A Hybrid Static/Dynamic DVS Scheduling for Real-time Systems With (m,k)-Guarantee

Linwei Niu Gang Quan
Department of Computer Science and Engineering
University of South Carolina
Columbia, SC 29208
{niul,gquan}@cse.sc.edu

Abstract

Energy reduction is critical to increase the mobility and to extend the mission period in the development of today's pervasive computing systems. On the other hand, however, energy reduction must be subject to the requirements not to compromise the quality of service (QoS) that these systems need to provide. 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 with the requirements of 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. To deal with the dynamic characteristics of such applications and systems, we propose a hybrid static/dynamic scheduling approach that can efficiently reduce the energy consumption while guaranteeing the (m,k)-constraints. The experimental results demonstrate that our proposed techniques outperform previous research significantly in terms of both the energy savings and QoS that can be achieved.

1 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., [2, 9, 25]) have been proposed to reduce the energy consumption of real-time computing system. Most of these techniques have targeted hard real-time systems, *i.e.*, the systems requiring that all the 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 which do not finish by their deadlines can still be completed with a reduced value [13]; or they can simply be dropped without compromising the desired QoS levels. Energy reduction under QoS requirement falls within the framework of more general resource management/scheduling, such as the QoS-based Resource Allocation Model (Q-RAM) [20]. 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. The techniques based on the traditional hard real-time systems become inefficient or inadequate when QoS requirements are imposed on the systems.

Recently, there has been increasing interest that incorporates the Dynamic Voltage Scaling (DVS) techniques in real-time scheduling to deal with the power/energy conservation 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. [23, 22, 14, 15]) 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 QoS-guarantee in mind. A predominated portion of the current guarantee research in power aware scheduling has to do with the *statistic* service guarantee. For example, Qiu *et al.* presented a technique [18] to statistically guarantee the QoS for a real-time embedded system. Yuan *et al.* [26] proposed incorporating the stochastic analysis into DVS for soft deadline systems to provide statistic QoS guarantee. With QoS requirements formulated as a tolerable statistic deadline miss rate, Hua *et al.* [8] introduced several techniques to exploit processor slack time due to the deadline miss to reduce the energy.

The statistic guarantee ensures a 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 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 the real-time system, the system should not only support the overall guarantee of the QoS statistically, but also be able to provide a lower bounded, predictable level of QoS locally. Hamdaoui *et al.* [6] proposed the (m,k)-model that can well serve 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. West *et al.* [24] introduced another similar model, called the *window-constrained* model. The major difference between these two models is that the (m,k)-constraint requires at least m jobs meet their deadlines for *any* k consecutive jobs, while the *window constraint* requires that within any *non-overlapped* and *consecutive* windows containing k jobs, at least m of them can meet their deadlines. It can be concluded that *window constraints* are weaker than the $(m,k)-$constraints, as if a schedule is feasible under the $(m,k)-$constraints, it is also feasible under the window constraints.

For its intuitiveness and capability of capturing not only statistical but also deterministic QoS requirements, (m,k)-model has been widely studied, e.g., [3, 19, 6, 21, 7]. Quan *et al.* [19] formally proved that the problem of scheduling with $(m,k)-$guarantee is NP-hard in the strong sense. To guarantee the (m,k)-constraints, Ramanathan *et al.* [21] 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 the computing resources. Quan *et al.* [19] improved this partitioning strategy by reducing the maximal interference between the mandatory jobs. Bernat *et al.* [4] proposed to use the Bi-Modal scheduler to schedule the systems with $(m,k)-$constraints. The tasks are first scheduled according to the generic scheduling policy in the *normal* mode, and switched to the *panic* mode if the dynamic failure is likely to occur. All these work primarily targets at systems with fixed priority assignment. Note that, even with the *window constraints*, the guaranteed scheduling problem is NP-hard as shown in [1]. Deterministic assurance with this model can only be guaranteed for very limited range of systems, such as those that all tasks have the same unit size execution times [1]. In addition, none of these approaches have taken energy/power consumption into consideration.

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 off-line and thus 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 partition problem as well as the schedulability analysis problem are NP-hard in the strong sense [19].

In this paper, we propose a hybrid approach to determine the mandatory job sets for real-time systems with (m,k)-constraints. In our approach the mandatory jobs are statically determined but can be dynamically updated during run-time while still guaranteeing the (m,k)-constraints. A scheduler, based on the dual priority scheduler [5], is designed to dynamically determine if a job should be mandatory/optional and the corresponding processor speed to maximizing 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 rest of the paper is organized as follows. Section 2 introduces the system model and problem formulation as well as the motivations. Section 3 presents some theoretical results that form the basis for our techniques. Section 4 introduces our new approaches in more details. The effectiveness and energy efficiency of our approach are demonstrated using simulation results in section 5. In section 6, we offer conclusions for this paper.

2 Preliminary

In this section, we first formulate the problem formally, followed by the introduction of some concepts as well as observations important to our approach. We then present the motivations for our approach.

2.1 System models and problem formulation

The real-time system considered in this paper contains n independent periodic tasks, $\mathcal{T} = \{\tau_0, \tau_1, \cdots, \tau_{n-1}\}$, scheduled according to the earliest deadline first (EDF) policy [12]. 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, D_i ($D_i \leq T_i$), and C_i represent the period, the deadline and 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 $\mathcal{V} = \{V_1, ..., V_{max}\}$, 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 $\mathcal{S} = \{S_1, ..., 1\}$. 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 $\frac{C_i}{S_j}$.

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

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

2.2 Mandatory/optional job partitioning

To solve Problem 1, one has to deal with two highly co-related problems: to determine if a job should be mandatory or optional, and to schedule these jobs most efficiently. As shown in [19], both problems are NP-hard problems even without the energy conservation consideration. 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.

Mandatory/optional partitioning involves the partition on the infinite job sequences. To ease the static analysis as well as to reduce the implementation cost, we adopt the concept of $(m,k) - pattern$ as introduced in [19].

Definition 1 *[19] The* ***(m,k)-pattern*** *of task* τ_i*, denoted by* Π_i*, is a binary string* $\Pi_i = \{\pi_{i0}\pi_{i1}...\pi_{i(k_i-1)}\}$ *which satisfies the following: (i)* τ_{ij} *is a mandatory job if* $\pi_{ij} = 1$ *and optional if* $\pi_{ij} = 0$*, and (ii)* $\sum_{j=0}^{k_i-1} \pi_{ij} = m_i$*.*

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.

Two static $(m,k)-$patterns are reported in literature. The first one is proposed by Ramanathan *et al.* [21] as follows.

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

The (m,k)-pattern defined with formula (1) has some interesting properties 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 equation (1). Then (i) for any* $t > 0$*, the interval* $[0,t]$ *has the largest number of mandatory jobs compared with any other intervals with the same length* $|t|$*; (ii) for any* k_i *consecutive jobs of* τ_i*, there are exactly* m_i *mandatory jobs; (iii) for any two subsequences of task* τ_i *that contain* $p_i (p_i > 0)$ *consecutive jobs, the difference of the numbers of mandatory jobs is no more than 1.*

Proof: Conclusion(i) has been proved in [21] and conclusion (ii) can be readily derived from Lemma 2 in [21]. We next prove conclusion (iii) as follows.

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 $(a_i \neq b_i, a_i, b_i \geq 0)$, 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 equation (1), for the first q_i jobs (from job 0 to job $q_i - 1$) of τ_i, there are $l_i(q_i) = \lceil \frac{m_i}{k_i} q_i \rceil$ jobs that are mandatory [21]. Therefore,

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

and similarly,

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

Assume $N_i(a_i, a_i + p_i - 1) \geq N_i(b_i, b_i + p_i - 1)$, then

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

(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 …

Figure 1. Examples of mandatory jobs based on R-patterns, E-patterns, and E^R-patterns.

Since $\lceil x_1+x_2\rceil \leq \lceil x_1\rceil+\lceil x_2\rceil$ and $\lceil x_1+x_2\rceil \geq \lceil x_1\rceil+\lfloor x_2\rfloor$ for any $x_1, x_2 \in R$, it follows that

$$\lceil \frac{m_i}{k_i}(a_i+p_i)\rceil \leq \lceil \frac{m_i}{k_i}(a_i+1)\rceil + \lceil \frac{m_i}{k_i}(p_i-1)\rceil,$$

and

$$\lceil \frac{m_i}{k_i}(b_i+p_i)\rceil \geq \lceil \frac{m_i}{k_i}(b_i+1)\rceil + \lfloor \frac{m_i}{k_i}(p_i-1)\rfloor.$$

Therefore, we have

$$\begin{aligned} & \mid N_i(a_i, a_i+p_i-1) - N_i(b_i, b_i+p_i-1) \mid \\ \leq\ & \lceil (p_i-1)\frac{m_i}{k_i}\rceil - \lfloor (p_i-1)\frac{m_i}{k_i}\rfloor \\ \leq\ & 1. \end{aligned}$$

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

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

□

Lemma 1 implies that, with formula (1), a minimal set of mandatory jobs are determined. Moreover, according to Lemma 1, formula (1) helps to spread out the mandatory jobs *evenly* in each task along the time. We therefore call this mandatory job pattern as *the evenly distributed pattern*, or simply *E-pattern*.

The second partition strategy was proposed by Koren *et al.* [10]. According to this scheme, a job τ_{ij}, *i.e.*, the jth job of task τ_i, is determined to be mandatory if

$$\pi_{ij} = \begin{cases} 1 & 0 \leq j \bmod k_i < m_i \\ 0 & \text{otherwise} \end{cases} \qquad j = 0, 1, \cdots, k_i - 1 \tag{2}$$

We borrow the initial terminology of this strategy and refer this mandatory job pattern as *the deeply-red pattern* or *R-pattern*. The mandatory jobs defined using their R-patterns have the following interesting property.

Lemma 2 *Let $\mathcal{J}_r$ be the mandatory job set selected from all the jobs in $\mathcal{T}$ according to equation (2). If $\mathcal{J}_r$ is schedulable, then the mandatory job set selected from any other $(m,k)-$pattern is also schedulable.*

Even though this property is proved in [19] for fixed-priority case, it can also be proved under the EDF scheduling policy [17]. This property is important in guaranteeing the (m,k)-constraints dynamically, as it ensure that 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. An immediate conclusion that follows this lemma is that mandatory jobs determined by E-patterns are easier to be schedulable than those by R-patterns. Figure 1 shows several examples of mandatory/optional jobs determined according to their R-patterns and E-patterns (The reverse evenly distributed pattern will be introduced later).

2.3 Motivations

Hua *et al.* [7] adopted a greedy approach to minimize the energy consumption for *underloaded* real-time systems running on a dual-voltage mode processor. When a new job arrives, it is executed 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 the task set can 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 $\frac{14.5}{15}$ and it is schedulable under EDF. For the dual-voltage mode processor, without loss of generality, we assume 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 all tasks start at time 0. Figure 2 shows the task schedule according to this greedy approach.(The rectangles represent the executions of jobs and the height of them represent the speeds of the jobs.)

At time $t = 0$, since task τ_1 has (m,k)-constraint of $(1,1)$ and thus each job is required to meet its deadline, τ_{11} is executed with high processor speed. At time $t = 2$, when τ_{11} finishes its execution, the low processor speed will be assigned to execute τ_{21} according to the greedy approach

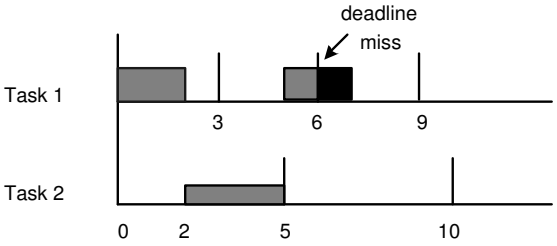

Figure 2. The greedy approach in [7] fails to guarantee the schedulability of an underloaded task set ($\tau_1 = (3,3,2,1,1)$; $\tau_2 = (5,5,1.5,1,2)$).

since τ_2 has the (m,k)-constraint of $(1,2)$ and τ_{21} is not required to meet the deadline. According to EDF, only after τ_{21} ends at $t = 5$ can τ_{12} be executed. However, even if it is executed with high speed, it will still miss the deadline at time $t = 6$ and therefore cause a dynamic failure.

The uniform treatment of mandatory and optional jobs during the scheduling process in this greedy approach is one of the major contributions to the dynamic failure. In considering that, we developed an approach [17] based on the dual priority scheme [5]. The mandatory jobs are pushed into the mandatory job queues with their priorities promoted to the high priority band when necessary, while the priorities of the optional jobs always stay at the low priority band. This ensures that the execution of optional jobs will not prevent mandatory jobs from meeting their deadlines and cause the problems as shown in Figure 2. The feasibility of the mandatory jobs and thus the (m,k)-guarantee can be ensured as long as mandatory jobs according to R-pattern are schedulable (from Lemma 2). The problem, however, lies in the conservation of feasibility test using R-pattern, *i.e.*, assuming all the mandatory workload are accumulated within a small interval. The consequence is that many mandatory job sets are in fact schedulable even though the feasibility analysis based on R-pattern predicts otherwise.

Recall that E-patterns help to evenly distributed the mandatory workload and therefore have a better schedulability. One intuitive approach would be the one that statically assigns all the mandatory jobs according to E-pattern to the higher priority band. Two problems may exist in this approach. First, even though the mandatory jobs for each task are evenly distributed, the overall mandatory workload are not necessarily evenly distributed. For example, as shown in Figure 3(a), the mandatory job set according to their E-patterns fails to be schedulable, while other mandatory job assignments such as that shown in Figure 3(b) can be well schedulable. Second, since the actual job execution time can be much less than its worst case execution time, an optional job can meet its deadline even running at a very low processor speed. This makes running other mandatory jobs at higher processor speed unnecessary and energy inefficient. Therefore it is desirable that the statically defined (m,k)-patterns be variable dynamically. The problem is, however, how to update the pattern dynamically while still guaranteeing the schedulability and (m,k)-constraints. In what follows, we present a hybrid approach to address this problem.

3 A hybrid mandatory/optional partition approach

The dual goals of (m,k)-guarantee and accommodation of dynamic variances call for an integrated static/dynamic approach to solve this problem. During the static phase, it is necessary to perform the analysis based on a prior defined specifications, such as the predefined (m,k)-patterns. It is important to select the appropriate specifications based on which the static analysis can be conducted and the guarantee criteria can be set up correspondingly. Otherwise, the static analysis may lead to poor feasibility predications and/or high computation and implementation cost.

3.1 The optimality of E-pattern

We perform the static analysis based on the mandatory job sets determined by their E-patterns. Elegant and analytical feasibility analysis formula can be derived (as shown in [17]) thanks to the regularity of E-patterns. Moreover, to further study the feasibility of mandatory jobs based on E-patterns, we first introduce the following lemma.

Lemma 3 *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 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 $[0, pT_i]$, $[pT_i, (p+1)T_i]$, ..., $[p(k_i-1)T_i, pk_iT_i]$. 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 3, we have the following important theorem.

Theorem 1 *Let* $\mathcal{T} = \{\tau_0, \tau_1, ..., \tau_{n-1}\}$, *where* $\tau_i = \{T_i, D_i, C_i, m_i, k_i\}$, *and* $k_i T_i, i = 0, 1, ..., n-1$ *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, then* $\mathcal{R}$ *is schedulable.*

Proof: Use contradiction. 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 [17], 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 3, 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 $k_i T_i$. If $k_i T_i, i = 0, 1, ..., n-1$ are co-prime, according to the General Chinese Remainder Theorem [11], all 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. □

Theorem 1 shows that the E-pattern is the optimal $(m,k)-$pattern when $k_i T_i, i = 0, 1, ..., n-1$ are co-prime. From the proof we can see that even if $k_i T_i, i = 0, 1, ..., n-1$ are slightly less than strictly co-prime, E-patterns still exhibit a relative good schedulability.

3.2 Reverse-Evenly-Distributed pattern

Using E-patterns to determine the mandatory jobs may have some problems. First, the schedulability of the mandatory job set may degrade severely if a large greatest common divisor (GCD) exists for $k_i T_i, i = 0, 1, ..., (n-1)$. As implied in Theorem 1, an extreme case would be

$$k_0 T_0 = k_1 T_1 = ... = k_{n-1} T_{n-1},$$

and $m_0 = m_1 = ... = m_{n-1}$. How to find a better (m,k)-pattern in such scenarios is beyond the scope of this paper and will be our future study. Second, according to E-patterns, the first job is always mandatory. As the successful completion of optional jobs at the lower processor

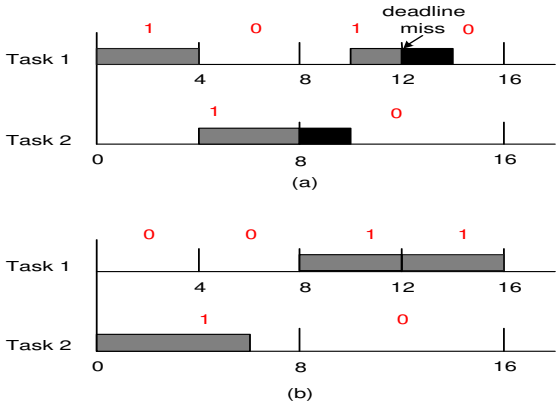

Figure 3. (a) The task set ($\tau_1 = (4,4,4,2,4)$; $\tau_2 = (8,8,6,1,2)$) is NOT schedulable with E-Pattern; (b) the same task set is schedulable with other (m,k)-pattern.

speed may potentially alleviate the necessity to run mandatory jobs at higher speed, we modified the E-pattern as follows.

$$\pi_{ij} = \begin{cases} 0 & \text{if } j = \lfloor \lceil \frac{j \times (k_i - m_i)}{k_i} \rceil \times \frac{k_i}{(k_i - m_i)} \rfloor \\ 1 & \text{otherwise} \end{cases} \qquad j = 0, 1, \cdots, k_i - 1 \tag{3}$$

The (m,k)-pattern determined by formula (3) reverses the E-pattern horizontally. We therefore call this $(m,k)-$pattern the *reverse evenly distributed pattern*, or simply E^R*-pattern*. Examples of E-pattern and its corresponding E^R-pattern are shown in Figure 1. As shown in Figure 1, the E^R-pattern maintains some of the characteristics of the E-pattern as stated in Lemma 1 (conclusion (ii) and (iii)). It also has some interesting properties that are summarized in the following lemma.

Lemma 4 *Let* $\mathcal{T} = \{\tau_0, \tau_1, ..., \tau_{n-1}\}$, *where* $\tau_i = \{T_i, D_i, C_i, m_i, k_i\}$, *and* $i = 0, 1, ..., n-1$. *Then*

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

Proof:

- The first property follows directly from the first conclusion in Lemma 1 that the number of mandatory jobs within $[0,t]$ determined by E-patterns is no less than that in any other intervals with the same length.
- To prove the second property, we use contradiction. Assume that a job from the mandatory job set according to E^R-patterns misses its deadline at t_2. Assume the starting point of the corresponding busy interval is $t_1 (t_1 < t_2)$. Then we have the total work demand of the mandatory jobs between $[t_1, t_2]$ (denoted

by $\sum_i W'(t_1,t_2)$) is greater than (t_2-t_1). Let $t=t_2-t_1$. From Lemma 1, it is ready to conclude that the mandatory work demand by E-pattern, *i.e.*, $\sum_i W(0,t)$ is no less than $\sum_i W'(t_1,t_2)$. Therefore,

$$\sum_i W(0,t) \geq \sum_i W'(0,t) > t = t_2 - t_1$$

which means that a mandatory job according to E-pattern must miss its deadline before t. This contradicts to that $\mathcal{T}$ is schedulable under E-pattern.

□

3.3 Dynamic (m,k)-pattern adjustment

The static analysis based on the predetermined (m,k)-patterns helps to ensure the feasibility of the mandatory job sets and thus guarantee the QoS levels. However, the static analysis is usually performed based on the worst case scenario and is rather pessimistic. Therefore, judiciously exploiting the irregularities and changes, inevitable in the runtime environment, dynamically can be extremely beneficial. One specific problem in this regard is that, when an optional job met its deadline, how it can help to demote other mandatory jobs to optional so that they can be executed with low processor speed or even be dropped without execution to save energy.

In our dynamic approach, the mandatory jobs are selected according to their E^R-patterns. Whenever an optional job meets its deadline, we will restart its corresponding E^R-pattern from its next job. This strategy can be illustrated with Figure 4. Figure 4(a) shows the original E^R-pattern of a task set ($\tau_1 = (2,2,1,3,7)$; $\tau_2 = (4,4,2,1,2)$) and its corresponding static schedule (the height of the rectangles represents the corresponding job speed.). Assuming that the fourth job (optional) of task τ_1 and the second job (optional) of task τ_2 met their deadlines with low speed, we will restart the E^R-patterns of the two tasks from the next job, as shown in Figure 4(b). Note that, by restarting the E^R-pattern, 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 offer more opportunities to further delay the executions of the mandatory jobs of other tasks. Significant amount of 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 the $(m,k)-$constraints can still be satisfied after restarting the E^R-pattern as stated above, we have the following lemma.

Lemma 5 *Let L be an infinite binary string by repeating the E^R-pattern, and let the ith character, i.e., $j_i = 0$. Then if we change j_i from 0 to 1, and restart the E^R-pattern from the $(i+1)$th character, the $(m,k)-$constraint is satisfied.*

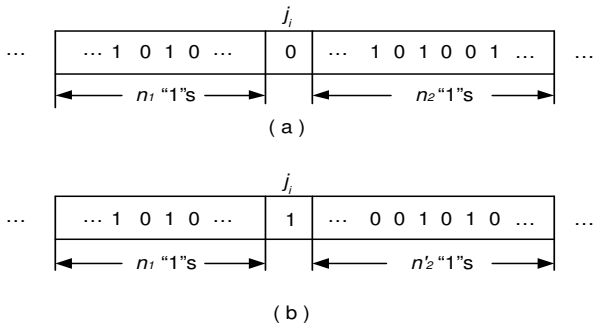

Figure 5. (a)The original mandatory jobs according to the E^R-pattern of task τ_i; (b) The mandatory jobs of τ_i after restarting the E^R pattern.

Proof: Without loss of generality, assume the original E^R-pattern is like that in Figure 5(a), and the (m,k)-pattern after the change is shown in Figure 5(b). It is not difficult to see that all 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 respectively, according to the original E^R-pattern as shown in Figure 5(a). Then we have $n_1+n_2 \geq m$. 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 4, we know that $0 \leq (n_2 - n'_2) \leq 1$. 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 that in the original window. □

4 DVS scheduling for the task set with (m,k)-constraints

After presenting our mandatory/optional partitioning strategy, we are now ready to introduce our scheduling approaches to reduce the energy consumption for systems with (m,k)-constraints.

Our new dynamic approach consists of two phases: an off-line phase followed by an on-line phase. In our approach, one processor speed is associated with each task and is determined during the off-line phase. Specifically, 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. We will then use C_i/S_j as the worst case execution time in the necessary and sufficient feasibility condition for tasks with E-patterns, as introduced in [17]. In order to minimize the energy consumption on the statically defined E-patterns, an exhaustive search approach (using branch and bound) is used to find the optimal processor speed for each task. The worst case response time for each task under its E^R-pattern is computed using the method similar to

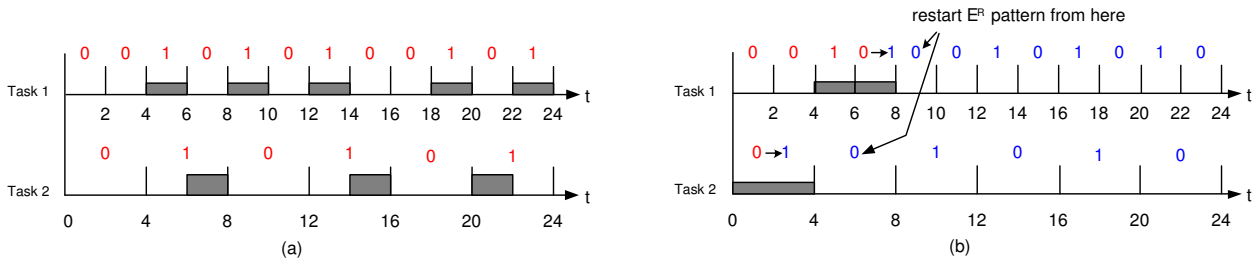

Figure 4. (a) Executing only the mandatory jobs of task set ($\tau_1 = (2,2,1,3,7)$; $\tau_2 = (4,4,2,1,2)$) according to their E^R-patterns; (b) Dynamically restarting the E^R-Pattern at $t = 8$ for τ_1 and $t = 4$ for τ_2.

that in [16] for the on-line use. Note that, according to Theorem 4, if the task set is schedulable with E-patterns, it is also schedulable with the corresponding E^R-patterns.

During the on-line phase, we adopt the dual-priority scheduling [5] scheme on the mandatory jobs. Algorithm 1 presents the salient part of our on-line scheduling algorithm.

Algorithm 1 The online phase of the dynamic approach. (Algorithm MK_{E^R})

```
1:  if HMQ is not empty then
2:      Run jobs in HMQ according to EDF;
3:  else if OPQ is not empty then
4:      J_o = jobs in OPQ;
5:      Select and run J_i ∈ J_o that have the maximum energy-
        saving potential;
6:      if J_i is finished by its deadline then
7:          Restart the E^R-pattern for task τ_i from its next job;
8:      end if
9:      if J_o = ∅ then
10:         Run jobs in LMQ;
11:     end if
12: else
13:     Run jobs in LMQ;
14: end if
```

As shown in Algorithm 1, 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, *i.e.*, $\tau_{ip} \in \tau_i$ is determined to be a mandatory or optional job based on its current E^R-pattern. The optional jobs are directly put in the OPQ, and the mandatory jobs will be first put in the LMQ, and later promoted to the HMQ after a fixed time offset, called *the promotion time* and represented as Y_i, which is computed by

$$Y_i = D_i - R_i, \tag{4}$$

where D_i is the relative deadline of τ_i and R_i is the worst case response time of τ_i which is computed during the off-line phase as stated above.

The jobs in the HMQ have the highest priority level among the three ready queues, and will be executed following the EDF scheme with the corresponding speeds determined during the off-line 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, non-preemptively by any other optional job, only when it could be finished by the earliest promotion time of the nearest mandatory job in the future. 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 execute.

It is not difficult to see that there may be more than one optional jobs available in the OPQ, and selecting which one to run may have profound impacts on future job executions. While the jobs in the OPQ can be simply run at the lowest speed without causing any dynamic failure, we use a more delicate 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 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 candidate job that has the largest energy gain ΔE_i will be chosen to be executed.

The energy efficiency of our dynamic approach lies in the fact that it adjusts the mandatory/optional partition adaptively with the run-time conditions. It is particularly efficient considering the fact that the actual execution time of a task can be much smaller than its worst case execution time. Moreover, during the executions of jobs in the HMQ, the dynamic resource reclaiming techniques such as the ones in [2, 9] can be exploited to further reduce the energy. To ensure the effectiveness and efficiency of the dynamic approach, we have the following theorem.

Theorem 2 *Algorithm 1, with complexity of $O(n)$, can en-*

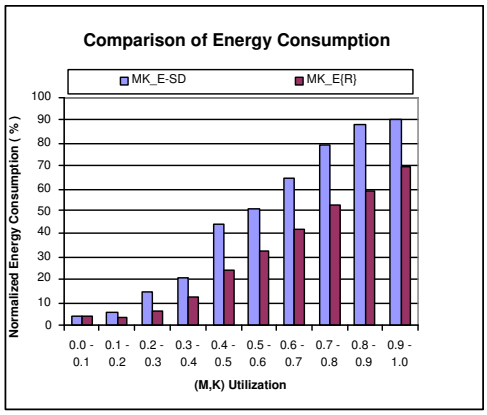

Figure 6. The average total energy consumption by MK_{E-SD} and MK_{E^R}.

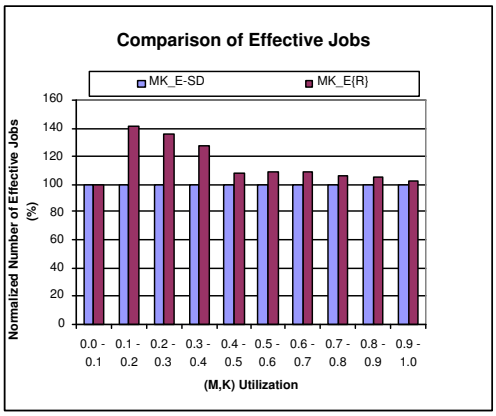

Figure 7. The average number of effective jobs by MK_{E-SD} and MK_{E^R}.

sure the (m,k)-requirements for $\mathcal{T}$ if $\mathcal{T}$ is schedulable under the E-patterns.

Proof: The details of the proof are presented in [17]. □

5 Experimental results

In this section, we compare our approach with other previous related approaches with experiments. Four different approaches are studied. In the first approach, the task sets are statically partitioned with E-patterns, and the mandatory jobs are executed with the highest processor speed. We refer this approach as (MK_E) and use its results as the reference results. The second approach (MK_{E-SD}) partitions the mandatory/optional jobs based on E-patterns first, and the processor speeds of the mandatory jobs for each task are slowed down based on the feasibility tests in [17]. The third approach (MK_R) is the dynamic approach proposed in [17], which is based on the R-patterns. The fourth approach (MK_{E^R}) is the hybrid approach presented in Section 4. The processor model used in the experiments has five discrete voltage levels with normalized speed as (0.2, 0.4, 0.6, 0.8, 1.0). We assume that the processor speed is proportional to the supply voltage and the processor power consumption is a cubic function of the processor speed.

Two separate sets of experiments were conducted. In the first set of experiments, we studied the energy-saving performance by our hybrid approach, *i.e.*, MK_{E^R}, compared with the one that uses only the static approach, *i.e.*, MK_{E-SD}. We randomly generated the periodic task sets with the periods randomly chosen in the range of $[10, 50]$ assuming the deadlines equal to their periods. The worst case execution time (WCET) of a task at the high voltage mode was set to be uniformly distributed from 1 to its deadline, and the actual execution time of a job was randomly picked from [0.4WCET, WCET]. The m_i and k_i for the (m,k)-constraints were also randomly generated such that k_i is uniformly distributed between 3 to 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., $\sum_i \frac{m_i C_i}{k_i T_i}$, into intervals of length 0.1. To reduce the statistical errors, we require that each interval contain at least 20 schedulable task sets , or at least 5000 task sets within each interval have been generated. The energy consumption for each approach was normalized to that by MK_E, and the results are shown in Figure 6(a). To evaluate the QoS that each approach can provide, we also collected the total number of jobs within $LCM(k_i T_i), i = 0, ..., n-1$, (LCM refers to *the least common multiple*) that met their deadlines (we call these jobs as the *effective jobs*) by each approach. These numbers are normalized to that by MK_E and the results are shown in Figure 6(b).

From Figure 6(a), one can immediately see that by adopting a static/dynamic hybrid approach, MK_{E^R} can achieve up to 55% energy-saving performance improvement, compared with MK_{E-SD} which uses the static approach alone. It is particularly interesting to notice that MK_{E^R} can achieve more energy savings while at the same time provide a better QoS level (up to 40% more effective jobs) than MK_{E-SD}, as shown in Figure 6(b). This is because, by running optional jobs at low processor speed and dynamically varying the (m,k)-pattern, MK_{E^R} saves the energy that is needed to run mandatory jobs at high processor speeds which are energy consuming. Therefore, more energy can be saved even though more effective jobs are executed.

We next study the (m,k)-guarantee capability of MK_{E^R} and MK_R. In this set of experiments, we randomly generated periodic task sets such that within each (m,k) utilization interval no less than 100 task sets were schedulable by MK_{E^R} or at least 5000 different task sets have been generated for each interval. The results are listed in Table 1. As shown in Table 1, when the (m,k)-utilization is low (less than 0.3), the (m,k)-guarantee capability between MK_{E^R} and MK_R are very close. However, when the (m,k)-

utilization is relatively high (larger than 0.3), MK_{E^R} exhibits a much stronger (m,k)-guarantee capability than MK_R. For example, when the (m,k)-utilization is over 0.8, the (m,k)-requirements and thus the QoS levels for more than 50% of the task sets that can be guaranteed with MK_{E^R} cannot be ensured with MK_R.

Overall, the experimental results show that our hybrid approach based on the E^R-pattern helps to significantly improve not only the energy saving performance and QoS levels of a scheduler, but also the range of the real-time systems with (m,k)-firm guarantee.

(m,k)	**Feasible Task Sets**	
Util	MK_{E^R}	MK_R
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

Table 1. The average numbers of feasible task sets by MK_{E^R} and MK_R

6 Conclusions

Energy consumption and QoS guarantee are two of the most critical factors for the successful design of pervasive real-time computing platforms. In this paper, we presented a hybrid DVS approach to reduce the energy consumption while guaranteeing the QoS requirement in terms of (m,k)-constraints. Our approach ensures the (m,k)-firm guarantee by conducting static analysis based on the evenly distributed (m,k)-pattern instead of the deeply-red pattern as suggested in the previous work. To accommodate the dynamic nature of run-time environment, a dynamic strategy is proposed to vary the (m,k)-pattern and a run-time scheduler is constructed based on the dual-priority scheme to dynamically determine if a job should be mandatory or optional as well as its corresponding processor speed. As shown in our experiments, with excellent adaptivity to the run-time conditions, the proposed approach outperforms previous research significantly in terms of energy savings, QoS levels, as well as the range of real-systems with (m,k)-firm guarantee.

References

[1] A.K.Mok and W.Wang. Window-constraint real-time periodic task schedulling. *RTSS*, 2001.

[2] H. Aydin, R. Melhem, D. Mosse, and P. Alvarez. Determining optimal processor speeds for periodic real-time tasks with different power characteristics. In *ECRTS01*, June 2001.

[3] G. Bernat and A. Burns. Combining (n,m)-hard deadlines and dual priority scheduling. In *RTSS*, Dec 1997.

[4] G. Bernat and R. Cayssials. Guarantted on-line weakly-hard real-time systems. In *RTSS*, 2001.

[5] R. Davis and A. Wellings. Dual priority scheduling. In *RTSS*, pages 100–109, 1995.

[6] M. Hamdaoui and P. Ramanathan. A dynamic priority assignment technique for streams with (m,k)-firm deadlines. *IEEE Transactions on Computes*, 44:1443–1451, Dec 1995.

[7] S. Hua and G. Qu. Energy-efficient dual-voltage soft real-time system with (m,k)-firm deadline guarantee. In *CASE'04*, 2004.

[8] S. Hua, G. Qu, and S. Bhattacharyya. Energy reduction techniques for multimedia applications with tolerance to deadline misses. *DAC*, pages 131–136, 2003.

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

[10] G. Koren and D. Shasha. Skip-over: Algorithms and complexity for overloaded systems that allow skips. In *RTSS*, 1995.

[11] J. Y.-T. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic,real-time tasks. *Performance Evaluation*, 2:237–250, 1982.

[12] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 17(2):46–61, 1973.

[13] J. Liu. *Real-Time Systems*. Prentice Hall, NJ, 2000.

[14] J. R. Lorch and A. J. Smith. Operating system modifications for task-based speed and voltage scheduling. In *MOBISYS 2003*, pages 215–229, 2003.

[15] T. Ma and K. Shin. A user-customizable energyadaptive combined static/dynamic scheduler for mobile applications. *RTSS*, pages 227–236, 2000.

[16] M.Spuri. Analysis of deadline scheduled real-time systems. In *Rapport de Recherche RR-2772, INRIA, France*, 1996.

[17] L. Niu and G. Quan. Energy-aware scheduling for real-time systems with (m,k)-guarantee. In *Technical Report TR-2005-005, Department of Computer Science and Engineering, University of South Carolina*, 2005.

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

[19] G. Quan and X. Hu. Enhanced fixed-priority scheduling with (m,k)-firm guarantee. In *RTSS*, pages 79–88, 2000.

[20] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek. A resource allocation model for qos management. In *RTSS*, Dec. 1997.

[21] P. Ramanathan. Overload management in real-time control applications using (m,k)-firm guarantee. *IEEE Trans. on Paral. and Dist. Sys.*, 10(6):549–559, Jun 1999.

[22] K. F. S. Reinhardt and T. Mudge. Automatic performance-setting for dynamic voltage scaling. *MOBICOM'01*, 2001.

[23] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced cpu energy. In *OSDI*, pages 13–23, 1994.

[24] R. West and K. Schwan. Dynamic window-constrained scheduling for multimedia applications. In *ICMCS*, 1999.

[25] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *AFCS*, pages 374–382, 1995.

[26] W. Yuan and K. Nahrstedt. Energy-efficient soft real-time cpu scheduing for mobile multimedia systems. In *SOSP*, 2003.