Dynamic Voltage Scaling for Multitasking Real-Time Systems with Uncertain Execution Time

Changjiu Xian
Department of Computer Science
Purdue University
West Lafayette, IN
cjx@cs.purdue.edu

Yung-Hsiang Lu
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN
yunglu@purdue.edu

ABSTRACT

Dynamic voltage scaling (DVS) for real-time systems has been extensively studied to save energy. Previous studies consider the probabilistic distributions of tasks' execution time to assist DVS in task scheduling. These studies use probability information for intra-task frequency scheduling but do not sufficiently explore the opportunities for inter-task scheduling to save more energy. This paper presents a new approach to integrate intra-task and inter-task frequency scheduling for better energy savings in hard real-time systems with uncertain task execution time. Our approach has two steps: (a) We calculate statistically optimal frequency schedules for multiple periodic tasks using earliest deadline first (EDF) scheduling for processors that can change frequencies continuously. (b) For processors with a limited range of discrete frequencies, we further present a heuristic algorithm to construct frequency schedules. Our evaluation shows that our approach saves up to 23% more energy than existing solutions.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: design studies

General Terms

Algorithms, Design, Performance

Keywords

Dynamic voltage scaling, hard real-time, probability, multitasking, low energy

1. INTRODUCTION

Energy consumption is an important design issue for battery-operated embedded systems. In these systems, the CPU is one major energy consumer. Dynamic voltage scaling (DVS) is an effective technique to reduce CPU energy because quadratic energy savings can be achieved with only linear decrease of the CPU frequency. Various DVS algorithms have been proposed, especially for real-time systems. A real-time task has a deadline that is the maximum time within which the task must completes its execution. A real-time scheduling algorithm can be either offline or online [1]. A "hard" real-time system does not allow missing any deadline. Consequently, for a task with uncertain execution time, the scheduler must consider the worst-case execution time (WCET) to guarantee meeting the deadline. Since this paper considers frequency and voltage scaling, we use "execution cycles" instead of execution time to express the workload of a task. The task's execution time is the execution cycles divided by the processor's frequency.

One solution to meet the deadline is assigning the processor a single frequency equal to the ratio of the worst-case execution cycles (WCEC) to the allowed execution time [8]. This can be too conservative if the average workload is much lighter than the worst case. Alternatively, the probability information of the workload can be used for better frequency scheduling. If a task requires variable numbers of cycles in different execution instances, the later cycles have lower probabilities to execute than the earlier cycles. For example, suppose three execution instances of a task require one million, two million, and three million cycles. Then the first million cycles execute three times, the second million cycles twice, and the third million only once. Some studies [2, 7, 10, 11] propose to assign lower frequencies to the earlier cycles and higher frequencies to the later cycles (called "accelerating frequency schedule") such that the deadline can still be met and the earlier cycles save energy. Even though the later cycles consume more energy when executing, they are rarely needed due to their lower probability. This can save more energy than assigning all cycles a single frequency.

The existing studies on accelerating frequency scheduling have two major limitations: (a) They use the probability information for only intra-task scheduling and do not sufficiently explore the opportunities for inter-task scheduling to save more energy. Inter-task scheduling determines how to allocate time for different tasks. (b) For processors with a limited range of discrete frequencies, the existing studies adjust the frequencies of each task individually to match the available frequencies without considering the effects of such adjustment on the other tasks.

This paper presents a new approach to integrate inter-task and intra-task scheduling in hard real-time multitasking systems with uncertain workloads. The key idea is combining inter-task and intra-task scheduling into a single optimization problem such that the frequency schedule of each task is determined by both its own probability distribution and the probability information from all other tasks. In other words, global information is used to build frequency schedules for individual tasks. Our approach has two steps: (a) We calculate statistically optimal frequency schedules for multiple periodic tasks using earliest deadline first (EDF) scheduling for processors that can change frequencies continuously. (b) For processors with a limited range of discrete frequencies, we further present a heuristic algorithm to construct frequency schedules based on global probability information and schedulability constraints. The computation of frequency schedules is offline and has no runtime overhead. Our evaluation shows that our approach saves up to 23% more energy than existing solutions.

2. BACKGROUND

2.1 Probability-Based DVS

Some studies have been conducted for DVS by considering the probability distributions of tasks' cycle demands. Lorch et al. [7] derive an accelerating frequency schedule for a single task and treat concurrent tasks as a single joint workload. Gruian [2] constructs accelerating intra-task schedules and allocates time to multiple tasks based on their WCEC. Yuan et al. [11] combine accelerating schedules with soft real-time constraints for multimedia applications. Xu et al. [10] study accelerating scheduling in systems with discrete frequencies. These studies use the probability information for only intra-task scheduling. In contrast, we can save more energy by combining the probability information from all tasks to integrate inter-task and intra-task scheduling. Zhang et al. [12] and Xu et al. [9] have considered multiple tasks in a special case where all periodic tasks share a common period such that all the tasks execute sequentially in each period and finish before the end of the period. Our approach allows different periods and uses EDF scheduling.

2.2 Accelerating Frequency Schedule

Suppose a task demands at most W cycles and the distribution of its cycle demand is expressed by the cumulative distribution function (CDF) shown in Figure 1 (a). The probability that the j^{th} cycle is needed is $P(j) = 1 - CDF(j-1)$, as shown in Figure 1 (b). Note that P is non-increasing because CDF is non-decreasing. Since a task may demand millions of cycles, it is impractical to store the distribution in cycles. Suppose the range $[0, W]$ is partitioned into M bins and each bin contains b cycles ($b = \lceil \frac{W}{M} \rceil$). The CDF is then a function of the bins, as shown in Figure 1 (d). The probability that the j^{th} bin is needed is $P(j) = 1 - CDF(j-1)$, as shown in Figure 1 (e).

Bins represent a more general form of granularity than cycles because we can choose 1 for b and make $M = W$. We introduce the accelerating frequency schedule based on bins. The frequency assigned to the j^{th} bin is f_j and the execution time for this bin is $\frac{b}{f_j}$. The processor's power is proportional to $v^2 f$ and $v \propto f$ (v: voltage). The energy for this bin is $(v_j^2 f_j) \times \frac{b}{f_j} \propto b f_j^2$. The expected energy consumption for this bin is proportional to the product of the energy and the probability: $b f_j^2 P(j)$. Suppose the task is released at time zero and the deadline is t. The goal is to find a schedule $\{f_1, f_2, ..., f_M\}$ to minimize the total expected energy. This is formulated as follows.

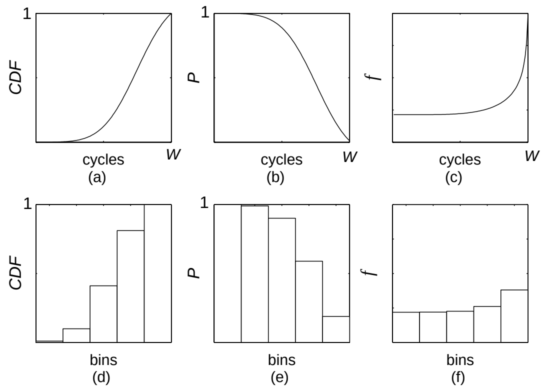

Figure 1: CDF, probability (P) of cycles or bins, and frequency schedule (f) using the granularity of (a)(b)(c) cycles and (d)(e)(f) bins.

$$\text{minimize} \sum_{1 \le j \le M} b f_j^2 P(j) \tag{1}$$

$$\text{subject to} \sum_{1 \le j \le M} \frac{b}{f_j} \le t \tag{2}$$

Based on earlier studies [7, 10, 11], the optimal solutions can be obtained by assigning f_j:

$$f_j = \frac{\sum_{j=1}^{M} b \sqrt[3]{P(j)}}{t \sqrt[3]{P(j)}} \tag{3}$$

This frequency schedule is shown in Figure 1 (c) for $b = 1$ and in Figure 1 (f) for $b \gg 1$. Both frequency schedules are non-decreasing. The reason is that $P(j) \ge P(j+1)$ so the denominator decreases and $f_j \le f_{j+1}$ in Equation (3).

3. INTEGRATED TASK SCHEDULING

This section presents our scheduling scheme for DVS. We first provide a motivational example to illustrate the basic concept. Then we explain how to integrate inter-task and intra-task scheduling to minimize the expected energy consumption in hard-real time systems with uncertain workloads. We extend the solution to processors with only a limited range of discrete frequencies in Section 3.4.

3.1 Motivational Example

Consider two periodic tasks $K1$ and $K2$, shown in Figure 2 (a), starting from the same time with periods $T_1 = 3s$ and $T_2 = 6s$. Both tasks have WCEC of 3 million cycles, divided into three equal-sized bins. Task K_1 always uses 3 million cycles (3 bins), so the probabilities of the three bins are all 100% (see Section 2.2 for the definition of such probability). Task K_2 has 90% probability to use 1 million cycles (the first bin), 5% probability to use 2 million cycles (the first two bins), and 5% probability to use 3 million cycles (all three bins). The probability that the first bin of K_2 is used is 100%, the second bin 10%, and the third bin 5%. Each task's deadline is the same as the task's period. We use EDF scheduling so K_2 executes after K_1's first instance. The second instance of K_1 has the same deadline as K_2 so it does not preempt K_2 and executes after K_2. We consider two different methods to assign frequencies so that both tasks can meet their deadlines.

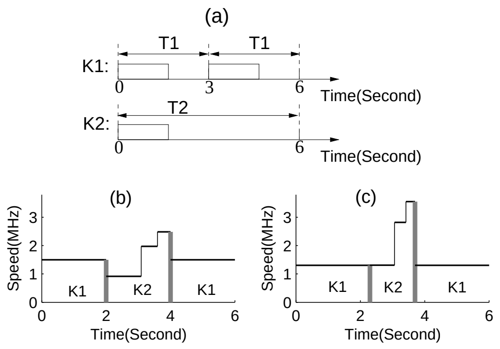

Figure 2: Frequency schedules from different time allocation methods.

The first method (shown in Figure 2 (b)) allocates the same amount of time, 2 seconds, to the three instances because they have the same WCEC. Within each 2 seconds, the optimal intra-task scheduling explained in Equation (3) is adopted. Figure 2 (b) shows the frequencies of the schedule. The frequency assigned to K_1 is the same across the 2-second duration because the actual cycle demand is the same as the WCEC. The frequency assigned to K_2 gradually increases. This solution is the state-of-the-art probability-based frequency scheduling for multiple tasks with EDF [11].

Our method (shown in Figure 2 (c)) allocates more time to K_1 than K_2 because K_1 requires more cycles than K_2 in most cases. Specifically, we allocate 2.3 seconds for K_1 and 1.4 seconds for K_2 and save 12% more energy than the first method. Section 3.3 explains how to obtain the duration assignments of 2.3 and 1.4 seconds by integrating inter-task and intra-task scheduling.

3.2 Task and System Model

We consider a set of periodic independent tasks denoted as $\{K_1, K_2, ..., K_N\}$. The tasks are preemptive. Task K_i has the following parameters: (a) T_i is its period. Each task has an execution instance per period. (b) W_i is its WCEC, i.e., the maximum number of cycles needed by one execution instance. The range $[0, W_i]$ is divided into M_i bins and each contains b cycles. (c) D_i is its deadline. Every execution instance must finish before its deadline. We assume $D_i = T_i$ in this study. (d) $P_i(j)$ is the probability that the j^{th} bin of cycles is needed in each period.

We first consider processors whose frequencies can be adjusted continuously from 0 to infinity. In Section 3.4, we consider processors with finite and discrete frequencies. The time to switch from one frequency to another is in the microseconds range [3, 11] and the execution time for tasks is normally the milliseconds range. Thus, we ignore the frequency switching time. We assume the processors consume both dynamic and static power during busy periods and only static power during idle periods. The dynamic power is proportional to $v^2 f$ and the static power is a constant. We assume DVS can adjust only dynamic power so we ignore the static power because it is constant throughout the whole duration. Overall, the energy per cycle is proportional to $v^2 f \frac{1}{f} \propto f^2$ since $v \propto f$. The scheduling is based on EDF.

3.3 Statistically Optimal Frequency Schedule

We integrate inter-task and intra-task scheduling such that the probabilities of the bins from all tasks are considered to build frequency schedules for individual tasks. The bins (from all tasks) with higher probabilities can use lower frequencies to save energy. For schedulability, the other bins may have to use higher frequencies. Even though these higher frequencies consume more energy, they are rarely needed due to their lower probabilities. Since a task with a shorter period executes more often than a task with a longer period, we divide task K_i's probability P_i by T_i to normalize the probability over time. The optimization problem is then to find a frequency f_{ij} for the j^{th} $(1 \le j \le M_i)$ bin of task K_i $(1 \le i \le N)$ to minimize the total expected energy of all tasks based on the bins' probabilities and under the schedulability constraint of EDF. EDF can schedule a set of tasks if the total CPU utilization is no more than one [6]. The mathematic formulation is as follows.

$$\text{minimize} \sum_{i=1}^{N} \sum_{j=1}^{M_i} b f_{ij}^2 \frac{P_i(j)}{T_i} \tag{4}$$

$$\text{subject to} \sum_{i=1}^{N} \frac{\sum_{j=1}^{M_i} \frac{b}{f_{ij}}}{T_i} \le 1 \tag{5}$$

Equation (4) is the total expected energy from all tasks, where $b f_{ij}^2$ and $\frac{P_i(j)}{T_i}$ are the energy consumption and the normalized probability of the j^{th} bin of task K_i, respectively. Equation (5) guarantees the schedulability of EDF, where $\sum_{j=1}^{M_i} \frac{b}{f_{ij}}$ is the worst-case execution time of task K_i. Note that Equation (5) implies that $\sum_{j=1}^{M_i} \frac{b}{f_{ij}} \le T_i$ and T_i is the deadline. Based on the convexity of f^2, Equations (4) and (5) can be solved based on Jensen's inequality [5]. All the optimization equations in this paper are solved with the same rationale. The derivation is omitted in this paper due to the space limit. The minimum expected total energy is

$$\left(\sum_{i=1}^{N} \frac{\sum_{j=1}^{M_i} b \sqrt[3]{P_i(j)}}{T_i} \right)^3 \tag{6}$$

The j^{th} bin of task K_i is assigned frequency f_{ij}:

$$f_{ij} = \frac{\sum_{i=1}^{N} \frac{\sum_{j=1}^{M_i} b \sqrt[3]{P_i(j)}}{T_i}}{\sqrt[3]{P_i(j)}} \tag{7}$$

The time (t_i) allocated to task K_i depends on the frequency schedule:

$$t_i = \sum_{j=1}^{M_i} \frac{b}{f_{ij}} = \frac{\sum_{j=1}^{M_i} \sqrt[3]{P_i(j)}}{\sum_{i=1}^{N} \frac{\sum_{j=1}^{M_i} \sqrt[3]{P_i(j)}}{T_i}} \tag{8}$$

The statistically optimal frequency schedules and the time allocated for each task can be computed from the periods (T_i), the bins (M_i, b), and the bins' probabilities $(P(i))$. Equation (7) indicates that the frequency for each task is accelerating because the dominator $\sqrt[3]{P_i(j)}$ decreases as j increases. Equation (8) is used by the second method in the motivational example (Section 3.1) to calculate the time allocation for the two tasks.

3.4 Finite and Discrete Frequencies

Section 3.3 assumes continuous frequencies from 0 to infinity. However, a practical processor has a maximum frequency f_{max} and a minimum frequency f_{min}. Meanwhile, the frequencies between f_{min} and f_{max} are discrete. Equation (7) has no guarantee to satisfy this constraint. This

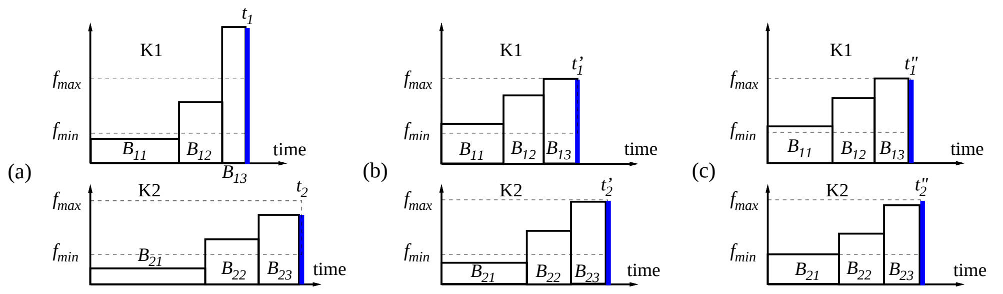

Figure 3: Steps to confine frequency schedules between $[f_{min}, f_{max}]$: (a) Compute schedules for tasks K_1 and K_2 for continuous frequencies. B_{ij} means the j^{th} bin of cycles for task K_i. (b) Assign f_{max} to B_{13} and recompute the frequencies for the remaining bins: B_{11}, B_{12}, B_{21}, B_{22}, B_{23}. (c) Assign f_{min} to B_{21} and recompute the frequencies for the remaining bins: B_{11}, B_{12}, B_{22}, B_{23}.

section presents a heuristic algorithm to adjust the frequencies within f_{min} and f_{max} based on global probability information and schedulability constraints. We first assume the frequencies are continuous within $[f_{min}, f_{max}]$ and then restrict the frequencies to be discrete.

Figure 3 (a) shows the frequency schedules for two tasks K_1 and K_2 using continuous frequencies from Equation (7). The frequencies of the first bins (B_{11} and B_{21}) are lower than f_{min}. The frequency for B_{13} is higher than f_{max}. Task K_1 is allocated time t_1 and task K_2 is allocated time t_2 (Equation (8)). We first consider reducing the frequencies higher than f_{max}, specifically, the frequency of B_{13}. Since we always assign higher frequencies to the bins with lower probabilities, B_{13} must have a probability lower than the other bins. Decreasing the frequency of bin B_{13} will prolong its execution time. Due to schedulability constraints, at least one of the other bins has to shorten its execution time and consequently use a higher frequency. Since all the other bins have higher probabilities than B_{13}, their frequencies should be kept low to minimize the expected energy. Hence, we decrease the frequency of B_{13} to only f_{max} for minimum increasing of other bins' frequencies.

To decrease B_{13}'s frequency to f_{max}, two methods can adjust the frequencies of the other bins to keep schedulability. The first adjusts only task K_1 within its allocated time t_1. Namely, bins B_{11} and B_{12} raise their frequencies to use less time such that the three bins of K_1 still use time t_1. This method does not consider the opportunity for global adjustment. The second method recomputes the frequency schedule for all the remaining bins from both tasks. Time t_1' and t_2' depend on the new frequency schedules. This is illustrated in Figure 3 (b). B_{13} is assigned f_{max} and the frequencies of the remaining bins of both tasks are adjusted.

This concept can be generalized to multiple bins whose frequencies are higher than f_{max}. Let H_i denote the number of bins that are assigned frequencies higher than f_{max} for task K_i from Equation (7). We assign f_{max} to these bins and and use the following formulation to calculate the frequencies of the other bins $1 \le j \le M_i - H_i$ from all tasks.

$$\text{minimize} \sum_{i=1}^{N} \sum_{j=1}^{M_i - H_i} b f_j^2 \frac{P_i(j)}{T_i} \tag{9}$$

$$\text{subject to} \sum_{i=1}^{N} \frac{\frac{H_i b}{f_{max}} + \sum_{j=1}^{M_i - H_i} \frac{b}{f_{ij}}}{T_i} \le 1 \tag{10}$$

The minimum energy is achieved if

$$f_{ij} = \frac{\sum_{i=1}^{N} \frac{\sum_{j=1}^{M_i - H_i} b \sqrt[3]{P_i(j)}}{T_i}}{\sqrt[3]{P_i(j)} \left(1 - \sum_{i=1}^{N} \frac{\frac{H_i b}{f_{max}}}{T_i}\right)} \tag{11}$$

If this new frequency schedule for the $1 \le j \le M_i - H_i$ bins still contains frequencies higher than f_{max}, we assign f_{max} to these bins and recursively perform the above adjustment for the remaining bins until all bins have frequencies no greater than f_{max}.

We next handle the bins with frequencies lower than f_{min}. In Figure 3 (b), B_{21}'s frequency is lower than f_{min}. Because we always assign lower frequencies to bins with higher probabilities, B_{21} must have a higher probability than the other bins. The frequency of B_{21} should be kept low to minimize the expected energy. Hence, we assign f_{min} to B_{21}, as illustrated in Figure 3 (c). Since this shortens the execution time of bin B_{21}, the spare time can be assigned to the other bins to lower their frequencies. We redistribute such spare time to the remaining bins of all tasks by recomputing their frequency schedules.

Let L_i denote the number of bins with frequencies lower than f_{min} for task K_i. We determine the frequencies for the remaining ($L_i + 1 \le j \le M_i - H_i$) bins using the following formulation.

$$\text{minimize} \sum_{i=1}^{N} \sum_{j=L_i+1}^{M_i - H_i} b f_j^2 \frac{P_i(j)}{T_i} \tag{12}$$

$$\text{subject to} \sum_{i=1}^{N} \frac{\frac{L_i b}{f_{min}} + \frac{H_i b}{f_{max}} + \sum_{j=L_i+1}^{M_i - H_i} \frac{b}{f_{ij}}}{T_i} \le 1 \tag{13}$$

The minimum energy is achieved if

$$f_{ij} = \frac{\sum_{i=1}^{N} \frac{\sum_{j=L_i+1}^{M_i} b \sqrt[3]{P_i(j)}}{T_i}}{\sqrt[3]{P_i(j)} \left(1 - \sum_{i=1}^{N} \frac{\frac{L_i b}{f_{min}} + \frac{H_i b}{f_{max}}}{T_i}\right)} \tag{14}$$

After the previous procedure, all bins have frequencies between f_{min} and f_{max}. We next consider available discrete frequencies. A simple method replaces each frequency with its two adjacent discrete frequencies [2, 4]. However, this method can waste energy when multiple bins have their frequencies between the same two adjacent frequencies. Figure

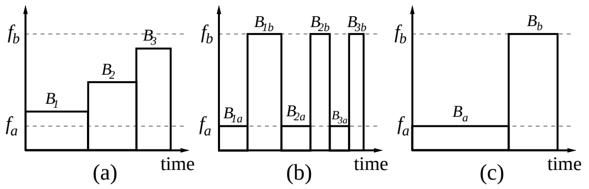

Figure 4: (a) Three bins' frequencies f_1, f_2, f_3 are between two adjacent discrete frequencies f_a and f_b. (b) Each bin breaks into two sub-bins to use f_a and f_b. (c) The three bins are combined into two bins to use f_a and f_b.

4 (a) shows an example when three bins B_1, B_2, and B_3 have frequencies between two adjacent discrete frequencies f_a and f_b. Figure 4 (b) shows that each bin breaks into two sub-bins to use f_a and f_b within the same execution time of the original bin. For example, B_1 breaks into B_{1a} and B_{1b}. The frequency rises and falls between f_a and f_b and the schedule is no longer accelerating. The early bins (e.g., B_{1b}) with higher probabilities may use f_b while the later bins (e.g., B_{3a}) with lower probabilities may use f_a; consequently, the expected total energy is not minimized. Alternatively, we propose to combine the three bins into two bins and use f_a and f_b within the duration of the three original bins, as illustrated in Figure 4 (c). This guarantees only one frequency switching between any two adjacent frequencies and the early bin uses the lower frequency. There can be more than three bins with frequencies between f_a and f_b. Let n be the number of bins with frequencies ($f_k, 1 \leq k \leq n$) between two adjacent discrete frequencies f_a and f_b. The total number of cycles of the n bins is nb and total duration is $\sum_{k=1}^{n} \frac{b}{f_k}$. We need to combine the n bins into two bins. Let x be the number of cycles assigned to the first bin. Then $nb - x$ is the number of cycles assigned to the second bin. The two bins should use the same duration as the original n bins. The value of x thus can be found by solving equation $\sum_{k=1}^{n} \frac{b}{f_k} = \frac{x}{f_a} + \frac{nb-x}{f_b}$.

3.5 Implementation Issues

Our method can be implemented in practical systems. First, the probability distributions of the cycle demands of tasks can be obtained by profiling their offline traces. Second, our method computes the frequency schedules for a task set offline, so there is no computation overhead at runtime. The computed schedules can be stored in the system memory for runtime usage. Third, at runtime, we only need to count how many cycles have been consumed by the current task and determine whether to switch to a higher frequency by looking up the accelerating frequency schedule. Previous study [11] has implemented cycle counting together with an EDF scheduler and shows the feasibility of using accelerating frequency schedule at runtime.

4. EXPERIMENTAL RESULTS

This section describes the simulation setup and presents the simulation results comparing the energy savings from our method and the existing solutions for both finite and continuous frequencies. When the number of finite frequencies increases, we are approaching the ideal case where continuous frequencies are available. Using continuous frequencies allows evaluating the maximum potential of our method.

Table 1: XScale's frequency/voltage and power.

Frequency(MHz)	150	400	600	800	1000
Voltage(V)	0.75	1.0	1.3	1.6	1.8
Power(mW)	80	170	400	900	1600

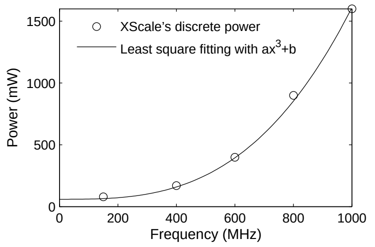

Figure 5: Discrete and continuous power.

4.1 Method of Simulation

We use the frequency/voltage settings and power consumption of Intel XScale [10] (as shown in Table 1) for finite frequencies. For continuous frequencies, we obtain their power numbers by applying least-square curve fitting using $af^3 + b$ ($a = 1.55 \times 10^{-6}$, $b = 60$) to the XScale's frequency and power, as shown in Figure 5.

Both multimedia applications and synthetic workloads are used in our evaluation. A task set of four multimedia programs is used: `mpegplay`, `madplay`, `tmndec`, and `toast`, as shown in Table 2. All these programs except `toast` have the same period, but their WCEC vary considerably. The distributions of their cycle demands are obtained by profiling offline traces with bins each containing 100,000 cycles.

We also perform evaluation over a synthetic task set with 30 tasks and with a wider range of periods, WCEC, and distributions. We construct the task set in three steps. The first step assigns each task a period randomly chosen uniformly between 10 milliseconds and 1 second. The second step chooses for each task its WCEC (W) randomly uniformly between 100,000 cycles and 100,000,000 cycles under the constraint that the worst-case CPU utilization at 1000MHz is no more than one. The third step determines a distribution function of cycle demand for each task. We consider three types of distributions (Gaussian, exponential, and uniform) for cycle demands as suggested in [10, 12]. We use the same type of distributions for all tasks and test the three distribution types separately. Gaussian distribution has two parameters: mean (μ) and standard deviation (σ). For task K_i, μ_i is randomly chosen within the range $(0, W_i]$ and $\sigma_i = W_i/6$. Exponential distribution has one parameter λ ($\mu = \frac{1}{\lambda}$, $\sigma^2 = \frac{1}{\lambda^2}$). We choose for each task its μ_i (or $\frac{1}{\lambda_i}$) also randomly within the range $(0, W_i]$. Uniform distribution is a constant equal to $\frac{1}{W_i}$ for task K_i. Four methods that handle multiple tasks are compared:

- *separated continuous (SC)*: Separating inter-task and intra-task scheduling. SC allocates time to tasks based on their WCEC [11] and then constructs a frequency schedule (Equation (3)) for each task. SC considers continuous frequencies.
- *separated discrete (SD)*: SD considers finite and discrete frequencies [10] and allocates time to multiple tasks based on their WCEC.
- *integrated continuous (IC)*: IC integrates inter-task and intra-task scheduling for continuous frequencies as explained in Section 3.3.

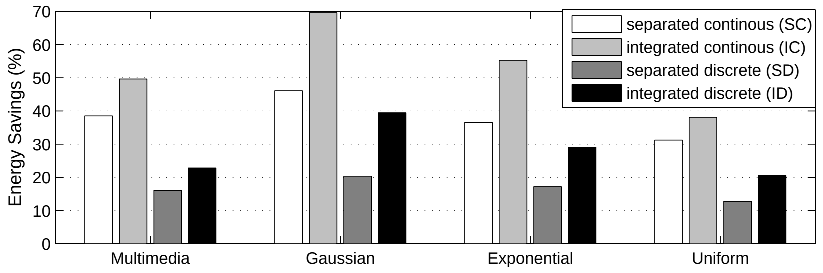

Figure 6: Energy savings from the set of multimedia benchmarks and the three synthetic task sets with Gaussian, exponential, and uniform distributions, respectively.

Table 2: Multimedia applications. * thousand cycles

Application	Description	$T(ms)$	W^*
mpegplay	MPEG video decoder	30	10,500
madplay	MP3 audio decoder	30	899
tmndec	H263 video decoder	30	12,700
toast	GSM speech encoder	25	240

- *integrated discrete (ID)*: ID integrates inter-task and intra-task scheduling for finite and discrete frequencies as explained in Section 3.4.

The first two methods are existing solutions. The third and fourth methods are proposed in this paper. The baseline method used to compare the four methods uses a single frequency $\sum_{i=1}^{N}(W_i/T_i)$ for all tasks.

4.2 Energy Savings

Figure 6 shows the simulation results. The four groups of bars show the energy savings from four task sets: the set of multimedia benchmarks and the three synthetic task sets with Gaussian, exponential, and uniform distributions, respectively. For each task set, the four bars represent the energy savings by the four methods.

Figure 6 shows that *IC* always saves more energy (ranging from 8% to 23%) than *SC*. Method *ID* always saves more energy (ranging from 7% to 19%) than *SD*. The reason is that the integrated scheduling methods consider the periods, the worst-case cycle demands, and the probabilistic distributions of all tasks to perform global adjustment for better energy savings. The figure also shows that, for finite and discrete frequencies, both separated and integrated scheduling methods save less energy than for continuous frequencies. Specifically, *SD* saves 13% to 26% less energy than *SC* and *ID* saves 17% to 30% less energy than *IC*. This is because finite and discrete frequencies restrict the options in the frequency schedules and produce inferior energy savings. The multimedia task set has only four tasks while each of the other task sets has 30 tasks. Intuitively, the overall cycle demands or CPU utilization by the multimedia set may be significantly lower and there is much more room for energy reduction. However, Figure 6 shows that the energy savings obtained by all methods from the multimedia task set are not significantly higher than the savings from the other data sets. The reason is that the four multimedia programs all have very small periods and their cycle demands are often close to their worst cases. This makes their overall CPU utilization actually close to the the other task sets.

5. CONCLUSIONS

This paper presents a DVS scheme for multitasking real-time systems with uncertain execution time. We use the probability distributions of the cycle demands of tasks and integrate inter-task and intra-task frequency scheduling for better energy savings. We consider both continuous frequencies and finite discrete frequencies. Our evaluation shows that our method outperforms the existing solutions for multimedia applications and synthetic workloads. Overall, our approach saves 7% to 23% more energy than the existing solutions. For future work, we plan to perform sensitivity analysis of the bin size (granularity) of tasks.

6. ACKNOWLEDGMENTS

This work is supported in part by National Science Foundation Career CNS-0347466. Any opinions, findings, and conclusions or recommendations are those of the authors and do not necessarily reflect the views of the sponsors.

7. REFERENCES

[1] G.C. Buttazzo. *Hard Real-Time Computing Systems.* Kluwer, 1997.

[2] F. Gruian. Hard Real-Time Scheduling for Low-Energy Using Stochastic Data and DVS Processors. In *the ISLPED*, pages 46–51, August 2001.

[3] Intel XScale Microarchitecture. (http://developer.intel.com/design/xscale).

[4] T. Ishihara and H. Yasuura. Voltage Scheduling Problem for Dynamically Variable Voltage Processors. In *ISLPED*, pages 197–202, 1998.

[5] J. L. W. V. Jensen. Sur Les Fonctions Convexes Et Les Inegalites Entre Les Valeurs Moyennes. *Acta Math*, 30:175–193, 1906.

[6] C. L. Liu and J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the Association for Computing Machinary*, 20(1):46–61, January 1973.

[7] J. R. Lorch and A. J. Smith. Improving Dynamic Voltage Scaling Algorithms with PACE. In *the 2001 ACM SIGMETRICS*, pages 50–61, June 2001.

[8] P. Pillai and K. G. Shin. Real-time Dynamic Voltage Scaling for Low-power Embedded Operating Systems. In *ACM SOSP*, pages 89–102, 2001.

[9] R. Xu, D. Moss, and R. Melhem. Minimize Expected Energy in Real-Time Embedded Systems. In *the 5th ACM EmSoft*, pages 251–254, September 2005.

[10] R. Xu, C. Xi, R. Melhem, and D. Moss. Practical PACE for Embedded Systems. In *the 4th ACM EmSoft*, pages 54–63, September 2004.

[11] W. Yuan and K. Nahrstedt. Energy-Efficient Soft Real-Time CPU Scheduling for Mobile Multimedia Systems. In *ACM SOSP*, pages 149–163, 2003.

[12] Y. Zhang, Z. Lu, J. Lach, K. Skadron, and M. R. Stan. Optimal Procrastinating Voltage Scheduling for Hard Real-Time Systems. In *the 42nd DAC*, pages 905–908, June 2005.