Energy-Aware Scheduling for Practical Mode Real-Time Systems with QoS Guarantee

Linwei Niu
Department of Math and Computer Science
Claflin University
Orangeburg, SC 29115
linwei.niu@claflin.edu

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

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 research in energy-aware real-time scheduling has been focused on hard real-time systems, we study the problem of minimizing energy for soft real-time systems with the requirements of QoS-guarantee. In this paper, 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. A dynamic DVS algorithm is presented to reduce the energy while guaranteeing the given (m, k)-constraints. The simulation results demonstrate that our proposed techniques can achieve significant energy saving performance while ensuring the QoS guarantee.

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., [3, 10, 30]) have been proposed to reduce the energy consumption of real-time computing systems. Most of these techniques have targeted the 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 that do not finish by their deadlines can still be completed with a reduced value [15] or they can simply be dropped without compromising the desired QoS levels. 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 DVS techniques in real-time scheduling to deal with the power/energy conservation with QoS constraints simultaneously. These approaches can be in general classified into two categories: *the best-effort* approaches and *the guarantee* approaches. The *best-effort* approaches (e.g. [28, 26, 16, 18]) 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 *guarantee* 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, e.g. [22, 9].

The statistic guarantee ensures the quality of service in a probabilistic manner which can be problematic for some real-time applications. To provide the real-time system with deterministic QoS, the systems should not only support the overall guarantee of the QoS statistically, but also be able to provide the lower bounded, predictable level of QoS at each specified time interval. The (m, k)-model, proposed by Hamdaoui *et. al.* [7], is such an effort in this endeavor. 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 consecutive k jobs more than $(k - m)$ job instances miss their deadlines. West *et. al.* [29] introduced another model, called the *window-constrainted* model, which requires that at least m jobs over any *non-overlapped* and *consecutive* windows containing k jobs meet their deadlines.

For its intuitiveness and capability of capturing not only statistical but also deterministic QoS requirements, $(m, k)-$model has been widely studied, e.g., [4, 23, 7, 25, 8]. Quan *et. al.* [23] formally proved that scheduling with $(m, k)-$guarantee is NP-hard in strong sense. To guarantee the (m, k)-constraints, Ramanathan *et. al.* [25] first proposed a strategy to partition the jobs into *mandatory* or *optional* jobs. The mandatory jobs are 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 ensure the schedulability of mandatory jobs, they introduced a closed-form formula based on the exact worst case response analysis similar to that in [12]. Quan *et. al.* [23] improved the partitioning strategy by reducing the maximal interference between the mandatory jobs. Bernat *et.al.* [5] proposed to use 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 *panic* mode if the dynamic failure is likely to occur. All these work primarily targeted at systems with fixed priority assignment. Note that, even with the *window constraints*, the guaranteed scheduling problem is still 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]. None of these approaches have taken energy/power consumption into consideration.

In this paper, we study the problem of reducing the energy consumption for real-time systems with (m, k)-guarantee requirement. Our approach consists of two phases. In the offline phase, 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 online phase, the mandatory job sets are dynamically varied to accommodate the dynamic nature of real-time embedded systems with (m, k)-constraints. Through our extensive experiments, the results show that our proposed approach 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 models and formulate our problem. Section 3 presents some theoretical results that form the basis for our techniques. Section 4 introduces our new approach in more detail. 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 System models

The real-time system considered in this paper contains n independent periodic tasks, $\mathcal{T} = \{\tau_0, \tau_1, \cdots, \tau_{n-1}\}$, scheduled according to the earliest deadline first (EDF) policy [14]. 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 consecutive k_i jobs of τ_i, at least m_i jobs must 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 $\mathcal{S} = \{S_1, ..., 1\}$.

3 Meeting the (m,k)-constraints

To reduce energy consumption for real-time systems while guaranteeing the (m, k)-constraints, we can partition the jobs into mandatory/optional jobs statically or dynamically. In this section, we study the properties of two special *statically* defined mandatory/optional partitions. These properties form the basis for our energy saving approaches in section 4.

3.1 Deeply-Red Partition

The first partition strategy is called *deeply-red partition* or R-partition, proposed by Koren emph et.al [11]. According to this scheme, a job τ_{ij}, *i.e.*, the j^{th} job of task τ_i, is determined to be mandatory if

$$0 \leq j \ mod \ k_i < m_i, \quad j = 0, 1, 2, ... \tag{1}$$

The following theorem provide us a necessary and sufficient condition based on the work demand analysis strategy similar to those in [32] and [13] to predict schedulability for the mandatory jobs determined by the R-partitions.

Theorem 1 *Let system $\mathcal{T} = \{\tau_0, \tau_1, ..., \tau_{n-1}\}$, where $\tau_i = \{T_i, D_i, C_i, m_i, k_i\}$, $\mathcal{R}$ be the mandatory job set according to their R-partitions, and L be either the ending point of the first busy period or the* least common multiple *of $T_i, i = 0, ..., (n-1)$, whichever is smaller. Also, let $W(0, t)$ represent the total workload from $\mathcal{R}$ that arrive at or after time 0 and have to be finished before t. Then $\mathcal{R}$ is schedulable* iff *all the mandatory jobs arriving within $[0, L]$ can meet their deadlines,* i.e.

$$\sum_i W_i(0, t) \leq t \tag{2}$$

for all $t \leq L$ and $t = pT_i + D_i, p \in Z, p \ mod \ k_i \leq m_i, i = 0, ..., n-1$;

3.2 Evenly Distributed Partition

The second partitioning strategy that we study is called *the evenly distributed partition*, or *E-partition*, proposed by Ramanathan [25]. According to this strategy, a job τ_{ij}, *i.e.*, the j^{th} 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.

The mandatory/optional job partitions according to equation(3) has the interesting property that it helps to spread out the required jobs *evenly* in each task along the time.

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

After presenting the schedulability results for a task set with (m, k)-constraints, we are now ready to introduce our approach to reduce the energy consumption for this type of applications. In this section, we will propose a dynamic DVS scheduling algorithm to minimize the energy consumption while guaranteeing the (m, k)-constraints. Our algorithm consists of two phases: an off-line phase followed by an on-line phase. During the off-line phase, to ensure the (m, k)-constraints, the feasibility of the mandatory job sets with the static R-partitions is tested and the speeds for the mandatory jobs are slowed down at the task level. During the on-line phase, the mandatory/optional job partitioning is dynamically adjusted to achieve better energy saving performance while guaranteeing the (m, k) requirements.

4.1 The off-line phase

Once the mandatory jobs are defined with the static R-partition, an intuitive approach to dynamically scale the processor speed is to adopt the well-known algorithm as introduced in [30], which can compute the voltage schedule for arbitrary job sets to optimize the energy consumption. However, the problem with this approach is that it can achieve the optimal energy conservation only under the assumption that the supply voltage for the processor can vary continuously, which is not practical, or the energy savings can be severely degraded, especially when there are only a few discrete voltage levels available in the commercial processors.

In the off-line phase of our approach, we tried to slowed down the processor speed at the task level first. In order to choose the best speed assignments that can optimize the energy savings, some classic exhaustive search algorithms such as branch-and-bound can be adopted to find the optimal solution. Next, we can further improve the energy performance at the on-line phase.

4.2 The online phase

Once the mandatory job sets are statically determined at the off-line phase, we can further reduce the energy consumption by varying the job partitions dynamically at the online phase. Algorithm 1 presents the salient part of our on-line scheduling algorithm.

As shown in Algorithm 1, two job ready queues are maintained: the mandatory queue (Q_M) and the optional queue (Q_P). Upon arrival, a job is determined as mandatory job or optional job based on the execution results of the $k_i - 1$ jobs in the most recent history. It is determined as mandatory only if one more deadline miss will incur dynamic failure. The optional jobs are put in the Q_P, and the mandatory jobs will be put in the Q_M.

The jobs in Q_M have higher priority than the jobs in Q_P, and will be executed following the EDF scheme with the corresponding speeds determined during the off-line phase. Since some jobs in the Q_P can also meet their deadlines, it provides us more opportunities to vary the mandatory/optional job partitions correspondingly to save energy because the execution of an optional job helps to reduce the possibility of having to run mandatory jobs at high processor speeds in the future. On the other hand, it is not difficult to see that there may be more than one optional jobs available in the Q_P, and selecting which one to execute may have profound impacts for the future job executions. While the jobs in the Q_P can be simply run at the lowest speed, we use a more delicate heuristic to achieve better energy saving performance. Specifically, when the Q_M is empty, we first compute the speed $\hat{S}_i$ that is required to finish each optional job in the Q_P by the earliest arrival time for the coming mandatory jobs or its own deadline, whichever is smaller. Then those optional jobs with required speed less than their predetermined speed S_i will be chosen as candidate jobs. After that, the energy gain ΔE_i for 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 chosen optional job will be executed non-preemptively to guarantee it can meet the deadline.

Algorithm 1 The online phase of the dynamic approach. (Algorithm MK_{DYN})

```
1:  Input: two job ready queues($Q_M$ and $Q_P$).
2:  if $Q_M$ is not empty then
3:      Run jobs in $Q_M$ according to EDF;
4:  else if $Q_P$ is not empty then
5:      $t_{cur}$ = the current time;
6:      $t_a$ = the earliest arrival time of the coming mandatory jobs;
7:      $\mathcal{J}_p$ =jobs in $Q_P$;
8:      Select and run $J_i \in \mathcal{J}_p$ non-preemptively if $C_{rem}(J_i) \leq (\min\{d_i, t_a\} - t_{cur})$ and $\Delta E_i$ is maximal;
9:      // $C_{rem}(J_i)$ is the remaining execution time of $J_i$;
10:     // $d_i$ is absolute deadline of $J_i$;
11: end if
```

The energy efficiency of our dynamic approach lies in the fact that it adjusts the mandatory/optional partition adaptively with the run-time condition. It is particular 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 execution of jobs in Q_M, the dynamic resource reclaiming techniques similar to the ones in [3, 10] 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 *Let $\mathcal{T} = \{\tau_0, \tau_1, ..., \tau_{n-1}\}$, where $\tau_i = \{T_i, D_i, C_i, m_i, k_i\}$. Algorithm 1, with complexity of $O(n)$, can ensure the (m,k)-requirements for $\mathcal{T}$ if $\mathcal{T}$ is schedulable under R-partition.*

5 Experimental results

In this section, three different approaches are studied using experiments. For the first approach, the task sets were statically partitioned with E-partition and all mandatory jobs from each task were executed with the highest speed. We referred this approach as (MK_{NoDVS}) and used its results as the reference results. For the second approach (MK_{ST}), the task sets were also statically partitioned with E-partition. But the processor speeds of the mandatory jobs for each task were slowed down according to the static approach introduced in [24]. The third approach MK_{DYN} is our dynamic approach, as explained in section 4. For all of the approaches to be compared, when dynamic slack time was available due to the early completion of mandatory jobs that did not present their worst case execution time, it was reclaimed using techniques similar to those introduced in [3, 10]. We assumed the processor model we used had five discrete voltage levels and the corresponding normalized speed frequencies were (0.2, 0.4, 0.6, 0.8, 1.0).

We first studied the energy-saving performance of these three approaches. The periodic task sets tested in our experiments were randomly generated with the periods randomly chosen in the range of $[10ms, 50ms]$ 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 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 2 to 10, and $m_i < k_i$. 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 task sets schedulable with MK_{DYN}, or until at least 5000 task sets within each interval have been generated. The energy consumption for each approach was normalized to that by MK_{NoDVS}, and the results are shown in Figure 1.

From Figure 1, compared with applying dynamic slack reclaiming alone on E-partitions, slowing down the speeds at the task level first and then applying dynamic slack reclaiming has much better energy saving performance. For example, MK_{ST} can reduce the energy consumption by up to 48% when compared with MK_{NoDVS}. Dynamically adjusting the partitions can reduce the energy much further. For example, compared with MK_{NoDVS}, MK_{DYN} can reduce the energy by up to 72%. Even when compared

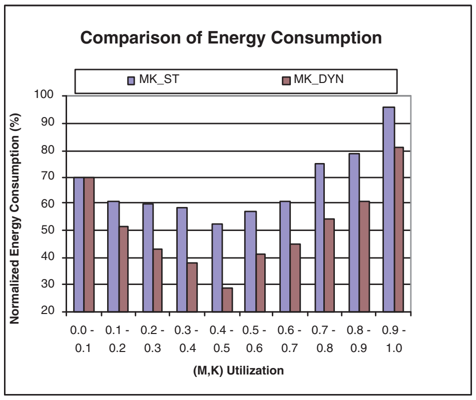

Figure 1. The energy consumption by the different approaches.

with MK_{ST}, MK_{DYN} can reduce the energy by more than 30% on average. The largest energy reduction can be up to 50%. It is also interesting to notice that MK_{DYN} has better energy-saving performance than MK_{NoDVS} and MK_{ST} even though more jobs were executed with MK_{DYN}. This is because selectively scheduling the optional jobs at relatively lower processor speeds helps to alleviate the pressure to run the mandatory workload at higher processor speeds in the future.

We next study the (m,k)-guarantee capability of MK_{ST} and MK_{DYN}. In this 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 MK_{ST} or at least 5000 different task sets have been generated for each interval. We collected the number of feasible task sets as well as the total number of jobs within $LCM(k_iT_i), i = 0, ..., n-1$, that can meet their deadlines (called the *effective jobs*) by each approach. These numbers are normalized to that by MK_{ST} and listed in Table 1.

As shown in Table 1, by distributing the mandatory jobs for each task evenly, MK_{ST} in general has a stronger (m,k)-guarantee capability than MK_{DYN}, especially when the utilization is high. On the other hand, when MK_{DYN} can feasibly schedule a task set, it tends to provide better QoS levels since because more effective jobs met their deadlines by MK_{DYN}. The QoS improvement can be up to 39% when the utilization is relatively low because there is more space to schedule the optional jobs with lower speeds. Even when the utilization is higher, the QoS of MK_{DYN} is still better than the other approaches due to the successful scheduling of the optional jobs. On the other hand, even though more jobs were scheduled by MK_{DYN}, the energy consumption by MK_{DYN} was much less than the other approaches(as shown in Figure 1).

6 Conclusions

Low power/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,

(m, k)	Schedulable Task Sets		Effective Jobs	
Util	MK_{ST}	MK_{DYN}	MK_{ST}	MK_{DYN}
0.0-0.1	100	100	100	213.6
0.1-0.2	100	100	100	187.2
0.2-0.3	100	100	100	181.9
0.3-0.4	100	97	100	191.1
0.4-0.5	100	85	100	165.7
0.5-0.6	100	81	100	139.0
0.6-0.7	100	77	100	140.0
0.7-0.8	100	70	100	127.6
0.8-0.9	100	65	100	118.5
0.9-1.0	100	54	100	108.8

Table 1. Average number of schedulable task sets and effective jobs of MK_{ST} and MK_{DYN} for randomly generated task sets

we first presented the necessary and sufficient conditions to check the schedulability for the static mandatory/optional workload partitioning. Then we proposed a dynamic DVS scheduling algorithm to achieve the dual goals of QoS guarantee and energy minimization. As shown in our experiments, with excellent adaptivity to the run-time environments, the proposed DVS scheduling algorithm outperforms previous research significantly in terms of both energy savings and QoS levels that can be provided.

References

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

[2] T. A. AlEnawy and H. Aydin. Energy-constrained scheduling for weakly-hard real-time systems. *RTSS*, 2005.

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

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

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

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

[10] 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.

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

[12] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *RTSS*, pages 166–171, 1989.

[13] J. Liebeherr, D. Wrege, and D. Ferrari. Exact admission control for networks with a bounded delay service. *IEEE Trans. on Networking*, 4(6):885–901, 1996.

[14] 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.

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

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

[17] A. R. M. Ltd. Introduction to thumb. version 2.0. *ARM DVI-0001A*, 1995.

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

[19] MPC860. Mpc860 powerpc hardware specifications. *MPC860EC/D*, Motorola 1998.

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

[21] L. Niu and G. Quan. A hybrid static/dynamic dvs scheduling for real-time systems with (m, k)-guarantee. *RTSS*, 2005.

[22] 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.

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

[24] G. Quan, L. Niu, and J. P. Davis. Power aware scheduling for real-time systems with (m,k)-guarantee. In *CNDS'04*, 2004.

[25] 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.

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

[27] I. Ripoll, A. Crespo, and A. Mok. Improvement in feasbility testing for real-time tasks. *Journal of Real-Time Systems*, 11:19–40, 1996.

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

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

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

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

[32] 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.