Energy Minimization for Real-Time Systems With (m, k)-Guarantee

Linwei Niu, *Student Member, IEEE*, and Gang Quan, *Member, IEEE*

***Abstract*—Energy consumption and quality of service (QoS) are two primary concerns in the development of today's pervasive computing systems. While most of the current research in energy-aware real-time scheduling has been focused on hard real-time systems, a large number of practical applications and systems exhibit more soft real-time nature. In this paper, we study the problem of minimizing energy for soft real-time systems while providing a QoS guarantee. The QoS requirements are deterministically quantified with the (m, k)-constraints, which require that at least m out of any k consecutive jobs of a task meet their deadlines. In this paper, we propose a hybrid approach to achieve the dual goals of QoS guarantee and energy minimization. We first present the necessary and sufficient schedulability conditions for the static mandatory/optional workload partitioning. Then, we propose to dynamically vary the statically defined mandatory/optional partitions to accommodate dynamic run-time variations while minimizing the energy consumption. The experimental results demonstrate that our proposed techniques outperform previous work significantly in terms of both the energy savings and achieved QoS.**

***Index Terms*—Dynamic volage scaling (DVS), earliest deadline first (EDF), embedded system, quality of service (QoS), real-time scheduling.**

I. Introduction

POWER-AWARE computing has come to be recognized as a critical enabling technology in the design of pervasive real-time embedded systems. A large number of techniques (e.g., [3], [13], and [35]) have been proposed to reduce the energy consumption of real-time computing systems. Most of these techniques have targeted hard real-time systems, i.e., systems that require that all task instances meet their deadlines. However, many practical real-time applications exhibit more complicated characteristics that can only be captured with more complex requirements, generally called the quality of service (QoS) requirements. For example, some applications may have soft deadlines where tasks that do not meet their deadlines can still be completed with a reduced value [19] or they can simply be dropped without compromising the desired QoS levels. A key to the success is the ability to integrate the QoS requirements into resource management/scheduling decisions in such a way that the overall "benefit" of the system is optimized. However, techniques based on the traditional hard real-time systems become inefficient or inadequate when QoS requirements are imposed.

Manuscript received July 5, 2005; revised January 9, 2006. This work was supported in part by the National Science Foundation under the CAREER Award CNS-0545913 and by the University of South Carolina Research Program under the Research Productive Scholarship Award.

The authors are with the Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208 USA (e-mail: niul@cse.sc.edu; gquan@cse.sc.edu).

Digital Object Identifier 10.1109/TVLSI.2006.878337

Recently, there has been increasing interest in incorporating dynamic voltage scaling (DVS) and real-time scheduling techniques as a means of satisfying power/energy conservation requirements with regard to QoS constraints. These approaches can be classified into two categories: the best-effort and the guaranteed approaches. The best-effort approaches (e.g., [20], [21], [30], and [33]) intend to enhance the QoS of the system and minimize the power/energy consumption in the context of power-aware scheduling, but with no assurance to either of them. The guaranteed approaches, on the other hand, optimize the energy usage with a QoS guarantee in mind. A predominant portion of the current guarantee research in power-aware scheduling has to do with the statistical service guarantee, e.g., using the average deadline miss rate as a QoS metric. The statistical guarantee ensures the quality of service in a probabilistic manner. This can be problematic for some real-time applications. For example, many real-time applications can tolerate occasional deadline misses of the real-time tasks, and the information carried by these tasks can be estimated to a reasonable accuracy using techniques such as interpolation. However, even a very low overall miss rate tolerance cannot prevent a large number of deadline misses from occurring in such a short period of time that the data cannot be successfully reconstructed. To avoid possible severe consequences, one can always treat the system as a hard real-time system. The problem, however, is that the energy savings can be seriously degraded, and the mission cycles can be severely reduced.

To provide a deterministic QoS to real-time systems, the system should not only support the overall guarantee of the QoS statistically, but also provide a lower bounded, predictable level of QoS locally. Hamdaoui *et al.* [8] proposed the (m, k)-model that can serve well for this purpose. According to this model, a repetitive task of the system is associated with an $(m, k)(0 < m \leq k)$ constraint requiring that m out of any k consecutive job instances of the task meet their deadlines. A dynamic failure occurs, which implies that the temporal QoS constraint is violated and the scheduler is thus considered failed, if, within any k consecutive jobs, more than $(k - m)$ job instances miss their deadlines. Based on this (m, k)-model, Ramanathan *et al.* [29] proposed to partition the jobs into mandatory and optional jobs. So long as all of the mandatory jobs can meet their deadlines, the (m, k)-constraints can be ensured.

Since all jobs to be scheduled are not required to meet their deadlines, energy can be saved by running as many *mandatory* jobs as possible at low voltage levels. The problem is how to judiciously select the set of mandatory jobs and their running speeds. The mandatory job set as well as the job running speeds

can be selected either statically or dynamically. The advantage of statically selecting the mandatory jobs and their speeds lies in the fact that the schedulability analysis can be performed offline and thus it is easier to guarantee the QoS constraints. However, due to the dynamic nature of the real-time systems under investigation, the energy-saving performance that the static approach can achieve is rather limited. On the other hand, the dynamic approach can generally utilize the system resources more efficiently by incorporating the run-time information. The problem, however, is how to ensure the schedulability of the mandatory jobs and hence the QoS guarantee. This is exacerbated when considering that both the mandatory/optional partitioning problem as well as the schedulability analysis problem are NP-hard in the strong sense [27].

In this paper, we propose a static/dynamic hybrid approach to minimize the energy consumption for real-time systems while providing the (m, k)-guarantee. In the static part, the mandatory job sets for the given systems are statically determined. The necessary and sufficient conditions to ensure the schedulability for the mandatory jobs are developed, which are then used to determine the processor speed associated with each task. In the dynamic part, the mandatory job sets are dynamically varied to accommodate the dynamic nature of real-time embedded systems with (m, k)-constraints. Specifically, a scheduler, based on the dual-priority scheduler [7], is designed to dynamically determine whether a job should be mandatory/optional as well as its corresponding processor speed to maximize the energy-saving performance. Through our extensive experiments, the results show that our proposed approaches can significantly improve the energy savings over previous ones while guaranteeing the (m, k)-constraints.

The remainder of this paper is organized as follows. Section II discusses the related work. Section III introduces the system model and problem formulation. Section IV presents the static mandatory/optional partitioning strategy to guarantee the (m, k)-constraints. Section V presents a technique to dynamically adjust the mandatory/optional partition. Section VI introduces our integrated static/dynamic DVS scheduling techniques. The effectiveness and energy efficiency of our approach are demonstrated using simulation results in Section VII. In Section VIII, we offer conclusions for this paper.

II. Related Work

Significant research efforts have been made for statistical QoS guarantee in conjunction with power savings. For example, Qiu *et al.* presented a technique [26] to statistically guarantee the QoS for a real-time embedded system. Yuan *et al.* [36] proposed incorporating the stochastic analysis into DVS for soft deadline systems to provide a statistical QoS guarantee. With QoS requirements formulated as a tolerable statistical deadline miss rate, Hua *et al.* [10] introduced several techniques to reduce energy by exploiting processor slack time due to the missed deadlines. However, in all of these works, dynamic failure cannot be avoided.

For its intuitiveness and capability of capturing not only statistical but also deterministic QoS requirements, the (m, k)-model has been widely studied, e.g., [4], [8], [9], [27], and [29]. The (m, k)-model was originally proposed by Hamdaoui *et al.* [8]. Koren *et al.* [14] proposed a skip-over model which is a special case of the (m, k)-model where $m = k - 1$. To guarantee the (m, k)-constraints, Ramanathan *et al.* [29] proposed a strategy to partition the jobs into mandatory and optional jobs. The mandatory jobs are the jobs that must meet their deadlines in order to satisfy the (m, k)-constraints, while the optional jobs can be executed to further improve the quality of the service or simply be dropped to save computing resources. Quan *et al.* [27] formally proved that the problem of scheduling with the (m, k)-guarantee for an arbitrary value of m and k is NP-hard in the strong sense. They further proposed to improve the mandatory/optional partitioning by reducing the maximal interference between mandatory jobs.

Bernat *et al.* [5] treated the (m, k)-model slightly differently. They assumed that the jobs that miss their deadlines still need to be executed. Under this assumption, they provided a number of theoretical analysis results regarding the properties and relationships of different (m, k)-constraints and developed an offline technique to test if the (m, k)-constraints can be satisfied. To improve the schedulability, Bernat *et al.* [6] proposed a Bi-Modal scheduler for systems with (m, k)-constraints. The tasks are first scheduled according to the generic scheduling policy in the normal mode and then switched to the panic mode if the dynamic failure is likely to occur. The schedulability is guaranteed using the worst case response time analysis similar to that in [5].

West *et al.* [34] introduced another related model, called the window-constrained model. The major difference between the (m, k)-model and the window-constrained model is that the (m, k)-constraint requires that at least m jobs meet their deadlines for any k consecutive jobs, while the window constraint requires that within any nonoverlapped and consecutive windows containing k jobs, at least m of them meet their deadlines. It can be readily concluded that window constraints are weaker than the (m, k)-constraints as a feasible schedule under the (m, k)-constraints must also be feasible under the window constraints. In [2], Mok *et al.* proved that even with the *dynamic window constraints*, the guaranteed scheduling problem is still NP-hard. They also proposed a P-fairness scheduling algorithm for real-time systems with dynamic window constraints. However, the deterministic assurance with this model can only be guaranteed for a very limited range of systems, such as those systems in which all tasks have the same unit size execution times. None of the approaches above has taken both the deterministic QoS guarantee and energy/power consumption simultaneously into considerations. While Hua and Qu [9] introduced a greedy approach to minimize the energy consumption for systems with (m, k)-constraints running on a processor with two different speeds, as we show later in this paper, their approach cannot guarantee the (m, k)-constraints, even though the system is underloaded.

III. Preliminary

Here, we provide the problem formulation followed by the motivations for our approach.

A. System Models and Problem Formulation

The real-time system considered in this paper contains n independent periodic tasks $\mathcal{T} = \{\tau_0, \tau_1, \ldots, \tau_{n-1}\}$, sched-

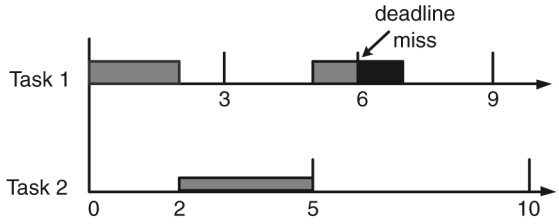


Fig. 1. The greedy approach in [9] fails to guarantee the schedulability of an underloaded task set ($\tau_1 = (3, 3, 2, 1, 1)$; $\tau_2 = (5, 5, 1.5, 1, 2)$).

uled according to the earliest deadline first (EDF) policy [18]. Each task contains an infinite sequence of periodically arriving instances called *jobs*. Task τ_i is characterized using five parameters, i.e., $(T_i,\ D_i,\ C_i,\ m_i,\ k_i)$. T_i represents the period, D_i $(D_i \leq T_i)$ is the deadline, and C_i is the worst case execution time for τ_i, respectively. A pair of integers, i.e., (m_i, k_i) $(0 < m_i \leq k_i)$, represent the QoS requirement for τ_i, requiring that, among any k_i consecutive jobs of τ_i, at least m_i jobs meet their deadlines.

The DVS processor used in our system can operate at a finite set of discrete supply voltage levels $\mathcal{V} = \{V_1, \ldots, V_{\max}\}$, each with an associated speed. To simplify the discussion, we normalize the processor speeds to $S_{\max}$, the speed corresponding to $V_{\max}$, which results in $\mathcal{S} = \{S_1, \ldots, 1\}$. We assume that C_i is the worst case execution time for task τ_i in the highest voltage mode. Therefore, if τ_i is executed under speed S_j, the worst case execution time for τ_i becomes C_i/S_j.

With the above system models, our problem can be formulated as follows.

1) Problem 1: Given system $\mathcal{T} = \{\tau_0, \tau_1, \ldots, \tau_{n-1}\}$, $\tau_i = (T_i, D_i, C_i, m_i, k_i)$, $i = 0, \ldots, (n-1)$, schedule $\mathcal{T}$ with EDF on a variable voltage processor with supply voltage levels $\mathcal{V} = \{V_1, \ldots, V_{\max}\}$ and corresponding processor speeds $\mathcal{S} = \{S_1, \ldots, 1\}$ such that all (m, k)-constraints are guaranteed and the energy consumption is minimized.

B. Motivations

As mentioned before, Hua *et al.* [9] adopted a greedy approach to minimize the energy consumption for real-time systems running on a dual-voltage mode processor. When a new job arrives, it is run at the low voltage level if the corresponding task can tolerate at least one more deadline miss. Otherwise, this job is to be executed at the high voltage level. Energy is saved since, if the jobs can meet their deadlines with low processor speeds, they reduce the necessity to run jobs at high processor speeds which consumes more energy. However, this approach cannot always guarantee the (m, k)-constraints even if all tasks of the task set could meet their deadlines with the highest processor speeds, i.e., the system is underloaded.

Consider a task set of two tasks, i.e., $\tau_1 = (3, 3, 2, 1, 1)$ and $\tau_2 = (5, 5, 1.5, 1, 2)$. The total utilization of this task set is 14.5/15 and it is schedulable under EDF. For the dual-voltage mode processor, without loss of generality, we assume that the high voltage corresponds to a processor speed of 1 and the low voltage corresponds to a processor speed of 0.5. Also, we assume that all tasks start at time 0. Fig. 1 shows the task schedule according to this greedy approach. (The rectangles represent the execution of jobs and their heights represent the speed of the jobs.)

At time $t = 0$, since task τ_1 has an (m, k)-constraint of (1,1) requiring each job of τ_1 to meet its deadline, τ_{11} is executed with high processor speed. At time $t = 2$, following the completion of τ_{11}, the low processor speed will be assigned to execute τ_{21} according to the greedy approach since τ_2 has an (m, k)-constraint of (1,2) not requiring job τ_{21} to meet its deadline. According to EDF, only after τ_{21} ends at $t = 5$ can τ_{12} be executed. However, even if it is executed with the highest speed, it will still miss the deadline at time $t = 6$ and thus cause a dynamic failure.

We feel that the uniform treatment of each job during the scheduling process in the greedy approach [9] is one of the major contributions to the dynamic failure. As such, we developed our techniques based on the predetermined mandatory/optional partitioning results. It is not difficult to see that different partition strategies can have tremendous impacts on the schedulability of the system and thus the energy consumption. In Sections IV–VIII, we first examine different partition strategies and then develop novel scheduling techniques to meet the dual goals of deterministic QoS guarantee and energy minimization.

IV. Mandatory/Optional Job Partitioning With (m, k)-Pattern

Here, we study the properties of two special statically defined mandatory/optional partitions. These properties form the basis for our energy saving approaches in Section VI. To ease the static analysis as well as to reduce the implementation cost, we adopt the concept of the (m, k)-pattern as introduced in [27].

Lemma 1: The **$(\mathbf{m}, \mathbf{k})$-pattern** of task τ_i, denoted by Π_i, is a binary string $\Pi_i = \{\pi_{i0}\pi_{i1} \ldots \pi_{i(k_i-1)}\}$ which satisfies the following: 1) τ_{ij} is a mandatory job if $\pi_{ij} = 1$ and optional if $\pi_{ij} = 0$ and 2) $\sum_{j=0}^{k_i-1} \pi_{ij} = m_i$.

By repeating the (m, k)-pattern Π_i, we get a mandatory job pattern for τ_i. It is not difficult to see that the (m, k)-constraint for τ_i can be satisfied if the mandatory jobs of τ_i are selected accordingly.

A. Deeply Red Pattern

The first partition strategy was proposed by Koren *et al.* [14]. According to this scheme, a job τ_{ij}, i.e., the jth job of task τ_i, is determined to be mandatory if

$$\pi_{ij} = \begin{cases} 1, & 0 \leq j \mod k_i < m_i \\ 0, & \text{otherwise} \end{cases} \qquad j = 0, 1, \ldots, k_i - 1. \tag{1}$$

We borrow the initial terminology of this strategy and refer to this mandatory job pattern as the deeply red pattern or R-pattern. The R-pattern has an interesting property that is formulated in the following theorem.

Theorem 1: Let $\mathcal{J}_r$ be the mandatory job set selected from all of the jobs in $\mathcal{T}$ according to R-pattern. If $\mathcal{J}_r$ is schedulable, then the mandatory job set selected from any other (m, k)—pattern is also schedulable.

Proof: For real-time tasks scheduled with EDF, Zheng *et al.* [37] and Liebeherr *et al.* [17] proposed a necessary and sufficient schedulability condition as follows:

$$\mathcal{T} \text{ is schedulable} \Longleftrightarrow \forall\, t > 0, \qquad \sum_i W_i(0,t) \le t \tag{2}$$

where $W_i(0,t)$ is the total workload from jobs of τ_i that arrive before t and must be finished by t.

For real-time system $\mathcal{T}$, let the mandatory workload within $[0,t)$ be $W(0,t)$ according to the R-pattern, and let be $\tilde{W}(0,t)$ according to any other arbitrary (m,k)-pattern. Then, from (2), we have

$$\forall\, t > 0, \qquad W(0,t) \le t. \tag{3}$$

In addition, based on Definition 1 and (1), we can conclude that

$$\forall\, t > 0, \quad \tilde{W}(0,t) \le W(0,t). \tag{4}$$

Therefore, from (3) and (4), we have

$$\forall\, t > 0, \quad \tilde{W}(0,t) \le t. \tag{5}$$

From (2), the conclusion holds. ∎

This property implies that, as long as the mandatory jobs are schedulable according to the R-pattern, any dynamic scheduling approach can guarantee the schedulability of the mandatory jobs as long as there are no more than m_i mandatory jobs among any k_i consecutive job instances from task τ_i. However, always assuming the first m_i jobs of each τ_i are mandatory inevitably increases the interference among them and makes the schedulability test overly pessimistic.

B. Evenly Distributed Pattern

The second partition strategy was proposed by Ramanathan *et al.* [29] as follows:

$$\pi_{ij} = \begin{cases} 1, & \text{if } j = \left\lfloor \left\lceil \frac{j \times m_i}{k_i} \right\rceil \times \frac{k_i}{m_i} \right\rfloor \\ 0, & \text{otherwise} \end{cases} \qquad j = 0, 1, \ldots, k_i - 1 \tag{6}$$

The (m,k)-pattern defined with formula (6) exhibits some interesting characteristics which are summarized in the following lemma.

Lemma 1: Let the mandatory jobs for task τ_i with (m,k) constraint (m_i,k_i) be determined by (6). Then, we have the following.

1) For any $t > 0$, the interval $[0,t]$ has the largest number of mandatory jobs compared with any other interval with the same length $|t|$.
2) For any k_i consecutive jobs of τ_i, there are exactly m_i mandatory jobs.
3) For any two subsequences of task τ_i that contain the same number of consecutive jobs, the difference of the numbers of mandatory jobs is no more than 1.

Proof: The first conclusion has been proved in [29] and the second conclusion can be readily derived from [29, Lemma 2]. We next prove the third conclusion as follows.

Assume two subsequences of task τ_i that both contain $p_i (p_i > 0)$ consecutive jobs. Let $N_i(x_i, y_i)$ be the number of mandatory jobs starting from job x_i to job y_i. For p_i jobs starting from job a_i and b_i $(a_i \neq b_i, a_i, b_i \ge 0)$, the numbers of mandatory jobs are denoted as $N_i(a_i, a_i + p_i - 1)$ and $N_i(b_i, b_i + p_i - 1)$, respectively. When the mandatory jobs are determined according to (6), for the first q_i jobs (from job 0 to job $q_i - 1$) of τ_i, there are $l_i(q_i) = \lceil (m_i/k_i) q_i \rceil$ jobs that are mandatory [29]. Therefore

$$\begin{aligned} N_i(a_i, a_i + p_i - 1) &= l_i(a_i + p_i) - l_i(a_i + 1) \\ &= \left\lceil \frac{m_i}{k_i}(a_i + p_i) \right\rceil - \left\lceil \frac{m_i}{k_i}(a_i + 1) \right\rceil \end{aligned}$$

and similarly

$$\begin{aligned} N_i(b_i, b_i + p_i - 1) &= l_i(b_i + p_i) - l_i(b_i + 1) \\ &= \left\lceil \frac{m_i}{k_i}(b_i + p_i) \right\rceil - \left\lceil \frac{m_i}{k_i}(b_i + 1) \right\rceil. \end{aligned}$$

Assume $N_i(a_i, a_i + p_i - 1) \ge N_i(b_i, b_i + p_i - 1)$, and, since $\lceil x_1 + x_2 \rceil \le \lceil x_1 \rceil + \lceil x_2 \rceil$ and $\lceil x_1 + x_2 \rceil \ge \lceil x_1 \rceil + \lfloor x_2 \rfloor$ for any $x_1, x_2 \in R$, it follows that

$$\begin{aligned} &|N_i(a_i, a_i + p_i - 1) - N_i(b_i, b_i + p_i - 1)| \\ &\quad \le \left\lceil (p_i - 1)\frac{m_i}{k_i} \right\rceil - \left\lfloor (p_i - 1)\frac{m_i}{k_i} \right\rfloor \\ &\quad \le 1. \end{aligned}$$

If $N_i(a_i, a_i + p_i - 1) < N_i(b_i, b_i + p_i - 1)$, we can similarly prove that

$$|N_i(a_i, a_i + p_i - 1) - N_i(b_i, b_i + p_i - 1)| \le 1.$$

∎

Lemma 1 implies that, with (6), a minimal set of mandatory jobs are determined. Moreover, according to Lemma 1, (6) helps to spread out the mandatory jobs of each task *evenly*. We therefore call this mandatory job pattern the evenly distributed pattern or simply the E-pattern. Fig. 2 shows several examples of mandatory/optional jobs determined according to their R-patterns and E-patterns (the E^R-patterns will be introduced later.)

Recall that any mandatory job missing its deadline will cause violation of the (m,k)-constraint. Therefore, to ensure the (m,k)-constraints, an accurate schedulability analysis is crucial since inaccurate prediction may either cause the violation of the system requirements, or lead to poor energy-saving performance. The closed-form necessary and sufficient condition presented in (2) can be used to predict the EDF schedulability for task system with arbitrary job arrivals and deadlines. Unfortunately, it is not practical in checking the feasibility, since

(m,k) constraints	Deeply-red Pattern	Evenly distributed Pattern	Reverse Evenly distributed Pattern
(1, 2)	1 0 1 0 1 0 1 0 ...	1 0 1 0 1 0 1 0 ...	0 1 0 1 0 1 0 1 ...
(2, 5)	1 1 0 0 0 1 1 0 0 0	1 0 1 0 0 1 0 1 0 0...	0 0 1 0 1 0 0 1 0 1 ...
(3, 6)	1 1 1 0 0 0 1 1 ...	1 0 1 0 1 0 1 0 ...	0 1 0 1 0 1 0 1 ...
(3, 7)	1 1 1 0 0 0 0 1 1 ...	1 0 1 0 1 0 0 1 0 ...	0 0 1 0 1 0 1 0 0 1 ...

Fig. 2. Examples of mandatory jobs based on R-patterns, E-patterns, and E^R-patterns.

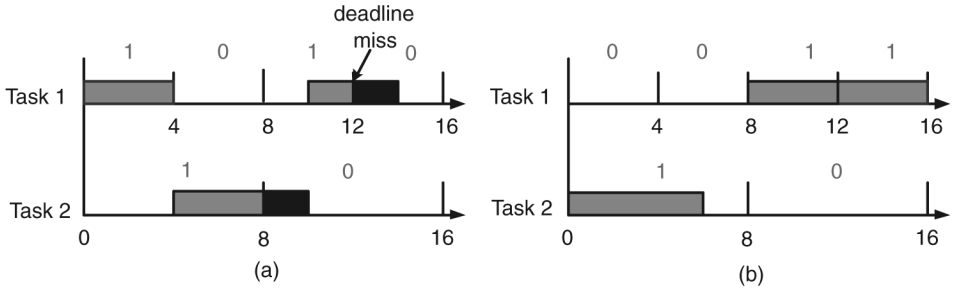


Fig. 3. (a) The task set ($\tau_1 = (4,4,4,2,4)$; $\tau_2 = (8,8,6,1,2)$) is NOT schedulable with the E-Pattern. (b) The same task set is schedulable with other (m,k)-patterns.

it requires checking an infinite number of time points ($t > 0$). More efficient methods were also introduced (e.g., Ripoll *et al.* [31]) such that the schedulability test can be done in a limited and short interval. However, these methods targeted synchronized periodic systems, i.e., all periodic tasks have the same initial arrival time and thus cannot readily be used to test the schedulability in our case.[1] In the following theorem (Theorem 2), we develop a necessary and sufficient condition based on the work demand analysis strategy. Before we introduce this theorem, we first present the following definition.

Definition 2: Let $w(t)$ represent the workload from the jobs that arrives in $[0,t]$ but not finished before t. A busy interval, i.e., $[t_s, t_e]$, is the interval such that $w(t_s^-) = 0$, $w(t_e^+) = 0$, and $w(t) > 0$ for $t_s \le t < t_e$.

Also, to ease our presentation, we adopt the following notation, i.e., $\lceil x \rceil^+$:

$$\lceil x \rceil^+ = 1 + \lfloor x \rfloor. \tag{7}$$

The following theorem allows one to predict the schedulability for a mandatory job set by checking only a limited number of points (see the Appendix for the detailed proof.)

Theorem 2: Let system $\mathcal{T} = \{\tau_0, \tau_1, \ldots, \tau_{n-1}\}$, where $\tau_i = \{T_i, D_i, C_i, m_i, k_i\}$, and $\mathcal{R}$ be the mandatory job set according to their E-patterns. Also, let L represent either the ending point of the first busy period when scheduling only the mandatory jobs or the LCM of T_i, $i = 0, \ldots, (n-1)$, whichever is smaller. Then, $\mathcal{R}$ is schedulable with EDF if and only if (iff) all the mandatory jobs arriving within $[0, L]$ can meet their deadlines, i.e.,

$$\sum_i W_i(0,t) = \sum_i \left(\left\lceil \frac{m_i}{k_i} \left\lceil \frac{t - D_i}{T_i} \right\rceil^+ \right\rceil \right) C_i \le t \tag{8}$$

[1]Even though the mandatory jobs defined with the E-pattern can be viewed as periodic with periods equal to $k_i T_i$, the result "periodic" task set is not synchronized.

for all $t \le L$ and $t = \lfloor p(k_i/m_i) \rfloor T_i + D_i$, $p \in Z$, $i = 0, \ldots, n-1$.

From Lemma 1, E-patterns help to distribute the mandatory workload evenly and thus have much better schedulability than R-patterns. In fact, we have the following important theorem (see the Appendix for the detailed proof.)

Theorem 3: Let $\mathcal{T} = \{\tau_0, \tau_1, \ldots, \tau_{n-1}\}$, where $\tau_i = \{T_i, D_i, C_i, m_i, k_i\}$, and $k_i T_i$, $i = 0, 1, \ldots, n-1$ are co-prime. Let $\mathcal{R}$ and $\mathcal{R}'$ be the mandatory job set constructed from $\mathcal{T}$ according to the E-patterns and any other (m,k)-patterns, respectively. Then, if $\mathcal{R}'$ is schedulable, $\mathcal{R}$ is schedulable.

Given the fact that E-patterns have better schedulability than R-patterns, one intuitive approach would be to statically assign and run all of the mandatory jobs according to the E-pattern. Two problems may exist in this approach. First, even though the mandatory jobs for each task are evenly distributed, the overall mandatory workload is not necessarily evenly distributed. For example, as shown in Fig. 3(a), the mandatory job set according to the E-pattern fails to be schedulable, while other mandatory job assignments as shown in Fig. 3(b) can be well schedulable. As implied in Theorem 3, the schedulability of the mandatory job set may degrade severely if a large greatest common divisor (GCD) exists for $k_i T_i$, $i = 0, 1, \ldots, (n-1)$. An extreme case would be

$$k_0 T_0 = k_1 T_1 = \cdots = k_{n-1} T_{n-1}$$

and $m_0 = m_1 = \cdots = m_{n-1}$. Finding a better (m,k)-pattern in such scenarios is beyond the scope of this paper and will be a subject of future study. Second, executing optional jobs can be beneficial in saving energy since, if an optional can meet its deadline, we do not have to run some mandatory jobs at a higher processor speed. In regard to this, we propose another (m,k)-pattern to statically partition the mandatory and optional jobs.

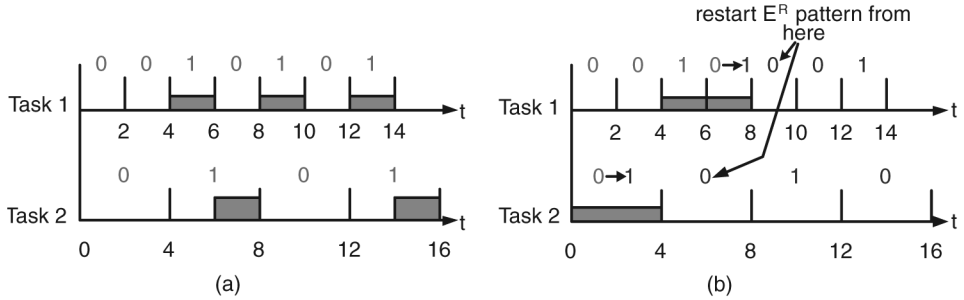


Fig. 4. (a) Executing only the mandatory jobs of task set ($\tau_1 = (2, 2, 1, 3, 7)$; $\tau_2 = (4, 4, 2, 1, 2)$) according to their E^R-patterns. (b) Dynamically restarting the E^R-pattern at $t = 8$ for τ_1 and $t = 4$ for τ_2.

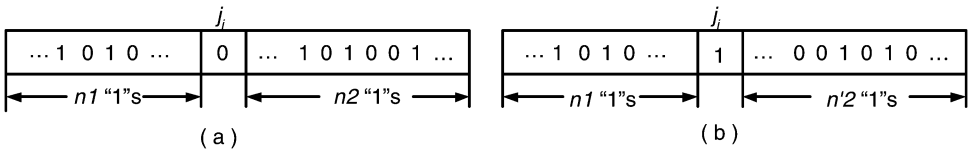


Fig. 5. (a) The original mandatory jobs according to the E^R-pattern of task τ_i. (b) The mandatory jobs of τ_i after restarting the E^R-pattern.

C. Reverse-Evenly-Distributed Pattern

According to E-patterns, the first job is always mandatory. As the successful completion of optional jobs at lower processor speeds may potentially alleviate the necessity to run mandatory jobs at higher speeds, we slightly modified the E-pattern as follows:

$$\pi_{ij} = \begin{cases} 0, & \text{if } j = \left\lfloor \left\lceil \frac{j\times(k_i-m_i)}{k_i} \right\rceil \times \frac{k_i}{(k_i-m_i)} \right\rfloor \\ 1, & \text{otherwise} \end{cases}$$
$$j = 0, 1, \ldots, k_i - 1. \tag{9}$$

The (m, k)-pattern determined by (9) reverses the E-pattern horizontally. We therefore call this (m, k)-pattern the reverse-evenly-distributed pattern or simply the E^R-pattern. Examples of E^R-patterns are shown in Fig. 2. As shown in Fig. 2, the E^R-pattern maintains some good characteristics of the E-pattern as stated in Lemma 1 [conclusions 2) and 3)]. It also has additonal interesting properties that are summarized in the following lemma.

Lemma 2: Let $\mathcal{T} = \{\tau_0, \tau_1, \ldots, \tau_{n-1}\}$, where $\tau_i = \{T_i, D_i, C_i, m_i, k_i\}$, and $i = 0, 1, \ldots, n-1$. Then the following is true.

- The number of optional jobs according to the E^R-patterns in the interval $[0, t]$ is no less than that in any other interval with the same length $|t|$.
- If $\mathcal{T}$ is schedulable under the E-pattern, it is also schedulable under the E^R-pattern.

The first property follows directly from Lemma 1 [see 1)] that the number of mandatory jobs within $[0, t]$ determined by the E-pattern is no less than that in any other interval with the same length. The proof of the second property can be done similarly to that of Theorem 2 and is thus omitted. Lemma 2 implies that the E^R-pattern maintains the optimality of the E-pattern in the sense that it is also the optimal (m, k)-pattern when k_iT_i, $i = 0, 1, \ldots, n-1$, are co-prime.

Nevertheless, the E^R-pattern is still a static mandatory/optional partition strategy. When an optional job can meet its deadline running at a very low processor speed, it makes running other mandatory jobs at higher processor speeds redundant and energy-inefficient. It is thus desirable that the statically defined E^R-pattern be dynamically variable. However, the problem is how to update the pattern dynamically while still guaranteeing schedulability and the (m, k)-constraints. We address the problem in Section V.

V. Dynamic E^R-Pattern Adjustment

When an optional job meets its deadline, it would be beneficial to demote some mandatory jobs to optional so that they can be executed with low processor speed or even be dropped to save energy. We developed a simple yet very effective strategy for this purpose. The algorithm works as follows. Whenever an optional job meets its deadline, we will restart its corresponding E^R-pattern. This strategy can be illustrated by Fig. 4. Fig. 4(a) shows two periodic tasks, i.e., τ_1 and τ_2, and their original E^R-patterns. Assuming that the first job of τ_2 (optional) meets its deadline, we restart the E^R-pattern for τ_2 from the second job. Similarly, assuming that the fourth job of τ_1 meets its deadline, we restart the E^R-pattern for τ_1 from the fifth job, as illustrated in Fig. 4(b). Note that, by restarting E^R-patterns, the need to execute the mandatory jobs at higher speeds is delayed. In addition, more optional jobs are "inserted" before the mandatory jobs which may further delay the execution of mandatory jobs. Significant energy can be saved since the number of jobs that need to be run at a high processor speed is greatly reduced. To guarantee that, after restarting the E^R-pattern as stated above, the (m, k)-constraints can still be satisfied, we have the following lemma.

Lemma 3: Let L be an infinite binary string by repeating the E^R-pattern, and let the ith character, i.e., $j_i = 0$. Then, there are at least m 1's in any k consecutive characters if j_i is changed from 0 to 1 and the E^R-pattern is restarted from the $(i+1)$th character.

Proof: Without loss of generality, assume that the original E^R-pattern is shown in Fig. 5(a), and the (m, k)-pattern after the change is shown in Fig. 5(b). It is not difficult to see that all

of the windows that do not contain j_i can meet their (m, k)-requirements. We then only need to consider the windows that contain j_i.

Consider an arbitrary window that contains j_i. Let n_1 and n_2 denote the number of mandatory jobs before and after j_i in Fig. 5(a), respectively. Then, we have

$$n_1 + n_2 \geq m. \tag{10}$$

After changing j_i from 0 to 1, n_1 remains the same but n_2 may be different. We use n_2' to denote the new value. From Lemma 1 [see 3)], we know that

$$0 \leq (n_2 - n_2') \leq 1. \tag{11}$$

Therefore, by adding n_1 and n_2' and counting j_i (which changes from 0 to 1), there are at least the same number of mandatory jobs as in the original window. ■

VI. DVS Scheduling for the Task Set With (m, k)-Constraints

After presenting the schedulability analysis results for the static (m, k)-pattern and the strategy to dynamically adjust the pattern, we are now ready to introduce our overall approach for Problem 1. Our approach consists of two phases: an offline phase followed by an online phase. The goal for the offline phase is to statically determine the processor speeds for the mandatory jobs such that they are feasible under the static (m, k)-pattern. We use an online phase to exploit the run-time variations to further save energy.

A. Offline Phase

In our offline approach, we associate one processor speed with each task. If processor speed S_j is assigned to task τ_i, then the worst case execution time for the mandatory jobs from τ_i becomes C_i/S_j. By replacing C_i with C_i/S_j in (8), we can therefore test if all of the mandatory jobs are schedulable under the processor speed assignment. The problem then becomes how to choose the best speed assignment that can optimize the energy savings. As our static analysis can be made offline, we can afford using some classic exhaustive search algorithms such as branch-and-bound to find the optimal solution. When using branch-and-bound, some techniques such as ordering the task set according to the values of $m_i C_i/k_i T_i$, $i = 0, \ldots, n-1$, can greatly improve the pruning efficiency. After the voltage level for each task is determined, the worst case response time for each task under its static E^R-pattern is computed using a method similar to that in [23] for the online use. Next, we further improve the energy performance during the online phase.

B. Online Phase

During the online phase, we adopt the dual-priority scheduling [7] scheme and use it to help adjust the mandatory/optional job partitioning adaptively to achieve better energy saving performance while guaranteeing the (m, k)-requirements.

In our algorithm, three job ready queues are maintained: the high mandatory queue (HMQ), the optional queue (OPQ), and the low mandatory queue (LMQ). Upon arrival, a job is determined as a mandatory job or an optional job based on the static (m, k)-pattern. The optional jobs are directly put in the OPQ. The mandatory jobs will be placed in the LMQ first and later promoted to HMQ after a fixed time offset (Y_i), called the promotion time, which is computed by

$$Y_i = D_i - R_i \tag{12}$$

where D_i is the relative deadline of τ_i and R_i is the worst case response time of τ_i under the corresponding static pattern, which is computed during the offline phase as stated above.

The jobs in the HMQ have the highest priority level among all jobs in the three ready queues and will be executed following the EDF scheme with the corresponding speeds determined during the offline phase. The jobs in the LMQ, on the other hand, always run at the lowest possible speed. Note that, while the jobs in the OPQ have a higher priority level than those in the LMQ, an optional job is executed, nonpreemptive by any other optional job, only when it could be finished by the earliest promotion time of the next mandatory job. This helps to avoid the execution of optional jobs that may miss their deadlines later, which has no benefit to either energy saving or improvement of QoS. The jobs in the LMQ are executed according to EDF only when the HMQ is empty and no optional jobs are qualified to be executed.

It is not difficult to see that there may be more than one optional job available in the OPQ, and selecting which one to run may have profound impacts on the future job executions. Searching for the optimal optional job to be executed is an interesting problem and will be studied further. In this paper, we adopt a very simple heuristic to achieve better energy saving performance. Specifically, when the HMQ is empty, we first compute the speed $\hat{S}_i$ that is required to finish each optional job in the OPQ by the promotion time of the next mandatory job. Then, those optional jobs requiring speed less than their predetermined speed S_i will be chosen as candidate jobs. After that, the potential energy gain ΔE_i of each candidate job J_i is computed, which is defined as

$$\Delta E_i = E(S_i) - E(\hat{S}_i)$$

where $E(S_i)$ is the energy consumption of J_i under its predetermined speed and $E(\hat{S}_i)$ is the energy consumption of J_i under its required speed. The optional job that has the largest energy gain will be chosen for execution. When an optional job meets its deadline, the (m, k)-pattern is adjusted according to the strategy in Section V. The algorithm is summarized in Algorithm 1.

Algorithm 1 The online DVS scheduling algorithm.

1: **Upon job arrival:**

2: Put the job into HMQ, OPQ, and LMQ based on its E^R-pattern;

3:

4: **Upon job execution:**

5: **if** HMQ is not empty **then**

6: Run jobs in HMQ according to EDF;

7: **else if** OPQ is not empty **then**

8: $\mathcal{J}_o =$ jobs in OPQ;

9: Select and run $J_i \in \mathcal{J}_o$ that have the maximum energy-saving gain ΔE_i and $\Delta E_i > 0$;

10: **if** $J_o = \emptyset$ **then**

11: Run jobs in LMQ;

12: **end if**

13: **else**

14: Run jobs in LMQ;

15: **end if**

16:

17: **Upon job completion:**

18: **if** the job is an optional job **then**

19: Restart the E^R-pattern from the next job;

20: **else**

21: Shift the E^R-pattern;

22: **end if**

To ensure the effectiveness and efficiency of the dynamic approach, we have the following theorem.

Theorem 4: Algorithm 1 ensures the (m, k)-requirements for $\mathcal{T}$ if $\mathcal{T}$ is schedulable under the corresponding E-pattern. The complexity of the algorithm is $O(n)$,

Proof: Lemma 3 ensures that, among any k consecutive jobs, there are at least m jobs that are *equivalent* to the mandatory jobs. In addition, note that the execution of the mandatory jobs will never be interfered by the optional job J_i, either before or after the patterns are changed. Assume at a certain time point t' that a mandatory job missed its deadline. Let t_0 be the earliest time such that $[t_0, t']$ is a busy interval. This means that the mandatory workload between t_0 and t', denoted as $\sum_i W_i(t_0, t')$, is larger than the length of the time interval $|t' - t_0|$, i.e., $\sum_i W_i(t_0, t') > |t' - t_0|$. Similar to the proof of Theorem 3, let $t = t' - t_0$, which contradicts (8) since Algorithm 1 is based on E^R-pattern whose feasibility is guaranteed by Theorem 2

The complexity of Algorithm 1 mainly comes from the selection of the candidate optional job to be executed. Since there are at most n optional jobs in OPQ, the complexity is $O(n)$. ■

The energy efficiency of our online scheduling algorithm lies in the fact that it adjusts the mandatory/optional partition adaptively with the run-time conditions. It is particularly efficient when considering the fact that the actual execution time of a task can be much smaller than its worst case execution time. Moreover, during executions of the jobs in the HMQ, the dynamic slack reclaiming techniques in [3], [13] can be exploited to further reduce the energy.

TABLE I
VALID OPERATING POINTS OF IBM POWERPC 405LP [11]

	Setting		
Operating Points	**Frequency(MHz)**	**Voltage(V)**	**Relative Power(%)**
0	33	1.0	4
1	100	1.0	12
2	133	1.3	28
3	200	1.5	63
4	266	1.8	100

VII. EXPERIMENTAL RESULTS

Here, we use experiments to demonstrate the effectiveness of our approach. Specifically, we compare the energy saving performance of four approaches.

- MK_E: The task sets are statically partitioned into mandatory and optional job sets with the E-pattern. The mandatory jobs are executed with the highest processor speed. We use its results as the reference results.
- MK_{E-ST}: The task sets are statically partitioned based on the E-pattern. The processor speeds of the mandatory jobs for each task are scaled down based on the feasibility conditions stated in Theorem 2.
- MK_{R-ST}: The task sets are statically partitioned based on the R-pattern. The processor speeds of the mandatory job set for each task are scaled down based on the feasibility condition that can be derived similarly to that in Theorem 2 (see [24] for more details).
- MK_{E^R-HYB}: The task sets are statically partitioned based on the E^R-pattern. The processor speeds of the mandatory jobs for each task are scaled down first, and then Algorithm 1 is applied to adjust the pattern dynamically.

We used two processor models in our experiments, a theoretical model $P1$ and a practical model $P2$. In $P1$, we assumed the processor had five discrete voltage levels and the corresponding normalized speed frequencies were (0.2, 0.4, 0.6, 0.8, 1.0). We assumed that the processor speed is proportional to the supply voltage and the processor power consumption is a cubic function of the processor speed. The practical processor model $P2$ was derived from a commercial DVS processor model, i.e., IBM PowerPC 405LP [12]. The PowerPC 405LP processor can scale voltage and frequency via user-defined operating points [11] ranging from a high end of 266 MHz at 1.8 V to a low end of 33 MHz at 1.0 V. According to [11], the PowerPC 405LP processor can be operated at five operating points which are shown in Table I. At operating point 0, the processor is idle running at 33 MHz at 1.0 V.

We also used two groups of real-time task sets in our experiments: one was synthesized and the other one was drawn from practical applications. The synthesized task sets were run on the processor model $P1$ and the practical applications were run on the processor model $P2$.

A. Experiments With the Synthesized Task Sets

The periods of the synthesized task sets were randomly chosen in the range of [10–50 ms], and the deadlines were assumed to be equal to their periods. The worst case execution time (WCET) of a task at the highest voltage mode was set to be uniformly distributed from 1 to its deadline, and the actual

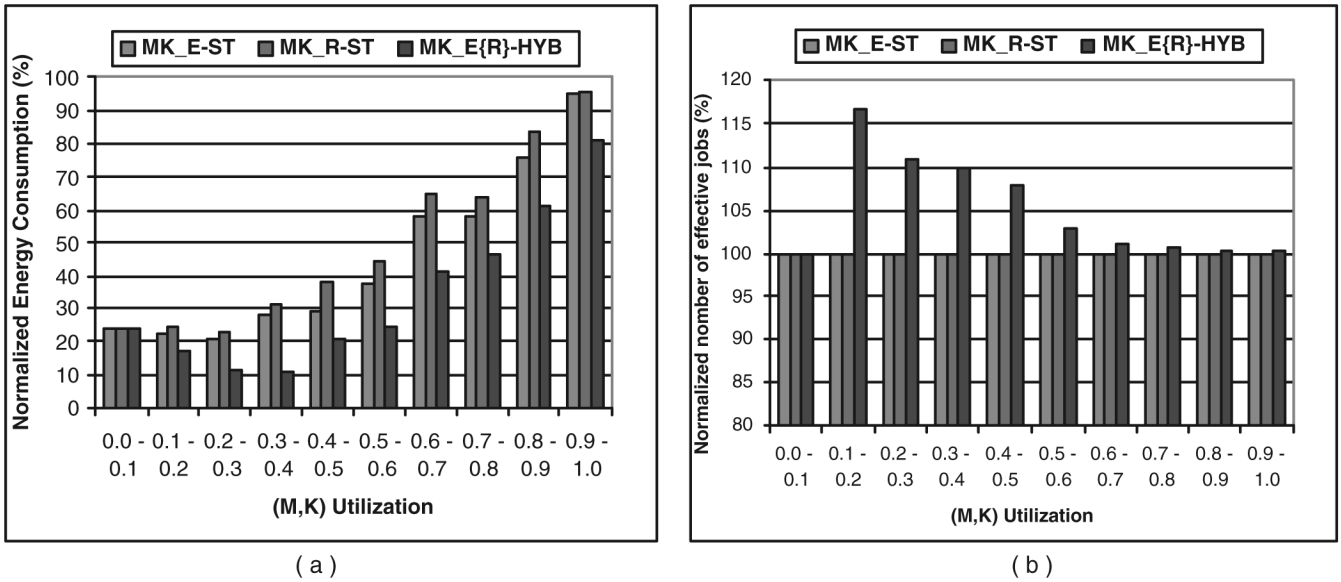


Fig. 6. (a) The average total energy consumption by the different approaches. (b) The average number of effective jobs by the different approaches.

execution time of a job was randomly picked from [0.4 WCET, WCET]. . The m_i and k_i for the (m, k)-constraints were also randomly generated such that k_i was uniformly distributed between 3–10, and $2 \leq m_i < k_i$. To investigate the energy performance of different approaches under different workload, we divided the total (m, k)-utilization, i.e., $\sum_i (m_i C_i / k_i T_i)$, into intervals of length 0.1. To reduce the statistical errors, we required that each interval contain at least 20 task sets schedulable with the R-pattern or until at least 5000 task sets within each interval had been generated. The energy consumption for each approach was normalized to that by MK_E, and the results are shown in Fig. 6(a). At the same time, we also collected the total number of jobs within $LCM(k_i T_i)$, $i = 0, \ldots, n-1$, that can meet their deadlines (we call them effective jobs) by each approach. These numbers were also normalized to that by MK_E shown in Fig. 6(b).

Fig. 6(a) and (b) shows clearly the effectiveness of our approach. Note that, with the help of a more accurate feasibility test, MK_{E-ST} can effectively lower the processor speed and reduce the energy consumption by up to 78% when compared with MK_E. When the offline approach is incorporated with the online approach, MK_{E^R-HYB} can achieve even better energy saving performance, i.e., up to 89% when compared with MK_E. It is particularly interesting to notice that MK_{E^R-HYB} can normally achieve better energy savings while at the same time provide better QoS levels (up to 17% more effective jobs) than MK_{E-ST}, as shown in Fig. 6(b). This is because, by running optional jobs at low processor speed and dynamically varying the (m, k)-pattern, MK_{E^R-HYB} saves the energy needed to run mandatory jobs at high processor speeds which are energy consuming. Therefore, more energy can be saved even more effective jobs are executed with MK_{E^R-HYB}. Further, Fig. 6(a) also shows that MK_{E^R-HYB} outperforms MK_{R-ST} significantly in term of energy savings, i.e., as much as 60% as shown in Fig. 6(a). This is because the R-patterns aggregates the mandatory workload in a relatively short time interval, which makes it more difficult to scale down the processor speeds.

TABLE II
AVERAGE NUMBERS OF FEASIBLE TASK SETS BY MK_{E^R-HYB} AND MK_{R-ST}

(m,k)	**Feasible Task Sets**	
Util	$E(E^R) - pattern$	$R - pattern$
0.0-0.1	100	100
0.1-0.2	100	100
0.2-0.3	100	100
0.3-0.4	100	97
0.4-0.5	100	85
0.5-0.6	100	81
0.6-0.7	100	70
0.7-0.8	100	56
0.8-0.9	100	47
0.9-1.0	100	30

From Fig. 6(a) and (b) we can also notice that the energy saving performance and the number of effective jobs vary with different (m, k)-utilizations. In general, MK_{E^R-HYB} can achieve better energy performance when the (m, k)-utilization is not extremely low or extremely high. When (m, k)-utilization is very low (e.g., between 0.0–0.1), there is not much energy-saving difference among MK_{E-ST}, MK_{R-ST}, and MK_{E^R-HYB}, since most mandatory jobs can be run at the lowest processor speed (even with the R-pattern) and, thus, the execution of optional jobs has no significant benefit in saving energy. Under such a scenario, not many optional jobs are selected for execution since their energy gains are not high enough. Therefore, the numbers of effective jobs are also very close for MK_{E-ST}, MK_{R-ST}, and MK_{E^R-HYB}. On the other hand, when the utilization is very high (e.g., between 0.9–1.0), MK_{E-ST} and MK_{R-ST} have similar energy saving performance. This is because m is very close to k under this circumstance and MK_{E-ST} and MK_{R-ST} tend to result in similar mandatory/optional job partitions. In addition, when utilization is very high, there are less opportunities to execute the optional jobs as shown in Fig. 6(b). Even so, MK_{E^R-HYB} still consumes 18% less energy consumption compared with that by MK_{E-ST}, as shown in Fig. 6(a), due to the pattern adjustment.

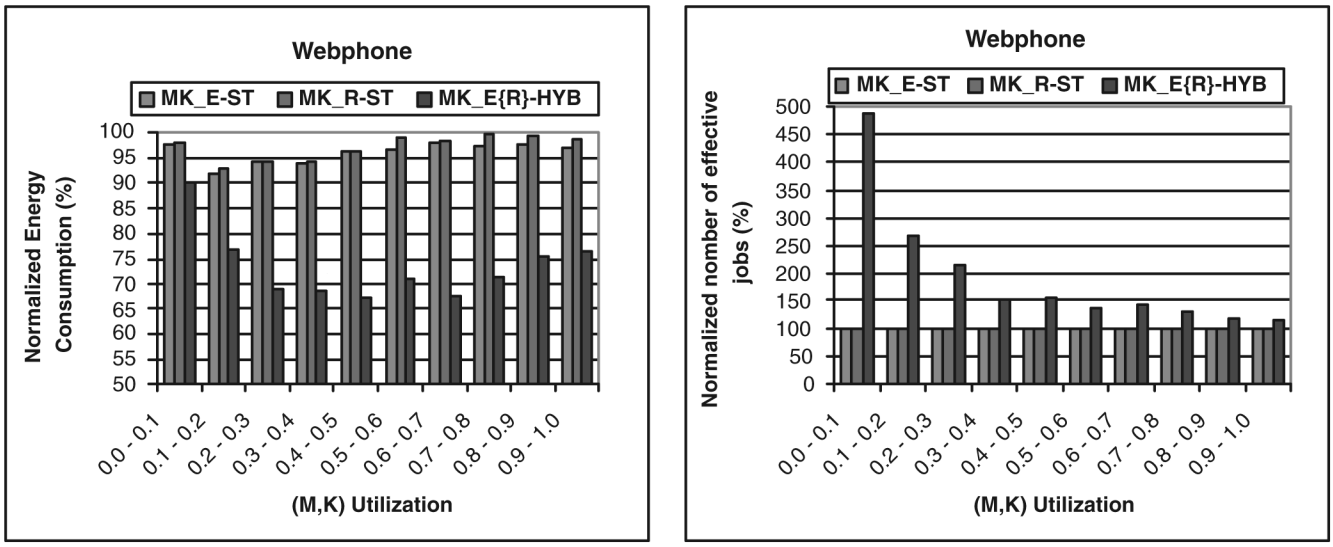


Fig. 7. (a) Energy consumption for webphone. (b) The average number of effective jobs for webphone.

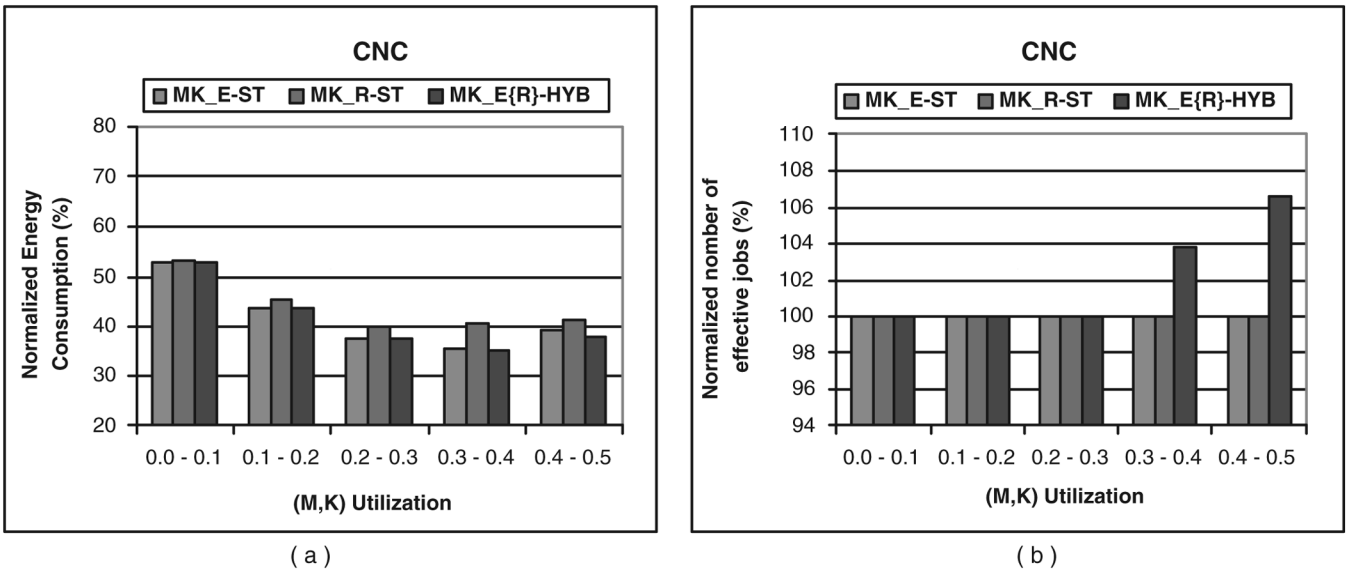


Fig. 8. (a) Energy consumption for CNC. (b) The average number of effective jobs for CNC.

To further study the (m,k)-guarantee capability by MK_{E-ST} and MK_{R-ST}, we performed another set of experiments. We randomly generated periodic task sets such that within each (m,k) utilization interval no less that 100 task sets were schedulable by the E-pattern or at least 5000 different task sets have been generated for each interval. At the same time, we evaluated and recorded how many of these task sets were schedulable with the R-pattern. The results are normalized and presented in Table II. As shown in Table II, when the (m,k)-utilization is relatively low, i.e., less than 0.3, the (m,k)-guarantee capability of these patterns are very close to each other. However, when the utilization is higher, i.e., larger than 0.3, the E-pattern and the E^R-pattern have much stronger (m,k)-guarantee capability than the R-pattern. For example, according to Table II, when the (m,k)-utilization is around 0.7–0.8, nearly half of the task sets that are schedulable with the E-pattern cannot be schedulable with the R-pattern. These results conform to our analysis before and shows that the E-pattern has a much better schedulability and therefore energy-saving potential than the R-pattern.

B. Experiments With Real Applications

Next, we tested our techniques in a more practical environment. The test cases contained three real-world applications: webphone [32], computerized numerical control (CNC) machine controller [25], and inertial navigation system (INS) [1]. The timing parameters such as the deadlines, periods, and execution times were adopted from these practical applications. The actual execution time of a job was randomly picked from [0.4 WCET, WCET] and the (m,k)-constraints were generated as we did for the synthesized task sets. The normalized (to those of MK_E) energy consumptions and numbers of effective jobs are and shown in Figs. 7–9.

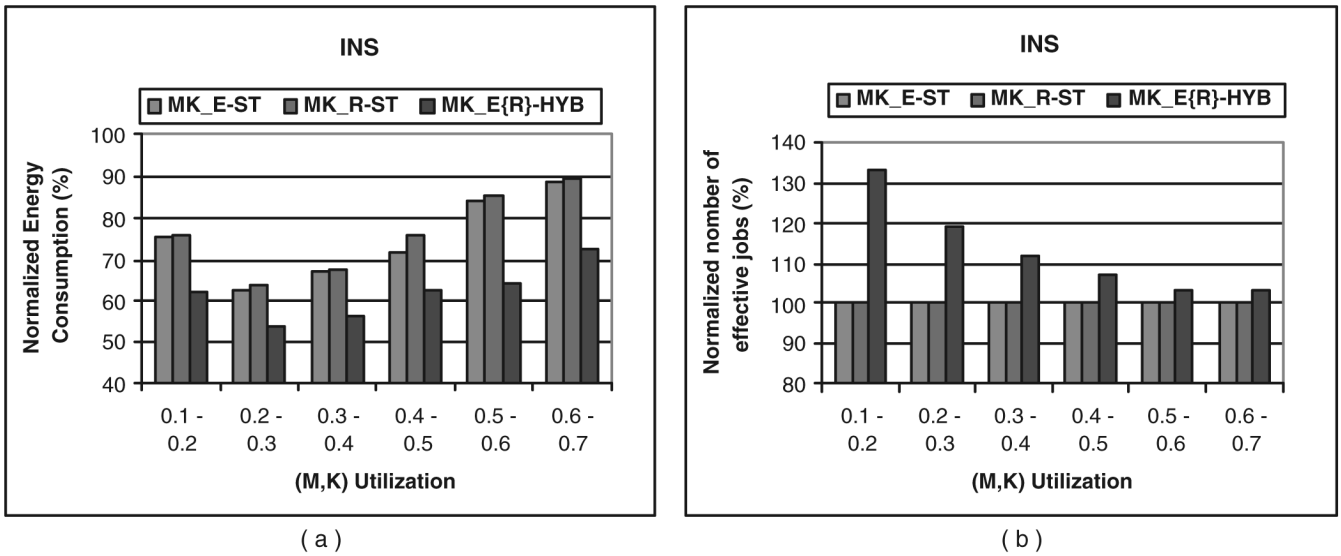


Fig. 9. (a) Energy consumption for INS. (b) The average number of effective jobs for INS.

The experimental results based on the practical applications further demonstrate the effectiveness of our static and dynamic hybrid approach in saving energy. For the webphone application, as shown in Fig. 7(a), the offline approach (MK_{E-ST}) reduces the energy consumption only by no more than 8%. After being incorporated with the online approach, the energy saving is increased up to 35%. For CNC application, as shown in Fig. 8(a), the static approach is much more helpful. This is because the utilization of this application is relatively low $(U \leq 0.489)$. Therefore, the processor speeds can be significantly scaled down by the static approach. Even so, the dynamic approach can still achieve up to 5% more additional energy savings. For the INS application, as shown in Fig. 9(a), the combined static/dynamic approach can achieve up to 47% energy savings. The experimental results also show that the static approach according to the E-pattern outperform that with the R-pattern in all three applications. In addition, the numbers of effective jobs with our integrated static/dynamic approach are also significantly increased as shown in Figs. 7(b), 8(b), and 9(b).

VIII. Conclusion

Energy consumption and QoS guarantee are two of the most critical factors for the successful design of pervasive real-time computing platforms. The beauty of the (m, k)-model lies in the fact that it defines the QoS constraints not only from the statistical but also the deterministic sense based on user's requirements. In this paper, we propose a hybrid approach to guarantee the (m, k)-requirements for real-time systems and minimize the energy consumption at the same time. Our hybrid approach consists of two phases, i.e., the offline phase and the online phase. During the offline phase, the real-time jobs are statically partitioned into mandatory/optional jobs. The processor speeds for the mandatory jobs are scaled down based on proper schedulability analysis. The online phase further improves the energy saving performance by exploiting the variations during the run-time environment. By judiciously executing the optional jobs at low processor speeds and adjusting the mandatory/optional job partitions correspondingly, significant energy savings can be obtained. As shown in our experiments, with excellent adaptivity to the run-time conditions, the proposed DVS scheduling algorithm outperforms previous research significantly in terms of energy savings, QoS levels, as well as the range of real-time systems with (m, k)-constraints that can be guaranteed.

Appendix I
Proof for Theorem 2

As shown [29] that, if the mandatory jobs are determined according to (8), for the first p_i jobs of τ_i, there are $l_i(t) = \lceil (m_i/k_i)p_i \rceil$ jobs that are mandatory. Therefore, the total mandatory workload within interval $[0, t]$ that has to be finished by time t, denoted by $W(0, t)$, can be formulated as follows:

$$W(0,t) = \sum_i \left(\left\lceil \frac{m_i}{k_i} \left\lceil \frac{t - D_i}{T_i} \right\rceil^+ \right\rceil \right) C_i. \tag{13}$$

The necessity follows from (2) directly. To prove the sufficiency, first, we show that it is sufficient to check the deadlines within the first busy interval. Suppose the first deadline miss occurs at t, which is not in the first busy interval. Then, we can always find a $t' < t$ such that the processor is busy during $[t', t]$ and t' is the starting point of this busy interval. Therefore, the total mandatory workload in interval $[t', t]$ can be formulated as follows:

$$W(t',t) = \sum_i \left(\left\lceil \frac{m_i}{k_i} \left\lceil \frac{t - D_i}{T_i} \right\rceil^+ \right\rceil - \left\lceil \frac{m_i}{k_i} \left\lceil \frac{t'}{T_i} \right\rceil^+ \right\rceil \right) C_i. \tag{14}$$

Since $\lceil x_1 + x_2 \rceil \leq \lceil x_1 \rceil + \lceil x_2 \rceil$ and $\lceil x_1 + x_2 \rceil^+ \leq \lceil x_1 \rceil^+ + \lceil x_2 \rceil^+$ for any $x_1, x_2 \in R$, we have

$$\left\lceil \frac{m_i}{k_i} \left\lceil \frac{t - D_i}{T_i} \right\rceil^+ \right\rceil \leq \left\lceil \frac{m_i}{k_i} \left\lceil \frac{t - t' - D_i}{T_i} \right\rceil^+ \right\rceil + \left\lceil \frac{m_i}{k_i} \left\lceil \frac{t'}{T_i} \right\rceil^+ \right\rceil. \quad (15)$$

Therefore,

$$W(t', t) \leq \sum_i \left\lceil \frac{m_i}{k_i} \left\lceil \frac{t - t' - D_i}{T_i} \right\rceil^+ \right\rceil C_i. \quad (16)$$

Since the deadline miss occurs at t, we have $W(t', t) > t - t'$, and thus

$$\sum_i \left\lceil \frac{m_i}{k_i} \left\lceil \frac{t - t' - D_i}{T_i} \right\rceil^+ \right\rceil C_i > t - t'. \quad (17)$$

On the other hand, since no deadline miss happens earlier than t, we have

$$W(0, t - t') = \sum_i \left\lceil \frac{m_i}{k_i} \left\lceil \frac{t - t' - D_i}{T_i} \right\rceil^+ \right\rceil C_i \leq t - t'. \quad (18)$$

This contradicts (17).

Next, we show that it is sufficient to check the deadlines within $t_0 = LCM(T_i)$, $i = 0, \ldots, (n-1)$, if the ending point of the first busy interval is larger than t_0. Again, we use contradiction. Assume that all deadlines were met between 0 and t_0 but some mandatory job J_i missed its deadline at time t' within the first busy interval, where $t' > t_0$. Then, according to the assumption, we have

$$\sum_i W_i(0, t') = \sum_i \left\lceil \frac{m_i}{k_i} \left\lceil \frac{t' - D_i}{T_i} \right\rceil^+ \right\rceil C_i > t'. \quad (19)$$

However, we already knew that all deadlines were met between 0 and t_0 and $(0, t_0]$ is within the first busy interval, so

$$\sum_i W_i(0, t_0) = \sum_i \left\lceil \frac{m_i}{k_i} \left\lceil \frac{t_0 - D_i}{T_i} \right\rceil^+ \right\rceil C_i \leq t_0. \quad (20)$$

From (19) and (20), we have

$$\sum_i W_i(t_0, t') = \sum_i W_i(0, t') - \sum_i W_i(0, t_0) > t' - t_0. \quad (21)$$

However, according to Lemma 1 [see 1)], we have

$$\sum_i W_i(0, t' - t_0) \geq \sum_i W_i(t_0, t') > t' - t_0. \quad (22)$$

This means that the mandatory job will miss the deadline at $t' - t_0$. If $(t' - t_0) \leq t_0$, then it contradicts the assumption. Otherwise, if $(t' - t_0) > t_0$, we can let $t'' = (t' - t_0)$ and repeat the above procedure by replacing t' with t''. Finally, we can always find a $\breve{t}$ within $(0, t_0]$ such that $\sum_i W_i(0, \breve{t}) > \breve{t}$, which still contradicts the assumption.

Appendix II
Proof for Theorem 3

To prove Theorem 3, we first introduce the following lemma.

Lemma 4: Let w be the number of mandatory jobs for τ_i between $[0, t]$ according to its E-pattern. For any other (m, k)-pattern, we can always find a $t'(t' \geq 0)$ such that the number of mandatory jobs according to the (m, k)-pattern within $[t', t' + t]$ is no less than w.

Proof: Use Contradiction. Let $\mathcal{R}$ and $\mathcal{R}'$ be the mandatory job sets determined by the E-pattern and any other (m, k)-pattern, respectively. Without loss of generality, we assume that $t = pT_i$, $p \in Z$. Consider the interval $[0, pk_iT_i]$. To satisfy the (m, k)-constraints, there are at least $p \times m_i$ mandatory jobs in this interval for both $\mathcal{R}$ and $\mathcal{R}'$. Specifically, for $\mathcal{R}$, there are exactly $p \times m_i$ mandatory jobs in this interval from Lemma 1. Now, think about the intervals $[0, pT_i]$, $[pT_i, (p+1)T_i], \ldots, [p(k_i - 1)T_i, pk_iT_i]$. From Lemma 1, in $\mathcal{R}$, the number of mandatory jobs within interval $[0, pT_i]$, i.e., w, is the largest, and the difference between the numbers of mandatory jobs within any two of these intervals is no more than 1. If, for $\mathcal{R}'$, we assume that the number of mandatory jobs in each interval is strictly less than w, then the overall number of mandatory jobs within $[0, pk_iT_i]$ must be less than $p \times m_i$. This contradicts the fact that R' is determined according to a valid (m, k)-pattern. ■

Based on Lemma 4, we use contradiction to prove Theorem 3 as follows. Suppose $\mathcal{R}'$ is schedulable and $\mathcal{R}$ is not. Let us assume some mandatory job in $\mathcal{R}$ first misses its deadline at t. From Theorem 2, we know that t must be located in the first busy interval. Then, we have the total mandatory work demand according to E-patterns between $[0, t]$, denoted as $\sum_i W_i(0, t)$, is larger than t, i.e, $\sum_i W_i(0, t) > t$.

On the other hand, for any other arbitrary pattern, from Lemma 4, we can always find an interval $[t1, t2]$ with $t2 - t1 = t$ such that the corresponding work demand for τ_i, denoted as $W_i'(t1, t1 + t)$, is no less than $W_i(0, t)$, i.e.,

$$W_i'(t1, t1 + t) \geq W_i(0, t).$$

Since the mandatory jobs are determined by repeating the (m, k)-patterns, we can therefore observe the workload from τ_i within interval $[t1, t2]$ periodically repeated with period k_iT_i. If k_iT_i, $i = 0, 1, \ldots, n-1$ are co-prime, according to the General Chinese Remainder Theorem [16], all of these "periodic events" will eventually start at one single time point t'. Since

$$\sum_i W_i'(t', t' + t) \geq \sum W_i(0, t) > t$$

there must be a deadline miss before $t' + t$ because the total mandatory work demand between $[t', t' + t]$ exceeds t. This contradicts that $\mathcal{R}'$ is schedulable.

REFERENCES

[1] A. Burns, K. Tindell, and A. Wellings, "Effective analysis for engineering real-time fixed priority schedulers," *IEEE Trans. Software Eng.*, vol. 21, no. 5, pp. 920–934, May 1995.

[2] A. K. Mok and W. Wang, "Window-constraint real-time periodic task scheduling," in *Proc. RTSS*, 2001, p. 15.

[3] H. Aydin, R. Melhem, D. Mosse, and P. Alvarez, "Determining optimal processor speeds for periodic real-time tasks with different power characteristics," in *Proc. ECRTSS01*, Jun. 2001, p. 225.

[4] G. Bernat and A. Burns, "Combining (n, m)-hard deadlines and dual priority scheduling," in *Proc. RTSS*, Dec. 1997, pp. 46–57.

[5] ——, "Weakly hard real-time systems," *IEEE Trans. Comput.*, vol. 50, no. 4, pp. 308–321, Apr. 2001.

[6] G. Bernat and R. Cayssials, "Guaranteed on-line weakly-hard real-time systems," in *Proc. RTSS*, 2001, p. 25.

[7] R. Davis and A. Wellings, "Dual priority scheduling," in *Proc. RTSS*, 1995, pp. 100–109.

[8] M. Hamdaoui and P. Ramanathan, "A dynamic priority assignment technique for streams with (m, k)-firm deadlines," *IEEE Trans. Comput.*, vol. 44, no. 12, pp. 1443–1451, Dec. 1995.

[9] S. Hua and G. Qu, "Energy-efficient dual-voltage soft real-time system with (m, k)-firm deadline guarantee," in *Proc. CASE'04*, 2004, pp. 116–123.

[10] S. Hua, G. Qu, and S. Bhattacharyya, "Energy reduction techniques for multimedia applications with tolerance to deadline misses," in *Proc. DAC*, 2003, pp. 131–136.

[11] "Dynamic power management for embedded systems," *IBM J. Res. Develop.*, pp. 1–25, Nov. 2002.

[12] G. C. K. Nowka and B. Brock, "The design and application of the PowerPC 405LP energy-efficient system on chip," *IBM J. Res. Develop.*, vol. 47, no. 5/6, pp. 631–639, Sep./Nov. 2003.

[13] W. Kim, J. Kim, and S. L. Min, "A dynamic voltage scaling algorithm for dynamic-priority hard real-time systems using slack analysis," in *Proc. DATE*, 2002, p. 788.

[14] G. Koren and D. Shasha, "Skip-over: Algorithms and complexity for overloaded systems that allow skips," in *Proc. RTSS*, 1995, p. 110.

[15] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: Exact characterization and average case behavior," in *Proc. RTSS*, 1989, pp. 166–171.

[16] J. Y.-T. Leung and J. Whitehead, "On the complexity of fixed-priority scheduling of periodic, real-time tasks," *Performance Eval.*, vol. 2, pp. 237–250, 1982.

[17] J. Liebeherr, D. W. Wrege, and D. Ferrari, "Exact admission control for networks with a bounded delay service," *IEEE/ACM Trans. Netw.*, vol. 4, pp. 885–901, Jul. 1996.

[18] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *J. ACM*, vol. 17, no. 2, pp. 46–61, 1973.

[19] J. Liu, *Real-Time Systems*. Upper Saddle River, NJ: Prentice-Hall, 2000.

[20] J. R. Lorch and A. J. Smith, "Operating system modifications for task-based speed and voltage scheduling," in *Proc. MOBISYS 2003*, 2003, pp. 215–229.

[21] T. Ma and K. Shin, "A user-customizable energy adaptive combined static/dynamic scheduler for mobile applications," in *Proc. RTSS*, 2000, pp. 227–236.

[22] B. Mochocki, X. Hu, and G. Quan, "A unified approach to variable voltage scheduling for nonideal DVS processors," *IEEE Trans. Comput.-Aided Design Integr. Circtuits Syst.*, vol. 23, no. 9, pp. 1370–1377, Sep. 2004.

[23] M. Spuri, Analysis of Deadline Scheduled Real-Time Systems. INRIA, France, Rapport de Recherche RR-2772, 1996.

[24] L. Niu and G. Quan, Energy-Aware Scheduling for Real-Time Systems With (m, k)-Guarantee. Dept. Comput. Sci. Eng., Univ. South Carolina, Tech. Rep. TR-2005-005, 2005.

[25] N. Kim, M. Ryu, S. Hong, M. Saksena, C. Choi, and H. Shin, "Visual assessment of a real-time system design: A case study on a CNC controller," in *Proc. RTSS*, Dec. 1996, p. 300.

[26] Q. Qiu, Q. Wu, and M. Pedram, "Dynamic power management in a mobile multimedia system with guaranteed quality-of-service," in *Proc. DAC*, 2001, pp. 834–839.

[27] G. Quan and X. Hu, "Enhanced fixed-priority scheduling with (m, k)-firm guarantee," in *Proc. RTSS*, 2000, pp. 79–88.

[28] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek, "A resource allocation model for QoS management," in *Proc. RTSS*, Dec. 1997, pp. 298–307.

[29] P. Ramanathan, "Overload management in real-time control applications using (m, k)-firm guarantee," *IEEE Trans. Parallel. Distrib. Syst.*, vol. 10, no. 6, pp. 549–559, Jun. 1999.

[30] K. F. S. Reinhardt and T. Mudge, "Automatic performance-setting for dynamic voltage scaling," in *Proc. MOBICOM'01*, 2001, pp. 260–271.

[31] I. Ripoll, A. Crespo, and A. Mok, "Improvement in feasbility testing for real-time tasks," *J. Real-Time Syst.*, vol. 11, pp. 19–40, 1996.

[32] D. Shin, J. Kim, and S. Lee, "Intra-task voltage scheduling for low-energy hard real-time applications," *IEEE Design Test Comput.*, vol. 18, no. 2, pp. 20–30, Mar.-Apr. 2001.

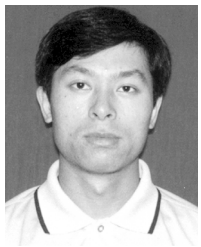
[33] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," *OSDI*, pp. 13–23, 1994.

[34] R. West and K. Schwan, "Dynamic window-constrained scheduling for multimedia applications," in *Proc. ICMS*, 1999, pp. 87–91.

[35] A. F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced cpu energy," in *Proc. AFCS*, 1995, pp. 374–382.

[36] W. Yuan and K. Nahrstedt, "Energy-efficient soft real-time cpu scheduing for mobile multimedia systems," in *Proc. SOSP*, 2003, pp. 149–163.

[37] Q. Zheng and K. G. Shin, "On the ability of establishing real-time channels in point-to-point packet-switched networks," *IEEE Trans. Commun.*, vol. 42, no. 2/3/4, pp. 1096–1105, Feb./Mar./Apr. 1994.



Linwei Niu (S'04) received the B.S. degree in computer science and technology from Peking University, Beijing, China, in 1998, and the M.S. degree in computer science from the State University of New York at Stony Brook in 2001. He is currently working toward the Ph.D. degree in the Department of Computer Science and Engineering, University of South Carolina, Columbia.

His research interests are hardware/software co-design and real-time embedded systems.

Gang Quan (S'01–M'03) received the B.S. degree from Tsinghua University, Beijing, China, the M.S. degree from the Chinese Academy of Sciences, Beijing, China, and the Ph.D. degree from the University of Notre Dame, South Bend, IN.

He is currently an Assistant Professor with the Department of Computer Science and Engineering, University of South Carolina. Columbia. His research interests include real-time systems, power-aware design, communication networks, and reconfigurable computing.

Dr. Quan was the recipient of the National Science Foundation CAREER Award in 2006 and the Best Paper Award at the 38th Design Automation Conference in 2001.