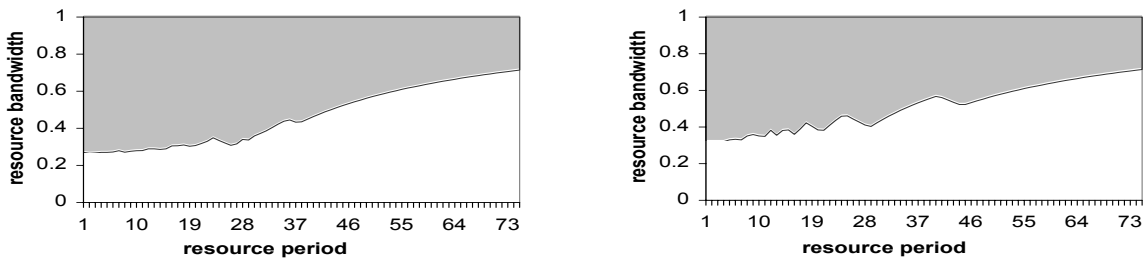
$$\forall t \text{ s.t. } 0 < t \leq LCM_C, dbf_C(t) \leq lsbf_R(t) \tag{5}$$

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*

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

The interface generation algorithms described in this paper 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 these algorithms tractable.

Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE



(a) Under EDF Scheduler

(b) Under RM Scheduler

Figure 3. Schedulable Region of Periodic Resource Model $R(\Pi, \Theta/\Pi)$ in Example 1

4. Compositional Schedulability Analysis

In our paper [16], 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 requirement of the component as a periodic resource model $R = (\Pi, \Theta/\Pi)$, where Π is user defined. This interface does not reveal the internal information of the component such as the number of elements in its workload or its scheduling algorithm. Note that 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. More specifically, given a simple component $C = \langle W, A\rangle$ and period Π, find a periodic resource model interface $R = (\Pi, \Theta/\Pi)$ that can schedule C and is "optimal." Here, optimality is with respect to minimizing the resource 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* Π, *we can find the smallest* Θ/Π *such that component* C *is schedulable according to Theorem 1 or Theorem 2. Figure 3 shows such a solution space as the gray area for resource periods in the range* 1 *to* 75, *under both RM and EDF schedulers. For instance, when* $\Pi = 10$, *the minimum resource bandwidth that guarantees schedulability of* C *is* 0.28 *under EDF scheduler or* 0.35 *under RM scheduler.*

We define the *(periodic) component composition* problem as the problem of generating an interface for a complex component from the interfaces of sub-components. More specifically, given a complex component $C = \langle\{C_1, \cdots, C_n\}, A\rangle$ and period Π, find a periodic resource model interface $R = (\Pi, \Theta/\Pi)$ that can schedule C and is "optimal." 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. We then treat C as consisting of n periodic tasks such that $C = \langle\{T_1 = (p_1, e_1), \cdots, T_n = (p_n, e_n)\}, A\rangle$, where for each i, $1 \leq i \leq n$, $R_i = (\Pi_i, \Theta_i/\Pi_i)$ implies $T_i = (\Pi_i, \Theta_i)$. We can now address this 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 the resource requirement of these interfaces, represented by their linear supply bound functions, are satisfied under scheduler A.

Theorem 3 *Let* $C = \langle\{C_1, \cdots, C_n\}, A\rangle$ *denote a complex component and let* $R_1 = (\Pi_1, \Theta_1/\Pi_1), \ldots, R_n = (\Pi_n, \Theta_n/\Pi_n)$ *denote interfaces of components* $C_1, \ldots, C_n$, *respectively. Consider a periodic task set* $\{T_1, \ldots, T_n\}$ *where each* $T_i = (\Pi_i, \Theta_i)$. *Let* $R = (\Pi, \Theta/\Pi)$ *and the task set* $\{T_1, \ldots, T_n\}$ *satisfy Theorem 1 if* A *is EDF or Theorem 2 if* A *is RM. Now, if* $\Theta/\Pi \leq 1$, *then interfaces* $R_1, \ldots, R_n$ *are* schedulable *under* A.

Proof Demand bound function dbf_T for a periodic task $T = (p, e)$ is given as $dbf_T(t) = \lfloor t/p \rfloor e$. Then from Equation (4), we get that for each i, $1 \leq i \leq n$, $dbf_{T_i}(t) \geq lsbf_{R_i}(t)$ for all t. This theorem can then be trivially proved using Theorem 1 if A is EDF, or Theorem 2 if A is RM. □

5. Incremental Schedulability Analysis

In this section, we extend the framework described in Section 4 to support incremental analysis and also to minimize resource bandwidths of interfaces. Our approach is to allow component interfaces to comprise of multiple periodic resource models for different period values. We also include context switch overhead incurred by components in these interfaces. We can then select a resource model that minimizes the overall resource bandwidth required to

Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE

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 requirement of components using a single periodic resource model. We assume that the resource period for this model is fixed a priori, and hence, the framework is unable to select a period value that minimizes the resource bandwidth of the model while taking into account context switch overhead. Note that if we do not consider context switch overhead, a smaller period requires less resource bandwidth. So, we extend component interfaces to include multiple periodic resource models [7]. Each interface then consists of a set of periodic resource models for different periods.

Definition 5 (Extended Component Interface) *An extended interface can be specified as,*

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

where P^* is an user-defined upper bound for the period.

For each period value, a component interface includes a resource model having minimum bandwidth that can schedule the component. Schedulability of an extended interface depends on schedulability of resource models in the interface. On a dedicated uniprocessor system, an interface I *can be scheduled* with period Π if and only if $\Theta/\Pi \leq 1$, where $(\Pi, \Theta/\Pi) \in I$. Interface I is then *schedulable* on a dedicated uniprocessor if and only if $\exists \Pi \in \{1, \ldots, P^*\}$ such that I *can be scheduled* with period Π. Furthermore, if a component interface I is *schedulable* then the corresponding component must also be schedulable. This can be ensured if resource models in the interface satisfy schedulability conditions given in Section 3, i.e., for all $R \in I$, $lsbf_R$ must satisfy Equation (5) if the component uses EDF scheduler or Equation (6) if it uses RM scheduler.

5.2. Extended Interface Generation

Algorithm 1 generates an extended interface I_k for a simple component C_k that uses EDF scheduler. The algorithm uses schedulability conditions given in Equation (5) to generate the interface. It computes the minimum bandwidth b_i that a resource model with period i must have in order to schedule component C_k. To generate interface I_k, the algorithm computes such minimum bandwidth 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

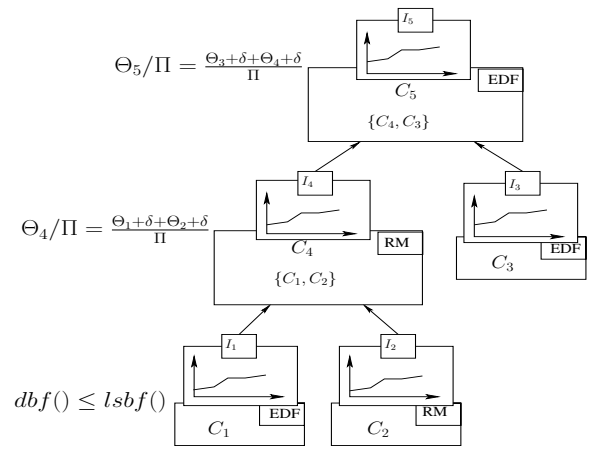


Figure 4. Extended Component Interfaces

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 $O(\text{LCM}_{C_k})$ *relevant* time instants in the time interval between 0 and LCM_{C_k}, minimum bandwidth resource model for any period can be computed in $O(\text{LCM}_{C_k})$ time using Equation (5). Therefore, Algorithm 1 computes interface I_k in $O(\text{LCM}_{C_k} \times P^*)$ time. In our earlier work [7] we have given a more involved, but efficient algorithm that can generate the interface in time $O(P^* + \text{LCM}_{C_k} \times \ln \text{LCM}_{C_k})$. In this paper, we describe a simpler but less efficient algorithm to simplify the presentation.

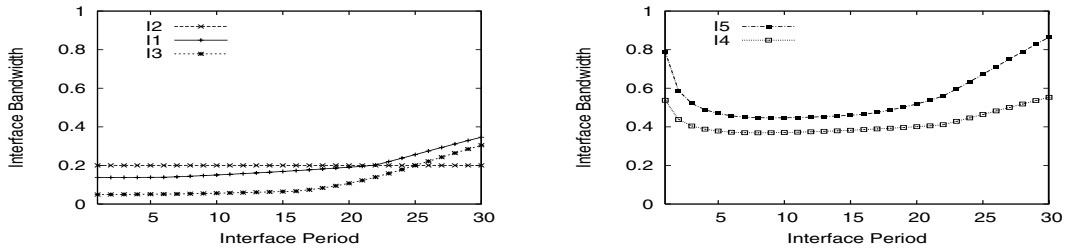
Algorithm 1 Interface Generation under EDF Scheduler

Input: Component $C_k = \langle \{T_1, \ldots, T_n\}, \text{EDF} \rangle$
Output: Extended interface I_k

```
for i = 1 to P* do
    Solve Equation (5) with Π = i to compute Θ
    Set R_i = (i, b_i = Θ/i) to denote the minimum bandwidth resource model for period i
    Update I_k = I_k ⋃ {R_i}
end for
```

Algorithm to generate compact interfaces for simple components that use RM scheduler can be derived from Algorithm 1. Schedulability conditions in Algorithm 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_i using the demand bound function $dbf_{C_k,i}$. For any task T_i in C_k there are $O(p_i)$ *relevant* time instants in $dbf_{C_k,i}$. Hence, for each period, the minimum bandwidth resource model that can schedule task T_i can be computed in $O(p_i)$ time. Then, the minimum bandwidth model with that period which can schedule component C_k can be computed in $O(\max_l \{p_l\} \times$

Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE



(a) Interfaces for Simple Components (C_1, C_2, C_3)

(b) Interfaces for Complex Components (C_4, C_5)

Figure 5. Interface Plots: Period vs. Bandwidth

n) time using Equation (6). Hence the interface generation algorithm for components that use RM scheduler can generate an extended interface in $O(\max_l\{p_l\} \times n \times P^*)$ time.

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 $T_2 = (75, 2)$. Extended interfaces I_1, I_2 and I_3 for components C_1, C_2 and C_3, respectively, are plotted in Figure 5(a) for resource periods in the range* 1 *to* 30.

5.3. 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 algorithm given in Section 5.2 do not account for context switch overhead incurred by components. Context switch overhead for a component depends on the period of the resource model that will be used to schedule it. 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 selects 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. Furthermore, all the components in the system are scheduled using resource models that have the same chosen period, from their respective interfaces. All the components in the workload of any complex component then have a single priority under both RM, as well as, EDF scheduler. Hence, each component is context switched exactly once per resource period in our framework. If δ denotes the execution overhead incurred by the system for each context switch, then a component scheduled using resource model $R = (\Pi, \Theta/\Pi)$ incurs a context switch overhead of δ in every Π time units. This means that given a resource model $R = (\Pi, \Theta/\Pi)$ to schedule a component, the actual bandwidth available for scheduling the component is $(\Theta - \delta)/\Pi$.

In our framework, we compose interfaces by adding the bandwidths of resource models in the interfaces. For each value of resource period, the resource model in the interface of a complex component is generated by adding the bandwidths of resource models in the interfaces of its workload. Since addition is an associative operation, this composition supports incremental schedulability analysis. Prior to composition, resource models of interfaces are modified to account for context switch overhead. For every resource model being composed we add overhead δ/Π to its bandwidth, where Π denotes the period of the model. For example, in Figure 4 interface I_4 for component C_4 is generated by adding the resource bandwidths of interfaces I_1 and I_2 along with appropriate context switch overheads.

Definition 6 (Interface Composition) *Interface I generated by composing interfaces $I_1, \ldots, I_n$ is given as,*

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

where for each period Π, $(\Pi, \Theta_i/\Pi) \in I_i$ for all $i = 1, \ldots, n$.

Any interface generated using Definition 6 satisfies *compositionality* only under resource period restriction, i.e., if a composed interface is schedulable using a resource model having period i, then each of the interfaces that were used in the composition are also schedulable as long as they use resource models having period i. This restriction, formally stated in Theorem 4, forces the framework to schedule all the components in the system using resource models that have the same period.

Theorem 4 (Compositionality of Extended Interface) *Let I denote an extended interface of a component with scheduler A, and generated by composing extended interfaces $I_1, \ldots, I_n$ using Definition 6. If I* can be scheduled *with period Π, then resource models $(\Pi, \Theta_1/\Pi), \ldots, (\Pi, \Theta_n/\Pi)$ are* schedulable *under A on a dedicated uniprocessor where for each i, $1 \leq i \leq n$, $(\Pi, \Theta_i/\Pi) \in I_i$.*

Proceedings of the 10th IEEE International Symposium on
Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)
0-7695-2765-5/07 $20.00 © 2007 IEEE

Proof Obvious from Definition 6. □

Interface composition can be applied iteratively in the hierarchical system until an interface I_r for the root component is generated. Then, the system is schedulable if I_r is *schedulable*. Also, the framework picks 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 its interface then guarantees schedulability of the component (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 resource period is 9. Then, components C_1, C_2, C_3 and C_4 are all schedulable using resource models having period 9 from interfaces I_1, I_2, I_3 and I_4, respectively.

6. Conclusion

In this paper, we have summarized our work [16, 7] 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 bandwidth 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 while taking into account context switch overhead of the component. Composition for extended interfaces was achieved by addition of resource bandwidths of individual interfaces. This composition supports incremental analysis.

In the future, we aim to extend this framework so that it can support analysis of hierarchical systems with inter-component task dependencies. Furthermore, we would also like to reduce the abstraction, as well as, composition overheads incurred by our framework.

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] R. I. Davis and A. Burns. Resource sharing in hierarchical fixed priority pre-emptive systems. In *Proc. of IEEE Real-Time Systems Symposium*, December 2006.

[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, I. Lee, and O. Sokolsky. Bounding preemptions under EDF and RM schedulers. Technical Report MS–CIS–06–07, University of Pennsylvania, USA, 2006.

[7] 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.

[8] 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.

[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. 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] 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.

[14] 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.

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

[16] 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.

[17] L. Thiele, S. Chakraborty, M. Gries, and S. Künzli. A framework for evaluating design tradeoffs in packet processing architectures. In *Proceedings of the 39th Design Automation Conference (DAC 2002)*, 2002.

[18] 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.

[19] 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.

[20] 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.