On the Complexity of Generating Optimal Interfaces for Hierarchical Systems

Arvind Easwaran, Madhukar Anand, Insup Lee, and Oleg Sokolsky
Department of Computer and Information Science,
University of Pennsylvania,
Philadelphia, PA, 19104, USA
{arvinde,anandm,lee,sokolsky}@cis.upenn.edu

Abstract

Compositional schedulability analysis of hierarchical real-time systems is a well-studied problem. Various techniques have been developed to abstract resource requirements of components in such systems, and schedulability has been addressed using these abstract representations (also called component interfaces). However, all these previous approaches incur resource overheads when they abstract components into interfaces. In this paper, we define various notions of resource optimality for component interfaces, and present techniques to generate such optimal interfaces. Specifically, we define two notions of optimality; one focusing on the average resource requirements of component, and the other focusing on the actual resource requirements of component. We develop efficient techniques to generate interfaces that are optimal under average resource requirements. However, under actual resource requirements, we show using an example that optimal interfaces are hard to generate.

1. Introduction

Hierarchical systems are used to model resource requirements of software developed under the component-based design paradigm. They represent a scheduling hierarchy under which resources are allocated from a parent component to its children, and each component is free to use its own scheduler. Schedulability analysis of such systems is a fundamental problem that must be addressed. Further, it is highly desirable that the analysis be *compositional*, *i.e.*, one should be able to deduce the system level resource requirements by composing interfaces that abstractly represent component level resource requirements.

Many studies have been proposed on compositional schedulability analysis of hierarchical systems [9, 19, 15, 20, 21, 1, 4, 17, 5, 23, 11, 22, 8, 7, 3, 10]. They have used resource models (*e.g.*, periodic [19, 15, 20, 8], bounded-delay [9, 21], EDP [7]) and demand bound functions [23, 22] to abstract component resource requirements. They have also proposed extensions supporting interactions between components using task abstractions [1, 4, 17] and resource-sharing protocols [5, 3, 10]. However, there is no work that characterizes the notion of *resource optimality* for such systems. Given a hierarchical system, resource optimality refers to a quantitative measure of the minimum total amount of resource required by this system. Without knowledge of this measure, it is not possible to quantitatively assess the various analysis techniques. Although previous studies have developed local component-level resource utilization bounds for interfaces [20], there is no global system-level measure for resource usage.

In this paper we address the issue of defining various notions of resource optimality for component interfaces. Specifically, we identify two notions of this optimality; *load-based* or *load optimality*, and *demand-based* or *demand optimality*. Intuitively, a component interface is load optimal, whenever the amount of resource required by the interface is the same as the average resource requirements of component workload. On the other hand, a component interface is demand optimal, whenever the amount of resource required by the interface is the same as the actual resource demand of component workload. In addition to defining notions of optimality, we also develop techniques that generate load optimal interfaces. Furthermore, since hierarchical systems can be either open (components are partially specified) or closed (complete knowledge of all the components in the system), we separately define and generate optimal interfaces for each of these cases.

Assuming component workloads are comprised of constrained deadline periodic tasks, we show that load optimal interfaces, for both open and closed hierarchical systems, can be generated in pseudo-polynomial time. Each load optimal interface is represented by a single constrained deadline periodic task, and hence the size of such an interface is constant in comparison to the input specification. Using an example, we also show that demand optimal interfaces are hard to generate for both open and closed hierarchical systems; the interface has exponentially larger number of tasks in comparison to number of tasks in the underlying component.

Techniques presented in this paper provide a baseline for resource utilization in hierarchical systems; they identify the minimum resource requirements of workloads scheduled un-

der a given scheduling hierarchy. In addition, these techniques also reveal an interesting trade-off between resource requirements of interfaces and their size in terms of number of tasks. In the example we consider for demand optimality, number of tasks in the interface is exponentially larger than number of tasks in the underlying component workload. Although, in general, this increase is unavoidable, demand imposed by a set of tasks in the workload may sometimes be represented by a smaller set of tasks, reducing the size of the interface. In Section 4.2, we characterize some of the cases when such a reduction is possible without loss of precision in demand. It is interesting to note that resource model based interfaces and load optimal interfaces offer an extreme case of such reduction, essentially over-approximating resource demand and collapsing the entire workload into a single task. The optimality characterization presented here, in turn, helps us to understand this trade-off between over-approximation of demand and interface size.

Related work. For real-time systems, there has been a growing attention to hierarchical systems. Since a two-level system was introduced by Deng and Liu [6], its schedulability has been analyzed under fixed-priority [12] and EDF [14, 16] scheduling. The bounded-delay resource model [18] has been proposed to achieve a clean separation in a multi-level hierarchical scheduling framework, and analysis techniques have been introduced for this resource model [9, 21].

Periodic resource model based interfaces, and their compositional analysis is a well known technique that has been studied extensively [15, 19, 20]. These models have been developed under fixed-priority [1, 4, 15, 19] and EDF [20] scheduling. Techniques have also been proposed to support interacting tasks [17] and mutually exclusive resource sharing between components [5, 3, 10]. Extensions to periodic models with more efficient interfaces have also been proposed [7]. There have also been studies on incremental analysis for hierarchical systems [23, 11, 22, 8]. They abstract resource requirements of components in the form of demand functions [23, 22], and bounded-delay [11] or periodic [8] resource models.

2. System model

In this section we describe hierarchical systems, state our assumptions, and define notions of resource optimality. In discrete computing systems, it is always possible to identify a time interval such that no successive events or actions can occur within this interval (*e.g.*, system clock frequency can be used to define this smallest time unit). We assume that all the parameters below are defined as multiples of this time unit.

2.1. Definitions and assumptions

Each instance of resource demand is called a *real-time job* and it can be specified using three parameters; release instant r with respect to an origin of time, maximum required resource capacity c, and relative deadline d with respect to the release instant. This job requires c units of resource in the time interval $(\mathrm{r}, \mathrm{r}+\mathrm{d}]$. A set of these jobs is called a real-time *job set* $\mathcal{J}$.

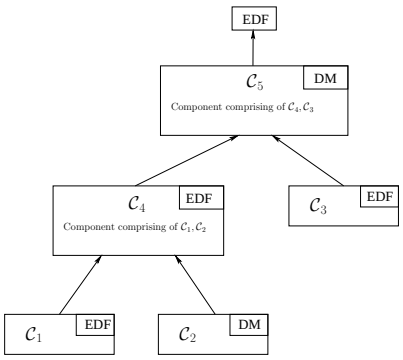


Figure 1. Hierarchical real-time system

A component in a hierarchical system is comprised of a real-time workload and a scheduling policy for the workload. We model workloads of components using job sets.

Definition 1 (Real-time component) *A* real-time component *$\mathcal{C}$ is specified as $\mathcal{C} = \langle \mathcal{W}, \mathcal{S} \rangle$, where $\mathcal{W}$ is a finite set of real-time components and job sets, and $\mathcal{S}$ is the scheduling policy. If $\mathcal{W}$ is comprised of only job sets then $\mathcal{C}$ is called* elementary component*. Otherwise $\mathcal{C}$ is called* non-elementary component*.*

This component structure can be viewed as a directed graph with components or job sets as nodes of the graph. Suppose $\mathcal{W}$ is comprised of components $\mathcal{C}_1, \ldots, \mathcal{C}_n$ and job sets $\mathcal{J}_1, \ldots, \mathcal{J}_m$. Then, there is an edge from node $\mathcal{C}$ to every node $\mathcal{C}_i$ and every node $\mathcal{J}_i$. A *hierarchical real-time system* is specified as $\mathcal{H} = \langle \mathcal{C}, \mathcal{S} \rangle$, where $\mathcal{C}$ denotes a real-time component whose underlying graph is a tree, and $\mathcal{S}$ denotes the scheduling algorithm under which $\mathcal{C}$ is scheduled. $\mathcal{C}$ is also called the *root component* of system $\mathcal{H}$. Figure 1 shows a hierarchical system, where $\mathcal{C}_1$,$\mathcal{C}_2$, and $\mathcal{C}_3$ are elementary components, and the whole system is scheduled under EDF on the hardware platform.

Assumptions. In this paper we assume that each job set is generated from a set of independent, constrained deadline periodic tasks. A constrained deadline periodic task $\tau = (\mathrm{T}, \mathrm{C}, \mathrm{D})$ has release separation T, maximum resource capacity requirement C, and relative deadline D, where $\mathrm{C} \leq \mathrm{D} \leq \mathrm{T}$. τ generates the job set $\{(t, \mathrm{C}, \mathrm{D}) | \ \mathrm{T} \text{ divides } t\}$. Given a periodic task set $\mathcal{T} = \{\tau_1 = (\mathrm{T}_1, \mathrm{C}_1, \mathrm{D}_1), \ldots, \tau_n = (\mathrm{T}_n, \mathrm{C}_n, \mathrm{D}_n)\}$, we denote its utilization as $U_{\mathcal{T}} = \sum_{i=1}^{n} \frac{\mathrm{C}_i}{\mathrm{T}_i}$, and without loss of generality assume $\mathrm{D}_1 \leq \ldots \leq \mathrm{D}_n$. We consider independent periodic tasks as a first step towards addressing the issue of resource optimality in hierarchical systems.

For simplicity of presentation, we assume that if a component in the hierarchical system has a job set in its workload, then it has exactly one job set. We also assume each component's scheduler is either Earliest Deadline First (EDF) or Deadline Monotonic (DM), and that the hierarchical system

is scheduled on a generalized uniprocessor platform having *bandwidth* $b\,(0 < b \leq 1)$. This platform is guaranteed to provide $b \times t$ units of processor capacity in every t time units.

2.2. Semantics of hierarchical systems

Consider an elementary component that uses EDF scheduler. Recall that the demand bound function of this component (dbf) gives its maximum resource demand in a given time interval [2, 13]. Equation (1) and Theorem 1 give the dbf and schedulability conditions respectively, for a component $\mathcal{C} = \langle \mathcal{T}, \mathrm{EDF} \rangle$ scheduled on an uniprocessor platform having bandwidth b [2].

$$\mathrm{dbf}_{\mathcal{C}}(t) = \sum_{i=1}^{n} \left(\left\lfloor \frac{t + \mathrm{T}_i - \mathrm{D}_i}{\mathrm{T}_i} \right\rfloor \mathrm{C}_i \right) \tag{1}$$

Theorem 1 (Schedulability under EDF) *Let* $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ *denote a set of constrained deadline periodic tasks. Component* $\mathcal{C} = \langle \mathcal{T}, \mathrm{EDF} \rangle$ *is schedulable on an uniprocessor platform having bandwidth* b *iff*

$$\forall t \text{ s.t. } 0 < t \leq L, \mathrm{dbf}_{\mathcal{C}}(t) \leq b \times t, \tag{2}$$

where $L = \min \left\{ \mathrm{LCM} + \max_{i=1}^{n} \mathrm{D}_i, \frac{U_{\mathcal{T}}(\max_{i=1}^{n}(\mathrm{T}_i - \mathrm{D}_i))}{b - U_{\mathcal{T}}} \right\}$, LCM *being the least common multiple of* $\mathrm{T}_1, \ldots, \mathrm{T}_n$.

The *schedulability load* of this component $\mathrm{LOAD}_{\mathcal{C}}$ is defined as $\max_{t \in (0,L]} \frac{\mathrm{dbf}_{\mathcal{C}}(t)}{t}$. If $b \geq \mathrm{LOAD}_{\mathcal{C}}$, then the uniprocessor platform can successfully schedule component $\mathcal{C}$. Note that since EDF is an optimal scheduler for periodic tasks, the *feasibility load* of task set $\mathcal{T}$ ($\mathrm{LOAD}_{\mathcal{T}}$) is also equal to $\mathrm{LOAD}_{\mathcal{C}}$. This means, if an uniprocessor has bandwidth smaller than $\mathrm{LOAD}_{\mathcal{C}}$, then it cannot successfully schedule $\mathcal{T}$ under any scheduling algorithm.

Similarly, consider an elementary component that uses DM scheduler. Recall that the request bound function of this component (rbf) gives the maximum resource requested in a given time interval [13]. Equation (3) and Theorem 2 give the rbf and schedulability conditions respectively, for a component $\mathcal{C} = \langle \mathcal{T}, \mathrm{DM} \rangle$ scheduled on an uniprocessor platform having bandwidth b [13].

$$\mathrm{rbf}_{\mathcal{C},i}(t) = \sum_{k \leq i} \left(\left\lceil \frac{t}{\mathrm{T}_k} \right\rceil \mathrm{C}_k \right) \tag{3}$$

Theorem 2 (Schedulability under DM) *Let* $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ *denote a set of constrained deadline periodic tasks. Component* $\mathcal{C} = \langle \mathcal{T}, \mathrm{DM} \rangle$ *is schedulable on an uniprocessor platform having bandwidth* b *iff*

$$\forall i, \exists t \in [0, \mathrm{D}_i] \text{ s.t. } \mathrm{rbf}_{\mathcal{C},i}(t) \leq b \times t \tag{4}$$

The *schedulability load* of this component is defined as $\mathrm{LOAD}_{\mathcal{C}} = \max_{i=1,\ldots,n} \min_{t \in (0,\mathrm{D}_i]} \frac{\mathrm{rbf}_{(\mathcal{C},i)}(t)}{t}$.

Given a job set, EDF is a dynamic-priority scheduler that selects for execution the job with earliest relative deadline. On the other hand, DM is a fixed-priority scheduler which prioritizes jobs based on the deadline parameter of tasks that generate those jobs. Then, for elementary components, scheduling its workload (constrained deadline periodic task set) under EDF or DM is straightforward as explained above. However, if a component $\mathcal{C}$'s workload comprises of other components, this is not the case. To schedule this workload under EDF we must present a set of jobs, and to schedule it under DM we must present a task set like periodic, sporadic, etc. In other words, each component $\mathcal{C}_i$ in the workload of $\mathcal{C}$ must be transformed into a set of tasks/jobs that $\mathcal{C}$'s scheduler can schedule. Furthermore, informally, these tasks/jobs should be such that the amount of resource required by them under $\mathcal{C}$'s scheduler, must be at least as much as the amount of resource required by component $\mathcal{C}_i$. We refer to these tasks/jobs as $\mathcal{C}_i$'s *interface*.

Definition 2 (Component interface) *We define the interface of a task set to be the task set itself. Consider a component* $\mathcal{C} = \langle \mathcal{W}, \mathcal{S} \rangle$ *with* $\mathcal{W} = \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$. *Let* $\mathcal{C}$ *itself be scheduled under* $\mathcal{S}'$, *and* $\mathcal{I}_{\mathcal{W}}$ *denote the set of interfaces of workload* $\mathcal{W}$. $\mathcal{I}_{\mathcal{C}}$ *is an interface for* $\mathcal{C}$ *iff* $\mathcal{I}_{\mathcal{C}}$ *is schedulable under* $\mathcal{S}'$ *implies* $\mathcal{I}_{\mathcal{W}}$ *is schedulable under* $\mathcal{S}$. *In this definition, we assume that* $\mathcal{I}_{\mathcal{W}}$ *executes under* $\mathcal{S}$ *whenever* $\mathcal{I}_{\mathcal{C}}$ *is scheduled by* $\mathcal{S}'$.

Thus, given a component $\mathcal{C} = \langle \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}, \mathcal{S} \rangle$, the fundamental question in scheduling $\mathcal{C}$ is, "What is the interface that each $\mathcal{C}_i$ must present to $\mathcal{S}$?". In this paper, our goal is to answer this question, and in the process generate component interfaces that are *optimal* with respect to resource utilization. Schedulability conditions in Theorems 1 and 2 are agnostic to preemptions, and hence we ignore preemption overheads in this paper.

Synchronization. An important aspect of hierarchical systems, both open and closed, is that of synchronization between components. Given a component $\mathcal{C} = \langle \mathcal{W}, \mathcal{S} \rangle$, release of jobs in the underlying workload of some component $\mathcal{C}_i \in \mathcal{W}$, need not be synchronized with the release of jobs in the underlying workload of any other component in $\mathcal{W}$ (*Horizontal Synchronization*). Note that horizontal synchronization can be enforced in closed systems, but this is not the case in open systems. This means that our interface generation technique cannot make any assumptions regarding synchronization between interfaces of components in $\mathcal{W}$. Now, suppose we synchronize the release time of first job in a component in $\mathcal{W}$ with the release time of first job in its interface (*Vertical Synchronization*). Then, observe that this forced vertical synchronization does not enforce any horizontal synchronization constraints in the system. In other words, we can vertically synchronize every component in $\mathcal{W}$ with its interface, without assuming any (horizontal) synchronization between components in $\mathcal{W}$ or between interfaces of those components. Therefore, we assume vertical synchronization in the interface generation technique presented in this paper. Note that this assumption only synchronizes the release times of first jobs in component and interface, and does not enforce any synchronization between other jobs.

2.3. Optimality in hierarchical systems

The *feasibility load* $\text{LOAD}_{\mathcal{I}_\mathcal{C}}$ of a component interface $\mathcal{I}_\mathcal{C}$, is the smallest bandwidth required from an uniprocessor platform to successfully schedule tasks in $\mathcal{I}_\mathcal{C}$ under some scheduler. Similarly, given a set of interfaces $\mathcal{I} = \{\mathcal{I}_{\mathcal{C}_1}, \ldots, \mathcal{I}_{\mathcal{C}_n}\}$ and a scheduler $\mathcal{S}$, the *schedulability load* $\text{LOAD}_{\mathcal{I},\mathcal{S}}$ is the smallest bandwidth required from an uniprocessor platform to successfully schedule $\mathcal{I}$ under $\mathcal{S}$. For instance, if $\mathcal{I}$ comprises of constrained deadline periodic tasks and $\mathcal{S}$ is EDF or DM, then we have presented these loads in the previous section. Also, a non-elementary component $\mathcal{C} = \langle\{\mathcal{C}_1, \ldots, \mathcal{C}_n\}, \mathcal{S}\rangle$ is said to be *feasible* on an uniprocessor platform having bandwidth b, if there exists interface set $\mathcal{I} = \{\mathcal{I}_{\mathcal{C}_1}, \ldots, \mathcal{I}_{\mathcal{C}_n}\}$ such that $\mathcal{I}$ is schedulable under $\mathcal{S}$ on this resource.

Load-based optimality. The first notion of resource optimality we define is called *load* optimality, which characterizes average resource requirements of components.

Definition 3 (Local load optimality) *Given a constrained deadline periodic task set, its interface (task set itself) is defined to be locally load optimal. Consider component $\mathcal{C}_i = \langle\{\mathcal{C}_{i_1}, \ldots, \mathcal{C}_{i_m}\}, \mathcal{S}_i\rangle$. If $\mathcal{I}_i$ denotes the set of interfaces of workload $\{\mathcal{C}_{i_1}, \ldots, \mathcal{C}_{i_m}\}$, then $\mathcal{I}_{\mathcal{C}_i}$ is* locally load optimal *iff* $\text{LOAD}_{\mathcal{I}_{\mathcal{C}_i}} = \text{LOAD}_{\mathcal{I}_i,\mathcal{S}_i}$.

If interface $\mathcal{I}_{\mathcal{C}_i}$ is locally load optimal, then the interface generation process does not introduce any overheads with respect to the average resource requirements of component $\mathcal{C}_i$. In other words, if $\mathcal{C}_i$ is feasible on an uniprocessor platform having bandwidth b, then $\mathcal{I}_{\mathcal{C}_i}$ is also feasible on that resource. There could be many possible locally load optimal interfaces for $\mathcal{C}_i$, and not all of them may result in a load optimal interface for $\mathcal{C}_i$'s parent. Therefore, for closed systems, since we have knowledge of other components scheduled with $\mathcal{C}_i$, it is useful to identify those locally load optimal interfaces of $\mathcal{C}_i$ that also result in load optimality for its parent.

Definition 4 (Global load optimality) *Given a constrained deadline periodic task set, its interface (task set itself) is defined to be globally load optimal. Consider component $\mathcal{C} = \langle\{\mathcal{C}_1, \ldots, \mathcal{C}_n\}, \mathcal{S}\rangle$. Let $\mathcal{I} = \{\mathcal{I}_{\mathcal{C}_1}, \ldots, \mathcal{I}_{\mathcal{C}_n}\}$ denote the set of locally load optimal interfaces of the workload of $\mathcal{C}$, and $\mathcal{I}_\mathcal{C}$ denote a locally load optimal interface generated from $\mathcal{I}$. Also, let $\mathcal{I}'$ denote any other set of locally load optimal interfaces of the workload of $\mathcal{C}$, and $\mathcal{I}'_\mathcal{C}$ denote the locally load optimal interface generated from $\mathcal{I}'$. Then, each interface $\mathcal{I}_{\mathcal{C}_i} \in \mathcal{I}$ is* globally load optimal *iff* $\text{LOAD}_{\mathcal{I}_\mathcal{C}} \leq \text{LOAD}_{\mathcal{I}'_\mathcal{C}}$ *for every* $\mathcal{I}'_\mathcal{C}$.

In Section 3 we present a technique that generates globally load optimal interfaces for both open and closed systems. Thus, we are able to generate globally load optimal interfaces for $\mathcal{C}_i$, even without knowledge of other components scheduled with it. Following theorem describes the significance of load optimal interfaces.

Theorem 3 *Consider a hierarchical system $\mathcal{H} = \langle\mathcal{C}, \mathcal{S}\rangle$, with $\mathcal{C}_1, \ldots, \mathcal{C}_m$ denoting all the components in the tree rooted at $\mathcal{C}$. Let interfaces $\mathcal{I} = \{\mathcal{I}_{\mathcal{C}_1}, \ldots, \mathcal{I}_{\mathcal{C}_m}\}$ of all these components be globally load optimal. Also, let $\mathcal{I}_\mathcal{C}$ denote a load optimal interface for $\mathcal{C}$ generated from $\mathcal{I}$. If each $\mathcal{C}_i$ is scheduled exclusively on an uniprocessor having bandwidth $\text{LOAD}_{\mathcal{I}_{\mathcal{C}_i}}$ $(= b_i)$, then $\mathcal{C}$ is not schedulable on any uniprocessor having bandwidth b that is smaller than $\text{LOAD}_{\mathcal{I}_\mathcal{C}}$.*

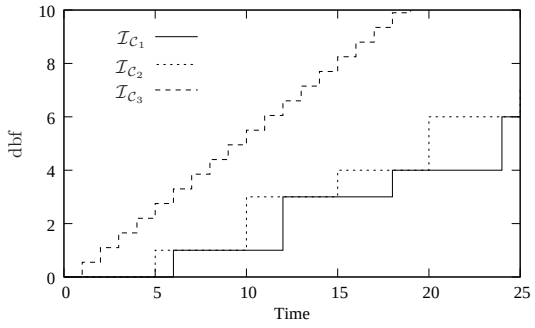
Proof We use induction on the height of node $\mathcal{C}$.

Base Case [$\mathcal{C}$ is a task set] : In this case $\text{LOAD}_{\mathcal{I}_\mathcal{C}} = \text{LOAD}_\mathcal{C}$, and by definition $\text{LOAD}_\mathcal{C}$ is the smallest bandwidth required from an uniprocessor platform to schedule $\mathcal{C}$.

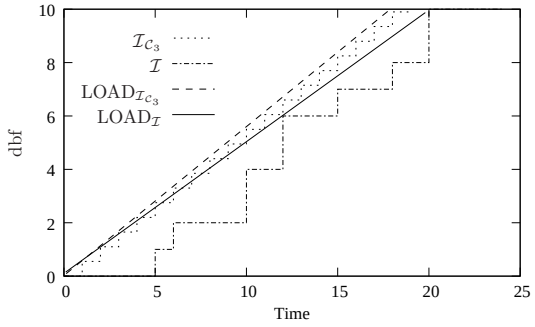
Induction Case [$\mathcal{C}$ is at height $n+1$] : Let $\mathcal{S}'$ denote the scheduler used by component $\mathcal{C}$ to schedule its workload, and $\mathcal{I}_n$ denote the set of globally load optimal interfaces of all components at height n. Then, from Definitions 3 and 4 we get $\text{LOAD}_{\mathcal{I}_\mathcal{C}} = \text{LOAD}_{\mathcal{I}_n,\mathcal{S}'}$ and $\text{LOAD}_{\mathcal{I}_\mathcal{C}} \leq \text{LOAD}_{\mathcal{I}'_n,\mathcal{S}'}$, where $\mathcal{I}'_n$ denotes any other set of locally load optimal interfaces of all components at height n. Also, by definition, $\text{LOAD}_{\mathcal{I}_\mathcal{C}}$ denotes the smallest bandwidth required from an uniprocessor platform to successfully schedule component $\langle\mathcal{I}_n, \mathcal{S}'\rangle$. Consider an uniprocessor platform having bandwidth $b < \text{LOAD}_{\mathcal{I}_\mathcal{C}}$. Then, some interface $\mathcal{I}_{\mathcal{C}_l} \in \mathcal{I}_n$ is not schedulable on this platform, *i.e.*, bandwidth of the uniprocessor available for scheduling workloads in $\mathcal{I}_{\mathcal{C}_l}$ is smaller than $\text{LOAD}_{\mathcal{I}_{\mathcal{C}_l}}$. But from induction hypothesis, $b_l (= \text{LOAD}_{\mathcal{I}_{\mathcal{C}_l}})$ denotes the smallest bandwidth required from an uniprocessor platform to successfully schedule component $\mathcal{C}_l$. This proves the result. □

Demand-based optimality. Consider a hierarchical system defined as follows. $\mathcal{C}_1 = \langle\{(6,1,6),(12,1,12)\}, \text{EDF}\rangle$, $\mathcal{C}_2 = \langle\{(5,1,5),(10,1,10)\}, \text{EDF}\rangle$, and $\mathcal{C}_3 = \langle\{\mathcal{C}_1, \mathcal{C}_2\}, \text{EDF}\rangle$. Consider the interfaces $\mathcal{I}_{\mathcal{C}_1} = (1, 0.25, 1), \mathcal{I}_{\mathcal{C}_2} = (1, 0.3, 1)$, and $\mathcal{I}_{\mathcal{C}_3} = (1, 0.55, 1)$. It is easy to see that $\text{LOAD}_{\mathcal{I}_{\mathcal{C}_1}} = \text{LOAD}_{\mathcal{C}_1}$, $\text{LOAD}_{\mathcal{I}_{\mathcal{C}_2}} = \text{LOAD}_{\mathcal{C}_2}$, and $\text{LOAD}_{\mathcal{I}_{\mathcal{C}_3}} = \text{LOAD}_{\mathcal{I}',\text{EDF}}$, where $\mathcal{I}' = \{\mathcal{I}_{\mathcal{C}_1}, \mathcal{I}_{\mathcal{C}_2}\}$. Hence $\mathcal{I}_{\mathcal{C}_1}$ and $\mathcal{I}_{\mathcal{C}_2}$ are globally load optimal. The demand bound functions of these interfaces and components are plotted in Figure 2(a). However, as seen in Figure 2(b), $\text{LOAD}_{\mathcal{I}_{\mathcal{C}_3}} > \text{LOAD}_\mathcal{I}$, where $\mathcal{I} = \{(6,1,6),(12,1,12),(5,1,5),(10,1,10)\}$. Assuming vertical synchronization, it is easy to see that total resource requirements of $\mathcal{I}$ is equal to the total resource requirements of components $\mathcal{C}_1$ and $\mathcal{C}_2$ combined. Therefore, $\mathcal{I}$ is an interface for component $\mathcal{C}_3$. This shows that even though $\mathcal{I}_{\mathcal{C}_3}$ is only feasible on an uniprocessor platform having bandwidth $\text{LOAD}_{\mathcal{I}_{\mathcal{C}_3}}$, component $\mathcal{C}_3$ itself is feasible on a platform having bandwidth strictly smaller than $\text{LOAD}_{\mathcal{I}_{\mathcal{C}_3}}$. Thus, although a load optimal interface minimizes the average resource utilization, it still incurs overheads with respect to the actual demand of underlying component.

We now introduce the notion of demand optimality, which characterizes interfaces that are both necessary and sufficient



(a) Load optimal interface



(b) Sub-optimality

Figure 2. Load vs. demand optimality

for component schedulability. We define two notions of demand optimality; local optimality for open systems and global optimality for closed systems. When the hierarchical system under consideration is an open system, a component in the system is not aware of other components scheduled with it. Therefore, when generating an interface for such a component, we must consider the worst-case interference from other components scheduled with it. This interference is defined as follows.

Definition 5 (Zero slack assumption) *Consider component $\mathcal{C}_i = \langle\{\mathcal{C}_{i_1}, \ldots, \mathcal{C}_{i_m}\}, \mathcal{S}_i\rangle$. Let $\mathcal{I}_i$ denote the set of interfaces of workload $\{\mathcal{C}_{i_1}, \ldots, \mathcal{C}_{i_m}\}$. If $\mathcal{C}_i$ is part of an open hierarchical system and $\mathcal{I}_i$ is schedulable under $\mathcal{S}_i$, then we assume that the schedule of $\mathcal{I}_i$ has* zero slack. *In other words, the amount of resource supplied to $\mathcal{C}_i$ is such that each job in $\mathcal{I}_i$ finishes as late as possible, subject to satisfaction of all job deadlines.*

The following definition then gives our notion of demand optimality for open systems.

Definition 6 (Local demand optimality (open systems)) *Given a constrained deadline periodic task set, its interface (task set itself) is defined to be locally demand optimal. Consider component $\mathcal{C}_i = \langle\{\mathcal{C}_{i_1}, \ldots, \mathcal{C}_{i_m}\}, \mathcal{S}_i\rangle$. Let $\mathcal{I}_i$ denote the set of interfaces of workload $\{\mathcal{C}_{i_1}, \ldots, \mathcal{C}_{i_m}\}$. Interface $\mathcal{I}_{\mathcal{C}_i}$ is locally demand optimal iff assuming zero slack for $\mathcal{C}_i$ (and hence for $\mathcal{I}_{\mathcal{C}_i}$), schedulability of $\mathcal{I}_i$ under $\mathcal{S}_i$ implies feasibility of $\mathcal{I}_{\mathcal{C}_i}$.*

If $\mathcal{I}_{\mathcal{C}_i}$ is locally demand optimal, then the interface generation process does not introduce any overheads with respect to the resource requirements of $\mathcal{I}_i$, assuming zero slack in the schedule. Locally demand optimal interfaces need not always lead to global optimality, because the actual interference from other components may be smaller than the worst-case scenario considered here. Therefore, in closed systems, where we have accurate knowledge of this interference, we can define global demand optimality as follows.

Definition 7 (Global demand optimality (closed systems)) *Given a constrained deadline periodic task set, its interface (task set itself) is defined to be globally demand optimal. Consider a hierarchical system $\langle\mathcal{C}, \mathcal{S}\rangle$. Let $\mathcal{I}_\mathcal{C}$ denote an interface for $\mathcal{C}$ generated using some set of interfaces for all components in $\mathcal{C}$. $\mathcal{I}_\mathcal{C}$ is globally demand optimal if and only if, whenever there exists interfaces for all the components in $\mathcal{C}$ such that the components are schedulable using those interfaces, $\mathcal{I}_\mathcal{C}$ is feasible.*

Intuitively, the significance of globally demand optimal interfaces can be explained as follows. Given a hierarchical system, if it is possible to schedule all the tasks in the system using some set of interfaces, then globally demand optimal interfaces will also schedule them.

3. Load optimal interfaces

3.1. Interface generation technique

The following definition presents an interface that is globally load optimal in both open and closed hierarchical systems.

Definition 8 (Schedulability load based abstraction) *If $\mathcal{C}_i$ is a constrained deadline periodic task set then abstraction $\mathcal{I}_{\mathcal{C}_i} = \mathcal{C}_i$. Otherwise $\mathcal{I}_{\mathcal{C}_i} = \{\tau_i = (1, \text{LOAD}_{\mathcal{W}_{\mathcal{C}_i}, \mathcal{S}_i}, 1)\}$, where $\mathcal{S}_i$ denotes scheduler used by $\mathcal{C}_i$, and $\mathcal{W}_{\mathcal{C}_i}$ denotes the set of schedulability load based abstractions of $\mathcal{C}_i$'s children. τ_i is a periodic task, and the release time of its first job coincides with the release time of the first job in component $\mathcal{C}_i$ (vertical synchronization).*

We now prove that the schedulability load based abstraction $\mathcal{I}_{\mathcal{C}_i}$ is a globally load optimal interface for component $\mathcal{C}_i$.

Lemma 4 *Given a component $\mathcal{C}_i = \langle\mathcal{T}, \mathcal{S}_i\rangle$, where $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ is a constrained deadline periodic task set, $\mathcal{I}_{\mathcal{C}_i}$ as given by Definition 8 is an interface for $\mathcal{C}_i$.*

Proof Let us define $L_\mathcal{T} = \{t\}_i$ to be the sequence of non-decreasing time units measured from the point when the first job in $\mathcal{T}$ is released. Each t_i in the sequence is such that there are no jobs of $\mathcal{T}$ with non-zero remaining executions, assuming $\mathcal{T}$ executes under $\mathcal{S}_i$ whenever $\mathcal{I}_{\mathcal{C}_i}$ executes. We prove this lemma by induction on this sequence.

Base case [t_1] : In the interval $(0, t_1]$, jobs of $\mathcal{T}$ are always executing whenever $\mathcal{I}_{\mathcal{C}_i}$ is executing. This means that the amount of resource used by $\mathcal{T}$ in $(0, t_1]$ is exactly equal to

the amount of resource used by $\mathcal{I}_{\mathcal{C}_i}$ in that interval. By Definition 8, the amount of resource used by $\mathcal{I}_{\mathcal{C}_i}$ in an interval $(0, t']$, for all $t' \leq t_1$, is $\text{LOAD}_{\mathcal{T},\mathcal{S}_i} \times t'$ (vertical synchronization).

If $\mathcal{S}_i = \text{EDF}$ the amount of resource required by $\mathcal{T}$ to meet all job deadlines up to t' is upper bounded by $\text{dbf}_{\mathcal{C}_i}(t')$. The result follows because $\text{LOAD}_{\mathcal{T},\mathcal{S}_i} \times t' \geq \text{dbf}_{\mathcal{C}_i}(t')$ by definition. If $\mathcal{S}_i = \text{DM}$, the amount of resource required by $\mathcal{T}$ to meet all job deadlines up to t', such that these are jobs of task τ_k, is upper bounded by $t' \times \min_{t_k \in (0,\text{D}_k]} \frac{\text{rbf}_{\mathcal{C}_i,k}(t_k)}{t_k}$. The result then follows because $\text{LOAD}_{\mathcal{T},\mathcal{S}_i} \times t' \geq t' \times \max_{k=1,\ldots,n} \min_{t_k \in (0,\text{D}_k]} \frac{\text{rbf}_{\mathcal{C}_i,k}(t_k)}{t_k}$.

Induction case [t_{n+1}] : By induction hypothesis, all job deadlines in the interval $(0, t_n]$ are satisfied by jobs of $\mathcal{I}_{\mathcal{C}_i}$ released before t_n. Observe that $t_{n+1} \geq t_n + 1$, and the first release of any job of $\mathcal{T}$ after t_n (t_r) can only occur at or after $t_n + 1$. Also, since period of task τ_i in interface $\mathcal{I}_{\mathcal{C}_i}$ is 1, some release of job of $\mathcal{I}_{\mathcal{C}_i}$ after t_n will coincide with t_r. Using logic similar to the base case, we can show that all deadlines of jobs of $\mathcal{T}$ in the interval $(t_r, t_{n+1}]$, are satisfied by jobs of $\mathcal{I}_{\mathcal{C}_i}$ released in that interval. This combined with the induction hypothesis proves the result. □

Theorem 5 *Interface $\mathcal{I}_{\mathcal{C}_i}$ generated using Definition 8 is an interface for component $\mathcal{C}_i$.*

Proof We prove this theorem by induction on the height of $\mathcal{C}_i$ in the underlying subtree rooted at $\mathcal{C}_i$.

Base case [$\mathcal{C}_i$ is a task set] : Definitions 2 and 8.

Induction case [$\mathcal{C}_i$ is at height $n+1$] : If $n = 1$, then $\mathcal{C}_i$'s child is a task set, and from induction hypothesis and Lemma 4 we directly get the result. If $n > 1$, then let $\mathcal{C}_i = \langle \{\mathcal{C}_{i_1}, \ldots, \mathcal{C}_{i_m}\}, \mathcal{S}_i \rangle$. Lemma 4 can again be used to show that $\mathcal{I}_{\mathcal{C}_i}$ is an interface for the component $\langle \mathcal{I}_i, \mathcal{S}_i \rangle$, where $\mathcal{I}_i$ is the set of interfaces of workload $\{\mathcal{C}_{i_1}, \ldots, \mathcal{C}_{i_m}\}$. But, by induction hypothesis, each $\mathcal{I}_{\mathcal{C}_{i_j}} \in \mathcal{I}_i$ is itself an interface for $\mathcal{C}_{i_j}$, and therefore $\mathcal{I}_{\mathcal{C}_i}$ is an interface for $\mathcal{C}_i$. □

We now prove that interface $\mathcal{I}_{\mathcal{C}_i}$ is globally load optimal.

Theorem 6 *Given component $\mathcal{C} = \langle \{\mathcal{C}_1, \ldots \mathcal{C}_i, \ldots, \mathcal{C}_n\}, \mathcal{S} \rangle$, if interfaces $\mathcal{I}_{\mathcal{C}}$ and $\mathcal{I} = \{\mathcal{I}_{\mathcal{C}_1}, \ldots, \mathcal{I}_{\mathcal{C}_i}, \ldots, \mathcal{I}_{\mathcal{C}_n}\}$ are as given by Definition 8, then $\mathcal{I}_{\mathcal{C}_i}$ is a globally load optimal interface.*

Proof We prove this theorem by induction on the height of $\mathcal{C}_i$ in the underlying subtree rooted at $\mathcal{C}$.

Base Case [$\mathcal{C}_i$ is a task set] : In this case, $\mathcal{I}_{\mathcal{C}_i}$ is globally load optimal directly from Definitions 4 and 8.

Induction Case [$\mathcal{C}_i$ is at height $n+1$] : From induction hypothesis we get that each $\mathcal{I}_{\mathcal{C}_i}$ is locally load optimal (interfaces of $\mathcal{C}_i$'s children of are globally load optimal). Since $\mathcal{W}_{\mathcal{C}} = \mathcal{I}$, $\text{LOAD}_{\mathcal{W}_{\mathcal{C}},\mathcal{S}} = \text{LOAD}_{\mathcal{I},\mathcal{S}}$. Also, since $\mathcal{I}_{\mathcal{C}} = (1, \text{LOAD}_{\mathcal{W}_{\mathcal{C}},\mathcal{S}}, 1)$, $\text{LOAD}_{\mathcal{I}_{\mathcal{C}}} = \text{LOAD}_{\mathcal{I},\mathcal{S}}$ and $\mathcal{I}_{\mathcal{C}}$ is locally load optimal from Definition 3. Furthermore, $\text{LOAD}_{\mathcal{I},\mathcal{S}} = \text{LOAD}_{\mathcal{I}}$, the feasibility load of $\mathcal{I}$. This is because all the tasks in $\mathcal{I}$ have period and deadline equal to 1, and EDF and DM are optimal schedulers for such tasks. Therefore, $\text{LOAD}_{\mathcal{I}_{\mathcal{C}}} = \text{LOAD}_{\mathcal{I}} \leq \text{LOAD}_{\mathcal{I}',\mathcal{S}}$, where $\mathcal{I}'$ denotes any other set of locally load optimal interfaces for $\mathcal{C}$'s workload. □

3.2 Discussions

Complexity. Interfaces in Definition 8 can be computed in pseudo-polynomial time with respect to input specification. $\text{LOAD}_{\mathcal{W}_{\mathcal{C}_i},\mathcal{S}_i}$ can be computed using Equation (2) or (4). Since these equations must be evaluated for all values of t in the range $(0, L]$ under EDF, and $(0, \text{D}_j]$ for each task $\tau_j \in \mathcal{W}_{\mathcal{C}_i}$ under DM, interface $\mathcal{I}_{\mathcal{C}_i}$ can be generated in pseudo-polynomial time. Also, interface $\mathcal{I}_{\mathcal{C}_i}$ only has $\mathcal{O}(1)$ storage requirements when compared to the input specification.

Task model. Although we have considered periodic tasks in this paper, interface generation technique in Definition 8 also generates load optimal interfaces for constrained deadline sporadic tasks, assuming all job release times are multiples of the basic time unit we defined in Section 2. The only modifications required in Definition 8 are that, (1) task τ_i is sporadic, and (2) τ_i is released whenever there are unfinished jobs active in $\mathcal{C}_i$, subject to these releases satisfying the minimum separation criteria. Using similar proof techniques, we can show that Theorems 5 and 6 hold for such interfaces as well.

Preemptions. In this paper, we have ignored preemptions because Theorems 1 and 2 are agnostic to such overheads. Existing studies assume preemption overheads can be upper bounded by a function that is monotonically decreasing with respect to task periods in interfaces (*e.g.*, [15, 8]). Under this assumption, our interface generation technique in Definition 8 will incur maximum preemption overhead. However, the technique can be modified such that task $\tau_i = (k, \text{LOAD}_{\mathcal{W}_{\mathcal{C}_i}} \times k, k)$, where k is any divisor of the GCD (greatest common divisor) of periods and deadlines of tasks in $\{\mathcal{W}_{\mathcal{C}_i}\} \bigcup \{\mathcal{W}_{\mathcal{C}_j} \,|j \neq i\}$. Here $\{\mathcal{C}_j \,|j \neq i\}$ denotes other components scheduled with $\mathcal{C}_i$. Thus, we can generate load optimal interfaces without forcing interface tasks to have period one.

Comparison to resource model based interfaces. It is well known that the feasibility load of interfaces generated using bounded delay [9, 21] or periodic [15, 20] resource models, is lower bounded by the schedulability load of underlying component. In fact, this schedulability load is achieved only when period Π for periodic models, or delay δ for bounded delay models, is 0 (see Theorems 7 and 8 in [20] and Theorems 4 and 5 in [21]). Note Π or $\delta = 0$ indicates that the interface is not realizable, because EDF and DM cannot schedule tasks generated from such models. In all other cases, the feasibility load of interface is strictly larger than the schedulability load of component. Hence, these interfaces are not load optimal. The reason for this sub-optimality is lack of vertical synchronization between the component and its interface. EDP resource model based interfaces can achieve load optimality whenever deadline of the model is equal to its capacity ($\Delta = \Theta$), and pe-

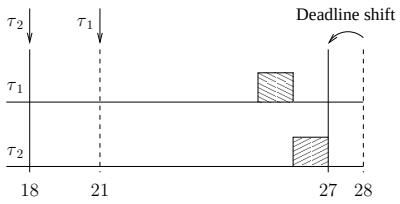


Figure 3. Partial schedule of component $\mathcal{C}_1$

riod $\Pi = 1$. Correctness of this statement follows from the fact that (1) in any time interval of length Π this model guarantees Θ units of resource, and (2) transformation from EDP model to periodic task is demand optimal (see Equation 6 and Definition 5.2 in [7]). Note that Π can also take values as described in the previous paragraph to account for preemption overheads.

4. Demand optimal interfaces

In this section, we present an example to show that size of demand optimal interfaces can be exponentially larger than the input specification. We then discuss scenarios under which load optimal interfaces also satisfy demand optimality.

4.1. Hardness of demand optimality

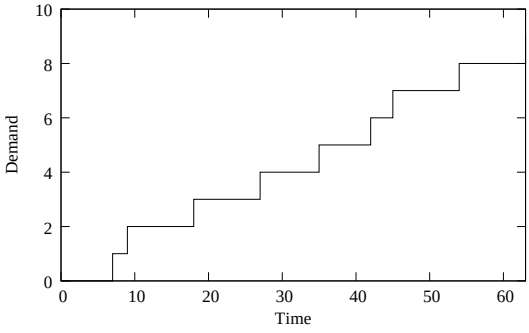
Before we present our example, we introduce some additional definitions. A constrained deadline, asynchronous periodic task set is specified as $\mathcal{T} = \{\tau_1 = (\mathrm{O}_1, \mathrm{T}_1, \mathrm{C}_1, \mathrm{D}_1), \ldots, \tau_n = (\mathrm{O}_n, \mathrm{T}_n, \mathrm{C}_n, \mathrm{D}_n)\}$, where each τ_i is a periodic task with offset O_i, exact separation T_i, worst case execution requirement C_i, and relative deadline D_i, such that $\mathrm{C}_i \leq \mathrm{D}_i \leq \mathrm{T}_i$. For each task τ_i, its jobs are released at times $\mathrm{O}_i, \mathrm{O}_i + \mathrm{T}_i, \mathrm{O}_i + 2\,\mathrm{T}_i, \ldots$, and each job requires C_i units of resource within D_i time units. We use these asynchronous tasks to abstract resource demand in our example.

Consider components $\mathcal{C}_1 = \langle\{\tau_1 = (7, 1, 7), \tau_2 = (9, 1, 9)\}, \text{DM}\rangle$, and $\mathcal{C} = \langle\{\mathcal{C}_1, \mathcal{C}_2\}, \text{EDF}\rangle$. Here, we assume that component $\mathcal{C}_2$ interferes with the execution of component $\mathcal{C}_1$ in an adversarial manner (zero slack assumption). In component $\mathcal{C}_1$, jobs of task τ_1 have a higher priority than jobs of task τ_2. Furthermore, because of zero slack assumption, each job of τ_2 will finish its execution only by its deadline. As a result, some jobs of τ_1 are required to finish their executions much before their deadline. For instance, consider the job of τ_2 released at time 18, with deadline at 27. Since schedule of $\mathcal{C}_1$ has zero slack, this job finishes its execution requirements only by time 27 (its latest possible finish time). Then, the job of τ_1 released at time 21 must also finish its execution by time 27 under DM. This scenario is shown in Figure 3.

The amount of resource required by jobs of task τ_1 in the interval $(0, \mathrm{LCM}]$, is given in Figure 4(b). Here, $\mathrm{LCM}(= 63)$ denotes the least common multiple of periods 7 and 9. Also, release constraints on these demands are given in Table 4(a). As per above discussion, these demands and release constraints

Interval	Demand
$(0, 7]$	1
$(7, 9]$	1
$(14, 18]$	1
$(21, 27]$	1
$(28, 35]$	1
$(35, 42]$	1
$(42, 45]$	1
$(49, 54]$	1
$(56, 63]$	1

(a) Demand intervals for τ_1



(b) Demand function of τ_1

Figure 4. Demand of task τ_1 in component $\mathcal{C}_1$

are exact in the sense that they are necessary and sufficient to guarantee schedulability of τ_1. Then, any locally demand optimal interface must reproduce this demand function and release constraints exactly to abstract τ_1. Suppose an asynchronous periodic task $\tau = (\mathrm{O}, \mathrm{T}, \mathrm{C}, \mathrm{D})$ is used to abstract the resource requirements of some jobs of τ_1. Then, O and D must be such that $(\mathrm{O}, \mathrm{O} + \mathrm{D}]$ is one of the entries in Table 4(a). Also, T must be such that, for all k, $\mathrm{O} + k\,\mathrm{T} = l\,\mathrm{LCM} + a$ and $\mathrm{O} + k\,\mathrm{T} + \mathrm{D} = l\,\mathrm{LCM} + b$ for some $l \geq 0$ and entry $(a, b]$ in the table. It is easy to see that these properties do not hold for any $\mathrm{T} < 63$. This means that task τ can be used to abstract the demand of only one job of τ_1 in the interval $(0, 63]$. Therefore, at least $\frac{\mathrm{LCM}}{\mathrm{T}_1} = 9$ tasks are required to abstract the demand of all jobs of τ_1. This shows that the exponential complexity of demand optimal interfaces cannot be avoided for the example under consideration. Since interference from other components can be adversarial in open and closed systems, this example illustrates necessity of increased interface size in both these cases.

4.2. Discussions

Comparison between locally demand optimal and load optimal interfaces. Consider a component $\mathcal{C} = \langle\mathcal{T}, \text{EDF}\rangle$, where $\mathcal{T} = \{\tau_1 = (\mathrm{T}_1, \mathrm{C}_1, \mathrm{D}_1), \ldots, \tau_n = (\mathrm{T}_n, \mathrm{C}_n, \mathrm{D}_n)\}$ is a constrained deadline periodic task set. From Lemma 4 we get that $(1, \mathrm{LOAD}_{\mathcal{C}}, 1)$ is a load optimal interface for $\mathcal{C}$. Also, from discussions in Section 3.2, we get that $\mathcal{I}_{\mathcal{C}} = (\mathrm{GCD}, \mathrm{LOAD}_{\mathcal{C}} \times \mathrm{GCD}, \mathrm{GCD})$ is also a load optimal inter-

face for $\mathcal{C}$, where GCD is the greatest common divisor of $\mathrm{T}_1, \ldots, \mathrm{T}_n, \mathrm{D}_1, \ldots, \mathrm{D}_n$. Consider the case $\mathrm{GCD} = \mathrm{T}_1 = \ldots = \mathrm{T}_n = \mathrm{D}_1 = \ldots = \mathrm{D}_n$. In this case, it is easy to see that $\mathcal{I}_\mathcal{C}$ also satisfies the notion of local demand optimality (Definition 6). This is because $\mathrm{dbf}_\mathcal{C} = \mathrm{dbf}_{\mathcal{I}_\mathcal{C}}$. On the other hand, if $\mathrm{T}_i \neq \mathrm{D}_i$ for some i or $\mathrm{T}_i \neq \mathrm{T}_j$ for some i and j, then we can show there exists t such that $\mathrm{dbf}_\mathcal{C}(t) < \mathrm{LOAD}_\mathcal{C} \times t = \mathrm{dbf}_{\mathcal{I}_\mathcal{C}}$ and $t = k \times \mathrm{GCD}$ for some integer k. For instance, if $\mathrm{T}_i \neq \mathrm{D}_i$ for some i, then at $t = \mathrm{LCM}$ this property holds, where LCM denotes least common multiple of $\mathrm{T}_1, \ldots, \mathrm{T}_n$. Similarly, if $\mathrm{T}_i \neq \mathrm{T}_j$ for some i and j and $\mathrm{D}_i = \mathrm{T}_i$ for all i, then at $t = \min_{i=1,\ldots,n} \mathrm{T}_i$ this property holds. Hence, $\mathcal{I}_\mathcal{C}$ does not satisfy the property of local demand optimality whenever $\mathrm{T}_i \neq \mathrm{D}_i$ for some i or $\mathrm{T}_i \neq \mathrm{T}_j$ for some i and j. Similar arguments also apply to components that use DM scheduler. Thus, load optimality also results in local demand optimality in one extremely restrictive case.

Comparison between globally demand optimal and load optimal interfaces. Consider a component $\mathcal{C}$, with $\mathcal{C}_1, \ldots, \mathcal{C}_m$ denoting all the elementary components in the tree rooted at $\mathcal{C}$. Let $\mathcal{C}_1, \ldots, \mathcal{C}_m$ be the only components in $\mathcal{C}$ with periodic tasks in their workloads, and $\mathcal{S}_1, \ldots, \mathcal{S}_m$ denote their respective schedulers such that each $\mathcal{S}_i = \mathrm{EDF}$. Also, let $\mathcal{I}_\mathcal{C}$ denote a load optimal interface for component $\mathcal{C}$. Then, assuming all interfaces in this system have period one, we get that $\mathrm{LOAD}_{\mathcal{I}_\mathcal{C}} = \sum_{i=1}^{m} \mathrm{LOAD}_{\mathcal{C}_i,\mathcal{S}_i}$ from Theorem 3. Now, suppose there exists a time t such that for each i, $\mathrm{LOAD}_{\mathcal{C}_i,\mathcal{S}_i} \times t = \mathrm{dbf}_{\mathcal{C}_i}(t)$. Note that all the task sets have their maximum load for the same time interval length. Then, in this case, $\mathcal{I}_\mathcal{C}$ is also globally demand optimal, because $\sum_{i=1}^{m} \mathrm{LOAD}_{\mathcal{C}_i,\mathcal{S}_i}$ is indeed the minimum bandwidth required from an uniprocessor platform to schedule $\mathcal{C}$. On the other hand, if such a t does not exist or if some $\mathcal{S}_i$ is DM, then $\sum_{i=1}^{m} \mathrm{LOAD}_{\mathcal{C}_i,\mathcal{S}_i}$ can be strictly larger than the minimum required bandwidth (for instance, see example in Section 2.3). In this case, $\mathcal{I}_\mathcal{C}$ is not globally demand optimal.

5 Conclusions and future work

In this paper we introduced two notions of resource optimality in hierarchical systems. We proposed efficient techniques to generate load optimal interfaces, which characterize optimality with respect to average resource requirements. Each load optimal interface comprises of a single task, and hence has $\mathcal{O}(1)$ storage requirements when compared to the input specification. For demand optimality, we showed using an example that interfaces are hard to generate.

Although, in general, the complexity of a demand optimal interface is unavoidable (as shown by the example in Section 4.1), demand imposed by a set of tasks in the interface may sometimes be represented by a smaller set of tasks, reducing the size of the interface. An interesting area of future work is to characterize the cases when such a reduction is possible.

References

[1] L. Almeida and P. Pedreiras. Scheduling within temporal partitions: response-time analysis and server design. In *EMSOFT*, 2004.

[2] S. Baruah, R. Howell, and L. Rosier. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Journal of Real-Time Systems*, 2:301–324, 1990.

[3] M. Behnam, I. Shin, T. Nolte, and M. Nolin. SIRAP: A synchronization protocol for hierarchical resource sharing in real-time open systems. In *EMSOFT*, pages 279–288, 2007.

[4] R. I. Davis and A. Burns. Hierarchical fixed priority pre-emptive scheduling. In *RTSS*, 2005.

[5] R. I. Davis and A. Burns. Resource sharing in hierarchical fixed priority pre-emptive systems. In *RTSS*, 2006.

[6] Z. Deng and J. W.-S. Liu. Scheduling real-time applications in an open environment. In *RTSS*.

[7] A. Easwaran, M. Anand, and I. Lee. Optimal compositional analysis using explicit deadline periodic resource models. In *RTSS*, 2007.

[8] A. Easwaran, I. Shin, O. Sokolsky, and I. Lee. Incremental schedulability analysis of hierarchical real-time components. In *EMSOFT*, 2006.

[9] X. A. Feng and A. K. Mok. A model of hierarchical real-time virtual resources. In *RTSS*, 2002.

[10] N. Fisher, M. Bertogna, and S. Baruah. The design of an edf-scheduled resource-sharing open environment. In *RTSS*, pages 83–92, 2007.

[11] T. A. Henzinger and S. Matic. An interface algebra for real-time components. In *RTAS*, 2006.

[12] T.-W. Kuo and C.-H. Li. A fixed-priority-driven open environment for real-time applications. In *RTSS*, 1999.

[13] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: exact characterization and average case behavior. In *RTSS*.

[14] G. Lipari and S. Baruah. Efficient scheduling of real-time multi-task applications in dynamic systems. In *RTAS*, 2000.

[15] G. Lipari and E. Bini. Resource partitioning among real-time applications. In *ECRTS*, July 2003.

[16] G. Lipari, J. Carpenter, and S. Baruah. A framework for achieving inter-application isolation in multiprogrammed hard-real-time environments. In *RTSS*, 2000.

[17] S. Matic and T. A. Henzinger. Trading end-to-end latency for composability. In *RTSS*, December 2005.

[18] A. Mok, X. Feng, and D. Chen. Resource partition for real-time systems. In *RTAS*, 2001.

[19] S. Saewong, R. R. Rajkumar, J. P. Lehoczky, and M. H. Klein. Analysis of hierar hical fixed-priority scheduling. In *ECRTS*, 2002.

[20] I. Shin and I. Lee. Periodic resource model for compositional real-time guarantees. In *RTSS*, 2003.

[21] I. Shin and I. Lee. Compositional real-time scheduling framework. In *RTSS*, 2004.

[22] L. Thiele, E. Wandeler, and N. Stoimenov. Real-time interfaces for composing real-time systems. In *EMSOFT*, 2006.

[23] E. Wandeler and L. Thiele. Interface-based design of real-time systems with hierarchical scheduling. In *RTAS*, 2006.