Expected Energy Consumption Minimization in DVS Systems with Discrete Frequencies

Jian-Jia Chen
Computer Engineering and Networks Laboratory,
Swiss Federal Institute of Technology Zurich (ETH Zurich)
jjchencstw@gmail.com

ABSTRACT

Energy-efficiency has been an important system issue in hardware and software designs to extend operation duration or cut power bills. This research explores systems with probabilistic distribution on the execution time of real-time tasks for systems with discrete frequencies. Most previous studies consider DVS systems with continuous frequencies for the minimization of expected energy consumption under timing constraints. However, these approaches cannot guarantee the minimization of expected energy consumption when only discrete frequencies are available. This paper presents new approaches to minimize the expected energy consumption. By applying intra-task frequency scheduling, we develop an efficient algorithm to derive optimal frequency scheduling for a single task. The algorithm is then extended to cope with periodic real-time tasks with different power characteristics. With inter-task and intra-task frequency scheduling, we present a linear-programming approach to derive optimal solutions for frame-based real-time tasks and an on-line algorithm for periodic real-time tasks. Experimental results show that the proposed algorithms can effectively reduce the expected energy consumption.

Keywords: Dynamic Voltage Scaling, Probability, Expected Energy Consumption Minimization, Energy-Efficient Scheduling.

1. INTRODUCTION

Power management and energy consumption minimization are now important system design issues for modern computing systems. To balance the power/energy consumption and the performance of a system, dynamic voltage scaling (DVS) was introduced, in which different supply voltages lead to different execution frequencies.

For systems with real-time demands, energy-efficient scheduling has been studied to minimize the energy consumption with timing guarantee. The survey by Chen and Kuo [2] provides a comprehensive study for DVS scheduling. For real-time systems, energy-efficient scheduling is to minimize the energy consumption without making any task miss its deadline, provided that each task executes as its worst-case estimation. However, a task instance might complete earlier than its worst-case estimation, and the unused time (slack) might be reclaimed to reduce the energy consumption. Slack reclamation was studied to reduce energy consumption by using the slack left by some completed task instances, e.g., [1, 9].

In addition to worst-case estimations, profiling can also help system designers to get the distribution information of the workload of a task. With the probability distribution of workload, some previous studies [3, 4, 7, 11–13] provide DVS scheduling strategies to reduce the *expected* energy consumption when there are continuous processors frequencies. Lorch and Smith [7] derived an accelerating frequency scheduling by executing a task at a lower frequency at the beginning and at higher frequencies for the rest, while concurrent tasks were treated as joint workload. Gruian [3] considered the scheduling of multiple tasks and allocated execution time to tasks based on their worst-case execution cycles. Yuan and Nahrstedt [12] used the accelerating scheduling strategy for soft real-time multimedia tasks. Xu et al. [11] and Zhang et al. [13] explored inter-task scheduling for frame-based real-time tasks, in which all tasks have the same arrival time and deadline. Gruian and Kuchcinski [4] developed heuristics for task ordering to reduce the expected energy consumption.

However, processors, now, can only provide discrete frequencies. For systems with discrete frequencies, Xian and Lu [10] and Xu et al. [11] extended the accelerating scheduling by applying the research result by Ishihara and Yasuura [5] to use two adjacent frequencies to execute tasks. However, we will show that the derived solution is not optimal on the minimization of expected energy consumption. Moreover, Lu et al. [8] applied Lagrange Multiplier Method to cope with single-task systems with continuous probability density functions. When the probability density function of workload information of tasks is discrete, the approach by Lu et al. [8] is not applicable.

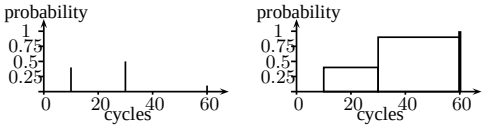
This paper presents new approaches to minimize the expected energy consumption for real-time tasks with discrete probability density functions of their workload information when there are only discrete frequencies available. Intra-task and inter-task scheduling scenarios are explored, in which the former considers frequency assignment during the execution of a task and the latter explores frequency assignment among tasks. For *intra-task* frequency scheduling, we develop an efficient algorithm to derive optimal frequency scheduling for a single task so that the expected energy consumption is minimized. The algorithm is also extended to cope with periodic real-time tasks that might have different power consumption characteristics due to different switching activities, while most previous work focused on tasks with the same power consumption characteristics. For *inter-task* frequency scheduling, we present a linear-programming approach to derive optimal solutions for frame-based real-time tasks when the execution order is specified, and develop an on-line algorithm for periodic real-time tasks. Experimental results show that the proposed algorithms can effectively reduce the expected energy consumption.

The rest of this paper is organized as follows: Section 2 provides system models and a motivational example. Section 3 presents the algorithm for a single task. Section 4 considers systems with multiple tasks. Experimental results of the proposed algorithms are presented in Section 5. Section 6 is the conclusion.

2. MODELS AND AN EXAMPLE

2.1 System Models

This research explores DVS systems with M frequencies, denoted by $f_1, f_2, \ldots, f_M$, where $f_1 < f_2 < \cdots < f_M$. The power consumption function at frequency f_i is $P(f_i)$. Distinct from many pre-



(a) Discrete PDF (b) Discrete CDF

Figure 1: An example for the probability functions of workload information of a task.

vious work [10, 11, 13] on processors with continuous frequencies for the assumption on $P(f_i)$ as f_i^3, this research can cope with $P(f_i)$ as an arbitrary function. When the system is idle, the power consumption of the system is assumed to be a constant α. Without loss of generality, this paper assumes $P(f_i) \geq \alpha$ for $i = 1, 2, \ldots, M$. Since the static power consumption is a constant, the power consumption function $P(f_i)$ is re-scaled by substituting α for each frequency f_i.

The number of CPU cycles executed in a time interval is linear in the processor frequency. That is, the number of CPU cycles completed in time interval $(t_1, t_2]$ is $\int_{t_1}^{t_2} f(t)\mathrm{d}t$, where $f(t)$ is the processor frequency at time t. The energy consumption in $(t_1, t_2]$ is $\int_{t_1}^{t_2} P(f(t))\mathrm{d}t$. Hence, executing a cycle at frequency f_j requires energy consumption $\frac{P(f_j)}{f_j}$. We only have to consider frequencies with $\frac{P(f_1)}{f_1} < \frac{P(f_2)}{f_2} < \cdots < \frac{P(f_M)}{f_M}$, since a schedule would not use a lower frequency with higher energy consumption [8].

2.2 Task Models

Tasks considered in this paper are periodic and independent in execution. A periodic task is an infinite sequence of task instances, referred to as *jobs*, where each job of a task comes in a regular period. Each task τ_i is associated with its period (denoted as p_i) and its computation requirement in CPU cycles in worst cases (denoted as c_i). The relative deadline of each task τ_i is equal to its period p_i. Given a task set $\mathbf{T}$, the *hyper-period* of $\mathbf{T}$, denoted by L, is defined as the minimum positive L so that L/p_i is an integer for any task τ_i in $\mathbf{T}$. We only consider task set $\mathbf{T}$ with $\sum_{\tau_i \in \mathbf{T}} \frac{c_i}{p_i f_M} \leq 1$, since there does not exist any feasible solution when $\sum_{\tau_i \in \mathbf{T}} \frac{c_i}{p_i f_M} > 1$.

The probability information of computation requirement of task τ_i is profiled as a discrete probability density function (discrete PDF) of execution cycle. For each task τ_i, the range $(0, c_i]$ is divided into K_i bins with different sizes. The j-th bin of task τ_i is associated with its amount of cycle $X_{i,j}$ and its probability density $\psi_i(j)$. That is, the probability for task τ_i with $\sum_{k=1}^{j} X_{i,k}$ cycles is $\psi_i(j)$. (Traditionally, profiling is done by using the same bin size to determine the probability. By eliminating the points with 0% probability during profiling, we can have bins with different sizes to reduce the input size.) The discrete cumulative density function (CDF) for task τ_i to have cycles no more than $\sum_{k=1}^{j} X_{i,k}$ is $\Psi_i(j) = \sum_{k=1}^{j} \psi_i(k)$. By definition, $\Psi_i(K_i) = 1$. For notational brevity, we define $\Psi_i(0)$ as 0. Therefore, the probability that the schedule has to execute the first $\sum_{k=1}^{j} X_{i,k}$ cycles of task τ_i is $1 - \Psi_i(j-1)$, denoted by $\Psi_i^\dagger(j)$. Figure 1 is an example of a task τ_i with $K_i = 3$, where $X_{i,1} = 10$, $X_{i,2} = 20$, and $X_{i,3} = 30$ with $\psi_i(1) = 0.4$, $\psi_i(2) = 0.5$, and $\psi_i(3) = 0.1$. Figure 1(a) is the discrete PDF, while Figure 1(b) is the discrete CDF. Hence, in the example, $\Psi_i^\dagger(1) = 1$, $\Psi_i^\dagger(2) = 0.6$, and $\Psi_i^\dagger(3) = 0.1$.

Tasks might have different power consumption characteristics due to different switching activities [6]. For task τ_i, its power consumption function at frequency f_j is denoted by $P_i(f_j)$. Similar to the argument in the last paragraph in Section 2.1, we only consider systems with $\frac{P_i(f_1)}{f_1} < \frac{P_i(f_2)}{f_2} < \cdots < \frac{P_i(f_M)}{f_M}$ for every task τ_i. For a set of real-time tasks with probability functions, this research explores the minimization of expected energy consumption under the specified timing constraints of real-time tasks.

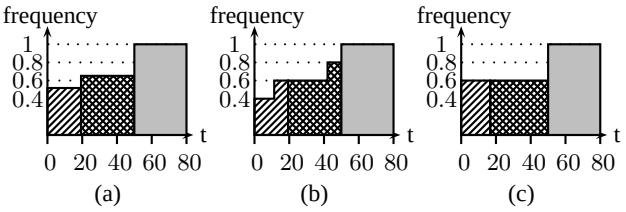


Figure 2: A motivational example for single-task scheduling.

2.3 A Motivational Example

Consider to schedule a single task τ_i with $p_i = 80$ and probability density function illustrated in Figure 1(a). The system has five frequencies: $0.2, 0.4, 0.6, 0.8$, and 1.0, where $P(f_j) = f_j^3$. By applying the algorithm proposed by Xian and Lu [10] or the algorithm proposed by Xu et al. [11], the system will first restrict the execution of the last 30 cycles at frequency 1, and then execute $X_{i,1}$ ($X_{i,2}$, respectively) cycles at frequency 0.518 (0.652, respectively) by treating that the processor has continuous frequencies in the range of $[0.2, 1]$, as shown in Figure 2(a). The expected energy consumption for such a case can be computed as $0.518^3 \frac{10}{0.518} + (1-0.4)0.652^3 \frac{20}{0.652} + (1 - 0.4 - 0.5)30 = 10.785$. By applying the research result by Ishihara and Yasuura [5] to use two adjacent frequencies for execution, $X_{i,1}$ will be executed at frequencies 0.4 and 0.6, and $X_{i,2}$ will be executed at frequencies 0.6 and 0.8, as shown in Figure 2(b). The expected energy consumption increases to 11.354. Although the approach works, it does not derive an optimal solution which minimizes the expected energy consumption. If we execute both $X_{i,1}$ and $X_{i,2}$ at frequency 0.6 as shown in Figure 2(c), the frequency assignment has expected energy consumption equal to $0.6^3 \frac{10}{0.6} + (1-0.4)0.6^3 \frac{20}{0.6} + (1-0.4-0.5)30 = 10.92$, which is better than the result in Figure 2(b).

Section 3 will present the derivation of optimal frequency assignments to minimize the expected energy consumption for a single task. Section 4 will extend the algorithm to systems with multiple tasks.

3. SINGLE-TASK SCHEDULING

This section provides an algorithm to derive a frequency assignment to minimize the expected energy consumption for a task. For notational brevity, let τ_1 be the task to be scheduled and D_1 be its available time for execution. The proposed algorithm first eliminates some frequencies that are energy-inefficient, and then iteratively reduces the expected energy consumption by slowing down the execution frequency.

Suppose that there exists a frequency f_j with $2 \leq j \leq M-1$ and

$$\frac{\frac{P_1(f_j)}{f_j} - \frac{P_1(f_{j-1})}{f_{j-1}}}{\frac{1}{f_{j-1}} - \frac{1}{f_j}} > \frac{\frac{P_1(f_{j+1})}{f_{j+1}} - \frac{P_1(f_j)}{f_j}}{\frac{1}{f_j} - \frac{1}{f_{j+1}}}.$$

That is, executing w cycles at frequency f_j is energy-inefficient, since executing $w\frac{\frac{1}{f_{j-1}} - \frac{1}{f_j}}{\frac{1}{f_{j-1}} - \frac{1}{f_{j+1}}}$ cycles at frequency f_{j+1} and $w\frac{\frac{1}{f_j} - \frac{1}{f_{j+1}}}{\frac{1}{f_{j-1}} - \frac{1}{f_{j+1}}}$ cycles at frequency f_{j-1} consumes less energy with the same execution time [8, Theorem 2]. Hence, the first step is to eliminate these frequencies. The elimination can be done intuitively in $O(K_1^2)$. Moreover, the remaining frequencies form a convex hull frontier so that the time complexity can be reduced to $O(K_1 \log K_1)$ by applying an efficient convex hull algorithm. Therefore, we can treat that

$$\frac{\frac{P_1(f_j)}{f_j} - \frac{P_1(f_{j-1})}{f_{j-1}}}{\frac{1}{f_{j-1}} - \frac{1}{f_j}} \leq \frac{\frac{P_1(f_{j+1})}{f_{j+1}} - \frac{P_1(f_j)}{f_j}}{\frac{1}{f_j} - \frac{1}{f_{j+1}}}, \quad \forall 2 \leq j \leq M-1 \quad (1)$$

after the above elimination procedure.

Initially, the frequency assignment executes all the computation of task τ_1 at the highest frequency. Then, slowing down the execution frequency reduces the expected energy consumption and stretches the

Algorithm 1 : GREEDY

Input: $\{\tau_1, D_1\}$;

1: **if** $\sum_{j=1}^{K_1} \frac{X_{1,j}}{f_1} \leq D_1$ **then**
2: execute task τ_1 at frequency f_1 all the time;
3: eliminate energy-inefficient frequencies so that Equation (1) holds;
4: $r_i \leftarrow M$, for $i = 1, 2, \ldots, K_1$;
5: $\theta \leftarrow \sum_{j=1}^{K_1} \frac{X_{1,j}}{f_M}$;
6: **while** $\theta < D_1$ **do**
7: let j^* be the index j with $r_j > 1$ and the greatest $\Psi_1^\dagger(j)\frac{\frac{P_1(f_{r_j})}{f_{r_j}} - \frac{P_1(f_{r_j-1})}{f_{r_j-1}}}{\frac{1}{f_{r_j-1}} - \frac{1}{f_{r_j}}}$;
8: **if** $\theta + X_{1,j^*}(\frac{1}{f_{r_{j^*}-1}} - \frac{1}{f_{r_{j^*}}}) < D_1$ **then**
9: $r_{j^*} \leftarrow r_{j^*} - 1$, $\theta \leftarrow \sum_{j=1}^{K_1} \frac{X_{1,j}}{f_{r_j}}$;
10: **else**
11: $x \leftarrow \frac{f^\dagger - f_{r_{j^*}-1}}{f_{r_{j^*}} - f_{r_{j^*}-1}}$, where $f^\dagger$ is $\frac{X_{1,j^*}}{D_1 - \theta + \frac{X_{1,j^*}}{f_{r_{j^*}}}}$;
12: return the frequency assignment by executing $X_{1,j}$ with $j \neq j^*$ at frequency f_{r_j} and X_{1,j^*} at frequency $f_{r_{j^*}-1}$ for $(1-x)X_{1,j^*}$ cycles and at frequency $f_{r_{j^*}}$ for $x \cdot X_{1,j^*}$ cycles;

worst-case completion time of the task. The basic idea behind the proposed algorithm is to slow down the execution frequency of some execution cycles so that the ratio of the reduced expected energy consumption to the increased execution time is the greatest. That is, suppose that $X_{1,j}$ cycles are assigned to be executed at frequency f_{r_j}, where $r_j = M$ at initialization. Let j^* be the index j with $r_j > 1$ and the greatest $\Psi_1^\dagger(j)\frac{\frac{P_1(f_{r_j})}{f_{r_j}} - \frac{P_1(f_{r_j-1})}{f_{r_j-1}}}{\frac{1}{f_{r_j-1}} - \frac{1}{f_{r_j}}}$. Reducing the execution frequency of X_{1,j^*} from frequency $f_{r_{j^*}}$ to frequency $f_{r_{j^*}-1}$ would be the most energy efficient. If the reduction of frequency makes the task violate its timing constraint, i.e., $(\sum_{k=1}^{K_1} \frac{X_{1,k}}{f_{r_k}}) + X_{1,j^*}(\frac{1}{f_{r_{j^*}-1}} - \frac{1}{f_{r_{j^*}}}) > D_1$, only some cycles in X_{1,j^*} are executed at frequency $f_{r_{j^*}}$ to satisfy the timing constraint. The detail is shown in Algorithm 1, denoted as Algorithm GREEDY. It is not difficult to see that τ_1 does not miss its deadline even in the worst case.

The example in Section 2.3 is used for demonstrating Algorithm GREEDY. First, because $P(f_j) = f_j^3$ is a convex and increasing function, there is no energy-inefficient frequency among the 5 frequencies. At initialization, $X_{1,1}$, $X_{1,2}$, and $X_{1,3}$ are assigned with f_5, in which r_1, r_2, and r_3 are 5, and θ is set as 60. Since 60 is less than deadline 80, Algorithm GREEDY enters the loop in Step 6-12 in Algorithm 1. The index j^* is 1 in this loop, and, hence, r_1 is set to 4, and θ is updated to 62.5. Because θ is still less than 80, the algorithm continues the loop, and finds j^* as 2. Then, r_2 is set to 4 and θ is updated to 67.5. The algorithms continues and finds j^* as 1 in the next loop, and sets r_1 as 3 and θ as 71.67. In the last loop, j^* is 2, and r_2 is set to 3 with θ as 80. Since θ is not less than deadline D_1, Algorithm GREEDY returns frequency assignment $(0.6, 0.6, 1)$ for $(X_{1,1}, X_{1,2}, X_{1,3})$, which is the frequency assignment in Figure 2(c) for task τ_1.

THEOREM 1. *Algorithm* GREEDY *derives an optimal frequency assignment to minimize the expected energy consumption for task τ_1 on a processor with discrete frequencies.*

PROOF. The theorem can be proved by showing that any feasible frequency assignment has no less expected energy consumption than the frequency assignment derived from Algorithm GREEDY. The proof is omitted due to the space limitation. □

4. MULTIPLE-TASK SCHEDULING

4.1 Intra-Task Scheduling

For intra-task scheduling, we have to determine the amount of time assigned to each task so that the resulting schedule will not make any job miss its deadline. It has been studied in the literature, such as [10, 12], with earliest-deadline-first (EDF) scheduling by setting a job with the earliest deadline as the highest-priority job. In [12], for systems with continuous frequencies, each task τ_i is allocated with $c_i/(\sum_{\tau_j \in \mathbf{T}} \frac{c_j}{p_j})$ amount of time for its execution. Then, for each job of τ_i, the allocated amount of time is used for single-task scheduling.

Algorithm GREEDY is revised into Algorithm M-GREEDY to provide optimal time allocation for intra-task scheduling. For periodic real-time tasks, EDF is an optimal scheduling algorithm. A task set can be scheduled without missing deadlines if and only if the system utilization is no more than 100%, in which the utilization of a task is defined as its execution time divided by its period.

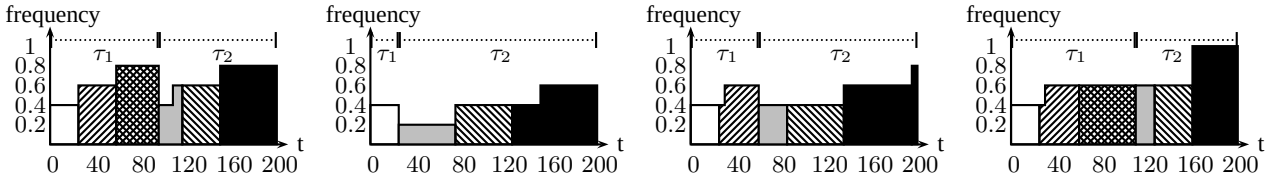
The idea behind Algorithm M-GREEDY is the same as Algorithm GREEDY. Initially, the frequency assignment executes all the computation of all the tasks in $\mathbf{T}$ at the highest frequency by setting $r_{i,j} = M$ for $\tau_i \in \mathbf{T}$ and $j = 1, 2, \ldots, K_i$. Then, slowing down the execution frequency reduces the expected energy consumption in the hyper-period and makes the system utilization greater. Similar to Algorithm GREEDY, we have to slow down the execution frequency of some execution cycles so that the ratio of the reduced expected energy consumption in the hyper-period L to the increased utilization is the greatest. Decreasing the execution frequency of $X_{i,j}$ from frequency $f_{r_{i,j}}$ to frequency $f_{r_{i,j}-1}$ increases $\frac{1}{p_i}(\frac{X_{i,j}}{f_{r_{i,j}-1}} - \frac{X_{i,j}}{f_{r_{i,j}}})$ utilization and decreases $\frac{L}{p_i}\Psi_i^\dagger(j)X_{i,j}(\frac{P_i(f_{r_{i,j}})}{f_{r_{i,j}}} - \frac{P_i(f_{r_{i,j}-1})}{f_{r_{i,j}-1}})$ amount of expected energy consumption in the hyper-period. Algorithm M-GREEDY greedily slows down the execution frequency of $X_{i,j}$ with $r_{i,j} > 1$ and the greatest $\Psi_i^\dagger(j)(\frac{P_i(f_{r_{i,j}})}{f_{r_{i,j}}} - \frac{P_i(f_{r_{i,j}-1})}{f_{r_{i,j}-1}})/(\frac{1}{f_{r_{i,j}-1}} - \frac{1}{f_{r_{i,j}}})$ until the system utilization is equal to 100% or all the tasks are executed at the slowest frequency. The detail is shown in Algorithm 2. Algorithm M-GREEDY can derive optimal time allocation for intra-task scheduling.

4.2 Inter-Task Scheduling

This paper has provided an algorithm for intra-task scheduling to allocate execute time to tasks in $\mathbf{T}$ in Section 4.1. However, if a job completes earlier than its worst-case estimation, the unused time is not used to reduce the energy consumption in intra-task scheduling. Hence, the energy consumption might not be minimized. Inter-task scheduling is to overcome the issue by reclaiming the unused time to reduce the expected energy consumption. This paper presents two methods for inter-task scheduling: One is for frame-based real-time tasks, in which all the tasks arrive at time 0 and are with the same period, i.e., $p_i = L$ for every task τ_i in $\mathbf{T}$, while another is for tasks with different periods.

4.2.1 Frame-Based Real-Time Tasks

This subsection explores tasks arriving at time 0 with the same period L. Suppose that the given tasks in $\mathbf{T}$ are indexed so that τ_i is executed right after τ_{i-1} for $i = 2, 3, \ldots, |\mathbf{T}|$. Suppose that we have two tasks τ_1 and τ_2 in $\mathbf{T}$, in which $X_{1,1} = 10, X_{1,2} = 20, X_{1,3} = 30$, $X_{2,1} = 10, X_{2,2} = 20, X_{2,3} = 40$, and $p_1 = p_2 = 200$. The PDFs of tasks τ_1 and τ_2 are $\psi_1(1) = 0.5$, $\psi_1(2) = 0.3$, and $\psi_1(3) = 0.2$, and $\psi_2(1) = 0.2$, $\psi_2(2) = 0.5$, and $\psi_2(3) = 0.3$, respectively. The system has five frequencies: $0.2, 0.4, 0.6, 0.8$, and 1.0, where $P_1(f_j) = P_2(f_j) = f_j^3$. By applying Algorithm M-GREEDY, we would have a frequency assignment shown in Figure 3(a). The expected energy consumption for intra-task scheduling in Figure 3(a) is



(a) frequency assignment by Algorithm M-GREEDY (b) optimal frequency assignment when τ_1 completes after 10 cycles (c) optimal frequency assignment when τ_1 completes after 30 cycles (d) optimal frequency assignment when τ_1 completes after 60 cycles

Figure 3: An illustrative example for scheduling 2 tasks

Algorithm 2 : M-GREEDY

Input: $\mathbf{T}$;

1: **if** $\sum_{\tau_i \in \mathbf{T}} \sum_{j=1}^{K_i} \frac{X_{i,j}}{p_i f_1} \leq 1$ **then**
2: execute task τ_i at frequency f_1 all the time for every task τ_i in $\mathbf{T}$;
3: eliminate energy-inefficient frequencies;
4: $r_{i,j} \leftarrow M$, for $j = 1, 2, \ldots, K_i$ and each task $\tau_i \in \mathbf{T}$;
5: $U \leftarrow \sum_{\tau_i \in \mathbf{T}} \sum_{j=1}^{K_i} \frac{X_{i,j}}{f_M \cdot p_i}$;
6: **while** $U < 1$ **do**
7: let (i^*, j^*) be the index (i, j) with $r_{i,j} > 1$ and the greatest $\Psi_i^\dagger(j) \frac{\frac{P_i(f_{r_{i,j}})}{f_{r_{i,j}}} - \frac{P_i(f_{r_{i,j}-1})}{f_{r_{i,j}-1}}}{\frac{1}{f_{r_{i,j}-1}} - \frac{1}{f_{r_{i,j}}}}$;
8: **if** $U + \frac{X_{i^*,j^*}}{p_{i^*}} (\frac{1}{f_{r_{i^*,j^*}-1}} - \frac{1}{f_{r_{i^*,j^*}}}) < 1$ **then**
9: $r_{i^*,j^*} \leftarrow r_{i^*,j^*} - 1$, $U \leftarrow \sum_{\tau_i \in \mathbf{T}} \sum_{j=1}^{K_i} \frac{X_{i,j}}{p_i f_{r_{i,j}}}$;
10: **else**
11: $x \leftarrow \frac{f^\dagger - f_{r_{i^*,j^*}-1}}{f_{r_{i^*,j^*}} - f_{i^*,r_{j^*}-1}}$, where $f^\dagger$ is $\frac{X_{i^*,j^*}}{p_{i^*}(1-U) + \frac{X_{i^*,j^*}}{f_{r_{i^*,j^*}}}}$;
12: return the frequency assignment by executing $X_{i,j}$ with $i \neq i^*$ and $j \neq j^*$ at frequency $f_{r_{i,j}}$ and X_{i^*,j^*} at frequency $f_{r_{i^*,j^*}-1}$ for $(1-x)X_{i^*,j^*}$ cycles and at frequency $f_{r_{i^*,j^*}}$ for $x \cdot X_{i^*,j^*}$ cycles;

23.96, while $X_{1,1}$, $X_{1,2}$, $X_{1,3}$ are executed at frequencies 0.4, 0.6, and 0.8, respectively.

If we use slack (which is the unused time of a job due to its earlier completion) reclamation, the expected energy consumption is as follows: If τ_1 completes after executing 10 cycles, τ_2 can execute in time interval $(25, 200]$, and the expected energy consumption in such a case can be computed as $0.4^2 \cdot 10 + 6.08 = 7.68$. If τ_1 completes after executing 30 cycles, τ_2 can execute in time interval $(58.34, 200]$, and the expected energy consumption in such a case can be computed as $0.4^2 \cdot 10 + 0.6^2 \cdot 20 + 8.48 = 17.28$. If τ_1 completes after executing 60 cycles, τ_2 can execute in time interval $(95.84, 200]$, and the expected energy consumption in such a case can be computed as $0.4^2 \cdot 10 + 0.6^2 \cdot 20 + 0.8^2 \cdot 30 + 16.04 = 44.04$. The above values 6.08, 8.48, and 16.04 are obtained by applying Algorithm GREEDY for τ_2 with execution times 175, 141.66, and 104.16, respectively. Therefore, the expected energy consumption for the task set can be computed as $7.68 \cdot 0.5 + 17.28 \cdot 0.3 + 44.04 \cdot 0.2 = 17.832$.

However, it will be better if more time is allocated to task τ_1. Consider the following frequency assignment: $X_{1,1}$ is executed at frequency 0.4, 10% of $X_{1,2}$ is executed at frequency 0.4, 90% of $X_{1,2}$ is executed at frequency 0.6, and $X_{1,3}$ is executed at frequency 0.6. Figure 3(b), Figure 3(c), and Figure 3(d) are the schedules when τ_1 completes after executing $10, 30$, and 60 cycles of τ_1, respectively. The expected energy consumption is 15.13, which is better.

Linear programming (LP). For clarity, we show the method by using two tasks as examples. Let $y_{j,m}$ be a variable between 0 and 1 to determine how much portion of cycles in $X_{1,j}$ of task τ_1 is executed at frequency f_m. Similarly, let $z_{j,k,m}$ be a variable between 0 and 1 to determine how much portion of cycles in $X_{2,k}$ of task τ_2 is executed at frequency f_m when τ_1 is known to execute $\sum_{\ell=1}^{j} X_{1,\ell}$ cycles only. When task τ_1 completes right after executing $\sum_{\ell=1}^{j} X_{1,\ell}$ cycles, the expected energy consumption is

$$\left(\sum_{\ell=1}^{j}\sum_{m=1}^{M} P_1(f_m) y_{\ell,m} \frac{X_{1,\ell}}{s_m}\right) + \sum_{k=1}^{K_2}\sum_{m=1}^{M} \Psi_2^\dagger(k) P_2(f_m) z_{j,k,m} \frac{X_{2,k}}{s_m}$$

under the timing constraint with

$$\left(\sum_{\ell=1}^{j}\sum_{m=1}^{M} y_{\ell,m} \frac{X_{1,\ell}}{s_m}\right) + \sum_{k=1}^{K_2}\sum_{m=1}^{M} z_{j,k,m} \frac{X_{2,k}}{s_m} \leq L.$$

The problem can be formulated as the following linear programming:

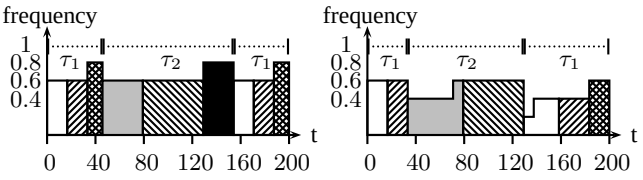
$$\begin{aligned}
\text{minimize} \quad & \textstyle\sum_{j=1}^{K_1} \psi_1(j) \cdot \Big(\Big(\sum_{\ell=1}^{j}\sum_{m=1}^{M} P_1(f_m) y_{\ell,m} \frac{X_{1,\ell}}{s_m}\Big) + \\
& \textstyle\sum_{k=1}^{K_2}\sum_{m=1}^{M} \Psi_2^\dagger(k) P_2(f_m) z_{j,k,m} \frac{X_{2,k}}{s_m}\Big) \\
\text{subject to} \quad & \textstyle\sum_{\ell=1}^{j}\sum_{m=1}^{M} y_{\ell,m} \frac{X_{1,\ell}}{s_m} + \sum_{k=1}^{K_2}\sum_{m=1}^{M} z_{j,k,m} \frac{X_{2,k}}{s_m} \leq L, \\
& \qquad \forall j = 1, \ldots, K_1, \\
& \textstyle\sum_{m=1}^{M} y_{j,m} = 1, \forall j \leq K_1, \\
& \textstyle\sum_{m=1}^{M} z_{j,k,m} = 1, \forall j \leq K_1, k \leq K_2, \\
& y_{j,m} \geq 0, \forall j \leq K_1, m \leq M, \text{ and} \\
& z_{j,k,m} \geq 0, \forall j \leq K_1, k \leq K_2, m \leq M.
\end{aligned} \tag{2}$$

Then, linear-programming solvers can be used to obtain an optimal frequency assignment for τ_1 and τ_2 in polynomial time. For the example in this section, $y_{1,2} = 1$, $y_{2,2} = 0.1$, $y_{2,3} = 0.9$, and $y_{3,3} = 1$, while $z_{1,1,1} = 1$, $z_{1,2,2} = 1$, $z_{1,3,2} = 0.25$, $z_{1,3,3} = 0.75$, $z_{2,1,2} = 1$, $z_{2,2,2} = 1$, $z_{2,3,3} = 0.9$, $z_{2,3,4} = 0.1$, $z_{3,1,3} = 1$, $z_{3,2,3} = 1$, $z_{3,2,5} = 1$, and all the other variables are 0.

The above linear programming can be extended to deal with any arbitrary number of tasks. However, the number of variables grows exponentially to the number of tasks. For n tasks, $M(\sum_{i=1}^{n}(\prod_{j=1}^{i} K_j))$ variables are needed to describe the linear programming. Therefore, the time complexity is exponential in the number of tasks. To overcome the issue, we show how to derive approximated solutions.

Approximated solutions . The LP formulation in Equation (2) has disadvantage on a large number of variables that are not used for task executions. At the time instant when τ_{i-1} completes, what we have to do is to determine the frequency assignment of task τ_i. Suppose that task τ_i is decided to execute for D_i time units. Then, we use Algorithm GREEDY to determine the execution frequency of task τ_i for D_i time units. Based on the frequency assignment, we can derive the expected energy consumption by applying slack reclamation for the rest of tasks. The best D_i is determined by applying a binary search so that the expected energy consumption is minimized.

Remarks on execution order. The best order for execution can only be achieved by finding the best order among all the enumer-



(a) A schedule derived by Algorithm M-GREEDY (b) The schedule with slack reclamation

Figure 4: An example for 2 periodic real-time tasks.

ated orders, and, hence, is time-consuming. Here, for frame-based real-time tasks, we use the same strategy in [11] for task ordering by sorting tasks τ_is according to $\frac{\sum_{k=1}^{K_i} \psi_i(k) X_{i,k}}{c_i}$ non-decreasingly.

4.2.2 Periodic Real-Time Tasks with Slack Reclamation

As shown in Section 4.2.1, the execution order of tasks affects the expected energy consumption. For periodic real-time tasks, a job might be preempted by another job with earlier deadline during execution. It is much complicated to model the execution of periodic real-time tasks. The algorithm for periodic real-time tasks adopts Algorithm M-GREEDY to determine the frequency assignment of task set $\mathbf{T}$. Suppose that task τ_i is assigned to have execution time e_i after applying Algorithm M-GREEDY in the worst cases. By Algorithm M-GREEDY, either $\sum_{\tau_i \in \mathbf{T}} \frac{e_i}{p_i} < 1$ by executing all the tasks at f_1 or $\sum_{\tau_i \in \mathbf{T}} \frac{e_i}{p_i} = 1$. Then, we execute tasks by applying the earliest-deadline-first policy. Once a job completes earlier than its worst-case estimation, the slack is used to slow down the to-be-executed job.

The slack reclamation scheme in [1, 9] can be adopted to calculate the available slack time that can be used for the to-be-executed job. Suppose that the EDF scheduler decides to have context switch at time t to execute a job J_i of task τ_k because a job finishes its execution at time t or job J_i is the job with the earliest deadline in the ready queue. The completed execution cycles for job J_i is χ_i before t. The total available time to execute job J_i at time t with the satisfaction of its timing constraint is $\hat{e}_i$. Then, we apply Algorithm GREEDY by taking the deadline of J_i as $\hat{e}_i$ for the remaining executions of job J_i. Since the available execution time $\hat{e}_i$ is longer than the execution time to execute the remaining $c_i - \chi_i$ cycles with the frequency assignment determined in off-line, we can use the frequency assignment derived in off-line as the initialized frequency assignment in Step 4 in Algorithm 1. The above method is denoted as Algorithm OM-GREEDY.

Here, we use an example to demonstrate Algorithm OM-GREEDY. Suppose that there are two tasks τ_1 and τ_2 in $\mathbf{T}$, in which $X_{1,1} = 10, X_{1,2} = 10, X_{1,3} = 10$, $X_{2,1} = 20, X_{2,2} = 30, X_{2,3} = 20$, $p_1 = 100$, and $p_2 = 200$. The PDFs of τ_1 and τ_2 are $\psi_1(1) = 0.5$, $\psi_1(2) = 0.3$, and $\psi_1(3) = 0.2$, and $\psi_2(1) = 0.2$, $\psi_2(2) = 0.5$, and $\psi_2(3) = 0.3$, respectively. The available frequencies and the power consumption functions of tasks τ_1 and τ_2 are the same as those in the example in Section 4.2.1. By applying Algorithm M-GREEDY, we would have a frequency assignment to execute both $X_{1,1}$ and $X_{1,2}$ at frequency 0.6 and execute $X_{1,3}$ at frequency 0.8 as shown in Figure 4(a). Suppose that τ_1 completes after executing 20 cycles at time $33.\bar{3}$. The available slack for task τ_2 is 12.5. By applying Algorithm GREEDY for task τ_2 with deadline $12.5 + 108.\bar{3} = 120.8\bar{3}$, 75% of $X_{2,1}$ will be executed at frequency 0.4, 25% of $X_{2,1}$ will be executed at frequency 0.6, $X_{2,2}$ will be executed at frequency 0.6, and $X_{2,3}$ will be executed at frequency 0.8. Suppose that τ_2 completes after executing 50 cycles at time $129.1\bar{6}$. The second instance of task τ_1 is executed by using the frequency assignment derived from Algorithm GREEDY with deadline $70.8\bar{3}$. The schedule is shown in Figure 4(b).

5. PERFORMANCE EVALUATIONS

This section provides the simulation setup and presents the simulation results on Intel XScale. There are five frequencies in Intel XScale: $(0.15, 0.4, 0.6, 0.8, 1)$GHz with corresponding power consumption $(80, 170, 400, 900, 1600)$mW. We assume that the power consumption of the processor when it is idle is 80mW.

Application	Description	Period	Worst-Case Cycle
mpegplay	MPEG video decoder	30 ms	10.5×10^6
madplay	MP3 audio decoder	30 ms	0.899×10^6
tmndec	H.263 video decoder	30 ms	12.7×10^6
toast	GSM speech encoder	25 ms	0.24×10^6

Table 1: Task parameters of multimedia applications

We evaluate both multimedia applications and synthetic real-time task sets. Four multimedia programs, *mpegplay*, *madplay*, *tmndec*, and *toast* are adopted for evaluations, in which their task parameters are shown in Table 1. All these applications except *toast* have the same period. The distributions of their required cycles are obtained by profiling their off-line traces.

We also perform evaluations for synthetic real-time tasks with *normal*, *uniform*, and *exponential* discrete probability density functions as suggested in [12, 13]. For a specified amount c_i of the worst-case execution cycle of a task τ_i, the probability density function for normal distribution is generated by setting the mean as a random variable in the range of $[\frac{c_i}{6}, \frac{5c_i}{6}]$ and the standard deviation as $\frac{c_i}{6}$. For exponential distribution, the mean is a random variable in the range of $[\frac{c_i}{6}, \frac{5c_i}{6}]$. For uniform distribution, the probability density function is $\frac{1}{c_i}$. Each synthetic task has 20 bins with the same size. For synthetic real-time tasks, the power consumption function $P_i(f_j)$ of task τ_i is $h_i \cdot P(f_j)$, where h_i is a random variable in the range of $[1, 2]$.

For single-task systems, we evaluate (1) the method by Xu et al. [11], denoted by Algorithm S-Rounding, to derive frequency assignments by assuming $P(s) = s^3$ (different from the algorithms in this paper) and then round to two consecutive frequencies, (2) the method by Xian and Lu [10], denoted by Algorithm S-Accelerating, to have an accelerating schedule after rounding to two consecutive frequencies, and (3) Algorithm GREEDY in Section 3. Both Algorithms S-Rounding and S-Accelerating take $O(K_i^2 \log K_i + K_i M)$ time complexity, while Algorithm GREEDY has $O(K_i M \log(K_i M))$ time complexity. For real-time systems with multiple tasks, we evaluate (1) the method by Yuan and Nahrstedt [12], denoted by Algorithm M-Rounding, (2) the method by Xian and Lu [10], denoted by Algorithm M-Accelerating, (3) Algorithm M-GREEDY, and (4) Algorithm OM-GREEDY, where the linear programming approach in Section 4.2.1 is evaluated for frame-based real-time tasks.

The schedule by executing all the tasks with 100% utilization at (at most) two consecutive frequencies is adopted as the baseline. The *normalized expected energy* is the expected energy consumption of the derived solution divided by that of the baseline schedule. The *normalized energy* is the actual energy consumption of the derived solution divided by that of the baseline schedule.

Figure 5 shows the normalized expected energy for systems with one task only. For a synthetic task, the worst-case execution cycle divided by its period is in $[0.4, 0.6]$. Since multimedia applications *madplay* and *toast* can be executed at the lowest frequency, all the evaluated methods would execute these two applications at the lowest frequency, and have the same result. Since Algorithm GREEDY is optimal for such a case, the expected energy consumption derived by Algorithm GREEDY is no more than the other evaluated algorithms for single-task scheduling. The improvement of algorithm GREEDY depends on the characteristics of the tasks. As shown in Figure 5, Algorithm GREEDY has about 3% improvement of Algorithm S-Accelerating on the normalized energy when the probability distribution function is normal distribution or uniform distribution, but the improvement is marginal for the *tmndec* application or when the probability density function is exponential distribution. The reason why the normalized energy is greater than 1 of Algorithms S-Rounding

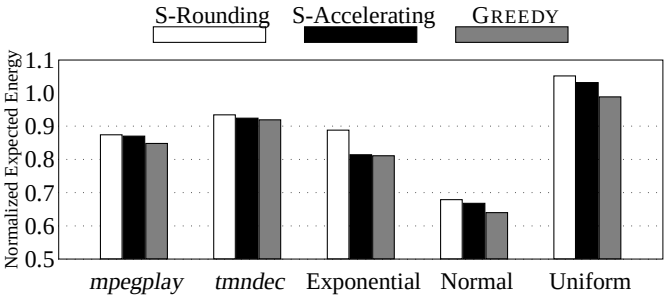


Figure 5: Experimental results for single-task scheduling.

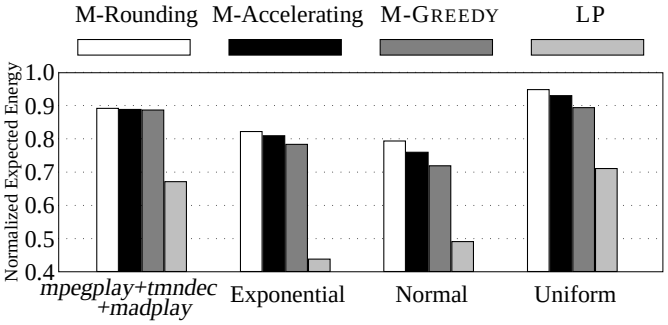


Figure 6: Experimental results for frame-based real-time tasks.

and S-Accelerating for the uniform distribution is because the incorrect estimation of power consumption by assuming $P(s) = s^3$, which might lead to a frequency assignment that is energy-inefficient for systems with discrete frequencies. As a result, we know that Algorithm GREEDY has better performance with less time complexity than both Algorithm S-Rounding and Algorithm S-Accelerating do.

Figure 6 shows the normalized expected energy for frame-based real-time tasks, in which *mpegplay*, *madplay*, and *tmndec* are considered in the multimedia benchmark, and synthetic task sets consist of 2 to 4 tasks with $\sum_{\tau_i \in \mathbf{T}} \frac{c_i}{p_i} = 0.6$ GHz. Algorithms M-Rounding, M-Accelerating, and M-GREEDY use the intra-task scheduling policy for frequency assignments. Since Algorithm M-GREEDY can derive optimal frequency assignments for the intra-task scheduling policy, Algorithm M-GREEDY, as shown in Figure 6, outperforms Algorithms M-Rounding and M-Accelerating. The reason why the improvement of Algorithm M-GREEDY is marginal in the multimedia application is because the high computation demand of the task set. To make the schedule feasible in worst cases, the processor has to execute more computation at higher frequencies, and, hence, the reduction on energy consumption is marginal. Moreover, applying the linear programming approach to determine the frequency assignment can greatly reduce the normalized expected energy consumption by about 20% to 30%, compared to Algorithm M-GREEDY with much more time complexity to derive the solution.

Figure 7 shows the normalized energy for periodic real-time tasks, in which each synthetic task set consists of 20 tasks. Applying Algorithm OM-GREEDY can reduce the normalized energy consumption by about 10% to 20%, compared to Algorithm M-GREEDY. As shown in Figure 6 and Figure 7, applying inter-task scheduling can significantly reduce the energy consumption. The proposed algorithms are effective in reducing (expected) energy consumption.

6. CONCLUSION

This paper presents new approaches to minimize the expected energy consumption for real-time tasks with discrete probability density functions of their workload information when there are only discrete frequencies available. For *intra-task* frequency scheduling, we develop an efficient algorithm to derive optimal frequency scheduling for a single task so that the expected energy consumption is minimized. For *inter-task* frequency scheduling, we present a linear-programming approach to derive optimal solutions for frame-based real-time tasks, and develop an on-line algorithm for periodic real-time tasks. Experimental results show that the proposed algorithms can effectively reduce the expected energy consumption. For future research, we would like to extend the approaches here to systems with considerations on peripheral devices.

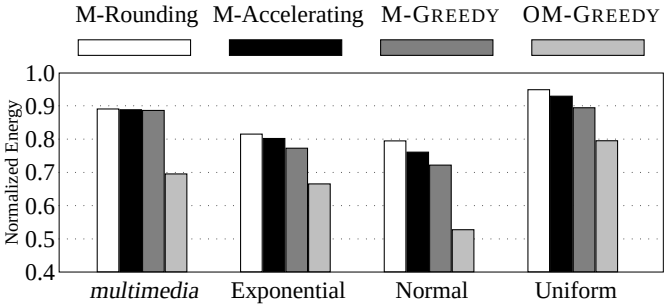


Figure 7: Experimental results for periodic real-time tasks.

Acknowledgment. The author would like to show the appreciation to Prof. Yung-Hsiang Lu and Mr. Changjiu Xian from Purdue University for their great help during preparing the manuscript.

References

[1] H. Aydin, R. Melhem, D. Mossé, and P. Mejía-Alvarez. Dynamic and aggressive scheduling techniques for power-aware real-time systems. In *Proceedings of the 22nd IEEE Real-Time Systems Symposium*, pages 95–105, 2001.

[2] J.-J. Chen and C.-F. Kuo. Energy-efficient scheduling for real-time systems on dynamic voltage scaling (DVS) platforms. In *RTCSA*, pages 28–38, 2007.

[3] F. Gruian. Hard real-time scheduling for low-energy using stochastic data and DVS processors. In *ISLPED*, pages 46–51, 2001.

[4] F. Gruian and K. Kuchcinski. Uncertainty-based scheduling: energy-efficient ordering for tasks with variable execution time. In *ISLPED*, pages 465–468, 2003.

[5] T. Ishihara and H. Yasuura. Voltage scheduling problems for dynamically variable voltage processors. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 197–202, 1998.

[6] W.-C. Kwon and T. Kim. Optimal voltage allocation techniques for dynamically variable voltage processors. In *Proceedings of the 40th Design Automation Conference*, pages 125–130, 2003.

[7] J. R. Lorch and A. J. Smith. Pace: A new approach to dynamic voltage scaling. *IEEE Trans. Computers*, 53(7):856–869, 2004.

[8] Z. Lu, Y. Zhang, M. Stan, J. Lach, and K. Skadron. Procrastinating voltage scheduling with discrete frequency sets. In *Design, Automation and Test in Europe (DATE)*, pages 456–461, 2006.

[9] P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, pages 21–24, 2001.

[10] C. Xian and Y.-H. Lu. Dynamic voltage scaling for multitasking real-time systems with uncertain execution time. In *ACM Great Lakes Symposium on VLSI*, pages 392–397, 2006.

[11] R. Xu, D. Mossé, and R. G. Melhem. Minimizing expected energy in real-time embedded systems. In *EMSOFT*, pages 251–254, 2005.

[12] W. Yuan and K. Nahrstedt. Energy-efficient soft real-time CPU scheduling for mobile multimedia systems. In *SOSP*, pages 149–163, 2003.

[13] Y. Zhang, Z. Lu, J. Lach, K. Skadron, and M. R. Stan. Optimal procrastinating voltage scheduling for hard real-time systems. In *DAC*, pages 905–908, 2005.