Power aware scheduling for real-time systems with (m, k) guarantee

Abstract

Energy consumption and Quality of Service (QoS) are two issues of primary concern for developers of today's pervasive computing systems. In this paper, we study the problem of minimizing the energy consumption while ensuring the designated QoS requirements for a real-time pervasive computing system on a two-level variable voltage processor. In our approach, the QoS requirements are quantified with so called (m, k)-constraints, requiring that at least m number of any k consecutive jobs of a task meet their deadlines. In our approach, the "redundant" jobs are discarded according to the (m, k) constraints. We develop a necessary and sufficient condition to check the schedulability for the remaining jobs, so that some of the remaining tasks can be executed at the low voltage mode to save the energy while still meeting the (m, k) constraints. We have conducted extensive experiments, both on synthesized systems and real applications, to demonstrate the effectiveness of our approach.

1 Introduction

Power aware computing has come to be recognized as a critical enabling technology in the design of real-time embedded systems for use in the pervasive computing applications. Today's embedded systems continue to evolve towards higher levels of performance requirements to provide rich features for a variety of dedicated and extensible applications. Such high performance requirements are inevitably accompanied by the high level of energy consumption required to run the pervasive computing platform—generally consisting of an embedded processor core, a real-time operating system, and a number of VLSI custom logic modules implementing specialized functions directly in hardware. On the other hand, designers have realized that these pervasive, mobile and battery-operated embedded systems require low power consumption in order to extend the battery life.

If a system is to be both energy efficient and have high performance available on-demand, it has to be a *power aware* system—a system with high scalability, adaptivity, and energy efficiency, allowing the designers, or even the users, to adjust the expected Quality of Service (QoS) dynamically and, in the mean time, optimize the energy consumption accordingly.

In recent years, there has been increasing interest in applying scheduling techniques to conserve energy in real-time computing applications. The prevalent approach to energy conservation—particularly common with laptop computers and cell phones— is to shut down the unit when it is idle. To achieve better energy saving performance, however, many approaches adopt the so-called *dynamic voltage scaling* (DVS) technique. DVS saves on the energy consumption by dynamically changing the processor supply voltage levels —a characteristic supported in many modern processors such as Intel's XScale [9], Transmeta's Crusoe [14], and AMD's Duron [2] processors. Specifically, a variable voltage processor is a CMOS circuit, so the dynamic power— the dominant component in the processor power consumption—can be

represented [5] by

$$P \propto \alpha C_L V^2 f, \tag{1}$$

where α is the switching activity, C_L is the load capacitance, V is the supply voltage, and f is the system clock frequency. Due to the quadratic relationship between the voltage and power consumption, reducing voltage can significantly save the power consumption for the processor. On the other hand, however, reducing the voltage supply increases the circuit delay, and thus the processor speed (s), which is given by

$$s \propto \frac{(V - V_T)^2}{V}, \tag{2}$$

where V_T is the threshold voltage. Care must be taken when reducing the voltage level since it potentially will cause the tasks in the real-time computing systems to miss their deadlines.

Many DVS techniques, e.g., [3, 10, 8, 11, 12, 23, 26, 30, 33], have been proposed to reduce the energy consumption for real-time computing system. They differ from the implementation strategies and real-time system characteristics, such as static *vs.* dynamic, fixed priority *vs.* dynamic priority assignment, periodic *vs.* non-periodic systems, preemptive *vs.* nonpreemptive effects, and the like. One common feature of these techniques is that they are targeted for the system requiring that all the task instances must meet their deadlines. While tremendous energy can be saved for a wide range of applications using these techniques, they become inefficient or inadequate when QoS requirements are imposed on the embedded system, in which some tasks are allowed to miss their deadlines or even simply be dropped without execution.

QoS requirements dictate under what conditions the system will provide its service to competing or cooperating tasks executed on the embedded processor, and at what quality level relative to the completion deadline requirements for the tasks. Typically, we think of QoS metrics such as response delay, sample loss rate, deadline miss rate, among others.

Recently, we have seen research on using the DVS technique for real-time systems also having QoS constraints. One approach, e.g. [31, 22, 18, 28], is to dynamically set the processor speed based on both the prediction of future workload as well as the QoS requirements. While this seems intuitive and easy to implement, we see that the effectiveness of this combined scheduling technique greatly depends on the precision of the prediction process. As system workload is usually volatile and difficult to predict, the energy-saving performance of these combined scheduling approaches can be seriously limited [6]. Qiu *et. al.* introduces another technique [24] based on using *generalized stochastic petri nets* to capture the probabilistic nature and complex behaviors of real-time embedded systems, which is later on transformed to a *continuous time Markov decision* model. In their approach, optimizing the energy consumption, while also incorporating QoS constraints (such as delay, jitter, and loss rate) over the Markov model, the energy consumption is minimized and the QoS requirements are satisfied. This approach seems suitable for systems where timing specifications can only be *statistically* determined. However, it cannot be readily applied to systems that have deterministic timing specifications–such as tasks' release times, inter arrival periods, deadlines, and worst case execution times– usually the case in many real-time embedded systems.

All of the above QoS related power-aware scheduling techniques have been focused on formulating the control problem using the *soft* real-time system approach–namely, a task should be finished by its designated deadline, but execution of a task after the deadline still contribute to the overall system performance. Application systems such as real-time databases, Internet servers, among others, can be classified as soft real-time systems. In considering QoS requirements, there is another common class systems, referred to as *firm-deadline* systems. In a firm-deadline system, a task can occasionally miss its deadline, but too many missed deadlines over *consecutive* processing cycles would not satisfy the

application service requirements, or those imposed by the environment. Examples of such systems include avionics navigational systems, aircraft engine control systems, and streaming audio/video applications.

In this paper, we are interested in the problem of minimizing energy consumption for a firm-deadline task system, containing a set of periodic real-time tasks scheduled by an earliest deadline first (EDF) policy [17] executing on a variable voltage processor. In addition to the deterministic timing specifications (such as task's period, deadline, and worst case execution time), an $(m, k)(0 < m \leq k)$ constraint [7] is associated with a task of the system requiring that m out of any k consecutive job instances of the task must meet their deadlines. Since not all executing jobs are required to meet specific deadlines (for example, a periodic sampling process to acquire new data), some "redundant" jobs can simply be discarded to save on the energy expenditure. In this paper, the "redundant" jobs are discarded according to techniques discussed in the literature [27]—although the use of this technique was originally applied to the problem of dealing with cases of task overloading in real-time systems.

We argue that this techniques can well serve our dual goals of improving the energy-saving performance, in addition to providing a guarantee of QoS formulated as (m, k)-constraints. For the remaining jobs, we propose a technique that is necessary and sufficient for their schedulability. Based on this capability, some of the remaining jobs not deemed "redundant" are allowed to be executed at a low voltage operating level to save on energy consumption while executing, so as to still meet their task completion deadlines. Moreover, this technique can be readily incorporated with known dynamic scheduling techniques, such as [3, 12] to further improve the energy efficiency. The significance of our approach is that, to the best of our knowledge, this is the first approach that can efficiently save on energy consumption while also providing a *deterministic* QoS guarantee for a real-time system.

This paper is organized as follows. Section 2 formally introduces our problem. Section 3 discusses how to guarantee the QoS requirements in terms of (m, k)-constraints. Section 4 presents the necessary and sufficient condition to predict the system's schedulability after the redundant jobs are pruned. We then introduce how this necessary and sufficient condition can help to adjust the processor speed dynamically to save the energy. The effectiveness and energy efficiency of our approach are demonstrated using simulation results in Section 6. In section 7, we offer conclusions as well as several future directions for this paper.

2 System models

The real-time system considered in this paper contains n independent periodic tasks, $\mathcal{T} = \{\tau_0, \tau_1, \cdots, \tau_{n-1}\}$, and each task contains an infinite sequence of periodically arrived instances, each of which called a *job*. Task τ_i is characterized using five parameters, i.e., $(T_i, D_i, C_i, m_i, k_i)$. T_i, D_i, and C_i represent the period, the deadline for each job instance, 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 consecutive k_i jobs of τ_i, at least m_i must meet their prescribed completion deadlines.

The variable voltage processors used in our approach can run in one of two modes: *high voltage mode* and *low voltage mode*. In high voltage mode, the processor requires a high supply voltage (V_H), running at a faster clock rate. In low voltage mode, the voltage (V_L) supplied to the processor circuitry is lower and, thus, the speed at which the processor operates is slower. Commercial processors such as the ARM7D [19] and Motorola PowerPC 860 [20] have this type of operating characteristic relative to voltage scaling. Since $V \gg V_T$ is usually true in most cases, from equation (2), we assume the ratio $s \propto \frac{1}{V}$ is valid, and use this in our formulation. We also assume that C_i is the worst case execution time for task τ_i in high voltage mode. Therefore, if $\alpha = \frac{V_H}{V_L}$, the worst case execution time for τ_i is αC_i if τ_i is executed in low voltage mode.

Given the above abstraction and set of assumptions, we formulate our model as follows:

Problem 1 *Given a real-time 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 the real-time system with EDF on a variable voltage processor, which has two discrete voltage levels* V_L *and* V_H, *such that all* (m, k)*-constraints for the system are guaranteed and the energy consumption is minimized.*

Since not all jobs scheduled are required to meet their deadlines due to (m,k)-constraints, there are opportunities to save energy by running as many jobs as possible at low voltage, such that just enough jobs meet their deadlines. The question, however, is how to minimize energy consumption while also providing a guarantee that no less than m_i of *any* k_i consecutive jobs from task τ_i meet their deadlines. We explore the theory and formulate an answer to this question in the following sections.

3 Meeting the (m,k)-constraints

There is much work, such as [4, 7, 32, 27, 25], that has investigated real-time scheduling with (m, k)-constraints since the time the problem was first introduced in [7]. Hamdaoui and Ramanathan [7] propose a dynamic assignment of a higher priority to tasks that are proximally "closer" to violating the (m, k)-constraints than others assigned lower priorities. In [4], Bernat and Burns apply the so called dual priority scheduling scheme, where a given task may be promoted to a higher priority level if there is high likelihood that it may not meet the (m, k)-constraint otherwise. In [32], West and Poellabauer have determined individual task priorities based on a combination of EDF and (m,k)-constraints. In this approach, the priority of a job is first determined by its deadline–the sooner the deadline, the higher the priority. For jobs having the same deadline, those with "tighter" $(m, k)-$constraints will be assigned higher priorities.

While these best-effort approaches help improve the opportunity for tasks to meet their (m, k) constraints, they cannot ensure (m, k)-firm guarantee, in the sense that the scheduling techniques cannot predict whether the (m, k)-constraints for the system can be met. Also, note that these approaches cannot avoid the execution of jobs that will miss their deadlines. Execution of such jobs consume precious system resources while adding little benefit to overall performance of the embedded system platform.

Ramanathan [27] proposes another approach to the scheduling problem under (m, k)-constraints, which is of particular interest to us. In this work, the recurring jobs associated with a task are statically partitioned to be either *mandatory* or *optional*. Specifically, the job τ_{ij}, *i.e.* the jth job of task τ_i, is determined to be mandatory if

$$j = \lfloor \lceil j \times \frac{m_i}{k_i} \rceil \times \frac{k_i}{m_i} \rfloor, \quad j = 0, 1, 2, ..., \tag{3}$$

and it is optional otherwise. For example, let τ_i have (m,k) constraint as (3,7). Then τ_{i2} is mandatory since $2 = \lfloor \lceil 2 \times \frac{3}{7} \rceil \times \frac{7}{3} \rfloor$, and τ_{i3} is optional since $3 \neq \lfloor \lceil 3 \times \frac{3}{7} \rceil \times \frac{7}{3} \rfloor$. It has been shown [27] that as long as all the mandatory jobs can meet their deadlines, the (m, k)-constraints are satisfied.

While this partitioning is initially intended for improving the stability performance under overloading situation for real-time control, we adopt it in our approach because of its ability to guarantee (m, k)-constraints, as well as for its potential to save energy. We summarize our observations 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 (3). Then (i) for any* k_i *consecutive jobs of* τ_i, *there are exactly* m_i *mandatory jobs; (ii) within any* $l_i (l_i > 0)$ *consecutive jobs of* τ_i, *the difference of the numbers of mandatory jobs is no more than 1.*

Proof: Conclusion (i) can be readily derived from Lemma 2 in [27] and therefore is omitted. We present the proof for conclusion (ii) as follows.

Let $N_i(x_i, y_i)$ be 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 ($a_i \neq b_i$), we have the number of mandatory jobs are $N_i(a_i, a_i + p_i - 1)$ and $N_i(a_i, a_i + p_i - 1)$, respectively. When the mandatory jobs are determined according to equation (3), it has been shown [27] that, for the first q_i jobs of τ_i, there are $l_i(q_i) = \lceil \frac{m_i}{k_i} q_i \rceil$ jobs that are mandatory. Therefore,

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

and similarly,

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

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

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

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 $x_1, x_2 \in R$, we have

$$|N_i(a_i, a_i + p_i - 1) - N_i(b_i, b_i + p_i - 1)| \leq \lceil (p_i - 1)\frac{m_i}{k_i} \rceil - \lfloor (p_i - 1)\frac{m_i}{k_i} \rfloor \leq 1.$$

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, by adopting the partitioning strategy as shown in (3), a minimal set of mandatory jobs are determined, no more and no less. Removing any job from this set will violate the (m, k)-constraints, and any other jobs added to this set will be redundant in terms of (m, k)-guarantee. Therefore, the energy is only consumed for the necessary jobs, not for jobs that are nonessential for the (m, k)-constraints. Moreover, since the power consumption is a convex function of the processor speed, drastic variation of the processor speed tends to increase the energy consumption [33] sharply. According to Lemma 1, formula (3) helps to spread out the required jobs *evenly* along the time to avoid the "burst" workload for the processor and, thus, has great energy-saving potential.

After getting the mandatory job sets according to equation (3), the problem then becomes how to schedule the mandatory jobs on the two-mode processor to save energy. Recall that any mandatory job missing its deadline will cause the violation of (m, k)-constraint. To ensure each mandatory job meet its deadline, an accurate schedulability analysis is crucial since inaccurate prediction may either cause the violation of the system requirements, or lead to a pessimistic design which can seriously degrade the energy performance.

4 Schedulability analysis for the mandatory job set

In this section, we investigate the schedulability condition for the mandatory job set determined using equation (3). There are two benefits to conduct the schedulability analysis. First, it helps to predict if a real-time system with (m, k)-constraints using the static partitioning technique as shown in equation (3) is schedulable or not. Second, it also helps to put the execution of some mandatory jobs in the low voltage mode to save the energy while meeting their deadlines, and hence, the (m, k)-constraints.

Koren *et. al.* [13] proposed a schedulability analysis technique based on the "skip-over" model, *i.e.*, a model that allows one job be "skipped" after every s jobs. Note that the "skip-over" model is only

a special case of (m,k) model with $m = k-1$, the schedulability technique for this model cannot be easily extended for a general (m,k) model. Since the mandatory jobs are scheduled according to EDF, it is desired to use the well-known Liu&Layland bound ($U \leq 1$) to predict the schedulability. However, $U \leq 1$ is only the necessary and sufficient condition for task set with tasks' deadlines equal their periods, or it is only a necessary condition. Using a network traffic model with periodic traffic arrivals, Zheng *et. al.* [34] proposed a necessary and sufficient EDF-schedulability condition with arbitrary deadlines. Their work is later extended by Liebeherr *et. al.* [16] for a more general case. For a preemptive real-time task system with arbitrary job arrivals and deadlines, their conditions can be formulated as follows:

Theorem 1 *System $\mathcal{T} = \{\tau_0, \tau_1, ..., \tau_{n-1}\}$ is EDF-schedulable* iff

$$\forall t > 0, \sum_i W_i(0,t) \leq t, \tag{4}$$

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

The close-form necessary and sufficient condition in Theorem 1 cannot be readily applied to check the feasibility for a mandatory job set, since it requires checking an infinite number of points ($t > 0$) to guarantee the schedulability. It is highly desirable that we should only be required to check a limited and small number of points to determine if a mandatory job set is schedulable or not. In what follows, we introduce another necessary and sufficient condition to precisely capture the schedulability information of the mandatory job set. Before we introduce this theorem, we first present the following definition.

Definition 1 *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 $w(t_s^-) = w(t_e^+) = 0$ and $w(t) > 0$ for $t_s \leq t < t_e$.*

The following theorem allows one to predict the schedulability for a mandatory job set by checking only a limited number of points.

Theorem 2 *Let system $\mathcal{T} = \{\tau_0, \tau_1, ..., \tau_{n-1}\}$, where $\tau_i = \{T_i, D_i, C_i, m_i, k_i\}$, and $\mathcal{R}$ be the mandatory job set generated from $\mathcal{T}$ according to equation (3). $\mathcal{R}$ is schedulable with EDF* iff *all the mandatory jobs within the first busy interval can meet their deadlines.*

Proof: The necessity of this statement follows naturally with the definition of the schedulability. To prove the sufficiency of this statement, we use the contradiction.

We use the following notation, *i.e.* $\lceil x \rceil^+$ in our proof. Specifically,

$$\lceil x \rceil^+ = \begin{cases} n & n-1 \leq x < n, n \in Z \\ 0 & x < 0 \end{cases} \tag{5}$$

Suppose the first deadline miss happens 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. It has been shown [27] that, if the mandatory jobs are determined according to equation (3), for the first p_i jobs of τ_i, there are $l_i(t) = \lceil \frac{m_i}{k_i} p_i \rceil$ jobs that are mandatory. Therefore, the total workload in interval $[t', t]$ can be formulated as follows:

$$W(t',t) = \sum_i (\lceil \frac{m_i}{k_i} \lceil \frac{t - D_i}{T_i} \rceil^+ \rceil - \lceil \frac{m_i}{k_i} \lceil \frac{t' - D_i}{T_i} \rceil^+ \rceil) C_i. \tag{6}$$

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

$$W(t', t) \leq \sum_i \lceil \frac{m_i}{k_i} \lceil \frac{t - t' - D_i}{T_i} \rceil^+ \rceil C_i. \tag{7}$$

Since the deadline miss happens at t, we have

$$\sum_i \lceil \frac{m_i}{k_i} \lceil \frac{t - t' - D_i}{T_i} \rceil^+ \rceil C_i > t - t'. \tag{8}$$

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

$$W(0, t - t') = \sum_i \lceil \frac{m_i}{k_i} \lceil \frac{t - t' - D_i}{T_i} \rceil^+ \rceil C_i \leq t - t'. \tag{9}$$

This contradicts to (8). □

The ending point of the first busy interval can be easily found by observing that it must be the smallest t such that the accumulated workload within interval $[0, t]$ equals t, that is,

$$\sum_i W_i(0, t) = t. \tag{10}$$

One can easily compute the ending point for the first busy interval by using fixed-point iteration on formula (10) with the initial value t set to $\min_i D_i, i = 0, \cdots, (n-1)$. The fixed-point iteration will converge rapidly as long as the task set is schedulable. After getting the ending point for the first busy interval, we can then apply Theorem 2 and check if a mandatory job set is schedulable.

5 DVS scheduling for the mandatory job set

After a mandatory job set is determined to be schedulable according to Theorem 2, one can use DVS techniques to schedule the mandatory jobs and save energy. In regard to the two-mode nature of the variable voltage processor and the characteristics of the mandatory jobs, one intuitive approach is to use the shut-down-when-idle strategy. That is, all the mandatory jobs are executed at the high level voltage, the processor is shut down if all the previously arrived jobs have been finished and new jobs have not arrived. While this approach is intuitive and easy to implement, it does not take the advantage of the DVS and, thus, the energy performance is severely limited.

Recall that the processor power consumption is a quadratic function of the voltage. Reducing the supply voltage of the processor can significantly reduce the power consumption. Therefore, to better save the energy, we can assign some tasks to the low voltage mode as long as the resultant mandatory jobs are schedulable. However, when assigned to the low voltage mode, a job may take the risk of missing its deadline or cause other jobs to miss their deadlines. Our goal is to assign as many tasks as possible in the low voltage mode to save the energy as long as all the mandatory jobs can meet their deadline. We can formulate the problem as follows.

Problem 2 *Given a real-time system* $\mathcal{T} = \{\tau_0, \tau_1, \cdots, \tau_{n-1}\}$ *with* $\tau_i = (T_i, D_i, \alpha_i C_i, m_i, k_i)$, *and* $\alpha_i = 1$ *or* α. *Select* $\{\alpha_i, i = 0, \cdots, (n-1)\}$ *such that all the* (m, k)*-constraints for* $\mathcal{T}$ *can be satisfied and the energy consumption is minimized.*

Note that, for a given set of $\{\alpha_i, i = 0, \cdots, (n-1)\}$, we can use Theorem 2 to analyze its schedulability. As this static analysis can be made off-line, we can afford of using some classic exhaustive search

algorithms such as branch-and-bound to find the optimal solution, when the number of tasks n of the task set is not extremely large (which is common for many real applications).

Note that, after we assign the voltage levels for each task, we can further improve the energy performance by incorporating the dynamic techniques such as the ones in [3, 15, 12] to exploit the dynamic slack time, *i.e.* the slack time due to the task's actual execution being usually smaller than its worst case value. For example, when a job finishes earlier than its estimated completion time, the next job from the ready queue can first run in the low voltage mode until the slack time is completely consumed, and then run in its designated voltage mode. More energy is saved in this way since a greater number of the mandatory jobs can be executed (either partially or completely) at this low voltage level. In next section, we use experiments to evaluate the energy performance of our approach.

6 Experimental Results

In this section, we use experiments to demonstrate the effectiveness of using our approach to schedule the real-time systems while guaranteeing the QoS requirements in terms of (m, k)-constraints.

A fair evaluation would need to compare our approach with any other ones that can deal with the energy saving and guarantee to satisfy the (m, k)-constraints *deterministically*. However, as explained before, no other previous work from our best knowledge has been reported in the literature. Therefore, in our experiments, we compare energy saving performance by our approach with two other previous approaches. One approach uses the shut-down-when-idle strategy, and the other one uses the dynamic slack reclaiming technique [15], which is also incorporated in our approach to reclaim the dynamic slacks after some of the tasks are put in the low voltage mode. For brevity, we use MK_{LP}, MK_{SD}, and MK_{DYN} to represent our approach, the one with shut-down-when-idle, and the one using dynamic slack reclaiming technique, respectively.

Our experiments contain two sets of test cases. The first set contains randomly generated real-time task sets and uses an ideal two-mode variable processor. For the second set of the tests, we use the practical applications reported in the literature and adopt the hardware specifications of a real two-mode processor, *i.e.* Motorola PowerPC 860 [20].

For the randomly generated task sets, the periods are randomly selected from interval 10,50 and the deadlines are assumed to be equal to their periods. The worst case execution time (WCET) of a task at the high voltage mode is uniformly distributed from 1 to its deadline, and the actual execution time of a job is randomly picked from 0.5WCET, WCET. The m_i and k_i for the (m, k)-constraints are also randomly generated such that k_i is uniformly distributed between 2 to 12, and $m_i < k_i$. In our experiment, we investigate the energy saving performance by different approaches under different task utilizations, since different task utilizations can result in significant differences for energy savings. Specifically, we assign the task sets into 10 groups according to their (m, k) *utilization*, *i.e.* $U_m = \sum_i \frac{m_i C_i}{k_i T_i}$. For example, a task set with $0 < U_m \leq 0.1$ is assigned to the first group. To reduce the statistical errors, we require that each group contain at least 20 schedulable task sets, or until at least 5000 task sets with the corresponding (m, k) utilization have been generated. For the processor used for the randomly generated task sets, we assume it is a two-mode variable voltage processor with $\alpha = \frac{V_H}{V_L} = 2$. The average energy consumption for each approach in each group is collected. All results are normalized against the ones by MK_{SD} and filled into Table 1.

Table 1 shows the average energy consumptions by the three approaches, *i.e.* MK_{SD}, MK_{DYN}, and MK_{LP}, as well as the energy saving improvements of MK_{LP} over MK_{SD} and MK_{DYN}. From Table 1, one can readily see that our approach have much better energy saving performance than the other two. Compared with the simply shut-down-when-idle strategy, our approach can save the average energy consumption up to around 56%. This is because that, our approach can assign some mandatory jobs that

(m,k)	**Average Energy Consumption**			**Improvement by MK_LP**	
Utilization	MK_{SD}	MK_{DYN}	MK_{LP}	MK_{SD}	MK_{DYN}
0.0-0.1	100	98.2	44.9	55.1%	53.3%
0.1-0.2	100	98.1	43.4	56.6%	55.7%
0.2-0.3	100	96.7	53.4	46.6%	44.8%
0.3-0.4	100	92.8	52.2	47.8%	43.8%
0.4-0.5	100	91.9	64.3	35.7%	30.0%
0.5-0.6	100	89.7	65.8	34.2%	26.6%
0.6-0.7	100	85.2	75.4	24.6%	11.5%
0.7-0.8	100	89.8	89.0	11.0%	0.89%
0.8-0.9	100	83.3	83.3	16.7%	0.00%
0.9-1.0	100	80.2	80.2	19.8%	0.00%

Table 1. Average energy consumption randomly generated task sets and the corresponding energy saving improvement of MK_{LP} over MK_{SD} and MK_{DYN}.

initially need to be executed in the high voltage mode to the low voltage mode and still guarantee their deadlines, and hence, the $(m,k)-$requirements. The more mandatory jobs can be assigned to the low voltage mode, the higher the energy saving improvement. The energy saving certainly also comes from the factor that, by incorporating the dynamic reclaiming technique in our approach, more mandatory jobs (either partially or completely) can be executed in the low voltage mode.

To investigate the impacts of the dynamic slack reclaim technique relative to our capability of statically assigning the mandatory jobs in the low voltage mode based on our accurate schedulability analysis, we only need to compare the energy saving results of MK_{DYN} and MK_{LP} as shown in Table 1. Compared with the dynamic slack reclaiming, our proposed technique has significant energy saving improvement when the system utilization is relative small. As the system utilization increases, the energy saving improvement is decreasing. For example, from Table 1, the energy saving improvement can be as high as around 56% when the system (m,k)-utilization is among 0.1 to 0.2, while the improvement becomes around 11% for system with (m,k)-utilization among 0.6 to 0.7. The reason is that, when the utilization is lower, the probability for statically assign the execution of mandatory jobs is higher. Therefore, our approach can have a much higher energy saving improvement than using the dynamic slack reclaiming technique alone. If the system utilization is very high, statically assigning a mandatory job to the low voltage mode will potentially cause it or other mandatory jobs to miss the deadlines, and thus the energy improvement is not as significant as that for the systems with low utilizations.

We also test our conclusions in a more practical environment. In the second set of experiments, we use three real-world applications: web phone [29], CNC (Computerized Numerical Control) machine controller [21], and INS (Inertial Navigation System) [1], as our task systems. For the (m,k)-constraints, we still resort to the extensive randomly generated samples as we do for the randomly generated task sets to provide us with some meaningful insights. The processor used in this set of experiments is adopted from the product specifications of a practical two-mode variable voltage processor, *i.e.* Motorola PowerPC 860. Specifically, it can be run at 50Mhz on 3.3 V, or 25Mhz at 2.4V. 20 sets of $(m,k)-$constraints are randomly generated for each applications. The average energy consumptions for the applications are listed in Table 2, after normalized against the ones by MK_{SD}.

From Table 2, we can see that our approach can lead to much less energy consumption, *i.e.* 13%

	Average Energy Consumption			Improvement by MK_LP	
Systems	MK_{SD}	MK_{DYN}	MK_{LP}	MK_{SD}	MK_{DYN}
web phone	100	99.1	90.1	19.9%	9.08%
CNC	100	92.1	71.0	29%	22.9%
INS	100	96.0	87.0	13%	9.38%

Table 2. Average energy consumptions for three real applications (web phone [29], CNC [21], and INS [1]) using the practical two-mode processor (Motorola PowerPC 860 [20]), as well as the corresponding energy saving improvement of MK_{LP} over MK_{SD} and MK_{DYN}.

to 29%, compared with the power down strategy. Compared with that by using the dynamic slack reclaiming technique alone, our approach can save more energy, *i.e.* from 9.08% to 22.9%. Overall, the experimental results based on both the randomly generated systems as well as the practical applications have clearly demonstrate the effectiveness of our approach in saving energy.

7 Conclusions and future work

Low power/energy consumption, along with QoS guarantee, are two of the most critical factors for the successful design of pervasive real-time computing platforms with the objectives as discussed in this paper. We have proposed in this paper the use of a power-aware scheduling technique that can be used in systems with a two-mode variable voltage processor. We use this feature to reduce energy consumption while also ensuring QoS requirements deterministically. The QoS requirements are quantified by a set of (m, k)-constraints. Jobs identified as redundant are discarded statically. A necessary and sufficient condition of feasibility is introduced that enables remaining tasks to be executed in low voltage mode for accrued energy savings. Finally, our technique can be readily incorporated with some existing dynamic scheduling techniques, which can lead to further enery savings. Experiments with parameters drawn from both synthesized systems as well as practical applications has shown the effectiveness of our approach.

There are several possible extensions of this work. In this paper, the mandatory jobs are statically determined, solely according to their (m, k)-constraints. It would be interesting to find some dynamic ways to decide if a job should be executed or not. However, to guarantee the schedulability for the dynamic determined mandatory jobs, and therefore the $(m, k)-$constraints, can be challenging. Also, this approach targets only on two-mode variable voltage processor. It would be interesting to extend further for the multiple level voltage processor.

References

[1] A.Burns, K. Tindell, and A. Wellings. Effective analysis for engineering real-time fixed priority schedulers. *IEEE Transactions on Software Engineering*, 21:920–934, May 1995.

[2] AMD. Mobile amd duron processor, 2001.

[3] 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.

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

[5] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen. Low-power cmos digital design. *IEEE Journal of Solid-State Circuits*, 27(4):473–484, April 1992.

[6] D. Grunwald, P. Levis, K. I. Farkas, C. B. Morrey III, and M. Neufeld. Policies for dynamic clock scheduling. *Proceedings of OSDI*, pages 73–86, 2000.

[7] 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.
[8] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. Srivastava. Power optimization of variable voltage core-based systems. In *DAC*, pages 176–181, 1998.
[9] Intel. Developer manual: Intel 80200 processor based on intel xscale microarchitecture, 2002.
[10] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *ISLPED*, pages 197–202, August 1998.
[11] R. Jejurkar and R. Gupta. Energy aware task scheduling with task synchronization for embedded real time systems. *CASES*, 2002.
[12] 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.
[13] G. Koren and D. Shasha. Skip-over: Algorithms and complexity for overloaded systems that allow skips. In *RTSS*, 1995.
[14] D. Laird. Crusoe processor products and technology, January 2000.
[15] Y. Lee, Y. Doh, and C. Krishna. Edf scheduling using two-mode voltage-clock-scaling for hard real-time systems. *CASE*, pages 221–228, 2001.
[16] J. Liebeherr, D.W.Wrege, and D. Ferrari. Exact admission control for networks with a bounded delay service. *IEEE/ACM Transactions on Networking*, 4:885–901, 1996.
[17] 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.
[18] J. R. Lorch and A. J. Smith. Improving dynamic voltage scaling algorithms with PACE. In *SIGMETRICS/Performance*, pages 50–61, 2001.
[19] A. R. M. Ltd. Introduction to thumb. version 2.0. *ARM DVI-0001A*, 1995.
[20] MPC860. Mpc860 powerpc hardware specifications. *MPC860EC/D*, Mortorola 1998.
[21] 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 *RTSS*, Dec 1996.
[22] T. Pering, T. Burd, and R. Brodersen. The simulation and evaluation of dynamic voltage scaling algorithms. *ISLPED*, pages 76–81, 1998.
[23] P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *SOSP*, 2001.
[24] 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.
[25] G. Quan and X. Hu. Enhanced fixed-priority scheduling with (m,k)-firm guarantee. In *2000 IEEE Real-Time System Symposium*, pages 79–88, 2000.
[26] G. Quan and X. S. Hu. Energy efficient fixed-priority scheduling for real-time systems on voltage variable processors. In *DAC*, pages 828–833, 2001.
[27] 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.
[28] K. F. S. Reinhardt and T. Mudge. Automatic performance-setting for dynamic voltage scaling. *Proceedings of the 7th Conference on Mobile Computing and Networking MOBICOM'01*, 2001.
[29] D. Shin, J. Kim, and S. Lee. Intra-task voltage scheduling for low-energy hard real-time applications. *IEEE Design and Test of Computers*, 18(2), March-April 2001.
[30] V. Swaminathan and K. Chakrabarty. Investigating the effect of voltage switching on low-energy task scheduling in hard real-time systems. In *ASP-DAC*, pages 251–254, June 2001.
[31] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced cpu energy. In *Proceedings of USENIX Symposium on Operating System Design and Implementation*, pages 13–23, 1994.
[32] R. West and K. Schwan. Dynamic window-constrained scheduling for multimedia applications. In *ICMCS*, 1999.
[33] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. In *IEEE Annual Found. of Comp. Sci.*, pages 374–382, 1995.
[34] Q. Zheng and K. G. Shin. On the ability of establishing real-time channels in point-to-point packet-switched networks. *IEEE Trans. on Comm.*, 42(2/3/4):1096–1105, 1994.