Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

Minimizing peak temperature in embedded real-time systems via thermal-aware periodic resources

Masud Ahmed[a], Nathan Fisher[a,*], Shengquan Wang[b], Pradeep Hettiarachchi[a]

[a] *Department of Computer Science, Wayne State University, 5057 Woodward Avenue, Suite 3010, Detroit, MI 48202, USA*
[b] *Department of Computer and Information Science, The University of Michigan - Dearborn, 4901 Evergreen Road, Dearborn, MI 48128, USA*

A R T I C L E   I N F O

Article history:
Received 5 January 2011
Received in revised form 6 May 2011
Accepted 12 May 2011

Keywords:
Real-time systems
Thermal-aware scheduling
Schedulability analysis
Approximation algorithms

A B S T R A C T

Over the years, thermal-aware designs have become a prominent research issue for real-time application development. A drastic increase in energy consumption of modern processors makes devices running real-time applications more prone to overheating and also decreases the lifespan. As a result, obtaining a reduction in peak temperature is considered the most desirable design aspect in developing such devices as it not only decreases the packaging cost, but also increases the lifetime of a device substantially. In this article, the *thermal-aware periodic resource model* is proposed which is a proactive scheme for minimizing peak temperature in a system with a microprocessor having basic energy saving features. For this model, polynomial-time algorithms are proposed to determine the lowest processing time (i.e., bandwidth) in periodic intervals with the minimum peak temperature for a specified sporadic task system scheduled by earliest-deadline first (EDF). The proposed algorithms only incur bounded error (based on a user-defined parameter) in determining temperature in exchange for a significant improvement in running time over exact algorithms. Furthermore, we derive the thermal equations to calculate the asymptotic temperature bound for a given thermal-aware periodic resource. These equations, along with the algorithm presented, will give a system designer not only a guarantee of schedulability for a given workload with minimum peak temperature, but also the freedom of choosing a trade-off between the accuracy and the efficiency of the algorithms.

1. Introduction

Real-time systems depend upon the correctness of both the logical results of the computation and the time at which these results are produced. Timing constraints of real-time systems are commonly specified as deadlines within which individual activities should complete execution. For hard-real-time systems, deadlines are crucially important; failure to meet deadlines may cause critical failures. Meeting deadlines are not the only constraints that real-time devices have to contend with; the temperature of the CPU plays a crucial role in determining the allowable execution speed. High temperature due to excessive heat generated from the CPU while executing heavy workloads can potentially stall the entire system; therefore, temperature control using workload management is especially important for real-time devices. Careful design and analysis of workload enables real-time system designers to determine worst-case demand. Controlling the system execution time based on this worst case demand could maintain temperature within a tolerable threshold. In addition to these constraints, minimization of temperature is also a crucial design choice as the packaging costs of cooling systems are increasing; the cost is now over \$1–3/W of heat dissipated [1] for systems using high-performance processors. Furthermore, temperature has been shown to be a major contributing factor in permanent hardware faults [2] which decrease a system's reliability and lifetime. For managing thermal constraints and objectives, there have been numerous techniques explored over the past few years. Some of these techniques are employed at design time and some at runtime through various forms of dynamic thermal management (DTM).

Modern top-of-the-line processors support various power management features. These features can be effectively used as DTM techniques. The dynamic voltage and frequency scaling (DVFS) approach has become standard for modern processors as a power management feature. DVFS permits control of the energy consumption in the CPU from the operating system. Using DVFS, an application can change the processor's operating voltage and frequency, which in turn can reduce energy consumption as well as the production of heat. For instance, Intel and AMD processors currently support this technology (Intel SpeedStep® and AMDPowerNOW!™). This DVFS technology can be used to min-

* Corresponding author. Tel.: +1 313 577 5421.
E-mail addresses: masud@wayne.edu (M. Ahmed), fishern@cs.wayne.edu (N. Fisher), shqwang@umd.umich.edu (S. Wang), pradeepmh@wayne.edu (P. Hettiarachchi).

2210-5379/$ – see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.suscom.2011.05.006

imize the peak temperature of the system by exploiting the processor idle time. Some low-end embedded processors support other DTM techniques (e.g., low power state, auto clock stop mode, enhanced auto clock freeze mode, and auto halt power down mode). These very basic power-management features can play a vital role in managing temperature of embedded systems as energy consumption contributes directly to excessive heat generation.

Most real-time, and embedded system devices use low-end processors due to manufacturing cost and volume constraints; therefore, these devices may not include the sophisticated DVFS capabilities of the higher-end processors for controlling peak temperature. Low power state, which is a basic power management feature, can be used conveniently for addressing thermal issues. Keeping the processor in a low power state during idle periods as a DTM technique might lower instantaneous temperature [3] as well as the peak temperature. Meisner et al. [4] deployed similar techniques (with the name PowerNap) for eliminating idle powers in datacenter. They developed circuitry for non-real-time systems which would reactivate the processors from low power states if it detected activity in the network interface controller. However, erratically putting the processor in a low power state makes the analysis of thermal behavior extremely difficult. This unpredictability in execution time also makes it extremely difficult to analyze the schedulability of hard-real-time systems tasks.

Unpredictability in execution makes it challenging to derive closed-form thermal equations for calculating peak temperature. So far, systems designed to opportunistically use low-power states have only been evaluated empirically. Furthermore, frequent switching between running state to low power state may incur significant transition overhead. A better approach (for the objective of obtaining a bound on peak system temperature) would be predicting maximum idle cycles in convenient periodic intervals by analyzing the workload beforehand. Allowing idle cycles at the beginning or end of a periodic interval and then switching the processing unit in low power state at those idle periods will allow the system to dissipate heat in a systematic way. This predefined periodicity in the processor running time not only decreases the peak temperature, but also makes the formal thermal and real-time analysis easier and more predictable. In this article, a new scheduling policy and analysis of sporadic task systems [5] upon an uniprocessor based on the processor's *active* and *inactive* modes at precomputed periodic intervals are presented.

1.1. Thermal-aware periodic resource

For maintaining temperature for a hard-real-time systems in a predictable way, we propose the *thermal-aware periodic resource* model which is an extension of the periodic resource model proposed by Shin and Lee [6] for compositional real-time systems. In the thermal-aware periodic resource model, the processing resource will be characterized with a two-tuple (Π, Θ). The parameter Π is the period of repetition and Θ is the resource capacity. We will assume that Π is a non-negative integer (likely subject to the system tick granularity). The interpretation is that processor will be executed in active-mode only Θ units of time in each successive Π-length intervals. A thermal-aware periodic resource differs from the traditional periodic resource in that the thermal-aware resource receives <u>all</u> capacity at the beginning of the Π-length interval; a traditional periodic resource may receive its processing allocation at any time over the interval. Fig. 1 shows the processor's execution pattern. The predictable execution of the thermal-aware periodic resource permits the peak temperature of the system to be derived and described in a function $\mathcal{T}(\Pi, \Theta)$. For a system τ of real-time tasks (specified in the *sporadic task model* and defined in Section 3), the goal is to find Θ and Π so that the system peak temperature $\mathcal{T}(\Pi, \Theta)$ is minimized and no real-time job generated by τ misses its deadline.

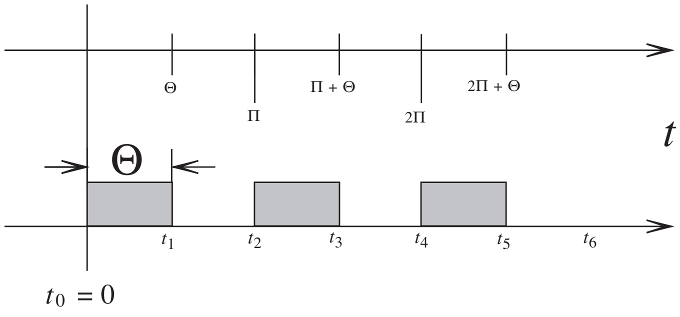

Fig. 1. Execution pattern in thermal-aware periodic resources. Solid rectangles in the bottom portion of the figure depicting capacity each of which has a length of Θ time unit.

1.2. Contribution

In this article, we consider the problem of scheduling a sporadic task system upon a thermal-aware periodic resource with the objective of minimizing the peak system temperature. We assume that our system is a basic single processor with only two discrete modes of operation: active mode and inactive mode. The thermal-aware periodic resource indicates that the processor will execute at the highest speed in the active-mode for Θ contiguous units of time, and at the inactive-mode for $\Pi - \Theta$ contiguous units of time during successive Π-length interval. We obtain algorithms that calculates both Θ and Π. Our result will give the following guarantee:

> Let the minimum peak temperature for executing task system τ upon a thermal-aware periodic resource be obtained using parameters (Π^*, Θ^*). Given an user-defined accuracy parameter $\epsilon > 0$, our algorithm returns $(\hat{\Pi}, \hat{\Theta})$ such that a thermal-aware periodic resource with parameters $(\hat{\Pi}, \hat{\Theta})$ will guarantee the schedulability of the task system τ upon the processor and the peak system temperature $\mathcal{T}(\hat{\Pi}, \hat{\Theta})$ is at most $(1 + \epsilon)\mathcal{T}(\Pi^*, \Theta^*)$; furthermore, our algorithm runs in time polynomial in the representation of τ and $1/\epsilon$. Simulation results confirm the validity of our thermal-aware periodic resources.

Due to the above strong theoretic guarantee, the algorithm may be called a *fully polynomial-time approximation scheme* (FPTAS) [7]. An FPTAS is set of algorithms $\{A_\epsilon \mid \epsilon > 0\}$, where each A_ϵ is an approximation algorithm that returns a solution whose cost (peak temperature in this paper) is at most $(1+\epsilon)$ times the cost of the optimal solution; this ratio is referred to as the *approximation ratio* of the algorithm. Furthermore, the running time of each A_ϵ is bounded by a polynomial function of the input size and $1/\epsilon$. In addition to the above guarantee, our simulation results show that thermal-aware periodic resources reduce the peak temperature compared with single-speed approach and also with the intuitive "sleep-when-idle" approach. Furthermore, the approximation algorithm has been observed to have low running time with high accuracy. Since the scheme has polynomial-time complexity, one could potentially use our scheme for obtaining a quick approximation of the resource parameters if dynamic system reconfiguration is essential; however, the main objective of the proposed scheme is to determine the parameters of the periodic resources at the design time.

Organization. The remainder of this article is organized as follows. Prior related research on thermal-aware real-time systems and periodic resources is reviewed in Section 2. We discuss necessary background and notations to describe the problem in the context of a sporadic task system in Section 3. Thermal equations for determining the peak system temperature of processor executing as a thermal-aware periodic resource are derived in Section 4. Using the thermal equations, an algorithm for calculating resource capacity is presented in Section 5. Another algorithm is presented in Section 6 for calculating the period-of-repetition for the resource. Section 7 shows simulation results for the proposed algorithms. Finally, we present the conclusion of the study and also the directions of the future research in Section 8.

2. Related work

In previous work on scheduling under thermal constraints, researchers have followed two main approaches: reactive and proactive schedulers. In a reactive scheduler, the processor speed is reduced in response to a thermal trigger (e.g., the maximum system operating temperature is reached). In the proactive setting, the speed schedule for the processor is determined at design time. The following section reviews related research on thermal-aware scheduling for proactive and reactive settings, dynamic thermal management strategies, and related work for periodic scheduling without considering thermal constraints.

Chen et al. [8] explored proactive speed scheduling for periodic real-time tasks under DVFS to meet both timing and thermal constraints. Chantem et al. [9] made an interesting observation for maximizing workload under thermal constraints. While working with proactive scheduling, the authors show that the scheduler which maximizes workload under thermal constraints must be a periodic one. The authors determined a speed schedule such that the peak temperature constraints were met and total work completed was maximized using a dynamic voltage and frequency scaling (DVFS) control policy for processors with discrete speed levels. The problem considered in our article is orthogonal to Chantem et al.'s work in the sense that their objective is maximizing the workload under thermal constraints; in this article, we consider the setting where the system designer has a fixed, pre-defined workload (i.e., a specified task system) and seeks to minimize the peak system temperature.

Wang et al. [10–12] studied temperature-constrained real-time systems and performed schedulability analysis of a task system under the reactive setting. The authors computed upper bound on the worst-case delay for tasks with arbitrary job arrivals for both first-in-first-out (FIFO) and static-priority (SP) scheduling algorithms, and also showed that this simple reactive speed control decreased the response time of tasks compared with any constant-speed scheme. However, the previous aforementioned work on proactive and reactive schemes assumes either simple task models or the existence of "ideal" processor speeds which may not be feasible even by using the recent top-of-the-line microprocessors. Our proposed thermal-aware periodic resource model may be considered a proactive scheduler. We remove some ideal assumptions made in prior work by assuming only two discrete operating modes and the more general sporadic task model. Furthermore, we account for the transition overhead due to switching between modes.

There has been extensive research work on finding an optimal feasible speed for minimizing energy consumption as well as temperature minimization. Finding an optimal speed is always possible under the assumption that the processor can run at any speed. Yao et al. [13] formulated speed scaling problems as scheduling problems to minimize energy consumption. The authors developed an optimal offline greedy polynomial-time algorithm, and also two online algorithms. Bansal et al. [14] explored algorithms for minimizing both peak temperature and energy efficiency, and proposed an algorithm with constant approximation ratio with respect to the optimal algorithm for managing temperature. Hung et al. [15] investigated both power-aware and thermal-aware approaches and found a power-aware approach alone is not able to address the temperature challenge; furthermore, many low-power techniques have insufficient impact on chip temperature because they do not directly target the spatial and temporal behavior of the operating temperature. DTM strategies also exist for multiprocessor real-time systems [16–18]; however, most of these focus upon static speed-assignment approaches and not on a proactive schedule. Real-time thermal analysis has been studied in the context of web servers [19], but hard deadlines are not guaranteed. Real-time scheduling under thermal constraints has also been evaluated for multi-function phased array radar systems via energy minimization [20,21].

Numerous results exist for the periodic resource model without thermal settings. Shin and Lee [22,6] obtained linear-time algorithms for determining the capacity of a periodic resource (given a periodic task system); however, the allocation may potentially over-provision the amount of capacity needed by the task system – leading to wasted processor resources. Easwaran [23,24] devised an exact algorithm for capacity determination and for determining the best period of repetition. While these exact algorithms ensure that no processing resources are wasted, they may require exponential-time in the worst-case. Bini et al. [25] studied minimization of energy consumption using periodic-resource considering discrete speed DVFS processor. Fisher and Dewan [26] developed a polynomial-time approximation algorithm which permits a system designer to choose a trade-off between accuracy and response time based on their requirements for non-thermal settings. In a related publication, Fisher [27] proposed a polynomial-time approximation for determining the period of repetition in a periodic resource. In this article, we leverage these two approximation results to obtain an approximation algorithm for thermal-aware periodic resources which minimizes peak system temperature.

3. Notations and background

We are interested in analyzing thermal characteristics of real-time systems. In this section, we present definitions, notation, and some preliminary fundamental results that are useful in the remainder of the article.

3.1. Processor model

We consider a processor with *active* and *inactive* modes. This model of execution is very general, and can be implemented even with basic power management features (e.g., auto clock stop mode). We assume that the processor generates heat while executing in the active-mode. For the inactive-mode, the processor will likely need to maintain a very low-power state; thus, we will assume that the processor continues to generate a small amount of heat during the inactive state. Even though the processor is minimally active while in the low-power state, we will assume that the processor is unavailable for task execution during this interval. The processor dissipates the produced heat through the entire execution time. Since the rate of heat generation is higher during the active-mode period, it is easy to see that the peak system temperature must occur at the end of an active-mode interval of execution; a similar observation was made by Wang and Bettati [10]. We also account for the transition time overhead between active and inactive modes by the parameter δ (e.g., the transition time for AMD K6-2+ processor is 0.4 μs [28]). For every active-mode interval, we will assume that the processor is unavailable to tasks for δ units while the processor is switching between modes. Furthermore, we assume that the processor is potentially executing at the full speed for δ units of time during the transitions to obtain a safe upper bound on temperature. (Note that these last two assumptions are pessimistic for the sake of obtaining safe temperature bounds and real-time schedulability. If the system does not satisfy either of these last two assumptions, then it will behave better than expected and the derived results will continue to hold.)

Formally, the system is ready for execution at time instant t_0. We assume that temperature at this time is normalized to zero and is equal to the surrounding ambient temperature. The processor will also start executing at time instant t_0. Assuming the processor starts its execution from an even numbered time instant, we define $t_{2i} = i\Pi$ and $t_{2i+1} = i\Pi + \Theta + \delta$ for each value of $i \in \mathbb{N}$. Therefore, the gap from an even time instant to an odd time instant is $\Theta + \delta$, whereas from an odd time instant to an even time instant – the interval length is $\Pi - \Theta - \delta$. Fig. 1 illustrates the time instants when δ equals zero. The processor generates heat during its execution interval (from an even time instant to an odd time instant) and dissipates heat in the remainder of the period-of-repetition. Since the peak temperature will occur at the end of the execution in every period-of-repetition, it suffices to consider the times $t \in \{t_{2i+1} \mid i \geq 0\}$ as the time instants where the peak temperature may occur.

3.2. Thermal model

Finding peak temperature is difficult as both heating and cooling systems are complicated dynamic processes which depend on the surrounding environment. We could approximately model this process by applying Fourier's Law of heat conduction [29,17,15,30]. Fourier's Law of heat conduction states that the rate of cooling is proportional to the difference in temperature between the object and the environment. We assume that the environment has a fixed temperature, and that temperature is scaled so that the ambient temperature is zero. Previous real-time research with thermal constraints [10–12] has also worked with the identical ambient temperature assumption. (In an ongoing research project, we are currently developing techniques for eliminating the assumption of a fixed ambient temperature.)

If we define $T(t)$ as the temperature at time instant t, then $T(t)$ can be calculated (e.g. [8]) as follows

$$T(t) = \left(\int_{t_0}^{t} (s(\tau))^{\gamma} e^{-\beta(t-\tau)}\, d\tau\right) + T(t_0) e^{-\beta(t-t_0)} \quad (1)$$

where

$$s(t) = \begin{cases} \varphi, & \text{if } t \text{ occurs during active mode;} \\ \varphi r_{\text{off}} & \text{if } t \text{ occurs during inactive mode;} \end{cases}$$

The function $s(t)$ can be thought of as a function of time that calculates the instantaneous speed of the processor at t. The parameters β and γ are processor specific constants. Typical settings for these two variables are $\beta \approx 0.228$, $\gamma \approx 3$ (e.g., see [8]). The term φ is the processor speed during active mode. r_{off} is the fraction of speed at which the processor executes in the inactive mode (compared to the speed during the active mode) where $0 \leq r_{\text{off}} < 1$.

3.3. Task model

A *sporadic task* $\tau_i = (e_i, d_i, p_i)$ is characterized by a *worst-case execution-time requirement* e_i (with respect to the active speed φ), a *(relative) deadline* d_i, and a *minimum inter-arrival separation* p_i. 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 generated by τ_i must successfully complete execution within d_i time units of its arrival and will execute for at most e_i time units. A *sporadic task system* $\tau \overset{\text{def}}{=} \{\tau_1, \ldots, \tau_n\}$ is a collection of n such sporadic tasks. Tasks are assumed to be independent. A useful metric

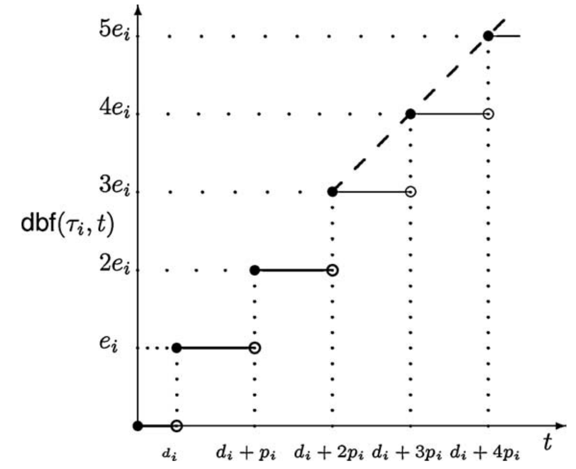

Fig. 2. The step function denotes a plot of $\mathsf{dbf}(\tau_i, t)$ as a function of t. The dashed line represents the function $\widetilde{\mathsf{dbf}}(\tau_i, t, k)$, approximating $\mathsf{dbf}(\tau_i, t)$. $\widetilde{\mathsf{dbf}}(\tau_i, t, k)$ is equal to $\mathsf{dbf}(\tau_i, t)$ for all $t < d_i + (k-1)p_i$ (k equals three in the above graph).

for a sporadic task τ_i is the task utilization $u_i = e_i/p_i$. The system utilization is denoted by $U(\tau) = \sum_{\tau_i \in \tau} u_i$. In this article, we assume that tasks are executed by the preemptive earliest-deadline-first (EDF) scheduling algorithm [32]; that is, at each time instant during which the processor is active, the job with the earliest absolute deadline (among all jobs with remaining execution) is scheduled on the processor.

3.4. Definitions

A task system τ is said to be EDF-*schedulable*, if for all legal job arrivals of task system τ, EDF can always meet the deadlines of all jobs generated by τ. For determining EDF-schedulability of a sporadic task system, it is often useful to quantify the maximum amount of execution that must be completed over any given interval. Baruah et al. [33] have derived the *demand-bound function*, defined in the following way.

Definition 1 *(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. [33] have shown that, for sporadic tasks, dbf can be calculated as follows.

$$\mathsf{dbf}(\tau_i, t) = \max\left(0, \left\lfloor \frac{t - d_i}{p_i} \right\rfloor + 1\right) \cdot e_i. \quad (2)$$

Fig. 2 gives a visual depiction of the demand-bound function for a sporadic task τ_i. Observe from the above definition and Fig. 2 that the dbf is a right continuous function with discontinuities at time points of the form $t = d_i + a \cdot p_i$ where $a \in \mathbb{N}$. Let $\mathsf{DBF}(\tau, t) \overset{\text{def}}{=} \sum_{\tau_i \in \tau} \mathsf{dbf}(\tau_i, t)$. It has been shown [33] 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:

$$\mathsf{TS}(\tau) \overset{\text{def}}{=} \bigcup_{\tau_i \in \tau} \{t = d_i + a \cdot p_i | (a \in \mathbb{N}) \wedge (t \leq P(\tau))\}. \quad (3)$$

where $P(\tau)$ is an upper bound on the maximum time instant that the schedulability condition must be verified. For EDF-scheduled sporadic task systems on preemptive unit-speed processors, $P(\tau)$ is at most $\mathsf{lcm}_{\tau_i \in \tau}\{p_i\} + \max_{\tau_i \in \tau}\{d_i\}$. 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 [34] proposed the following approximation to dbf to reduce the number of discontinuities (and, thus, points in the testing set).

$$\widetilde{\mathsf{dbf}}(\tau_i, t, k) \stackrel{\text{def}}{=} \begin{cases} \mathsf{dbf}(\tau_i, t), & \text{if } t < d_i + (k-1)p_i; \\ u_i \cdot (t - d_i) + e_i, & \text{otherwise.} \end{cases} \tag{4}$$

Let $\widetilde{\mathsf{DBF}}(\tau, t, k)$ equal $\sum_{i=1}^{n} \widetilde{\mathsf{dbf}}(\tau_i, t, k)$. Furthermore, $\widetilde{\mathsf{DBF}}(\tau, t, k) \leq t$ for all $t \geq 0$ is a sufficient condition for schedulability, as $\widetilde{\mathsf{DBF}}(\tau, t, k) \geq \mathsf{DBF}(\tau, t)$ for all $t \geq 0$. The effect of redefining dbf to $\widetilde{\mathsf{dbf}}$ is a reduction of TS to a smaller set of kn testing set points, where k is a parameter ($k \in \mathbb{N}^+$) chosen by the system designer:

$$\widetilde{\mathsf{TS}}(\tau, k) \stackrel{\text{def}}{=} \bigcup_{\tau_i \in \tau} \{t = d_i + a \cdot p_i | (a \in \mathbb{N}) \wedge (a < k) \wedge (t \leq P(\tau))\}. \tag{5}$$

Albers and Slomka [34] make the following observation regarding the relationship between dbf and $\widetilde{\mathsf{dbf}}$. We will use this observation for our approximation scheme presented in Section 5.

Lemma 1 (from [34]). *Given a fixed integer* $k \in \mathbb{N}^+$, $\mathsf{dbf}(\tau_i, t) \leq \widetilde{\mathsf{dbf}}(\tau_i, t, k) \leq ((k+1)/k)\mathsf{dbf}(\tau_i, t)$ *for all* $\tau_i \in \tau$ *and* $t \in \mathbb{R}_{\geq 0}$.

Consider any $t \in \widetilde{\mathsf{TS}}(\tau, k)$ and $D_t = \widetilde{\mathsf{DBF}}(\tau, t, k)$. For the algorithm derived in Section 5, it is useful to view the "step" of the demand-bound function originating at (t, D_t) as a *half-line* starting at (t, D_t) and extending infinitely to the right. Let $\langle (t, D_t), \alpha_{t,k} \rangle$ denote the half-line in Euclidean space $\mathbb{R}^2$, originating at point $(t, D_t) \in \mathbb{R}^2$ with slope $\alpha_{t,k}$ where $0 \leq \alpha_{t,k} \leq 1$ (i.e., $\langle (t, D_t), \alpha_{t,k} \rangle = \{(x, y) \in \mathbb{R}^2 | (x \geq t) \wedge (y = \alpha_{t,k}(x - t) + D_t)\}$). Additionally, we may calculate $\alpha_{t,k}$, the slope (of $\widetilde{\mathsf{DBF}}(\tau, t, k)$), at any time t:

$$\alpha_{t,k} \stackrel{\text{def}}{=} \sum_{\tau_i \in \tau : t \geq d_i + (k-1)p_i} u_i. \tag{6}$$

Note that $\alpha_{t,\infty}$ is zero for all t. The above characterization of dbf as a set of half-lines was used by Dewan and Fisher [26] and will serve as the basis of our algorithm in Section 5.

Definition 2 *(Supply-Bound Function).* For any $t > 0$, the supply-bound function (sbf) quantifies the minimum execution supply that a task system executed upon a processor may receive over any interval of length t. In thermal-aware periodic resource model, the processor executes the workload by Θ amount in successive Π-length interval (at the end). For this model, sbf takes the following form:

$$\mathsf{sbf}(t) = \begin{cases} (q-1)\Theta, & \text{if } t \in ((q-1)\Pi, q\Pi - \Theta] \\ (t - q(\Pi - \Theta)), & \text{if } t \in (q\Pi - \Theta, q\Pi]. \end{cases} \tag{7}$$

where $q = \lceil (t/\Pi) \rceil$. q can be thought of as the processor execution cycle number for the corresponding t.

EDF-schedulability conditions have been developed [22,35,24], as given in the following theorem.

Theorem 1 (from [24]). *A sporadic task system τ is* EDF*-schedulable upon a thermal-aware periodic resource* (Π, Θ), *if and only if,*

$$(\mathsf{DBF}(\tau, t) \leq \mathsf{sbf}(t),\ \forall t \leq P(\tau)) \wedge \left(U(\tau) \leq \frac{\Theta}{\Pi} \right) \tag{8}$$

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

See Fig. 3 for a visual depiction of sbf. We may substitute $\widetilde{\mathsf{DBF}}$ into the above theorem to obtain a sufficient condition for schedulability [26].

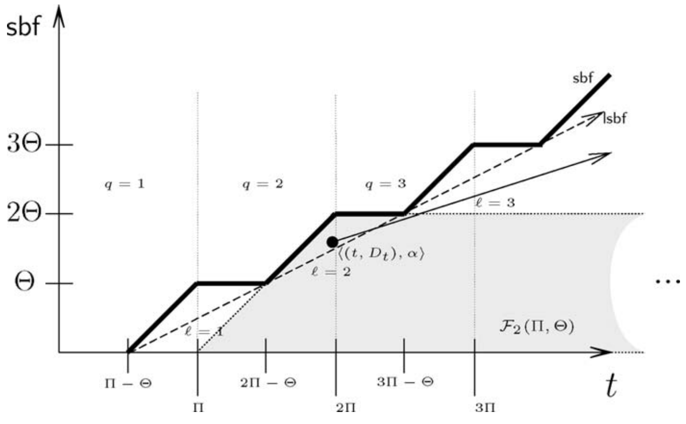

Fig. 3. The solid line "step" function is sbf. The shaded area is the "space" below the second step of the sbf. The half-line originating at (t, D_t) must fall below the sbf for all values after t.

4. Peak temperature of periodic resources

In this section, we derive asymptotic bounds on the highest temperature for a given thermal-aware periodic resource (Π, Θ) using Eq. (1). We denote the highest temperature as the asymptotic peak temperature of the system. Recall that the ambient temperature is normalized to zero; therefore, the initial temperature of the processor will be also zero ($T(t_0) = 0$). We will use Eq. (1) to derive the temperature at the beginning and at the ending point of each execution cycle. As mentioned in Section 3, the peak temperature will occur at the end of the execution in every period-of-repetition; therefore, we check the temperature at each point in the set $\{t_{2i+1} | t_{2i+1} = i\Pi + \Theta + \delta, i \in \mathbb{N}\}$.

Theorem 2. *Given $i \in \mathbb{N}$, the temperature of a processor executing as a thermal-aware periodic resource with the parameters $\Pi > 0$ and $\Theta \geq 0$ at the time instant t_{2i+1} can be obtained*

$$T(t_{2i+1}) = \frac{\varphi^\gamma}{\beta}(1 - e^{-\beta(\Theta+\delta)})\frac{1 - (e^{-\beta\Pi})^{i+1}}{1 - e^{-\beta\Pi}} + \frac{(\varphi r_{\text{off}})^\gamma}{\beta} e^{-\beta(\Theta+\delta)}(1 - e^{-\beta(\Pi-\Theta-\delta)})\frac{1 - (e^{-\beta\Pi})^i}{1 - e^{-\beta\Pi}} \tag{9}$$

Proof. The proof is by induction on the periodic time instants indexed by i.

(Base Case): Consider $i = 0$; at the time instant t_0, the temperature $T(t_0)$ is equal to the ambient temperature (assumed to be zero). For showing the base case, we find the temperature at t_1. Here, it is to be noted that $s(t) = \varphi$ for all $t \in [t_0, t_1)$ as the processor will be executing the workload with a constant speed in this interval. We

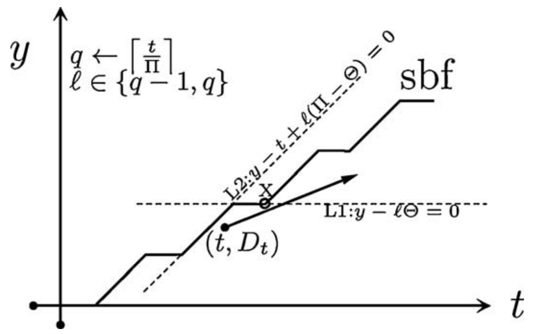

Fig. 4. Determination of capacity for the half-line originating at (t, D_t) must fall below the sbf for all values after t. Solving $L1$, $L2$, and X with (t, D_t) will give $\Theta_\ell^{\min}$ associated with (t, D_t).

know from the definition of t_{2i} and t_{2i+1} in Section 3 that $t_1 = \Theta + \delta$ and $t_0 = 0$.

$$\begin{aligned}T(t_1) &= \int_{t_0}^{t_1} s^{\gamma}(t) e^{-\beta(t_1 - t)}\,\mathrm{d}t + T(t_0)e^{-\beta(t_1-t_0)}\\ &= \int_0^{\Theta+\delta} \varphi^{\gamma} e^{-\beta(\Theta+\delta-t)}\,\mathrm{d}t + 0\cdot e^{-\beta(\Theta+\delta)}\end{aligned}$$

$$\begin{aligned}&= \frac{\varphi^{\gamma}}{\beta}(1 - e^{-\beta(\Theta+\delta)}) + 0\cdot e^{-\beta(\Theta+\delta)}\\ &= \frac{\varphi^{\gamma}}{\beta}(1 - e^{-\beta(\Theta+\delta)})\frac{1-(e^{-\beta\Pi})^1}{1-(e^{-\beta\Pi})} + \frac{(\varphi r_{\text{off}})^{\gamma}}{\beta} e^{-\beta(\Theta+\delta)}(1-e^{-\beta(\Pi-\Theta-\delta)})\frac{1-(e^{-\beta\Pi})^0}{1-(e^{-\beta\Pi})}\end{aligned} \tag{10}$$

The last step follows by noting that $1-(e^{-\beta\Pi})^0$ equals zero.

(Induction Hypothesis): Assume that Eq. (9) is true for $i = 1, 2, \ldots m$. Therefore

$$\begin{aligned}T(t_{2m+1}) = {}& \frac{\varphi^{\gamma}}{\beta}(1-e^{-\beta(\Theta+\delta)})\frac{1-(e^{-\beta\Pi})^{m+1}}{1-e^{-\beta\Pi}}\\ &+ \frac{(\varphi r_{\text{off}})^{\gamma}}{\beta} e^{-\beta(\Theta+\delta)}(1-e^{-\beta(\Pi-\Theta-\delta)})\frac{1-(e^{-\beta\Pi})^m}{1-e^{-\beta\Pi}}\end{aligned} \tag{11}$$

(Inductive Step): In order to find the temperature at $t_{2(m+1)+1}$ using Eq. (1), we will use the temperature at $t_{2(m+1)}$ as the base temperature. From the definition, we know $t_{2m+1} = m\Pi + \Theta + \delta$, $t_{2(m+1)} = (m+1)\Pi$ and $t_{2(m+1)+1} = (m+1)\Pi + \Theta + \delta$. Temperature at $t_{2(m+1)}$ can be obtained by the following equation:

Corollary 1. *The asymptotic temperature of a thermal-aware periodic resource with the parameters* $\varphi > 0$, $\Pi > 0$, $\delta \geq 0$, *and* $\Theta \geq 0$ *is*

$$\begin{aligned}\mathcal{T}(\Pi,\Theta) = {}& \frac{\varphi^{\gamma}}{\beta}\frac{(1-e^{-\beta(\Theta+\delta)})}{(1-e^{-\beta\Pi})} + \frac{(\varphi r_{\text{off}})^{\gamma}}{\beta} \times e^{-\beta(\Theta+\delta)}\\ &\times \frac{1-e^{-\beta(\Pi-\Theta-\delta)}}{1-e^{-\beta\Pi}}.\end{aligned} \tag{13}$$

Proof. The corollary follows directly from Theorem 2. As the highest temperature will occur at the end of the execution in every period-of-repetition and Eq. (9) is non-decreasing over i, we can find the asymptotic peak temperature by taking $\lim_{i\to\infty}$ of Eq. (2), resulting in Eq. (13). □

5. Capacity determination

From Eq. (13), the only free parameters which may be easily changed by the system designer are the period-of-repetition Π and capacity Θ. So, for optimizing temperature, there is no other alternative except choosing an optimal (Π^*, Θ^*) pair which will result minimum peak temperature. In the next section, we develop an algorithm named MinimumCap for optimizing capacity Θ given period-of-repetition Π. Later, we will develop the final algorithm SelectPeriod for calculating the optimal period-of-repetition for the processor based on minimum peak temperature using this minimum capacity determination algorithm MinimumCap. Both MinimumCap and SelectPeriod are thermal-aware extensions of the approximation algorithms for compositional real-time systems [26,27],

$$\begin{aligned}&T(t_{2(m+1)})\\ &= \int_{t_{2m+1}}^{t_{2(m+1)}} (\varphi r_{\text{off}})^{\gamma} e^{-\beta(t_{2(m+1)}-t)}\,\mathrm{d}t + T(t_{2m+1})e^{-\beta(t_{2(m+1)}-t_{2m+1})}\\ &= \int_{m\Pi+\Theta+\delta}^{(m+1)\Pi} (\varphi r_{\text{off}})^{\gamma} e^{-\beta(t_{2(m+1)}-t)}\,\mathrm{d}t + T(t_{2m+1})e^{-\beta(t_{2(m+1)}-t_{2m+1})}\\ &= \frac{(\varphi r_{\text{off}})^{\gamma}}{\beta}(1-e^{-\beta(\Pi-\Theta-\delta)}) + e^{-\beta(\Pi-\Theta-\delta)} \times \left[\frac{\varphi^{\gamma}}{\beta}(1-e^{-\beta(\Theta+\delta)}).\frac{1-(e^{-\beta\Pi})^{m+1}}{1-e^{-\beta\Pi}} + \frac{(\varphi r_{\text{off}})^{\gamma}}{\beta}e^{-\beta(\Theta+\delta)}(1-e^{-\beta(\Pi-\Theta-\delta)})\frac{1-(e^{-\beta\Pi})^m}{1-e^{-\beta\Pi}}\right]\\ &= e^{-\beta(\Pi-\Theta-\delta)}\frac{\varphi^{\gamma}}{\beta}(1-e^{-\beta(\Theta+\delta)})\frac{1-(e^{-\beta\Pi})^{m+1}}{1-e^{-\beta\Pi}} + \frac{(\varphi r_{\text{off}})^{\gamma}}{\beta}(1-e^{-\beta(\Pi-\Theta-\delta)})\frac{1-\left(e^{-\beta\Pi}\right)^{m+1}}{1-e^{-\beta\Pi}}\end{aligned} \tag{12}$$

Therefore,

$$\begin{aligned}&T(t_{2(m+1)+1})\\ &= \int_{t_{2(m+1)}}^{t_{2(m+1)+1}} s^{\gamma}(t)e^{-\beta(t_{2(m+1)+1}-t)}\,\mathrm{d}t + T(t_{2(m+1)})e^{-\beta(t_{2(m+1)+1}-t_{2(m+1)})}\\ &= \int_{(m+1)\Pi}^{(m+1)\Pi+\Theta+\delta} \varphi^{\gamma} e^{-\beta((m+1)\Pi+\Theta+\delta-t)}\,\mathrm{d}t + T(t_{2(m+1)})e^{-\beta(\Theta+\delta)}\\ &= \frac{\varphi^{\gamma}}{\beta}(1-e^{-\beta(\Theta+\delta)}) + \left[e^{-\beta(\Pi-\Theta-\delta)}\frac{\varphi^{\gamma}}{\beta}(1-e^{-\beta(\Theta+\delta)})\frac{1-(e^{-\beta\Pi})^{m+1}}{1-e^{-\beta\Pi}} + \frac{(\varphi r_{\text{off}})^{\gamma}}{\beta}(1-e^{-\beta(\Pi-\Theta-\delta)})\frac{1-(e^{-\beta\Pi})^{m+1}}{1-e^{-\beta\Pi}}\right] \times e^{-\beta(\Theta+\delta)}\end{aligned}$$

$$\begin{aligned}&= \frac{\varphi^{\gamma}}{\beta}(1-e^{-\beta(\Theta+\delta)}) + \left[e^{-\beta(\Pi-\Theta-\delta)}\frac{\varphi^{\gamma}}{\beta}(1-e^{-\beta(\Theta+\delta)})\frac{1-(e^{-\beta\Pi})^{m+1}}{1-e^{-\beta\Pi}} + \frac{(\varphi r_{\text{off}})^{\gamma}}{\beta}(1-e^{-\beta(\Pi-\Theta-\delta)})\frac{1-(e^{-\beta\Pi})^{m+1}}{1-e^{-\beta\Pi}}\right] \times e^{-\beta(\Theta+\delta)}\\ &= \frac{\varphi^{\gamma}}{\beta}(1-e^{-\beta(\Theta+\delta)})\frac{1-(e^{-\beta\Pi})^{m+2}}{1-e^{-\beta\Pi}} + \frac{(\varphi r_{\text{off}})^{\gamma}}{\beta}e^{-\beta(\Theta+\delta)}(1-e^{-\beta(\Pi-\Theta-\delta)})\frac{1-(e^{-\beta\Pi})^{m+1}}{1-e^{-\beta\Pi}}\end{aligned}$$

which proves the theorem. □

Algorithm 1. MinimumCap(Π, τ, k).

```
1:  $\Theta^{\min} \leftarrow U(\tau)\cdot\Pi$
2:  for all $t \in \widetilde{\mathrm{TS}}(\tau, k)$ do
3:      $D_t \leftarrow \widetilde{\mathrm{DBF}}(\tau, t, k)$
4:      $\alpha_{t,k} \leftarrow \sum_{\tau_i\in\tau : t \ge d_i+(k-1)p_i} u_i$
5:      $\Theta_t^{\min} \leftarrow \infty$
6:      $q \leftarrow \left\lceil \frac{t}{\Pi} \right\rceil$
7:      for all $\ell \in \{q-1, q\}$ do
8:          $\Theta_\ell^{\min} \leftarrow \max\left\{ \alpha_{t,k}\Pi, \frac{D_t - t + \ell\Pi}{\ell}, \frac{D_t}{\ell}, \frac{D_t + \alpha_{t,k}((\ell+1)\Pi - t)}{\ell + \alpha_{t,k}} \right\}$
9:          $\Theta_t^{\min} \leftarrow \min\{\Theta_t^{\min}, \Theta_\ell^{\min}\}$
10:     end for
11:     $\Theta^{\min} \leftarrow \max\{\Theta^{\min}, \Theta_t^{\min}\}$
12: end for
13: retrun $\Theta^{\min}$
```

5.1. Description

We now give a description of pseudocode for the algorithm, MinimumCap(Π, τ, k), for determining the minimum capacity required to correctly schedule task system τ according to EDF upon a processor with given fixed Π parameter. (Pseudocode is given in Algorithm 1.) We will show how to determine the optimal Π for which peak temperature is minimized in the next section. The intuition behind algorithm MinimumCap is as follows: for each value t in the testing set $\widetilde{\mathrm{TS}}(\tau, k)$ find the minimum capacity $\Theta_t^{\min}$ required to guarantee that the half-line $\langle(t, \widetilde{\mathrm{DBF}}(\tau, t, k)), \alpha_{t,k}\rangle$ is completely beneath $\mathrm{sbf}((\Pi, \Theta_t^{\min}), t)$. To ensure that the half-line $\langle(t, \widetilde{\mathrm{DBF}}(\tau, t, k)), \alpha_{t,k}\rangle$ is below the ℓ th step of the sbf, Line 8 of MinimumCap determines the minimum capacity required for that step. Fig. 4 illustrates the reasoning behind the value set at Line 8.[1] The half-line is equal to $\widetilde{\mathrm{DBF}}(\tau, t, k)$ until the next testing set time-point. So, any capacity greater than this minimum capacity $\Theta_t^{\min}$ will ensure that $\widetilde{\mathrm{DBF}}$ falls below sbf up until the next point in the testing set. If we set $\Theta^{\min}$ to be the maximum of all these $\Theta_t^{\min}$ (Line 11 of MinimumCap) and $U(\tau)\cdot\Pi$, then we ensure that each "step" of $\widetilde{\mathrm{DBF}}(\tau, t, k)$ is strictly less than $\mathrm{sbf}((\Pi, \Theta^{\min}), t)$ and $U(\tau) \le (\Theta^{\min}/\Pi)$. Since $\widetilde{\mathrm{DBF}}(\tau, t, k) \ge \mathrm{DBF}(\tau, t)$ for all t, this implies schedulability condition; thus, τ is EDF-schedulable upon a thermal-aware periodic resource. Note that when $k = \infty$, the algorithm is exact.

5.2. Complexity and correctness

Formally in order to prove the validity of the algorithm, all we have to prove the theorem below.

Theorem 3. *Given Π, τ, and $1 > \epsilon > 0$, the procedure* MinimumCap(Π, τ, $\lceil(1/\epsilon)\rceil$) *returns $\Theta^{\min}$ such that τ is* EDF-*schedulable upon thermal-aware periodic resource* (Π, $\Theta^{\min}$) *and*

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

where $\Theta^*(\Pi, \tau)$ denote the optimal capacity for the task systems τ to be schedulable. Furthermore, MinimumCap(Π, τ, $\lceil(1/\epsilon)\rceil$) *has time complexity $O((n\lg n)/\epsilon)$.*

A formal proof is nearly identical to [26]. For maintaining the continuity and due to the similarity with [26], we omit the formal proof in this section. Instead, in Appendix A, we summarize a formal proof and highlight the necessary changes in the proof of [26] required for the thermal-aware periodic resource model.

6. Period selection

In Section 4, we presented a formula for determining asymptotic peak temperature which depends upon the capacity and period-of-repetition parameters. However, it is not immediately obvious which choice of Π gives us the minimum peak temperature (given the capacity determination algorithm of Section 5 which calculates Θ from a given Π). The example below shows that it may *not* suffice to just consider the minimum or maximum possible value of Π. Thus, to obtain the parameters for the minimum peak temperature, we may be required to check *every* Π (and resulting Θ) with the formula presented in Corollary 1. In Section 5, we propose an efficient approximation algorithm for choosing a near optimal value of Π.

6.1. Motivating example

Example 1. Consider a sequence of jobs from two tasks [denoted by (e_i, d_i, p_i)]: (1, 5, 10) and (1, 10, 20). Let the legal values of Π be {2, 3, 4, 5, 6}. In order to be fully schedulable, the task set requires a capacity $\Theta = 0.5$ when $\Pi = 2$; $\Theta = 1$ when $\Pi = 3$; $\Theta = 1$ when $\Pi = 4$; $\Theta = 1$ when $\Pi = 5$; and $\Theta = 2$ when $\Pi = 6$. (For this example, we are assuming the following processor and thermal parameters: $\beta = 0.228$, $\gamma = 3$, $\varphi = 1$ and $P_{off} = 0.05$.) Among all these pairs, $\left(\Pi = 2, \Theta = 0.5\right)$ generates the largest peak temperature of approximately 10.50 °C while $\left(\Pi = 5, \Theta = 1\right)$ generates the smallest peak temperature of 9.81 °C. Thus, neither the smallest nor the largest value of Π in a range necessarily result in the minimum peak temperature; in fact, the minimum peak temperature can happen anywhere in the range.

From the example in the previous paragraph, our observation is that the temperature minimization requires period to be selected prudently from a range $\left\{\Pi_{lower}, \ldots, \Pi_{upper}\right\}$ based on the thermal-equation presented in Section 4. A lower limit (Π_{lower}) of this domain depends on the characteristics of the CPU and should be no less than the transition overhead δ. A safe upper bound for period-of-repetition is the LCM of task-periods. A large period could result in a relatively higher capacity, thus jeopardizing the objective of temperature minimization. On the other hand, a smaller period would result in a large number of transitions between active and inactive modes; thus, increasing the overhead. Considering such trade-offs, we propose a period selection algorithm in the next section for finding the (near) optimal period-of-repetition with respect to minimizing peak temperature.

6.2. Period selection algorithm

We consider, in this section, the problem of determining the optimal choice of period-of-repetition parameter for the thermal-aware periodic resource. Here the main objective is to find an approximation to Π^*, Θ^*, and the peak temperature with the bounded deviation from the optimal solution. Recall from the first section, $\mathcal{T}(\Pi, \Theta)$ denotes the peak temperature if the system is active only for Θ amount of time in a successive Π-length intervals. As Π^* and Θ^* are chosen in such way for which systems get

[1] The second expression in the right-hand side of Line 8 ensures that (t, D_t) is less than any point on the supply line during the ℓ th active state (line L2 in Fig. 4); the third expression on the right-hand side of Line 8 ensures the supply received before the $\ell + 1$ th inactive state (line L1 in Fig. 4) is sufficiently large. Together the first and fourth expression on the right-hand side of instruction at Line 8 ensures sufficient supply until the intersection (circle on the sbf of L1 and L2, labeled as X) of the supply for the $\ell + 1$ inactive and active states. A more formal justification may be found in Appendix A.

the lowest peak temperature; $\mathcal{T}(\Pi^*, \Theta^*)$ denotes the optimal peak temperature of thermal-aware periodic resource.

Algorithm 2. SelectPeriod(τ, Π_{lower}, Π_{upper}, ϵ).

```
1:  $T_{\min} \leftarrow \infty$
2:  $\Pi_{\text{last}} \leftarrow \Pi_{\text{lower}}$
3:  $\Theta_{\text{last}} \leftarrow$ MinimumCap($\Pi_{\text{last}}, \tau, \lceil \frac{3}{\epsilon} \rceil$)
4:  repeat
5:    $T_c \leftarrow \mathcal{T}(\Pi_{\text{last}}, \Theta_{\text{last}})$
6:    if $T_c < T_{\min}$ then
7:      $T_{\min} \leftarrow T_c$
8:      $\hat{\Theta} \leftarrow \Theta_{\text{last}}$
9:      $\hat{\Pi} \leftarrow \Pi_{\text{last}}$
10:   end if
11:   $\Theta \leftarrow \left(1 + \frac{\epsilon}{3}\right) \Theta_{\text{last}}$
12:   $\Pi_l \leftarrow \Pi_{\text{last}}$
13:   $\Pi_m \leftarrow \Pi_{\text{upper}}$
14:   {Perform binary search for maximum $\Pi$ for}
15:   {which capacity less than $\Theta$.}
16:   while $\Pi_l \le \Pi_m$ do
17:     $\Pi_{\text{mid}} \leftarrow \lfloor \frac{\Pi_l + \Pi_m}{2} \rfloor$
18:     $\Theta_c \leftarrow$ MinimumCap($\Pi_{\text{mid}}, \tau, \lceil \frac{3}{\epsilon} \rceil$)
19:     if $\Theta_c \le \Theta$ then
20:       $\Pi_{\text{lused}} \leftarrow \Pi_{\text{mid}}$
21:       $\Pi_l \leftarrow \Pi_{\text{mid}} + 1$
22:     else
23:       $\Pi_m \leftarrow \Pi_{\text{mid}} - 1$
24:     end if
25:   end while
26:   $\Pi_{\text{last}} \leftarrow \Pi_{\text{lused}}$
27:   $\Theta_{\text{last}} \leftarrow$ MinimumCap($\Pi_{\text{last}}, \tau, \lceil \frac{3}{\epsilon} \rceil$)
28:   {Test temperature for $\Pi$ found from binary search}
29:   $T_c \leftarrow \mathcal{T}(\Pi_{\text{last}}, \Theta_{\text{last}})$
30:   if $T_c < T_{\min}$
31:     $T_{\min} \leftarrow T_c$
32:     $\hat{\Theta} \leftarrow \Theta_{\text{last}}$
33:     $\hat{\Pi} \leftarrow \Pi_{\text{last}}$
34:   end if
35:   {Proceed to next iteration}
36:   $\Pi_{\text{last}} \leftarrow \Pi_{\text{last}} + 1$
37:   $\Theta_{\text{last}} \leftarrow$ MinimumCap($\Pi_{\text{last}}, \tau, \lceil \frac{3}{\epsilon} \rceil$)
38: until $\Pi_{\text{last}} > \Pi_{\text{upper}}$
39: retrun $\left(\hat{\Pi}, \hat{\Theta}\right)$
```

The pseudocode for the algorithm, SelectPeriod, is presented in Algorithm 2. The input parameters include the minimum and maximum value of the period, task system and the accuracy parameter ϵ. We use the algorithm MinimumCap, a monotonically non-decreasing capacity-determination algorithm over $\{\Pi_{\text{lower}}, \ldots, \Pi_{\text{upper}}\}$ (i.e., MinimumCap returns monotonically non-decreasing capacity for increasing periods), as subroutine of SelectPeriod. Let us define $\Theta_{\min}^{\mathcal{MC}}(\Pi)$ to be the value returned by sufficient capacity-determination algorithm MinimumCap with $k = \left\lceil \frac{3}{\epsilon} \right\rceil$ and $\Theta_{\min}^{\mathcal{OPT}}(\Pi)$ to be the optimal capacity-determination algorithm for a given thermal-aware periodic resource. The value of k has been chosen such that the peak temperature of the system will be upper bounded by the approximation ratio $1+\epsilon$.

Inside the algorithm SelectPeriod, Π_{last} tracks the value of Π to be checked for the optimal peak temperature in the current iteration. Lines 1 through 3 initialize the first choice for Π (equal to Π_{lower}) and other variables. The repeat loop of Lines 4 and 38 iterate through successive choices of Π. To find the next choice of Π, we use a binary search (Line 16) over the range of remaining values of Π that have not been selected. The binary search returns the largest Π such that the capacity is no more than $(1+\epsilon)$ times the capacity of the previously-selected value of Π. Π is incremented (Line 36) within the repeat loop to ensure the next capacity-value for Π is more than $(1+\epsilon)$ times the previous. Finally, the repeat loop terminates when Π_{last} get a value which is greater than Π_{upper}. The values $\hat{\Pi}$ and $\hat{\Theta}$ maintain the interface parameters of the resource for which the minimum temperature is observed among all values of Π that have been evaluated. The algorithm returns interface parameters with minimum peak temperature over all evaluated choices of Π.

6.3. Correctness

The algorithm SelectPeriod simply checks $\Theta_{\min}^{\mathcal{MC}}(\Pi)$ for selected values of Π between Π_{lower} and Π_{upper} and returns $(\Pi, \Theta_{\min}^{\mathcal{MC}}(\Pi))$ with the minimum temperature $\mathcal{T}(\Pi, \Theta_{\min}^{\mathcal{MC}}(\Pi))$. The values of Π are selected based on an accuracy parameter ϵ. We will show the following theorem which states that the value returned by the algorithm $\mathcal{T}(\hat{\Pi}, \Theta_{\min}^{\mathcal{MC}}(\hat{\Pi}))$ is at most $(1+\epsilon)$ times minimum obtainable peak temperature $\mathcal{T}(\Pi^*, \Theta_{\min}^{\mathcal{OPT}}(\Pi^*))$ of the system. Observe that $\hat{\Theta}$ is equivalent to $\Theta_{\min}^{\mathcal{MC}}(\hat{\Pi})$ and Θ^* is equivalent to $\Theta_{\min}^{\mathcal{OPT}}(\Pi^*)$.

Theorem 4. *Given sporadic task system τ and accuracy parameter $\epsilon : 0 < \epsilon \le 1$, the procedure* SelectPeriod(τ, Π_{lower}, Π_{upper}, ϵ) *returns* $(\hat{\Pi}, \Theta_{\min}^{\mathcal{MC}}(\hat{\Pi}))$ *such that*

$$\mathcal{T}(\hat{\Pi}, \Theta_{\min}^{\mathcal{MC}}(\hat{\Pi})) \le (1+\epsilon)\mathcal{T}(\Pi^*, \Theta_{\min}^{\mathcal{OPT}}(\Pi^*)) \tag{14}$$

In order to prove the above theorem, we require additional definitions and lemmas to be proved first. In SelectPeriod routine, we use the minimum capacity determination algorithm in a binary search over the range $\left\{\Pi_{lower}, \ldots, \Pi_{upper}\right\}$. The validity of this code block entirely depends on the condition that MinimumCap is monotonically non-decreasing on Π. The MinimumCap algorithm is invoked from SelectPeriod with accuracy parameter $\lceil (3/\epsilon) \rceil$. In the next paragraph, we present a lemma which states that MinimumCap is monotonically non-decreasing over the range $\left\{\Pi_{lower}, \ldots, \Pi_{upper}\right\}$.

Lemma 2. *For a given τ and $\epsilon \ge 0$,* MinimumCap *is monotonically, non-decreasing over* $\left\{\Pi_{lower}, \ldots, \Pi_{upper}\right\}$.

Proof. The lemma follows from the observation that values of $\Theta_{\min}$ and $\Theta_{\min}^{\ell}$ (set in Lines 1 and 8 of Algorithm 1) are monotonically, non-decreasing in Π. □

In the next lemma, we will establish an algebraic property used for deriving a simplified approximation ratio.

Lemma 3. *Given $f(x) = (1 - e^{-(1+b)x})/(1 - e^{-x})$ where $x > 0$ and $b > 0$, the following is true*

$$f(x) \le (1+b) \tag{15}$$

Proof. For proving the above lemma, we first show the limits of the function at values of x approaching 0 and ∞.

$$\begin{aligned} \lim_{x\to 0} f(x) &= \lim_{x\to 0} \frac{1 - e^{-(1+b)x}}{1 - e^{-x}} \\ &= \frac{\lim_{x\to 0}(1+b)e^{-(1+b)x}}{\lim_{x\to 0} e^{-x}} \\ &\quad \text{(using L' Hospital' s rule)} \\ &\le (1+b) \end{aligned}$$

and

$$\lim_{x\to\infty} f(x) = \lim_{x\to\infty} \frac{1 - e^{-(1+b)x}}{1 - e^{-x}} = 1$$

We now observe that between the two extreme values of x, the function $f(x)$ is continuously decreasing. To show this, we must prove that the slope (first derivative $f'(x)$) is negative for $x > 0$ where

$$f'(x) = \frac{(1+b)e^{-(1+b)x}(1 - e^{-x}) - e^{-x}(1 - e^{-(1+b)x})}{(1 - e^{-x})^2}$$

For the sake of contradiction, assume that $f'(x) \geq 0$. We can write

$$0 \leq \frac{(1+b)e^{-(1+b)x}(1-e^{-x}) - e^{-x}(1-e^{-(1+b)x})}{(1-e^{-x})^2}$$
$$\Rightarrow 0 \leq (1+b)e^{-(1+b)x}(1-e^{-x}) - e^{-x}(1-e^{-(1+b)x})$$
$$\Rightarrow 0 \leq (1+b)(1-e^{-x}) - e^{bx}(1-e^{-(1+b)x})$$
$$\Rightarrow 0 \leq (1+b) - (1+b)e^{-x} - e^{bx} + e^{-x}$$
$$\Rightarrow 0 \leq (1+b) - be^{-x} - e^{bx}$$
$$\Rightarrow (1+b) \geq \frac{b}{e^x} + e^{bx}$$
$$(\text{Assuming } g(x) \overset{\text{def}}{=} \tfrac{b}{e^x} + e^{bx})$$
$$\Rightarrow (1+b) \geq g(x) \tag{16}$$

The value of $g(x)$ is $(1+b)$ at $x=0$. However, its value is continuously increasing for all $x>0$ as $g'(x) = b(e^{bx} - (1/e^x)) > 0$ when $x>0$ and $b>0$. Therefore, Eq. (16) is a contradiction, which proves $f(x)$ is decreasing between 0 and ∞. Since $\lim_{x\to 0} f(x) \leq (1+b)$, the term $(1+b)$ is an upper bound for all $x>0$, which implies the lemma. □

We now establish a lower bound on the peak temperature for a specific domain from $\{\Pi_{lower}, \ldots, \Pi_{upper}\}$. If we select a subset of period values from a domain (e.g., $\{\Pi_i, \ldots, \Pi_j\} \subseteq \{\Pi_{lower}, \ldots, \Pi_{upper}\}$), then we may calculate a lower bound on the minimum peak temperature obtainable from any period in this subset by taking largest Π (i.e., Π_j) and using the capacity associated with minimum Π (i.e., $\Theta^{\mathcal{MC}}_{\min}(\Pi_i)$) in the peak temperature calculation.

Lemma 4. *Consider any $\Pi_i, \Pi_j \in \{\Pi_{lower}, \ldots, \Pi_{upper}\}$ where $\Pi_i \leq \Pi_j$; Given any monotonically non-decreasing capacity determination algorithm $\mathcal{A}$, the following is true*

$$\mathcal{T}(\Pi_j, \Theta^{\mathcal{A}}_{\min}(\Pi_i)) \leq \min_{\Pi \in \{\Pi_i,\ldots,\Pi_j\}} \mathcal{T}(\Pi, \Theta^{\mathcal{A}}_{\min}(\Pi)) \tag{16}$$

Proof. Observe that the first partial derivative of $\mathcal{T}(\Pi, \Theta)$ (as defined in Corollary 1) with respect to Π equals

$$\frac{\varphi^{\gamma}\beta^2 e^{-\beta\Pi}[(1-r^{\gamma}_{\text{off}})e^{-\beta(\Theta+\delta)} - (1-r^{\gamma}_{\text{off}})]}{[\beta\left(1-e^{-\beta\Pi}\right)]^2}.$$

Since $(1-r^{\gamma}_{\text{off}}) > 0$ and $0 < e^{-\beta(\Theta+\delta)} \leq 1$ for all Θ and δ greater than zero, $\frac{\partial\mathcal{T}(\Pi,\Theta)}{\partial\Pi} \leq 0$ for any $\Theta \geq 0$ and $\delta \geq 0$. Thus, for any non-negative Θ and δ,

$$\min_{\Pi \in \{\Pi_i,\ldots,\Pi_j\}} \{\mathcal{T}(\Pi_j, \Theta^{\mathcal{A}}_{\min}(\Pi))\} \leq \min_{\Pi \in \{\Pi_i,\ldots,\Pi_j\}} \{\mathcal{T}(\Pi, \Theta^{\mathcal{A}}_{\min}(\Pi))\}. \tag{17}$$

The first partial derivative of $\mathcal{T}(\Pi, \Theta)$ with respect to Θ is

$$\frac{\varphi^{\gamma}e^{-\beta(\Theta+\delta)}}{1-e^{-\beta\Pi}}(1-r^{\gamma}_{\text{off}}).$$

The above expression is positive for all non-negative Π, Θ, and δ. Because $\Theta^{\mathcal{A}}_{\min}(\cdot)$ is monotonically non-decreasing with respect to Π and $\mathcal{T}(\Pi, \Theta)$ is monotonically non-decreasing with respect to Θ,

$$\mathcal{T}(\Pi_j, \Theta^{\mathcal{A}}_{\min}(\Pi_i)) \leq \min_{\Pi \in \{\Pi_i,\ldots,\Pi_j\}} \{\mathcal{T}(\Pi_j, \Theta^{\mathcal{A}}_{\min}(\Pi))\}. \tag{18}$$

Combining Eqs. (17) and (18) proves the lemma. □

Let $\{\Pi_1, \ldots, \Pi_m\}$ be the set of different values that Π_{last} held throughout execution of SELECTPERIOD(τ, Π_{lower}, Π_{upper}, ϵ) (set in Lines 26 and 36). Consider adjacent values $\Pi_i, \Pi_{i+1} \in \{\Pi_1, \ldots, \Pi_m\}$. The final lemma for the proof of correctness obtains a relation between the calculated capacities of the successive values of Π_{last}.

Lemma 5. *For any adjacent Π_i, $\Pi_{i+1} \in \{\Pi_1, \ldots, \Pi_m\}$, either $\Pi_{i+1} = \Pi_i + 1$ or*

$$\Theta^{\mathcal{MC}}_{\min}(\Pi_{i+1}) \leq \left(1 + \frac{\epsilon}{3}\right)\Theta^{\mathcal{MC}}_{\min}(\Pi_i) \tag{19}$$

Proof. Let $\Theta^{\mathcal{MC}}_{\min}(\Pi_i)$ be the value of Θ_{last} at the beginning of the "repeat" loop on Line 4 in Algorithm 2. The lemma follows immediately by observing that the loop of Lines 16–25 searches for the maximum value of Θ such that $\Theta \leq (1+\epsilon/3)\Theta_{\text{last}}$. Then, in Line 36, the next value of Π is set to be $\Pi_{\text{last}} + 1$. □

Using the above lemmas, we now prove Theorem 4 which states that SelectPeriod returns a valid value based on the approximation ratio and task system provided as arguments.

Proof of Theorem 4. For proving the theorem, we need to consider two cases.

1. $\Pi^* \in \{\Pi_1, \ldots, \Pi_m\}$ or
2. $\Pi^* \notin \{\Pi_1, \ldots, \Pi_m\}$

For the Case 1, it is obvious that $\min_{\Pi \in \{\Pi_i,\ldots,\Pi_j\}} \mathcal{T}(\Pi, \Theta^{\mathcal{MC}}_{\min}(\Pi))$ equals the $\mathcal{T}(\Pi^*, \Theta^{\mathcal{OPT}}_{\min}(\Pi^*))$; thus, Theorem 4 follows for this case.

For Case 2, there must exist adjacent values $\Pi_j, \Pi_{j+1} \in \{\Pi_1, \ldots, \Pi_m\}$ such that $\Pi_j \leq \Pi^* \leq \Pi_{j+1}$. Furthermore, $\Pi_{j+1} \neq \Pi_j + 1$ otherwise, Π^* would equal either Π_j or Π_{j+1}, and Case 1 would apply.

Since the SelectPeriod algorithm returns $(\hat{\Pi}, \Theta^{\mathcal{MC}}_{\min}(\hat{\Pi}))$, the following must be true:

$$\mathcal{T}(\hat{\Pi}, \Theta^{\mathcal{MC}}_{\min}(\hat{\Pi})) \leq \mathcal{T}(\Pi_{j+1}, \Theta^{\mathcal{MC}}_{\min}(\Pi_{j+1})) \tag{20}$$

As $\Pi^* \in \{\Pi_j \ldots \Pi_{j+1}\}$ and the optimal capacity algorithm is also monotonically non-decreasing on Π, we can write using Lemma 4:

$$\mathcal{T}(\Pi_{j+1}, \Theta^{\mathcal{OPT}}_{\min}(\Pi_j)) \leq \mathcal{T}(\Pi^*, \Theta^{\mathcal{OPT}}_{\min}(\Pi^*))$$

Now using Corollary 1 and above two equations, we can obtain the following equations. For notational brevity, we will denote $\Theta^{\mathcal{MC}}_{\min}(\Pi_i)$ by Θ_i and $\Theta^{\mathcal{OPT}}_{\min}(\Pi_i)$ by Θ^*_i.

$$\begin{aligned}
&\frac{\mathcal{T}(\hat{\Pi}, \Theta^{\mathcal{MC}}_{\min}(\hat{\Pi}))}{\mathcal{T}(\Pi^*, \Theta^{\mathcal{OPT}}_{\min}(\Pi^*))} \\
&\leq \frac{\mathcal{T}(\Pi_{j+1}, \Theta^{\mathcal{MC}}_{\min}(\Pi_{j+1}))}{\mathcal{T}(\Pi_{j+1}, \Theta^{\mathcal{OPT}}_{\min}(\Pi_j))} \\
&= \frac{1-(1-r^{\gamma}_{\text{off}})e^{-\beta(\Theta_{j+1}+\delta)} - r^{\gamma}_{\text{off}}e^{-\beta\Pi_{j+1}}}{1-(1-r^{\gamma}_{\text{off}})e^{-\beta(\Theta^*_j+\delta)} - r^{\gamma}_{\text{off}}e^{-\beta\Pi_{j+1}}} \\
&= \frac{1-e^{-\beta(\Theta_{j+1}+\delta)} + r^{\gamma}_{\text{off}}(e^{-\beta(\Theta_{j+1}+\delta)} - e^{-\beta\Pi_{j+1}})}{1-e^{-\beta(\Theta^*_j+\delta)} + r^{\gamma}_{\text{off}}(e^{-\beta(\Theta^*_j+\delta)} - e^{-\beta\Pi_{j+1}})} \\
&\leq \frac{1-e^{-\beta(\Theta_j(1+\epsilon/3)+\delta)} + r^{\gamma}_{\text{off}}(e^{-\beta(\Theta^*_j+\delta)} - e^{-\beta\Pi_{j+1}})}{1-e^{-\beta(\Theta^*_j+\delta)} + r^{\gamma}_{\text{off}}(e^{-\beta(\Theta^*_j+\delta)} - e^{-\beta\Pi_{j+1}})} \\
&\leq \frac{1-e^{-\beta(\Theta^*_j(1+\epsilon/3)^2+\delta)} + r^{\gamma}_{\text{off}}(e^{-\beta(\Theta^*_j+\delta)} - e^{-\beta\Pi_{j+1}})}{1-e^{-\beta(\Theta^*_j+\delta)} + r^{\gamma}_{\text{off}}(e^{-\beta(\Theta^*_j+\delta)} - e^{-\beta\Pi_{j+1}})} \\
&\leq \frac{1-e^{-\beta(1+\epsilon)(\Theta^*_j+\delta)} + r^{\gamma}_{\text{off}}(e^{-\beta(\Theta^*_j+\delta)} - e^{-\beta\Pi_{j+1}})}{1-e^{-\beta(\Theta^*_j+\delta)} + r^{\gamma}_{\text{off}}(e^{-\beta(\Theta^*_j+\delta)} - e^{-\beta\Pi_{j+1}})}
\end{aligned} \tag{21}$$

The last inequality follows Lemma 5 and the fact that $e^{-\beta x}$ is decreasing in x and $(1+(\epsilon/3))^2 \leq (1+\epsilon)$.

Finally, observe for any $x \geq y > 0, z \geq 0$, and $((x+z)/(y+z)) \leq (x/y)$. This observation combined with Eq. (21) results in

$$\frac{\mathcal{T}(\hat{\Pi}, \Theta_{\min}^{\mathcal{MC}}(\hat{\Pi}))}{\mathcal{T}(\Pi^*, \Theta_{\min}^{\mathcal{MC}}(\Pi^*))}$$
$$\leq \frac{1-e^{-\beta(1+\epsilon)(\Theta_j+\delta)}}{1-e^{-\beta(\Theta_j+\delta)}}$$
$$\leq (1+\epsilon)(\text{using Eq. 15}).$$

The last inequality implies the theorem. □

The algorithm SelectPeriod works under the assumption of integer values for the period-of-repetition parameter (subject to the system tick granularity). However, we allow MinimumCap to return a fine-grained capacity in real numbers as capacity is smaller compared to the coarse grained period-of-repetition. For the devices that can only work with integer value for capacity along with period-of-repetition, SelectPeriod can be modified to add a line at the end for taking the ceiling of the returned capacity (i.e., $\lceil\hat{\Theta}\rceil$). In that case, the returned capacity may increase by at most by one from the $(1+(\epsilon/3))$ times optimal capacity. The resulting peak-temperature approximation ratio from this modification may be shown to be at most $(2+\epsilon)$.

6.4. Complexity

We now address the time complexity of SelectPeriod.

Theorem 5. SelectPeriod$(\tau, \Pi_{lower}, \Pi_{upper}, \epsilon)$ *has time complexity equal to*

$$O\left(\frac{nlgn}{\epsilon}\cdot lg\left(\frac{\Theta_{upper}}{\Theta_{lower}}\right)\cdot lg(\Pi_{upper})/\epsilon\right) \tag{22}$$

where $\Theta_{lower} = \Theta_{\min}^{\mathcal{MC}}(\Pi_{lower})$ *and* $\Theta_{upper} = \Theta_{\min}^{\mathcal{MC}}(\Pi_{upper})$.

Proof. For determining the complexity of SelectPeriod$(\Pi_{lower}, \Pi_{upper}, \epsilon)$, observe that the running time is dominated by the *repeat . . . until* loop in Lines 4 through 38. The complexity of this loop can (informally) be determined by

Number of iterations of while loop
×Number of values to be checked in binary search
×Execution time to check value of Π

The number of iterations of the *repeat . . . until* loop can be determined by observing that Θ_{last} increases by at least a factor of $(1+(\epsilon/3))$ upon every iteration of the *while* loop from Lines 16–25. Thus, the number of iterations is equal to the smallest integer value of ℓ such that the following equation is true:

$$\left(1+\frac{\epsilon}{3}\right)^{\ell}\Theta_{lower} \geq \Theta_{upper} \tag{23}$$

Solving for ℓ,

$$\begin{aligned}\ell &= \left\lceil \log_{(1+(\epsilon/3))}\left(\frac{\Theta_{upper}}{\Theta_{lower}}\right)\right\rceil \\ &= \left\lceil \frac{lg(\Theta_{upper}/\Theta_{lower})}{lg\left(1+(\epsilon/3)\right)}\right\rceil \\ &\leq \left\lceil \frac{(1+(\epsilon/3))lg(\Theta_{upper}/\Theta_{lower})}{\frac{\epsilon}{3}}\right\rceil \in O\left(lg\left(\frac{\Theta_{upper}}{\Theta_{lower}}\right)/\epsilon\right)\end{aligned} \tag{24}$$

The third inequality above follows from the well-known identity $x/(1+x) \leq \lg(1+x)$ for all $x > -1$.

The binary search of Lines 16–25 searches over the range $\{\Pi_{last}, \ldots, \Pi_{upper}\}$. This range contains at most Π_{upper} number of elements. Thus, the number of values of $\Pi \in \{\Pi_{last}, \ldots, \Pi_{upper}\}$ that needs to be evaluated for $\Theta_{\min}^{\mathcal{MC}}(\Pi)$ is

$$\in O(lg\left(\Pi_{upper}\right)) \tag{25}$$

Finally, the execution time for each calculation of $\Theta_{\min}^{\mathcal{MC}}(\Pi)$ is equal to $O(n\lg n)$ from Theorem 3. Combining this observation with Eqs. (23)–(25), implies the running time given in the lemma. □

The value of Θ_{lower} in Eq. (22) may be arbitrarily small; therefore, it may seem that SelectPeriod can have very large running time. However, we show a lower bound for Θ_{lower}. In a related publication, Fisher [27] determined the lower bound on system utilization, as given in the following theorem.

Lemma 6 (from [27]). *For any sporadic task system τ with positive integer parameters,*

$$U(\tau) \geq \frac{n}{p_{\max}} \tag{26}$$

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

The algorithm MinimumCap ensures that $\Theta_{lower} \geq U(\tau)\Pi_{lower}$, which with Lemma 6 implies:

$$\Theta_{lower} \geq \frac{n\Pi_{lower}}{p_{\max}} \tag{27}$$

Finally, we show that the parameter selection algorithm has polynomial-time complexity in the representation of τ, the range $\{\Pi_{upper}, \ldots, \Pi_{upper}\}$, and $1/\epsilon$.

Corollary 2. SelectPeriod$(\tau, \Pi_{lower}, \Pi_{upper}, \epsilon)$ *has time complexity equal to*

$$O(nlgn(lg^2\Pi_{upper} + lg\Pi_{upper}lg(p_{max}))/\epsilon^2) \tag{28}$$

Proof. The corollary follows from the fact that the second term $(lg(\Theta_{upper}/\Theta_{lower}))$ in Eq. (22) is upper bounded by $(\lg((\Pi_{upper})/(\Pi_{lower}\times(n/p_{max}))))$ according to Lemma 6, Eq. (27) and $\Theta_{upper} \leq \Pi_{upper}$ (otherwise, we cannot guarantee τ's schedulability); this term is $O(\lg(\Pi_{upper}) + \lg(p_{max}))$. Substituting above terms into Eq. (22) directly implies the claim of this corollary. □

7. Simulations

This section presents simulation results for the proposed algorithms. We have performed simulation over various parameters. During simulations, we have the following simulation parameters and value ranges:

1. The number of tasks in the system τ is 2, 4, 8, 16, or 32.
2. The system utilization $U(\tau)$ is taken from the range [0.55, 0.9] at 0.025-increments and each individual task utilizations u_i is generated using uniform distribution. The UUniFast algorithm [36] generates taskset using an uniform distribution for a given total utilization.
3. Each sporadic task τ_i has an integer period p_i uniformly drawn from $\{4, \ldots, 8\}$.
4. The valid range of periods-of-repetition are integers from $\left\{2, \ldots, \text{lcm}_{\tau_i\in\tau}\{p_i\}\right\}$.
5. The value of the accuracy parameter is set to 0.05, 0.15, 0.15, and 0.25. The parameters for the processor is set to $\varphi = 1$, $r_{off} = 0.10$, $\beta = 0.228$, $\delta = 0.10$ and $\gamma = 3$.

7.1. Difference in peak temperature

Each point in the following plots represents the mean of 200 simulation results. We show the result for $n = 8$, $\epsilon = 0.15$ only as the results for the rest of the values follow the same pattern in the

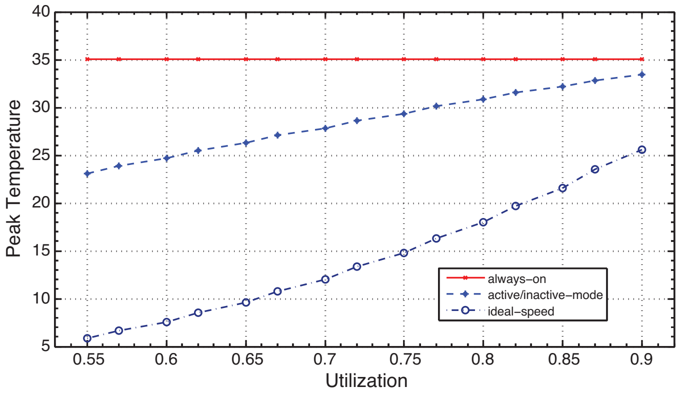

Fig. 5. Utilization vs. Temperature ($n = 8$ and $\epsilon = 0.15$).

output graphs. Fig. 5 shows the peak temperature in three different cases. The solid line shows the asymptotic peak temperature if the processor is operating always in the active-mode, whereas the dashed line shows for optimal *ideal*-speed. In the simulation figures, we refer to the results from SelectPeriod for thermal-aware periodic resource parameters as "active/inactive mode". The peak temperature using for active/inactive mode will lie between these two extremities, which is shown by the dashed line. The temperature generated from the ideal speed is better for all utilization. However, we emphasize that such a temperature is a lower-bound since even modern high-performance DVFS processors may not be able to support an arbitrary ideal speed.

In Fig. 6, we compare the performance of sufficient solution SelectPeriod for finding the peak temperature. This plot defines the error in percentage with respect to the optimal-obtainable peak temperature for thermal-aware periodic resources. For this graph, we define error as $error(T) = (T - \mathcal{T}_{\text{opt}})/(\mathcal{T}_{\text{opt}}) \times 100$ where $\mathcal{T}_{\text{opt}}$ is the optimal minimum temperature obtained from $\mathcal{T}(\Pi^*, \Theta^*)$ for each generated task system. In the graph, the dotted-line curve represents relative error for approximate peak temperature, while the solid-line curve represents relative error if it is running in *active*-mode all the time. For SelectPeriod, the mean relative error is less than 2%, whereas for the active-always approach ranges from 5% to 45%. The results for 95% confidence intervals are shown in the figure. This result establishes the near optimality of our algorithm for the thermal-aware periodic resource setting.

7.2. Comparison with sleep-when-idle

We are not aware of any standard proactive scheduling algorithm on a discrete-speed processor for sporadic task systems which takes thermal constraints and schedulability into account. As there is no existing solution to the problem specified, we compare

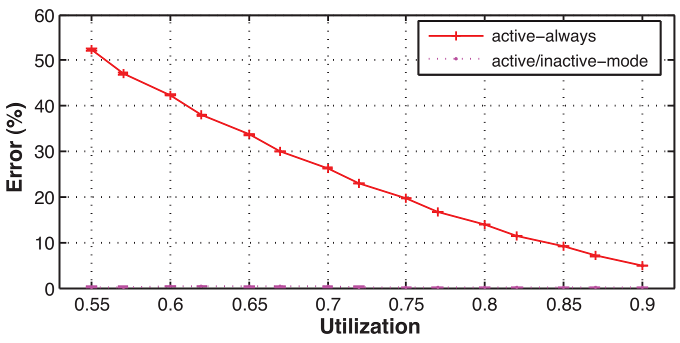

Fig. 6. Error in Peak Temperature (Respect to Optimal capacity) ($n = 8$; $\epsilon = 0.15$;).

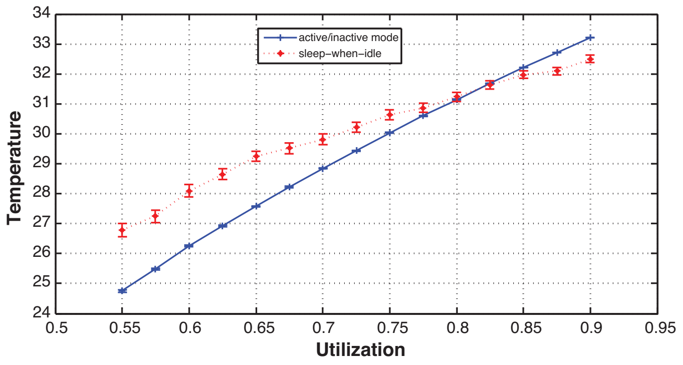

Fig. 7. Comparison between sleep-when-idle and our active/inactive algorithm (n=8; e=0.15;).

the results of our algorithm to a very basic greedy, non-real-time algorithm *sleep-when-idle* which simply turns the processor to a low power state when the system is idle. Please observe there is no schedulability analysis associated with the sleep-when-idle algorithm when $\delta > 0$; thus, even when there is a small transition overhead between active and inactive states, it is possible that the sleep-when-idle may miss deadlines. Our approach always guarantees deadlines are met. Furthermore, there is currently no theoretical method for calculating the peak temperature of the sleep-when-idle method except via empirical analysis. Thus, we measure the peak temperature of the sleep-when-idle by taking the maximum observed temperature up to 10,000 clock ticks. Fig. 7 shows the difference in peak temperature calculated of our algorithm compared to the observed temperature of the sleep-when-idle algorithm. For Fig. 7, we take the mean (showing also 95% confidence interval) of 1000 samples. For this simulation, we assume that there is no transition overhead (i.e., $\delta = 0$). The solid line represents the temperature calculated using our proposed model whereas the dotted line represents the temperature for the sleep-when-idle algorithm. Our model ensures a lower peak temperature than sleep-when-idle except for very high utilization (0.85 or greater).

The main conclusion that we observe from the results of Fig. 7 (other simulations we executed gave similar results) is that despite several advantages given to sleep-when-idle (i.e., no schedulability requirement, no transition overhead, and using observed temperature) our algorithm performs competitively. The lack of transition overhead in the simulation is an advantage for sleep-when-idle algorithm as it may quickly take advantage of cooling during idle times without penalty. Furthermore, by comparing observed temperature of sleep-when-idle with calculated temperature of our algorithm, we are comparing an empirical observation with a worst-case upper-bound that may not actually occur. However, sleep-when-idle cannot capitalize on these advantages since it never gets ample time between successive jobs to radiate enough heat for lowering the instantaneous temperature, whereas our active/inactive approach accumulates idle cycles at the beginning of each period-of-repetition and the proactive scheduler uses this periodic interval for cooling the processor.

7.3. Difference in runtime complexity

Fig. 8 shows the difference in run-time complexity for determining the optimal peak temperature with that of the approximate one (using the same randomly-generated systems as shown in Figs. 5 and 6). The solid-line curve in the graph represents number of testing-set points required by the optimal algorithm and the dotted-line curve represents the number of testing set points required by the approximate solution (i.e., invoking SelectPeriod

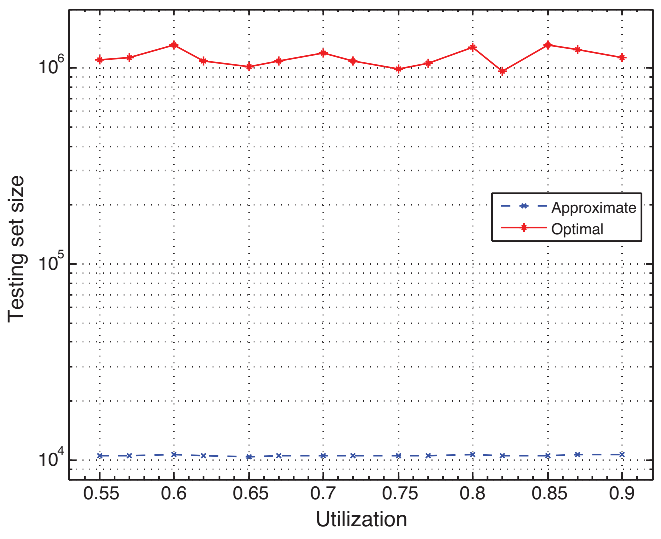

Fig. 8. Complexity difference($n = 8$, $\epsilon = 0.15$).

with $0 < \epsilon \le 1$). The results suggest a significant performance improvement of the approximate algorithm over the exact algorithm. For finding the optimal solution, the exact algorithm needs to check Eq. (8) for each testing set point up to $\mathcal{P}(\tau)$ for each task system. The approximation algorithm only needs to check schedulability for a polynomial number of testing set points. From Figs. 6 and 8, we may observe that the approximation algorithm achieves a reduction in running time equal to two orders of magnitude (compared with the exact algorithm) with only a 2% accuracy loss in terms of peak temperature.

8. Conclusion

This research has analyzed temperature sensitivity with respect to real-time workload executing upon a processor supporting only basic power management features, as might be found in an embedded system. Previous research in real-time thermal-aware scheduling has primarily focused upon more sophisticated power-management features such as DVFS. For modeling basic processors, we proposed the thermal-aware periodic resource model and developed methods for quantifying peak temperature in this model. Two polynomial-time approximation algorithms, MinimumCap and SelectPeriod, are proposed for minimizing the peak temperature for such a processor with only active/inactive modes. Furthermore, our simulation results have established that the algorithms are effective at reducing the relative error over synthetically generated tasks while maintaining a low runtime complexity. Our techniques ensure that real-time deadlines may be met while ensuring the peak temperature of the system is bounded and minimized. Ultimately, these techniques may be used to reduce the packaging costs due to cooling and increase the system lifetime and reliability.

Our research has leveraged previously-proposed techniques for compositional real-time systems (i.e., periodic resources [6,26,27]) which naturally models the behavior of a processing platform with active/inactive states. However, if more than two discrete modes are available, further research is required to determine whether techniques for more general compositional models (e.g., real-time calculus [37]) are necessary to analyze the system, or if using only two of the available modes suffice to minimize peak temperature. Additionally, for ensuring the strict timing constraints, hard-real-time systems require the worst case execution-time estimation which might be overly pessimistic; reclamation of idle times if jobs complete earlier than their worst-case estimate may further reduce the instantaneous temperature of the system. Thus, an interesting research direction is developing online reclamation techniques for reducing the actual peak temperature at runtime, without affecting the ability to obtain worst-case upper bounds on the peak temperature and guarantee hard deadlines. Considering such issues, we are planning to extend our peak-temperature minimization research for more than two power modes on both uniprocessor and multiprocessor systems. We are also exploring online idle-time reclamation techniques for minimizing observed temperature and are building a hardware test bench to validate the efficacy of our proposed thermal-aware techniques.

Acknowledgements

We are sincerely thankful to the reviewers for the detailed comments, corrections, and insightful directions which helped us to improve the quality of the article. This research has been supported in part by NSF CAREER Grants (Nos. CNS-0953585 and CNS-0746906), a grant from Wayne State University's Office of Vice President of Research, and a Rackham Faculty Research Grant at the University of Michigan.

Appendix A. Correctness of MinimumCap

To prove the correctness of MinimumCap, we will establish the fact using the following theorems that the value returned by the algorithm (i.e., $\Theta^{\min}$) is at least the optimal minimum capacity value $\Theta^*(\Pi, \tau)$. More formally in order to prove the validity of our algorithm, all we have to do is to establish the following theorem.

Theorem 6. *For all $k \in \mathbb{N}^+ \cup \{\infty\}$, MinimumCap returns $\Theta^{\min} \ge \Theta^*(\Pi, \tau)$. Furthermore, if $k = \infty$, $\Theta^{\min} = \Theta^*(\Pi, \tau)$.*

Before proving the above theorem, we need to establish some additional statements and also define some additional terms. We first prove some lemmas which we will use to prove the above theorem at the end. Fisher and Dewan [26] have defined minimum capacity Θ that is required for sbf to upper-bound a half-line $\langle (t, D_t), \alpha \rangle$.

Definition 3 *(Minimum Capacity for $\langle (t, D_t), \alpha \rangle$).*

$$\Theta^*(\Pi, \langle (t, D_t), \alpha \rangle) \overset{\text{def}}{=} \inf\{\Theta \in \mathbb{R}^+ | (\forall x \ge t : \alpha(x - t) + D_t \le \mathrm{sbf}((\Pi, \Theta), t))\}. \quad (A.1)$$

Fisher and Dewan [26] defined another term ℓ-feasibility as it is difficult to calculate $\Theta^*(\Pi, \tau)$ in a straight forward linear way because of difficulty in proving for the discontinuity in the step function. They break the region below the sbf function into overlapping subregions with the name ℓ-feasibility region. An ℓ-feasibility region is the area below the ℓ^{th} "step" of the sbf function extending downward and infinitely to the right. Fig. 3 gives a visual depiction; the definition below formalizes the ℓ-feasibility region concept.

Definition 4 *(ℓ-Feasibility Region).*

$$\mathcal{F}_\ell(\Pi, \Theta) \overset{\text{def}}{=} \left\{ \langle (t, D_t), \alpha \rangle \in \mathbb{R}^2_{\ge 0} \times \mathbb{R}_{\ge 0} \;\middle|\; \begin{array}{l} (0 \le \alpha \le \frac{\Theta}{\Pi}) \\ \wedge \;\; (\Theta \ge \frac{D_t - t + \ell\Pi}{\ell}) \\ \wedge \;\; (\Theta \ge \frac{D_t}{\ell}) \\ \wedge \;\; \left(\Theta \ge \frac{D_t + \alpha((\ell+1)\Pi - t)}{\ell + \alpha}\right) \end{array} \right\}. \quad (A.2)$$

The next defined function determines the minimum capacity for any given half-line $\langle (t, D_t), \alpha \rangle$ to be an element of the ℓ-feasibility region.

Definition 5 *(ℓ-Minimum Capacity for $\langle(t, D_t), \alpha\rangle$).*

$$\Theta^*_\ell(\Pi, \langle(t, D_t), \alpha\rangle) \overset{\text{def}}{=} \inf\{\Theta \in \mathbb{R}^+ | \left\langle(t, D_t), \alpha\right\rangle \in \mathcal{F}_\ell(\Pi, \Theta)\}. \quad (A.3)$$

Fisher and Dewan [26] proved that for any half-line $\langle(t, D_t), \alpha\rangle$ is always below the sbf, if and only if $\langle(t, D_t), \alpha\rangle$ is contained in some ℓ-feasibility region with $\ell > 0$.

Lemma 7. *For any $\langle(t, D_t), \alpha\rangle \in \mathbb{R}^2_{\geq 0} \times \mathbb{R}_{\geq 0}$, $\alpha(t' - t) + D_t \leq \text{sbf}((\Pi, \Theta), t') \forall t' \geq t$ if and only if there exists $\ell \in \mathbb{N}$ such that $\langle(t, D_t), \alpha\rangle \in \mathcal{F}_\ell(\Pi, \Theta)$.*

Furthermore, Fisher and Dewan [26] showed that $\Theta^*_\ell(\Pi, \langle(t, D_t), \alpha\rangle)$ can be efficiently determined by taking the maximum of four values based on the lines that form the boundaries of the ℓ-feasibility region.

Lemma 8. *For any $\ell \in \mathbb{N}^+$,*

$$\Theta^*_\ell(\Pi, \langle(t, D_t), \alpha\rangle) = \max \left\{ \begin{array}{r} \alpha\Pi, \\ \dfrac{D_t - t + \ell\Pi}{\ell}, \\ \dfrac{D_t}{\ell}, \\ \dfrac{D_t + \alpha((\ell+1)\Pi - t)}{\ell + \alpha} \end{array} \right\}. \quad (A.4)$$

The following results are specific to thermal-aware periodic resources. Lemmas 9 and 10 and Corollary 3 will establish that for any half-line $\langle(t, D_t), \alpha\rangle$, we will need to search for only two feasibility regions. These two regions can be obtained from considering t only. The intuition behind this lemma is that there is only one new step in each execution cycle which overlaps immediately with the step from previous cycle and also with the next cycle. Therefore, in any cycle we need to consider only two steps.

Lemma 9. *Given $q = \lceil(t/\Pi)\rceil$, $\Pi \geq 0$, $l < q - 1$, $\Theta \geq 0$, then $\langle(t, D_t), \alpha\rangle \in \mathcal{F}_\ell(\Pi, \Theta) \Rightarrow \langle(t, D_t), \alpha\rangle \in \mathcal{F}_{q-1}(\Pi, \Theta)$*

Proof. Assume that $\langle(t, D_t), \alpha\rangle \in \mathcal{F}_\ell(\Pi, \Theta)$. We must show that all four conditions of Eq. (A.2) are satisfied for $\mathcal{F}_{q-1}$. First note that $\langle(t, D_t), \alpha\rangle \in \mathcal{F}_\ell(\Pi, \Theta)$ implies that $\alpha \leq (\Theta/\Pi)$; thus, the first condition of Eq. (4) is trivially satisfied for $\mathcal{F}_{q-1}$. By the third condition of Eq. (A.2) for $\mathcal{F}_\ell$, $D_t \leq \ell\Theta$. Since $q - 1 > \ell$, it must also be that $D_t \leq (q-1)\Theta$ (which satisfies third condition for $\mathcal{F}_\ell$). Similarly, the fourth condition of Eq. (A.2) for $\mathcal{F}_\ell$ implies that $\ell(\Theta - \alpha\Pi) \geq D_t - \alpha t - \Theta\alpha$. Since $\Theta \geq \alpha\Pi$ and $\ell < q - 1$, $(q-1)(\Theta - \alpha\Pi) \geq D_t - \alpha t - \Theta\alpha$; thus the fourth condition for $\mathcal{F}_{q-1}$ is also satisfied.

In the previous paragraph, we have shown that $(q-1)\Theta \geq D_t$. From the definition of q, we know that

$$\begin{aligned} & t \geq (q-1)\Pi \\ & \Rightarrow t - (q-1)\Pi \geq 0 \end{aligned}$$

From the above two inequalities, we can write

$$\begin{aligned} & \Theta(q-1) + t - (q-1)\Pi \geq D_t \\ & \Rightarrow \Theta \geq \frac{D_t - t + (q-1)\Pi}{(q-1)} \end{aligned}$$

The above derivation implies the second condition for $\mathcal{F}_\ell$ which proves the lemma. With this, all four conditions of Definition 4 are shown true in case of $\mathcal{F}_{q-1}(\Pi, \Theta)$ for all value of $\ell < q - 1$. So $\langle(t, D_t), \alpha\rangle \in \mathcal{F}_\ell(\Pi, \Theta) \Rightarrow \langle(t, D_t), \alpha\rangle \in \mathcal{F}_{q-1}(\Pi, \Theta)$ holds true for all value of $\ell < q - 1$. □

Lemma 10. *Given $q = \lceil(t/\Pi)\rceil$, $\Pi \geq 0$, $l > q$, $\Theta \geq 0$,* then *$\langle(t, D_t), \alpha\rangle \in \mathcal{F}_\ell(\Pi, \Theta) \Rightarrow \langle(t, D_t), \alpha\rangle \in \mathcal{F}_q(\Pi, \Theta)$*

Proof. We prove the above lemma by showing its contrapositive is true – that means we will show that $\langle(t, D_t), \alpha\rangle \notin \mathcal{F}_q(\Pi, \Theta) \Rightarrow \langle(t, D_t), \alpha\rangle \notin \mathcal{F}_\ell(\Pi, \Theta)$. Assume that $\langle(t, D_t), \alpha\rangle \notin \mathcal{F}_q(\Pi, \Theta)$. This implies that at least one of the following four conditions of Definition 4 is true:

$$\begin{array}{ll} \Theta < \alpha\Pi & (A.5a) \\ \Theta < \dfrac{D_t - t + q\Pi}{q} & (A.5b) \\ \Theta < \dfrac{D_t}{q} & (A.5c) \\ \Theta < \dfrac{D_t + \alpha((q+1)\Pi - t)}{q + \alpha} & (A.5d) \end{array} \quad (A.5)$$

We will show that each of the above inequality implies $\langle(t, D_t), \alpha\rangle \notin \mathcal{F}_\ell(\Pi, \Theta)$ for any ℓ. As the first condition is not depending on q, this condition will be trivially true for any value of ℓ. We must establish the above theorem for last three conditions. If (A.5b) is true, then

$$\begin{aligned} & D_t > t - q\Pi + \Theta q \\ & \Rightarrow D_t > t - q(\Pi - \Theta) \\ & \Rightarrow D_t > t - l(\Pi - \Theta) \,\text{As}\, l > q \\ & \Rightarrow \Theta < \frac{D_t - t + l\Pi}{l} \end{aligned}$$

Thus, if Eq. (A.5b) is true, then $\langle(t, D_t), \alpha\rangle \notin \mathcal{F}_\ell(\Pi, \Theta)$.

If Eq. (A.5c) is true, then consider the following expression:

$$\begin{array}{ll} t - \ell(\Pi - \Theta) & \\ \leq q\Pi - \ell(\Pi - \Theta) & (\text{by supposition on}\, t) \\ \leq \Pi(q - \ell) + \ell\Theta & \\ \leq \ell\Theta & (\text{since}\, l > q) \\ < D_t & (\text{by Eq.}\, A.5c) \end{array}$$

Thus, $\Theta < ((D_t - t + l\Pi)/l)$ which implies $\langle(t, D_t), \alpha\rangle \notin \mathcal{F}_\ell(\Pi, \Theta)$.

Finally, if Eq. (A.5d) is true, then

$$\begin{aligned} & q\Theta < D_t + \alpha((q+1)\Pi - \Theta - t) \\ & \Rightarrow q\Theta < D_t + ((q+1)\Pi - \Theta - t) \\ & \Rightarrow (q+1)\Theta < D_t + ((q+1)\Pi - t) \\ & \Rightarrow \Theta < \frac{D_t - t + (q+1)\Pi}{(q+1)} \end{aligned}$$

The first implication is due to $\alpha \leq 1$. The last inequality implies that when Eq. (A.5d) is true, $\langle(t, D_t), \alpha\rangle \notin \mathcal{F}_{q+1}(\Pi, \Theta)$. In the same way, it can be shown successively that $\langle(t, D_t), \alpha\rangle \notin \mathcal{F}_{q+1}(\Pi, \Theta) \Rightarrow \langle(t, D_t), \alpha\rangle \notin \mathcal{F}_{q+2}(\Pi, \Theta)$. Therefore, the implication can be easily drawn up to any $\ell > q$ which implies $\langle(t, D_t), \alpha\rangle \notin \mathcal{F}_q(\Pi, \Theta) \Rightarrow \langle(t, D_t), \alpha\rangle \notin \mathcal{F}_l(\Pi, \Theta)$

As the contrapositive of Lemma (A.5) is true for each conditions of (A.5), the Lemma is proved. □

From Lemmas 9 and 10, the following corollary follows directly.

Corollary 3. *Given $q = \lceil(t/\Pi)\rceil$, $\Pi \geq 0$, $l \geq 0$, $\Theta \geq 0$, then $\langle(t, D_t), \alpha\rangle \in \mathcal{F}_\ell(\Pi, \Theta) \Rightarrow \langle(t, D_t), \alpha\rangle \in \mathcal{F}_{q-1}(\Pi, \Theta) \bigcup \mathcal{F}_q(\Pi, \Theta)$*

Given t, Π, and $q = \lceil(t/\Pi)\rceil$. For any D_t, α and ℓ which satisfy the supposition of the above corollary, it must be that.

$$\{\Theta \in \mathbb{R}^+ | \langle(t, D_t), \alpha\rangle \in \mathcal{F}_\ell(\Pi, \Theta)\} \subseteq \{\Theta \in \mathbb{R}^+ | \langle(t, D_t), \alpha\rangle \in \mathcal{F}_{q-1}(\Pi, \Theta) \cup \mathcal{F}_q(\Pi, \Theta)\}.$$

By the above expression and Eq. (A.3) of Definition 5, the corollary below follows immediately.

Corollary 4. *For any $\ell' \in \mathbb{N}^+$ and $D_t, t, \alpha \in \mathbb{R}_{\geq 0}$, if $(\alpha \leq 1)$, then*

$$\Theta^*_{\ell'}(\Pi, \langle(t, D_t), \alpha\rangle) \geq \Theta^*_{\ell \in \{q, q-1\}}(\Pi, \langle(t, D_t), \alpha\rangle).$$

Proof of Theorem 6. Now it is easier to prove that $\Theta^{\min}$ returned from MinimumCap corresponds to the minimum capacity value required to schedule all of the half-line in system. The loop from Line 7–12 iterates through each value t_a in $\widetilde{\mathsf{TS}}(\tau, k)$ and calculate the minimum $\Theta^{\min}$ for all values t_a in $\widetilde{\mathsf{TS}}(\tau, k)$. Finally, in Line 11, $\Theta^{\min}$ is set to the maximum of $U(\tau) \cdot \Pi$ and $\Theta^*(\Pi, \langle(t_a, \widetilde{\mathsf{DBF}}(\tau, t_a, k)), \psi(\tau, t_a, k)\rangle)$ over all values t_a in $\widetilde{\mathsf{TS}}(\tau, k)$.

For each half-line, we have shown in Corollary 4 that only two ℓ-feasibility regions need to be checked. As the algorithm returns the maximum Θ from all of these feasibility regions, it ensures EDF-schedulability conditions of Theorem 1. From the definition of l-feasibility region and Lemma 7, $\widetilde{\mathsf{DBF}}(\tau, t, k) \leq \mathsf{sbf}((\Pi, \Theta^{\min}), t)$ for all $t \geq 0$ and $U(\tau) \leq \frac{\Theta}{\Pi}$. By Lemma 1, $\mathsf{DBF}(\tau, t) \leq \widetilde{\mathsf{DBF}}(\tau, t, k) \leq \mathsf{sbf}((\Pi, \Theta^{\min}), t)$ for all $t \geq 0$. Therefore, the supposition of Theorem 1 is satisfied and the τ will always meet all deadlines when scheduled by EDF upon the processor $(\Pi, \Theta^{\min})$. When $k = \infty$, $\widetilde{\mathsf{DBF}}(\tau, t, k)$ equals $\mathsf{DBF}(\tau, t)$ for all $t \geq 0$; in this case, $\Theta^{\min}$ equals $\Theta^*(\Pi, \tau)$ (i.e., $\Theta^{\min}$ is an exact value) due to the fact that Theorem 1 is necessary and sufficient. □

A.1. Approximation ratio

In the previous subsection in this Appendix A, we have shown that MinimumCap gives a valid answer when k is finite and an exact answer when k is infinite. When k is finite, we have not yet given any details on how accurate the returned $\Theta^{\min}$ will be. The following theorem can be shown which states that we may trade computational efficiency for accuracy; that is, as k increases, the guaranteed accuracy of MinimumCap increases along with its running time. Theorem 7 quantifies this tradeoff for a given k.

Theorem 7. *Given Π, τ, and $k \in \mathbb{N}$, the procedure* MinimumCap *returns $\Theta^{\min}$ such that*

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

where $\epsilon = \frac{1}{k}$. Furthermore, MinimumCap(Π, τ, k) *has time complexity $O(n/(\epsilon \lg n))$*

While the previous subsection (proof of Theorem 6) is similar to the results of Fisher and Dewan [26], it is not identical due to differences in the model. However, the proof of Theorem 7 is identical to Fisher and Dewan [26] and therefore omitted. Together, Theorems 6 and 7 imply Theorem 3.

References

[1] K. Skadron, M.R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, D. Tarjan, Temperature-aware microarchitecture, in: International Symposium on Computer Architecture, IEEE Computer Society Press, 2003.

[2] Y. Xiang, T. Chantem, R.P. Dick, X.S. Hu, L. Shang, System-level reliability modeling for mpsocs, in: International Conference on Hardware/Software Codesign and System Synthesis, 2010.

[3] E. Shih, P. Bahl, M.J. Sinclair, Wake on wireless: an event driven energy saving strategy for battery operated devices, in: Proceedings of the 8th Annual International Conference on Mobile Computing and Networking, ACM, New York, NY, USA, Atlanta, Georgia, USA, 2002, pp. 160–171.

[4] D. Meisner, B.T. Gold, T.F. Wenisch, Powernap: eliminating server idle power, in: Proceeding of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS'09, 2009.

[5] A.K. Mok, Fundamental design problems of distributed systems for the hard-real-time environment, Ph.D. thesis, Laboratory for Computer Science, Massachusetts Institute of Technology, available as Technical Report No. MIT/LCS/TR-297, 1983.

[6] I. Shin, I. Lee, Compositional real-time scheduling framework with periodic model, ACM Trans. Embedded Comput. Syst. 7 (3) (2008).

[7] V.V. Vazirani, Approximation Algorithms, Springer-Verlag, Berlin–Heidelberg–New York–Barcelona–Hong Kong–London–Milan –Paris–Singapore–Tokyo, 2001.

[8] J.-J. Chen, S. Wang, L. Thiele, Proactive speed scheduling for frame-based real-time tasks under thermal constraints, in: IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2009.

[9] T. Chantem, H.X. Sharon, R.P. Dick, Online work maximization under a peak temperature constraint, in: ISLPED'09: Proceedings of the 14th ACM/IEEE International Symposium on Low Power Electronics and Design, 2009.

[10] S. Wang, R. Bettati, Reactive speed control in temperature-constrained real-time systems, in: Euromicro Conference on Real-Time Systems, 2006.

[11] S. Wang, R. Bettati, Reactive speed control in temperature-constrained real-time systems, Real-Time Syst. J. 39 (1–3) (2008) 658–671.

[12] S. Wang, R. Bettati, Delay analysis in temperature-constrained hard real-time systems with general task arrivals, in: IEEE Real-Time Systems Symposium, 2006.

[13] F. Yao, A. Demers, S. Shenker, A scheduling model for reduced cpu energy, in: Proceedings of the 36th Annual Symposium on Foundations of Computer Science, FOCS'95, 1995.

[14] N. Bansal, K. Pruhs, Speed scaling to manage temperature, in: Symposium on Theoretical Aspects of Computer Science, 2005, pp. 460–471.

[15] W.-L. Hung, Y. Xie, N. Vijaykrishnan, M.T. Kandemir, M.J. Irwin, Thermal-aware task allocation and scheduling for embedded systems, in: ACM/IEEE Conference of Design, Automation, and Test in Europe, 2005, pp. 898–899.

[16] J.-J. Chen, C.-M. Hung, T.-W. Kuo, On the minimization of the instantaneous temperature for periodic real-time tasks, in: IEEE Real-Time and Embedded Technology and Applications Symposium, 2007.

[17] T. Chantem, R.P. Dick, X.S. Hu, Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs, in: Design, Automation and Test in Europe, 2008.

[18] N. Fisher, J.-J. Chen, S. Wang, L. Thiele, Thermal-aware global real-time scheduling on multicore systems, in: Proceedings of the 15th IEEE Real-Time and Embedded Technology and Applications Symposium, IEEE Computer Society Press, 2009.

[19] A. Ferreira, D. Mosse, J. Oh, Thermal faults modeling using a rc model with an application to web farms, in: Proceedings of the Euromicro Conference on Real-Time Systems, IEEE Computer Society, 2007.

[20] S. Gopalakrishnan, M. Caccamo, C.-S. Shih, C.-G. Lee, L. Sha, Finite-horizon scheduling of radar dwells with online template construction, in: Proceedings of the 25th IEEE International Real-Time Systems Symposium, IEEE Computer Society, Washington, DC, USA, 2004, pp. 23–33.

[21] S. Ghosh, J. Hansen, R. Rajkumar, J. Lehoczky, Integrated resource management and scheduling with multi-resource constraints, in: Real-Time Systems Symposium, 2004. Proceedings 25th IEEE International, 2004, pp. 12–22.

[22] I. Shin, I. Lee, Periodic resource model for compositional real-time guarantees, in: Proceedings of the IEEE Real-Time Systems Symposium, IEEE Computer Society, 2003, pp. 2–13.

[23] A. Easwaran, Compositional schedulability analysis supporting associativity, optimality, dependency and concurrency, Ph.D. thesis, Computer and Information Science, University of Pennsylvania, 2007.

[24] A. Easwaran, M. Anand, I. Lee, Compositional analysis framework using EDP resource models, in: Proceedings of the IEEE Real-time Systems Symposium, IEEE Computer Society, Tuscon, Arizona, 2007.

[25] E. Bini, G.C. Buttazzo, G. Lipari, Minimizing cpu energy in real-time systems with discrete speed management, ACM Trans. Embedded Comput. Syst. 8 (4) (2009).

[26] N. Fisher, F. Dewan, Approximate bandwidth allocation for compositional real-time systems, in: ECRTS 09: Proceedings of the 21st Euromicro Conference on Real-Time Systems, IEEE Computer Society Press, Dublin, Ireland, 2009.

[27] N. Fisher, An FPTAS for interface selection in the periodic resource model, in: Proceedings of 17th International Conference on Real-Time and Network Systems, Paris, France, 2009.

[28] P. Pillai, K.G. Shin, Real-time dynamic voltage scaling for low-power embedded operating systems, SIGOPS Oper. Syst. Rev. 35 (5) (2001).

[29] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, G.D. Micheli, Temperature-aware processor frequency assignment for mpsocs using convex optimization, in: IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis, 2007, pp. 111–116.

[30] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, L. Benini, G.D. Micheli, Temperature control of high-performance multi-core platforms using convex optimization, in: Design, Automation and Test in Europe, 2008, pp. 110–115.

[32] C. Liu, J. Layland, Scheduling algorithms for multiprogramming in a hard real-time environment, J. ACM 20 (1) (1973) 46–61.

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

[34] K. Albers, F. Slomka, An event stream driven approximation for the analysis of real-time systems, in: Proceedings of the EuroMicro Conference on Real-Time Systems, IEEE Computer Society Press, Catania, Sicily, 2004, pp. 187–195.

[35] I. Shin, I. Lee, Compositional real-time scheduling framework, in: Proceedings of the IEEE Real-Time Systems Symposium, IEEE Computer Society, 2004, pp. 57–67.

[36] E. Bini, G. Buttazzo, Biasing effects in schedulability measures, in: Proceedings of the 16th Euromicro Conference on Real-Time Systems, IEEE Computer Society, 2004, pp. 196–203.

[37] L. Thiele, S. Chakraborty, M. Naedele, Real-time calculus for scheduling hard real-time systems, in: IEEE International Symposium on Circuits and Systems, IEEE Computer Society, Geneva, Switzerland, 2000, pp. 101–104.

Masud Ahmed received his M.S. degree in Computer Science from Wayne State University, Detroit, Michigan and his B.Sc.Engg degree in Computer Science & Engineering from Bangladesh University of Engineering & Technology (BUET), Dhaka, Bangladesh. He is currently a Ph.D. candidate in Computer Science at Wayne State University working under the supervision of Prof. Nathan W. Fisher. His research interests include real-time systems, embedded systems, and sustainable computing.

Nathan Fisher is an Assistant Professor in the Department of Computer Science at Wayne State University. He received his Ph.D. from the University of North Carolina at Chapel Hill in 2007, his M.S. degree from Columbia University in 2002, and his B.S. degree from the University of Minnesota in 1999, all in computer science. His research interests are in real-time and embedded computer systems, parallel and distributed algorithms, resource allocation, algorithmic mechanism design, and approximation algorithms.

Dr. Shengquan Wang received his B.S. degree in Mathematics from Anhui Normal University, China, in 1995, and his M.S. degree in Applied Mathematics from the Shanghai Jiao Tong University, China. He also received M.S. degree in Mathematics in 2000 and Ph.D. in Computer Science from Texas A&M University in 2006. He is currently Assistant Professor in the Department of Computer and Information Science at the University of Michigan-Dearborn. He is a recipient of the US NSF CAREER Award. He also received the Best Paper Award in ECRTS 2006. His research interests are in real-time systems, networking and distributed systems, sustainable computing, security and privacy, and optimization.

Pradeep Hettiarachchi received his B.Sc. in Electronics and Telecommunication Engineering from University of Moratuwa, Sri Lanka in 2000 and his M.S. in Computer Science from St. Cloud State University, Minnesota in 2008. Currently he is a Ph.D. candidate in Computer Science at Wayne State University in Detroit, Michigan. His research interests are in real-time operating systems, thermal-aware control-theoretic real-time systems design, and optimized and energy-efficient real-time and embedded systems design.