# Energy-Aware Task Allocation for Rate Monotonic Scheduling

Tarek A. AlEnawy and Hakan Aydin
Computer Science Department
George Mason University
Fairfax, VA 22030
{thassan1,aydin}@cs.gmu.edu

## Abstract

*We consider the problem of energy minimization for periodic preemptive hard real-time tasks that are scheduled on an identical multiprocessor platform with dynamic voltage scaling capability. We adopt partitioned scheduling and assume that the tasks are assigned rate-monotonic priorities. We show that the problem is NP-Hard in the strong sense on $m \geq 2$ processors even when the feasibility is guaranteed a priori. Because of the intractability of the problem, we propose an integrated approach that consists of three different components: RMS admission control test, the partitioning heuristic and the speed assignment algorithm. We discuss possible options for each component by considering state-of-the-art solutions. Then, we experimentally investigate the impact of heuristics on feasibility, energy and feasibility/energy performance dimensions. In off-line settings where tasks can be ordered according to the utilization values, we show that Worst-Fit dominates other well-known heuristics. For on-line settings, we propose an algorithm that is based on reserving a subset of processors for light tasks to guarantee a consistent performance.*

## 1. Introduction

Recently, the energy awareness has been promoted to a first-class computer system design and evaluation factor. It has been reported that the CPU accounts for a significant fraction of the total system energy consumption in both battery-operated computers and high-end servers [19, 30]. One common CPU energy management technique is **Dynamic Voltage Scaling (DVS)**. DVS entails reducing the system energy consumption by reducing the CPU supply voltage and the clock frequency (CPU speed) simultaneously. Despite the (potentially) quadratic energy gains, the response times increase when the CPU speed is reduced. Thus, many recent real-time (RT) research studies investigate ways to guarantee the timing constraints on DVS-enabled settings for various task/system models and off-line/on-line scheduling algorithms [6, 7, 8, 28, 31, 32, 33]. Maximizing the system performance with limited energy budget for a given operation time has been studied in [1]. The same work also proves the *intractability* of the feasibility problem for RT systems with discrete speed levels and fixed energy budget.

Multiprocessor RT Scheduling is one of the most extensively studied areas in RT systems research. Two main approaches for multiprocessor RT scheduling can be identified in the research literature: partitioned scheduling and global scheduling [15, 18]. In the partitioned approach, the admission control module permanently assigns each task to a processor, i.e. task migration at run-time is not allowed. Each processor has its own ready queue and scheduler. Partitioned multiprocessor real-time scheduling is known to be NP-Hard in the strong sense [20, 25]. On the other hand, simple and effective partitioning heuristics such as First-Fit and Best-Fit are shown to have reasonable average-case performance [13, 29]. Moreover, well-established algorithms (e.g. RMS or EDF) from uniprocessor RT scheduling theory can be adopted once the tasks are allocated to different processors. Global scheduling, on the other hand, takes a different approach: A global scheduler selects from a single ready queue for execution the highest-priority $n$ tasks on $m$ processors. Tasks are allowed to migrate between processors. One fundamental result in multiprocessor RT scheduling theory states that partitioned and global scheduling are incomparable, in the sense that there are task sets that can be scheduled in a feasible manner by only one approach [25]. The reader is referred to the recent surveys and studies on partitioned and global scheduling for more information [2, 3, 4, 9, 21]. However, the partitioned approach is arguably more common in current multiprocessor platforms thanks to its simplicity and ease of implementation. The traditional dichotomy of static- and dynamic-priority periodic scheduling policies also holds on multiprocessor settings. In the former, each task is permanently assigned a fixed priority level; while task/job priorities may change dynamically in the latter [14]. Although dynamic-priority algorithms usually lead to better processor utilization, a considerable majority of real-time and embedded systems are still implemented using static-priority policies. This may be partially explained by the well-established theory/status of Rate Monotonic Analysis (which is known to be optimal for uniprocessor static-priority scheduling [27]) and the limited number of priority levels in most commercial operating systems.

**Energy-aware multiprocessor RT scheduling**. Although numerous research papers have explored energy management on uniprocessor RT systems, fewer studies have considered the problem of energy management on multiprocessor platforms. The problem of energy minimization by dynamic slack reclamation and dynamic speed adjustment for global scheduling is considered in [36]. This work is extended in [35] to address the case of dependent tasks with AND/OR dependency constraints. The authors also propose a few variations of speculative algorithms exploiting the statistical information about the workload. In [34], the problem of power-aware resource allocation for independent periodic hard RT tasks on heterogeneous multiprocessors is formulated as an extended *Generalized Assignment Problem* (*GAP*), based on integer linear programming formulation. In [8], Aydin et al. address the problem of energy minimization for periodic hard RT tasks on identical multiprocessors with DVS when partitioned scheduling is used. The authors adopt the EDF scheduling policy and investigate the joint effect of partitioning heuristics on the energy consumption and the feasibility. Finally,

Baruah and Anderson address the system synthesis problem of periodic RT tasks on identical multiprocessors, using global EDF in [10].

**This work**. We consider the problem of energy minimization for periodic preemptive hard RT tasks that are scheduled on an identical multiprocessor platform with DVS capability. We adopt **partitioned scheduling** and assume that tasks are assigned **static (rate-monotonic) priorities**. An obvious benefit of adopting partitioned approach stems from the fact that well-established *dynamic reclaiming* techniques [6, 28, 31, 33] for *periodic energy-aware scheduling* can be readily adopted on each processor once the task allocation is made. In contrast, dynamic reclaiming for global periodic multiprocessor scheduling is an open problem to a considerable extent. We believe that avoiding the overhead of task migrations is another incentive to focus on the partitioned approach. To the best of our knowledge, this is the first research effort considering the problem of energy minimization on multiprocessors with static-priority periodic scheduling.

Our work is based on the observation that the **task allocation can have a significant impact on the overall energy consumption of the system**. In particular, two different and feasible task allocations can result in very different energy consumption levels. As in any partitioned real-time scheduling approach, resorting to heuristics for **task allocation (partitioning)** appears to be a necessity. Another dimension is the **admission control algorithm (feasibility test)** to be used during the task allocation phase. The wealth and diversity of existing feasibility tests, from the seminal work of Liu and Layland [27] to the exact characterization of Lehoczky et al. [24] and the more recent hyperbolic test [12], call for a trade-off analysis on computational complexity, feasibility, and energy consumption dimensions. Finally, once the tasks are assigned permanently to the processors, a **speed assignment** scheme must be chosen to reduce the energy consumption while preserving the feasibility. Here again, simple schemes scaling up the effective utilization by a constant factor or more sophisticated ones based on the critical instant analysis of static-priority tasks [33] are possible.

We summarize below the main **contributions** of this paper:

- We formally define the problem of energy minimization in multiprocessor periodic RT scheduling with partitioning, and show that the problem remains NP-Hard in the strong sense even when the feasibility is guaranteed a priori.
- We study and evaluate a number of well-known partitioning heuristics, RMS admission control algorithms, and speed assignment schemes in terms of the feasibility performance and overall energy consumption.
- We consider two different settings of the problem: In *off-line partitioning*, all the tasks and their characteristics are known to the scheduler. Thus, it is possible to order tasks first according to utilization or period values to improve the performance. However, if the scheduler is to allocate tasks in a given order without the possibility of waiting for the rest of the tasks (as in the case of *on-line partitioning*, where tasks arrive dynamically to the system), then the problem becomes more difficult. We present an analysis of both cases. We show that Worst-Fit is a clear winner for off-line partitioning. However, Worst-Fit's performance degrades rapidly in on-line settings. For this case, we propose a hybrid algorithm that reserves a number of processors for tasks with small utilizations (*light tasks*).

The paper is organized as follows. In Section 2 we present the system model and assumptions. In Section 3 we define the problem and discuss the tractability. Section 4 discusses our solution approach. In Section 5, we present the experimental evaluation of task allocation, admission control, and speed assignment schemes. We conclude the paper in Section 6.

# 2. System model and assumptions

## 2.1 Power and energy consumption model

We consider a multiprocessor platform $M$ with $m$ processors $M_1, \ldots, M_m$. The number of processors $m$ is fixed during the operation; that is, on-demand addition of processors is not possible at run-time. We also assume that all $m$ processors are identical in terms of the processing power and speed/energy characteristics. Each processor $M_i$ has **dynamic voltage scaling** (**DVS**) capability according to which it can adjust its operating speed $S$ (expressed as cycles per unit time). We denote the maximum speed level available in the system by $S_{max}$ . Without loss of generality, the speed values will be normalized with respect to $S_{max}$ (i.e. $S_{max}$ = 1.0). The CPU power dissipation function when running at speed $S$ is denoted by $g(S)$. In current DVS architectures, $g(S)$ is taken to be a strictly convex and increasing function, often a polynomial of at least the second degree [6, 33]. In any time interval $[t_1,t_2]$ the total energy consumption of a processor is given by:

$$E(t_1, t_2) = \int_{t_1}^{t_2} g(S(t))dt \quad (1)$$

where $S(t)$ is the processor speed as a function of time. In current DVS-enabled systems, it is not always possible to find a speed level that exactly corresponds to a desired target speed (often obtained assuming a continuous speed spectrum). Instead, a DVS-enabled processor has a finite number of discrete speeds $\{s_1, \ldots, s_k\}$. Often, the solutions developed for an ideal DVS architecture (with continuous speed) can be adapted to these settings by choosing the lowest speed level available in the system that is equal to or greater than a target speed $S$. The focus of this study is the CPU energy consumption; the energy profiles of memory and I/O subsystems, albeit important, are beyond the scope of this paper.

## 2.2 Task and scheduling model

We consider a set of $n$ independent periodic hard real-time tasks $T = \{T_1, \ldots, T_n\}$. Each task $T_i = (C_i, P_i)$ is characterized by two parameters: the worst-case number of processor cycles $C_i$ required by $T_i$ and a period $P_i$. We assume that the period $P_i$ is equal to the relative deadline $D_i$. On variable-speed settings, the worst-case workload of a task $T_i$ is given by the worst-case number of cycles. The worst-case execution time of task $T_i$ when running at speed $S$ is given by $c_i = C_i/S$. We consider a preemptive scheduling model; the preemption and speed change overheads can be incorporated in $C_i$ if necessary.

The utilization of task $T_i$ under CPU speed $S$ is given by $u_i(S) = C_i/(P_i S)$. Note that under maximum CPU speed (i.e. $S$ = 1.0), $u_i(1.0) = C_i/P_i$. The aggregate utilization of the task set (under maximum speed) is given by $U_{tot} = \sum_i C_i/P_i$. Note that a necessary (but not sufficient) condition for feasibility on a system of $m$ identical multiprocessors is to have a task set whose total utilization does not exceed the computing capacity.

Consequently, we assume that the condition $U_{tot} \leq m$ holds throughout the paper.

We adopt a partitioning-based approach to multiprocessor scheduling. Tasks are assigned *permanently* to processors. The total utilization of tasks assigned to processor $M_i$ (under maximum speed) is denoted by $U_i$. Note that $U_{tot} = \sum_k U_k$. Similarly, $n_i$ denotes the number of tasks assigned to processor $M_i$. On each processor, the well-known Rate Monotonic Scheduling (RMS) policy is adopted: tasks are assigned static priorities that are inversely proportional to their periods.

In multiprocessor RT system analysis, for a given task set, it is often helpful to determine the largest utilization among all the tasks in the set. This parameter, called the *utilization factor,* will be denoted by $\alpha$. Formally, $\alpha = max(u_i) \; \forall \; T_i \in T$.

## 3. Energy minimization with partitioning for static-priority scheduling

Traditionally, the multiprocessor real-time scheduling studies have focused mostly on the *feasibility* issue. One of the motivations of this research effort is to consider the total *energy* consumption as an additional performance metric. Ideally, on energy-constrained settings, the objective should be to allocate tasks and compute CPU speed assignments while minimizing the total energy consumption *and* preserving the feasibility. In fact, it is possible to formally state the problem (denoted by RMS-ENERGY-PARTITION) as follows:

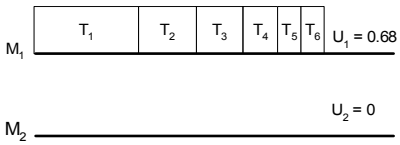
**RMS-ENERGY-PARTITION**: Given a set $T$ of periodic hard real-time tasks and a set $M$ of identical processors, *find* a task-to-processor assignment (i.e. *partition*) and compute task-level speeds on each processor such that:

1. the workload can be scheduled by RMS in a feasible manner, and
2. the total energy consumption on all processors $M_1, \ldots, M_m$ is *minimum* (among all feasible partitions).

As stated above, RMS-ENERGY-PARTITION is an optimization problem in which the objective function is the total energy consumption of $M$, subject to the constraint that the workload on each processor will be *feasible* when scheduled by rate-monotonic priorities.

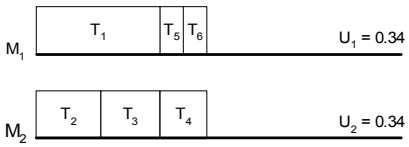
In view of the convex relationship between the CPU speed and the power consumption, in general, energy-aware partitioning across DVS-enabled multiprocessors suggests load balancing techniques [8]. In fact, a perfectly balanced partition, if it exists, is also provably optimal in terms of energy consumption when tasks are scheduled by the Earliest-deadline-first (EDF) policy [8]. In the context of RMS, however, the problem gains new dimensions as shown by the following example.

**Motivational Example**: Consider a set of six periodic hard real-time tasks $T = \{T_1, \ldots, T_6\}$ to be scheduled using partitioning on two identical processors. The individual task utilizations under maximum speed are $u_1 = 0.32$, $u_2 = 0.2$, $u_3 = 0.1$, $u_4 = 0.04$, $u_5 = 0.01$ and $u_6 = 0.01$. Since the total utilization $U_{tot} = 0.68$ is less than the asymptotic Liu-Layland bound *ln 2*, *any* partitioning of these tasks is feasible with both EDF and RMS. Figures 1-3 depict three different partitions and the energy consumption patterns under RMS and EDF policies.



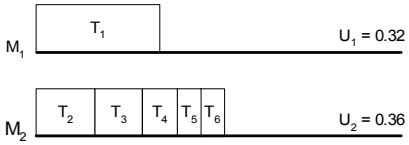

| | RMS | EDF |
|---|---|---|
| $S_1$ | 0.925 | 0.68 |
| $S_2$ | Idle | Idle |
| $E_{tot}$ | **5818** | **3144** |

**Figure 1. A feasible partition (Partition 1)**




| | RMS | EDF |
|---|---|---|
| $S_1$ | 0.436 | 0.34 |
| $S_2$ | 0.436 | 0.34 |
| $E_{tot}$ | **1295** | **786** |

**Figure 2. A feasible and perfectly balanced partition (Partition 2)**




| | RMS | EDF |
|---|---|---|
| $S_1$ | 0.32 | 0.32 |
| $S_2$ | 0.484 | 0.36 |
| $E_{tot}$ | **1171** | **794** |

**Figure 3. A feasible and slightly unbalanced partition (Partition 3)**

Observe that the partitioning shown in Figure 1 corresponds to the schedule that would be produced by well-known heuristics First-Fit and Best-Fit. The partitioning shown in Figure 2 is the perfectly balanced one. Finally, the one in Figure 3 corresponds to the case where the first processor is exclusively dedicated to the task with the largest utilization, while all other tasks are assigned to the second processor. We underline that when judged solely by the feasibility criterion, all three partitions are equally acceptable.

Yet, the energy dimension yields a different picture. Let us compute the energy consumption $E_{tot}$ of each partition for both EDF and RMS scheduling. Using simple algebraic manipulation it is possible to show that, the energy consumption of processor $M_i$ when running at constant speed $S_i$ during the interval $[0,P]$, where P is the hyperperiod of all the tasks (i.e. $P = lcm(P_1, P_2, \ldots, P_6)$), is $E(M_i) = P \cdot U_i \cdot g(S_i) / S_i$ [8]. Recall that $U_i$ is the total utilization of tasks assigned to $M_i$ under CPU speed $S_i = 1.0$. For illustration purposes, we take $P = 10000$. For EDF, the speed assignment scheme used on each processor is the optimal one: $S_i = U_i$ [6, 31]. For RMS, the speed is determined through the uniform slow-down approach [31]. Thus, in the RMS case, the speed $S_i$ of processor $M_i$ with $n_i$ tasks is $U_i / U_{bound}(n_i)$, where $U_{bound}(k) = k \; (2^{1/k} - 1)$ is the well-known Liu-Layland schedulability bound [27]. The power consumption function $g(S) = S^3$ [6, 33].

It is easy to see that Partition 1 has the maximum energy among all three partitions considered for both EDF and RMS. However, the partition with least energy is different for EDF and RMS. Under EDF, the perfectly balanced partition, Partition 2, has the minimum energy consumption among all three partitions. On the other hand, under RMS, the slightly unbalanced Partition 3 has about 10% lower energy consumption than Partition 2. Although the difference in processor utilizations between partitions 2 and 3 is minor, there is a significant difference in the number of tasks assigned to each processor in each case. In particular, in Partition 3 processor $M_1$ has only 1 task, compared to 3 in partition 2. This

results in a situation where the speed of the first processor can be reduced to 0.32, well below 0.436 which was possible with the second partition.

This simple example shows that feasible partitions can have significantly different energy characteristics. For RMS, Partition 1 has the largest energy consumption among all three partitions considered, which is almost five times as much as that of Partition 3. Load balancing while partitioning does help to reduce the energy consumption. However, in the case of RMS, a perfectly balanced partition is not necessarily the most energy efficient option: The example illustrates the fact that, with the uniform slow-down approach of RMS, the optimal partition is a function of both the processor utilization and the number of tasks. Although omitted for lack of space, similar examples can be constructed for other speed assignment schemes proposed in literature [28, 31, 33].

From the computational complexity point of view, RMS-ENERGY-PARTITION is NP-Hard in the strong sense, since it is a more general form of the feasibility problem in partitioned multiprocessor scheduling which is known to be NP-Hard in the strong sense [13]. More interestingly, the problem remains NP-Hard in the strong sense *even when the task set is known to be trivially schedulable on one processor*. In this case, *any* partitioning would yield a feasible schedule, but computing the one with the minimum energy consumption is still intractable.

**Proposition 1.** RMS-ENERGY-PARTITION is NP-Hard in the strong sense even when the feasibility is guaranteed a priori.

**Proof:** Consider a special instance of RMS-ENERGY-PARTITION where the task periods are *harmonic*, i.e. each task period is an *exact multiple* of another one. In this case, the schedulability bound for RMS is $U_{bound} = 1.0$ [27], exactly the same as that of EDF. Under this assumption, the problem is identical to POWER-PARTITION with EDF which is proved to be NP-Hard in the strong sense in [8] even for trivially schedulable task sets with $U_{tot} \leq 1.0$. Hence, the general problem of RMS-ENERGY-PARTITION with RMS is also NP-Hard in the strong sense for trivially schedulable task sets. ■

# 4. Our framework

The intractability of the problem necessitates a heuristic-based approach. Moreover, the solution has to integrate multiple design components as a number of issues need to be addressed in an energy-aware multiprocessor platform:

- **What RMS admission control algorithm to use on each processor?** The set of eligible processors for a given task is determined by ensuring that the RMS feasibility is preserved. Consequently, a uniprocessor RMS admission control algorithm has to be adopted. The *accuracy* of the algorithm/test is certainly important in order to be able to admit more tasks, but the *efficiency* is also a concern as the test will be potentially invoked on each of the *m* processors. We classify existing RMS admission algorithms in two general classes: those using the utilization and period information when making a decision [12, 22, 27], and those based on *Time Demand Analysis* ([24]). We provide an extended discussion of the RMS admission control algorithms considered in this study in Section 4.1.
- **What partitioning heuristic to use?** When multiple processors are eligible for a given task, an *efficient* partitioning heuristic determines the processor to which the task will be assigned. A number of efficient heuristics are available from *bin-packing* research area (see Section 4.2).
- **What speed assignment scheme to adopt?** Once the tasks are allocated in such a way that the workload on each processor is feasible according to RMS, the last step will involve the computation of the CPU speed. The schedule must remain *feasible* even with the reduced speed. Uniprocessor speed assignment schemes that are recently proposed in the literature ([28, 31, 33]) for RMS are potential candidates, offering a spectrum of computational complexity and energy savings (see Section 4.3).

We underline that *none* of these dimensions admits an *exact* and *polynomial-time* solution as of today. Moreover, **our objective is to provide a computational cost and performance benefit analysis of different schemes in terms of both feasibility and overall energy consumption.** Note that the use of RMS instead of EDF as the scheduling algorithm on each processor makes the *admission control* and *speed assignment* phases even more difficult. For EDF, the necessary and sufficient schedulability condition on processor $M_i$ is $U_i \leq 1.0$ [27]. Moreover, the optimal speed to minimize the total energy consumption while meeting all the deadlines is known to be $S_{EDF} = U_i$ [6]. In contrast, for RMS no uniprocessor polynomial-time exact feasibility test (i.e. one that provides necessary and sufficient condition(s) for feasibility) exists to this date. The Time Demand Analysis technique is exact, but it runs in pseudo-polynomial time [5, 24]. It follows that determining optimal speed assignments to minimize the total energy consumption (while ensuring feasibility) does not assume an efficient solution as of today. We now provide an overview of design options on each dimension.

## 4.1 RMS Admission Control Algorithms

The selection of the RMS admission control to be used at processor level can have significant impact on the feasibility performance of the system. Several uniprocessor RMS feasibility tests with varying accuracy and time complexity characteristics exist (see [26] for a detailed discussion). For our purposes, we classify the tests in two major categories:

### *4.1.1 Utilization-based feasibility tests*

These are fast (i.e. polynomial-time), yet *approximate* tests using the information about the utilization of the task set and/or individual tasks. The tests in these categories provide *sufficient conditions* for feasibility: a task set is deemed feasible if the condition is not violated. Further, some of the proposed solutions exploit the *additional* information about task periods, yielding two sub-categories.

Basic utilization-based tests

The simplest (and consequently, *fastest)* schemes require only the information about the task utilizations.

- **Exact Liu-Layland test (ELL)** is by far the most frequently used/adopted feasibility test for RMS. A task set with $n$ tasks is schedulable on one processor if the total utilization $U_{tot}$ does not exceed the Liu-Layland bound $U_{bound}(n) = n(2^{1/n} - 1)$. Note that there is a different version of Liu-Layland test that uses the *asymptotic* bound of *ln 2*; we use the word "exact" to underline the difference.
- **Hyperbolic test (HYP),** recently proposed by Bini et al. [11], provides a tighter bound than ELL by considering individual task utilizations: $\prod_{i=1}^{n}(1+u_i) \le 2$.

Utilization-based tests exploiting the period information

In their seminal paper [27], Liu and Layland had observed that the schedulability bound of RMS was equal to 100% for *harmonic* (or, *simply periodic)*[1] task sets. Later, Kuo and Mok established that any *utilization-bound* test for RMS schedulability could be used with the number of harmonic task subsets (as opposed to the number of tasks) [22]. Over the years, the research community developed more sophisticated tests by exploiting similar properties of task periods:

- **Burchard test (Burc)** builds on the idea that having harmonic tasks helps to improve the utilization bound. The utilization bound provided is yet another generalization of Liu-Layland bound. It is expressed as a function of the number of tasks $n$ and an additional parameter that quantifies how close the tasks are to being harmonic [13].
- **R-bound test** transforms a given task set into one where the ratio $r$ of maximum period to minimum period does not exceed 2. Lauzac et al. show that the *original* task set is schedulable if the transformed task set is schedulable. The utilization bound depends on $r$ and the number of tasks $n$: $U_{bound}(n,r) = (n-1)(r^{1/(n-1)} - 1) + 2/r - 1$ [23].

It is worth mentioning that Burchard et al. proposed two other schemes, *RMST* and *RMGT* specifically designed for partitioned static-priority scheduling on multiprocessors [13]. Similar to *Burc*, they make use of the fact that the utilization bound improves as tasks are closer to being harmonic. They also require that the tasks be pre-ordered by a parameter that quantifies how close the tasks are to being harmonic. *RMST* is designed for task sets with $U_{tot} \le 0.5$, while *RMGT* is the extension proposed for generic task sets.

### *4.1.2 Time-demand-analysis-based tests*

Computationally more complex, but also more accurate tests can be provided by constructing the *critical instant phasing* [24], implicitly using the information about the worst-*case task execution times* and *periods.*

- **Time demand analysis (TDA),** developed by Lehoczky et al. [24], provides an exact characterization of schedulability under RMS. It provides a necessary and sufficient condition[2], but runs in pseudo-polynomial-time. The *time demand* function $w_i(t)$ of task $T_i$ is defined as:

$$w_i(t) = c_i + \sum_{k=1}^{i-1}\left\lceil \frac{t}{P_k} \right\rceil c_k \text{ , for } 0 < t \le P_i \quad (2)$$

where tasks are assumed to be sorted in non-increasing order according to the priorities. The task set is considered feasible if all the tasks can meet their deadlines under the critical-instant phasing; that is, if for *each* task $T_i$, it is possible to find a time instant $t$ where $w_i(t) \le t \le D_i$ . It is sufficient to check this condition at time instants that correspond to period boundaries [5, 24].

- **Pillai-Shin test (PS)** is a polynomial-time heuristic scheme ([31]), proposed for RMS on DVS-enabled settings, providing a sufficient condition for schedulability. It provides a sufficient but not necessary condition for feasibility by checking whether the time demand function $w_i(t)$ does not exceed the relative deadline $D_i$ at the task's period boundary $t = P_i = D_i$ for each task:

$$w_i(t = P_i) = \sum_{k=1}^{i}\left\lceil \frac{P_i}{P_k} \right\rceil c_k \le P_i \;\forall\; T_i \in T \quad (3)$$

where tasks are again sorted in non-increasing order according to the priorities.

| Test Name | $U_{tot}$ | $u_i$ | $P_i$ | $c_i$ | Complexity | Exact |
|---|---|---|---|---|---|---|
| *ELL* | √ | × | × | × | $O(n)$ | × |
| *Hyp* | × | √ | × | × | $O(n)$ | × |
| *R-Bound* | √ | × | √ | √ | $O(n)$ | × |
| *Burc* | √ | × | √ | × | $O(n)$ | × |
| *PS* | × | × | √ | √ | $O(n^2)$ | × |
| *TDA*[3] | × | × | √ | √ | $O(n^2 r)$ | √ |

**Table 1. RMS admission control algorithms used in this paper**

It is worth noting that all the RMS feasibility tests discussed above, except for *TDA*, are *inexact* in the sense that they provide a sufficient but not necessary condition for feasibility. Table 1 summarizes the task set parameters used by each feasibility test as well as its computational complexity on *one processor*, as a function of the number of tasks $n$.

## 4.2 Partitioning Heuristics

The second dimension of the design space involves the selection of the partitioning scheme. Since this problem is also intractable, several fast heuristics were developed over time. These include **First-Fit (FF), Best-Fit (BF), Worst-Fit (WF)** and **Next-Fit (NF)** [16, 29].

Note that WF and NF tend to distribute the workload evenly among the available processors, resulting in (more or less) balanced partitions. On the other hand, FF and BF attempt to greedily pack as many tasks as possible on one processor while keeping the other processors idle to accommodate tasks yet to be assigned, thus occasionally yielding "unbalanced" partitions. It is known that FF and BF tend to outperform NF

[1] A task set is harmonic if, given any two periods $P_i$ and $P_j$, either $P_i$ is a multiple of $P_j$ or $P_j$ is a multiple of $P_i$ .

[2] The feasibility condition provided by *TDA* is considered necessary and sufficient assuming that critical-instant phasing will actually occur at run-time.

[3] $r$ in the complexity expression is the ratio of the largest period to the smallest period: $r = max(P_i) / min(P_i)$.

and WF from the feasibility point of view [29]. The effect on the energy consumption will be evaluated in Section 5.

## 4.3 Speed Assignment Schemes

Once a feasible partition is obtained using a given partitioning heuristic and an admission control algorithm, the final step is to perform speed assignment to minimize the system energy consumption on each processor. Again, we classify these speed assignment heuristics into two groups based on the approach used.

### *4.3.1 Uniform slow-down technique*

Arguably, the simplest technique consists in computing a unique speed across all the tasks in such a way that the new *effective utilization* does not exceed the utilization bound suggested by the admission control algorithm. As such, it is easy to see that this technique can be used in conjunction with any utilization-based admission control algorithm. For example, suppose that, two tasks are assigned to processor $M_i$ with total utilization $U_i = 0.424$ in a feasible partition. If the Liu-Layland feasibility bound is used, then ($U_{bound}(n=2) = 0.848$), and the speed on processor $M_i$ is set to 0.424 / 0.848 = 0.5.

### *4.3.2 Time-demand-analysis-based speed assignment techniques*

More sophisticated speed assignment techniques are possible through an analysis of *critical-instant-phasing* at the cost of increased computational complexity.

- **Pillai-Shin speed assignment technique** [31] was proposed in conjunction with the corresponding Pillai-Shin feasibility test for RMS on DVS-enabled settings. A single speed is chosen for all tasks running on a given processor. For each task $T_i$, a tentative target speed $\alpha_i$ is determined from the following equality:

$$\sum_{k=1}^{i} \left\lceil \frac{P_i}{P_k} \right\rceil c_k = \alpha_i P_i \;\; \forall \; T_i \in T \qquad (4)$$

  Then, to ensure feasibility, the processor speed $S$ is set to the maximum among all tentative speeds (i.e. $S = max(\alpha_i)$). Observe that the speed $S$ on a processor with $n$ tasks can be computed in time $O(n^2)$.
- **Sys-Clock (SysC) and PM-Clock (PMC)**: Two pseudo-polynomial-time speed assignment schemes have been proposed in [33] for static-priority scheduling. Sys-Clock chooses a single speed for all tasks running on the same processor, and is optimal for fixed priority preemptive scheduling policies that use a single speed. PM-Clock, on the other hand, allows different tasks running on the same processor to have different speeds, but with added complexity. Both schemes use the idle time in a given schedule to reduce the execution speed, and hence decrease the energy consumption, while maintaining the feasibility. Sys-Clock and PM-Clock work naturally with the time demand analysis feasibility test.

# 5. Experimental results

In Section 4 we discussed our solution approach for energy minimization with partitioned Rate Monotonic Scheduling, which comprises three major components: partitioning, admission control, and processor speed assignment. Note that the large number of alternatives on each dimension yields a rather broad design spectrum. In this section we provide an experimental evaluation of these heuristics.

**Performance metrics**. Our problem has two equally important performance dimensions: *feasibility* and *energy*. Given a task set to be scheduled on a multiprocessor platform, our goal is to select an algorithm for each dimension, with high feasibility performance, low energy consumption, and low computational cost. As we will see shortly, there is an inherent tradeoff between feasibility and energy performances of the schemes we investigated. Hence, judging by the feasibility and energy it is not always possible to point to a "clear winner". Consequently, we propose an additional hybrid metric (called *feasibility/energy*) that combines both performance dimensions. Observe that the hybrid metric captures the aim of designing a heuristic with high feasibility performance and low energy consumption. Heuristics will be compared based on feasibility, energy, and feasibility/energy metrics. In summary, we measure the performance of a given heuristic $H$ in terms of three metrics:

- The feasibility metric (*FH*): the *percentage* of task sets that are feasibly scheduled by $H$ out of the total number of task sets generated during the experiments.
- The energy consumption metric (*ECH*): the *average* energy consumption for each task set scheduled by $H$ in a feasible manner (in other words, the energy consumption of an infeasible partition is not taken into account, however low it may be).
- The feasibility/energy metric (*FEH*): defined as *FH* / *ECH*. This metric favors heuristics with high feasibility performance and low energy consumption.

**Simulation settings**. We measured each of the performance metrics discussed above as a function of two task set parameters: the total task set utilization $U_{tot}$ and the task utilization factor $\alpha$ (see Section 2.2). For a fixed number of processors $m$, we varied $U_{tot}$ between $m$ / 10 (light load condition) and $m$ (heavy load condition). There is a natural constraint on the possible values of $\alpha$: $U_{tot} / n \le \alpha \le 1.0$, must hold for any task set. Note that the case of $\alpha = 1.0$ corresponds to having no constraint on (or knowledge of) upper bounds on individual task utilization. The parameter $\alpha$ was varied between $U_{tot} / n$ and 1.0. For each data point, we generated 1000 task sets by varying task periods $P_i$ and utilizations $u_i$. Each task has a uniform probability of having short (1-10ms), medium (10-100ms), or long (100-1000ms) period. Task periods are uniformly distributed in each range. Note that the same period generation scheme is used in [31, 33]. Task utilizations $u_i$ are generated uniformly in the interval $[0.001, \alpha]$ while making sure that $\sum_i u_i = U_{tot}$. A task's worst-case execution time at maximum speed is then determined as $c_i = P_i \, u_i$. We experimented with different number of processors and tasks, but due to space limitation we only show results for 8 processors and 80 tasks. However, the trends we report hold in other settings as well. The results we present are obtained by assuming a quadratic energy-speed relationship; however, in Section 5.3, we comment on results obtained through the specifications of an *actual* DVS architecture, namely Intel XScale [17].

## 5.1 Results for off-line partitioning

It is known that, if all the task characteristics are known in advance, then pre-ordering tasks according to a well-chosen task parameter helps to improve the feasibility performance. In general, ordering tasks according to *utilization* values (in non-increasing order) can have significant impact on the performance [29]. This applies to all the feasibility tests we considered in Section 4.1 except for RMGT and R-BOUND that are designed to order tasks according to period values (otherwise their feasibility performance deteriorates significantly).

**Effect of partitioning schemes**. We start by investigating the effect of partitioning schemes on system performance. In fact, much of the performance differences among the partitioning heuristics, in terms of both feasibility and energy, can be explained in terms of their load-balancing behavior. We already discussed in Section 4.2 that FF and BF tend to yield "unbalanced" partitions, while WF and NF tend to produce balanced ones. This different load-balancing behavior leads to different feasibility and energy characteristics. On the other hand, by greedily packing as many tasks as possible on a few processors (just in the case of FF and BF), it is possible to accommodate additional tasks on the remaining (idle) processors. This greedy behavior improves the feasibility, a result from the study of the *bin-packing* problem [16]. From the energy perspective, however, load balancing tends to reduce the energy consumption in general, although it does not necessarily lead to the minimum energy when RMS is used. In a balanced partition the load is divided among the different processors and thus it is possible to use DVS to lower the speed on each processor, reducing the total energy of the system. In an "unbalanced" partition, however, some of the processors are heavily loaded while the remaining ones are lightly loaded. Thus, the heavily loaded processors will have to run at a speed which is close to the maximum to guarantee the feasibility of the task set, which significantly increases the total energy of the system due to the convex relationship between the speed and the power consumption.

It is worth noting, that for off-line partitioning, FF and BF are almost identical in terms of feasibility and energy performance. This behavior has also been observed in [29] for the feasibility aspect. Since BF has a higher computational complexity than FF, it is not included in our discussion of off-line partitioning.

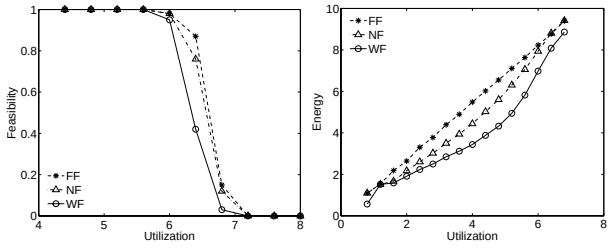



**Figure 4. Feasibility and energy performance of offline partitioning heuristics using ELL for $\alpha = 1.0$**

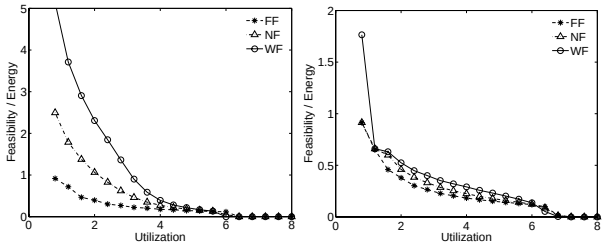



**Figure 5. Feasibility/energy performance of offline partitioning heuristics using ELL: $\alpha = 0.5$ (left) and $\alpha = 1.0$ (right)**

Figure 4-5 show the performance in terms of each of the three metrics as a function of total utilization $U_{tot}$ for the partitioning heuristics discussed in Section 4.2 when used with Liu-Layland (ELL) schedulability bound and uniform slow-down technique. Figure 4 (left) shows the feasibility performance for $\alpha = 1.0$. Under low to medium load conditions feasibility can be easily achieved and all heuristics yield 100% feasibility. As utilization increases and the system becomes heavily loaded, the feasibility performance of all schemes drops sharply, and eventually it becomes practically zero (when $U_{tot}$ exceeds 7). Our experiments confirmed the good performance of FF as reported in previous studies [29]. Because of its load-balancing behavior, the feasibility performance of WF is not the best, but it is comparable to the performance of FF even at high utilizations and $\alpha$ values. NF, on the other hand, has a feasibility performance which lies between FF and WF.

In Figure 4 (right), we present the energy consumption with different partitioning techniques for $\alpha = 1.0$. We show the energy consumption as a function of the utilization for $U_{tot}$ up to 7, since the feasibility beyond that threshold is practically zero, making the energy performance undefined. We follow the same approach when presenting other energy performance results throughout the paper. The system energy consumption increases with the increase of utilization, since this implies an increase in CPU speed with additional power dissipation. It is clear that the different heuristics exhibit significantly different energy characteristics especially under medium load conditions. FF is the worst heuristic in terms of energy since it tends to generate very unbalanced partitions. On the other extreme, WF has the lowest energy consumption, because of its load-balancing behavior. NF's energy performance lies in-between.

Since there is a trade-off between the feasibility and energy performances exhibited by the partitioning schemes, it is not possible to determine the best performing scheme just by considering feasibility and energy separately. The hybrid *feasibility/energy* metric provides a better picture. Figure 5 shows the feasibility/energy performance $FE_H$ for $\alpha = 0.5$ (left) and $\alpha = 1.0$ (right). Recall that $FE_H$ is defined as $F_H / E_H$, and that $F_H$ decreases while the utilization increases (Figure 4). Hence, as expected, the hybrid metric decreases quickly with the increase in the utilization. Moreover, WF is the best heuristic in terms of overall performance, judging by its feasibility/energy performance. This follows from the fact that WF has much lower energy consumption than the other schemes while its feasibility is comparable to that of the other schemes. By the same argument, FF is the worst heuristic in terms of overall performance due to its high energy consumption. Increasing $\alpha$ significantly reduces the feasibility/energy performance especially for WF and NF, since by increasing the task

utilization factor the partitions tend to become less balanced and the energy consumption is affected (note the *scale* difference on the *y*-axis for the right and left plots of Figure 5). In summary, we conclude that, for off-line partitioning the best partitioning heuristic is Worst-Fit, followed by Next-Fit, then First-Fit. The analysis of the relative performance of partitioning heuristics presented above applies not only to Liu-Layland feasibility test and the uniform slow-down technique but to other alternatives as well.

**Effect of admission control and speed assignment schemes**. To investigate the effect of admission control and speed assignment schemes, we adopted the following methodology. Since it is not reasonable to compare techniques resulting from all possible combinations, we matched each of the admission control schemes presented in Table 1 with one speed assignment technique. Specifically, all utilization-based approaches of Table 1 (namely, ELL, HYP, BURC, and R-BOUND) are matched with the *uniform-slowdown* technique. Pillai-Shin slow-down technique uses its own polynomial-time feasibility test (PS) as proposed in [31]. Finally, SYS-Clock and PM-Clock algorithms use the exact time-demand-analysis technique as part of their operation [33][4]. We believe that this approach is justified, since matching an involved TDA-based speed assignment technique with relatively simple utilization-based admission control algorithm would make little sense from the implementation or overall complexity point of view. We also implemented and evaluated RMST and RMGT algorithms [13], but we exclude them from our detailed analysis, since they were consistently outperformed by a related algorithm, BURC, proposed by the same authors. Finally, the results we report are obtained by assuming Worst-Fit partitioning heuristic, which is shown to outperform Next-Fit, First-Fit and Best-Fit above. The only exception is R-BOUND, whose extension to multiprocessor settings (namely, RBOUND-MP) was proposed and justified with the First-Fit heuristic. Since our experiments with R-BOUND and Worst-Fit yielded extremely poor performance, we used First-Fit as it was originally proposed in [23].

From the feasibility viewpoint (Figure 6 - left), the scheme TDA-SYSC, which uses the sophisticated time-demand analysis and Sys-Clock algorithm, is a clear winner. However, it runs in pseudo-polynomial time. The feasibility performance of R-BOUND is comparable to that of TDA-SYSC even at high utilization values, despite its lower (linear-time) complexity. The hyperbolic test HYP has average performance, while the remaining techniques (PS, BURC, and ELL) trail the list. It is interesting to note that the simple technique HYP outperforms relatively more involved techniques such as BURC and PS.

In terms of the energy consumption (Figure 6, right), TDA-SYSC is again the best scheme, this time followed by PS and HYP. Observe that R-BOUND has the worst energy performance among all schemes, except at high utilization values. R-BOUND uses FF for partitioning, thus yielding highly unbalanced partitions especially for low to medium loads with corresponding high energy consumption levels. However, for heavy loads the partitions produced by R-BOUND tend to be more balanced, since in this case all processors have to get a share of the workload to maintain feasibility. Again, note the good performance of the simple technique HYP.

Figure 7 shows the feasibility/energy performance for $\alpha$ = 0.5 (left) and $\alpha$ = 1.0 (right). As expected, TDA-SYSC has the best overall performance. Among the polynomial-time schemes HYP has the best overall performance. PS, BURC, and ELL have lower overall performance at high utilization values. R-BOUND has the worst overall performance at low to medium utilization values, but its performance is comparable to that of TDA-SYSC under heavy load conditions, thanks to its remarkable feasibility performance. Based on these observations, **we can suggest the use of HYP at light to medium load values, and that of R-BOUND at heavy loads** if the overhead associated with TDA-SYSC cannot be afforded.

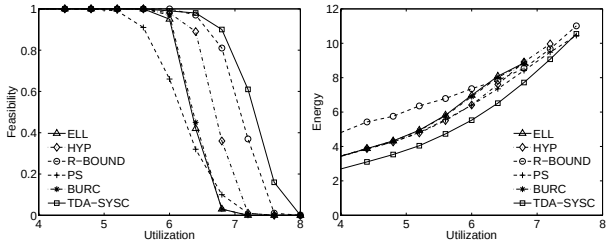



**Figure 6. Feasibility and energy performance of the different techniques (off-line partitioning, $\alpha$= 1.0)**

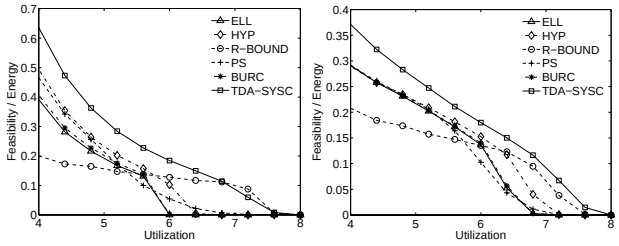



**Figure 7. Feasibility/energy performance of the different techniques (off-line partitioning, $\alpha$= 0.5 (left) and $\alpha$= 1.0 (right))**

## 5.2 Results for on-line partitioning

If the task parameters are not known in advance, then the scheduler will not be able to pre-order tasks before the task allocation phase. This can be the case for *on-line* settings where the scheduler has to assign tasks as they arrive dynamically to the system without seeing the rest of the task sequence. By examining the performance of traditional partitioning heuristics FF, BF, NF, and WF, we notice that there is no clear winner in terms of overall performance throughout the utilization spectrum. FF, BF, and NF have good feasibility performance, but they have poor energy and feasibility/energy performance, particularly at low utilization values. WF, on the other hand, has good energy performance, but it has very low feasibility and feasibility/energy performance, especially at medium to high utilization.

To overcome these shortcomings, we propose a partitioning algorithm called RESERVATION. The algorithm is based on the observation that the poor performance of WF in on-line settings is often due to distributing tasks with small utilization values to multiple processors, preventing the allocation of a subsequent task with large utilization to a separate processor at the end. Thus, the algorithm *reserves* a pool of *k* processors ($k \leq$

[4] Although computationally more complex and sophisticated, the energy performance of PM-Clock did not show any clear advantage over Sys-Clock, therefore we show only the results of Sys-Clock in this section. This observation is consistent with the findings of the original paper [33].

$m$) for *light* tasks, and the remaining $m - k$ processors, for *heavy* tasks.

A task $T_i$ is said to be *light* if its utilization $u_i$ does not exceed the average utilization per processor (if $u_i \leq U_{tot} / m$). Otherwise, it is said to be *heavy*. When presented a task $T_i$, the algorithm first determines if it is light or heavy and tries to assign it to a processor *from the corresponding pool using Worst-Fit*. If none of the processors in the corresponding ("preferred") pool can accommodate it, then an attempt is made to assign it to a processor from the other pool, again using Worst-Fit. Note that RESERVATION($k$) is a family of schemes parameterized by $k$; for each $k$, the algorithm's initial processor reservation for light tasks changes. Observe that in the extreme cases of $k = 0$ or $k = m$, the algorithm reduces to Worst-Fit.

**Effect of partitioning schemes**. RESERVATION($k$) is effectively an attempt to maintain a balance between the good feasibility performance of First-Fit/Best-Fit and the good energy performance of Worst-Fit. In this section we investigate the effect of the parameter $k$ on the performance of RESERVATION algorithm and compare it to the traditional partitioning heuristics. For the sake of comparison, we combine each partitioning heuristic with Liu-Layland feasibility test (ELL), but we underline that the patterns we observe hold also for all admission control algorithms. In Figure 8 and 9, we compare the performance of three RESERVATION schemes, with $k$ set to 2, 4, and 6 (denoted by RSRV2, RSRV4, and RSRV6 respectively), against FF, BF, and WF. NF's performance is consistently worse than that of RSRV2 and RSR4, thus, it is omitted from the discussion. Also recall that RSRV0 and RSRV8 are not included in the analysis, because both of them are effectively identical to WF.

Figure 8 (left) shows that FF and BF still have the best feasibility performance for on-line partitioning. However, WF in this case has very low feasibility performance especially at high utilization and $\alpha$ values. Among all the RESERVATION schemes (including the ones not shown in the figures), RSRV2 has the best feasibility for $\alpha = 1.0$ and is in fact comparable to FF and BF. It is worth noting, however, that the choice of a RESERVATION scheme with the best feasibility performance depends of the value of $\alpha$ for the task set at hand. For example, when $\alpha = 0.5$ the scheme is RESERVATION($k = 4$). Due to space limitations, we are providing feasibility results corresponding to $\alpha = 1.0$.

From the energy viewpoint, Figure 8 (right) points to the high energy consumption of BF, that tends to yield unbalanced partitions. However, it is not possible in this case to name a single winner scheme throughout the utilization spectrum. At low to medium utilization values WF has the lowest energy consumption, while RSRV2 is the winner at heavy loads.

Figure 9 shows the feasibility/energy performance for $\alpha = 0.5$ (left) and $\alpha = 1.0$ (right). Once again, there is no clear winner throughout the utilization spectrum. For $\alpha = 1.0$, under heavy load conditions FF is the best, while RSRV4 is the best under light to medium loads. However, RSRV2 has consistently good overall performance and is comparable to the best scheme under each load condition. For $\alpha = 0.5$, on the other hand, RSRV4 provides the best overall performance, and is slightly outperformed by FF and BF in a small region observed at heavy loads.

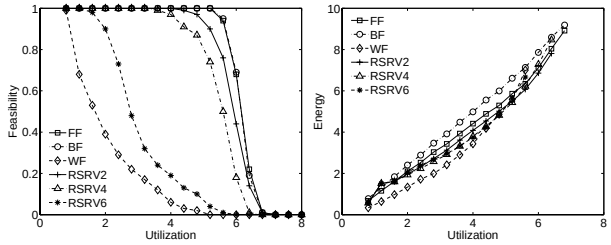



**Figure 8. Feasibility and energy performance of on-line partitioning heuristics using ELL for $\alpha = 1.0$**

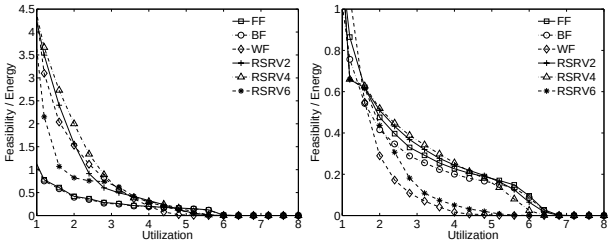



**Figure 9. Feasibility/energy performance of on-line partitioning heuristics using ELL: $\alpha = 0.5$ (left) and $\alpha = 1.0$ (right)**

**Effect of admission control and speed assignment schemes**. We adopt the RESERVATION(k = 2) scheme for online settings based on its consistently good performance at different load conditions and α values. However, we underline that the patterns we report are valid for other partitioning schemes as well. Figure 10 (left) compares the different feasibility tests used in the case of on-line partitioning in terms of feasibility performance. Note that the R-BOUND is excluded from the evaluation in on-line settings, since it mandates the knowledge of the periods and task pre-ordering as part of its operation. TDA-SYSC still exhibits the best feasibility performance, but the second best scheme appears to be PS. HYP still provides consistently good feasibility performance.

In terms of energy consumption, there is no clear winner throughout the utilization and α spectrum. For α = 1.0 (Figure 10 - right), TDA-SYSC is still the best scheme. The best polynomial-time scheme is PS followed by HYP. However, for smaller α values, such as α = 0.5, TDA- SYSC still appears to be the best scheme at low and high utilizations, but its performance deteriorates at medium utilizations and its energy consumption slightly exceeds all other schemes in a small region. In this limited region, PS has the best overall energy consumption. Due to space limitation we are showing energy results for α = 1.0 only.

In terms of the hybrid performance metric, TDA-SYSC, with its sophisticated mechanism is the clear winner throughout the utilization and $\alpha$ spectra (Figure 11). Note that the simple scheme PS yields strikingly good performance among the remaining schemes, and its advantage over others becomes even more emphasized at small $\alpha$ values. The performance of HYP approaches that of PS only at large $\alpha$ and utilization values.

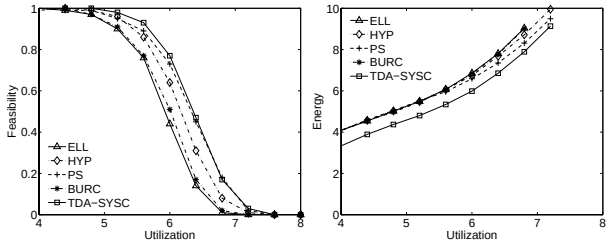



**Figure 10. Feasibility and energy performance of different techniques (on-line partitioning, $\alpha = 1.0$)**

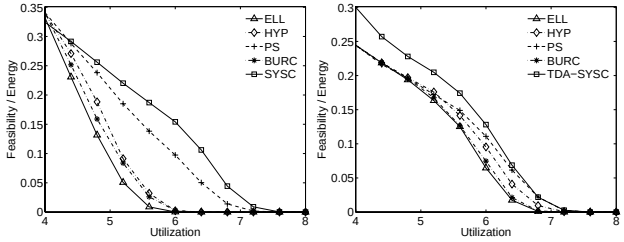



**Figure 11. Feasibility/energy performance of different techniques (on-line partitioning, $\alpha = 0.5$ (left), $\alpha = 1.0$ (right))**

## 5.3 Effect of discrete speed levels

In this section we investigate the effect of having a finite number of discrete speeds on the relative performance of the schemes presented in this paper. To this aim, we incorporated the Intel XScale speed/power specifications [17] in our simulator. Note that the speed assignment takes place after the partitioning phase, thus, only the energy and feasibility/energy metrics are affected by the change in speed levels. Due to space limitations, we only present results for feasibility/energy performance of different techniques for $\alpha = 1.0$ for both off-line and on-line partitioning in Figure. There is a slight decrease in the feasibility/energy performance of all schemes. In the presence of a finite number of discrete speeds, one often has to select an existing (but, higher) CPU speed level, leading to an in increase in total energy consumption. Subsequently, this results in an overall decrease of the feasibility/energy performance. However, the relative performance of the different feasibility tests is unchanged compared to the continuous speed case. We underline that this conclusion applies to all the schemes we discussed above and for the different settings considered. Moreover, the feasibility/energy performance of the off-line schemes (Figure 12 - left) is still significantly better than the on-line ones (Figure 12 - right), since the feasibility performance deteriorates in on-line settings (again, note the scale difference for the *y* axis in both figures).

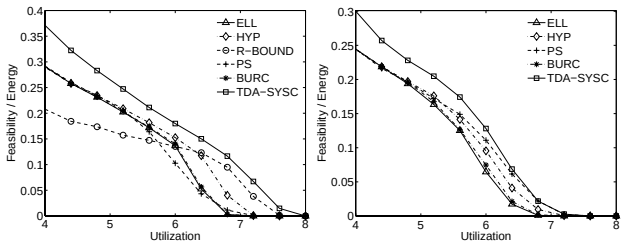



**Figure 12. Feasibility/energy performance of the different techniques for $\alpha = 1.0$ and Intel XScale specifications (off-line case (left) and on-line case (right)).**

# 6. Conclusion

To the best of our knowledge, this is the first research effort addressing the energy-aware scheduling of static-priority periodic RT task sets on multiprocessors with partitioned approach. We showed that the problem is NP-Hard in the strong sense on m ≥ 2 processors even when the feasibility is guaranteed a priori. We considered two different settings of the problem: off-line partitioning, where all the tasks and their characteristics are known to the scheduler, and on-line partitioning, where the scheduler has to make the task allocation decisions in a given order. We evaluated experimentally the impact of the partitioning heuristics, admission control algorithms, and speed assignment schemes on both feasibility and energy performances. To better capture these two dimensions, we introduced a hybrid metric defined as feasibility/energy. Our experiments show that the admission control based on the Time Demand Analysis when combined with Sys-Clock speed assignment scheme has the best overall performance in both off-line and on-line settings, at the cost of pseudo-polynomial time complexity. Moreover, in off-line settings Worst-Fit has the best overall performance among partitioning heuristics. In these settings, the admission control with the Hyperbolic test combined with a uniform slow-down approach for speed assignment has the best overall performance among polynomial-time schemes. In on-line settings, the performance of Worst-Fit deteriorates significantly. This led us to introduce the RESERVATION(*k*) scheme which exhibits a competitive overall performance. Admission control and speed assignment based on Pillai-Shin's technique [31] yields the best overall performance among polynomial-time schemes in these settings.

# References

[1] T.A. AlEnawy and H. Aydin. On energy-constrained real-time scheduling. *Proceedings of the 16th EuroMicro Conference on Real-Time Systems (ECRTS'04)*, June 2004.

[2] B. Andersson, S. Baruah, and J. Jonsson. Static-priority scheduling on multiprocessors. *Proceedings of the IEEE International Real-Time Systems Symposium*, pp 193-202, 2001.

[3] B. Andersson and J. Jonsson. The utilization bounds of partitioned and pfair static-priority scheduling on multiprocessors are 50%. *Proceedings of the 15th Euromicro Conference on Real-Time Systems*, pp. 33-40, 2003.

[4] J. Anderson, P. Holman, and A. Srinivasan. Fair multiprocessor scheduling. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, Joseph Y. Leung (ed.), Chapman and Hall/CRC, Boca Raton, Florida, pages 31.1–31.21, 2004.

[5] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A.J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, Volume: 8 Issue: 5, Sept. 1993.

[6] H. Aydin, R. Melhem, D. Mossé and P.M. Alvarez. Dynamic and aggressive scheduling techniques for power-aware real-time systems. *Proceedings of the Real-Time Systems Symposium (RTSS'01)*, pages 95-105, 2001.

[7] H. Aydin, R. Melhem, D. Mossé and P.M. Alvarez. Power-aware scheduling for periodic real-time tasks. *IEEE Transactions on Computers*, vol 53 (5), pp. 584-600, May 2004.

[8] H. Aydin and Q. Yang. Energy-aware partitioning for multiprocessor real-time systems. Proceedings of the 17th International Parallel and Distributed Processing Symposium (IPDPS'03), Workshop on Parallel and Distributed Real-Time Systems, 2003. Available online at http://cs.gmu.edu/~aydin/ipdps03.ps

[9] T.P. Baker. Multiprocessor EDF and deadline monotonic schedulability analysis. *Proceedings of the 24th IEEE International Real-Time Systems Symposium*, 2003.

[10] S. Baruah and J. Anderson. Energy-efficient synthesis of periodic task systems upon identical multiprocessor platforms. *Proceedings of the 24th IEEE International Conference on Distributed Computing Systems*, March 2004**.**

[11] E. Bini, G.C. Buttazzo, and G. Buttazzo. A hyperbolic bound for the rate monotonic algorithm. *Proceedings of the 13th Euromicro Conference on Real-Time Systems*, 2001.

[12] E. Bini, G.C. Buttazzo, and G. Buttazzo. Rate monotonic analysis: the hyperbolic bound. *IEEE Transactions on Computers*, Volume: 52 Issue: 7, July 2003.

[13] A. Burchard, J. Liebeherr, Y. Oh, and S.H. Son. New strategies for assigning real-time tasks to multiprocessor systems. *IEEE Transactions on Computers*, Volume: 44 Issue: 12, Dec. 1995.

[14] G. Buttazzo. Rate monotonic vs. EDF: judgment day. Proceedings of the 3rd ACM International Conference on Embedded Software (EMSOFT 2003), 2003.

[15] J. Carpenter, S. Funk, P. Holman, and A. Srinivasan, J. Anderson, and S. Baruah. A categorization of real-time multiprocessor scheduling problems and algorithms. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, Joseph Y. Leung (ed.), Chapman and Hall/CRC, Boca Raton, Florida, pp. 30.1–30.19, 2004.

[16] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Approximation Algorithms for Bin Packing: A Survey. In *Approximation Algorithms for NP-Hard Problems*, PWS Publishing, Boston (1997).

[17] http:// developer.intel.com/design/intelxscale/benchmarks.htm

[18] S.K. Dhall and C.L. Liu. On a real-time scheduling problem. *Operations Research*, 26(1):127–140, 1978.

[19] M. Elnozahy, M. Kistler, and R. Rajamony. Energy Conservation Policies for Web Servers. *Fourth USENIX Symposium on Internet Technologies and Systems*, 2003.

[20] M.R. Garey and D.S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freman, NewYork, 1979.

[21] J. Goossens, S. Funk, and S. Baruah. Priority-driven scheduling of periodic task systems on multiprocessors. *Real-time Systems* 25(2-3), pp. 187-205. 2003.

[22] T.K. Kuo and A.K. Mok. Load adjustment in adaptive real-time systems. *Proceedings of the IEEE Real-Time Systems Symposium*, 1991.

[23] S. Lauzac, R. Melhem, and D. Mosse. An efficient RMS admission control algorithm and its application to multiprocessor scheduling. *Proceedings of Parallel Processing Symposium*, pp. 511-518, 1998.

[24] J.P. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: exact characterization and average case behavior. *Proceedings of the IEEE Real Time Systems Symposium*, pp. 166 -171, 1989.

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

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

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

[28] Y. Liu and A. Mok. An Integrated Approach for Applying Dynamic Voltage Scaling to Hard Real-Time Systems. *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'03)*, 2003.

[29] J.M. Lopez, M. Garcia, J.L. Diaz, and D.F. Garcia. Utilization bounds for multiprocessor rate-monotonic scheduling. *Real-Time Systems* 24(1): 5-28 (2003).

[30] J.R. Lorch and A.J. Smith. Energy consumption of Apple Macintosh computers. *IEEE Micro*, 18(6), November/December 1998.

[31] P. Pillai and K.G. Shin. Real-time dynamic voltage scaling for low power embedded operating systems. *Symposium on Operating Systems Principles*, 2001.

[32] A. Qadi, S. Goddard, and S. Farritor. A Dynamic Voltage Scaling Algorithm for Sporadic Tasks. *Proceedings of the IEEE Real-Time Systems Symposium*, 2003.

[33] S. Saewong and R. Rajkumar. Practical Voltage-Scaling for Fixed-Priority Real-time Systems. *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'03)*, May 2003.

[34] Y. Yu and V.K. Prasanna. Power-aware resource allocation for independent tasks in heterogeneous real-time systems. *Proceedings of 9th International Conference on Parallel and Distributed Systems*, 2002.

[35] D. Zhu, N. AbouGhazaleh, D. Mossé, and R. Melhem. Power aware scheduling for AND/OR graphs in multi-processor real-time systems. *Proceedings of the International Conference on Parallel Processing* (*ICPP'02*), 2002.

[36] D. Zhu, R. Melhem, and B. Childers. Scheduling with dynamic voltage/speed adjustment using slack reclamation in multi-processor real-time systems. *Proceedings of the 22nd IEEE Real-Time Systems Symposium*, 2001.