Removing Abstraction Overhead in the Composition of Hierarchical Real-Time System

Sanjian Chen  
*University of Pennsylvania*, sanjian@cis.upenn.edu

Linh T.X. Phan  
*University of Pennsylvania*, linhphan@cis.upenn.edu

Jaewoo Lee  
*University of Pennsylvania*, jaewoo@cis.upenn.edu

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

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

Follow this and additional works at: [http://repository.upenn.edu/cis_papers](http://repository.upenn.edu/cis_papers)

Recommended Citation


This paper is posted at ScholarlyCommons. [http://repository.upenn.edu/cis_papers/463](http://repository.upenn.edu/cis_papers/463)

For more information, please contact repository@pobox.upenn.edu.
Removing Abstraction Overhead in the Composition of Hierarchical Real-Time System

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

Comments

This conference paper is available at ScholarlyCommons: http://repository.upenn.edu/cis_papers/463
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 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 \( \Gamma \) which generates a set of PRMs from any given single PRM \( \Omega' \), each of which has the same bandwidth as \( \Omega' \) and guarantees schedulability. Using \( \Gamma \), 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.

**II. BACKGROUND AND OUR APPROACH**

The real-time task sets discussed in this paper are characterized by the demand-bound function \( \text{dbf}(i) \), 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 \( T = \{ T_1 = (p_1, e_1), \ldots, T_n = (p_n, e_n) \} \) 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.
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 $T_i$.

$$dbf_T(t) = \frac{\sum_{i=1}^{n} \left\lfloor \frac{t}{p_i} \right\rfloor e_i}{1 - \left(1 - \frac{1}{p_i} \right)^{n}}$$

$$dbf_T(t,i) = e_i + \sum_{t_i \in HP_T(i)} \left\lfloor \frac{t_i}{p_k} \right\rfloor e_k$$

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 $C = \langle W, I, A \rangle$, where $W$ 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 $sbf(I)$, which gives the minimum supply of a PRM $\Omega$ over any time interval $t$, is given by [2] as Equation 3, where $x = 2(\Pi - \Theta)$ and $y = \left(1 - \frac{1}{1 - \left(1 - \frac{1}{\Pi} \right)^{n}} \right)$. The functions $lsbf$ and $usbf$, the tight linear lower and upper bounds for $sbf$, are given by [2] as Equations 4 and 5.

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

$$lsbf_{\Omega}(t) = \max \left\{ \frac{\Theta}{\Pi} (t - 2(\Pi - \Theta)), 0 \right\}$$

$$usbf_{\Omega}(t) = \max \left\{ \frac{\Theta}{\Pi} (t - (\Pi - \Theta)), 0 \right\}$$

Figure 2 shows the $sbf$, $lsbf$ and $usbf$ of a PRM $\Omega = (\Pi, \Theta)$. As shown in the figure, the starvation time of $\Omega$, the longest time interval during which there is no resource supply in the worst case, is $2(\Pi - \Theta)$, i.e., $\forall t \leq 2(\Pi - \Theta), sbf_{\Omega}(t) = 0$. 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 $\Omega$ 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 $\Omega$ under EDF, if and only if $\forall t \leq T_{CMW}$ $dbf_W(t) \leq sbf_\Omega(t)$, where $T_{CMW}$ is the least common multiple of $p_i$ for all $T_i \in W$. $W$ is schedulable by $\Omega$ under RM, if and only if $\forall t \in W \exists \gamma \in [0, p_i]$ $dbf_W(t,i) \leq sbf_\Omega(t)$

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 $\forall t, sbf_{\Omega_1}(t) \geq sbf_{\Omega_2}(t)$, then for a workload $W$ which consists of periodic tasks, $\Omega_1$ can schedule $W$ under EDF/RM if $\Omega_2$ can schedule $W$ under EDF/RM.

**Proof:** Suppose $\Omega_2$ can schedule $W$ under EDF. Then, $\forall t, dbf_W(t) \leq sbf_{\Omega_2}(t) \leq sbf_{\Omega_1}(t)$. Hence $\Omega_1$ 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 $\Omega_1$ is said to be better than $\Omega_2$. 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 $B = \frac{\Pi}{\Pi - \Theta}$. 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 subinterfaces. In a leaf component, the bandwidth of a workload $W = \{(p_1, e_1), \ldots, (p_n, e_n)\}$ is $B = \sum \frac{e_i}{p_i}$. 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, 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 subinterfaces 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 Ω = (Π, Θ), 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 Ω1 = (Π1, Θ1) and Ω2 = (Π2, Θ2) are two PRMs with the same bandwidth. If Π1 > Π2, then Ω1 is not better than Ω2. This is trivially true because the starvation time of a PRM (Π, Θ) is 2(Π – Θ) = 2Π1(1 – Θ1). Since Θ1 = Θ2 = Θ, if Π1 > Π2 then the starvation time of Ω1 is longer than that of Ω2. Therefore, a necessary condition for Ω1 to be better than Ω2 is 0 < Π1 ≤ Π2.

Lemma 3.1: Given two PRMs Ω1 = (Π1, Θ1) and Ω2 = (Π2, Θ2) with the same bandwidth, i.e., Θ1 = Θ2 = Θ, if 0 < Π1 ≤ Π2, then ∀t, sbf1(t) ≥ sbf2(t), i.e., Ω1 is better than Ω2.

Proof: We show that ∀t, sbf1(t) ≥ usbf1(t), and then it follows sbf1(t) ≥ lsbf1(t) ≥ usbf1(t) ≥ sbf2(t)

We have Θ1 = Π1/Π2, so Θ1 = Π1/Π2. Thus ∀t,

\[
\text{sbf}_1(t) - \text{usbf}_1(t) = \frac{\Theta_1}{\Pi_1} (t - 2(\Pi_1 - \Theta_1)) - \frac{\Theta_1}{\Pi_1} (t - (\Pi_1 - \Theta_1)) = \frac{\Theta_1}{\Pi_1} ((\Pi_1 - \Theta_1) - 2(\Pi_1 - \Theta_1))
\]

\[
= \frac{\Theta_1}{\Pi_1} ((\Pi_1 - \Theta_1) - 2(\Pi_1 - \Theta_1)) = \frac{\Theta_1}{\Pi_1} ((\Pi_1 - \Theta_1) - 2(\Pi_1 - \Theta_1)) = \frac{\Theta_1}{\Pi_1} (\Pi_1 - 2)(\Pi_1 - \Theta_1)
\]

0 < Π1 ≤ Π2 =⇒ Π1 ≥ 2 and Π1 ≥ Θ1, therefore ∀t, sbf1(t) – usbf1(t) ≥ 0, and thus sbf1(t) ≥ lsbf1(t) ≥ usbf1(t) ≥ sbf2(t).

Note that in this proof, we exploit the fact that although by definition \text{sbf}_1(t) = \max(\frac{\Theta_1}{\Pi_1}(t - 2(\Pi_1 - \Theta_1)), 0) and \text{usbf}_1(t) = \max(\frac{\Theta_1}{\Pi_1}(t - (\Pi_1 - \Theta_1)), 0), it suffices to show ∀t, \frac{\Theta_1}{\Pi_1}(t - 2(\Pi_1 - \Theta_1)) ≥ \frac{\Theta_1}{\Pi_1}(t - (\Pi_1 - \Theta_1)) because ∀x, y ∈ ℝ, max(x, y) ≥ max(y, x) if x ≥ y.

From Lemma 3.1, if 0 < Π1 ≤ Π2, then Ω1 is better than Ω2. If Π1 = 1, then Ω1 = Ω2, and by definition, Ω2 is better than Ω2. We already showed if Π1 > Π2, then Ω1 is not better than Ω2. Therefore, the only unknown case is 0 < Π1 < 1, which is addressed by Lemmas 3.2 and 3.3. We show that when \frac{\Pi_1}{\Pi_2} ∈ (\frac{1}{2}, 1), Ω_2 is better than Ω2 if and only if the ratio \frac{\Pi_1}{\Pi_2} is in the set \{\frac{k+1}{2k+1}, k ∈ ℕ\}. Note that \{\frac{k+1}{2k+1}, k ∈ ℕ\} gives a non-increasing series \{1, \frac{2}{3}, \frac{3}{5}, \frac{4}{7}, \ldots\} with limit \lim_{k→∞} \frac{k+1}{2k+1} = \frac{1}{2}.

Lemma 3.2: Given two PRMs Ω1 = (Π1, Θ1) and Ω2 = (Π2, Θ2) with the same bandwidth, i.e., Ω1 = Ω2, if \exists k ∈ ℕ such that \frac{\Pi_1}{\Pi_2} ∈ \{\frac{k+1}{2k+1}, k ∈ ℕ\}, then ∀t, \text{sbf}_1(t) ≥ \text{sbf}_2(t), i.e., Ω_1 is better than Ω_2.

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 \text{sbf}_1 intersects \text{sbf}_2 and P denote the points where \text{lsbf}_1 intersects \text{sbf}_2.

From Equation 6, when \frac{\Pi_1}{\Pi_2} > \frac{1}{2}, ∀t, \text{lsbf}_1(t) < \text{usbf}_1(t). And since Π1 ≤ Π2, obviously ∀t \text{usbf}_1(t) ≤ \text{usbf}_2(t). Therefore usbf_1 lies between lsbf and usbf, and it must intersect \text{sbf}_2. Let Q and R denote the points where \text{sbf}_2 intersects the horizontal and sloped segments of usbf_2, respectively.

Since \frac{\Pi_1}{\Pi_2} = \frac{k+1}{2k+1}, we introduce two auxiliary variables B and c: B = Π1 = Σ1 and Π1 = c(2k + 1), cB = (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:

\[
\text{lsbf}_2(t) = \max(B(t - 2c(k + 1)(1 - B)), 0)
\]

(7)

\[
\text{usbf}_2(t) = \max(B(t - c(2k + 1)(1 - B)), 0)
\]

(8)

\[
t_A = c(2k + 1)(1 - B)
\]

(9)

\[
\text{lsbf}_2(t) = \max(B(t - c(2k + 1)(1 - B)), 0)
\]

(10)

\[
t_0 = t_B - c(1 - B)
\]

(11)

\[
t_R = t_P + cB
\]

(12)

where m,n ∈ 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_1, there are three possible scenarios, as illustrated in Figure 3.

The key part of the proof is the following observation: first, if sbf_1 and sbf_2 interleave with each another, i.e., \exists t’ such that \text{sbf}_1(t') < \text{sbf}_2(t'), then the interleaving point (t', \text{sbf}_1(t')) can only lie within the region between lsbf and usbf_2, because lsbf_2(t') ≤ \text{sbf}_1(t') < \text{sbf}_2(t') ≤ usbf_2(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* in Figure 3(a)), then let t_A be the coordinate of the nearest type A point such that t_A ≥ t’. From t_A to t*, sbf keeps increasing at a slope of 1, which is the maximum slope that an sbf can increase at. Since
Given two PRMs \( \Omega_a = (\Pi_a, \Theta_a) \) and \( \Omega_b = (\Pi_b, \Theta_b) \) with the same bandwidth, i.e., \( \Theta_a = \Theta_b \), if \( \exists k \in \mathbb{N} \) and \( \frac{2k+2}{2k+3} < \varepsilon < 1 \), such that \( \frac{\Pi_a}{\Pi_b} = \frac{k+1}{2k+1} \), then \( \exists \mathcal{A} \) such that \( \text{sbf}_b(\mathcal{A}) < \text{sbf}_a(\mathcal{A}) \), i.e., \( \Omega_b \) is not better than \( \Omega_a \).

Proof: The structure of the proof is similar to the proof of Lemma 3.2. Define the auxiliary variables \( B \) and \( c \) as: \( B = \Theta_b \), \( \Pi_b = c(k + \varepsilon) \), \( \Pi_a = c(2k + 1) \). It can be derived that,

\[
\text{sbf}_b(t) = \max(B(t - 2c(k + \varepsilon)(1 - B)), 0) \\
\text{usbf}_b(t) = \max(B(t - c(2k + 1)(1 - B)), 0) \\
t_a = c(2k + 1)(1 - B) + nc(2k + 1) \\
t_p = 2c(k + \varepsilon)(1 - B) + mc(k + \varepsilon) \\
t_q = t_p - c(1 - B)(2 \varepsilon - 1) \\
t_r = t_p + cB(2 \varepsilon - 1)
\]

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 interlaces with usbf, and furthermore \( \text{sbf}_a(t_a) < \text{sbf}_b(t_a) \), i.e., \( \Omega_b \) is not better than \( \Omega_a \).

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 \( t_a - t_p < cB(2 \varepsilon - 1) \):

\[
t_a - t_p = c(2k + 1)(1 - B) + nc(2k + 1) - 2c(k + \varepsilon)(1 - B) - (2c(k + \varepsilon))(2k + 1 - c(k + \varepsilon))
\]

Note that

\[
\text{sbf}(t) \geq \text{sbf}_b(t)
\]

Given Lemmas 3.1 and 3.2, the only remaining case that we need to check is when the ratio \( \frac{\Pi_b}{\Pi_a} \) lies in \( \left( \frac{1}{2}, 1 \right) \) but is not in the set \( \left\{ \frac{k+1}{2k+1}, k \in \mathbb{N} \right\} \). In this case, since \( \lim_{k \to \infty} \frac{k+1}{2k+1} = \frac{1}{2} \)

the ratio \( \frac{\Pi_b}{\Pi_a} \) should lie between two consecutive elements in the series \( \left\{ \frac{k+1}{2k+1}, k \in \mathbb{N} \right\} \), i.e., \( \exists k', \frac{k'+1}{2k'+1} < \frac{\Pi_b}{\Pi_a} < \frac{k'+2}{2k'+1} \). With an auxiliary variable \( \varepsilon \in \mathbb{R} \), such condition is equivalent to \( \frac{\Pi_b}{\Pi_a} = \frac{k+1}{2k+1} \) and \( \frac{2k+2}{2k+3} < \varepsilon < 1 \). Lemma 3.3 addresses this case.
Therefore,
\[ t_A - t_P - cB(2e - 1) = c[(2k + 1)(1 - e) - (2e - 1)] < 0 \] (22)

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

\[

t_A - t_P - c(B - 1)(2e - 1) \\
= c(2k + 1)(1 - B) + (k + 1)c(2k + 1) \\
- 2c(k + e)(1 - B) - (2k + 1)c(k + e) \\
- c(B - 1)(2e - 1) \\
= c(2k + 1)(1 - e) > 0.
\] (23)

Hence, the lemma holds.

We define the \( \Gamma \) function as follows.

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

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

**Theorem 3.5:** Given two PRMs \( \Omega = (\Pi, \Theta) \) and \( \Omega = (\Pi', \Theta') \) with the same bandwidth, \( \Omega' \) is better than \( \Omega \) if and only if \( \Omega' \in E(\Omega) \), where \( E(\Omega) \) is a set of PRMs: \( E(\Omega) = (\Omega, \Theta) = \{\Omega' = (\Pi', \Theta')|\frac{\Pi}{\Pi'} < \frac{1}{2} \} \cup \{\Pi|\exists k \in \mathbb{N}, \frac{\Pi}{\Pi'} = \frac{k+1}{2k+1} \} \in \Gamma(\Pi)\} \).

Note that by Definition 3.4, \( \Pi_k \in \Gamma(\Pi_n) \) (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 \( \Omega \), i.e., if \( \Omega \) 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 \( C = (W, I, A) \) is defined to be \( I = E(\Omega_0) = (\Pi_0, \Theta_0) \), 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 \( C = (W, I, A) \) is defined to be \( I = (B, P_I) \), in which \( B = \frac{\Pi_0}{\Pi} \) and \( P_I = \frac{\Pi}{\Pi_0} \), where \( \Omega_0 = (\Pi_0, \Theta_0) \) is a bandwidth-optimal PRM that guarantees the schedulability of \( C \).

To transform a bandwidth-periods interface \( I = (B, P) \) into a model-set one, we simply construct a PRM for each period \( p_i \) in \( P \) as \( (p_i, B \ast p_i) \), as illustrated in Example 1. Obviously, if a leaf component \( C \) is schedulable by PRMs, then \( \Pi_0 > 0 \); meaning there exists a PRM \( \Omega_0 \) that can schedule \( C \).

**Example 1 (Interface Generation):** Given a leaf component \( C = (W, I, A) \) 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 \( I = (B, P) \), where \( B = 0.12 \) and \( P = \Gamma(5) \). For example, \( 3 \in \Gamma(5) \), so that the PRM \( \Omega' = (3, 3 \ast 0.12) \in E(\Omega_0) \). From Theorem 3.5, \( \forall t \in \Omega_0 \), \( \Omega_0(t) \geq \Omega(t) \), 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 \( \Pi \) of the selected PRM is propagated to all components in the tree so that each component can choose the specific PRM with period \( \Pi \) 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 \( C = (W, I, A) \) is given by \( I = (B, P) \) where \( B = \bigcup_i B_i \), \( P = \bigcap_i P_i \), where each \( I_i = (B_i, P_i) \) 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 \( P = \bigcap_i P_i \neq \emptyset \). 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 \( \{ l_0 = (B_0, \Gamma(\Pi_0)), \ldots, l_m = (B_m, \Gamma(\Pi_m)) \} \). Assume each leaf component \( i \) is schedulable so that \( \forall i, \Pi_i > 0 \); then for any intermediate component \( C_j \), its interface \( I_j = (B_j, \Gamma(j)) \), which is obtained by Definition 3.8, is non-empty, i.e., \( P_j \neq \emptyset \).

Proof: Let \( I_j = (B_j, \Gamma(\Pi_j)), \ldots, I_{jm} = (B_{jm}, \Gamma(\Pi_{jm})) \} \) 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 \( P_j = \bigcap_{k=0}^{m} \Gamma(\Pi_k) \).

The period \( \Pi \) picked by the root component is propagated to all components in the tree, and that means each interface \( I_i = (B_i, P_i) \) is reduced to a single PRM \( \Omega_i = (\Pi_i, P_i) \). In the composition step, the parent component is obtained by taking the intersection of the periods of all its children, therefore the value \( \Pi \) picked by the root component is within the period set of every interface, i.e., \( \forall i, \Pi_i \in P_i \). This guarantees the PRM \( \Omega_i \) 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 \( C_i, \Omega_i \) can schedule its workload because it is in the interface.

For each intermediate component \( C_j = (W_j, I_j, A_j) \), the subinterfaces in its workload are reduced to single PRMs once the root component has determined the resource period, i.e., \( W_j = \{ I_{j1}, \ldots, I_{jm} \} = \Omega_{j1}, \ldots, \Omega_{jm} \}. \) To show the schedulability, we need to prove that \( \Omega_{j} \) can schedule \( W_j = \{ I_{j1}, \ldots, I_{jm} \} \). Notice that \( \Omega_{j} \) and \( \Omega_{j1}, \ldots, \Omega_{jm} \) have the same period \( \Pi \), 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 \( \{ T_{j1} = (\Pi, \Theta_{j1}), \ldots, T_{jm} = (\Pi, \Theta_{jm}) \} \) using resource model \( \Omega_{j} = (\Pi, \Theta_{j}) \). From Definition 3.8, we know \( \Theta_{j} = \sum \Theta_{jk} \).

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 \( \{ T_{j1}, \ldots, T_{jm} \} \) are released at the same time instance \( t_0 \), and the resource supply of \( \Omega_{j} \) 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 \( T = \{ (p, e_1), \ldots, (p, e_n) \} \), the PRM \( \Omega = (p, \Sigma) \) 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 \( \Omega \) also starts at \( t_0 \).

Proof: Without loss of generality, let \( t_0 = 0 \). Then within each period \( [kp, (k+1)p] \) for any \( k \in \mathbb{N} \), \( \Omega \) only needs to schedule the jobs released at \( kp \) and due by \( (k+1)p \), because jobs are only released at time instances \( 0, p, 2p, \ldots, kp \). The total amount of resources provided by \( \Omega \) within \( [kp, (k+1)p] \) is \( \sum e_i \), which is equal to the total resource demand of \( T \) within that time interval. Under a work-conserving policy, resources provided by \( \Omega \) 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 \( \Omega \) 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 \( P(h) = \text{True} \) 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 \( C_1, \ldots, C_n \) 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 \( B = \sum_{i \in I} B_i \) and \( P = \bigcap_{i \in I} P_i \). Here it is unnecessary to calculate the equivalent set for each PRM in the composed interface \( I = (\sum_{i \in I} B_i, \bigcap_{i \in I} P_i) \), because the \( \Gamma \) function has the following closure property such that for any PRM \( \Omega \in I = (\sum_{i \in I} B_i, \bigcap_{i \in I} P_i) \), the equivalent set \( E(\Omega) \subseteq I \).

Theorem 3.12 (Closure Property of \( \Gamma \):) Given any \( n \) positive real numbers \( p_1, \ldots, p_n \), let \( P_i = \Gamma(p_i) \) and \( P = \bigcap_{i \in I} P_i \). Assume \( \forall i \leq k, P(k) = \text{True} \).

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

Recall that \( \Gamma(x) = \{ y \mid 0 < \xi < \frac{1}{x} \} \bigcup \{ y \mid \exists \xi \in \mathbb{N}, \xi = \frac{k+1}{2x+1} \} \).\ni Case 1: \( y = x, \Gamma(y) = \Gamma(x) \subseteq \Gamma(x) \).

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

Case 3: \( y = \frac{k+1}{2x+1} x \). Then \( \exists z \in \Gamma(y), z = y \in \Gamma(x) \).

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 \( \Omega \) can schedule \( T \).

The theorem states that schedulability is guaranteed in our framework.
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 \( C = (W, I, A) \) for any scheduling tree is \( I = (\sum_{j \in S_i} B_j, \bigcap_{j \in S_i} P_j) \), in which

\[
C_i = \{W_1, (B_j, P_j), A_1\}, \ldots, C_n = \{W_n, (B_n, P_n), A_n\}
\]

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

\[
\forall i, i' = (\sum_{j \in S_i} B_j, \bigcap_{j \in S_i} P_j)
\]

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 \( \forall i \neq j, \bigcap_{i \in S_i} S_j = \Phi \). Therefore, from Definition 3.8,

\[
C = (W, I, A) = (\bigcup_{i, j} B_i, \bigcap_{i, j} P_j, A) = (\bigcup_{i, j} B_i, \bigcap_{i, j} P_j, A) \quad (24)
\]

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

The optimality is proved by contradiction. Given a set of leaf components \( \{C_1 = \{W_1, (B_1, P_1), A_1\}, \ldots, C_n = \{W_n, (B_n, P_n), A_n\}\} \), 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 \( C' = (W', I', A') \) 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 \( a_i, a_i + p_i, \ldots, a_i + k \cdot p_i \). Similarly, for a PRM \( \Omega = (\Pi, \Theta) \), let offset \( a_\Theta \) denote the time instant at which \( \Theta \) starts providing resource, i.e., \( \Theta \) guarantees at least \( \Theta \) units of resource supply over any time interval \([a_\Theta + k\Pi, a_\Theta + (k + 1)\Pi]\). Given a set of real-time tasks \( W = \{(p_1, e_1), \ldots, (p_n, e_n)\} \), 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 \( \Omega \) is usually higher than the bandwidth of \( W \). For example, in Figure 1, let \( W = \{T_1 = (5.1), T_2 = (5.1)\} \) be scheduled under the EDF policy and assume arbitrary offsets for \( T_1 \) and \( T_2 \). Given \( \Pi = 5 \), the bandwidth optimal PRM for \( W \) is \( \Omega = (5.3.5) \). As shown in Figure 5, the sbf\( \Omega \) curve intersects the dbf\( W \) curve at \( t = 5 \), i.e., if \( \Theta < 3.5 \), then sbf\( \Omega \) will be
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 $C_i = (W_i, I_i, A_i)$, suppose its corresponding interface task is $(a_i, \Pi_i, \Theta_i)$, where $a_i$ is the offset and $\Theta_i = (\Pi_i, \Theta_i)$ is a PRM in $I_i$. If $\Omega_i$ can schedule $W_i$ under some offset $a_i'$, then $\Omega_i$ 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, sbf$_i(t)$ gives the worst-case resource supply of $\Omega$ 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 $\Omega$ starts. Hence, for any $a_i$ and $a_i'$, sbf$_{(a_i, \Pi_i, \Theta_i)}(t) = sbf_{(a_i', \Pi_i, \Theta_i)}(t)$. 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 $C_2$ 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 $\Theta$'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 sbf-based schedulabilit-
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


