# Priority-Monotonic Energy Management for Real-Time Systems with Reliability Requirements*

Dakai Zhu, Xuan Qi
Department of Computer Science
University of Texas at San Antonio
San Antonio, TX 78249
{dzhu,xqi}@cs.utsa.edu

Hakan Aydin
Department of Computer Science
George Mason University
Fairfax, VA 22030
aydin@cs.gmu.edu

## Abstract

*Considering the impact of the popular energy management technique Dynamic Voltage and Frequency Scaling (DVFS) on system reliability, the* Reliability-Aware Power Management (RA-PM) *problem has been recently explored to save energy while maintaining system reliability. In this work, focusing on Rate Monotonic Scheduling (RMS) policy, we study static RA-PM schemes for periodic real-time tasks. After showing the intractability of the problem, we focus on two widely-known feasibility tests for RMS (namely, the Liu-Layland bound and Time Demand Analysis) and propose a number of heuristics based on the priority-monotonic speed assignment. The heuristics are evaluated through extensive simulations.*

## 1 Introduction

Due to the drastically increased power densities in computing systems, energy has recently become an important concern in system design. Moreover, as a traditional design metric, the reliability concerns triggered by increased *transient fault* rates have become prominent with the continued scaling of CMOS technologies and reduced design margins [5]. In general, energy management schemes exploit available system *slack* and operate system components at lower performance states, whenever possible, to save energy consumption. For example, *Dynamic Voltage and Frequency Scaling (DVFS)* technique exploits unused CPU time to scale down processor frequency and supply voltage simultaneously to save energy [18]. Furthermore, system slack can also be used to tolerate transient faults by the temporal-redundancy-based *backward error recovery* techniques that restore the system state to a previous safe state and repeat the computation/task [13].

*The research of Hakan Aydin was supported by NSF CAREER Award CNS-0546244.

Although energy management [1, 4, 8] and fault tolerance have been studied extensively, these two major research streams have surprisingly remained somewhat isolated to this date. Considering the *conflicting* nature of the DVFS-based energy management and backward recovery based transient-fault tolerance techniques, where both techniques are based on the active use of the system slack [20], the interplay between energy management and reliability introduces non-trivial problems, especially considering the implications of DVFS on transient faults [23].

Taking the *negative* effect of DVFS on reliability into consideration, we have proposed the *reliability-aware power management (RA-PM)* framework, which reserves part of the slack to schedule a recovery task and thus to preserve the system reliability before utilizing the remaining slack for energy savings [20]. The scheme was further extended to consider multiple real-time tasks that share a common deadline [21] and periodic real-time tasks scheduled by the *earliest-deadline-first* scheduling algorithm [22].

In this paper, focusing on the *Rate Monotonic Scheduling (RMS)*, we study the static RA-PM schemes for periodic real-time tasks. Note that RMS is known as the *optimal* fixed-priority periodic scheduling algorithm [10]. Moreover, with the direct support in common real-time operating systems and well-established timing analysis methodologies, RMS remains as the most well-known and common real-time scheduling policy in practice.

In RMS, tasks are assigned static (fixed) priorities inversely proportional to their periods. Any adjustment for high priority tasks will affect the execution of low priority tasks and thus the feasibility of the whole task set. In [14], the *priority-monotonic* analysis, where speed assignment is performed in decreasing order of task priorities, was used for RMS-based energy management. In this paper, following a similar priority-monotonic approach, we study static RA-PM schemes for the design of energy-efficient and reliable real-time systems.

1-4244-1258-7/07/$25.00 ©2007 IEEE

The remainder of this paper is organized as follows. The background on RMS scheduling and our models are presented in Section 2. In Section 3, we illustrate that the static RMS-based RA-PM problem is NP-hard and propose a few priority monotonic based heuristics. Simulation results are discussed in Section 4 and Section 5 concludes the paper.

## 2 Background and System Models

### 2.1 Task Model and RMS Scheduling

We consider a set of independent periodic real-time tasks $\Gamma = \{T_1, \ldots, T_n\}$, where the task $T_i$ is represented by its worst case execution time (WCET) $c_i$ and period $p_i$ ($i = 1, \cdots, n$). On DVFS settings, it is assumed that $c_i$ is given under the maximum processing speed $f_{max}$ and it scales *linearly* with the reduced processing speed[1]. That is, at speed $f$, the execution time of task $T_i$ is assumed to be $c_i \cdot \frac{f_{max}}{f}$. Moreover, $p_i$ coincides with the $T_i$'s relative deadline. The $j^{th}$ job of $T_i$ that arrives at time $(j-1) \cdot p_i$ needs to finish its execution by the time $j \cdot p_i$.

Without loss of generality, we assume that $p_i \leq p_{i+1}$ ($i = 1, \cdots, n-1$). That is, with RMS, $T_i$ will have higher priority than $T_{i+1}$. For RMS, various feasibility tests have been studied with different accuracy and complexity (see [11] for a detailed discussion). In this paper, for simplicity and illustration purposes, we focus on two widely-known feasibility tests: **Liu-Layland Bound (LLB)** [10] and **Time Demand Analysis (TDA)** [9].

The *system utilization* is defined as $U = \sum_{i=1}^{n} u_i$, where $u_i = \frac{c_i}{p_i}$ is task $T_i$'s utilization. The well-known feasibility condition states that a task set is schedulable under preemptive RMS if the system utilization $U$ does not exceed the *Liu-Layland bound (LLB)* given by [10]:

$$U \leq n(2^{\frac{1}{n}} - 1) = LLB(n) \tag{1}$$

where $n$ is the number of tasks.

Although LLB provides a linear-time test, it only states a *sufficient* condition for feasibility. In [9], Lehoczky *et al.* developed the exact *time demand analysis (TDA)* as the *sufficient* and *necessary* RMS feasibility test [11]. Here, the time demand function $wq_i(t)$ of task $T_i$ is defined as:

$$wq_i(t) = c_i + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil c_k, \text{for } 0 < t \leq p_i \tag{2}$$

The task set is considered feasible if, for each task $T_i$, it is possible to find a time instant $t$ such that $wq_i(t) \leq t \leq p_i$.

[1]A number of studies have indicated that the execution time of tasks does not scale linearly with reduced processing speed due to accesses to memory [16] and/or I/O devices [2]. However, exploring the full implications of this observation is beyond the scope of this paper and is left as our future work.

### 2.2 Power Model

Although dynamic power dissipation, which is quadratically related to supply voltage and linearly related to frequency [3], dominates in processors, system-wide power management has become a necessity considering the ever-increasing static leakage power and other power consuming components (e.g., memory) [1, 4, 8, 14]. In this work, we adopt the system-level power model introduced in [23]:

$$P(f) = P_s + \hbar(P_{ind} + P_d) = P_s + \hbar(P_{ind} + C_{ef} f^m) \tag{3}$$

where $P_s$ is the *static power*, $P_{ind}$ is the *frequency-independent active power* and $P_d$ is the *frequency-dependent active power*. When the system is *active* (where computation is in progress) $\hbar = 1$; otherwise, the system is in sleep modes or turned off and $\hbar = 0$. The effective switching capacitance $C_{ef}$ and the dynamic power exponent $m$ (which is, in general, no smaller than 2) are system dependent constants [3] and $f$ is the processing frequency.

With non-zero $P_s$ and $P_{ind}$ values, it may not be energy efficient to execute all tasks at the *lowest* frequency that satisfies the feasibility requirement. Hence, an energy-efficient frequency $f_{ee}$, below which the tasks start to consume more *total energy*, does exist [8, 14, 23]. Considering the prohibitive overhead of turning on/off a system, if the system is put to sleep states for saving energy when idle, it is possible to obtain an expression for $f_{ee}$ as $\sqrt[m]{\frac{P_{ind}}{C_{ef} \cdot (m-1)}}$ [23].

### 2.3 Fault Model and Problem Description

Considering the dominance of transient faults among all fault types in computing systems [5, 7, 17], we focus on transient faults in this paper. We assume that the fault inter-arrival rate follows the Poisson distribution [19]. Note that the number of transient faults increases with reduced *critical charge* (which is the smallest charge needed to cause a transient fault) at lower supply voltages [6, 15]. Therefore, for DVFS-enabled computing systems, the average transient fault rate $\lambda(f)$ at scaled frequency $f(\leq f_{max})$ (and corresponding supply voltage $V$) can be modeled as [23]:

$$\lambda(f) = \lambda_0 \cdot g(f) \tag{4}$$

where $\lambda_0$ is the average fault rate at $f_{max}$ (and $V_{max}$). That is, $g(f_{max}) = 1$. For scaled frequencies and supply voltages, the fault rate generally increases and $g(f) > 1$ for $f < f_{max}$.

Taking the effects of DVFS on transient faults into consideration, the specific problem that will be addressed in this paper can be stated as follows: **for a periodic real-time task set that is schedulable by the preemptive RMS, how to utilize the system slack (i.e., the spare CPU capacity) for power management to save energy without reducing the system reliability**.

### 2.4 Reliability-Aware Power Management

Before presenting our solutions for the problem, we first review the concept of *Reliability-Aware Power Management (RA-PM)* [20]. Defining the *original reliability* of task $T_i$ as $R_i^0$, which is the probability of completing one job of $T_i$ correctly with its WCET at $f_{max}$ (i.e. without DVFS), from the Poisson fault arrival model and the average fault rate of $\lambda_0$, we can obtain $R_i^0 = e^{-\lambda_0 c_i}$. To preserve reliability, the RA-PM scheme uses part of the available slack to schedule a recovery job (in the form of *re-execution* at $f_{max}$) in case that a fault is detected at the end of task execution through *sanity* or *consistency* checks [13]. The remaining slack is used to scale down the job's execution to save energy. Suppose that the scaled frequency is $f(< f_{max})$. The job's *effective reliability* can be given by [20]:

$$R_i = e^{-\lambda(f)\cdot c_i/f} + \left(1 - e^{-\lambda(f)\cdot c_i/f}\right) \cdot R_i^0 > R_i^0 \qquad (5)$$

where the first part is the probability of the scaled job being executed correctly and the second part presents the probability of incurring faults during the scaled execution while the recovery job completes successfully. That is, **if the amount of available slack is sufficient, scheduling a recovery job for every job to be scaled can always preserve the overall task system's reliability [20]**.

## 3 Fixed Priority RA-PM Schemes

From the above discussion, to address the effects of DVFS on transient faults and preserve the system reliability, we need to schedule a recovery job for every *scaled job* (that refers to any job whose execution is slowed down through DVFS, for energy management purposes) within its deadline. Hence, by exploiting the spare processor capacity, we could construct a *recovery task* that has the *same* timing parameters (i.e., WCET and period) as those of a scaled task [22]. Then, for *any* job of a scaled task, the recovery task can provide a recovery job within its deadline. The recovery job will be invoked and executed at the maximum frequency $f_{max}$, should a fault occur during the execution of the scaled job, to preserve system reliability.

Without loss of generality, suppose that a subset $\Phi$ ($\subseteq \Gamma$) of tasks are *selected* for scaling down and the scaled frequency for task $T_k \in \Phi$ is $f_k$ ($f_{ee} \leq f_k < f_{max}$) (the remaining tasks run at $f_{max}$). To preserve reliability, a recovery task will be constructed for each task $T_k \in \Phi$. If the *augmented* task set is schedulable, without considering the energy consumed by recovery tasks (which normally have a small probability of being executed), the *fault-free* energy consumption within the *least common multiple (LCM)* of the tasks' periods will be:

$$E(\Phi) = \sum_{T_i \in (\Gamma - \Phi)} \frac{LCM}{p_i} P(f_{max}) c_i + \sum_{T_i \in \Phi} \frac{LCM}{p_k} P(f_k) \frac{c_k f_{max}}{f_k} \qquad (6)$$

where the first part is the energy consumed by *unscaled* tasks and the second part is the energy consumed by the scaled tasks. Hence, the RA-PM problem for real-time tasks scheduled by RMS can be re-phrased as follows: **for a *given* feasibility test, find the subset $\Phi$ of tasks and corresponding scaled frequencies to minimize $E(\Phi)$ while preserving the system reliability.**

**Intractability of the problem:** Note that, if all tasks have the same period, the special case of the problem becomes essentially an RA-PM problem for multiple tasks with a shared deadline, which has been shown to be NP-hard [21]. Therefore, the RA-PM problem for RMS is also *intractable*.

In what follows, we will study a number of priority-monotonic based heuristics to determine the subset $\Phi$ and the scaled frequencies for different feasibility tests.

### 3.1 LLB-based RA-PM Schemes

First, we study the utilization-based schemes and use the Liu-Layland bound given in Equation (1). If the system utilization $U$ of a task set with $n$ tasks satisfies $U \leq LLB(n)$, the task set is schedulable and the spare capacity is $sc = LLB(n) - U$, which can be used to scale down the execution of tasks and accommodate the required recovery tasks. Note that, when calculating the Liu-Layland bound, we only need to count the number of tasks that have different periods [10]. Since the periods of the *newly-constructed* recovery tasks are the same as those of the scaled tasks, we have the following corollary:

**Corollary 1** *The addition of the recovery tasks will not change the Liu-Layland bound for the task set under consideration.*

Therefore, the generalized static LLB-based RA-PM problem is to find out the subset $\Phi$ and the scaled frequencies so as to

$$minimize(E(\Phi))$$

subject to

$$\sum_{T_i \in (\Gamma - \Phi)} u_i + \sum_{T_k \in \Phi} \frac{u_k \cdot f_{max}}{f_k} + \sum_{T_k \in \Phi} u_k \leq LLB(n) \qquad (7)$$

where the left hand-side of the above condition shows the *effective system utilization* of the augmented task set by considering the scaled executions as well as the potential load of the required recovery tasks.

Considering the convex relation between system power consumption and processing speed (see Equation (3)), for a given subset $\Phi$ of tasks to be scaled down, the optimal solution to minimize $E(\Phi)$ will consist in *uniformly scaling down* all the tasks in the subset $\Phi$. Suppose that the common scaled frequency is $f_{llb}$ and the accumulated utilization for tasks in the subset $\Phi$ is $U_\Phi = \sum_{T_k \in \Phi} u_k$. From the above equations, in order to have a feasible augmented task set with the Liu-Layland bound, the lowest scaled frequency should be at least $f_{llb} = max\{f_{ee}, \frac{U_\Phi}{LLB(n)-U} f_{max}\} = max\{f_{ee}, \frac{U_\Phi}{sc} f_{max}\}$ , where $sc$ is the spare capacity.

By substituting $f_{llb}$ in Equation (6) and differentiating $E(\Phi)$ over $U_\Phi$, we can find that $E(\Phi)$ will be minimized when $U_\Phi = su \cdot \left(\frac{P_{ind}+C_{ef}}{m \cdot C_{ef}}\right)^{\frac{1}{m-1}} = U_\Phi^{opt}$. If $U_\Phi^{opt} \geq U$, all tasks will be scaled down accordingly to minimize the energy consumption. Otherwise, this becomes essentially a task selection problem, where the accumulated utilization of the selected tasks should be as close as possible to $U_\Phi^{opt}$.

**RAPM-LLB:** Building on the idea of *priority-monotonic* speed assignment [14], the *RAPM-LLB* scheme will choose the $k$ highest priority tasks for DVFS and schedule the corresponding recovery tasks, where $k$ is the largest integer that satisfies $\sum_{i=1}^{k} u_i \leq U_\Phi^{opt}$. The whole computation can be done in time $O(n)$. Note that, when a scaled job completes its execution correctly, the corresponding recovery job can be freed and the time slots allocated for the recovery job will become slack, which could be exploited for more energy savings at run-time by low priority tasks. Therefore, the idea of priority-monotonic energy management could be further justified by considering that the slack generated from freeing the recovery jobs of high priority tasks is more likely to be re-used by low priority tasks at run-time; thus, more energy savings could be expected.

### 3.2 TDA-based RA-PM Schemes

The second class of the RA-PM schemes we investigate are based on *Time Demand Analysis (TDA)* for RMS. With the scaled tasks in the subset $\Phi$ and their corresponding recovery tasks, the *modified* time demand function $mwq_i(t)$ for task $T_i \in \Gamma$ will be:

$$mwq_i(t) = \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil c_k + \sum_{T_k \in \Phi, 1 \leq k \leq i-1} \left\lceil \frac{t}{p_k} \right\rceil \frac{c_k \cdot f_{max}}{f_k} + \begin{cases} c_i & \text{if } T_i \notin \Phi; \\ c_i \cdot (1 + \frac{f_{max}}{f_i}) & \text{if } T_i \in \Phi. \end{cases} \quad (8)$$

where $f_k$ is the scaled frequency for task $T_k \in \Phi$ and $0 < t \leq p_i$. The function incorporates the time demand from all (scaled and unscaled) tasks as well as the required recovery tasks. It is not difficult to see that the augmented task set is schedulable if, for every task $T_i \in \Gamma$, there is a time instant $t$ such that $mwq_i(t) \leq t \leq p_i$. Therefore, the TDA-based RA-PM problem can be formulated as: find out the subset $\Phi$ and the scaled frequencies so as to

$$minimize(E(\Phi))$$

subject to

$$\forall T_i \in \Gamma, \exists t, mwq_i(t) \leq t \leq p_i, \text{where } 0 < t \leq p_i$$

In general, there are two basic steps in solving the problem: a.) selecting the subset $\Phi$ of tasks; and b.) assigning the scaled frequencies for tasks in the subset $\Phi$. For the first step of task selection, again, we will take the *priority monotonic* heuristic approach. That is, we suppose that the first $x$ high-priority tasks are selected; in other words, $\Phi = \{T_1, \cdots, T_x\}$. Again, recall that the tasks are ordered by their priorities where $T_1$ has the highest priority and $T_n$ has the lowest priority.

Next, in increasing level of sophistication, we discuss various schemes to determine the scaled frequency(ies) for the selected tasks.

**Single scaled frequency with PS test:** Suppose that the $x$ highest priority tasks are selected for energy management. Starting with the single frequency assignment assumption for the selected tasks, the **RAPM-PS** scheme approximates the TDA technique by focusing on the first deadline of every task, as in Pillai-Shin test [12]. The scaled frequency $f_{RAPM-PS}(x)$ is calculated as:

$$f_{RAPM-PS}(x) = \max \left\{ f_{ee}, \max_{T_i \in \Gamma} \{f_i(x)\} \right\}, \text{ where}$$

$$f_i(x) = \begin{cases} \frac{\sum_{k=1}^{i} \left\lceil \frac{p_i}{p_k} \right\rceil c_k}{p_i - \sum_{k=1}^{i} \left\lceil \frac{p_i}{p_k} \right\rceil c_k} f_{max} & \text{if } i \leq x; \\ \frac{\sum_{k=1}^{x} \left\lceil \frac{p_i}{p_k} \right\rceil c_k}{p_i - \sum_{k=1}^{i} \left\lceil \frac{p_i}{p_k} \right\rceil c_k} f_{max} & \text{if } i > x. \end{cases} \quad (9)$$

Here, the calculated scaled frequency $f_{RAPM-PS}(x)$, is for the selected tasks only and the remaining tasks run at $f_{max}$. Moreover, the calculation in Equation (9) considers the recovery tasks needed for reliability preservation.

If, for a given $x$, the resulting frequency $f_{RAPM-PS}(x) > f_{max}$ (that is, $\exists T_i \in \Gamma$ where $f_i(x) > f_{max}$), it means that selecting $x$ highest priority tasks is not feasible with the approximate TDA test. Otherwise, applying Equation (6) to all feasible task selections, we can get the optimal number $x$ of selected tasks and the corresponding scaled frequency $f_{RAPM-PS}(x)$ that minimize $E(\Phi)$. The worst-case time complexity of RAPM-PS is $\mathbf{O}(n^3)$.

**Single scaled frequency with exact TDA test:** Instead of checking only the first period of every task, the **RAPM-TDA** scheme resorts to the *exact* TDA test that considers

*all* the time instants within a task's period to get a lower scaled frequency and thus better energy savings. The scaled frequency, $f_{RAPM-TDA}(x)$, for the selected $x$ highest priority tasks will be:

$$f_{RAPM-TDA}(x) = \max\left\{f_{ee}, \max_{T_i \in \Gamma}\{f_i(x)\}\right\} \quad (10)$$

$$f_i(x) = \min_{0<t\leq p_i}\{f_i(x,t)\} \quad (11)$$

$$f_i(x,t) = \begin{cases} \frac{\sum_{k=1}^{i}\left\lceil \frac{t}{p_k}\right\rceil c_k}{\sum_{k=1}^{i}\left\lceil \frac{t}{p_k}\right\rceil c_k + (t - mwq_i(t))} f_{max} & \text{if } i \leq x; \\ \frac{\sum_{k=1}^{x}\left\lceil \frac{t}{p_k}\right\rceil c_k}{\sum_{k=1}^{x}\left\lceil \frac{t}{p_k}\right\rceil c_k + (t - mwq_i(t))} f_{max} & \text{if } i > x. \end{cases} \quad (12)$$

where $f_i(x)$ is the scaled frequency determined by task $T_i$ and $mwq_i(t)$ is the modified time demand as defined in Equation (8) assuming that $f_k = f_{max}$. As in RAPM-PS, for the augmented task set with the $x$ highest priority tasks being scaled and with recovery tasks, if there does not exist a feasible time instant for any task $T_i$ (i.e., $mwq_i(t) > t$ for $0 < t \leq p_i$), the augmented task set is not schedulable with the exact TDA test. Otherwise, by applying Equation (6), we can get the energy consumption after scaling the $x$ highest priority tasks. Searching through all the feasible task selections, RAPM-TDA can find out the optimal $x_{opt}$ and the corresponding scaled frequency that give the minimal energy consumption of $E(\Phi)$ in pseudo-polynomial time. The complexity of RAPM-TDA can be easily found to be $\mathbf{O}(n^3 r)$, where $r = \frac{p_n}{p_1}$ is the ratio of the largest period to the smallest period.

**Multiple scaled frequencies with exact TDA test:** Note that, in RAPM-TDA, it is possible that the resulting single scaled frequency is constrained by a high priority task $T_k$ in the subset $\Phi$. That is, $T_k$ has more stringent timing constraints and requires a higher scaled frequency. For such cases, by exploiting the slack from the high frequency assignment for high priority tasks, the **RAPM-TDAM** scheme will re-calculate and assign a lower frequency for low priority tasks in the subset $\Phi$, and thus save more energy.

More specifically, for a given subset $\Phi$ with $x$ highest priority tasks, if the single scaled speed obtained by RAPM-TDA is $f(x) = f_k(x)$ and $k < x$, we can assign the scaled frequency for the first $k$ high priority tasks and re-calculate the scaled frequency for the remaining tasks in the subset $\Phi$. With the frequency assignment for the first $k$ highest priority tasks being fixed, the modified work demand function and scaling factor for task $T_i$ ($k < i \leq n$) can be re-calculated as:

$$mwq_i(t) = \sum_{j=1}^{k}\left\lceil \frac{t}{p_j}\right\rceil \left(1 + \frac{f_{max}}{f_j}\right) c_j + \begin{cases} \sum_{j=k+1}^{i}\lceil t/p_j\rceil\, 2\cdot c_j & \text{if } i \leq x; \\ \sum_{j=k+1}^{x}\lceil t/p_j\rceil\, 2\cdot c_j + \sum_{j=x+1}^{i}\left\lceil \frac{t}{p_j}\right\rceil c_j & \text{if } i > x. \end{cases} \quad (13)$$

$$f_i(x,t) = \begin{cases} \frac{\sum_{j=k+1}^{i}\left\lceil \frac{t}{p_j}\right\rceil c_j}{\sum_{j=1}^{i}\left\lceil \frac{t}{p_j}\right\rceil c_j + (t - mwq_i(t))} f_{max} & \text{if } i \leq x; \\ \frac{\sum_{j=k+1}^{x}\left\lceil \frac{t}{p_j}\right\rceil c_j}{\sum_{j=1}^{x}\left\lceil \frac{t}{p_j}\right\rceil c_j + (t - mwq_i(t))} f_{max} & \text{if } i > x. \end{cases} \quad (14)$$

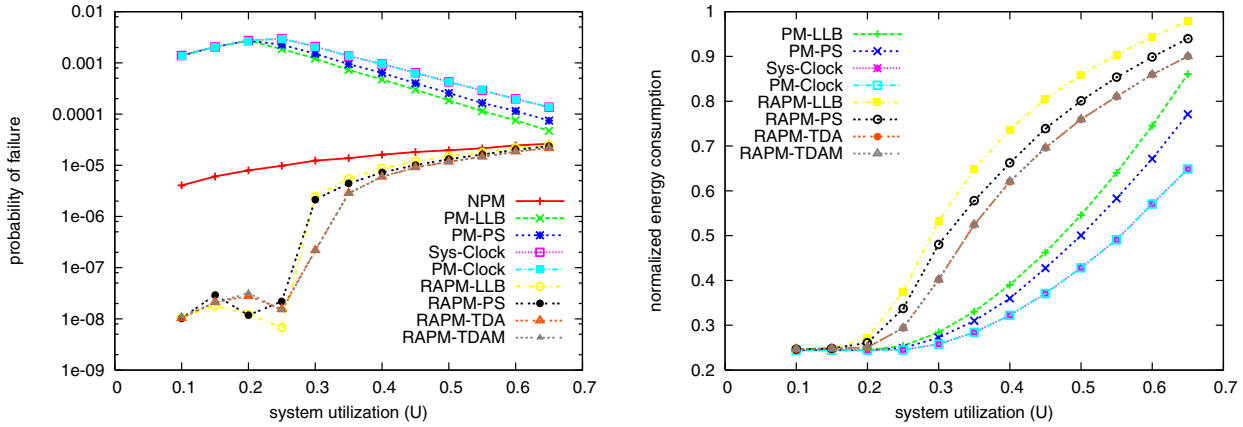
After re-calculating the scaled frequencies for tasks $T_{k+1}$ to $T_n$, we can obtain the maximum frequency $f^{new}(x)$. Suppose that $f^{new}(x) = f_q^{new}(x)$, where $k+1 \leq q \leq n$. If $q \geq x$, the scaled frequency for the remaining tasks in the subset $\Phi$ will be $f^{new}(x)$. Otherwise, we can assign $f^{new}(x)$ as the scaled frequency for tasks $T_{k+1}$ to $T_q$, and then repeat the above process until we complete the frequency assignment for all tasks in the subset $\Phi$. After that, the energy consumption for the case of selecting $x$ highest priority tasks can be calculated. Checking through all possible values of $x$, finally we could obtain the optimal value of $x_{opt}$ and corresponding frequency settings that result in the minimum energy consumption. With an additional round to assign the possible different scaled frequencies for tasks in the subset, one can derive the complexity of the RAPM-TDAM scheme as $\mathbf{O}(n^4 r)$, where, again, $r = \frac{p_n}{p_1}$ is the ratio of the largest period to the smallest period.

# 4 Simulation Results

We evaluate our proposed heuristic schemes through extensive simulations with synthetic real-time task sets. For comparison, in addition to the RA-PM schemes proposed in this work (i.e., **RAPM-LLB**, **RAPM-PS**, **RAPM-TDA** and **RAPM-TDAM**), we implemented the following *reliability-ignorant* static fixed priority power management schemes: **PM-LLB** (which scales all tasks uniformly to speed $f_{PM-LLB} = \frac{U}{LLB(n)}$; see Equation 1), **PM-PS** [12], **Sys-Clock** and **PM-Clock** [14]. Moreover, as the baseline for evaluation, we consider the scheme of *no power management (***NPM***)*, which executes all tasks at $f_{max}$ and puts the system to sleep states when idle.

Focusing on the active power, we assume that $P_{ind} = 0.05$, $C_{ef} = 1$ and $m = 3$. Considering normalized frequency with $f_{max} = 1$, the energy efficient frequency is $f_{ee} = 0.29$ (see Section 2). Moreover, as in [22], the transient faults are assumed to follow the Poisson distribution with an average fault rate of $\lambda_0 = 10^{-5}$ at $f_{max}$ (and corresponding supply voltage). For the fault rates at lower frequencies/voltages, we adopt the exponential fault rate model $g(f) = \lambda_0 10^{\frac{d(1-f)}{1-f_{min}}}$ [23] and assume that $d = 2$. That is, the average fault rate is 100 times higher at the lowest frequency $f_{ee}$ (and corresponding supply voltage). The effects of different values of $d$ have been evaluated previously [20, 21, 23].

For the synthetic real-time task sets, each set contains 20 periodic tasks and the periods of tasks are generated randomly following a uniform distribution within the range of $[20, 200]$. For task sets containing fewer tasks (e.g., 5)




a. probability of failure (1−reliability)

b. normalized energy consumption

**Figure 1. Reliability and energy for different static schemes.**

or shorter periods (e.g., $[10, 20]$), similar results have been obtained and are not shown due to space limitation. The WCETs of tasks are uniformly distributed in the range of 1 and their periods. Finally, the WCETs of tasks are scaled such that the system utilization of tasks is set to a desired value [12]. To ensures that all task sets are schedulable under RMS, the maximum system utilization considered is 0.65 [10]. Assuming that tasks take their WCET (i.e., without considering the variability of tasks' execution time), we emulate the execution of a task set for $10^8$ time units. That is, approximately 20 million jobs are executed during each run. Moreover, for each result point in the graphs, 100 task sets are generated and the presented results correspond to the average.

**Reliability Performance:** First, Figure 1a shows the probability of failure (i.e., $1-$*reliability*) under different system utilizations for all the schemes. Here, the probability of failure shown is the ratio of the number of failed jobs over the total number of jobs executed.

From the figure, we can see that, as system utilization increases, for NPM, the probability of failure increases slightly since the computational load of each task increases and tasks run longer, which increases the probability of being subject to transient fault(s). The probability of failure for ordinary static power management schemes (i.e., PM-LLB, PM-PS, Sys-Clock and PM-Clock) is much higher than that of NPM due to both increased fault rates and extended execution time. Note that, the minimum energy efficient frequency is $f_{ee} = 0.29$. For very low system utilization (i.e., $U \leq 0.2$), all ordinary schemes execute all tasks at $f_{ee}$ and the probability of failure is the same. Then, it increases slightly with increased utilization for the same reason as that for NPM. However, when system utilization further increases (i.e., $U \geq 0.3$), all schemes need to run the tasks with a higher speed (thus, lower fault rates) and the probability of failure for all schemes decreases. Moreover, as indicated in [14], PM-Clock assigns the same speed as Sys-Clock for most of the task sets, which results in almost the same level of probability of failure.

For reliability-aware schemes (i.e., RAPM-LLB, RAPM-PS, RAPM-TDA and RAPM-TDAM), by incorporating a recovery task for each task to be scaled, the probability of failure is lower than that of NPM and system reliability is preserved, which confirms the theoretical result described in Section 2. For the same reason as NPM, the probability of failure increases at very low system utilization. Interestingly, the probability of failure becomes smaller for all reliability-aware schemes at system utilization of 0.2 or 0.25. The reason is that, all tasks can still be recovered but need to run at a higher frequency than $f_{ee}$, which results in lower fault rates and better system reliability. Moreover, by exploiting exact time demand analysis, RAPM-TDA and RAPM-TDAM could manage more tasks and better system reliability can be obtained for most of the cases, but at the cost of increased complexity. For the same reason as PM-Clock versus Sys-Clock, RAPM-TDAM assigns the same speed as RAPM-TDA for most of the task sets, which leads to almost the same level of probability of failure.

**Energy Performance:** Figure 1b further shows the normalized energy consumption with NPM as a baseline. Not surprisingly, with some spare capacity being utilized for recovery tasks to preserve system reliability, the reliability-aware schemes have less spare capacity for power management and generally consume more (up to 35%) energy compared to ordinary power management schemes. Even for very low system utilization, although not distinguishable,

the reliability-aware schemes consume $0.2\%$ to $0.5\%$ more energy due to the execution of the recovery jobs. Moreover, the energy consumption for Sys-Clock and PM-Clock is almost the same, which is consistent with previous results [14]. And for similar reasons, RAPPM-TDA and RAPM-TDA also consume almost the same amount of energy.

From the figure, we can also see that RAPM-TDA and RAPM-TDAM could save $12\%$ more energy compared with RAPM-LLB. Furthermore, with the exact TDA test, it is expected that RAPM-TDA and RAPM-TDAM can schedule task sets with higher system utilization.

## 5 Conclusions

Energy has recently become an important design metric and reliability issue also becomes more prominent with the scaled technology sizes and reduced design margin. Considering the negative effects of *Dynamic Voltage and Frequency Scaling (DVFS)*, a popular energy management technique, on system reliability, we investigated in this work static *reliability-aware power management (RA-PM)* schemes for real-time tasks scheduled by RMS policy. Focusing on two different feasibility tests for RMS, namely, *Liu-Layland bound (LLB)* and *time demand analysis (TDA)*, we first showed that the problem is NP-hard. Then, following the *priority-monotonic* speed assignment idea, a number of RA-PM heuristics were proposed for both LLB and TDA tests. We evaluated the proposed heuristics through extensive simulations with synthetic real-time tasks. The results confirm that the RA-PM schemes can preserve the system reliability while achieving significant energy savings. In comparison, the existing reliability-ignorant power management schemes could lead to drastically reduced system reliability. Moreover, with the complex feasibility test, TDA-based RA-PM schemes could obtain $12\%$ more energy savings when compared with LLB-based schemes.

## References


[1] H. Aydin, V. Devadas, and D. Zhu. System-level energy management for periodic real-time tasks. In *Proc. of The* $27^{th}$ *IEEE Real-Time Systems Symposium (RTSS)*, Piscataway, NJ, USA, Dec. 2006. IEEE CS Press.

[2] E. Bini, G. Buttazzo, and G. Lipari. Speed modulation in energy-aware real-time systems. In *Proc. of the* $17^{th}$ *Euromicro Conference on Real-Time Systems*, 2005.

[3] T. D. Burd and R. W. Brodersen. Energy efficient cmos microprocessor design. In *Proc. of The HICSS Conference*, Jan. 1995.

[4] Y. Cho and N. Chang. Memory-aware energy-optimal frequency assignment for dynamic supply voltage scaling. In *Proc. of the 2004 int'l symposium on Low power electronics and design (ISLPED)*, pages 387–392, 2004.

[5] D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N. S. Kim, and K. Flautner. Razor: circuit-level correction of timing errors for low-power operation. *IEEE Micro*, 24(6):10–20, 2004.

[6] P. Hazucha and C. Svensson. Impact of cmos technology scaling on the atmospheric neutron soft error rate. *IEEE Trans. on Nuclear Science*, 47(6):2586–2594, 2000.

[7] R. Iyer, D. J. Rossetti, and M. Hsueh. Measurement and modeling of computer reliability as affected by system activity. *ACM Trans. on Computer Systems*, 4(3):214–237, Aug. 1986.

[8] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proc. of the* $41^{st}$ *Design automation conference*, 2004.

[9] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: exact characterization andaverage case behavior. In *Proc. of the IEEE Real-Time Systems Symposium*, pages 166–171, Dec. 1989.

[10] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *J. ACM*, 20(1):46–61, 1973.

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

[12] P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *Proc. of* $18^{th}$ *ACM Symposium on Operating Systems Principles*, Oct. 2001.

[13] D. K. Pradhan. *Fault-Tolerant Computing: Theory and Techniques*. Prentice Hall, 1986.

[14] S. Saewong and R. Rajkumar. Practical voltage scaling for fixed-priority rt-systems. In *Proc. of the* $9^{th}$ *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2003.

[15] N. Seifert, D. Moyer, N. Leland, and R. Hokinson. Historical trend in alpha-particle induced soft error rates of the alpha$^{TM}$ microprocessor. In *Proc. of the* $39^{th}$ *Annual International Reliability Physics Symposium*, 2001.

[16] K. Seth, A. Anantaraman, F. Mueller, and E. Rotenberg. Fast: Frequency-aware static timing analysis. In *Proc. of the* $24^{th}$ *IEEE Real-Time System Symposium*, 2003.

[17] N. Wang, J. Quek, T. Rafacz, and S. Patel. Characterizing the effects of transient faults on a high-performance processor pipeline. In *Proc. of the 2004 International Conference on Dependable Systems and Networks (DS N)*, Jun. 2004.

[18] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for reduced cpu energy. In *Proc. of The First USENIX Symposium on Operating Systems Design and Implementation*, Nov. 1994.

[19] Y. Zhang, K. Chakrabarty, and V. Swaminathan. Energy-aware fault tolerance in fixed-priority real-time embedded systems. In *Proc. of Int'l Conference on Computer Aided Design*, Nov. 2003.

[20] D. Zhu. Reliability-aware dynamic energy management in dependable embedded real-time systems. In *Proc. of the IEEE Real-Time and Embedded Technology and Applications Symposium*, 2006.

[21] D. Zhu and H. Aydin. Energy management for real-time embedded systems with reliability requirements. In *Proc. of the Int'l Conf. on Computer Aidded Design*, Nov. 2006.

[22] D. Zhu and H. Aydin. Reliability-aware energy management for periodic real-time tasks. In *Proc. of the IEEE Real-Time and Embedded Technology and Applications Symposium*, 2007.

[23] D. Zhu, R. Melhem, and D. Mossé. The effects of energy management on reliability in real-time embedded systems. In *Proc. of the Int'l Conf. on Computer Aidded Design*, 2004.