17th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications

Energy-Efficient Scheduling Algorithms for Periodic Power Management for Real-Time Event Streams

Kai Huang, Jian-Jia Chen[†], Lothar Thiele[*]
fortiss GmbH, Guerickestr. 25, 80805 Munich, Germany
Email:{khuang}@fortiss.org
[†] Institute for Process Control and Robotics, Karlsruhe Institute of Technology, Germany
Email: {jian-jia.chen}@kit.edu
[*] Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland
Email: {thiele}@tik.ee.ethz.ch

***Abstract*—As modern VLSI technology is scaling to the deep sub-micron domain, embedded systems face a power-efficiency problem, i.e., static power consumption caused by the leakage current. This paper explores how to use dynamic power management to reduce static power consumption while guaranteeing hard real-time properties. To tackle event arrivals with non-deterministic patterns, the arrival curve model is adopted to describe event arrivals in the interval domain. To reduce runtime overhead, periodic power management is investigated, which turns on and off a system with a fixed period. To reduce the timing complexity of computing such a periodic scheme, two algorithms, which are based on a linear-segmented representation of the arrival curve model, are proposed to trade the complexity with accuracy for energy reduction. We also present simulation results to demonstrate the effectiveness of our algorithms.**

I. Introduction

Power dissipation has been an important design issue for embedded systems, in particular for the class of mobile and hand-held systems which are battery-powered. Batteries for these systems are, however, limited in both their power and energy output. The limited capacity of these batteries thus severely constrains the lifespan of this class of embedded systems. Therefore, reducing the power consumption is paramount importance in order to prolong the lifespan of these embedded systems.

Two major sources for the power consumption of the integrated transistors within an embedded system are dynamic power consumption due to switching activities and static power consumption due to leakage current. For micrometer-scale semiconductor technologies, the dynamic power dominates the power consumption of a chip. However, as modern VLSI technology is scaling down to the deep sub-micron domain, the static power is comparable and even larger than dynamic power [14], as smaller transistors leak more. Therefore, how to effectively reduce the static power is crucial in order to reduce the overall power consumption of modern and future embedded systems.

We explore energy-efficient scheduling to reduce the static power consumption while providing hard real-time guarantees in this paper. We consider a system with active, standby, and sleep modes with different levels of power consumption, and try to compute a periodic power management scheme to turn on (active or standby modes) and off (sleep mode) the system periodically during runtime. The computed periodic scheme should on the one hand accumulate as much as possible arrived workload for each on period, on the other hand guarantee real-time properties, e.g., preventing deadline violation. To cope with worst-case arrivals of incoming workload with non-deterministic patterns, we use event streams to describe the incoming workload of the system and adopt the arrival curve model to describe event arrivals in the interval domain.

We have studied such energy-efficient scheduling problem in [10]. Our previous investigation revealed that finding such a periodic scheme is computationally intensive, even computed offline. The reason is twofold. First of all, there is no regular pattern for the on/off periods with respect to the energy optimization of a given system. To derive the optimal period with respect to energy saving, exhaustively checking all possible combinations of the on and off values is the only possible way, which is not scalable for complex systems. Secondly, due to the adopted interval-domain model, the derivation of the worst-case guaranteed bounds for real-time properties requires numerical curve operations in $(min, +)$ algebra, which incurs considerable computational and memory overheads. Such numerical computation prohibits the derivation of an optimal periodic scheme for event streams with complex patterns.

This paper investigates how to reduce the computational complexity for the aforementioned energy-efficient scheduling problem. To reduce the numerical curve operations, a segmented representation of the arrival curves is proposed, which approximates an arrival curve with a sequence of linear segments. Based on the segmented representation, we present two algorithms to compute the optimal solution as well as an approximated solution with different levels of timing complexity. We also present proofs for the correctness of the proposed algorithms. With the presented techniques, a large portion of the numerical curve operations is excluded and computing an energy-efficient periodic scheme becomes much more light-weight in terms of both computing power and memory usage. To cope with scenarios of multiple event streams, an algorithm is developed to retrieve the segmented representation from a set of event streams. We also provide simulation results to demonstrate the effectiveness of our algorithms.

The rest of this paper is organized as follows: Section II reviews related work in the literature. Section III states our system models and formalizes the problem. Section IV presents our algorithms. Simulations results are presented in Section V and Section VI concludes the paper.

II. Related Work

Energy-efficient scheduling has often be exploited in order to minimize the energy consumption while meeting the timing satisfaction of real-time tasks. Applying DVFS for real-time systems, for instance, has received much attention in the past decade, and has been extensively studied. In deep sub-micron technologies, static power consumption in CMOS circuits has a significant contribution to the total power consumption. As a result, by considering both dynamic power management (DPM) and DVFS, Irani et al. [12], Jejurikar et al. [14], [13], and Chen and Kuo [3], [4] propose to execute tasks at certain speeds and to control the procrastination of real-time events.

When only considering DPM to change the device/system modes dynamically, Baptiste [2] develops an algorithm based on dynamic programming to control when to activate and deactivate the device/system for a set of aperiodic real-time jobs with the same execution time. Shrivastava et al. [22] develop a framework for code transformations such that the idle time can be aggregated for energy reduction. Park et al. [20] use dynamic programming to decide where to insert instructions to put some execution units to low-power mode. By controlling shutting down and waking up system devices for energy efficiency, Swaminathan et al. [23] explore dynamic power management of real-time events. Yang et al. [28] propose to control preemption to reduce the idle energy consumption in devices. Devadas and Aydin [8] adopt device forbidden regions for preemption to prolong the idle time so that a device can be put to the sleep mode as long as possible. Moreover, Augustine et al. [1] explore systems with multiple low-power states, and design an on-line competitive algorithm to decide which low-power state the system should enter when there is no event in the ready queue.

Most of the above approaches require regular event arrivals, such as periodic real-time events [4], [28], [8], or precise information about event arrivals, such as aperiodic real-time events with known arrival time [2]. However, when the events might have burst or may come to the system without regular or precise event arrival patterns, these approaches cannot be applied. The non-determinism in the timing of event arrivals comes from two main reasons: (a) An event may be triggered by the physical environment, which, in general, is not able to be accurately predicted. (b) When a distributed system is considered, an event might be triggered by other events on different processing components, in which variable execution workloads, communication delays, and interferences on shared resources would make the prediction of precise information on event arrivals extremely complicated. As a result, when analyzing and designing embedded systems with real-time constraints, contemporary methods abstract over individual events and their individual processing.

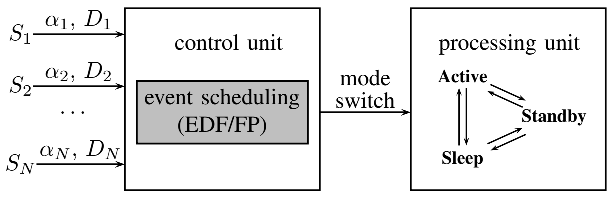
To model irregular event arrivals, Real-Time Calculus (RTC) [24], which is based on Network Calculus [7], [16], can be adopted. Specifically, the arrival curves in RTC model an upper bound and a lower bound of the number of events arrivals or the demand of computation under specified time interval lengths. Even though arrival curves provide upper and lower bounds, the actual event arrivals are non-deterministic. By considering DVFS systems, Maxiaguine et al. [18] compute a safe frequency at periodical interval with predefined length to prevent buffer overflow of a system, while Chen et al. [5] Perathoner et al. [21] explore the schedulability for online DVS scheduling algorithms proposed by Yao et al. [29]. Considering only DPM, Huang et al. [11] explore online algorithms to turn on and off a system based on runtime decision. In [10], Huang et al. also consider offline algorithms to compute a periodic power scheme. All above RTC-based energy-efficient scheduling analyses rely on computational expensive numerical computation which incurs non-negligible memory and computational overheads and certain approximation heuristics must be applied to compromise accuracy of the computed results and usage of the computing resources. This paper presents low-overhead algorithms, trying to solve the timing and spacing complexity of the energy-efficient scheduling problem in a particular way.

III. Models and Problem Definition

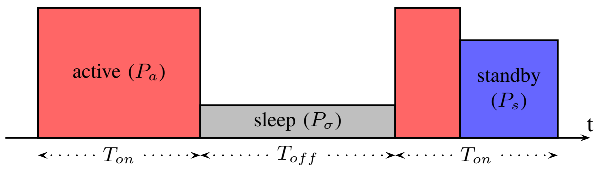
This section states the system models on which our work is based. The problem we study is also formally defined.

A. Power Model

We consider that a system consists of a processing unit and a control unit, as shown in Fig. 1(a). The control unit schedules input events and decides when to change the power mode of the processing unit. The processing unit has three power consumption modes, namely *active, standby,* and *sleep* modes. The power consumption in the sleep mode is P_σ. To serve an event, the processing unit must be in the active mode with power consumption P_a, in which $P_a > P_\sigma$. Once there is no event to serve, the processing unit can enter the sleep mode. However, switching from the sleep mode to the active mode takes time, denoted by $t_{sw,on}$, and requires additional energy overhead, denoted by $E_{sw,on}$. To prevent from frequent mode switches, the processing unit can also stay in the standby mode with power consumption P_s where $P_a > P_s > P_\sigma$. Fig. 1(b) graphically depicts the power model of the processing unit. In this paper, we assume that switching between the standby mode and the active mode has negligible overhead, compared to the other switches, which is the same as the assumption in [30], [27]. Moreover, switching from the active (also standby) mode to the sleep mode takes time, denoted by $t_{sw,sleep}$, and requires additional energy overhead, denoted by $E_{sw,sleep}$.



(a) Abstract view of the system



(b) Power model of the processing unit

Fig. 1. The abstract model of the periodic power management problem.

B. System Model

We focuses on events that arrive irregularly. To model irregular arrival of events, we adopt the arrival curves $\bar{\alpha}(\Delta) = [\bar{\alpha}^u(\Delta), \bar{\alpha}^l(\Delta)]$ from Real-Time Calculus [24], in which $\bar{\alpha}_i^u(\Delta)$ and $\bar{\alpha}_i^l(\Delta)$ are the upper and lower bounds on the number of arrival events for a stream S_i in any time interval of length Δ, respectively. The concept of arrival curves unifies many other common timing models of event streams. For example, a periodic event stream can be modeled by a set of step functions where $\bar{\alpha}^u(\Delta) = \left\lfloor \frac{\Delta}{p} \right\rfloor + 1$ and $\bar{\alpha}^l(\Delta) = \left\lfloor \frac{\Delta}{p} \right\rfloor$. For a sporadic event stream with minimal inter arrival distance p and maximal inter arrival distance p', the upper and lower arrival curve is $\bar{\alpha}^u(\Delta) = \left\lfloor \frac{\Delta}{p} \right\rfloor + 1$, $\bar{\alpha}^l(\Delta) = \left\lfloor \frac{\Delta}{p'} \right\rfloor$, respectively. A widely used model to specify an arrival curve is the PJD model by which an arrival curve is characterized with a period p, jitter j, and minimal inter arrival distance d. The upper arrival curve is thus $\bar{\alpha}^u(\Delta) = \min\{\left\lceil \frac{\Delta+j}{p} \right\rceil, \left\lceil \frac{\Delta}{d} \right\rceil\}$. Analogous to arrival curves that provide an abstract event stream model, a tuple $\beta(\Delta) = [\beta^u(\Delta), \beta^l(\Delta)]$ defines an abstract resource model which provides an upper and lower bounds on the available resources in any time interval Δ. For details, please refer to [24].

Note that an arrival curve $\bar{\alpha}_i(\Delta)$ specifies the number of events of stream S_i whereas a service curve $\beta(\Delta)$ specifies the available amount of time for execution, for interval length Δ. Therefore, $\bar{\alpha}_i(\Delta)$ has to be transformed to $\alpha_i(\Delta)$ to indicate the amount of computation time required for the arrived events in intervals. Suppose that the execution time of any event in stream S_i is w_i. The transformation can be done by $\alpha_i^u = w_i\bar{\alpha}_i^u$, $\alpha_i^l = w_i\bar{\alpha}_i^l$ and back by $\bar{\alpha}_i^u = \alpha^u/w_i$, $\bar{\alpha}_i^l = \alpha^l/w_i$ thereof. For variable workloads in an event stream, our algorithms can be revised slightly by adopting the workload curves in [19]. Moreover, to satisfy the real-time constraint, the response time of an event in event stream S_i must be no more than its specified relative deadline D_i, where the response time of an event is its finishing time minus the arrival time of the event. On the arrival of an event of stream S_i at time t, the absolute deadline is $t + D_i$.

C. Problem Definition

This paper explores how to efficiently and effectively minimize the energy consumption of the processing unit to serve a set of event streams under hard real-time requirements. Of course, one could determine when to transit between modes to reduce the energy consumption dynamically, but the computational overhead is quite significant since the decision must be done by evaluating the arrival curves dynamically.

An abstract model of our periodic power management (PPM), is illustrated in Fig. 1, in which the power management is done by analyzing the arrival curves of a set of event streams $\mathcal{S}$ statically. Specifically, the PPM scheme schedules the processing unit with fixed period $T = T_{on} + T_{off}$ for power management, i.e., switching the system to the active or standby modes for T_{on} time units, following by T_{off} time units in the sleep mode.

Based on the model in Fig. 1, the energy consumption for a time interval L, where $L \gg T$ and $\frac{L}{T}$ is an integer, consists of mode-switch overheads, the energy in `standby` and `sleep` modes, and the energy for processing arrived events. Supposing that $\gamma_i(L)$ is the number of events of event stream S_i served in interval L and all the served events finish in time interval L, the energy consumption $E(L, T_{on}, T_{off})$ can be computed as follows:

$$\begin{aligned} E(L, T_{on}, T_{off}) = &\ \frac{L}{T_{on} + T_{off}} E_{sw} \\ &+ \frac{L \cdot T_{on}}{T_{on} + T_{off}} P_s + \frac{L \cdot T_{off}}{T_{on} + T_{off}} P_\sigma \\ &+ \sum_{S_i \in \mathcal{S}} w_i \cdot \gamma_i(L) \cdot (P_a - P_s) \\ = &\ \frac{L \cdot E_{sw}}{T_{on} + T_{off}} + \frac{L \cdot T_{on} \cdot (P_s - P_\sigma)}{T_{on} + T_{off}} \\ &+ L \cdot P_\sigma + \sum_{S_i \in \mathcal{S}} w_i \cdot \gamma_i(L) \cdot (P_a - P_s) \end{aligned} \tag{1}$$

where E_{sw} is $E_{sw,on} + E_{sw,sleep}$ for brevity.

Based on (1), we define the average idle power consumption: Given a sufficiently large L, the average idle power consumption P is computed as:

$$\begin{aligned} P(T_{on}, T_{off}) &\overset{\text{def}}{=} \frac{\frac{L \cdot E_{sw}}{T_{on}+T_{off}} + \frac{L \cdot T_{on} \cdot (P_s - P_\sigma)}{T_{on}+T_{off}}}{L} \\ &= \frac{E_{sw} + T_{on} \cdot (P_s - P_\sigma)}{T_{on} + T_{off}} \end{aligned} \tag{2}$$

According to (1), $L \cdot P_\sigma + \sum_{S_i \in \mathcal{S}} w_i \cdot \gamma_i(L)(P_a - P_s)$ is a constant for a given L. Therefore, for an L that is sufficiently large, without changing the scheduling policy, the minimization of energy consumption $E(L, T_{on}, T_{off})$ of a PPM scheme requires to find T_{on} and T_{off} so that the average idle power consumption $P(T_{on}, T_{off})$ is minimized. We now define the PPM problem studied in this section as follows:

Given a set of event streams $\mathcal{S}$ under real-time requirements, the objective of the studied problem is to find a periodic power management characterized by T_{on} and T_{off} that minimizes the average idle power consumption $P(T_{on}, T_{off})$, in which the response time of any event of event stream S_i in $\mathcal{S}$ must be no more than D_i.

IV. Our Approaches

Using the interval-based model, one can perform worst-case scheduling analysis by means of Real-Time Calculus [24]. Within this context, the processing unit is said to provide guaranteed output service $\beta^G(\Delta)$. For instance, if the processing unit offers computation time Δ in any time interval of length Δ, then $\beta^G(\Delta) = \Delta$. Suppose now that an event stream S_i needs service curve $\beta^A(\Delta)$, denoted as service demand, to operate correctly, i.e., to satisfy responsiveness constraints. The processing unit is schedulable if and only if the condition

$$\beta^G(\Delta) \geq \beta^A(\Delta), \forall \Delta \geq 0 \tag{3}$$

holds. For example, if the relative deadline D_i of each event in event stream S_i is given, one can say the processing unit has to provide at least $\beta^A(\Delta) = \alpha_i(\Delta - D_i)$ service to prevent deadline misses of events in S_i. In the rest of this article, we show how to properly construct $\beta^G(\Delta)$ and $\beta^A(\Delta)$ for a given system to not only reduce average idle power consumption P but also guarantee hard real-time properties.

Reviewing the formulation of the average idle power consumption $P(T_{on}, T_{off}) = \frac{E_{sw} + T_{on}(P_s - P_\sigma)}{T_{on} + T_{off}}$, there are two cases. (1) If $\frac{E_{sw}}{P_s - P_\delta} \geq T_{off}$, we know that $P(T_{on}, T_{off})$ is minimized when T_{on} is set to ∞. (2) If $\frac{E_{sw}}{P_s - P_\delta} < T_{off}$, the minimal T_{on} under the service constraint $\beta^G(\Delta)$ minimizes the average idle power consumption $P(T_{on}, T_{off})$. In this sense, $\frac{E_{sw}}{P_s - P_\delta}$, can be seen as the break-even time of the system. Our approaches proposed in this paper are based on (1) the finding of the minimal T_{on} under the service constraint $\beta^G(\Delta)$, provided that T_{off} is given, and (2) the exploration of the best T_{off}. One could also derive solutions in another direction by searching the best T_{off} for a specified T_{on} along with the exploration on T_{on}, but the procedure would be more complicated.

A. Motivation Example

The goal of our algorithms is to find a pair $(T_{on}, T_{off}) \in \mathbb{R}^+ \times \mathbb{R}^+$ so that on the one hand the average idle power consumption of the processing unit is minimized, on the other hand the constructed β^G based on (T_{on}, T_{off}) bounds the service demand of the event stream S_1, i.e., fulfilling (3). Note that for a given T_{on} and T_{off} pair, the guaranteed service of the processing unit is computed by:

$$\beta^G(\Delta) = \max\left(\left\lfloor \frac{\Delta}{T_{on} + T_{off}} \right\rfloor \cdot T_{on},\ \Delta - \left\lceil \frac{\Delta}{T_{on} + T_{off}} \right\rceil \cdot T_{off}\right) \tag{4}$$

By β^G in (4), the service demand curve of S_1, i.e., $\beta^A = \alpha_1^u(\Delta - D_1)$, and the schedulability definition in (3), the minimal T_{on} to fulfill the schedulability requirement in terms of a given T_{off} can be defined as:

$$T_{on}^{\min} = \min\left\{T_{on} : \beta^G(\Delta) \geq \beta^A(\Delta), \forall \Delta \geq 0\right\} \tag{5}$$

To our best knowledge, there is no explicit form to compute T_{on}^{min}. Furthermore, due to the complex shape of the arrival curves, exhaustively testing (3) is the only way to determine the minimal average idle power from all possible T_{on}. Fig. 2 presents an example for illustrating the irregular pattern of the average idle power P by solving (5). As a result, exhaustively checking all possible (T_{on}, T_{off}) pairs is the only way to find the minimum of the average idle power P.

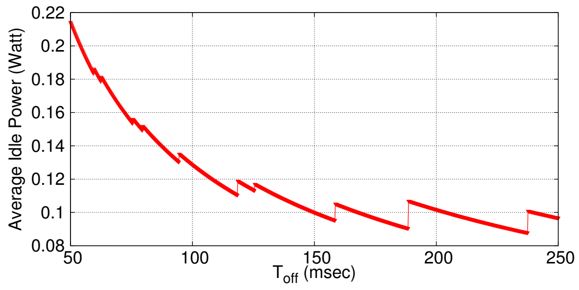


Fig. 2. The relation of the minimal average idle power consumption and T_{off} for an IMB Microdrive processing stream S_1, see Tab. I and II in Section V-A.

B. Segmented Arrival Curves

According to [15], an arrival curve can be represented as a set of staircase functions, each of which is defined by a 2-tuple $\langle N, \delta\rangle$ where N is the initial numbers of events at time 0, δ is the width of the step, and the stair is 1. A staircase function, on the other hand, can be approximated by a linear function $y = \frac{1}{\delta}x + N$ in the Cartesian coordinate system. Based on above knowledge, we can approximate an arrival curve with a set of linear segments, as defined in the following.

Def. 1 (Curve Segment): A curve segment is a 2-tuple $\langle \nu, \rho\rangle$, that specifies a straight line in the Cartesian coordinate system, starting from point $\nu = (x_\nu, y_\nu)$ with slope ρ where $x_\nu, y_\nu, \rho \in \mathbb{R}_{\geq 0}$

Def. 2 (Linear Segmented Curve): A linear segmented curve is the minimum of a finite set of ordered curve segments

$$\min\{\langle \nu_1, \rho_1\rangle, \langle \nu_2, \rho_2\rangle, \ldots, \langle \nu_m, \rho_m\rangle\} \tag{6}$$

Def. 3 (Segmented Upper Arrival Curve): A segmented upper arrival curve is defined as segments

$$\tilde{\alpha}^u = \min_{1 \leq i \leq m}\{\langle \nu_i, \rho_i\rangle\} \tag{7}$$

where $\rho_i > \rho_{i+1}$, $x_{\nu_i} < x_{\nu_{i+1}}$ and $y_{\nu_i} \leq y_{\nu_{i+1}}$ for all $1 \leq i < m$.

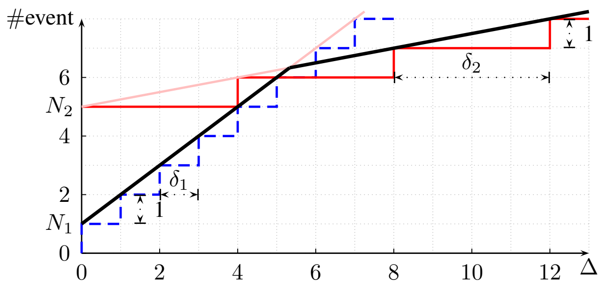


Fig. 3. A graphical view of the segmented-curve approximation from a PJD curve where $p = 4$, $j = 16$, and $d = 1$.

For example, an upper arrival curve specified in the PJD model (Section III-B) can be represented as two staircase functions $\{\langle N_1, \delta_1\rangle, \langle N_2, \delta_2\rangle\}$ where

$$N_1 = 1;\ \delta_1 = d;\ N_2 = \left\lceil \frac{j}{p} \right\rceil + 1;\ \delta_2 = p$$

The 2-staircase representation of an arrival curve can then be approximated by a 3-segment linear curve $\{\langle(x_1, y_1), \rho_1\rangle, \langle(x_2, y_2), \rho_2\rangle, \langle(x_3, y_3), \rho_3\rangle\}$. A graphical illustration of above statements is depicted in Fig. 3, where the dashed and solid staircase curves represent staircase functions $\langle N_1, \delta_1\rangle$ and $\langle N_2, \delta_2\rangle$, respectively. From the figure, it can be easily seen that

$$x_1 = 0, \quad y_1 = 0, \quad \rho_1 = +\infty;$$
$$x_2 = 0, \quad y_2 = N_1, \quad \rho_2 = \frac{1}{\delta_1}; \quad \rho_3 = \frac{1}{\delta_2}$$

The x_3 and y_3 can be computed as follows

$$x_3 = \frac{N_2 - N_1}{\rho_1 - \rho_2} = \frac{N_2 - N_1}{\frac{1}{\delta_1} - \frac{1}{\delta_2}} = \frac{\left\lceil \frac{j}{p} \right\rceil \cdot d \cdot p}{p - d}$$

$$y_3 = \frac{1}{\delta_2} x_3 + N_2 = \frac{N_2\delta_2 - N_1\delta_1}{\delta_2 - \delta_1} = \frac{(\left\lceil \frac{j}{p} \right\rceil + 1) \cdot p - d}{p - d}$$

In summary, an upper arrival curve defined in PJD model can be transformed into a 3-segment linear curve $\{\langle(0, 0), +\infty\rangle, \langle(0, 1), \frac{1}{d}\rangle, \langle(\frac{\lceil \frac{j}{p} \rceil \cdot d \cdot p}{p-d}, \frac{(\lceil \frac{j}{p} \rceil + 1)\cdot p - d}{p-d}), \frac{1}{p}\rangle\}$. The bold line in Fig. 3 depicts the resulting segmented curve from the above computation.

C. Algorithms for Segmented Curves

Based on the segmented curve, this section provide approximated and exact algorithms for the PPM minimization problem defined in Section III-C. The basic idea is to compute a T_{on} with the smallest idle power P for each T_{off}, then find the minimum P for all possible T_{off}.

To prevent input events from deadline missing, we use the concept of bounded delay function [17]:

Def. 4 (Bounded Delay): A bounded-delay function $\textbf{bdf}(\Delta, \rho, T_{off})$, defined by slope ρ and T_{off} for interval length Δ, is

$$\boldsymbol{bdf}(\Delta, \rho, T_{off}) \overset{\text{def}}{=} \max\{0, \rho \cdot (\Delta - T_{off})\} \tag{8}$$

The ***bdf*** function can be considered as a linear service curve that provides services with rate ρ, starting from T_{off} time.

This section first presents the basic algorithms with single event stream, e.g., S_1. How to cope with cases of multiple event streams is presented in the next section.

1) Approximated Algorithm: The idea of this algorithm is, for a given T_{off}, to first compute a ***bdf*** function with minimal slope $\rho_{\min, T_{off}}$ that bounds the segmented service curve, i.e.,

$$\rho_{\min, T_{off}} = \inf\left\{\rho : \textbf{bdf}(\Delta, \rho, T_{off}) \geq \beta^A, \forall\Delta \geq 0\right\}, \tag{9}$$

then compute T_{on}^{min} based on $\rho_{min, T_{off}}$. Before presenting the detailed algorithm, we state another definition.

Def. 5 (Point Slope): We define ρ_{ν_i} as the slope of the line connecting points $(T_{off}, 0)$ and ν_i in the Cartesian coordinate system.

The pseudo code of computing T_{on}^{min} with the smallest average idle power P for a given T_{off} is depicted in Algo. 1. In Line 4, the tangent point of β^A is computed, denoted as $\nu^\top$, with which curve segment $\langle(T_{off}, 0), \rho_{\nu^\top}\rangle$ is the tangent line of β^A starting from $(T_{off}, 0)$. With the computed $\rho_{\nu^\top}$, ***bdf***$(\Delta, \rho_{\nu^\top}, T_{off})$ is the minimal ***bdf*** that bound β^A, i.e., $\rho_{\min, T_{off}} = \rho_{\nu^\top}$. In Line 8, T_{on} is computed based on the tangent slope $\rho_{\nu^\top}$.

Algorithm 1 LSC-BDA

```
Input: the set of curve segments for service demand $\beta^A$
    $\{\langle\nu_1, \rho_1\rangle, \langle\nu_2, \rho_2\rangle, \ldots, \langle\nu_m, \rho_m\rangle\}$ and $T_{off}$
Output: $T_{on}$
 1: if $\rho_{\nu_m} \leq \rho_m$ then
 2:     $T_{on} \leftarrow \frac{\rho_m \cdot T_{off}}{1-\rho_m}$
 3: else
 4:     $\nu^\top \leftarrow \{\nu_i | \min\{i | \rho_{\nu_i} \geq \rho_i\}, \langle\nu_i, \rho_i\rangle \in \beta^A\}$
 5:     if $x^\top < y^\top + T_{off}$ then
 6:         $T_{on} \leftarrow -1$                    ▷ no feasible solution
 7:     else
 8:         $T_{on} \leftarrow \frac{\rho_{\nu^\top} \cdot T_{off}}{1-\rho_{\nu^\top}}$
 9:     end if
10: end if
```

Fig. 4 shows an example for Algo. 1. The dotted line is the upper arrival curve $\tilde{\alpha}_1^u$ which is composed of curve segments $\{\langle(0, 0), +\infty\rangle, \langle(0, 4), 4\rangle, \langle(1, 8), 0.4\rangle, \langle(6, 10), 0.25\rangle\}$. The last segment is extended to positive infinity. The relative deadline D_1 of this event stream is 12. The dashed line is the corresponding service demand β^A. For a given $T_{off} = 2$, the tangent point $\nu^\top$ is ν_3 and the solid line crossing ν_3 is the tangent of β^A. The bold line is the resulting β^G.

As shown in Fig. 4, the β^G constructed by the T_{on} obtained from Algo. 1 is not the tightest bound for a given β^A and T_{off}. The algorithm, however, offers three advantages. First of all, the numerical curve comparisons to find $\rho_{\min, T_{off}}$ in Eqn. (9) are replaced by simple number comparisons, i.e., Line 4 of the algorithm, which eliminates the timing and space costs by the complex numerical curve comparisons. Second, because of the segmented representation of an arrival curve,

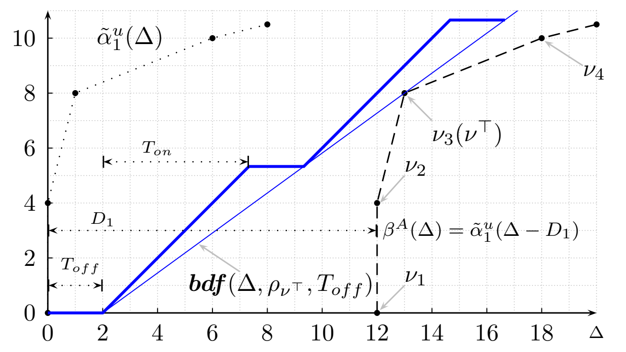


Fig. 4. An example for the LSC-BDA algorithm.

$\rho_{\min, T_{off}} = \rho_{\nu^\top}$ and $\nu^\top$ in Line 4 can be obtained by a simple binary search on all curve segments of β^A. The timing complexity of the algorithm is thereby $O(\log m)$ where m is the number of segments of β^A. Third, by applying Algo. 1, T_{on} can be considered as a function of T_{off}, denoted as $T_{on} = f(T_{off})$. This function is strictly increasing for all possible T_{off}. By using this monotonic function, the objective function $P(T_{on}, T_{off})$ in (2) can be transformed into a convex function $P(f(T_{off}), T_{off})$. The convexity of the transformed objective function enables a fast computation of the minimal P, e.g., by applying a bisection search for the feasible region of T_{off}. The timing complexity for the minimization of the PPM problem is reduced to $O(\log n \cdot \log m)$ where n is the number of possible T_{off}. The reduction of the timing complexity from $O(n)$ to $O(\log n)$ makes sense because n is pseudo-polynomial by considering T_{off} in all the values in the range computed in Section IV-C3. The following Lem. 1 and Thm. 1 prove the monotonicity and convexity, respectively.

Lem. 1: Given a segmented β^A, function $f(T_{off})$ defined in Algo. 1 is strictly increasing.

Proof: From the definition of f, one has $\rho_{\min, T_{off}} < \rho_{\min, (1+\epsilon)T_{off}}$ and thereby $f(T_{off}) < f((1+\epsilon)T_{off})$, which proves the property for strict increase. ∎

Thm. 1: Using Algo. 1 to compute $T_{on} = f(T_{off})$ for every T_{off}, $P(T_{on}, T_{off}) = P(f(T_{off}), T_{off})$ is a convex function.

The proof is similar to the one in [10]. We attach the formal proof in the appendix for completeness.

2) Optimal Algorithm: Given a segmented β^A, it is also possible to compute T_{on}^{min} with which the constructed β^G by (4) is the tangent of β^A (Remind that the minimal T_{on} results in smallest idle power P in (2) for a given T_{off}). The intuition is that the tangent point of β^G, denoted as ν^*, is near the $\nu^\top$ obtained from Algo. 1. In this section, we first present the algorithm to obtain T_{on}^{min}, then prove the tightness of the algorithm.

The pseudo code of the algorithm is shown in Algo. 2. By the $\nu^\top$ obtained from Algo. 1, we first compute the number of on/off periods that bounds β^A (Line 5), denoted as τ. Because τ is the maximal number of periods that bounds β^A, we know that $\boldsymbol{bdf}(\Delta, 1, (\tau + 1) \cdot T_{off})$ is a secant of β^A. We then compute the upper secant point by Lines 6–7, denoted as ν^*. With the point slope ρ_{ν^*} for ν^*, we obtain T_{on}^{min}.

Algorithm 2 LSC-OPT

```
Input: the set of curve segments for service demand $\beta^A$
    $\{\langle\nu_1, \rho_1\rangle, \langle\nu_2, \rho_2\rangle, \ldots, \langle\nu_m, \rho_m\rangle\}$ and $T_{off}$
Output: $T_{on}$
1: get $\nu^\top$ using Algo. 1
2: if $x^\top < y^\top + T_{off}$ then
3:     $T_{on} \leftarrow -1$                          ▷ no feasible solution
4: else
5:     $\tau \leftarrow \left\lfloor \frac{x^\top - (y^\top + T_{off})}{T_{off}} \right\rfloor$          ▷ number of periods
6:     $\beta^A \leftarrow \min\{\langle\nu_\top, \rho_\top\rangle, \langle\nu_{\top+1}, \rho_{\top+1}\rangle, \ldots, \langle\nu_m, \rho_m\rangle\}$
7:     $\nu^* \leftarrow \{\beta^A \bigcap \boldsymbol{bdf}(\Delta, 1, (\tau+1)\cdot T_{off})\}$
8:     $T_{on} \leftarrow \frac{\rho_{\nu^*} \cdot T_{off}}{1-\rho_{\nu^*}}$
9: end if
```

Fig. 5 depicts how Algo. 2 works using the same example shown in Fig. 4. For the given $T_{off} = 2$, the tangent point $\nu^\top$ for $\boldsymbol{bdf}(\Delta, \rho_{min,2}, 2)$ is ν_3. The number of periods τ, i.e., the horizontal distance between ν_3 and $\boldsymbol{bdf}(\Delta, 1, 2)$, is 1. The upper secant point of $\boldsymbol{bdf}(\Delta, 1, 2{*}2)$ and β^A is $\nu^* = (\frac{44}{3}, \frac{26}{3})$ and the resulting $T_{on}^{min} = \frac{13}{3}$.

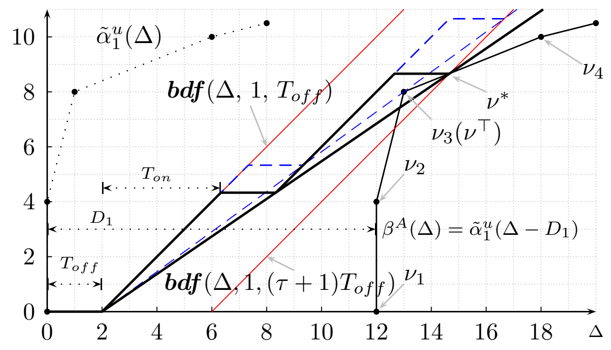


Fig. 5. An example for the LSC-OPT algorithm.

Now we prove the tightness of the algorithm.

Lem. 2: If $\rho_m < \rho_{\nu_m}$, $\nu^\top$ always exists and it is unique.

Proof: If $\rho_m < \rho_{\nu_m}$, the line $\langle(T_{off}, 0), \rho_{\nu_m}\rangle$ always intersects with β^A. Therefore $\nu^\top$ exists. Because ρ_i of β^A is strictly decreasing by Def. 3, $\nu^\top$ is unique. ∎

Thm. 2: With a given T_{off} and the computed T_{on} by Algo. 2, the β^G computed by (4) satisfies (3).

Proof: We sketch the proof as follows. According to LEM. 2, $\nu^\top$ is unique. Therefore, ν^* is unique as well. The constructed β^G can thereby be considered as an upper and a lower parts, separating by ν^*, designated as $\beta^{\wedge}_{\nu^*}$ and $\beta^{\nu^*}_{\vee}$, respectively. We first consider the upper part $\beta^{\wedge}_{\nu^*}$. Since ν^* is the intersection of curve segments $\langle\nu^\top, \rho_{\nu^\top}\rangle$ and $\langle(T_{off}, 0), \rho_{\nu^*}\rangle$, it implies $\rho_{\nu^*} > \rho_{\nu^\top}$. Therefore, segment $\langle\nu_{\nu^*}, \rho_{\nu^*}\rangle$ and $\beta^{\wedge}_{\nu^*}$ bound $\langle\nu_{\nu^*}, \rho_{\nu^\top}\rangle$ and consequently all segments with slopes less than $\rho_{\nu^\top}$. Now we consider the lower part $\beta_{\vee}^{\nu^*}$. As τ is the number of periods between $\nu^{\top}$ to $\boldsymbol{bdf}(\Delta, 1, T_{off})$, $\boldsymbol{bdf}(\Delta, 1, \tau \cdot T_{off})$ bounds line segment $\langle \nu^{\top}, \nu^* \rangle$. The last serving period of $\beta_{\vee}^{\nu^*}$ follows $\boldsymbol{bdf}(\Delta, 1, \tau \cdot T_{off})$ and will stop serving from $x_{\nu^*} - T_{off}$ until x_{ν^*}. Therefore, $\beta_{\vee}^{\nu^*}$ bounds line segment $\langle \nu^{\top}, \nu^* \rangle$. As slopes of all ν_i are larger than ρ_{ν^*}, $\beta_{\vee}^{\nu^*}$ bounds the portion of β^A from ν_1 until $\nu^{\top}$. ■

Thm. 3: With a given T_{off} and the computed T_{on} by Algo. 2, the β^G computed by (4) is the minimal periodic service curve that guarantees (3).

Proof: As ν^* is the tangent point of the constructed β^G to β^A, any smaller T_{on} will make β^G intersect with β^A. Therefore, the computed T_{on} is the minimum. ■

Because of the monotonicity of a segmented curve, binary search can also be used to find ν^* by checking points $\nu^{\top}$ to ν_m (Line 7 of the algorithm), resulting in $O(\log m)$ timing complexity for the algorithm. However, the function $T_{on} = f(T_{off})$ by applying this algorithm is not strictly increasing anymore. The objective function P in this case is not convex, resulting in $O(n \log m)$ complexity of the overall computation for the PPM problem.

3) Feasible Region of T_{off}: We discuss the feasible region of T_{off} in this section. Intuitively, if T_{off} is smaller than the break-even time, i.e., $\frac{E_{sw}}{P_s - P_\delta}$, turning the system to the sleep mode consumes more energy than the energy overhead E_{sw} for mode switching. The sleep mode thereby introduces additional energy consumption. Therefore, the region $\left[0, \frac{E_{sw}}{P_s - P_\delta}\right]$ can be safely discarded. Moreover, as T_{off} must also satisfy the timing overhead for mode switches, we also know that T_{off} must be no less than $t_{sw,sleep} + t_{sw,on}$.

There is also an upper bound for T_{off}. On one hand, T_{off} should be smaller than $D_1 - c_1$. Otherwise, no event can be finished before its deadline. On the other hand, as the system provides no service when it is off, that imposes a maximum service $\beta_{\top}^G(\Delta) = \max\{0, \Delta - T_{off}\}$. According to Real-Time Interface in (3), we know that predicate

$$\beta_{\top}^G(\Delta) = \max\{0, \Delta - T_{off}\} \geq \beta_1^A(\Delta) = \alpha_1(\Delta - D_1) \quad (10)$$

has to be true to satisfy the timing constraint. By inverting (10), we can compute the maximum T_{off} as

$$T_{off}^{max} = \max\left\{T_{off} : \beta_{\top}^G(\Delta) \geq \beta_1^A(\Delta), \forall \Delta \geq 0\right\}. \quad (11)$$

In summary, the feasible region of $T_{off} \in [T_{off}^l, T_{off}^r]$ can be bounded as follows:

$$T_{off}^l = \max\left\{t_{sw,sleep} + t_{sw,on}, \frac{E_{sw}}{P_S - P_\delta}\right\} \quad (12)$$

$$T_{off}^r = \min\left\{D_1 - c_1, T_{off}^{max}\right\} \quad (13)$$

D. Multiple Event Streams

Both algorithms presented in the previous section can be applied to cases of multiple event streams. Before applying the algorithms, we need to first compute the total service demand β_{total}^A for an event stream set $\mathcal{S} = \{S_1, S_2, \ldots, S_N\}$. For preemptive earliest-deadline-first (EDF) scheduling, the service demand $\beta_{total}^A(\Delta)$ is simply $\sum_{S_i \in \mathcal{S}} \alpha_i^u(\Delta - D_i)$. For first-come first-serve (FCFS) scheduling, the service bound $\beta_{total}^A(\Delta)$ is $\sum_{S_i \in \mathcal{S}} \alpha_i^u(\Delta - D_{\min})$, where $D_{\min}$ is the minimum relative deadline of all event streams in $\mathcal{S}$. For preemptive fixed-priority (FP) scheduling, one can apply the backward approach used in [11].

The problem remained to solve is that the resulting β_{total}^A by applying different scheduling policies might not be segmented curve anymore. As the dotted line shown in Fig. 6, the curve segments are not strictly increasing. We provide an algorithm to retrieve a correct segmented curve in this section.

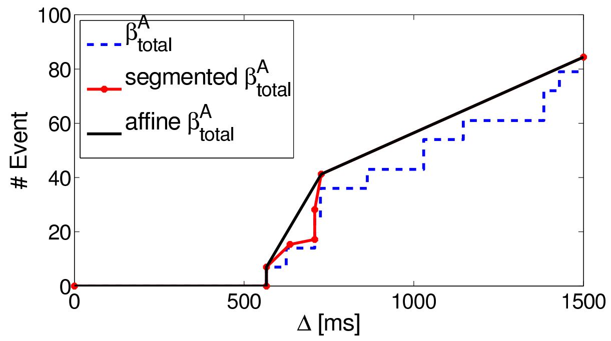


Fig. 6. Example for an affine segmented service curve from event streams S_3 and S_4 in Tab. I with EDF scheduling.

The pseudo code of the algorithm is shown in Algo. 3. For the starting point ν_i of each segment $\langle \nu_i, \rho_i \rangle$, the algorithm checks all subsequent segments and finds the maximal slope ρ_{max} for all ν_i and ν_j pair, $j \in [i+1, m]$, as shown in Lines 3–9. If $\rho_{max} > \rho_i$, we add a new segment $\langle \nu_i, \rho_{max} \rangle$ and skip segments from i to k (Line 13). Otherwise, we keep the current segment.

Algorithm 3 LSC-MSR

```
Input: the set of curve segments for service demand $\beta^A_{total}$
    $\mathcal{C} = \{\langle \nu_1, \rho_1 \rangle, \langle \nu_2, \rho_2 \rangle, \ldots, \langle \nu_m, \rho_m \rangle\}$
Output: the set of curve segments $\tilde{\mathcal{C}}$ for the affine $\beta^A_{total}$
 1: $i \leftarrow 2$;   $\tilde{\mathcal{C}} = \emptyset$
 2: while $i < m$ do
 3:     $\rho_{max} \leftarrow 0$;   $k \leftarrow 0$
 4:     for $j \leftarrow i+1$ to $m$ do
 5:         $\rho \leftarrow \frac{y_{\nu_j} - y_{\nu_i}}{x_{\nu_j} - x_{\nu_i}}$
 6:         if $\rho > \rho_{max}$ then
 7:             $\rho_{max} \leftarrow \rho$;   $k \leftarrow j$
 8:         end if
 9:     end for
10:     if $\rho_i > \rho_{max}$ then
11:         $\tilde{\mathcal{C}} \leftarrow \tilde{\mathcal{C}} \uplus \{\langle \nu_i, \rho_i \rangle\}$;   $i \leftarrow i+1$
12:     else
13:         $\tilde{\mathcal{C}} \leftarrow \tilde{\mathcal{C}} \uplus \{\langle \nu_i, \rho_{max} \rangle\}$;   $i \leftarrow k$
14:     end if
15: end while
```

An example for a two-stream set with EDF scheduling is shown in Fig. 6. As the figure shown, the affine β_{total}^A regains the monotonicity property.

V. Experimental Results

This section provides simulation results for solving the PPM minimization problem defined in (2) by applying the aforementioned LSC-BDA nad LSC-OPT algorithms. The implementation is based on the Matlab RTC-toolbox [26] and all results are obtained from a simulation host with an Intel 2.6 *GHz* processor and 3 *GB* RAM.

A. Experiment Setup

We take the stream set studied in [25], [9] for our case study. Table I specifies 10 event streams with different characteristics. An event stream S_i is specified by the PJD model with period p_i, jitter j_i, and minimal inter-arrival distance d_i. as defined in Section III. In additional, a worst-case execution time c_i is associated with S_i. The relative deadline D_i of S_i is defined as $D_i = \chi * p_i$ and varies according to the *deadline factor* χ. Given the PJD specification, the corresponding 3-segment linear curve is computed as $\{\langle(0,0),+\infty\rangle, \langle(0,1),\frac{1}{d}\rangle, \langle(\frac{\lceil\frac{j}{p}\rceil\cdot d\cdot p}{p-d}, \frac{(\lceil\frac{j}{p}\rceil+1)\cdot p-d}{p-d}), \frac{1}{p}\rangle\}$. In our simulations, we adopt the power profiles for four different processing units in [6], presented in Table II.

TABLE I
Event stream setting.

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}
p (msec)	198	102	283	354	239	194	148	114	313	119
j (msec)	387	70	269	387	222	260	91	13	302	187
d (msec)	48	45	58	17	65	32	78	-	86	89
c (msec)	12	7	7	11	8	5	13	14	5	6

TABLE II
Power profiles for processing units.

Device Name	P_a (W)	P_s (W)	P_σ (W)	t_{sw} (S)	E_{sw} (mJ)
Realtek Ethernet	0.19	0.125	0.085	0.01	0.8
Maxstream	0.75	0.1	0.05	0.04	7.6
IBM Microdrive	1.3	0.5	0.1	0.012	9.6
SST Flash	0.125	0.05	0.001	0.001	0.098

We simulate scenarios for both one event stream and multiple event streams. Due to space limitation, we only report random subsets of the 10-stream set for multiple event streams. $S(3,4)$, for instance, represents a scenario considering only event streams S_3 and S_4 in Table I. In the case of fixed-priority scheduling for multiple event streams, the priority corresponds to the stream index, e.g., S_3 has higher priority than S_4 for $S(3,4)$.

B. Results

First, we show the effectiveness of the LSC-BDA and LSC-OPT based approaches by deriving the minimum average idle power P defined in (2). Fig. 7 and Fig. 8 show the *normalized values* with respect to the minimums obtained from exhaustive searches. We explore single and multiple stream scenarios, subjected to different processing units and scheduling policies. In the cases of multiple event streams, Algorithm LSC-MSR

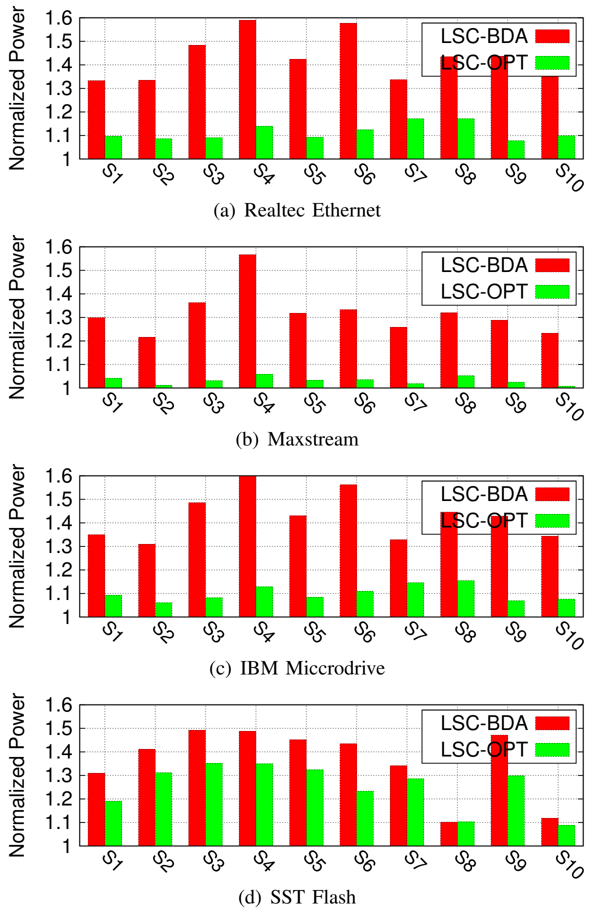
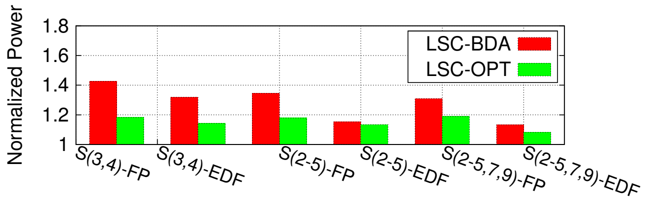


Fig. 7. Normalized minimal average idle power of the PPM problem for single-stream cases with deadline factor $\chi = 1.6$.

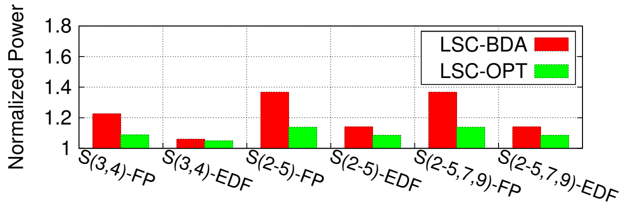
is applied. Otherwise, memory overflow will occur when a stream set contains more than 2 event streams. As the results shown in these two figures, the minimal idle power P derived by our new methods reasonably approximates the minimums with respect to different subset of the stream set, different processing units, and different scheduling policies.

In general, the LSC-BDA based approach performs better for multiple-stream scenarios than for single-stream scenarios. The reason is that with more event streams involved, the derived demand curve of the stream set is smoother in a linear shape, resulting in a closer match to the bounded delay function. The β^G derived from the bounded delay function approximates better the optimum accordingly.

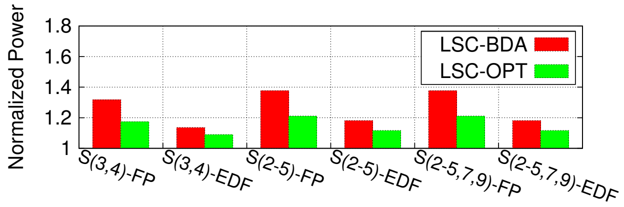
Second, we demonstrate the efficiency of all our approaches by reporting the computing time of computing the minimum P for different cases. The reported computing time is the mean value of ten separate runs. Fig. 9(a) depicts the computing time for different stream combinations, given a fixed deadline factor $\chi = 1.6$. Fig. 9(b) shows the relation of computation time and the deadline factor for stream set $S(3-6,9)$. As these two figures depicted, the computing time for our algorithms are within half an second for all cases. Even for long deadline



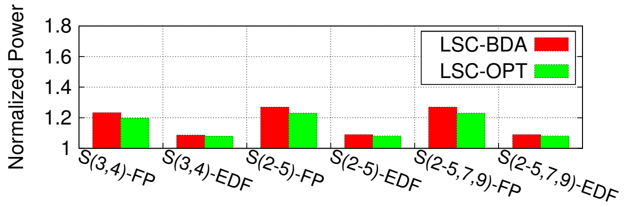
(a) Realtec Ethernet



(b) Maxstream

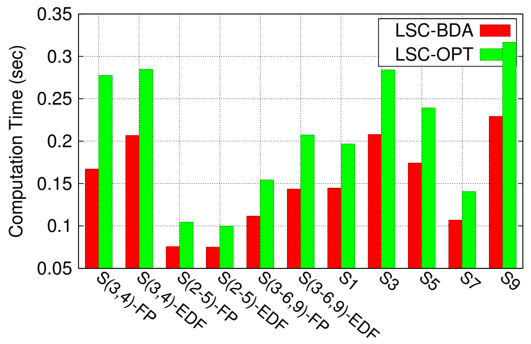


(c) IBM Miccrodrive

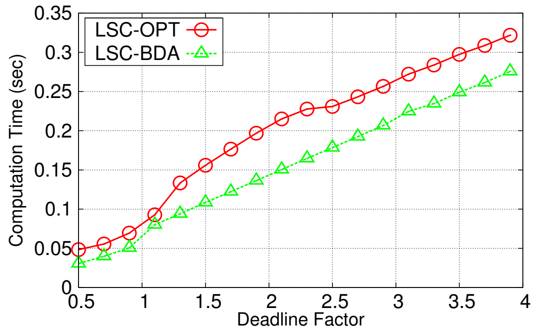


(d) SST Flash

Fig. 8. Normalized minimal average idle power of the PPM problem for multiple-stream cases with deadline factor $\chi = 2$.



(a) Deadline factor $\chi = 1.6$



(b) Stream set $S(3-6,\,9)$ with EDF scheduling

Fig. 9. Computation time of different scenarios for the IBM Microdrive for multiple streams.

settings, which will quadratically enlarge the search space, the computing time for both of our algorithms remains within a second. Note that the computing time can be further reduced, as current implementation is based on the Matlab RTC-toolbox which involves lots of Java-Matlab interactions. We expect much lower runtime for a pure C implementation. We also do not report the computing time for the exhaustive search which heavily depends on the search step as well as the complexity of the arrive curves. We have to linearize an arrival curve after ten periods for the cases of multiple event streams and set the search step to 1 millisecond. Otherwise, the exhaustive search will fall into memory overflow or need days to finish.

VI. Conclusion

This paper explores how to apply dynamic power management to reduce the energy consumption under the real-time constraints. We consider a system with active, standby, and sleep modes with different levels of power consumption. Based on a linear segmented representation of the arrival curve model, we develop two algorithms to derive optimal and approximated solutions for the minimization of the periodic power management problem, respectively. Extensions to multiple events streams are also presented. To demonstrate the performance of the proposed algorithms, several case studies are explored, in which the results reveal the effectiveness of our approaches. The reported computing time of our algorithms in different scenarios reveals that our algorithms have light run-time overhead and is applicable for systems that have limited power on computation. In the future, we would like to explore the feasibility of our approach for online power management as well as complex case studies which involve multiple processing cores(devices).

Acknowledgment

This work was supported by the European Project Pro3D (grant no. 248776).

References

[1] J. Augustine, S. Irani, and C. Swamy. Optimal power-down strategies. In *45th Symposium on Foundations of Computer Science (FOCS)*, pages 530–539, Oct. 2004.

[2] P. Baptiste. Scheduling unit tasks to minimize the number of idle periods: A polynomial time algorithm for offline dynamic power management. In *Proceedings of the 17th annual ACM-SIAM symposium on Discrete algorithm (SODA)*, pages 364–367, 2006.

[3] J.-J. Chen and T.-W. Kuo. Procrastination for leakage-aware rate-monotonic scheduling on a dynamic voltage scaling processor. In *ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)*, pages 153–162, 2006.

[4] J.-J. Chen and T.-W. Kuo. Procrastination determination for periodic real-time tasks in leakage-aware dynamic voltage scaling systems. In *Int'l Conf. on Computer-Aided Design (ICCAD)*, pages 289–294, 2007.

[5] J.-J. Chen, N. Stoimenov, and L. Thiele. Feasibility analysis of on-line dvs algorithms for scheduling arbitrary event streams. In *Proceedings of the 30th IEEE Real-Time Systems Symposium (RTSS)*, 2009.

[6] H. Cheng and S. Goddard. Online energy-aware I/O device scheduling for hard real-time systems. In *Proceedings of the 9th Design, Automation and Test in Europe (DATE)*, pages 1055–1060, 2006.

[7] R. Cruz. A calculus for network delay. I. network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, Jan 1991.

[8] V. Devadas and H. Aydin. Real-time dynamic power management through device forbidden regions. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 34–44, 2008.

[9] A. Hamann and R. Ernst. Tdma time slot and turn optimization with evolutionary search techniques. In *Proceedings of the 8th Design, Automation and Test in Europe (DATE)*, pages 312–317, 2005.

[10] K. Huang, L. Santinelli, J.-J. Chen, L. Thiele, and G. C. Buttazzo. Periodic power management schemes for real-time event streams. In *the 48th IEEE Conf. on Decision and Control (CDC)*, pages 6224–6231, Shanghai, China, 2009.

[11] K. Huang, L. Santinelli, J.-J. Chen, L. Thiele, and G. C. Buttazzo. Applying real-time interface and calculus for dynamic power manageme nt in hard real-time systems. *Real-Time Systems Journal*, 47(2):163–193, 2011.

[12] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 37–46, 2003.

[13] R. Jejurikar and R. K. Gupta. Procrastination scheduling in fixed priority real-time systems. In *Proceedings of the ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)*, pages 57–66, 2004.

[14] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *ACM/IEEE Design Automation Conference (DAC)*, pages 275–280, 2004.

[15] K. Lampka, S. Perathoner, and L. Thiele. Analytic real-time analysis and timed automata: A hybrid methodology for the performance analysis of embedded real-time systems. *Design Automation for Embedded Systems*, 14(3):193–227, 2010.

[16] J. Le Boudec and P. Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet.* Springer, 2001.

[17] J.-Y. Le Boudec. Application of network calculus to guaranteed service networks. *Information Theory, IEEE Transactions on*, 44(3):1087–1096, May 1998.

[18] A. Maxiaguine, S. Chakraborty, and L. Thiele. DVS for buffer-constrained architectures with predictable QoS-energy tradeoffs. In *the International Conference on Hardware-Software Codesign and System Synthesis (CODES+ISSS)*, pages 111–116, 2005.

[19] A. Maxiaguine, S. Künzli, and L. Thiele. Workload characterization model for tasks with variable execution demand. In *Proceedings of the 7th Design, Automation and Test in Europe (DATE)*, 2004.

[20] D. Park, J. Lee, N. S. Kim, and T. Kim. Optimal algorithm for profile-based power gating: A compiler technique for reducing leakage on execution units in microprocessors. In *Int'l Conf. on Computer-Aided Design (ICCAD)*, pages 361–364, 2010.

[21] S. Perathoner, K. Lampka, N. Stoimenov, L. Thiele, and J.-J. Chen. Combining optimistic and pessimistic dvs scheduling: An adaptive scheme and analysis. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 131 – 138, Nov. 2010.

[22] A. Shrivastava, E. Earlie, N. Dutt, and A. Nicolau. Aggregating processor free time for energy reduction. In *Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/-software codesign and system synthesis (CODES+ISSS)*, pages 154–159, 2005.

[23] V. Swaminathan and K. Chakrabarty. Pruning-based, energy-optimal, deterministic I/O device scheduling for hard real-time systems. *ACM Transactions in Embedded Computing Systems*, 4(1):141–167, 2005.

[24] L. Thiele, S. Chakraborty, and M. Naedele. Real-time calculus for scheduling hard real-time systems. In *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 4, pages 101–104, 2000.

[25] E. Wandeler and L. Thiele. Optimal TDMA time slot and cycle length allocation for hard real-time systems. In *11th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 479–484, 2006.

[26] E. Wandeler and L. Thiele. Real-Time Calculus (RTC) Toolbox. `http://www.mpa.ethz.ch/Rtctoolbox`, 2006.

[27] C.-Y. Yang, J.-J. Chen, C.-M. Hung, and T.-W. Kuo. System-level energy-efficiency for real-time tasks. In *the 10th IEEE International Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC)*, pages 266–273, 2007.

[28] C.-Y. Yang, J.-J. Chen, and T.-W. Kuo. Preemption control for energy-efficient task scheduling in systems with a dvs processor and non-dvs devices. In *the 17th IEEE Int'l Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 293–300, 2007.

[29] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 374–382, 1995.

[30] J. Zhuo and C. Chakrabarti. System-level energy-efficient dynamic task scheduling. In *Proceedings of the 42nd ACM/IEEE Design Automation Conference(DAC)*, pages 628–631, 2005.

APPENDIX

THM. 1. *U*sing the bounded-delay algorithm approach to compute $T_{on} = f(T_{off})$ as depicted in Algo. 1, $P(T_{on}, T_{off}) = P(f(T_{off}), T_{off})$ is a convex function.

Proof: The objective function $P(T_{on}, T_{off})$ can be split into two parts: $\frac{E_{sw}}{T_{on}+T_{off}}$ and $(P_s + P_\delta) \cdot \frac{T_{on}}{T_{on}+T_{off}}$. For the first part $\frac{E_{sw}}{T_{on}+T_{off}} = \frac{E_{sw}}{f(T_{off})+T_{off}}$, $f(T_{off}) + T_{off}$ is strictly increasing according to Lemma 1. Therefore $\frac{E_{sw}}{T_{on}+T_{off}}$ is a monotonically decreasing convex function. For the second part $(P_s + P_\delta) \cdot \frac{T_{on}}{T_{on}+T_{off}} = \frac{P_s+P_\delta}{1+\frac{T_{off}}{f(T_{off})}}$, we first prove $\frac{T_{off}}{f(T_{off})} > \frac{(1+\epsilon)T_{off}}{f((1+\epsilon)T_{off})}$ for any $\epsilon > 0$. Because $\rho_{\min, T_{off}} < \rho_{min, (1+\epsilon)T_{off}}$, we can derive $\frac{1}{1+\frac{T_{off}}{f(T_{off})}} < \frac{1}{1+\frac{(1+\epsilon)T_{off}}{f((1+\epsilon)T_{off})}}$, then, $\frac{T_{off}}{f(T_{off})} > \frac{(1+\epsilon)T_{off}}{f((1+\epsilon)T_{off})}$. Therefore, we have that $\frac{1}{1+\frac{T_{off}}{f(T_{off})}}$ is monotonically increasing and is convex as well. As a linear combination of convex functions is a still a convex function, the original function $P(T_{on}, T_{off})$ is a convex function of T_{off}. ■