An FPTAS for Interface Selection in the Periodic Resource Model

Nathan Fisher

In Proc. of the 17th International Conference on Real-Time and Network Systems
RTNS'2009, Paris, ECE, 26-27 October, 2009

An FPTAS for Interface Selection in the Periodic Resource Model*

Nathan Fisher
Department of Computer Science
Wayne State University
fishern@cs.wayne.edu

Abstract

The periodic resource model *of Shin and Lee [19] provides a flexible, simple framework for designing compositional real-time systems. Each component in the periodic resource model has an interface which specifies the* period *and* capacity *of the resource used to schedule the component. Unfortunately, the best-known exact algorithms for determining the interface parameters for a component in the periodic resource model potentially require exponential or pseudo-polynomial time. In this paper, we obtain an FPTAS for the problem of selecting an interface period given a capacity-determination algorithm. We also apply our approach to obtain an FPTAS for the problem of selecting both a period and capacity for components consisting of sporadic task systems. Our algorithms obtain interface parameters with interface bandwidth at most* $(1+\epsilon)$ *times the optimal for any* $\epsilon > 0$.

1 Introduction

Compositional analysis for real-time systems has recently received considerable research attention due to its well-known benefits of reducing system-design complexity. For component-based systems, reduction in system complexity is typically achieved via *component abstraction*. Component abstraction hides the complexity and internal details of each component from developers of other components and only exposes information necessary to use the component via an interface. Numerous compositional frameworks have been proposed to support the design and analysis of compositional real-time systems (for a non-exhaustive list see [1, 7, 10, 19]). In each of these compositional frameworks, a component expresses its computational requirements to the system via a *real-time interface*. One important attribute of a real-time interface is the *interface bandwidth*. The interface bandwidth simultaneously quantifies the fraction of the total system resource supply that a component C will require to meet its real-time constraints and the component C's "interference" on the resource supply provided to other system components. Thus, a fundamental goal in the design and analysis of compositional real-time systems is the *minimization of real-time interface bandwidth* (MIB-RT).

The *periodic resource model* [19] is a simple, yet flexible real-time compositional framework. A periodic resource, denoted by $\Gamma = (\Pi, \Theta)$, guarantees that a component C executed upon resource Γ will receive at least Θ units of execution (not necessarily contiguous) between successive time points $\{t \equiv t_0 + \ell\Pi \mid \ell \in \mathbb{N}\}$, given some initial resource start-time t_0. The parameters Π and Θ are respectively referred to as the *period* and *capacity* of the periodic resource Γ. It will be assumed throughout the paper that Π is a positive integer while Θ may be a real number. The ratio $\frac{\Theta}{\Pi}$ represents the interface bandwidth of resource Γ. A system-level scheduling algorithm allocates the processor time among the different periodic resources that share the same processor, such that each resource receives (for every period) aggregate processor time equivalent to its capacity. A subsystem's tasks are then hierarchically scheduled by a subsystem-level scheduling algorithm upon the processing time supplied to resource Γ. If the system-level scheduling algorithm is earliest-deadline-first (EDF), then it is known (e.g., see [19]) that periodic resources $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_m\}$ can successfully guarantee the capacity parameters to their respective components, if and only if, the total system bandwidth does not exceed one; i.e., $\sum_{i=1}^{m} \frac{\Theta_i}{\Pi_i} \leq 1$. From the aforementioned condition, it is clear that minimizing the interface bandwidth of each resource increases system schedulability.

In this paper, we specifically consider the problem of determining the optimal choice of period parameter (i.e., Π) for resource Γ, given a component C and a *capacity-determination algorithm* $\mathcal{A}$. A capacity-determination algorithm returns the minimum-schedulable capacity for a given component C to meet all deadlines upon a periodic resource with a fixed period Π. Such algorithms have already

*This research has been supported by a Wayne State University Faculty Research Award.

been devised for sporadic tasks; e.g., see [9, 12, 19]. Let $\Theta^{\mathcal{A}}_{\min}(\Pi, C)$ be the value returned by capacity-determination algorithm $\mathcal{A}$ for a given C and Π. Furthermore, let us consider only integer values of Π in the range $\{\Pi_{\text{lower}}, \ldots, \Pi_{\text{upper}}\}$ as possible periods. The optimal choice period for the MIB-RT problem is:

$$\Pi^*(\mathcal{A}, C) \stackrel{\text{def}}{=} \arg\min_{\Pi \in \mathbb{N}^+} \left\{ \frac{\Theta^{\mathcal{A}}_{\min}(\Pi, C)}{\Pi} \mid \Pi_{\text{lower}} \leq \Pi \leq \Pi_{\text{upper}} \right\}. \quad (1)$$

Let OPT denote the optimal capacity-determination algorithm. The problem of exactly determining $\Pi^*(\mathsf{OPT}, C)$ for the period resource model (and slightly more general models) has been extensively studied by Easwaran [8]. However, each of the proposed methods has runtime dependent of the difference between Π_{lower} and Π_{upper}. If this difference is large, determining $\Pi^*(\mathcal{A}, C)$ may be prohibitively expensive from a time-complexity perspective, especially if the capacity-determination algorithm $\mathcal{A}$ also has significant computational complexity. Such a large computation cost prohibits exact algorithms from being used for "on-the-fly" computation of interfaces.

§Our Contributions. Our main objective is to find an approximation to $\Pi^*(\mathcal{A}, C)$ with bounded deviation from the optimal solution to MIB-RT. In this paper, we present an approximation scheme with the following guarantee:

> Given a component C, capacity-determination algorithm $\mathcal{A}$, range of possible period values $\{\Pi_{\text{lower}}, \ldots, \Pi_{\text{upper}}\}$, and an accuracy parameter $\epsilon > 0$, if our algorithm returns $\widehat{\Pi}$ for the given parameters, then $\frac{\Theta^{\mathcal{A}}_{\min}(\Pi^*(\mathcal{A},C),C)}{\Pi^*(\mathcal{A},C)} \leq \frac{\Theta^{\mathcal{A}}_{\min}(\widehat{\Pi},C)}{\widehat{\Pi}} \leq (1+\epsilon) \cdot \frac{\Theta^{\mathcal{A}}_{\min}(\Pi^*(\mathcal{A},C),C)}{\Pi^*(\mathcal{A},C)}$. Furthermore, our algorithm runs in time polynomial in the representation of C, Π_{lower}, Π_{upper}, $\frac{1}{\epsilon}$, and the complexity of $\mathcal{A}$.

In other words, our algorithm is a fully-polynomial-time approximation scheme (FPTAS) [20] for the MIB-RT problem (with respect to the computation complexity of $\mathcal{A}$). The $(1+\epsilon)$ factor is called the *approximation ratio* of the produced solution. We also give an FPTAS for determining both the period and capacity of a component consisting of sporadic tasks.

§Organization. The remainder of the paper is organized as follows. Section 2 briefly describes prior related research on the MIB-RT problem. Section 3 presents our proposed approximation scheme for period selection (with respect to any given capacity-determination algorithm) and proves its associated approximation ratio. Section 4 shows how the approximation scheme given in the Section 3 may be used to obtain an overall approximation scheme (with respect to the optimal capacity-determination) for components consisting of sporadic real-time tasks.

2 Related Work

The MIB-RT problem has previously been studied for the periodic resource model and the *explicit-deadline periodic resource* (EDP) model [9] where each component is represented by a *sporadic task system* [16]. Easwaran et al. [9] also obtain exact solutions to MIB-RT in this context (i.e., if the bandwidth provided by the system to component C is less than the exact solution to MIB-RT, then some real-time constraint will be violated for C). This solution is based upon exact schedulability techniques for uniprocessor real-time systems [4, 14], which may be computationally expensive. Easwaran [8] also explored period selection in the EDP model (which can trivially be applied to the periodic resource model); however, the worst-case complexity of the proposed algorithms could be potentially exponential in the number of tasks in the component. Shin and Lee [19] have obtained $O(n)$-time, sufficient solutions to MIB-RT for the periodic resource. The advantage of this approach is that bandwidth allocation may be determined quickly for a component C. However, our previous analysis [11] of Shin and Lee's linear-time algorithm showed that, for a fixed resource period, there exists sporadic task systems that would cause the algorithm to return a bandwidth that is a factor of 1.5 greater than the optimal bandwidth. (It is also shown that the most that the returned bandwidth could exceed optimal is by a factor of 3). As an intermediate solution between computationally-expensive, exact solutions and efficient, inexact solutions, Fisher and Dewan [12] obtain an FPTAS for capacity-determination in the periodic resource model with fixed resource periods. In this paper, we remove the assumption of fixed resource periods.

A number of other results on MIB-RT for different compositional models exist. For components scheduled by fixed-priority on *temporal partitions*, Lipari and Bini [15], developed an exact, pseudo-polynomial-time algorithm for MIB-RT. While Almeida and Pedreiras [3] developed sufficient, polynomial-time bandwidth allocation techniques for fixed-priority scheduling upon temporal partitions. Recently, researchers have also focused on characterizing components by processor-demand curves which describe the minimum amount of processing that a component requires over any interval. For example, Wandeler and Thiele [21] proposed the concept of *interface-based design* which uses real-time

calculus [6] to compute demand curves and service curves for each component in a compositional real-time system. In another demand-based model, Albers et al. [1] have developed parametric algorithms for MIB-RT (without known approximation ratios) for the *hierarchical event stream model*.

3 An Algorithm for Selecting the Interface Period

In this section, we describe an algorithm for selecting a "near-optimal" interface period for a component C, with respect to a given capacity-determination algorithm $\mathcal{A}$. However, to guarantee that the approximation ratio of our proposed algorithm is not too large, we need to formally restrict the types of capacity-determination algorithms that we consider. Informally, we will only consider algorithms where the capacity grows with increasing periods. The formal definition is given in the following.

Definition 1 *For any component C, an algorithm $\mathcal{A}$ is a* **monotonically non-decreasing capacity-determination algorithm** *over $\{\Pi_{\text{lower}}, \ldots, \Pi_{\text{upper}}\}$, if for all $\Pi_1, \Pi_2 \in \{\Pi_{\text{lower}}, \ldots, \Pi_{\text{upper}}\}$ such that $\Pi_1 \leq \Pi_2$, it must be that $\Theta^{\mathcal{A}}_{\min}(\Pi_1, C) \leq \Theta^{\mathcal{A}}_{\min}(\Pi_2, C)$.*

Note that this definition does not place a constraint on the types of problems that can be solved by our period-selection algorithm, as all known capacity-determination algorithms [9, 12, 19] for the periodic resource model possess this property. Section 4 will formally show that the algorithm of [12] is monotonically non-decreasing in Π. In the remainder of this section, we will briefly describe our algorithm (Section 3.1) and prove its correctness (Section 3.2).

3.1 Algorithm Description

The period-selection algorithm, $\mathsf{SelectInterface}(C, \mathcal{A}, \Pi_{\text{lower}}, \Pi_{\text{upper}}, \epsilon)$, is simple; the algorithm evaluates $\Theta^{\mathcal{A}}_{\min}(\Pi, C)$ for select values of Π between Π_{lower} and Π_{upper} and returns the Π and $\Theta^{\mathcal{A}}_{\min}(\Pi, C)$ with the minimum $\frac{\Theta^{\mathcal{A}}_{\min}(\Pi, C)}{\Pi}$. The values of Π are selected based on an accuracy parameter ϵ. Pseudocode for $\mathsf{SelectInterface}(C, \mathcal{A}, \Pi_{\text{lower}}, \Pi_{\text{upper}}, \epsilon)$ is given below.

Lines 1 through 5 initialize the first choice for Π (equal to Π_{lower}), the corresponding minimum capacity (as determined by $\mathcal{A}$), and other bookkeeping variables. The *while* loop of Lines 6 and 21 iterate through successive choices of Π that have capacity ratios (i.e., the ratio between the capacity of Π and the capacity for the previous choice of Π) of at most $(1+\epsilon)$. To find the next choice of Π, we use a *binary search* (Line 7) over the range of remaining values of Π that have not been selected. The binary search returns the largest Π such that the capacity of Π is no more than $(1+\epsilon)$ times the capacity of the previously-selected value of Π. Π is incremented (Line 8) within the while loop to ensure that the next capacity-value for Π is more than $(1+\epsilon)$ times the previous. Finally, the while loop terminates when our returned capacity exceeds the capacity of Π_{upper}. The values $\widehat{\Theta}$ and $\widehat{\Pi}$ maintain the interface parameters of the resource with the minimum capacity of all values of Π that have been evaluated. The algorithm returns the minimum-bandwidth interface over all evaluated choices of Π.

Algorithm 1 $\mathsf{SelectInterface}(C, \mathcal{A}, \Pi_{\text{lower}}, \Pi_{\text{upper}}, \epsilon)$.

Require: Component C, resource-capacity determination algorithm $\mathcal{A}$, positive integers Π_{lower} and Π_{upper}, and positive real number $\epsilon : 0 < \epsilon \leq 1$.

Ensure: $\frac{\Theta^{\mathcal{A}}_{\min}(\Pi^*_{\mathcal{A}}, C)}{\Pi^*_{\mathcal{A}}} \leq \frac{\Theta^{\mathcal{A}}_{\min}(\widehat{\Pi}, C)}{\widehat{\Pi}} \leq (1+\epsilon) \cdot \frac{\Theta^{\mathcal{A}}_{\min}(\Pi^*_{\mathcal{A}}, C)}{\Pi^*_{\mathcal{A}}}$ where $\Pi_{\text{lower}} \leq \Pi^*_{\mathcal{A}} \leq \Pi_{\text{upper}}$.

1: $\Pi_{\text{last}} \Leftarrow \Pi_{\text{lower}}$
2: $\Theta_{\text{last}} \Leftarrow \Theta^{\mathcal{A}}_{\min}(\Pi_{\text{last}}, C)$
3: $\Theta_{\text{upper}} \Leftarrow \Theta^{\mathcal{A}}_{\min}(\Pi_{\text{upper}}, C)$
4: $\widehat{\Pi} \Leftarrow \Pi_{\text{last}}$
5: $\widehat{\Theta} \Leftarrow \Theta_{\text{last}}$
6: **while** $(1+\epsilon)\Theta_{\text{last}} \leq \Theta_{\text{upper}}$ **do**
7: Perform a binary search over Π_{last} to Π_{upper} for *largest* Π s.t. $\left(\Theta \stackrel{\text{def}}{=} \Theta^{\mathcal{A}}_{\min}(\Pi, C)\right) \leq (1+\epsilon)\Theta_{\text{last}}$.
8: $\Pi_{\text{last}} \Leftarrow \Pi + 1$
9: $\Theta_{\text{last}} \Leftarrow \Theta^{\mathcal{A}}_{\min}(\Pi_{\text{last}}, C)$
10: **if** $\frac{\Theta}{\Pi} > 1$ **then**
11: **return** "C not schedulable."
12: **end if**
13: **if** $\frac{\Theta}{\Pi} < \frac{\widehat{\Theta}}{\widehat{\Pi}}$ **then**
14: $\widehat{\Theta} \Leftarrow \Theta$
15: $\widehat{\Pi} \Leftarrow \Pi$
16: **end if**
17: **if** $\frac{\Theta_{\text{last}}}{\Pi_{\text{last}}} < \frac{\widehat{\Theta}}{\widehat{\Pi}}$ **then**
18: $\widehat{\Theta} \Leftarrow \Theta_{\text{last}}$
19: $\widehat{\Pi} \Leftarrow \Pi_{\text{last}}$
20: **end if**
21: **end while**
22: **if** $\frac{\Theta_{\text{upper}}}{\Pi_{\text{upper}}} < \frac{\widehat{\Theta}}{\widehat{\Pi}}$ **then**
23: **return** $\widehat{\Gamma} = (\Pi_{\text{upper}}, \Theta_{\text{upper}})$
24: **else**
25: **return** $\widehat{\Gamma} = \left(\widehat{\Pi}, \widehat{\Theta}\right)$
26: **end if**

3.2 Proof of Correctness

We now must show that $\mathsf{SelectInterface}(C, \mathcal{A}, \Pi_{\text{lower}}, \Pi_{\text{upper}}, \epsilon)$ returns an interface with bandwidth no more than a factor of $(1+\epsilon)$ greater than the optimal

bandwidth. That is, we need to show that $\mathsf{SelectInterface}(C, \mathcal{A}, \Pi_{\text{lower}}, \Pi_{\text{upper}}, \epsilon)$ has an approximation ratio equal to $(1+\epsilon)$. The first result that we give towards this goal is a lower bound on the bandwidth for any contiguous range of periods $\{\Pi_i, \ldots, \Pi_j\}$.

Lemma 1 *Consider any* $\Pi_i, \Pi_j \in \{\Pi_{\text{lower}}, \ldots, \Pi_{\text{upper}}\}$ *where* $\Pi_i \leq \Pi_j$ *Given any monotonically non-decreasing capacity-allocation algorithm* $\mathcal{A}$ *and component* C*, the following is true.*

$$\frac{\Theta^{\mathcal{A}}_{\min}(\Pi_i, C)}{\Pi_j} \leq \min_{\Pi \in \{\Pi_i, \Pi_{i+1}, \ldots, \Pi_j\}} \left\{ \frac{\Theta^{\mathcal{A}}_{\min}(\Pi, C)}{\Pi} \right\}. \tag{2}$$

Proof: Observe that, since $\mathcal{A}$ is monotonically non-decreasing over $\{\Pi_i, \ldots, \Pi_j\}$, $\Theta^{\mathcal{A}}_{\min}(\Pi_i, C) \leq \Theta^{\mathcal{A}}_{\min}(\Pi, C)$ for all $\Pi \in \{\Pi_i, \ldots, \Pi_j\}$. Also, $\frac{1}{\Pi_j} \leq \frac{1}{\Pi}$ is trivially true for all $\Pi \in \{\Pi_i, \ldots, \Pi_j\}$. Equation 2 follows from these two observations. ■

We now show that if we select period values from $\{\Pi_{\text{lower}}, \ldots, \Pi_{\text{upper}}\}$ such that consecutive choices from this domain are either adjacent (i.e., the values are different by one) or have computed capacities different by at most a multiplicative factor of $(1+\epsilon)$, then the minimum bandwidth resulting from these choices is at most a factor of $(1+\epsilon)$ greater than the optimal minimum bandwidth.

Lemma 2 *Consider monotonically non-decreasing capacity-allocation algorithm* $\mathcal{A}$*, component* C*, and real number* $\epsilon > 0$*. Let* $\{\Pi_1, \ldots, \Pi_m\}$ *be any (ordered) subset of*$\{\Pi_{\text{lower}}, \ldots, \Pi_{\text{upper}}\}$ *such that* $\Pi_1 = \Pi_{\text{lower}}$*,* $\Pi_m = \Pi_{\text{upper}}$*, and, for all* $i : 1 \leq i < m$*, either*

$$\Pi_{i+1} = \Pi_i + 1, \tag{3}$$

or

$$\Theta^{\mathcal{A}}_{\min}(\Pi_i, C) \leq \Theta^{\mathcal{A}}_{\min}(\Pi_{i+1}, C) \leq (1+\epsilon) \cdot \Theta^{\mathcal{A}}_{\min}(\Pi_i, C) \tag{4}$$

is true. Then, the following inequalities holds:

$$\begin{aligned} &\frac{\Theta^{\mathcal{A}}_{\min}(\Pi^*(\mathcal{A}, C), C)}{\Pi^*(\mathcal{A}, C)} \\ &\leq \min_{\Pi \in \{\Pi_1, \ldots, \Pi_m\}} \left\{ \frac{\Theta^{\mathcal{A}}_{\min}(\Pi, C)}{\Pi} \right\} \\ &\leq (1+\epsilon) \cdot \frac{\Theta^{\mathcal{A}}_{\min}(\Pi^*(\mathcal{A}, C), C)}{\Pi^*(\mathcal{A}, C)}. \end{aligned} \tag{5}$$

Proof: There are two cases to consider:

1. $\Pi^*(\mathcal{A}, C) \in \{\Pi_1, \ldots, \Pi_m\}$; or
2. $\Pi^*(\mathcal{A}, C) \notin \{\Pi_1, \ldots, \Pi_m\}$.

For Case 1, it obvious that $\min_{\Pi \in \{\Pi_1, \ldots, \Pi_m\}} \left\{ \frac{\Theta^{\mathcal{A}}_{\min}(\Pi, C)}{\Pi} \right\}$ equals $\frac{\Theta^{\mathcal{A}}_{\min}(\Pi^*(\mathcal{A}, C), C)}{\Pi^*(\mathcal{A}, C)}$; Equation 5 follows.

For Case 2, there must exist adjacent values $\Pi_i, \Pi_{i+1} \in \{\Pi_1, \ldots, \Pi_m\}$ such that $\Pi_i \leq \Pi^*(\mathcal{A}, C) \leq \Pi_{i+1}$. Furthermore, $\Pi_{i+1} \neq \Pi_i + 1$; otherwise, $\Pi^*(\mathcal{A}, C)$ would equal either Π_i or Π_{i+1}, and Case 1 would apply. By Lemma 1,

$$\begin{aligned} & \frac{\Theta^{\mathcal{A}}_{\min}(\Pi_i, C)}{\Pi_{i+1}} && \leq \min_{\Pi \in \{\Pi_i, \ldots, \Pi_{i+1}\}} \left\{ \frac{\Theta^{\mathcal{A}}_{\min}(\Pi, C)}{\Pi} \right\} \\ & && = \frac{\Theta^{\mathcal{A}}_{\min}(\Pi^*(\mathcal{A}, C), C)}{\Pi^*(\mathcal{A}, C)} \\ \Rightarrow \quad & \frac{(1+\epsilon)\Theta^{\mathcal{A}}_{\min}(\Pi_i, C)}{\Pi_{i+1}} && \leq \frac{(1+\epsilon)\Theta^{\mathcal{A}}_{\min}(\Pi^*(\mathcal{A}, C), C)}{\Pi^*(\mathcal{A}, C)} \\ \Rightarrow \quad & \frac{\Theta^{\mathcal{A}}_{\min}(\Pi_{i+1}, C)}{\Pi_{i+1}} && \leq \frac{(1+\epsilon)\Theta^{\mathcal{A}}_{\min}(\Pi^*(\mathcal{A}, C), C)}{\Pi^*(\mathcal{A}, C)} \\ & \text{(by Equation 4).} \end{aligned} \tag{6}$$

The final inequality implies

$$\begin{aligned} &\min_{\Pi \in \{\Pi_1, \ldots, \Pi_m\}} \left\{ \frac{\Theta^{\mathcal{A}}_{\min}(\Pi, C)}{\Pi} \right\} \\ &\leq \frac{(1+\epsilon)\Theta^{\mathcal{A}}_{\min}(\Pi^*(\mathcal{A}, C), C)}{\Pi^*(\mathcal{A}, C)}. \end{aligned}$$

Obviously,

$$\begin{aligned} &\frac{\Theta^{\mathcal{A}}_{\min}(\Pi^*(\mathcal{A}, C), C)}{\Pi^*(\mathcal{A}, C)} \\ &\leq \min_{\Pi \in \{\Pi_1, \ldots, \Pi_m\}} \left\{ \frac{\Theta^{\mathcal{A}}_{\min}(\Pi, C)}{\Pi} \right\}, \end{aligned}$$

by definition of $\Pi^*(\mathcal{A}, C)$. From the preceding two inequalities, Equation 5 of the lemma follows. ■

Finally, we use Lemma 1 and 2 to show that $\mathsf{SelectInterface}(C, \mathcal{A}, \Pi_{\text{lower}}, \Pi_{\text{upper}}, \epsilon)$ is an approximation scheme with approximation ratio $(1+\epsilon)$. Furthermore, we quantify the running time of the algorithm in the following theorem.

Theorem 1 $\mathsf{SelectInterface}(C, \mathcal{A}, \Pi_{\text{lower}}, \Pi_{\text{upper}}, \epsilon)$ *has an approximation ratio of* $(1+\epsilon)$ *(where* $0 < \epsilon \leq 1$*) for period selection with respect to any monotonically, non-decreasing capacity-allocation algorithm* $\mathcal{A}$*. Furthermore, the algorithm has time complexity equal to*

$$O\left(\chi^{\mathcal{A}}(C) \cdot \lg\left(\frac{\Theta_{\text{upper}}}{\Theta_{\text{lower}}}\right) \cdot \lg\left(\Pi_{\text{upper}}\right) \Big/ \epsilon\right), \tag{7}$$

where $\Theta_{\text{lower}} \stackrel{def}{=} \Theta^{\mathcal{A}}_{\min}(\Pi_{\text{lower}}, C)$*,* $\Theta_{\text{upper}} \stackrel{def}{=} \Theta^{\mathcal{A}}_{\min}(\Pi_{\text{upper}}, C)$*, and* $\chi^{\mathcal{A}}(C)$ *equals the time complexity of the capacity-determination algorithm* $\mathcal{A}$ *given a component* C*.*

Proof Sketch: Let $\{\Pi_1, \ldots, \Pi_m\}$ be the ordered set of values that variables Π_{last} and Π (set in Line 7) are assigned throughout the execution of $\mathsf{SelectInterface}(C, \mathcal{A}, \Pi_{\text{lower}}, \Pi_{\text{upper}}, \epsilon)$. Obviously, $\{\Pi_1, \ldots, \Pi_m\} \subseteq \{\Pi_{\text{lower}}, \ldots, \Pi_{\text{upper}}\}$ such that $\Pi_1 = \Pi_{\text{lower}}$ and $\Pi_m = \Pi_{\text{upper}}$. Furthermore, it is easy to verify that subsequent values of $\Pi_i, \Pi_{i+1} \in \{\Pi_1, \ldots, \Pi_m\}$ satisfy either Equation 3 (see Line 8) or Equation 4 (see Line 7) of

Lemma 2. Thus, by Equation 5 of Lemma 2, the $\widehat{\Gamma} = \left(\widehat{\Pi}, \widehat{\Theta}\right)$ computed by Lines 4, 5, 13, 17, and 22 is at most $(1+\epsilon) \cdot \frac{\Theta^{\mathcal{A}}_{\min}(\Pi^*(\mathcal{A},C),C)}{\Pi^*(\mathcal{A},C)}$. This shows that $\mathsf{SelectInterface}(C, \mathcal{A}, \Pi_{\text{lower}}, \Pi_{\text{upper}}, \epsilon)$ has an approximation ratio of at most $(1+\epsilon)$.

For determining the complexity of $\mathsf{SelectInterface}(C, \mathcal{A}, \Pi_{\text{lower}}, \Pi_{\text{upper}}, \epsilon)$, observe that the running time is dominated by the *while* loop in Lines 6 through 21. The complexity of the *while* loop can (informally) be determined by

$$\begin{array}{ll} & \text{(Number of iterations of } \textit{while} \text{ loop)} \\ \times & \text{(Number of } \Pi \text{ values to be checked in } \textit{binary search}\text{)} \\ \times & \text{(Execution time to check value of } \Pi\text{)}\,. \end{array} \tag{8}$$

The number of iterations of the *while* loop can be determined by observing that Θ_{last} increases by at least a factor of $(1+\epsilon)$ upon every iteration of the *while* loop of Lines 6 to 21. Thus, the number of iterations is equal to smallest integer value of ℓ such that the following equation is true:

$$(1+\epsilon)^{\ell}\,\Theta_{\text{lower}} \geq \Theta_{\text{upper}}$$

Solving for ℓ,

$$\begin{aligned} \ell &= \left\lceil \log_{(1+\epsilon)} \left(\frac{\Theta_{\text{upper}}}{\Theta_{\text{lower}}} \right) \right\rceil \\ &= \left\lceil \frac{\ln\left(\frac{\Theta_{\text{upper}}}{\Theta_{\text{lower}}} \right)}{\ln(1+\epsilon)} \right\rceil \\ &\leq \left\lceil \frac{(1+\epsilon)\cdot\ln\left(\frac{\Theta_{\text{upper}}}{\Theta_{\text{lower}}} \right)}{\epsilon} \right\rceil \\ &= O\left(\left(\lg \frac{\Theta_{\text{upper}}}{\Theta_{\text{lower}}} \right) / \epsilon \right). \end{aligned} \tag{9}$$

The third equality above follows from the well-known identity $\frac{x}{1+x} \leq \ln(1+x)$ for all $x > -1$.

The *binary search* of Line 7 searches over the range $\{\Pi_{\text{last}}, \ldots, \Pi_{\text{upper}}\}$. This range contains at most Π_{upper} number of values. Thus, the number of values of $\Pi \in \{\Pi_{\text{last}}, \ldots, \Pi_{\text{upper}}\}$ that have to calculate $\Theta^{\mathcal{A}}_{\min}(\Pi, C)$ is

$$O\left(\lg(\Pi_{\text{upper}})\right). \tag{10}$$

Finally, the execution time for each calculation of $\Theta^{\mathcal{A}}_{\min}(\Pi, C)$ is equal to the execution cost of algorithm $\mathcal{A}$ for a given component, which is denoted $\chi^{\mathcal{A}}(C)$. Combining this observation with Equations 8, 9, and 10, implies the running time given in the lemma. ■

4 Interface Selection for Sporadic Task Systems

In this section, we explore an application of the algorithm proposed in the previous section to determine the minimum-bandwidth interface for a component consisting of sporadic tasks scheduled by EDF. We will show that we may, in fact, obtain an FPTAS for the MIB-RT problem in the context of sporadic tasks. The remainder of the section is organized as follows. In Section 4.1, we introduce notations and prior analytic results for the task, workload, and periodic resource models. In Section 4.2, we state the capacity-determination algorithm we use in deriving our FPTAS; the capacity-determination algorithm was previously proposed in [12]. In Section 4.3, we give a simple example to show that using Π_{lower} is not always the optimal choice for minimizing the interface bandwidth. In Section 4.4, we give a description of our FPTAS and prove its correctness.

4.1 Models and Notation

In this subsection, we present background and notation for the task model, workload functions, and periodic resource model that we use throughout the remainder of this section.

§Sporadic Task Model. A **sporadic task** $\tau_i = (e_i, d_i, p_i)$ is characterized by a *worst-case execution requirement* e_i, a *(relative) deadline* d_i, and a *minimum inter-arrival separation* p_i, which is, for historical reasons, also referred to as the *period* of the task. Such a sporadic task generates a potentially infinite sequence of jobs, with successive job-arrivals separated by at least p_i time units. Each job has a worst-case execution requirement equal to e_i and a deadline that occurs d_i time units after its arrival time. We will assume that task parameters are positive integers. Furthermore, obviously, $e_i \leq d_i$ and $e_i \leq p_i$ for any task τ_i; otherwise, cannot be scheduled to meet its deadline by any scheduling algorithm. A *sporadic task system* $\tau \stackrel{\text{def}}{=} \{\tau_1, \ldots, \tau_n\}$ is a collection of n such sporadic tasks. A useful metric for a sporadic task τ_i is the *task utilization* $u_i \stackrel{\text{def}}{=} e_i/p_i$. The system utilization is denoted $U(\tau) \stackrel{\text{def}}{=} \sum_{\tau_i \in \tau} u_i$.

The following lemma on system utilization will be useful in defining an FPTAS.

Lemma 3 *For any sporadic task system τ with positive integer parameters,*

$$U(\tau) \geq \frac{n}{p_{\max}} \tag{11}$$

where $p_{\max} \stackrel{def}{=} \max_{i=1}^{n}\{p_i\}$.

Proof: By definition, $U(\tau)$ equals $\sum_{\tau_i \in \tau} \frac{e_i}{p_i}$. Equation 11 follows from observing that $e_i \geq 1$ and $p_i \leq p_{\max}$ for all $\tau_i \in \tau$. ■

§Workload Functions. For determining schedulability of a sporadic task system, it is often useful to quantify the maximum amount of execution that must complete over any given interval. For this purpose, researchers [5] have derived the *demand-bound function*, defined below.

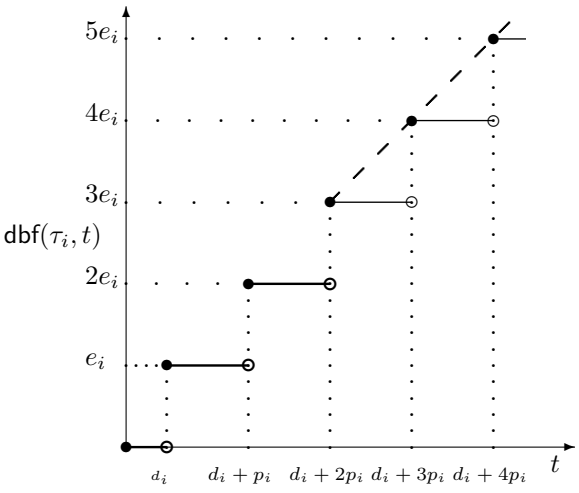

Figure 1. The step function denotes a plot of $\mathsf{dbf}(\tau_i, t)$ as a function of t. The dashed line represents the function $\widetilde{\mathsf{dbf}}(\tau_i, t, k)$, approximating $\mathsf{dbf}(\tau_i, t)$. $\widetilde{\mathsf{dbf}}(\tau_i, t, k)$ is equal to $\mathsf{dbf}(\tau_i, t)$ for all $t < d_i + (k-1)p_i$ (k equals three in the above graph).

Definition 2 (Demand-Bound Function) *For any $t > 0$ and task τ_i, the* **demand-bound function** *(dbf) quantifies the maximum cumulative execution requirement of all jobs of τ_i that could have both an arrival time and deadline in any interval of length t. Baruah et al. [5] have shown that, for sporadic tasks, dbf can be calculated as follows.*

$$\mathsf{dbf}(\tau_i, t) = \max\left(0, \left\lfloor \frac{t - d_i}{p_i} \right\rfloor + 1\right) \cdot e_i. \qquad (12)$$

Figure 1 gives a visual depiction of the demand-bound function for a sporadic task τ_i. Observe from the above definition and Figure 1 that the dbf is a right continuous function with discontinuities at time points of the form $t \equiv d_i + a \cdot p_i$ where $a \in \mathbb{N}$. Let $\mathsf{DBF}(\tau, t) \stackrel{\text{def}}{=} \sum_{\tau_i \in \tau} \mathsf{dbf}(\tau_i, t)$. It has been shown [5] that condition $\mathsf{DBF}(\tau, t) \leq t, \forall t \geq 0$ is necessary and sufficient for sporadic task system τ to be EDF-schedulable upon a preemptive uniprocessor platform of unit speed. Furthermore, it has also been shown that the aforementioned condition needs to be verified at only time points in the following *ordered* set (with elements are in non-decreasing order):

$$\mathsf{TS}(\tau) \stackrel{\text{def}}{=} \bigcup_{\tau_i \in \tau} \{t \equiv d_i + a \cdot p_i \mid (a \in \mathbb{N}) \wedge (t \leq P(\tau))\}. \qquad (13)$$

where $P(\tau)$ is an upper bound on the maximum time instant that the schedulability condition must be verified at. For EDF-scheduled sporadic task systems on preemptive unit-speed processors, $P(\tau)$ is at most $\mathsf{lcm}_{\tau_i \in \tau}\{p_i\}$. The above set is known as the **testing set** for sporadic task system τ. For any $t_a \in \mathsf{TS}(\tau)$, $t_a \leq t_{a+1}$; if t_a is the last element of the set, we use the convention that t_{a+1} equals ∞. Also, we will assume that t_0 is equal to zero.

Albers and Slomka [2] proposed the following approximation to dbf to reduce the number of discontinuities (and, thus, points in the testing set).

$$\widetilde{\mathsf{dbf}}(\tau_i, t, k) \stackrel{\text{def}}{=} \begin{cases} \mathsf{dbf}(\tau_i, t), & \text{if } t < d_i + (k-1)p_i; \\ u_i \cdot (t - d_i) + e_i, & \text{otherwise.} \end{cases} \qquad (14)$$

The main intuition behind $\widetilde{\mathsf{dbf}}(\tau_i, t, k)$ is that it "tracks" dbf for exactly k discontinuities (i.e., "steps"). After k discontinuities, $\widetilde{\mathsf{dbf}}(\tau_i, t, k)$ using a linear interpolation of the subsequent discontinuous points (with slope equal to u_i). The steps with the thick lines and the sloped-dotted line in Figure 1 correspond to $\mathsf{dbf}(\tau_i, t, 3)$. We will abuse notation slightly and use the convention that $\mathsf{dbf}(\tau_i, t, \infty)$ corresponds to $\mathsf{dbf}(\tau_i, t)$. Let $\widetilde{\mathsf{DBF}}(\tau, t, k) \stackrel{\text{def}}{=} \sum_{\tau_i \in \tau} \widetilde{\mathsf{dbf}}(\tau_i, t, k)$. Albers and Slomka show [2], for any fixed $k \in \mathbb{N}^+$, the condition $\widetilde{\mathsf{DBF}}(\tau, t, k) \leq t, \forall t \geq 0$ is sufficient for sporadic task τ to be EDF-schedulable upon a preemptive uniprocessor platform of unit speed. The ordered testing set of this condition is reduced to

$$\widetilde{\mathsf{TS}}(\tau, k) \stackrel{\text{def}}{=} \bigcup_{\tau_i \in \tau} \{t \equiv d_i + a \cdot p_i \mid (a \in \mathbb{N}) \wedge (a < k) \wedge (t \leq P(\tau))\}. \qquad (15)$$

§Periodic Resource Model. Throughout this section, we assume that each component C is a sporadic task system. Let C be composed of a sporadic task system τ that is to be EDF-scheduled upon periodic resource $\Gamma = (\Pi, \Theta)$. (From now on, we use τ in the context of component C). We will now present some concepts that have been previously-introduced by researchers to determine whether τ will meet all deadlines when scheduled upon Γ.

Definition 3 (Supply-Bound Function) *For any $t > 0$, the* **supply-bound function** *(sbf) quantifies the minimum execution supply that a component executed upon periodic resource Γ may receive over any interval of length t. Shin and Lee [19] have quantified the supply bound function for an periodic resource in the following (using the notation of Easwaran et al. [9]):*

$$\mathsf{sbf}(\Gamma, t) = \begin{cases} y_\Gamma \Theta + \max(0, t - x_\Gamma - y_\Gamma \Pi), & \text{if } t \geq \Pi - \Theta \\ 0, & \text{otherwise.} \end{cases} \qquad (16)$$

where $y_\Gamma = \left\lfloor \frac{t - (\Pi - \Theta)}{\Pi} \right\rfloor$ *and* $x_\Gamma = (2\Pi - 2\Theta)$.

EDF-schedulability conditions for periodic resource Γ have been developed [9, 17, 18], as given in the following theorem.

Theorem 2 (from [9]) *A sporadic task system τ is* EDF*-schedulable upon an EDP resource $\Gamma = (\Pi, \Theta)$, if and only if,*

$$(\mathsf{DBF}(\tau, t) \leq \mathsf{sbf}(\Gamma, t),\ \forall t \leq P(\tau)) \bigwedge \left(U(\tau) \leq \frac{\Theta}{\Pi}\right) \quad (17)$$

where $P(\tau)$ equals $\mathsf{lcm}_{\tau_i \in \tau}\{p_i\} + \max_{\tau_i \in \tau}\{d_i\}$.

4.2 Capacity-Determination Algorithm

The algorithm that we consider for determining the minimum-capacity of a periodic resource (for a fixed period) is given by Algorithm 2. Informally, the algorithm works as follows: for each value t in the testing set $\widetilde{\mathsf{TS}}(\tau, k)$, MinimumCapacity determines the minimum capacity $\Theta_t^{\min}$ such that the $\widetilde{\mathsf{DBF}}(\tau, t)$ is "below" the supply-bound function $\mathsf{sbf}((\Pi, \Theta_t^{\min}), t)$. The algorithm is exact when k equals ∞, since $\widetilde{\mathsf{TS}}(\tau, \infty)$ equals $\mathsf{TS}(\tau)$. More details on the algorithm and a proof of correctness is available from [12]. Please note that the algorithm found in [12] is given for the more general explicit-deadline periodic resource (EDP) model [9]; the algorithm stated here has been adapted to for the periodic resource model.

Algorithm 2 MinimumCapacity(Π, τ, k).

Require: Sporadic task system τ, resource period Π (positive integer), and positive integer k.

1: $\Theta^{\min} \leftarrow U(\tau) \cdot \Pi$
2: **for all** $t \in \widetilde{\mathsf{TS}}(\tau, k)$ **do**
3: $\quad D_t \leftarrow \widetilde{\mathsf{DBF}}(\tau, t, k)$
4: $\quad \alpha_t \leftarrow \sum_{\tau_i \in \tau : t \geq d_i + (k-1)p_i} u_i$
5: $\quad \Theta_t^{\min} \leftarrow \infty$
6: $\quad$ **for** $\ell = \max\left\{1, \left\lfloor \frac{t}{\Pi} \right\rfloor - 1\right\}$ to $\left\lceil \frac{t}{\Pi} \right\rceil$ **do**
7: $\quad\quad \Theta_\ell^{\min} \leftarrow \max \left\{ \begin{array}{r} \alpha_t \Pi, \\ \frac{D_t - t + (\ell+1)\Pi}{\ell+1}, \\ \frac{D_t}{\ell}, \\ \frac{D_t + \alpha_t((\ell+2)\Pi - t)}{\ell + 2\alpha_t} \end{array} \right\}$
8: $\quad\quad \Theta_t^{\min} \leftarrow \min\{\Theta_t^{\min}, \Theta_\ell^{\min}\}$
9: $\quad$ **end for**
10: $\quad \Theta^{\min} \leftarrow \max\{\Theta^{\min}, \Theta_t^{\min}\}$
11: **end for**
12: **return** $\Theta^{\min}$

The following result (restated from [12]) shows that MinimumCapacity is an FPTAS for capacity-determination given a fixed period.

Theorem 3 (from [12]) *Given Π, τ, and $\epsilon > 0$, the procedure* MinimumCapacity $\left(\Pi, \tau, \left\lceil \frac{1}{\epsilon} \right\rceil\right)$ *returns $\Theta^{\min}$ such that*

$$\Theta^*(\Pi, \tau) \leq \Theta^{\min} \leq (1 + \epsilon) \cdot \Theta^*(\Pi, \tau)$$

where $\Theta^*(\Pi, \tau)$ is the optimal minimum capacity required to EDF*-schedule τ upon a periodic resource with period of Π. Furthermore,* MinimumCapacity $\left(\Pi, \tau, \left\lceil \frac{1}{\epsilon} \right\rceil\right)$ *has time complexity* $O\left(\frac{n \lg n}{\epsilon}\right)$.[1]

4.3 A Motivating Example

At this point in the paper, the reader may wonder, if a monotonically non-decreasing capacity-determination algorithm is used, does the minimum bandwidth occur for smallest possible value of Π (i.e., Π_{lower}) – eliminating the need for more complex period-selection algorithm? In this subsection, we show that such a period-selection algorithm is definitely required. We give a small motivating example to illustrate that the minimum bandwidth is not always achieved using the period Π_{lower}.

Consider a component consisting of a single sporadic task $\tau_1 \stackrel{\text{def}}{=} (e_1, d_1, p_1) = (1, 301, 1000)$. If the range of possible Π values is $\{80, 81, \ldots, 150\}$ and we use an exact capacity determination algorithm (i.e., MinimumCapacity with $k = \infty$), then we will find that the minimum capacity to successfully schedule τ_1 is equal to 0.5 for all $\Pi \in \{80, 81, \ldots, 100\}$ and 1.0 for all $\Pi \in \{101, 102, \ldots, 150\}$. Thus, the minimum bandwidth interface for τ_1 is $\Gamma = (100, 0.5)$ which has a bandwidth of 0.005. However, the approach of checking all values of Π would have required us to call MinimumCapacity a total of 71 times. Furthermore, observe that the minimum bandwidth interface did not have a period of Π_{lower}, but in fact occurred 20 units greater than Π. We expect that this example can be generalized so that the difference from Π_{lower} to the optimal period is arbitrarily large; thus, checking all values of Π for the minimum bandwidth interface is potentially very expensive. On the other hand, SelectInterface$(\{\tau_1\}$, MinimumCapacity$, 80, 150, .1)$ would call MinimumCapacity a total of nine times and, in fact, return the interface $\Gamma = (100, 0.5)$ for this example. Thus, for this particular example, a significant reduction in time complexity can be obtained with no loss of accuracy. The next subsection shows how we may obtain an FPTAS that guarantees the desired level of accuracy for the selected interface of any possible component.

4.4 An FPTAS for Interface Selection

In this subsection, we present the main result of the section: an FPTAS for interface selection for periodic resources scheduling sporadic tasks. Before we state our main result, we require two technical lemmas. The first lemma gives an upper and lower bound on the value returned by MinimumCapacity.

[1] Note that there is an error in the statement of this theorem in [12]. It incorrectly stated that the value $\max\left(1, \left\lfloor \frac{1}{\epsilon} \right\rfloor\right)$ should be used in the third argument to MinimumCapacity. However, the technical report [13] has corrected the value of the argument to $\left\lceil \frac{1}{\epsilon} \right\rceil$.

Lemma 4 *For a given* Π*, sporadic task system* τ*, and* $k \in \mathbb{N}^+ \cup \{\infty\}$*, if* $t \geq \widetilde{\mathsf{DBF}}(\tau, t, k)$ *for all* $t \in \widetilde{\mathsf{TS}}(\tau, k)$ *and* $U(\tau) \leq 1$*, then the capacity* $\Theta^{\min}$ *returned from* MinimumCapacity (Π, τ, k) *satisfies*

$$\Pi \cdot \frac{n}{p_{\max}} \leq \Theta^{\min} \leq \Pi \tag{18}$$

Proof: Line 1 of MinimumCapacity sets $\Theta^{\min}$ to the minimum default value of $U(\tau) \cdot \Pi$. Thus, $\Theta^{\min} \geq U(\tau) \cdot \Pi$. By Lemma 3, $\Theta^{\min} \geq \Pi \cdot \frac{n}{p_{\max}}$.

To see the upper bound on $\Theta^{\min}$, consider Line 7 when ℓ equals $\lceil \frac{t}{\Pi} \rceil$. The value $\alpha_t \cdot \Pi \leq U(\tau) \cdot \Pi$, by Line 4; $\alpha_t \cdot \Pi \leq \Pi$, since $U(\tau) \leq 1$ by supposition. The value $\frac{D_t - t + (\ell+1)\Pi}{\ell+1}$ is at most Π since $D_t \leq t$. The value $\frac{D_t}{\ell}$ is also at most $\frac{D_t \cdot \Pi}{t}$ since $\ell = \lceil \frac{t}{\Pi} \rceil \geq \frac{t}{\Pi}$; since $t \geq D_t$, $\frac{D_t}{\ell} \leq \Pi$. Finally, the value $\frac{D_t + \alpha_t((\ell+2)\Pi - t)}{\ell + 2\alpha_t}$ is increasing in α_t when ℓ equals $\lceil \frac{t}{\Pi} \rceil$; thus, $\frac{D_t + \alpha_t((\ell+2)\Pi - t)}{\ell + 2\alpha_t} \leq \frac{D_t + (\ell+2)\Pi - t}{\ell+2}$ which is at most Π since $t \geq D_t$. We have, thus, shown that $\Theta^{\min}_{\ell} \leq \Pi$ when ℓ equals $\lceil \frac{t}{\Pi} \rceil$. Line 8 sets $\Theta^{\min}$ to the minimum value of $\Theta^{\min}_{\ell}$ for any $t \in \widetilde{\mathsf{TS}}(\tau, k)$. The inequality $\Theta^{\min} \leq \Pi$ follows. ■

The second technical lemma states that MinimumCapacity is monotonically, non-decreasing.

Lemma 5 *For a given* τ *and* $\epsilon > 0$*,* MinimumCapacity *is monotonically, non-decreasing over* $\{\Pi_{\text{lower}}, \ldots, \Pi_{\text{upper}}\}$.

Proof Sketch: The lemma follows from observing that the values of $\Theta^{\min}$ and $\Theta^{\min}_{\ell}$ (set in Lines 1 and 7, respectively) are monotonically, non-decreasing in Π. ■

Finally, we give the FPTAS for the MIB-RT with respect to sporadic tasks. The theorem below uses both SelectInterface and MinimumCapacity to obtain the FPTAS.

Theorem 4 *Given sporadic task system* τ *and accuracy parameter* $\epsilon : 0 < \epsilon \leq 1$*, if* $t \geq \widetilde{\mathsf{DBF}}(\tau, t, k)$ *for all* $t \in \widetilde{\mathsf{TS}}(\tau, k)$ *and* $U(\tau) \leq 1$*, then the procedure* SelectInterface$(\tau, \mathcal{A}, \Pi_{\text{lower}}, \Pi_{\text{upper}}, \frac{\epsilon}{3})$ *where* $\mathcal{A}$ *equals* MinimumCapacity $\left(\cdot, \tau, \lceil \frac{3}{\epsilon} \rceil\right)$ *returns* $\widehat{\Gamma} = \left(\widehat{\Pi}, \Theta^{\mathcal{A}}_{\min}(\widehat{\Pi}, \tau)\right)$ *such that*

$$\begin{aligned} &\frac{\Theta^{\mathsf{OPT}}_{\min}(\Pi^*(\mathsf{OPT}, \tau), \tau)}{\Pi^*(\mathsf{OPT}, \tau)} \\ &\leq \frac{\Theta^{\mathcal{A}}_{\min}(\widehat{\Pi}, \tau)}{\widehat{\Pi}} \\ &\leq (1+\epsilon) \cdot \frac{\Theta^{\mathsf{OPT}}_{\min}(\Pi^*(\mathsf{OPT}, \tau), \tau)}{\Pi^*(\mathsf{OPT}, \tau)}. \end{aligned} \tag{19}$$

Furthermore, the above algorithm has time complexity that is polynomial in the number of tasks n*,* $1/\epsilon$*, and the number of bits required to represent* Π_{upper} *and task system* τ*'s parameters.*

Proof Sketch: Let $\{\Pi_1, \ldots, \Pi_m\}$ be the set of values that Π and Π_{last} are set to throughout execution of SelectInterface$(\tau, \mathcal{A}, \Pi_{\text{lower}}, \Pi_{\text{upper}}, \frac{\epsilon}{3})$. Consider adjacent values $\Pi_i, \Pi_{i+1} \in \{\Pi_1, \ldots, \Pi_m\}$ such that $\Pi_i \leq \Pi^*(\mathsf{OPT}, \tau) \leq \Pi_{i+1}$. Let $\widehat{\Theta}_i$ and $\widehat{\Theta}_{i+1}$ equal the values determined by MinimumCapacity $\left(\cdot, \tau, \lceil \frac{3}{\epsilon} \rceil\right)$ evaluated at Π_i and Π_{i+1}, respectively. If $\Pi_{i+1} = \Pi_i + 1$, then $\Pi^*(\mathsf{OPT}, \tau)$ is equal to either Π_i or Π_{i+1}; w.l.o.g., assume that $\Pi^*(\mathsf{OPT}, \tau)$ equals Π_i. By MinimumCapacity's approximation ratio, $\widehat{\Theta}_i \leq \left(1 + \frac{\epsilon}{3}\right) \Theta^{\mathsf{OPT}}_{\min}(\Pi^*(\mathsf{OPT}, \tau), \tau) \leq (1+\epsilon)\, \Theta^{\mathsf{OPT}}_{\min}(\Pi^*(\mathsf{OPT}, \tau), \tau)$. Equation 19 follows in this case.

In the case that $\Pi_{i+1} \neq \Pi_i + 1$, we know that MinimumCapacity is monotonically, non-decreasing in Π (by Lemma 5). Thus, the binary search of SelectInterface$(\tau, \mathcal{A}, \Pi_{\text{lower}}, \Pi_{\text{upper}}, \frac{\epsilon}{3})$ ensures that $\widehat{\Theta}_{i+1} \leq \left(1 + \frac{\epsilon}{3}\right) \widehat{\Theta}_i$. The approximation ratio of MinimumCapacity also guarantees that $\widehat{\Theta}_i \leq \left(1 + \frac{\epsilon}{3}\right) \Theta^{\mathsf{OPT}}_{\min}(\Pi^*(\mathsf{OPT}, \tau), \tau)$. Thus,

$$\begin{aligned} &\widehat{\Theta}_{i+1} \\ &\quad \leq \left(1 + \tfrac{\epsilon}{3}\right)^2 \Theta^{\mathsf{OPT}}_{\min}(\Pi^*(\mathsf{OPT}, \tau), \tau) \\ &\quad \leq (1+\epsilon) \Theta^{\mathsf{OPT}}_{\min}(\Pi^*(\mathsf{OPT}, \tau), \tau), \end{aligned}$$

since $\epsilon \leq 1$. The above inequality and $\Pi_{i+1} \geq \Pi^*(\mathsf{OPT}, \tau)$ implies that

$$\frac{\widehat{\Theta}_{i+1}}{\Pi_{i+1}} \leq (1+\epsilon) \frac{\Theta^{\mathsf{OPT}}_{\min}(\Pi^*(\mathsf{OPT}, \tau), \tau)}{\Pi^*(\mathsf{OPT}, \tau)}$$

Equation 19 follows from the fact that $\frac{\widehat{\Theta}_{i+1}}{\Pi_{i+1}} \geq \frac{\Theta^{\mathcal{A}}_{\min}(\widehat{\Pi}, \tau)}{\widehat{\Pi}}$.

The time complexity of the approach follows from Equation 7 of Theorem 1. By Theorem 3, $\chi^{\mathcal{A}}(\tau)$ is $O(n \lg n/\epsilon)$. The second term of Equation 7 $\left(\lg\left(\frac{\Theta_{\text{upper}}}{\Theta_{\text{lower}}}\right)\right)$ is upper bounded by $\left(\lg\left(\frac{\Pi_{\text{upper}}}{n/p_{\max}}\right)\right)$ according to Lemma 4; this term is $O\left(\lg(\Pi_{\text{upper}}) + \lg(p_{\max})\right)$. Thus, the entire time complexity of the approach of the Theorem is

$$O\left(n \lg n (\lg^2 \Pi_{\text{upper}} + \lg \Pi_{\text{upper}} \lg p_{\max})/\epsilon^2\right)$$

which is polynomial in the number of tasks, number of bits to represent both the task parameters and Π_{upper}, and $1/\epsilon$. ■

5 Conclusions

In this paper, we propose an approximation algorithm for the minimization of interface bandwidth (MIB-RT) problem in a real-time compositional framework, the periodic resource model. We first propose a general algorithm for determining the interface parameter, given a capacity determination algorithm. This approach is general and can apply to

any component task model. Next, we explore interface selection for components consisting of entirely sporadic tasks, and propose an algorithm based on a previous capacity-determination algorithm [12]. Our algorithm returns bandwidth that is at most a factor of $(1 + \epsilon)$ greater than the optimal minimum bandwidth, for any $\epsilon > 0$. Furthermore, it is shown that our algorithm is an FPTAS as it has time complexity that is polynomial in the number of tasks in the sporadic task system, the number of bits to represent the task parameters, the number of bits to represent the maximum task period Π_{upper}, and the term $1/\epsilon$. Previous work [8] has shown that exact algorithms for MIB-RT problem on periodic resources may require pseudo-polynomial or exponential time. Thus, our results may provide a significant reduction in the time necessary to determine the minimum-bandwidth interface parameters.

In future work, we hope to explore interface selection in the presence of shared global resources; we also would like to study the effect of overheads in the choice of interface parameters. We believe that our the central idea of our FPTAS proposed in this paper is general enough to extend to these more practical and complex settings.

References

[1] K. Albers, F. Bodmann, and F. Slomka. Advanced hierarchical event-stream model. In *Proceedings of the EuroMicro Conference on Real-Time Systems*, pages 211–220, Prague, Czech Republic, July 2008. IEEE Computer Society.

[2] K. Albers and F. Slomka. An event stream driven approximation for the analysis of real-time systems. In *Proceedings of the EuroMicro Conference on Real-Time Systems*, pages 187–195, Catania, Sicily, July 2004. IEEE Computer Society Press.

[3] L. Almeida and P. Pedreiras. Scheduling within temporal partitions: response-time analysis and server design. In *EMSOFT '04: Proceedings of the 4th ACM international conference on Embedded software*, pages 95–103, New York, NY, USA, 2004. ACM.

[4] S. Baruah, R. Howell, and L. Rosier. Feasibility problems for recurring tasks on one processor. *Theoretical Computer Science*, 118(1):3–20, 1993.

[5] S. Baruah, A. Mok, and L. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proceedings of the 11th Real-Time Systems Symposium*, pages 182–190, Orlando, Florida, 1990. IEEE Computer Society Press.

[6] S. Chakraborty, S. Kunzli, and L. Thiele. A general framework for analysing system properties in platform-based embedded system designs. In *DATE '03: Proceedings of the conference on Design, Automation and Test in Europe*, page 10190, Washington, DC, USA, 2003. IEEE Computer Society.

[7] Z. Deng and J. Liu. Scheduling real-time applications in an Open environment. In *Proceedings of the Eighteenth Real-Time Systems Symposium*, pages 308–319, San Francisco, CA, December 1997. IEEE Computer Society Press.

[8] A. Easwaran. *Compositional Schedulability Analysis Supporting Associativity, Optimality, Dependency and Concurrency*. PhD thesis, Computer and Information Science, University of Pennsylvania, 2007.

[9] A. Easwaran, M. Anand, and I. Lee. Compositional analysis framework using EDP resource models. In *Proceedings of the IEEE Real-time Systems Symposium*, Tuscon, Arizona, December 2007. IEEE Computer Society.

[10] X. A. Feng and A. Mok. A model of hierarchical real-time virtual resources. In *Proceedings of the IEEE Real-Time Systems Symposium*, pages 26–35. IEEE Computer Society, 2002.

[11] N. Fisher. Approximation algorithms for compositional real-time systems: Trading bandwidth for speed-of-analysis. In *Proceedings of the Workshop on Compositional Theory and Technology for Real-Time Embedded Systems*, Barcelona, Spain, December 2008. IEEE Computer Society Press.

[12] N. Fisher and F. Dewan. Approximate bandwidth allocation for compositional real-time systems. In *Proceedings of the EuroMicro Conference on Real-Time Systems*, Dublin, Ireland, July 2009. IEEE Computer Society Press.

[13] N. Fisher and F. Dewan. Approximate bandwidth allocation for compositional real-time systems. Technical report, Department of Computer Science, Wayne State University, 2009. Available at `http://www.cs.wayne.edu/~fishern/papers/PRM-Approx-TR.pdf`.

[14] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *Proceedings of the Real-Time Systems Symposium - 1989*, pages 166–171, Santa Monica, California, USA, Dec. 1989. IEEE Computer Society Press.

[15] G. Lipari and E. Bini. Resource partitioning among real-time applications. In *Proceedings of the EuroMicro Conference on Real-time Systems*, pages 151–160, Porto, Portugal, 2003. IEEE Computer Society.

[16] A. K. Mok. *Fundamental Design Problems of Distributed Systems for The Hard-Real-Time Environment*. PhD thesis, Laboratory for Computer Science, Massachusetts Institute of Technology, 1983. Available as Technical Report No. MIT/LCS/TR-297.

[17] I. Shin and I. Lee. Periodic resource model for compositional real-time guarantees. In *Proceedings of the IEEE Real-Time Systems Symposium*, pages 2–13. IEEE Computer Society, 2003.

[18] I. Shin and I. Lee. Compositional real-time scheduling framework. In *Proceedings of the IEEE Real-Time Systems Symposium*, pages 57–67. IEEE Computer Society, 2004.

[19] I. Shin and I. Lee. Compositional real-time scheduling framework with periodic model. *ACM Transactions on Embedded Computing Systems*, 7(3), April 2008.

[20] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, Berlin-Heidelberg-New York-Barcelona-Hong Kong-London-Milan-Paris-Singapur-Tokyo, 2001.

[21] E. Wandeler and L. Thiele. Real-time interfaces for interface-based design of real-time systems with fixed priority scheduling. In *EMSOFT '05: Proceedings of the 5th ACM international conference on Embedded software*, pages 80–89, New York, NY, USA, 2005. ACM.