Optimal DPM and DVFS for Frame-based Real-time Systems

MARCO E.T. GERARDS and JAN KUPER, University of Twente

Dynamic Power Management (DPM) and Dynamic Voltage and Frequency Scaling (DVFS) are popular techniques for reducing energy consumption. Algorithms for optimal DVFS exist, but optimal DPM and the optimal combination of DVFS and DPM are not yet solved.

In this paper we use well established models of DPM and DVFS for frame based systems. We show that it is not sufficient – as some authors argue – to consider only individual invocations of a task. We define a schedule that also takes interactions between invocations into account and prove – in a theoretical fashion – that this schedule is optimal.

Categories and Subject Descriptors: B.8.2 [Performance and Reliability]: Performance Analysis and Design Aids; D.4.1 [OPERATING SYSTEMS]: Process Management—Scheduling; D.4.7 [OPERATING SYSTEMS]: Organization and Design—Real-time Systems and embedded systems

General Terms: Algorithms, Performance, Reliability, Theory

Additional Key Words and Phrases: Dynamic Power Management, Dynamic Voltage and Frequency Scaling, Energy Minimisation

ACM Reference Format:

1. INTRODUCTION

In many systems, ranging from battery-operated embedded systems to heavily cooled supercomputers and data-centers, it is essential to reduce energy consumption. For battery-operated embedded systems battery life is an important quality metric, while for supercomputers and data-centers energy efficiency is expected to be one of the key purchasing arguments [Poess and Nambar 2008]. In this article, two important techniques are considered for energy reduction, namely Dynamic Power Management (DPM) and Dynamic Voltage and Frequency Scaling (DVFS).

With DPM, devices and/or the microprocessor are switched to a low power state when they are not used, resulting in a decreased energy consumption. Switching to a low power state has non-negligible time and energy overheads, hence switching to a low power state only takes place when the idle time of the processor is at least the break-even time. An overview of DPM techniques is given in the survey article by Benini
et al. [2000]. DPM is widely used since many computers have support for the Advanced Configuration and Power Interface (ACPI) [ACPI 2011]. The general problem of globally minimising energy for a real-time system by using DPM is NP-Hard [Devadas and Aydin 2008], we will show that for frame-based systems with concave energy costs for idle periods, the optimum can be found in constant time when all tasks have to execute the same amount of work, or in linear time otherwise.

DVFS [Weiser et al. 1996] allows the voltage and the clock frequency to be decreased, to trade time for energy. A lot of research is done in the area of DVFS. By considering the consumed energy as a cost function, while considering deadlines as constraints, a mathematical programming problem can be derived and the optimal clock frequencies – the solution to this problem – can be determined for many kinds of real-time systems, see e.g., [Yao et al. 1995; Huang and Wang 2009].

For the combination of DPM and DVFS, trade-offs between the two techniques should be considered. When DVFS is used, the clock frequency is decreased to reduce the energy consumption during the execution of tasks, while the execution time increases and the idle time decreases. Decreasing the clock frequency can reduce the length of an idle period to such extent that its length is below the break-even time. Hence, there is an interplay (see also [Devadas and Aydin 2012]) between DVFS and DPM that should be carefully considered when minimising the energy consumption using both techniques. Neither maximising the length of the idle period or minimising the clock frequency results in a guaranteed minimised energy consumption, instead a combination of both techniques is required, as is argued in [Devadas and Aydin 2012].

One particular example of a real-time system is a frame-based system (see, e.g., [Rusu et al. 2003; Devadas and Aydin 2012; Kong et al. 2010]) for which optimal DVFS [Rusu et al. 2003; Xu et al. 2007] and combinations of DVFS and DPM [Devadas and Aydin 2012; Kong et al. 2010] were studied. In contrast to prior work in this area which considers the problem for individual invocations, we consider the problem globally – rather than for individual invocations – for systems that can use both DVFS and DPM and give an analytical and easy to calculate (with polynomial time complexity) provably optimal solution to this problem. An example will be given to show that for DPM considering individual invocations can lead to maximisation of the costs, while by considering the global problem the costs can be minimised. The state of the art work by Devadas and Aydin [2012] shows experimentally that an optimisation approach can reduce the energy consumption significantly. Since our model is similar and we use several of their results, we focus on the theory of DPM and DVFS.

The contributions of this work are as follows.

— We present a schedule for a frame-based system (with multiple tasks) that globally minimises the energy consumption for DPM (with multiple devices) and the combination of DPM and DVFS where the interplay between DPM and DVFS is taken into account.
— A method is presented for modelling DPM scheduling decisions for multiple low power states in a graph, such that energy minimisation can be reduced to a shortest path problem. This approach is not limited to frame-based systems.
— Important general properties of schedules that optimally use DPM are given and proven.
— Optimal clock frequencies are given for both the case that the clock frequencies are chosen from an interval of clock frequencies and for the realistic case that the clock frequencies have to be selected from a finite set of clock frequencies.
— For all problems that are considered in this article, the schedules can be found in either constant or linear time. The optimal clock frequencies can be determined in polynomial time.
The remainder of the article is organised as follows. Related work is discussed by Section 2. In Section 3, the model of the system is discussed and the notation is introduced. Section 4 discusses DPM, general properties of optimal DPM and the optimal solution for frame-based systems, Section 5 discusses the optimal combination of DPM and DVFS. An evaluation of the theory is given in Section 6, where the energy savings that can be attained for real devices are shown. Section 7 concludes this article with a summary, conclusions and discusses future work.

2. RELATED WORK

DVFS has been used for more than a decade [Weiser et al. 1996]. In a recent empirical study [Sueur and Heiser 2010], the effectiveness of DVFS in modern systems is evaluated. The authors argue that when the idle time energy consumption is to be considered, DVFS is still effective, especially for embedded systems. Since for frame based systems idle times have to be considered, DVFS is still effective for such systems.

The state of the art work by Devadas and Aydin [2012] addresses energy minimisation using a combination of DVFS and DPM for frame-based systems. In this excellent work, the authors demonstrate the effectiveness of their models and algorithms experimentally. Their theory is used in our work related to DVFS (Subsection 5.2) to find the optimal clock frequencies in the case clock frequencies are chosen from a continuous interval. Devadas and Aydin [2012] implicitly assume that invocations of tasks start as soon as they arrive, this is a valid assumption since the authors study the interplay of DPM and DVFS for individual frames. In contrast we allow tasks to start at any time within the frame as long as their deadlines are met, which allows the global optimum to be found. Although scheduling becomes slightly harder, we show that by allowing tasks to start at any time within the frame the energy consumption can be significantly reduced.

The approach from [Devadas and Aydin 2012] produces a local minimiser, while our approach produces a global minimiser, leading to a lower overall energy consumption. In some situations, their algorithm cannot find any opportunities to use DPM and save energy that way, while our approach considers the problem globally and finds every opportunity to save energy and will never require more energy compared to the approach presented by Devadas and Aydin [2012]. In addition, we give a solution for the case that only a finite number of clock frequencies is available. A comparison between both approaches shows that our approach can reduce the energy consumption up to 50%.

Similar work was done in [Kong et al. 2010], where minimising the energy using DPM and DVFS in presence of multiple tasks that use multiple devices is modelled as an Integer Linear Program (ILP). In contrast to this work, we assume that all devices are active during the execution of any task and we do not solve an ILP but give an analytic solution that can provably be found in polynomial time. The combination of DVFS and DPM was studied in a stochastic setting by Xu et al. [2005], our approach is deterministic. Our results on DPM can also be applied in a stochastic setting.

In [Baptiste et al. 2007], an optimal DPM schedule for aperiodic jobs is found in polynomial time, but only works when there is only an active mode and a sleep mode. In our work, we do not restrict the number of low power modes and also optimise for DVFS.

We model the costs as a function of the length of the idle period. This function turns out to be piecewise-linear, increasing and concave, a similar model is used in [Augustine et al. 2004].

In [Devadas and Aydin 2008], it is proven that optimal DPM for periodic real-time systems without preemption is in general an NP-Hard problem. Methods are presented for decreasing the energy for task systems with rate-monotonic properties. By restricting
Marco E.T. Gerards and Jan Kuper

In our attention to frame-based systems with concave energy costs, the problem is no longer NP-Hard and an optimal solution can be found in polynomial time.

Several theoretical results for optimal DVFS are discussed in [Ishihara and Yasuura 1998]. We use this theory to determine the optimal clock frequencies in combination with the optimal schedule for DPM. While globally optimal DVFS for real-time systems is the main topic in several papers (e.g., [Yao et al. 1995; Huang and Wang 2009]), we present a globally optimal combination of DVFS and DPM for frame-based systems.

3. SYSTEM MODEL

3.1. Task Model

We first consider a system with $K$ periodic tasks, with $N$ invocations of each task and we will later show that many of our results still hold for infinitely many invocations. The tasks are frame-based, meaning that for a period $T$ the $n$-th invocation of each task does not start before the (shared) arrival time $a_n = (n−1)T$ and does not finish later than the deadline $d_n = nT$. This time interval in which the $n$-th invocations of all tasks are executed, given by $[(n−1)T, nT]$, is called a frame [Rusu et al. 2003; Devadas and Aydin 2012; Kong et al. 2010]. Although during each frame $K$ tasks are executed, for ease of notation we assume (without loss of generality) that there is a single task by grouping the work of all $K$ tasks. This does not have a negative impact on both DPM and DVFS as will be discussed later.

The work — the number of clock cycles — that is done by each invocation is bounded from above by the Worst Case Work (WCW), denoted by $W \in \mathbb{R}^+$. Invocation $n$ has to do $w_n \in (0, W]$ work. The execution time is given by $e_n \in (0, T]$ and depends on the work and the clock frequency, as will be discussed later.

We assume that for at least one invocation $n$ it holds that $e_n < T$, since otherwise DPM and DVFS cannot be used to reduce the energy consumption and the schedule is trivially determined. During the frame for invocation $n$, the processor is idle for a time $I_n = T − e_n$.

In contrast to [Devadas and Aydin 2012; Kong et al. 2010], we do not assume that the begin time of the execution of the $n$-th invocation coincides with arrival time $a_n$, and denote the begin time by $b_n \in [(n−1)T, nT]$. Note that if an invocation $n$ starts as early as possible and invocation $n+1$ starts as late as possible and both invocations execute without interruptions, the idle periods are adjacent, leading to one big idle period of length $I_n + I_{n+1}$ as depicted in Fig. 1.

3.2. Dynamic Power Management

We assume that during the execution of the task, the processor and all peripheral devices are active, while when no task is executed the processor and peripheral devices are not used and can be switched to a low power state. In total there are $M$ devices that have at least an idle state and a sleep state, although more low power states can be available. Device $m$ has $L_m$ low power states (i.e., deeper sleep states), where for each state $\ell$ the power consumption is $P_{m,\ell}$. For ease of notation, we assume that the idle state (the normal mode of operation during the idle period) is state 1 and for ease of notation the active state is denoted as state 0. When switching to state $\ell$ and back to
the idle state, $T_{m,\ell}$ time is spent (switching time), while the costs for both transitions are given by $E_{m,\ell}$. Since switching to a low power state and back costs both time and energy, a minimal idle time – called the break-even time – is required before this switch takes place because otherwise time or energy is wasted. Switching to state $\ell$ is only worthwhile whenever the idle time is at least $T_{m,\ell}$. Similarly, switching to a low power state $\ell$ should only be considered when the energy consumption does not increase due to switching. This is the case whenever the idle time is at least [Devadas and Aydin 2012]:

$$T_{m,\ell}^E = \frac{E_{m,\ell} - T_{m,\ell}P_{m,\ell}}{P_{m,1} - P_{m,\ell}},$$

which is the time before switching to a low power state saves more energy than it costs. Now the break-even time for low power state $\ell$ of device $m$ is given by:

$$B_{m,\ell} = \max \{T_{m,\ell}, T_{m,\ell}^E\}.$$

Hence, the break-even time is either the time it costs to switch to the low power state and back or the time it takes before the energy consumption of switching to the low power state is worthwhile, the biggest of the two is used. We assume (as in [Augustine et al. 2004] and discussed below) that $T_{m,\ell}^E \geq T_{m,\ell}$, i.e. the switching cost determines the break-even time.

Then the energy consumption of device $m$ as function of the idle time is given by:

$$E_{m,\ell}^\text{DPM}(\tau) = \min_{\ell \in \{1,\ldots,L_m\}} \left[ E_{m,\ell} + P_{m,\ell}(\tau - T_{m,\ell}) \right].$$

A similar model is used in [Augustine et al. 2004]. The functions $E_{m,\ell}^\text{DPM}$ are increasing piecewise-linear concave\(^1\), since a linear function is concave and the minimum of multiple concave functions is again concave [Corbae et al. 2009]. The sum over all devices determines the total energy consumed during the idle period, which is given by the following function:

$$E^\text{DPM}(\tau) = \sum_{m=1}^{M} E_{m,\ell}^\text{DPM}(\tau).$$

This expresses the energy that is consumed during the entire idle period as a function of the length of the idle period and is therefore called the idle time energy function. Since the sum of increasing piecewise-linear concave functions is again increasing piecewise-linear concave, the function that gives the total energy for the idle time is also increasing piecewise-linear concave. To ease the notation, we assume that the piecewise-linear function $E^\text{DPM}$ consists of $D$ pieces and can be described as:

$$E^\text{DPM}(\tau) = \min_{i \in \{1,\ldots,D\}} [\beta_i \tau + \alpha_i],$$

for some values $\alpha_i \geq 0$ and $\beta_i \geq 0$. We assume that $\alpha_1 = 0$ (hence, $E^\text{DPM}(0) = 0$), which models that the idle state can be used during the idle period.

In the derivation of the DPM model, it was assumed that the switching costs determine the break-even time, i.e. $T_{m,\ell}^E \geq T_{m,\ell}$, which is the case when the power for startup/shutdown is higher than or equal to the power during normal operation [Benini et al. 2000]. This means that the time that has to be spent in a low power state before

\(^1\)A function $f : C \to \mathbb{R}$ is concave if and only if for all $x, y \in C$ and $\lambda \in [0, 1]$: $f(\lambda x + (1 - \lambda)y) \geq \lambda f(x) + (1 - \lambda)f(y)$

ACM Transactions on Architecture and Code Optimization, Vol. 9, No. 4, Article 41, Publication date: January 2013.
Table I. Power consumption and break-even time for DPM capable devices

<table>
<thead>
<tr>
<th>Device</th>
<th>$P_{m,ℓ}$</th>
<th>$T_{m,ℓ}$</th>
<th>Break-even time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor node</td>
<td>1040/400/270/200/10 mW</td>
<td>5/15/20/50ms</td>
<td>8/20/25/50 ms</td>
</tr>
<tr>
<td>Harddisk (Hitachi DK23AA-60)</td>
<td>0.77/0.0 W</td>
<td>10.61s</td>
<td>24.41s</td>
</tr>
<tr>
<td>Network card (Linksys NP 100)</td>
<td>0.76/0.0 mW</td>
<td>2.75s</td>
<td>3.61s</td>
</tr>
<tr>
<td>Harddisk (IBM Ultrastar 36Z15)</td>
<td>10.2/2.5 W</td>
<td>12.4s</td>
<td>15.2s</td>
</tr>
<tr>
<td>Beowolf cluster node</td>
<td>1/0.766/0.1/0.1 $^2$</td>
<td>3/7/70s</td>
<td>6/10/100s</td>
</tr>
<tr>
<td>Laptop LCD</td>
<td>21.1/17.1 W</td>
<td>7.6s</td>
<td>15.6s</td>
</tr>
<tr>
<td>WLAN card</td>
<td>0.9/0 W</td>
<td>0.3s</td>
<td>0.7s</td>
</tr>
<tr>
<td>Ethernet card (WaveLAN)</td>
<td>1.43/0.05 W</td>
<td>0.34s</td>
<td>0.39s</td>
</tr>
</tbody>
</table>

Fig. 2. Concave idle time energy function ($E^{DPM}$) for sensor node.

Energy is saved is higher than the time it takes to switch to this low power state and back. Consider the devices in Table I, for all these devices this is actually the case, which shows this assumption holds for many devices that occur in practise. For ease of notation, we assume there is no idle period before the first invocation and after the last invocation, otherwise dummy invocations can be added to reflect this idle period.

**Example 3.1.** The sensor node as discussed in [Sinha and Chandrakasan 2001] has four power saving states as shown in Table I. In this example we use the characteristics from Table I together with some fictional workload to illustrate the trade-offs that are required for optimal DPM. The energy consumption for the idle time depends on the length of the idle period, as is illustrated in Fig. 2.

Consider the frame-based system with frame size $100\text{ms}$ ($T = 100$) and the workload $w_1 = 15$, $w_2 = 78$, $w_3 = 90$, $w_4 = 100$ (all in ms). When it is assumed (as in [Devadas and Aydin 2012]) that all tasks start at the beginning of the frame, the idle periods are as

$^2$normalised power
depicted as in Fig. 3a. The first idle period of 85ms is long enough for the deepest sleep state, the second idle period of 22ms which is long enough for the second low power state and for the third idle period of 10ms only the first low power state can be used. The total energy consumption is in that case 336.92mJ.

An alternative schedule is given shown in Fig. 3b, for which the total energy consumption is 330.01mJ. The energy is lower with respect to the previous schedule since instead of switching to low power states 1 and 2 for the two idle periods, these idle periods are merged to a single big idle period and only one transition to the low power state 3 is used.

A third possible schedule is given shown in Fig. 3c with the total energy consumption of 323.48mJ. Since two idle periods are merged, one transition to the deepest low power state and back can be avoided, which explains the decrease in energy. Although even more schedules are possible, it can be shown that this is the optimal schedule (see Section 4). This shows that the schedule influences the lengths of the idle periods and thus also the available low power states and how often a transition to low power states is made. When the idle period is long, a lot of energy can be saved as the number of transitions to low power states and the energy consumption do to these transitions overhead can be halved.

3.3. Dynamic Voltage and Frequency Scaling

Whereas with DPM the energy consumption during idle time is considered, with DVFS the focus is on energy spent during the execution of some task. Time can be traded for energy by changing the clock frequency. The clock frequency for an invocation \( n \) is given by a function that maps time to a clock frequency, this function is called the clock frequency function and is given by \( \phi_n : [0, e_n] \rightarrow \mathcal{F} \). Many papers (e.g., [Yao et al. 1995; Ishihara and Yasuura 1998]) take this function to be a constant function that can attain any value in the interval \( \mathcal{F} = [f_{\min}, f_{\max}] \), while other papers (e.g., [Kwon and Kim 2005]) use a finite set \( \mathcal{F} = \{\bar{f}_1, \ldots, \bar{f}_P\} \subset [f_{\min}, f_{\max}] \) of available clock frequencies.

The power – energy per time – is often modelled as a convex function (see e.g., [Yao et al. 1995]) \( p : [f_{\min}, f_{\max}] \rightarrow \mathbb{R}^+ \), while the energy consumption can also be modelled...
using the energy per work function \( \tilde{p} : [\tilde{f}_{\text{min}}, \tilde{f}_{\text{max}}] \to \mathbb{R}^+ \) which are related by \( \tilde{p}(f) = \frac{p(f)}{f} \) (see e.g., [Zitterell and Scholl 2010]). In addition, we require that \( \tilde{p}(f) \) is strictly convex. Since the function \( \tilde{p} \) is strictly convex, this function has a unique minimum \( \tilde{f}_{\text{min}} \) below which the function is decreasing and above which the function is increasing. Using any clock frequency below \( \tilde{f}_{\text{min}} \) instead of using \( \tilde{f}_{\text{min}} \) will increase the execution time and the energy consumption, this clock frequency \( \tilde{f}_{\text{min}} \) is called the critical frequency (see, e.g., [Jejurikar and Gupta 2004]). The permitted clock frequencies (i.e., the domain of \( \tilde{p} \) and thus \( \tilde{f}_{\text{min}} \) are chosen such that the function \( \tilde{p} \) is increasing on its domain.

Together with the amount of work, the clock frequency determines the execution time of each invocation. For an invocation \( n \), the execution time, clock frequency function and work are related by:

\[
w_n = \int_0^{\tilde{f}_n} \varphi_n(\tau) d\tau.
\]

4. DYNAMIC POWER MANAGEMENT

In this section, only scheduling for dynamic power management is discussed. Since DVFS is not yet applied, it will be assumed that the processor runs at the highest clock frequency, i.e., for each invocation \( i \), \( \varphi_i(\tau) = \tilde{f}_{\text{max}} \) and hence \( c_i = \frac{w_i}{\tilde{f}_{\text{max}}} \).

4.1. Properties of Optimal DPM

Since the idle time energy function \( E_{\text{DPM}} \) is concave, we can derive several properties of optimal solutions. In the given system model, the properties determine the optimal solution.

The first property shows that, instead of two idle periods, it is better to merge them to one single idle period.

**Lemma 4.1.** The function \( E_{\text{DPM}} \) is subadditive, i.e.,

\[
E_{\text{DPM}}(x + y) \leq E_{\text{DPM}}(x) + E_{\text{DPM}}(y).
\]

**Proof.** Recall that \( E(0) = 0 \), i.e., when there is no idle period there is no energy consumption. Then using the definition of concavity of \( E_{\text{DPM}} \):

\[
E_{\text{DPM}}(x) = E_{\text{DPM}} \left( \frac{x}{x + y}(x + y) + \frac{y}{x + y} 0 \right) \\
\geq \frac{x}{x + y} E_{\text{DPM}}(x + y) + \frac{y}{x + y} E_{\text{DPM}}(0).
\]

Similarly \( E_{\text{DPM}}(y) \geq \frac{y}{x + y} E_{\text{DPM}}(x + y) \). After adding these two inequalities, the result directly follows. \( \square \)

When a device is used for which the low power mode requires zero power (shutdown), merging the idle periods of length \( x \) and \( y \) halves the energy consumption when \( x \) and \( y \) are both bigger than the break-even time. The reason for this is that the shutdown and startup costs have to be taken into account only once, instead of twice.

The result of Lemma 4.1 and also the next result are not limited to frame-based systems, since they are properties of the DPM model and not of the task model (i.e., the theory holds for all task models). Whereas Lemma 4.1 merges two idle periods of length \( x \) and \( y \) to decrease the energy consumption, the next lemma shows that decreasing one idle period by \( \delta \) and increasing the other idle period by \( \delta \) also decreases the energy consumption (with \( \delta = x \) as a special case in Lemma 4.1).
**Lemma 4.2.** For $0 \leq \delta \leq x \leq y$:

$$E_{DPM}(x - \delta) + E_{DPM}(y + \delta) \leq E_{DPM}(x) + E_{DPM}(y).$$

**Proof.** Define $z_1 = x - \delta$, $z_2 = y + \delta$, then $z_1 \leq x \leq y \leq z_2$. Then for a certain $\lambda \in [0, 1]$ and $\mu \in [0, 1]: x = \lambda z_1 + (1 - \lambda) z_2$ and $y = \mu z_1 + (1 - \mu) z_2$. It can be readily checked that $\mu = 1 - \lambda$. Then by using Lemma 4.1:

$$E_{DPM}(x) = E_{DPM}(\lambda z_1 + (1 - \lambda) z_2) \geq \lambda E_{DPM}(z_1) + (1 - \lambda) E_{DPM}(z_2)$$

and

$$E_{DPM}(y) = E_{DPM}(\mu z_1 + (1 - \mu) z_2) = E_{DPM}((1 - \lambda) z_1 + \lambda z_2) \geq (1 - \lambda) E_{DPM}(z_1) + \lambda E_{DPM}(z_2).$$

Adding both inequalities gives:

$$E_{DPM}(x - \delta) + E_{DPM}(y + \delta) \leq E_{DPM}(x) + E_{DPM}(y).$$

This shows that for two idle periods (of length $x$ and $y$, with $x \leq y$) that are interrupted by the execution of a part of some task, it is better to unbalance the length of the idle periods by starting the execution earlier or later such that the smallest idle period (of length $x$) decreases in length, while the longest idle period (of length $y$) increases in length.

A direct result of this lemma is given by the following corollary.

**Corollary 4.3.** Given is a task of which an invocation is interrupted by an idle period. The energy consumption does not increase when the invocation is executed without interruptions.

**Proof.** This directly follows from Lemma 4.2.

This corollary gives the reason it was allowed to assume w.l.o.g. that there is a single task: if there are multiple tasks within a frame, it is best to execute them consecutively.

### 4.2. Non-variable Work

In general, determining an optimal schedule for a real-time system that exploits DPM is NP-Hard [Devadas and Aydin 2008]. By using Lemma 4.2 it can be shown that the optimal solution has certain properties which makes it easier to find the optimal solution. In specific situations – like for frame-based real-time applications – the following corollary can be used to determine the optimal schedule directly.

**Corollary 4.4.** For frame-based systems, there is an optimal schedule in which each invocation $n$ has either the begin time $b_n = a_n$ or $b_n = d_n - e_n$.

**Proof.** This follows directly from Lemma 4.2.

This corollary shows that for frame-based systems with $N$ invocations, each invocation has two possible begin times, reducing the number of possible schedules to $2^N$. A direct result is the following corollary.

**Corollary 4.5.** For any schedule given by $b_1, \ldots, b_N$ with $b_1 > a_1$, the costs do not increase when the begin time of invocation 1 is changed to $a_1$.

**Proof.** This is a direct result of Lemma 4.2.
For the special case that all invocations have the same amount of work, an optimal schedule can be determined by using the following theorem.

**Theorem 4.6.** For the frame-based system with for each invocation \( i \): \( w_i = W \) (i.e., \( e_i = \frac{W}{T_{max}} \)), the schedule implied by

\[
   b_n = \begin{cases} 
   a_n, & \text{if } n \text{ is odd;} \\
   d_n - e_n, & \text{if } n \text{ is even,}
   \end{cases}
\]

globally minimises the energy consumption.

**Proof.** The proof is omitted since this proof is a special case of Theorem 5.2 (to be discussed in Section 5).

This result holds for any number of invocations. Hence for an infinite number of invocations – for any \( n \) – the first \( n \) invocations are scheduled optimally.

The following examples show the implications of this schedule and the requirements to the invocations.

**Example 4.7 (Global and local optimisation).** Assume the period is given by \( T = 10 \), execution times are given by \( e_n = 5 \) for all \( n \), while the break-even time is given by \( B_{1,1} = 6 \). If each invocation starts when the frame begins, the idle periods are of length 5 and are shorter than the break-even time. When the optimal schedule (see Theorem 4.6) is used, the idle periods have length 10, which is longer than the break-even time. Hence, the schedule that executes the invocations at the beginning of each frame cannot switch to a low power state, while the optimal schedule can use the low power state for each idle period.

This illustrates that starting each invocation at the beginning of the frame – as is done by some papers that were discussed in Section 2 – leads to a global maximisation of the energy consumption, while global minimisation (using Theorem 4.6) can lead to a significant reduction of the energy consumption.

We have assumed that the function \( E^{\text{DPM}} \) is concave, the results from this section do not necessarily hold when this function is not concave. This is illustrated by the following example.

**Example 4.8 (Non-concavity of \( E^{\text{DPM}} \)).** Consider 3 invocations \( (N = 3) \) with \( T = 100 \), for all \( i \): \( e_i = 50 \) and hence \( I_i = 50 \). Then an optimal schedule is given by \( b_1 = 0, b_2 = 150, b_3 = 250 \), as shown in Fig. 4.

\[\text{Fig. 4. Schedule for non-concave function } E^{\text{DPM}}.\]
Now consider the same scenario, but now $E_{\text{DPM}}$ is not concave, but is given by (see Fig. 5):

$$E_{\text{DPM}}(\tau) = \begin{cases} 4\tau, & \text{if } \tau < 75; \\ 150 + 0.2\tau, & \text{if } \tau \geq 75. \end{cases}$$

The costs for the given schedule using this non-concave function $E_{\text{DPM}}$ are 370. However, note that Corollary 4.4 does not hold, hence in an optimal solution invocations might start elsewhere in a frame. In fact the unique optimal solution is given by $b_1 = 0$, $b_2 = 125$ and $b_3 = 250$ which has the costs 330. For this solution:

$$E_{\text{DPM}}(75) + E_{\text{DPM}}(75) \leq E_{\text{DPM}}(100) + E_{\text{DPM}}(50).$$

This shows that when $E_{\text{DPM}}$ is not concave, the schedule from Theorem 4.6 is not necessarily optimal.

### 4.3. Variable Work

For Theorem 4.6 it is used that for each invocation $n$, $w_n = W$. The importance of this property for optimality of the schedule given by Theorem 4.6 is demonstrated by the following example:

**Example 4.9.** Consider a task with period $T = 100$ and four invocations ($N = 4$) with the execution times $e_1 = 75$, $e_2 = 25$, $e_3 = 25$ and $e_4 = 75$. For demonstration purposes, we assume that the processor can operate in two states: an idle state and a sleep state. The idle time energy function is given by $E_{\text{DPM}}(\tau) = \min\{\tau, 100 + 0.2\tau\}$, where for the sleep state the break-even time is 125. When the schedule as suggested by Theorem 4.6 is used – and the preconditions for using the theorem are disregarded – one would get $b_1 = 0$, $b_2 = 175$, $b_3 = 200$ and $b_4 = 325$. As Fig. 6 shows, all idle periods are too small for switching to the sleep state, hence the energy consumption cannot be reduced by using DPM. However, the schedule given by $b_1 = 0$, $b_2 = 100$, $b_3 = 275$ and $b_4 = 325$ creates an idle period of length 150 between the second and the third invocation of the task, hence the energy consumption can be reduced using DPM (from 200 to 180). Note that the minimisation of the number of idle periods does not necessarily lead to a minimisation of the energy.
The invocations $n, \ldots, m$ choices. The weight of an edge $v_n$ is given by $v_n \in N$ introduced as sink vertex that models the completion of the computation, hence there represents the execution of invocation energy graph $n$ starts. An additional vertex $v_{N+1}$ is introduced as sink vertex that models the completion of the computation, hence there are $N+1$ nodes in the energy graph. Clearly, the vertices $V(G)$ of energy graph $G$ are given by $v_1, \ldots, v_N, v_{N+1}$.

The edges represent the scheduling decisions: an edge $v_n v_{n+1}$ implies that invocations $n$ and $n+1$ are scheduled as $b_n = a_n$ and $b_{n+1} = a_{n+1}$, while an edge $v_n v_{n+2}$ implies that invocations $n$, $n+1$ and $n+2$ are scheduled as $b_n = a_n$, $b_{n+1} = a_{n+1} - e_{n+1}$ and $b_{n+2} = a_{n+2}$. Note that, given the previous discussion, these are the only relevant scheduling choices. The weight of an edge $v_n v_m$ (denoted by $\omega_{n,m}$) is the energy consumption of the invocations $n, \ldots, m-1$, hence the edge $v_n v_{n+1}$ has weight $\omega_{n,n+1} = E^{\text{DPM}}(T - e_n)$.

![Fig. 6](image-url) Four invocations with variable work, $B_{1,1} = 125$. This shows that when not all invocations have the same work, the schedule that is suggested by Theorem 4.6 does not necessarily produce an optimal result. As mentioned before, there are $2^N$ candidates for the optimal solution. We will first reduce the number of candidates, before we give a procedure for finding the optimal solution for variable work in linear time.

Given an invocation $n$ and a begin time $b_n$, there are at most two choices for $b_{n+1}$. The following lemma shows that when $b_n = d_n - e_n$, the next scheduling choice follows directly:

**Lemma 4.10.** Given $b_n = d_n - e_n$, then using $b_{n+1} = a_{n+1}$ never costs more energy than using $b_{n+1} = d_{n+1} - e_{n+1}$.

**Proof.** There is an idle period of length $I_{n+1}$ between invocations $n$ and $n+1$. When $b_{n+1} = a_{n+1}$ is used as begin time, the idle period is only moved in time, the length of this idle period – or any other idle period – is not reduced. In fact, when $b_{n+2} = d_{n+2} - e_{n+2}$ the idle periods in frames $n+1$ and $n+2$ are adjacent, this can lead to a further reduction of the energy consumption. □

Given a scheduling choice ($b_n$) for invocation $n$, there is a limited number of choices for invocation $n+1$. First, if invocation $n$ starts at $b_n = a_n$, the next invocation begins at either $b_{n+1} = a_{n+1}$, with a small idle period for invocation $n$ of which the costs can be directly computed to be $E^{\text{DPM}}(T - e_n)$, or the invocation begins at $b_{n+1} = d_{n+1} - e_{n+1}$, leading to an idle period starting in frame $n$ and ending in frame $n+1$ for which the costs can be directly computed to be $E^{\text{DPM}}(2T - e_n - e_{n+1})$. Second, if invocation $n$ starts at $b_n = d_n - e_n$, we can assume (by Lemma 4.10) that the next invocation begins at $b_{n+1} = a_{n+1}$ and the costs for the idle period following invocation $n+1$ depend on when invocation $n+2$ starts.

We will transform the problem to a weighted directed acyclic graph $G$ – called the energy graph – in which the edges represent energy costs. A vertex $v_n$ ($1 \leq n \leq N$) represents the execution of invocation $n$ at time $b_n = a_n$. An additional vertex $v_{N+1}$ is introduced as sink vertex that models the completion of the computation, hence there are $N+1$ nodes in the energy graph. Clearly, the vertices $V(G)$ of energy graph $G$ are given by $v_1, \ldots, v_N, v_{N+1}$.
while the edge $v_nv_{n+2}$ has weight $\omega_{n,n+2} = E_{\text{DPM}}(2T - e_n - e_{n+1})$. The edges $E(G)$ of energy graph $G$ for $n \in \{1, \ldots, N\}$ are thus given by $v_nv_{n+1}$ and for $n \in \{1, \ldots, N-1\}$ there are edges given by $v_nv_{n+2}$.

Since we can assume w.l.o.g. (see Corollary 4.5) that the first invocation starts at time $b_1 = a_1$ (represented by $v_1$), the shortest path from $v_1$ to $v_{N+1}$ gives the minimal energy consumption, since the weighted length of each path from $v_1$ to $v_{N+1}$ represents the total energy consumption for all invocations.

The shortest path is easily determined using Dijkstra’s algorithm [Dijkstra 1959] which has a polynomial time complexity. Since our energy graph has a special structure (e.g., no cycles, a low degree, etc), Dijkstra’s algorithm can be reduced to Algorithm 1 and executed in linear time. In this algorithm, the costs of the shortest path up to vertex $v_n$ is given by $\text{costs}_n$ and the predecessor in the shortest path is given by $\text{pred}_n$, such that the shortest path can be easily reconstructed.

\begin{algorithm}
\textbf{costs}_1 = 0;
\textbf{for} $i = \{1, \ldots, N\}$ \textbf{do}
\begin{algorithmic}
  \STATE // Shortest path to $v_i$;
  \textbf{if} $i = 1$ \textbf{or} $\text{costs}_{i-1} + \omega_{i-1,i} < \text{costs}_{i-2} + \omega_{i-2,i}$ \textbf{then}
  \STATE $\text{costs}_i = \text{costs}_{i-1} + \omega_{i-1,i}$;
  \STATE $\text{pred}_i = i - 1$;
  \textbf{else}
  \STATE $\text{costs}_i = \text{costs}_{i-2} + \omega_{i-2,i}$;
  \STATE $\text{pred}_i = i - 2$;
\end{algorithmic}
\textbf{end}
\end{algorithm}

\textbf{Algorithm 1:} Shortest path for energy graph.

This procedure is illustrated using the following example:

\textbf{Example 4.11.} Consider the situation from Example 4.9, given by $T = 100, e_1 = 75, e_2 = 25, e_3 = 25$ and $e_4 = 75$. The idle time energy function is given by $E_{\text{DPM}}(\tau) = \min\{\tau, 100 + 0.2\tau\}$ and has as break-even time 125. An energy graph is created with 5 vertices: $v_1, \ldots, v_5$. The edges represent the energy consumption and are given as follows. When invocation 1 starts at $b_1 = a_1$, the energy consumption of the respective idle period is $E_{\text{DPM}}(T - e_1) = 25$ after which invocation 2 is executed at $b_2 = a_2$, this is modelled using the edge $v_1v_2$ with weight 25. Otherwise the idle time of invocations 1 and 2 together have an idle time $2T - e_1 - e_2$ and the energy consumption for the idle time of both invocations is $E_{\text{DPM}}(2T - e_1 - e_2) = 100$, this is modelled using the edge $v_1v_3$ with weight 100. The complete energy graph is depicted in Fig. 7. The shortest path is easily determined using Algorithm 1 and is given by $v_1v_2v_4v_5$. Each invocation $n$ of which the node is in the path is scheduled as $b_n = a_n$, hence $b_1 = 0, b_2 = 100, b_4 = 300$.

---

**Fig. 7.** Energy graph for Example 4.11.
When the node for invocation \( n \) is not in the path, \( b_n = d_n - e_n \), hence \( b_3 = 75 \). The energy consumption is the weighted length of the path, hence 180 energy is consumed.

Minimising the shortest path is not the same as maximising the length of the biggest the idle period, as it is illustrated by the following example.

**Example 4.12.** Take \( T = 100, e_1 = 35, e_2 = 25, e_3 = 25 \) and \( e_4 = 35 \) with the idle time energy function given by \( E^{DPM}(\tau) = \min\{\tau, 100 + 0.2\tau\} \), i.e., the break-even time is 125. Fig. 8 shows the respective energy graph. The biggest possible idle period (of length 130) can be created by using the edge \( v_2v_4 \), the only path from \( v_1 \) to \( v_5 \) containing this edge is \( v_1v_2v_4v_5 \) which has costs 260. The shortest path (of length 256) is \( v_1v_3v_5 \). Hence, maximising the biggest idle period does not necessarily lead to a minimised energy consumption.

Although the general problem of finding an energy optimal schedule is NP-Hard, several problem instances can be solved in polynomial time using a shortest path algorithm.

### 5. Dynamic Voltage and Frequency Scaling

Since the results on scheduling hold – as will be shown – independently of the clock frequency function, we initially assume that the clock frequency for each invocation \( n \) is given by a function \( \phi_n : [0, e_n] \rightarrow \mathcal{F} \) without requiring this function to be known. The combination of DPM and DVFS might seem counter-intuitive at first: neither maximising the idle period length or minimising the clock frequency will necessarily result in a minimisation of the energy. Instead, a simultaneous optimisation that considers both DPM and DVFS is required and local minima have to be considered.

#### 5.1. Non-variable Work

When all tasks have to do the same amount of work, it becomes easier to determine an optimal schedule. The following lemma helps in finding an optimal schedule.

**Lemma 5.1.** Assume the work for all invocations \( i \) is given by \( w_i = W \). Given is that \( b_n = a_n \) and \( b_{n+1} = a_{n+1} \) for some \( n \). When using the begin time \( b_{n+1} = d_{n+1} - e_{n+1} \) instead of using \( b_{n+1} = a_{n+1} \), clock frequency functions exist by which the energy costs do not increase.

**Proof.** Assume the clock frequency functions for invocations \( k, k+1 \) and \( k+2 \) are given by \( \varphi_A, \varphi_B \) and \( \varphi_C \) respectively. Now \( b_{n+1} \) is changed according to the lemma, while invocations \( k, k+1 \) and \( k+2 \) receive new clock frequency functions. Two situations have to be considered (by Corollary 4.4); in both situations the energy consumption does not increase:

(i) \( k + 2 > N \) or \( b_{k+2} = a_{k+2} \): In this situation, only the begin time of invocation \( k+1 \) changes, while the clock frequency function for each invocation remains the
same. Then the idle period that follows invocation \( k+1 \) — which is not followed by an idle period of a next task — now occurs before invocation \( k+1 \) as depicted (for \( b_{k+2} = a_{k+2} \)) in Fig. 9a. Since no idle period becomes smaller and all clock frequency functions remain the same, the energy consumption does not increase. Actually, an opportunity for further energy reduction might be created since the idle periods of invocations \( k \) and \( k+1 \) are adjacent.

(ii) \( k+2 \leq N \) and \( b_{k+2} = d_{k+2} - e_{k+2} \): In this situation, only the begin time of invocation \( k+1 \) changes, while the clock frequency functions of some invocations are interchanged as described below. In the old situation, there was an idle period of length \( I_k \) after the execution of invocation \( k \) and there is an idle period of length \( I_{k+1} + I_{k+2} \) between invocations \( k+1 \) and \( k+2 \). By using \( b_{k+1} = d_{k+1} - e_{k+1} \) and interchanging the clock frequency functions such that in the new situations for invocations \( k, k+1 \) and \( k+2 \) respectively the clock frequency functions \( \varphi_B, \varphi_C \) and \( \varphi_A \) are used, the situation as depicted in Fig. 9b is created. The idle periods and clock frequency functions of invocations \( k, k+1 \) and \( k+2 \) are now interchanged, such that the energy consumption remains the same.

\( \square \)

The problem of finding a schedule and clock frequencies that together minimise the energy consumption is a hard problem, since this requires simultaneous optimisation for both DPM and DVFS. For frame-based systems with non-variable work we will show that first optimising for DPM and then for DVFS will lead to a global minimum, this is stated in the following theorem.

**THEOREM 5.2.** Assume the frame-based system with for each invocation \( i \): \( w_i = W \). Take the begin times given by \( b_1, \ldots, b_N \) and clock frequency functions given by \( \varphi_1, \ldots, \varphi_N \). Then there are clock frequency functions \( \tilde{\varphi}_1, \ldots, \tilde{\varphi}_N \) such that together with the schedule implied by

\[
\tilde{b}_n = \begin{cases} 
  a_n & \text{if } n \text{ is odd;} \\
  d_n - e_n & \text{if } n \text{ is even,}
\end{cases}
\]

the energy consumption is not higher than the energy consumption of the schedule given by \( b_1, \ldots, b_N \) and \( \varphi_1, \ldots, \varphi_N \).

**PROOF.** For invocation 1, using \( \tilde{b}_1 = a_1 \) does not increase the energy consumption by Corollary 4.5. Now assume the theorem does not hold, only a schedule can be derived for which \( \tilde{b}_1 = b_1, \tilde{b}_2 = b_2, \ldots, \tilde{b}_k = b_k \) and \( \tilde{b}_{k+1} \neq b_{k+1} \) with \( k \in \{1, \ldots, N-1\} \) as high as possible. Then either:

\[\begin{array}{|c|c|c|}
\hline
\varphi_A & \varphi_B & \varphi_C \\
\hline
k & k+1 & k+2 \\
\hline
\varphi_A & \varphi_B & \varphi_C \\
\hline
k & k+1 & k+2 \\
\hline
\end{array}\]

(a) \( b_k = a_k, b_{k+1} = a_{k+1}, b_{k+2} = a_{k+2} \)

\[\begin{array}{|c|c|c|}
\hline
\varphi_A & \varphi_B & \varphi_C \\
\hline
k & k+1 & k+2 \\
\hline
\varphi_B & \varphi_C & \varphi_A \\
\hline
k & k+1 & k+2 \\
\hline
\end{array}\]

(b) \( b_k = a_k, b_{k+1} = a_{k+1}, b_{k+2} = d_{k+2} - e_{k+2} \)
is odd: Hence \( b_k = \tilde{b}_k = a_k \) and by Lemma 5.1, there is a schedule such that \( \tilde{b}_{k+1} = d_{k+1} - e_{k+1} \) while the begin times for invocations 1, \ldots, k remain the same. This contradicts that \( k \) was chosen as high as possible.

- \( k \) is even: Hence \( b_k = \tilde{b}_k = d_k - e_k \) and by Lemma 4.10, there is a schedule such that \( \tilde{b}_{k+1} = a_{k+1} \) while the begin times for invocations 1, \ldots, k remain the same. This contradicts that \( k \) was chosen as high as possible.

Since in both cases, the assumption that the theorem does not hold leads to a contradiction, it is proven that the theorem holds.

This shows that for the optimal combination of DVFS and DPM, one can take the optimal DPM schedule from Theorem 5.2 and then find the optimal clock frequency functions in a second step. Since the clock frequencies are either chosen from the set \([f_{\text{min}}, f_{\text{max}}]\) or from the finite set \(\{f_1, \ldots, f_F\} \subset [f_{\text{min}}, f_{\text{max}}]\), both cases are treated separately.

### 5.2. Optimal Continuous Clock Frequencies

In the state of the art work presented by Devadas and Aydin [2012], the power function is given by \( p(f) = Af^3 + P_{\text{static}} \) where \( A \) is the switching capacitance and \( P_{\text{static}} \) is the static power consumption. We use the same model for DVFS and use results from [Devadas and Aydin 2012] to find an analytic solution to our problem. If the clock frequencies have to be chosen from the set \([f_{\text{min}}, f_{\text{max}}]\), according to Ishihara and Yasuura [1998] there is an optimal solution for which \( \varphi_n \) is a constant function, i.e. \( \varphi_n(t) = f_n \) for some constant \( f_n \in [f_{\text{min}}, f_{\text{max}}] \).

As increasing the clock frequency decreases the active time, the active power for all devices should be taken into account. The active time energy consumption for invocation \( n \) with device \( m \) is determined as \( P_{m,0}(T - \frac{w_n}{f_n}) \). For ease of notation, we assume that for the remainder of this article that the active time energy consumption is part of \( E_{\text{DPM}} \).

Given the schedule from Theorem 5.2, there are two possible situations for each invocation \( n \). The first situation that is considered is that for some \( n \), invocation \( n \) starts at time \( a_n \) and invocation \( n+1 \) starts at time \( d_{n+1} - e_{n+1} \), hence the idle period of both invocations together have length \( 2T - e_n - e_{n+1} \).

For finding the optimal clock frequency, one should solve:

\[
\begin{align*}
\min_{f_n, f_{n+1} \in [f_{\text{min}}, f_{\text{max}}]} & \quad \bar{p}(f_n)w_n + \bar{p}(f_{n+1})w_{n+1} \\
& \quad + E_{\text{DPM}} \left( 2T - \frac{w_n}{f_n} - \frac{w_{n+1}}{f_{n+1}} \right),
\end{align*}
\]

subject to
\[
\begin{align*}
\frac{w_n}{f_n} & \leq T, \\
\frac{w_{n+1}}{f_{n+1}} & \leq T.
\end{align*}
\]

Either the minimum is attained inside the interior of the feasible region (the clock frequencies that are allowed by the constraints) and then the unconstrained problem has to be solved, or the minimum is attained on the boundaries of the feasible region. The unconstrained problem (i.e., only minimising the cost function) has a minimum for which \( f_n = f_{n+1} \) (by Ishihara and Yasuura [1998], by which the problem reduces to a single dimensional problem (since \( f_n = f_{n+1} \), leaving a single degree of freedom) which is solved by Devadas and Aydin [2012]. For finding a solution on the boundary, consider...
the cases for \(f_n = \min\{f_{\text{min}}, \frac{w_n}{T}\}\), \(f_n = f_{\text{max}}\), \(f_{n+1} = \{f_{\text{min}}, \frac{w_{n+1}}{T}\}\) and \(f_{n+1} = f_{\text{max}}\) which are minimal clock frequencies, maximal clock frequencies and the lowest clock frequencies by which the deadline is met. When looking for a solution on the boundary, one dimension of the problem is removed, reducing it to the problem that was solved in [Devadas and Aydin 2012]. Since there are five candidate solutions that can be analytically found in \(O(D)\) time using the theory from [Devadas and Aydin 2012], one can determine the costs for each of the five candidate solutions and use the one that minimises the costs.

In the second situation, frame \(n\) has an idle time of length \(I_n\) and is not adjacent to an idle period of another invocation (i.e., invocation \(n-1\) or invocation \(n+1\)). In that case the following optimisation problem has to be solved:

\[
\min_{f_n \in [f_{\text{min}}, f_{\text{max}}]} \bar{p}(f)w_n + E_{\text{DPM}}\left(T - \frac{w_n}{f_n}\right),
\]

subject to \(\frac{w_n}{f_n} \leq T\).

Again one can use [Devadas and Aydin 2012] to find a solution to this problem.

As the solution in [Devadas and Aydin 2012] can be found in \(O(D)\) time (i.e., it depends on the number of low power states for all devices), while the schedule is determined in \(O(1)\) time, the time complexity for finding a schedule that optimally combines DPM and DVFS is \(O(D)\). As finding the solution to these problems is already discussed in [Devadas and Aydin 2012], we will not repeated this. Instead we give a brief example to illustrate this approach.

**Example 5.3.** Consider a frame-based real-time system with period \(T = 10\), work \(w_n = 5\) and break-even time \(B_{1,1} = 0.4\). A single device is used with active power and idle power of 1 and when put to sleep the power consumption is 0.01. For illustration purposes, we assume the clock frequencies in the interval \([0.1, 1]\) are available and we use the power function \(p(f) = f^3\) for the processor.
In this example the second case is discussed, with idle period of length $I = 10 - \frac{5}{f_n}$ that is not merged with another idle period. The clock frequency should be at least $\frac{W}{T} = \frac{1}{2}$, otherwise the deadline will be missed. The energy consumption for DVFS as a function of the clock frequency is given by:

$$E^{\text{DVFS}}(f_n) = w_n f_n^2.$$  

To ease the optimisation, the energy consumption for DPM will also be expressed in terms of the clock frequency. Since $E^{\text{DPM}}$ is a function of the idle time, the idle time has to be determined in terms of the clock frequency, which is:

$$I(f_n) = T - \frac{w_n}{f_n}.$$  

The total energy as a function of the clock frequency,

$$E^{\text{DPM}}(I(f_n)) + E^{\text{DVFS}}(f_n),$$

is shown in Fig. 10. The idle time is long enough to switch to sleep mode when the clock frequency is at least $f^* = \frac{w_n}{T - B_{\text{1}}} \approx 0.52$, which is the peak in the figure. Clock frequencies below this clock frequency use DVFS only, above this clock frequency the device can be put to sleep mode during the idle period. Devadas and Aydin [2012] show that these two regions should be considered to find a minimum and give an analytic solution to this problem. Since the minimal clock frequency should be at least $\frac{1}{2}$, the clock frequency that is feasible and minimises the energy consumption is close to 0.8 as can be seen from inspection of Fig. 10. This shows that there is an interplay between DPM and DVFS: instead of choosing the lowest clock frequency that is allowed ($\frac{1}{2}$) or instead of using a clock frequency that maximises the idle period (1), a clock frequency is used that allows energy savings using both DPM and DVFS. For a thorough discussion and exact calculation of the optimal clock frequencies, we refer the interested reader to the work of Devadas and Aydin [2012].

### 5.3. Optimal Discrete Clock Frequencies

In case only $F$ clock frequencies are allowed, i.e. clock frequencies from the set $\mathcal{F} = \{f_1, \ldots, f_F\}$, a different procedure is deployed to determine the optimal clock frequencies. When $w_n$ work is executed, $r_{n,1} \geq 0$ work is executed at clock frequency $f_1$, $r_{n,2} \geq 0$ work is executed at clock frequency $f_2$, etc. In total, $\sum_{i=1}^{F} r_{n,i} = w_n$, hence all work is done. The solution determines the amount of work that is done at each clock frequency. As in determining the optimal continuous clock frequencies, two cases are considered.

The first cases is that for some $n$, invocation $n$ starts at time $a_n$ and invocation $n+1$ starts at time $d_{n+1} - e_{n+1}$ and the idle period of both invocations together have length $I_n + I_{n+1}$. For this cases: $I_n = T - \left[\sum_{i=1}^{F} \frac{r_{n,i}}{f_i}\right]$. The function $E^{\text{DPM}}$ is piecewise linear with $D$ pieces, with for piece $j$ the energy consumption for the idle period is determined.
as \( \beta_j I_j + \alpha_j \). Hence, given a piece \( j \), the optimal clock frequency is determined by solving:

\[
\min_{r_{n,1}, \ldots, r_n, F_{n+1,1}, \ldots, r_{n+1, F}} \left[ \sum_{i=1}^{F} \bar{p}(f_i)(r_{n,i} + r_{n+1,i}) \right] + \beta_j \left[ 2T - \sum_{i=1}^{F} \frac{r_{n,i} + r_{n+1,i}}{f_i} \right] + \alpha_j + \beta_0 \sum_{i=1}^{F} \frac{r_{n,i} + r_{n+1,i}}{f_i},
\]

subject to

\[
\sum_{i=1}^{F} r_{n,i} \leq T,
\]

\[
\sum_{i=1}^{F} r_{n+1,i} \leq T,
\]

\[
\sum_{i=1}^{F} r_{n,i} = w_n,
\]

\[
\sum_{i=1}^{F} r_{n+1,i} = w_{n+1},
\]

\[
r_{n,i} \geq 0, \text{ for all } i \in \{1, \ldots, F\},
\]

\[
r_{n+1,i} \geq 0, \text{ for all } i \in \{1, \ldots, F\}.
\]

Here the first term of the cost function is the energy consumption of the processor during the active period, the term with the constant \( \beta_0 \) is the active power of the devices during the active period and the energy during the idle period is given by the terms with \( \alpha_j \) and \( \beta_j \). This problem is a linear program, which can be solved in polynomial time. Solving the problem yields a candidate solution corresponding to piece \( j \), while all \( D \) pieces have to be considered.

The optimisation problem for the second case is given by:

\[
\min_{r_{n,1}, \ldots, r_n, F} \left[ \sum_{i=1}^{F} \bar{p}(f_i)r_{n,i} \right] + \beta_j \left[ T - \sum_{i=1}^{F} \frac{r_{n,i}}{f_i} \right] + \alpha_j + \beta_0 \sum_{i=1}^{F} \frac{r_{n,i}}{f_i},
\]

subject to

\[
\sum_{i=1}^{F} r_{n,i} \leq T,
\]

\[
\sum_{i=1}^{F} r_{n,i} = w_n,
\]

\[
r_{n,i} \geq 0, \text{ for all } i \in \{1, \ldots, F\}.
\]

This is also a linear problem, which can be solved in polynomial time. Since there are \( 2D \) candidate solutions for the first problem and \( D \) candidate solutions for the second problem, the costs for each candidate solution should be calculated and the feasible candidate with the lowest costs minimises the energy.

In Example 5.3, it was shown how to determine the optimal DPM and DVFS settings for continuous DVFS. The next example repeats this for a finite set of clock frequencies.

**Example 5.4.** Again consider the frame-based real-time system with period \( T = 10 \), work \( w_n = 5 \) and break-even time \( B_{1,1} = 0.4 \). A single device is used with active power and idle power of 1 and when put to sleep the power consumption is 0.10. The clock
frequencies $\mathcal{F} = \{0.1, 0.25, 0.5, 0.75, 0.9, 1\}$ are available and we again use the power function $p(f) = f^3$ for the processor.

As in Example 5.3, the energy consumption for optimal DPM can be determined if the clock frequency is fixed. Since not all clock frequencies in the interval $[0.1, 1]$ are available, the unavailable clock frequencies are determined by using their neighboring clock frequencies, an optimal standard approach [Ishihara and Yasuura 1998]. The average clock frequency that is obtained this way is referred to as the effective clock frequency. Fig. 11 shows the energy consumption as function of the effective clock frequency. The optimal clock frequency is now 0.75.

5.4. Variable Work
Note that Lemma 4.10 also holds when DVFS is used. Because of this, the structure of the energy graph for frame-based systems with DVFS is the same as the structure for frame-based systems without DVFS. To find the optimal schedule and clock frequencies, the local optimal costs that correspond to each edge can be determined by using the procedure that was discussed in Subsection 5.2 and Subsection 5.3. Where in Subsection 4.3 the optimal energy consumption for DPM was used as weight on the edges, with DPM and DVFS the energy consumption for DPM and DVFS together are used as weight.

For $w_n$ work and when only considering DPM, $E^{DPM}(T - \frac{w_n}{f_{\text{min}}})$ is used as the weight for an edge that encodes the costs of invocation $n$. When continuous DVFS is used, $\bar{p}(f_n)w_n + E^{DPM}(T - \frac{w_n}{f_n})$ is used as costs for this edge, however $f_n$ is not yet known. When constructing the graph, $f_n$ can be determined and the energy for invocation $n$ can be used as weight in the graph. After constructing this graph shortest path algorithm can be applied to find the schedule and clock frequencies that minimise the energy consumption. This last step works exactly as in Subsection 4.3.

6. EVALUATION
To evaluate optimal scheduling for DPM, we use the accurate measurements of the devices given in Table I and compare our method to the approach of Devadas and Aydin.
In our evaluation, we consider two situations to show what can be gained when the theory from Section 4 is applied. In the first situation, we assume all invocations of tasks start as soon as they arrive, as is assumed by, e.g., Devadas and Aydin [2012] and Kong et al. [2010], this situation is illustrated by the graphs “Beginning of frame” in Fig. 12. For the other situation we assume the tasks are scheduled according to our theory from Section 4, denoted by “Optimal schedule” in Fig. 12. Here, we assume many tasks are scheduled and thus the idle period can be merged into a bigger idle period, which enables significant energy savings. We use the average energy per frame, to make it possible to compare the energy savings to the first situation.

Fig. 12a shows the energy consumption of the sensor node from [Sinha and Chandrakasan 2001]. The break-even time for the first low power state is 5ms, which is shown for the situation where the invocations of tasks start at the beginning of the frame. When the tasks are scheduled optimally, the idle period of two frames can be combined and an idle period of 2.5ms in a single frame is combined to a bigger idle period of 5ms shared by both frames (again, see Fig. 1), making it possible to switch earlier to a low power state. Furthermore, by joining two idle periods, the costs for
switching to a low power state and back have to be taken into account only once. This explains why the costs for an idle period of length 60ms per frame is halved when the theory from Section 4 is used: the state transition only takes place in 50% of the frames, hence only 50% of the transition energy is required.

7. CONCLUSIONS

In this theoretical article, we discussed several general properties of optimal DPM. One important property – given the assumptions in this article – is that it is best to either start each invocation as soon as possible or as late as possible. For frame-based systems, these properties are used to find a globally optimal schedule that minimises the energy consumption using DPM. This is done for nonvariable work and for variable work. The approach of using an energy graph that was used for variable work is not limited to frame-based systems, but can be extended to various real-time systems where the execution order is predetermined.

In addition to optimal scheduling for DPM, we have shown how to combine DPM and DVFS optimally. We have proven that, when first optimising for DPM and then for DVFS, the energy is globally minimised. Simultaneous optimisation for clock frequency (DVFS) and idle period length (DPM) is required, since maximising the idle period length or minimising the clock frequency will not necessarily minimise the energy consumption.

Our work creates more opportunities to use DPM (i.e., to switch off devices) and will therefore use less energy. We have shown that in case the energy consumption in the low power state is zero (shutdown of the device), the idle time energy consumption is reduced by 50% with respect to the state-of-the-art. Future work will address optimal DPM for a broader class of real-time systems and optimal energy efficient scheduling of tasks with precedence constraints on multiprocessor systems.

ACKNOWLEDGMENTS

We would like to thank Ellen Franchimon, Robert de Groote and the four anonymous reviewers for their valuable feedback.

REFERENCES

Optimal DPM and DVFS for Frame-based Real-time Systems


Received June 2012; revised September 2012; accepted November 2012