Removing Abstraction Overhead in the Composition of Hierarchical Real-Time Systems*

Sanjian Chen Linh T.X. Phan Jaewoo Lee Insup Lee Oleg Sokolsky
Department of Computer and Information Science, University of Pennsylvania
Email: {sanjian, linhphan, jaewoo, lee, sokolsky}@cis.upenn.edu

***Abstract*—The hierarchical real-time scheduling framework is a widely accepted model to facilitate the design and analysis of the increasingly complex real-time systems. Interface abstraction and composition are the key issues in the hierarchical scheduling framework analysis. Schedulability is essential to guarantee that the timing requirements of all components are satisfied. In order for the design to be resource efficient, the composition must be bandwidth optimal. Associativity is desirable for open systems in which components may be added or deleted at run time. Previous techniques on compositional scheduling are either not resource efficient in some aspects, or cannot achieve optimality and associativity at the same time. In this paper, several important properties regarding the periodic resource model are identified. Based on those properties, we propose a novel interface abstraction and composition framework which achieves schedulability, optimality, and associativity. Our approach eliminates abstraction overhead in the composition.**

***Keywords*-Real-time embedded systems, compositional scheduling, bandwidth-optimal interface, abstraction overhead.**

I. Introduction

Real-time embedded systems are composed of computing resources, real-time tasks, and scheduling policies. Such systems are becoming highly complex due to increasing computational demand. However, the amount of resources available in most real-time applications, e.g., wireless sensor networks and automobiles, are constrained by other design factors such as size, weight, and power. Therefore, resource-efficient solutions are crucial for real-time embedded systems.

Component-based design has been widely accepted as a paradigm to facilitate the design of complex real-time systems [1]–[3]. In the component-based approach, a complex real-time system is decomposed into multiple simple *components*, each of which contains a workload consisting of a real-time task set and a scheduling policy. Schedulability analysis is then performed within each component, and an *interface* is abstracted for each component. The interface of a component represents the collective resource demand of the workload without revealing the detailed information about the task set and the scheduling policy. A real-time interface is often represented by a *resource model*, which is an abstract characterization of a resource supply pattern, and system-level analysis is done by interface composition. Component-based design is incorporated into the compositional scheduling framework, where components are arranged in a tree so that the workload of a component is either a real-time task set generated by applications, or a set of interfaces of the sub-components. In this paper, we distinguish those two kinds of components in a scheduling tree as: leaf components, whose workloads consist of real-time task sets generated by applications, and intermediate components, whose workloads consist of interfaces of the sub-components. Figure 1 gives an example of hierarchical scheduling frameworks.

Compositional scheduling framework design requires that properties established for each component are preserved at all levels of the tree. A basic property for all real-time systems is that components should be schedulable, i.e., the timing requirements of the workloads should be satisfied under the scheduling policies. In the scheduling tree, the decomposition and integration of components should maintain *schedulability* in the sense that each component is schedulable if its interface is schedulable by its immediate parent component. In this paper, the interfaces are represented as *periodic resource models (PRMs)* as introduced in [4], [5]. A PRM (Π, Θ) guarantees Θ units of resource supply for every Π time units on a uniprocessor platform, and its *bandwidth* is $\frac{\Theta}{\Pi}$. The semantics of PRMs are supported by many existing real-time schedulers, and a PRM can be generically transformed into a periodic real-time task, which is important for the analysis of component-based hierarchical scheduling frameworks.

The minimum bandwidth needed by the root component represents the overall resource requirement of the system. For the framework to be resource efficient, the root bandwidth should be minimized. In this paper, we address the *optimality* issue in terms of bandwidth, i.e., to find the minimum-bandwidth interface for the root component. In a PRM-based hierarchical system, finding the optimal root interface can be broken down into two sub-problems: identifying bandwidth-optimal PRMs for each leaf component, and minimizing the compositional overhead. The first problem has been addressed in previous studies, such as [2]. For the latter problem, the component-abstraction overhead has been defined in [2] as $U_P/U_W - 1$, where U_P and U_W are bandwidth of the interface and workload of a component, respectively. In this paper, we propose an approach to eliminate this abstraction overhead incurred by the composition. Preemption overhead is not addressed in the current framework.

In *open* real-time systems, components may be added or deleted at run time. An example is the integrated medical device systems in which devices may be activated and deac-

*This research was supported in part by NSF CNS-0931239, NSF CNS-0930647, NSF CNS-0834524, and NSF CNS-0720518.

tivated according to patients' physiological states that change over time. Schedulability of open systems can be analyzed more efficiently if the composition is *associative*, i.e., for a given set of leaf components, the interface of the root component does not depend on the order in which the leaf components are composed. With associativity, when a component C' is added or deleted, the new optimal root interface can be directly calculated from the interface of C' and the previous root interface. In contrast, without associativity, in order to obtain the new optimal root interface, the interface abstraction and composition need to be redone for every component in the system.

Related Work. Compositional scheduling frameworks have garnered growing attention in the real-time community [1]–[18]. A two-level scheduling framework was first introduced by Deng and Liu [19], and the schedulability analysis has been done for both fixed-priority schedulers [20] and dynamic-priority schedulers [21], [22]. Mok and Feng [6], [23] proposed a bounded-delay resource model. Component interface abstraction and composition techniques for the bounded-delay resource model were later developed in [8]. None of these approaches, however, is associative.

The PRM was introduced as an alternative interface representation in [2], [4], [5]. The interface abstraction and composition for PRMs have been addressed under fixed-priority and dynamic-priority schedulers in [2], [4], [5], [7], [9], [10]. Again, none of the composition methods is associative. The Explicit Deadline Periodic (EDP) model has been proposed in [16] as an extension to the original PRM, but the composition is not associative either. In addition, there have been studies [13]–[15] on incremental design based on interface theory [24], [25]. An interface representation with multiple choices of periods and an associative composition technique were proposed in [1]. However, the schedulability condition used by [1] is sufficient but not necessary, which incurs additional abstraction overhead in the composition.

In previous studies on PRM-based compositional framework analysis [2], the sub-interfaces are composed by a 2-step approach: first, each sub-interface is transformed into a real-time task; then the composed interface is computed as a resource model that can schedule those tasks. At the composition step it was assumed that the first jobs of the tasks may be released at arbitrary time instances. However, the incremental analysis proposed in [1] implicitly dropped this assumption as the proposed composition method is correct only if the first job release of each task is synchronized with the start time of the resource supply. Neither the synchronization assumption nor its impact on bandwidth optimality was explicitly elaborated on in [1]. In this paper, we propose the synchronized release times assumption for interfaces and show that the arbitrary release times assumption made in the previous study is not only unnecessary but also incurs additional component abstraction overhead in the composition.

Contributions. Our main contributions include:

- We identify several important properties regarding the supply bound function of a PRM. Based on those properties, we define a function called Γ which generates a set of PRMs from any given single PRM Ω', each of which has the same bandwidth as Ω' and guarantees schedulability. Using Γ, we develop a succinct interface that represents an infinite number of candidate PRMs with only a few variables.
- We propose a new interface abstraction and composition framework that achieves schedulability, optimality, and associativity. The composition incurs no abstraction overhead.
- Based on inferences drawn from [1], we show that the arbitrary release times assumption for the sub-interfaces in the workloads of intermediate components is unnecessary. We further show that the start times of all the interfaces can be aligned without changing the schedulability of the corresponding components.

The analysis is performed on the interface level, and the proposed framework is generally applicable to any real-time system in which the workload and resource can be characterized by a demand-bound function and a supply-bound function, respectively.

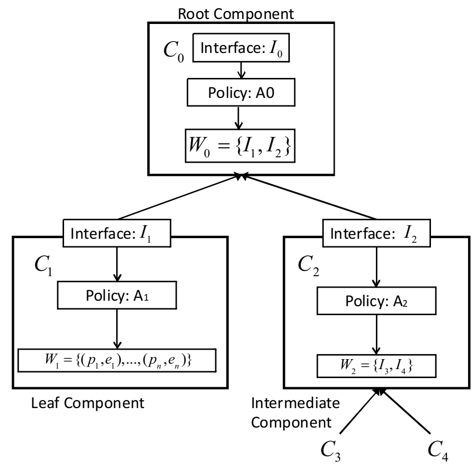


Fig. 1. Compositional scheduling framework.

II. Background and Our Approach

The real-time task sets discussed in this paper are characterized by the demand-bound function $\mathrm{dbf}(t)$, which gives the maximum possible resource demand of a single task or a task set within any time interval of length t [26], [27]. For example, a well-known workload model is periodic tasks, which are sequences of jobs released in a periodic fashion. For periodic, independent, and preemptable tasks, it is known that the earliest-deadline-first (EDF) algorithm is an optimal dynamic-priority scheduling policy and the rate-monotonic (RM) algorithm is an optimal fixed-priority scheduling policy [28]. The demand-bound function of a periodic task set $T = \{T_1 = (p_1, e_1), \dots, T_n = (p_n, e_n)\}$ scheduled under EDF is given by [26] as Equation 1, where p_i and e_i denote the period and the worst-case execution time of T_i, respectively.

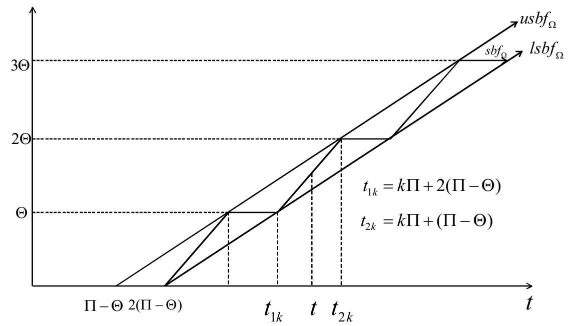


Fig. 2. sbf, lsbf, and usbf of a PRM $\Omega=(\Pi,\Theta)$ [5]

The demand-bound function for a task T_i in the periodic task set T scheduled under RM is given by [27] as Equation 2, in which $HP_T(i)$ is the tasks in T that have higher priorities than T_i.

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

$$\text{dbf}_T(t,i)=e_i+\sum_{T_k\in HP_T(i)}\lceil\frac{t}{p_k}\rceil e_k \tag{2}$$

A real-time component consists of a real-time workload, a scheduling policy, and a real-time interface.

Definition 2.1 (Real-time Component): A real-time component C is defined as $C=\langle W,I,A\rangle$, wherein W is either a set of real-time tasks or a set of interfaces of sub-components, I is an interface, and A is a scheduling policy.

The *interface I* represents the collective resource demand of W in order for W to be schedulable under A. In this paper, we focus on PRM-based interfaces. The supply bound function sbf(t), which gives the minimum supply of a PRM Ω over any time interval t, is given by [2] as Equation 3, where $x=2(\Pi-\Theta)$ and $y=\left\lfloor\frac{t-(\Pi-\Theta)}{\Pi}\right\rfloor$. The functions lsbf$_\Omega$ and usbf$_\Omega$, the tight linear lower and upper bounds for sbf$_\Omega$, are given by [2] as Equations 4 and 5.

$$\text{sbf}_\Omega(t)=\begin{cases}y\Theta+\max\{0,t-x-y\Pi\} & t\geq\Pi-\Theta\\ 0 & \text{Otherwise}\end{cases} \tag{3}$$

$$\text{lsbf}_\Omega(t)=\max(\frac{\Theta}{\Pi}(t-2(\Pi-\Theta)),0) \tag{4}$$

$$\text{usbf}_\Omega(t)=\max(\frac{\Theta}{\Pi}(t-(\Pi-\Theta)),0) \tag{5}$$

Figure 2 shows the sbf, lsbf and usbf of a PRM $\Omega=(\Pi,\Theta)$. As shown in the figure, the *starvation time* of Ω, the longest time interval during which there is no resource supply in the worst case, is $2(\Pi-\Theta)$, i.e., $\forall t\leq 2(\Pi-\Theta),\text{sbf}_\Omega(t)=0$. From Equations (3), (4), and (5), it can be calculated that lsbf and usbf intersect sbf at time points defined by $t_{1k}=k\Pi+2(\Pi-\Theta)$ and $t_{2k}=k\Pi+(\Pi-\Theta)$, in which $k\in\mathbb{N}$, respectively.

A component $C=\langle W,I,A\rangle$ is *schedulable* if and only if any resource model Ω represented in I satisfies the timing requirements of W under A. The schedulability condition for a periodic task set scheduled under EDF and RM are given in [2]:

Theorem 2.2 (Schedulability of EDF/RM [2]): A real-time workload $W=\{T_1=(p_1,e_1),\ldots,T_n=(p_n,e_n)\}$ is schedulable by a resource model Ω under EDF, if and only if $\forall 0<t\leq LCM_W$ $\text{dbf}_W(t)\leq\text{sbf}_\Omega(t)$, where LCM_W is the least common multiple of p_i for all $T_i\in W$. W is schedulable by Ω under RM, if and only if $\forall T_i\in W$ $\exists t_i\in[0,p_i]$ $\text{dbf}_W(t_i,i)\leq\text{sbf}_\Omega(t_i)$

For both EDF and RM, the schedulability is checked based on the comparison between the dbf and sbf. It immediately follows that:

Lemma 2.3: Given two PRMs $\Omega_1=(\Pi_1,\Theta_1)$ and $\Omega_2=(\Pi_2,\Theta_2)$, if $\forall t,\text{sbf}_{\Omega_1}(t)\geq\text{sbf}_{\Omega_2}(t)$, then for a workload W which consists of periodic tasks, Ω_1 can schedule W under EDF/RM if Ω_2 can schedule W under EDF/RM.

Proof: Suppose Ω_2 can schedule W under EDF. Then, $\forall t$, $\text{dbf}_W(t)\leq\text{sbf}_{\Omega_2}(t)\leq\text{sbf}_{\Omega_1}(t)$. Hence Ω_1 can schedule W. The same reasoning applies for RM. ■

This lemma defines a partial order of scheduling capability between two PRMs. If the condition stated in the lemma holds, then Ω_1 is said to be *better than* Ω_2. It should be noted that the partial order applies to any real-time systems in which the schedulability is expressed in terms of the sbf being the upper bound of a resource demand characteristic function of the workload. The analysis in Section III is based on the partial order defined here and does not make any assumptions on how a dbf is calculated from specific scheduling policies and task models. Therefore our work is applicable to a wide range of real-time systems.

We define the notion of *bandwidth B* as follows. The bandwidth of a resource model $\Omega=(\Pi,\Theta)$ is $B=\frac{\Theta}{\Pi}$. In our framework, an interface is represented by a set of PRMs that share the same bandwidth, which is referred to as the bandwidth of the interface. In a leaf component, the bandwidth of a workload W is equal to the total bandwidth of the sub-interfaces. In a leaf component, the bandwidth of a workload $W=\{(p_1,e_1),\ldots,(p_n,e_n)\}$ is $B=\sum_i\frac{e_i}{p_i}$. Abstraction overhead in composition is eliminated if for any intermediate component $C=\langle W,I,A\rangle$, the bandwidth of I is equal to the bandwidth of W.

Our Approach. Given a hierarchical real-time system in which the workloads of all the leaf components are known, assume the schedulability condition is expressed in terms of the sbf being the upper bound of the dbf, we do the following:

- For each leaf component, abstract its resource demand by an interface such that the component itself is schedulable if its interface is schedulable by the parent component.
- For each intermediate component, compose its sub-interfaces into one single interface while preserving schedulability.
- Justify the schedulability, optimality, and associativity of the proposed framework.
- Evaluate the approach by comparing it with previous techniques.

III. Interface Abstraction and Composition

In this section, we propose our approaches for interface abstraction and composition. We first identify several key properties regarding the supply bound function of a PRM and develop the schedulability condition for synchronous periodic tasks where all tasks and resource models share the same period and are first released at the same time. Based on these results, we then propose our interface abstraction and composition framework and show that it achieves schedulability, optimality, and associativity.

A. Bandwidth Equivalent Interface Class

Recall Lemma 2.3, which defines a partial order between two PRMs. It is trivially true that for any fixed period Π, a PRM (Π,Θ) with greater bandwidth always has a better scheduling capability. However, to achieve bandwidth optimality, it is desirable to enhance the scheduling capability of a resource model without increasing its bandwidth. Thus, an interesting question is: given a PRM $\Omega=(\Pi,\Theta)$, are there other PRMs with the same bandwidth but better than Ω? This section gives a necessary and sufficient answer to the above question, which – to the best of our knowledge – has not been addressed before.

Intuitively, for a fixed bandwidth, a PRM with a shorter period could be better than one with a longer period. Suppose $\Omega_l=(\Pi_l,\Theta_l)$ and $\Omega_s=(\Pi_s,\Theta_s)$ are two PRMs with the same bandwidth. If $\Pi_s>\Pi_l$, then Ω_s is not better than Ω_l. This is trivially true because the starvation time of a PRM (Π,Θ) is $2(\Pi-\Theta)=2\Pi(1-\frac{\Theta}{\Pi})$. Since $\frac{\Theta_l}{\Pi_l}=\frac{\Theta_s}{\Pi_s}$, if $\Pi_s>\Pi_l$ then the starvation time of Ω_s is longer than that of Ω_l. Therefore, a necessary condition for Ω_s to be better than Ω_l is $0<\frac{\Pi_s}{\Pi_l}\le 1$. Note that $\Pi_s>0$ and $\Pi_l>0$ trivially hold here. Lemma 3.1 addresses the $0<\frac{\Pi_s}{\Pi_l}\le\frac{1}{2}$ case.

Lemma 3.1: Given two PRMs $\Omega_s=(\Pi_s,\Theta_s)$ and $\Omega_l=(\Pi_l,\Theta_l)$ with the same bandwidth, i.e., $\frac{\Theta_s}{\Pi_s}=\frac{\Theta_l}{\Pi_l}$, if $0<\frac{\Pi_s}{\Pi_l}\le\frac{1}{2}$, then $\forall t,\mathrm{sbf}_s(t)\ge\mathrm{sbf}_l(t)$, i.e., Ω_s is better than Ω_l.

Proof: We show that $\forall t,\mathrm{lsbf}_s(t)\ge\mathrm{usbf}_l(t)$, and then it follows $\mathrm{sbf}_s(t)\ge\mathrm{lsbf}_s(t)\ge\mathrm{usbf}_l(t)\ge\mathrm{sbf}_l(t)$

We have $\frac{\Theta_s}{\Pi_s}=\frac{\Theta_l}{\Pi_l}$ so $\Theta_l=\frac{\Pi_l\Theta_s}{\Pi_s}$. Thus $\forall t$,

$$\begin{aligned}
&\mathrm{lsbf}_s(t)-\mathrm{usbf}_l(t)\\
&=\frac{\Theta_s}{\Pi_s}(t-2(\Pi_s-\Theta_s))-\frac{\Theta_l}{\Pi_l}(t-(\Pi_l-\Theta_l))\\
&=\frac{\Theta_s}{\Pi_s}((\Pi_l-\Theta_l)-2(\Pi_s-\Theta_s))\\
&=\frac{\Theta_s}{\Pi_s}((\Pi_l-\frac{\Pi_l\Theta_s}{\Pi_s})-2(\Pi_s-\Theta_s))\\
&=\frac{\Theta_s}{\Pi_s}(\frac{\Pi_l}{\Pi_s}(\Pi_s-\Theta_s)-2(\Pi_s-\Theta_s))\\
&=\frac{\Theta_s}{\Pi_s}(\frac{\Pi_l}{\Pi_s}-2)(\Pi_s-\Theta_s)
\end{aligned}\tag{6}$$

$0<\frac{\Pi_s}{\Pi_l}\le\frac{1}{2}\Rightarrow\frac{\Pi_l}{\Pi_s}\ge 2$ and $\Pi_s\ge\Theta_s$, therefore $\forall t\,\mathrm{lsbf}_s(t)-\mathrm{usbf}_l(t)\ge 0$, and thus $\mathrm{sbf}_s(t)\ge\mathrm{lsbf}_s(t)\ge\mathrm{usbf}_l(t)\ge\mathrm{sbf}_l(t)$. Note that in this proof, we exploit the fact that although by definition $\mathrm{lsbf}_s(t)=\max(\frac{\Theta_s}{\Pi_s}(t-2(\Pi_s-\Theta_s)),0)$ and $\mathrm{usbf}_l(t)=\max(\frac{\Theta_l}{\Pi_l}(t-(\Pi_l-\Theta_l)),0)$, it suffices to show $\forall t$, $\frac{\Theta_s}{\Pi_s}(t-2(\Pi_s-\Theta_s))\ge\frac{\Theta_l}{\Pi_l}(t-(\Pi_l-\Theta_l))$, because $\forall x,y\in\mathbb{R}$, $\max(x,0)\ge\max(y,0)$ if $x\ge y$. ■

From Lemma 3.1, if $0<\frac{\Pi_s}{\Pi_l}\le\frac{1}{2}$, then Ω_s is better than Ω_l. If $\frac{\Pi_s}{\Pi_l}=1$, then $\Omega_s=\Omega_l$, and by definition, Ω_s is better than Ω_l. We already showed if $\frac{\Pi_s}{\Pi_l}>1$, then Ω_s is not better than Ω_l. Therefore, the only unknown case is $\frac{1}{2}<\frac{\Pi_s}{\Pi_l}<1$, which is addressed by Lemmas 3.2 and 3.3. We show that when $\frac{\Pi_s}{\Pi_l}\in(\frac{1}{2},1)$, Ω_s is better than Ω_l if and only if the ratio $\frac{\Pi_s}{\Pi_l}$ is in the set $\{\frac{k+1}{2k+1},\ k\in\mathbb{N}\}$. Note that $\{\frac{k+1}{2k+1},\ k\in\mathbb{N}\}$ gives a non-increasing series $\{1,\frac{2}{3},\frac{3}{5},\frac{4}{7},\ldots\}$ with limit $\lim\limits_{k\to+\infty}\dfrac{k+1}{2k+1}=\dfrac{1}{2}$.

Lemma 3.2: Given two PRMs $\Omega_s=(\Pi_s,\Theta_s)$ and $\Omega_l=(\Pi_l,\Theta_l)$ with the same bandwidth, i.e., $\frac{\Theta_s}{\Pi_s}=\frac{\Theta_l}{\Pi_l}$, if $\exists k\in\mathbb{N}$ such that $\frac{\Pi_s}{\Pi_l}=\frac{k+1}{2k+1}$, then $\forall t,\mathrm{sbf}_s(t)\ge\mathrm{sbf}_l(t)$, i.e., Ω_s is better than Ω_l.

Proof: From Figure 2, it is clear that the sbf curve consists of two types of segments: horizontal and sloped. Let A denote the points where usbf_l intersects sbf_l and P denote the points where lsbf_s intersects sbf_s.

From Equation 6, when $\frac{\Pi_s}{\Pi_l}>\frac{1}{2}$, $\forall t\,\mathrm{lsbf}_s(t)<\mathrm{usbf}_l(t)$. And since $\Pi_s\le\Pi_l$, obviously $\forall t\,\mathrm{usbf}_l(t)\le\mathrm{usbf}_s(t)$. Therefore usbf_l lies between lsbf_s and usbf_s, and it must intersect sbf_s. Let Q and R denote the points where sbf_s intersects the horizontal and sloped segments of usbf_l, respectively.

Since $\frac{\Pi_s}{\Pi_l}=\frac{k+1}{2k+1}$, we introduce two auxiliary variables B and c: $B=\frac{\Theta_s}{\Pi_s}=\frac{\Theta_l}{\Pi_l}$ and $\Pi_s=c(k+1),\Pi_l=c(2k+1)$. It can be derived from Equations 3, 4, and 5 that type A, P, Q, and R points are defined by following equations:

$$\mathrm{lsbf}_s(t)=\max(B(t-2c(k+1)(1-B)),0)\tag{7}$$

$$\mathrm{usbf}_l(t)=\max(B(t-c(2k+1)(1-B)),0)\tag{8}$$

$$t_A=c(2k+1)(1-B)+nc(2k+1)\tag{9}$$

$$t_P=2c(k+1)(1-B)+mc(k+1)\tag{10}$$

$$t_Q=t_P-c(1-B)\tag{11}$$

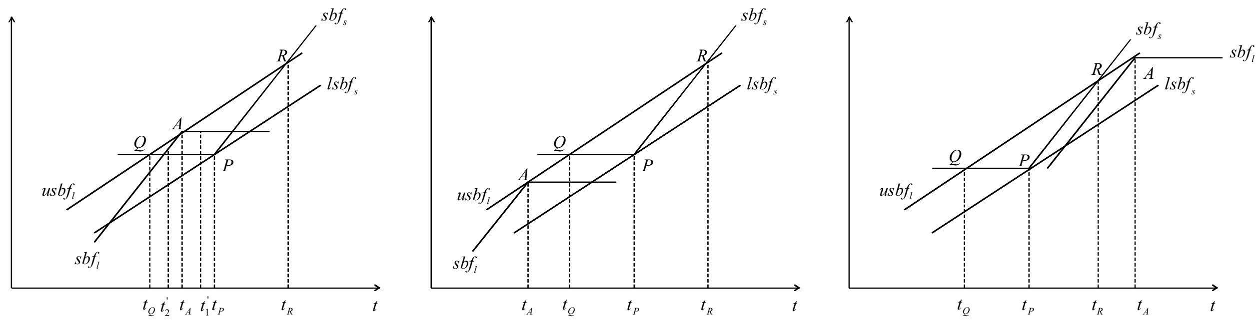
$$t_R=t_P+cB\tag{12}$$

where $m,n\in\mathbb{N}$.

Based on the relationship of the four sets of points (A, P, Q, and R), especially the relative position of type A points and "QPR corners" cut by usbf_l, there are three possible scenarios, as illustrated in Figure 3.

The key part of the proof is the following observation: first, if sbf_s and sbf_l interleave with each another, i.e., $\exists t'$ such that $\mathrm{sbf}_s(t')<\mathrm{sbf}_l(t')$, then the interleaving point $(t',\mathrm{sbf}_l(t'))$ can only lie within the region between lsbf_s and usbf_l, because $\mathrm{lsbf}_s(t')\le\mathrm{sbf}_s(t')<\mathrm{sbf}_l(t')\le\mathrm{usbf}_l(t')$. Furthermore, if such a t' exists, one can always identify an interleaving scenario shown in Figure 3(a), which is shown by the following case study.

Note that sbf is non-decreasing, and the slope of non-horizontal sections is 1. If t' lies on a slope (t'_2 in Figure 3(a)), then let t_A be the coordinate of the nearest type A point such that $t_A\ge t'_2$. From t'_2 to t_A, sbf_l keeps increasing at a slope of 1, which is the maximum slope that an sbf can increase at. Since



(a) $t_Q < t_A < t_R$ (b) $t_A \leq t_Q < t_P < t_R$ (c) $t_Q < t_P < t_R \leq t_A$

Fig. 3. 3 scenarios of sbf_s and sbf_l

$\mathrm{sbf}_l(t_2') > \mathrm{sbf}_s(t_2')$, we have $\mathrm{sbf}_l(t_A) = \mathrm{sbf}_l(t_2') + (t_A - t_2') > \mathrm{sbf}_s(t_2') + (t_A - t_2') \geq \mathrm{sbf}_s(t_A)$.

If t' lies on a horizontal segment (t_1' in Figure 3(a)), then let t_A be the coordinate of the nearest type A point such that $t_A \leq t_1'$. Obviously $\mathrm{sbf}_l(t_1') = \mathrm{sbf}_l(t_A)$. Since $\mathrm{sbf}_l(t_1') > \mathrm{sbf}_s(t_1')$ and sbf_s is non-decreasing, $\mathrm{sbf}_l(t_A) > \mathrm{sbf}_s(t_1') > \mathrm{sbf}_s(t_A)$.

In either case, one can identify a type A point $(t_A, \mathrm{sbf}_l(t_A))$ such that $\mathrm{sbf}_l(t_A) > \mathrm{sbf}_s(t_A)$. From Figure 3, $\mathrm{sbf}_l(t_A) > \mathrm{sbf}_s(t_A)$ yields an interleaving scenario shown in Figure 3(a), i.e., from Equations 9 to 12, $\exists m,n \in \mathbb{N}$ such that $t_Q < t_A < t_R$.

Conversely, if the interleaving scenario in Figure 3(a) happens, then it is trivially true that Ω_s is not better than Ω_l since $\mathrm{sbf}_l(t_A) > \mathrm{sbf}_s(t_A)$. Therefore, Ω_s is not better than Ω_l if and only if there exists one combination of (t_A, t_P, t_Q, t_R) such that $t_Q < t_A < t_R$, i.e., the interleaving scenario happens. Equivalently, Ω_s is better than Ω_l if and only if for any combination of (t_A, t_P, t_Q, t_R), either $t_A \geq t_R$ or $t_A \leq t_Q$.

From Equations 9 to 12, t_Q and t_R are fully defined by t_P, B, and c.

$$t_A \geq t_R \textit{ or } t_A \leq t_Q \Leftrightarrow t_A - t_P \geq cB \textit{ or } t_A - t_P \leq c(B-1)$$

Hence we have: $\forall t, \mathrm{sbf}_s(t) \geq \mathrm{sbf}_l(t)$ if and only if $\forall m,n \in \mathbb{N}$, either $t_A - t_P \geq cB$ or $t_A - t_P \leq c(B-1)$.

We now show this condition is satisfied.

$$\begin{aligned} & t_A - t_P \\ & = c(2k+1)(1-B) + nc(2k+1) \\ & - (2c(k+1)(1-B) + mc(k+1)) \\ & = nc(2k+1) - mc(k+1) - c(1-B) \\ & = c[n(2k+1) - m(k+1) - 1 + B] \end{aligned} \tag{13}$$

Note that $n,m,k \in \mathbb{N}$, let $x = n(2k+1) - m(k+1) - 1 \in \mathbb{Z}$, then $t_A - t_P = c(x+B)$.

Since $x \in Z$, either $x \geq 0$ or $x \leq -1$. If $x \geq 0$, $t_A - t_P = c(x+B) \geq cB$; if $x \leq -1$, $t_A - t_P = c(x+B) \leq c(B-1)$. From our previous conclusion, $\forall t, \mathrm{sbf}_s(t) \geq \mathrm{sbf}_l(t)$ ■

Given Lemmas 3.1 and 3.2, the only remaining case that we need to check is when the ratio $\frac{\Pi_s}{\Pi_l} \in (\frac{1}{2}, 1)$ but is not in the set $\{\frac{k+1}{2k+1},\ k \in \mathbb{N}\}$. In this case, since $\lim\limits_{k \to +\infty} \dfrac{k+1}{2k+1} = \dfrac{1}{2}$, the ratio $\frac{\Pi_s}{\Pi_l}$ should lie between two consecutive elements in the series $\{\frac{k+1}{2k+1}, k \in \mathbb{N}\}$, i.e., $\exists k'$, $\frac{(k'+1)+1}{2(k'+1)+1} < \frac{\Pi_s}{\Pi_l} < \frac{k'+1}{2k'+1}$. With an auxiliary variable $\varepsilon \in \mathbb{R}$, such condition is equivalent to $\frac{\Pi_s}{\Pi_l} = \frac{k'+\varepsilon}{2k'+1}$ and $\frac{2k'+2}{2k'+3} < \varepsilon < 1$. Lemma 3.3 addresses this case.

Lemma 3.3: Given two PRMs $\Omega_s = (\Pi_s, \Theta_s)$ and $\Omega_l = (\Pi_l, \Theta_l)$ with the same bandwidth, i.e., $\frac{\Theta_s}{\Pi_s} = \frac{\Theta_l}{\Pi_l}$, if $\exists k \in \mathbb{N}$ and $\frac{2k+2}{2k+3} < \varepsilon < 1$, such that $\frac{\Pi_s}{\Pi_l} = \frac{k+\varepsilon}{2k+1}$, then $\exists t'$ such that $\mathrm{sbf}_s(t') < \mathrm{sbf}_l(t')$, i.e., Ω_s is not better than Ω_l.

Proof: The structure of the proof is similar to the proof of Lemma 3.2. Define the auxiliary variables B and c as: $B = \frac{\Theta_s}{\Pi_s}$, $\Pi_s = c(k+\varepsilon)$, $\Pi_l = c(2k+1)$. It can be derived that,

$$\mathrm{lsbf}_s(t) = \max(B(t - 2c(k+\varepsilon)(1-B)), 0) \tag{14}$$

$$\mathrm{usbf}_l(t) = \max(B(t - c(2k+1)(1-B)), 0) \tag{15}$$

$$t_A = c(2k+1)(1-B) + nc(2k+1) \tag{16}$$

$$t_P = 2c(k+\varepsilon)(1-B) + mc(k+\varepsilon) \tag{17}$$

$$t_Q = t_P - c(1-B)(2\varepsilon - 1) \tag{18}$$

$$t_R = t_P + cB(2\varepsilon - 1) \tag{19}$$

where $m,n \in \mathbb{N}$. Following from the observation we made in the previous proof, if there exists t_A and t_P such that $c(B-1)(2\varepsilon-1) < t_A - t_P < cB(2\varepsilon - 1)$, then sbf_s interleaves with sbf_l, and furthermore $\mathrm{sbf}_s(t_A) < \mathrm{sbf}_l(t_A)$, i.e., Ω_s is not better than Ω_l.

Next, we show that this condition is satisfied when $m = 2k+1, n = k+1$, i.e., $m = 2k+1, n = k+1$ gives a combination of (t_A, t_P, t_Q, t_R) such that $c(B-1)(2\varepsilon - 1) < t_A - t_P < cB(2\varepsilon - 1)$.

First, we show that $t_A - t_P < cB(2\varepsilon - 1)$:

$$\begin{aligned} & t_A - t_P - cB(2\varepsilon - 1) \\ & = c(2k+1)(1-B) + (k+1)c(2k+1) \\ & - 2c(k+\varepsilon)(1-B) - (2k+1)c(k+\varepsilon) \\ & - cB(2\varepsilon - 1) \\ & = c[(2k+1)(1-\varepsilon) + (1-2\varepsilon)] \\ & = c[(2k+1)(1-\varepsilon) - (2\varepsilon - 1)] \end{aligned} \tag{20}$$

Note that

$$\frac{2k+2}{2k+3} < \varepsilon < 1 \Leftrightarrow \frac{2k+1}{2k+3} < 2\varepsilon - 1 < 1 \tag{21}$$

and,

$$\begin{aligned} & \frac{2k+2}{2k+3} < \varepsilon < 1 \Leftrightarrow 0 < 1 - \varepsilon < \frac{1}{2k+3} \\ & \Rightarrow 0 < (2k+1)(1-\varepsilon) < \frac{2k+1}{2k+3} < 2\varepsilon - 1 \end{aligned}$$

Therefore,

$$t_A - t_P - cB(2\varepsilon - 1) = c[(2k+1)(1-\varepsilon) - (2\varepsilon - 1)] < 0 \tag{22}$$

Next, we show $c(B-1)(2\varepsilon-1) < t_A - t_P$:

$$\begin{aligned} & t_A - t_P - c(B-1)(2\varepsilon-1) \\ & = c(2k+1)(1-B) + (k+1)c(2k+1) \\ & - 2c(k+\varepsilon)(1-B) - (2k+1)c(k+\varepsilon) \\ & - c(B-1)(2\varepsilon-1) \\ & = c(2k+1)(1-\varepsilon) > 0. \end{aligned} \tag{23}$$

Hence, the lemma holds. ■

We define the Γ function as follows.

Definition 3.4 (Γ Function): Given a positive real number $\Pi_x > 0$, the function $\Gamma(\Pi_x)$ gives a set of positive real numbers $\Gamma(\Pi_x) = \{\Pi | 0 < \frac{\Pi}{\Pi_x} \leq \frac{1}{2}\} \bigcup \{\Pi | \exists k \in \mathbb{N}, \frac{\Pi}{\Pi_x} = \frac{k+1}{2k+1}\}$.

Theorem 3.5 directly follows from Lemmas 3.1, 3.2, and 3.3.

Theorem 3.5: Given two PRMs $\Omega = \langle \Pi, \Theta \rangle$ and $\Omega' = \langle \Pi', \Theta' \rangle$ with the same bandwidth, Ω' is better than Ω if and only if $\Omega' \in E(\Omega)$, where $E(\Omega)$ is a set of PRMs: $E(\Omega = \langle \Pi, \Theta \rangle) = \{\Omega' = \langle \Pi', \Theta' \rangle | \frac{\Pi}{\Theta} = \frac{\Pi'}{\Theta'} \text{ and } \Pi' \in \Gamma(\Pi)\}$.

Note that by Definition 3.4, $\Pi_x \in \Gamma(\Pi_x)$ (when $k = 0$) and therefore, $\Omega \in E(\Omega)$. In the rest of this paper, the set $E(\Omega)$ is referred to as the *equivalent set* of Ω, i.e., if Ω can schedule a component, then any PRM from $E(\Omega)$ can also schedule the component, and all PRMs in $E(\Omega)$ have the same bandwidth.

B. Component Interface Generation

The interface of a leaf component is generated by a 2-step procedure: first, a single bandwidth optimal PRM $\Omega_0 = (\Pi_0, \Theta_0)$ is identified; and second, $E(\Omega_0)$ is derived as the interface of the leaf component. The first step involves choosing a minimum bandwidth PRM that can schedule a given leaf component, which has been addressed by previous studies, such as [2]. Here, we focus on the second step, and subsequently, the composition of interfaces (c.f. Section III-C).

In our framework, the interface of a leaf component is defined as below:

Definition 3.6 (Model-Set Interface): The interface I of a leaf component $C = \langle W, I, A \rangle$ is defined to be $I = E(\Omega_0 = \langle \Pi_0, \Theta_0 \rangle)$, where $\Omega_0 = (\Pi_0, \Theta_0)$ is a bandwidth-optimal PRM that guarantees the schedulability of C.

Observe that each interface as defined above can be fully represented by the bandwidth, which is shared by all PRMs in the interface, and the set of periods of all PRMs. We define the "Bandwidth-Periods" representation, which is used for both leaf and intermediate component interfaces in our framework.

Definition 3.7 (Bandwidth-Periods Interface): The interface I of a component $C = \langle W, I, A \rangle$ is defined to be $I = \langle B, P \rangle$, in which $B = \frac{\Theta_0}{\Pi_0}$ and P is the set of periods of all feasible PRMs, e.g., for a leaf component, $P = \Gamma(\Pi_0)$, where $\Omega_0 = \langle \Pi_0, \Theta_0 \rangle$ is a bandwidth-optimal PRM that guarantees the schedulability of C.

To transform a bandwidth-periods interface $I = \langle B, P \rangle$ into a model-set one, we simply construct a PRM for each period p_i in P as $(p_i, B * p_i)$, as illustrated in Example 1. Obviously, if a leaf component C is schedulable by PRMs, then $\Pi_0 > 0$, meaning there exists a PRM Ω_0 that can schedule C.

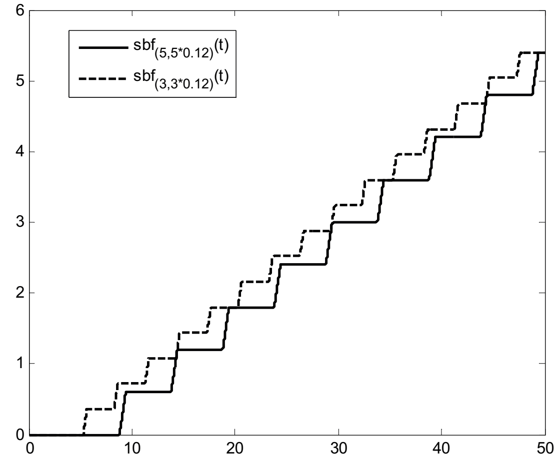


Fig. 4. $\text{sbf}_{(5,5*0.12)}$ and $\text{sbf}_{(3,3*0.12)}$

Example 1 (Interface Generation): Given a leaf component $C = \langle W, I, EDF \rangle$ whose workload consists of two periodic real-time tasks $W = \{(35,2),(50,3)\}$. First, using the techniques introduced in [2], given $\Pi = 5$, one can find the optimal single PRM $\Omega_0 = (5, 0.6)$ with bandwidth of 0.12 (the bandwidth of W is 0.1171). Then, the interface is $I = \langle B, P \rangle$, where $B = 0.12$ and $P = \Gamma(5)$. For example, $3 \in \Gamma(5)$, so that the PRM $\Omega' = (3, 3*0.12) \in E(\Omega_0)$. From Theorem 3.5, $\forall t, \text{sbf}_{(3,3*0.12)}(t) \geq \text{sbf}_{(5,5*0.12)}(t)$, which is verified by numerical calculation as shown in Figure 4.

C. Interface Composition

In the scheduling hierarchy, leaf components present their interfaces to higher level intermediate components. Each intermediate component then composes its sub-interfaces into a new interface. The interface for an intermediate component is also a set of PRMs that share the same bandwidth. Interfaces are composed bottom-up in the scheduling tree until the root component's interface is obtained. The root component then select one specific PRM from its own interface. The period Π of the selected PRM is propagated to all components in the tree so that each component can choose the specific PRM with period Π from its own interface.

The interface composition of an intermediate component is done by taking the intersection of the period sets and summing up the bandwidth of the sub-interfaces, as given below.

Definition 3.8 (Interface Composition): The interface I of an intermediate component $C = \langle W, I, A \rangle$ is given by $I = \langle B, P \rangle$ where $B = \sum_i^n B_i$, $P = \bigcap_i^n P_i$, where each $I_i = \langle B_i, P_i \rangle$ is a sub-interface in W, i.e., $W = \{I_1, \ldots, I_n\}$.

In this definition, the composed interface I is only feasible if $P = \bigcap_i^n P_i \neq \emptyset$. Theorem 3.9 shows that the intersection is always non-empty, under the reasonable assumption that each leaf component is schedulable.

Theorem 3.9 (Non-Empty Intersection): Given a scheduling hierarchy in which the interfaces of leaf components are $\{I_0 = \langle B_0, \Gamma(\Pi_0)\rangle, \dots, I_n = \langle B_n, \Gamma(\Pi_n)\rangle\}$. Assume each leaf component i is schedulable so that $\forall i,\ \Pi_i > 0$; then for any intermediate component C_j, its interface $I_j = \langle B_j, P_j\rangle$, which is obtained by Definition 3.8, is non-empty, i.e., $P_j \neq \emptyset$.

Proof: Let $\{I_{j0} = \langle B_{j0}, \Gamma(\Pi_{j0})\rangle, \dots, I_{jm} = \langle B_{jm}, \Gamma(\Pi_{jm})\rangle\}$ denotes the interfaces of the leaf components in the subtree rooted at C_j. Apply the composition in Definition 3.8 to each intermediate component, and it is clear that $P_j = \bigcap_{x=0}^{m} \Gamma(\Pi_{jx})$. From Definition 3.4, $\forall 0 < w \leq \frac{1}{2}\Pi_{jx}\ \ w \in \Gamma(\Pi_{jx})$. Let $\Pi' = \frac{1}{2}\min(\Pi_{j0}, \dots, \Pi_{jm})$. Then $\forall 0 \leq x \leq m$ and $\forall 0 < w \leq \Pi'$, $w \leq \Pi' \leq \frac{1}{2}\Pi_{jx}$, and therefore $w \in \Gamma(\Pi_{jx})$. So $\forall 0 < w \leq \Pi'$, $w \in P_j$, i.e., $P_j \neq \emptyset$. ■

The period Π picked by the root component is propagated to all components in the tree, and that means each interface $I_i = \langle B_i, P_i\rangle$ is reduced to a single PRM $\Omega_i = (\Pi, \Pi * B_i)$. In the composition step, the period set of a parent component is obtained by taking the intersection of the period sets of all its children, therefore the value Π picked by the root component is within the period set of every interface, i.e., $\forall i, \Pi \in P_i$. This guarantees the PRM Ω_i is already in the interface I_i, and thus it is a valid candidate resource model for component C_i. Then for a leaf component C_i, Ω_i can schedule its workload because it is in the interface.

For each intermediate component $C_j = \langle W_j, I_j, A_j\rangle$, the sub-interfaces in its workload are reduced to single PRMs once the root component has determined the resource period, i.e., $W_j = \{I_{j1}, \dots, I_{jn}\} = \{\Omega_{j1}, \dots, \Omega_{jn}\}$. To show the schedulability, we need to prove that Ω_j can schedule $W_j = \{\Omega_{j1}, \dots, \Omega_{jn}\}$. Notice that Ω_j and $\Omega_{j1}, \dots, \Omega_{jn}$ have the same period Π, which is picked by the root component. Each $\Omega_{jk} = (\Pi, \Theta_{jk}) \in W_j$ is taken as a periodic real-time task $T_{jk} = (\Pi, \Theta_{jk})$ by C_j. Then each intermediate component C_j needs to schedule a task set $\{T_{j1} = (\Pi, \Theta_{j1}), \dots, T_{jn} = (\Pi, \Theta_{jn})\}$ using resource model $\Omega_j = (\Pi, \Theta_j)$. From Definition 3.8, we know $\Theta_j = \sum_k \Theta_{jk}$. Lemma 3.10 shows that such a schedule is feasible under any work-conserving scheduling policy, by exploiting the assumption that the first jobs of the tasks $\{T_{j1}, \dots, T_{jn}\}$ are released at the same time instance t_0, and the resource supply of Ω_j also starts at t_0. This synchronization assumption will be justified and further discussed in Section IV-A.

Lemma 3.10: Given a set of periodic real-time tasks with identical period $T = \{(p, e_1), \cdots, (p, e_n)\}$, the PRM $\Omega = (p, \sum_i e_i)$ can schedule T under any work-conserving scheduling algorithm, under the assumption that the first jobs of all tasks are released at the same time instance t_0 and the resource supply of Ω also starts at t_0.

Proof: Without loss of generality, let $t_0 = 0$. Then within each period $[kp, (k+1)p]$ for any $k \in \mathbb{N}$, Ω only needs to schedule the jobs released at kp and due by $(k+1)p$, because jobs are only released at time instances $0, p, 2p, \dots, kp$. The total amount of resources provided by Ω within $[kp, (k+1)p]$ is $\sum_i e_i$, which is equal to the total resource demand of T within that time interval. Under a work-conserving policy, resources provided by Ω cannot be wasted if there are unfinished jobs. Therefore, all jobs released at kp can be finished within $[kp, (k+1)p]$. Apply this reasoning for all $k \in \mathbb{N}$, and it is clear that Ω can schedule T. ■

The following theorem states that schedulability is guaranteed in our framework.

Theorem 3.11 (Schedulability): The interface generation and composition proposed in Definitions 3.7 and 3.8 guarantee the schedulability, in the sense that as long as the root component is schedulable, all components in the hierarchy are schedulable.

Proof: We will prove this through top-down induction on the height of the scheduling tree h. This statement in the theorem is denoted as $P(h)$, and $P(h) = True$ if the statement is true for a scheduling tree with height h.

Base case: $h = 1$. The scheduling tree has only one intermediate component, the root component. From Definition 3.8 and Lemma 3.10, the root component can schedule all its sub-components under any work-conserving policy. Therefore $P(1) = \text{True}$.

Induction step: Assume $\forall i \leq k, P(i) = \text{True}$. Consider a scheduling tree with $h = k+1$; since $P(1) = \text{True}$, root component C_0 can schedule all its children $C_1, \dots, C_n$ with $depth = 1$. By $\forall i \leq k, P(k) = \text{True}$, each $depth = 1$ component C_i can schedule all the components in the sub-tree rooted at C_i because the height of the sub-tree is no more than k. Therefore $P(k+1) = \text{True}$. ■

In Definition 3.8, the interface I of an intermediate component is taken by $B = \sum_i B_i$ and $P = \bigcap_i P_i$. Here it is unnecessary to calculate the equivalent set for each PRM in the composed interface $I = \langle \sum_i B_i, \bigcap_i P_i\rangle$, because the Γ function has the following closure property such that for any PRM $\Omega \in I = \langle \sum_i B_i, \bigcap_i P_i\rangle$, the equivalent set $E(\Omega) \subseteq I$.

Theorem 3.12 (Closure Property of Γ Function): Given any n positive real numbers $p_1, \dots, p_n$, let $P_i = \Gamma(p_i)$ and $P = \bigcap_i P_i$, $\forall p' \in P, \Gamma(p') \subseteq P$.

Proof: First we will prove by case study that $\forall x > 0, \forall y \in \Gamma(x), \Gamma(y) \subseteq \Gamma(x)$.

Recall that $\Gamma(x) = \{x' | 0 < \frac{x'}{x} \leq \frac{1}{2}\} \bigcup \{x' | \exists k \in \mathbb{N}, \frac{x'}{x} = \frac{k+1}{2k+1}\}$.

Case 1: $y = x$, $\Gamma(y) = \Gamma(x) \subseteq \Gamma(x)$.

Case 2: $\exists k \in \mathbb{N}^+$, such that $y = \frac{k+1}{2k+1}x$. In order to prove $\Gamma(y) \subseteq \Gamma(x)$, we need to show that $\forall z \in \Gamma(y), z \in \Gamma(x)$.

- $z = y = \frac{k+1}{2k+1}x$, $z = y \in \Gamma(x)$.
- $\exists k' \in \mathbb{N}^+, z = \frac{k'+1}{2k'+1}y = \frac{k'+1}{2k'+1}\frac{k+1}{2k+1}x$. Note that $\forall k \in \mathbb{N}^+, \frac{k+1}{2k+1} - \frac{2}{3} = \frac{1/3 - k/3}{2k+1} \leq 0$ so $\forall k \in \mathbb{N}^+, \frac{k+1}{2k+1} \leq \frac{2}{3}$. Therefore $z = \frac{k'+1}{2k'+1}\frac{k+1}{2k+1}x \leq \frac{2}{3}\frac{2}{3}x = \frac{4}{9}x < \frac{1}{2}x$, and thus $z \in \Gamma(x)$.
- $z \leq \frac{1}{2}y = \frac{1}{2}\frac{k+1}{2k+1}x < \frac{1}{2}x$, therefore $z \in \Gamma(x)$.

Case 3: $y \leq \frac{1}{2}x$, then $\forall z \in \Gamma(y)$, $z \leq y \leq \frac{1}{2}x$, so $z \in \Gamma(x)$ and $\Gamma(y) \subseteq \Gamma(x)$

Since $P = \bigcap_i P_i$ and $p' \in P$, $\forall i, p' \in P_i = \Gamma(p_i)$. Apply the previous conclusion, let $x = p_i$ and $y = p'$, and we have $\forall i, \Gamma(p') \subseteq \Gamma(p_i) = P_i$. Therefore $\Gamma(p') \subseteq P$. ■

From Definition 3.8, the bandwidth of each intermediate component is the sum of the bandwidth of its sub-components. Therefore our approach removes the abstraction overhead during the composition, and the composition is associative, since addition and intersection are both associative operations. Furthermore, the bandwidth of the root component is equal to the sum of the bandwidth of all leaf components, and thus our approach is bandwidth optimal. There properties are formally proved below.

Theorem 3.13 (Bandwidth Optimality): The interface generation and composition proposed in Definitions 3.7 and 3.8 guarantee the resulting bandwidth of the root component in a scheduling tree is equal to the sum of the bandwidth of the interfaces of all leaf components. Therefore the proposed scheduling framework is bandwidth optimal, in the sense that given a set of leaf components, assume the single optimal PRM of each leaf component can be identified, any feasible PRM-based scheduling framework requires at least as much root-level bandwidth as our framework.

Proof: First we proved that in our approach, the interface of root component $C = \langle W, I, A\rangle$ for any scheduling tree is $I = \langle \sum_i B_i, \bigcap_i P_i\rangle$, in which

$$\{C_1 = \langle W_1, \langle B_1, P_1\rangle, A_1\rangle, \ldots, C_n = \langle W_n, \langle B_n, P_n\rangle, A_n\rangle\}$$

are the leaf components. We will denote such a predicate as $P(h)$. Proved by induction on the height of scheduling tree h:

Base case: $h = 1$. The root component C is the immediate parent of all leaf components. $P(1)$ is trivially true by Definition 3.8.

Induction step: Assume $\forall i \le k, P(i) = \text{True}$. Given any scheduling tree with height $k+1$, consider all the components with depth 1: $\{C'_1 = \langle W'_1, \langle B'_i, P'_i\rangle, A'_1\rangle, \ldots, C'_m = \langle W'_m, \langle B'_m, P'_m\rangle, A'_m\rangle\}$. Since the height of any component in $\{C'_1, \ldots, C'_m\}$ is at most k, the induction assumption applies.

$$\forall i, I'_i = \langle \sum_{j\in S_i} B_j, \bigcap_{j\in S_i} P_j\rangle$$

in which S_i denotes the leaf components of the sub-tree rooted at C_i. By the tree property, $\bigcup_i S_i$ is the set of all leaf components and $\forall i \ne j, S_i \bigcap S_j = \Phi$. Therefore, from Definition 3.8,

$$\begin{aligned} C &= \langle W, I, A\rangle \\ &= \langle \{I'_1, \ldots, I'_m\}, \langle \sum_i \sum_{j\in S_i} B_j, \bigcap_i \bigcap_{j\in S_i} P_j\rangle, A\rangle \\ &= \langle \{I'_1, \ldots, I'_m\}, \langle \sum_i B_i, \bigcap_i P_i\rangle, A\rangle \end{aligned} \tag{24}$$

in which $\{C_1 = \langle W_1, \langle B_1, P_1\rangle, A_1\rangle, \ldots, C_n = \langle W_n, \langle B_n, P_n\rangle, A_n\rangle\}$ are the leaf components of the tree.

The optimality is proved by contradiction. Given a set of leaf components $\{C_1 = \langle W_1, \langle B_1, P_1\rangle, A_1\rangle, \ldots, C_n = \langle W_n, \langle B_n, P_n\rangle, A_n\rangle\}$, suppose there is a scheduling tree in which the bandwidth of interface at root component is strictly less than $\sum_i B_i$, then there is at least one component $C' = \langle W', I', A'\rangle$ in the tree such that the bandwidth of I' is strictly less than bandwidth of its workload W', which makes W' unschedulable. ■

Theorem 3.14 (Associativity): The interface generation and composition proposed in Definitions 3.7 and 3.8 guarantee the associativity.

Proof: This is trivially true since both intersection of period sets and addition of bandwidth are associative operations. ■

IV. Methodology Evaluation and Comparison to Previous Work

In this section, we evaluate our approach by comparing it with two previous techniques: the PRM-based compositional scheduling framework proposed in [2] and the incremental analysis framework introduced by [1].

The compositional analysis in [2] assumes that the tasks in the workload of an intermediate component can start at arbitrary times, just like the tasks in the workload of a leaf component. This assumption was implicitly dropped by the incremental analysis proposed by [1], but the underlying issue has not been elaborated on in [1]. In this section, we show that this assumption may incur additional bandwidth overhead during composition, and that it is, in fact, unnecessary.

The incremental analysis in [1] assumes a single PRM for each leaf component is identified under the lsbf-based schedulability condition, which is sufficient but not necessary, and it incurs more abstraction overhead than the sbf-based condition. Our approach allows the single PRM to be identified under the sbf-based schedulability condition.

A. Aligned Offsets Assumption for Intermediate Components

For a periodic task $T_i = (p_i, e_i)$, let *offset* a_i denote the release time of the first job of T_i, i.e., the jobs of T_i are released at time instants $a_i, a_i + p_i, \ldots, a_i + k * p_i$. Similarly, for a PRM $\Omega = (\Pi, \Theta)$, let offset a_Ω denote the time instant at which Ω starts providing resource, i.e., Ω guarantees at least Θ units of resource supply over any time interval $[a_\Omega + k\Pi, a_\Omega + (k+1)\Pi]$. Given a set of real-time tasks $W = \{(p_1, e_1), \ldots, (p_n, d_n)\}$, $\text{dbf}_W(t)$ is the worst-case demand over any time interval with length t, regardless of the offsets $\{a_1, \ldots, a_n\}$.

In the scheduling tree, the workload of an intermediate component is composed of a set of periodic tasks, which are transformed from the interfaces of sub-components. On the other hand, the workload of a leaf component consists of real-time tasks of applications. To differentiate these two kinds of workloads, we call the tasks in the workloads of intermediate components *interface tasks*, in contrast to *system tasks* by which we refer to those in the workloads of leaf components.

Since the offsets of system tasks are not determined by schedulers, for the schedulability analysis of a leaf component, it is necessary to assume the offsets $\{a_1, \ldots, a_n\}$ have arbitrary values and that the resource model should be able to schedule the workload for any offsets $\{a_1, \ldots, a_n\}$.

In order to achieve schedulability under arbitrary offsets, the bandwidth of Ω is usually higher than the bandwidth of W. For example, in Figure 1, let $W_1 = \{T_1 = (5,1), T_2 = (5,1)\}$ be scheduled under the EDF policy and assume arbitrary offsets for T_1 and T_2. Given $\Pi = 5$, the bandwidth optimal PRM for W_1 is $\Omega = (5, 3.5)$. As shown in Figure 5, the sbf_Ω curve intersects the dbf_{W_1} curve at $t = 5$, i.e., if $\Theta < 3.5$, then $\text{sbf}_\Omega(5)$ will be

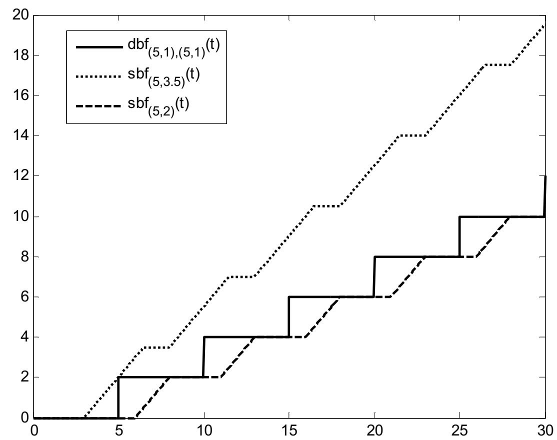


Fig. 5. $\mathrm{sbf}_{(5,3.5)}$, $\mathrm{sbf}_{(5,2)}$, and $\mathrm{dbf}_{\{(5,1),(5,1)\}}$

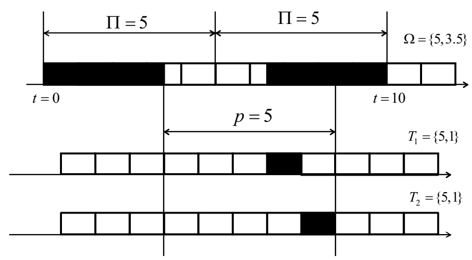


Fig. 6. worst-case alignment under arbitrary offsets assumption.

less than $\mathrm{dbf}_{W_1}(5)$. In this case, the bandwidth of Ω and W_1 are 0.7 and 0.4, respectively, i.e., the overhead is 0.3. Such overhead is caused by the arbitrary offsets assumption and starvation times of PRMs. Figure 6 illustrates the worst-case scenario in which $\Omega = (5, 3.5)$ is tight for W_1. The starvation of Ω happens when the resource is supplied in the first Θ time units in one period and in the last Θ time units of the next period. For $\Omega = (5, 3.5)$, the starvation length is $2*(5-3.5) = 3$. If two jobs of T_1 and T_2 are released at the beginning of the starvation interval, which is $t = 3.5$, then within time interval $[3.5, 8.5]$, Ω can only guarantee $5-3 = 2$ units of resource supply, which is equal to the resource demand of T_1 and T_2 during $[3.5, 8.5]$.

In [2], it is assumed that interface tasks may also have arbitrary offsets. Therefore, the interfaces of intermediate components and those of leaf components are derived under the same schedulability condition. In that case, the bandwidth overhead incurred by the arbitrary offsets assumption also applies to the interface composition of intermediate components. For example, in Figure 1, let $W_2 = \{I_3 = (5,1), I_4 = (5,1)\}$ and $A_2 = EDF$; under the arbitrary offsets assumption, W_2 is just the same as W_1. And therefore the composed interface is $I_2 = (5, 3.5)$.

However, the arbitrary offsets assumption for interface tasks is unnecessary. For an intermediate component, the interface tasks in its workload are not tasks generated by applications but abstract transformations of resource models for the sub-components. For an intermediate component, the scheduling of the interface tasks is the process of allocating resources top down to the sub-components. Therefore, for an intermediate component, the scheduler can determine the offset of each interface task. Theorem 4.1 shows that the schedulability property is still preserved while dropping the arbitrary offsets assumption for interface tasks.

Theorem 4.1 (Offset of Interface Tasks): Given a scheduling hierarchy in which the schedulability is checked by an sbf-dbf based condition, for any intermediate component $C_i = \langle W_i, I_i, A_i \rangle$, suppose its corresponding interface task is (a_i, Π_i, Θ_i), where a_i is the offset and $\Omega_i = (\Pi_i, \Theta_i)$ is a PRM in I_i. If Ω_i can schedule W_i under some offset a_i', then Ω_i can schedule W_i under any offset a_i, i.e., changing the offset of an interface task does not affect the schedulability of the sub-components of C_i.

Proof: The proof is based on the fact that by definition, $\mathrm{sbf}_\Omega(t)$ gives the worst-case resource supply of Ω over any time interval of length t. The worst-case supply is identified by sliding a window of length t in the timeline, therefore it does not depend on when Ω starts. Hence, for any a_i and a_i', $\mathrm{sbf}_{(a_i,\Pi_i,\Theta_i)}(t) = \mathrm{sbf}_{(a_i',\Pi_i,\Theta_i)}(t)$. Since the schedulability is checked by an sbf-dbf based condition and changing a_i does not change the sbf, it immediately follows that changing the offset of an interface task does not affect the schedulability of the sub-components. ■

Taking the $W_2 = \{I_3, I_4\}$ and $A_2 = EDF$ in Figure 1 for example, we show that the workload of C_3 is schedulable regardless of the offset of I_3. If C_3 is a leaf component, then I_3 is derived from an sbf-dbf based schedulability condition, and Theorem 4.1 applies. If C_3 is an intermediate component, then the offsets of the workload W_3 and the offset of I_3 are aligned, in which case the schedulability is guaranteed by Lemma 3.10. This shows that the offsets of all interface tasks can be aligned without changing the schedulability of all the components. The schedulability under the proposed interface abstraction and composition technique has been formally proved in Theorem 3.11. For example, in Figure 1, let $W_2 = \{I_3 = (5,1), I_4 = (5,1)\}$ and $A_2 = EDF$, $I_2 = (5,2)$ suffices to schedule W_2, under the assumption that the offsets of I_2, I_3, and I_4 are aligned. In contrast, under the arbitrary offsets assumption, at least $I_2 = (5, 3.5)$ is needed to schedule W_2, as discussed before. Note that in our framework, under the aligned offsets assumption and identical interface period condition, the starvation scenario illustrated in Figure 6 will not happen.

The incremental schedulability analysis framework proposed in [1] composes two interfaces $I_1 = (\Pi, \Theta_1)$ and $I_2 = (\Pi, \Theta_2)$ by simply adding up the Θ's, i.e., the composed interface is $I = (\Pi, \Theta_1 + \Theta_2)$. In order for such composition to be correct, the arbitrary offsets assumption must be dropped, which has not been explicitly stated in [1].

B. Comparison with Incremental Analysis

In incremental schedulability analysis [1], the interfaces of leaf components are derived under the lsbf-based schedulabil-

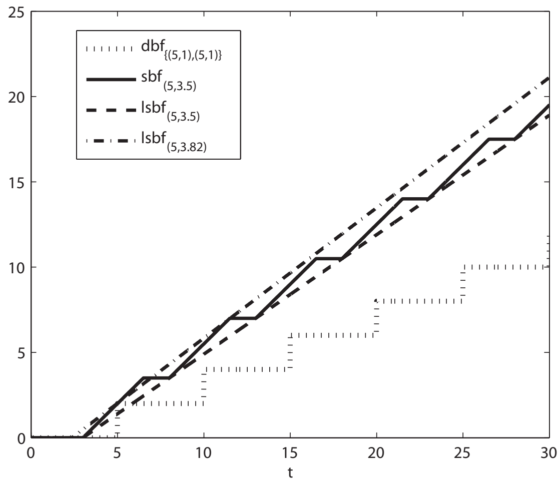


Fig. 7. schedulability under sbf vs. lsbf

ity condition, e.g., for EDF it is $\forall t, \text{lsbf}(t) \geq \text{dbf}(t)$. Such a condition is sufficient but not necessary, and it incurs more bandwidth overhead than the sbf-based condition, which for EDF is $\forall t, \text{sbf}(t) \geq \text{dbf}(t)$. This is because the lsbf is a lower bound of an sbf. As an example, given two system tasks $W = \{(5,1),(5,1)\}$ scheduled under EDF, and let the period of the resource model be $\Pi = 5$, from the previous section it is known that $\Omega = (5,3.5)$ is optimal under the sbf-based schedulability condition $\forall t, \text{sbf}(t) \geq \text{dbf}(t)$ and arbitrary offsets assumption. If instead, $\forall t, \text{lsbf}(t) \geq \text{dbf}(t)$ is used for checking the schedulability, at least a PRM $(5,3.82)$ will be needed. The sbf and dbf curves are shown in Figure 7. In our approach, the optimal PRM can be derived under the sbf-based schedulability condition, and therefore it reduces the bandwidth overhead for leaf components.

V. Conclusion

In this paper, we have identified several important properties regarding the supply bound function of a PRM. Based on those properties, we have proposed a new interface abstraction and composition framework which achieves schedulability, optimality, and associativity. Our approach eliminates the abstraction overhead in composition. The proposed framework is applicable to a wide range of real-time systems. One of our future research directions is extending this framework to take into account preemption overhead.

References

[1] A. Easwaran, I. Shin, O. Sokolsky, and I. Lee, "Incremental schedulability analysis of hierarchical real-time components," in *Proc. the 6th ACM & IEEE International conference on Embedded software (EMSOFT'06)*, 2006.

[2] I. Shin and I. Lee, "Compositional real-time scheduling framework with periodic model," *ACM Transactions on Embedded Computing Systems*, 2008.

[3] A. Easwaran, I. Lee, and O. Sokolsky, "Interface algebra for analysis of hierarchical real-time systems," in *Foundations of Interface Technologies, FIT'08, Satellite workshop of ETAPS'08*, 2009.

[4] G. Lipari and E. Bini, "Resource partitioning among real-time applications," in *Proc. 15th Euromicro Conference on Real-Time Systems (ECRTS'03)*, 2003.

[5] I. Shin and I. Lee, "Periodic resource model for compositional real-time guarantees," in *Proc. the 24th IEEE International Real-Time Systems Symposium (RTSS'03)*, 2003.

[6] X. A. Feng and A. K. Mok, "A model of hierarchical real-time virtual resources," in *Proc. the 23rd IEEE Real-Time Systems Symposium (RTSS'02)*, 2002.

[7] S. Saewong, R. R. Rajkumar, J. P. Lehoczky, and M. H. Klein, "Analysis of hierar hical fixed-priority scheduling," in *Proc. the 14th Euromicro Conference on Real-Time Systems (ECRTS'02)*, 2002.

[8] I. Shin and I. Lee, "Compositional real-time scheduling framework," in *Proc. the 25th IEEE Real-Time Systems Symposium (RTSS'04)*, 2004.

[9] L. Almeida and P. Pedreiras, "Scheduling within temporal partitions: response-time analysis and server design," in *Proc. The 4th ACM international conference on Embedded software (EMSOFT'04)*, 2004.

[10] R. I. Davis and A. Burns, "Hierarchical fixed priority pre-emptive scheduling," in *Proc. the 26th IEEE International Real-Time Systems Symposium (RTSS'05)*, 2005.

[11] S. Matic and T. A. Henzinger, "Trading end-to-end latency for composability," in *Proc. the 26th IEEE International Real-Time Systems Symposium (RTSS'05)*, 2005.

[12] R. I. Davis and A. Burns, "Resource sharing in hierarchical fixed priority pre-emptive systems," in *Proc. the 27th IEEE International Real-Time Systems Symposium (RTSS'06)*, 2006.

[13] E. Wandeler and L. Thiele, "Interface-based design of real-time systems with hierarchical scheduling," in *Proc. the 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06)*, 2006.

[14] T. A. Henzinger and S. Matic, "An interface algebra for real-time components," in *Proc. the 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06)*, 2006.

[15] L. Thiele, E. Wandeler, and N. Stoimenov, "Real-time interfaces for composing real-time systems," in *Proc. the 6th ACM & IEEE International conference on Embedded software (EMSOFT'06)*, 2006.

[16] A. Easwaran, M. Anand, and I. Lee, "Compositional analysis framework using edp resource models," in *Proc. the 28th IEEE International Real-Time Systems Symposium (RTSS'07)*, 2007.

[17] M. Behnam, I. Shin, T. Nolte, and M. Nolin, "SIRAP: A synchronization protocol for hierarchical resource sharing in real-time open systems," in *Proc. the 7th ACM & IEEE International Conference on Embedded Software (EMSOFT'07)*, 2007.

[18] N. Fisher, M. Bertogna, and S. Baruah, "Resource-locking durations in edf-scheduled systems," in *Proc. the 13th IEEE Real Time and Embedded Technology and Applications Symposium (RTAS'07)*, 2007.

[19] Z. Deng and J. W.-S. Liu, "Scheduling real-time applications in an open environment," in *Proc. the 18th IEEE Real-Time Systems Symposium (RTSS'97)*, 1997.

[20] T.-W. Kuo and C.-H. Li, "A fixed-priority-driven open environment for real-time applications," in *Proc. the 20th IEEE Real-Time Systems Symposium (RTSS'99)*, 1999.

[21] G. Lipari and S. Baruah, "Efficient scheduling of real-time multi-task applications in dynamic systems," in *Proc. the Sixth IEEE Real Time Technology and Applications Symposium (RTAS'00)*, 2000.

[22] G. Lipari, J. Carpenter, and S. Baruah, "A framework for achieving inter-application isolation in multiprogrammed hard-real-time environments," in *Proc. the 21st IEEE Real-Time Systems Symposium (RTSS'00)*, 2000.

[23] A. Mok, X. Feng, and D. Chen, "Resource partition for real-time systems," in *Proc. the 7th IEEE Real-Time Technology and Applications Symposium (RTAS'01)*, 2001.

[24] L. de Alfaro and T. A. Henzinger, "Interface automata," in *Proc. the 9th Annual Symposium on Foundations of Software Engineering*, 2001.

[25] ——, "Interface theories for component-based design," in *Proc. the 1st International Workshop on Embedded Software*, 2001.

[26] S. Baruah, R. Howell, and L. Rosier, "Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor," *Real-Time Syst.*, 1990.

[27] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: exact characterization and average case behavior," in *Proc. the 10th IEEE Real-Time Systems Symposium (RTSS'89)*, 1989.

[28] J. Liu, *Real-Time Systems*. Prentice Hall, 2000.