IJCSES International Journal of Computer Sciences and Engineering Systems, Vol.1, No.3, July 2007
CSES International ©2007 ISSN 0973-4406

Enhanced Rate Monotonic Time Demand Analysis

Nasro MIN-ALLAH[1,3], Yong-Ji WANG[2], and Jian-Sheng XING[1]

[1]Laboratory for Internet Software Technologies, Institute of Software,
Graduate University, Chinese Academy of Sciences, Beijing 100080, China
E-mail: nasar, jiansheng@itechs.iscas.ac.cn

[2]Key Laboratory of Computer Science, Chinese Academy of Sciences, Beijing 100080, China
E-mail: ywang@itechs.iscas.ac.cn

[3]COMSATS Institute of Information Technology,
Department of Computer Science, Islamabad 44000, Pakistan
E-mail: nasar@comsats.edu.pk

Abstract

Latest real-time systems are capable of changing its speed at run time to prolong its battery life and, demand efficient techniques for online decisions making. Available real time feasibility tests have pseudo-polynomial complexity and can not be executed online. This paper proposes that the time complexity of existing techniques could be lowered when task schedulability is analyzed at points, where task feasibility is expected to be true. This paper also identifies False Points (FP) in feasibility analysis. By propagating false points, our technique greatly reduces the number of points that would be tested with existing approaches. We have extending the Time Demand Analysis under Rate Monotonic (RM) policy for periodic tasks. Our proposed technique can be executed online for guaranteeing task feasibility. Mathematical formulation and simulation results show the significance of our technique.

Key words: *Real Time Systems, Fixed priority Scheduling, Rate Monotonic Feasibility Analysis, Time Demand Approach*

1. Introduction

On one hand, current devices offer high computation power at the expense of more energy consumption, while on the other, the gap between energy demand and improvements in battery capacity is widening day by day. Although, the law of Gordon Moore [1] is maintained for micropcessor improvements, increase in battery capacity has only tripled since 1990[2]. In CMOS circuitry dynamic power consumption is directly related to the voltage/speed [3, 4, 5] i.e. the more is operating speed, the higher is the power consumption of the system. A number of solutions such as Dynamic Power Management (DMP) and Dynamic Voltage Scaling (DVS) are recently proposed to lower the total energy consumption of the power hungry devices. Among these, Dynamic Voltage Scaling (DVS) is the cutting edge technique to adjust system speed on the fly for reducing the power consumption.

Real-time systems are promising target for DVS because such systems are usually under utilized. As DVS is exploited at run time, more efficient online solutions are needed to lower the run time overheads (space and time complexity) associated with current techniques. This paper is an attempt to lower the complexity of the feasibility test for hard periodic tasks, which can be deployed to DVS-enabled systems so that power consumption could be reduced up to maximum extent.

For real time systems, to maintain system reliability, feasibility tests are required at run time. However, the available tests are too complex to be applied online. In this paper we discuss the feasibility of periodic tasks, driven by priority driven algorithms which fall into two types: fixed priority and dynamic priority. A fixed-priority algorithm assigns the same priority to all jobs in each task. In contrast, a dynamic priority algorithm assigns different priorities to the individual jobs in each task [6]. Although, dynamic algorithms promise higher system utilization as compared to fixed priority algorithms, they become unpredictable, when transient overload occurs, while real time system must provide predictable response times. Therefore, this requirement makes the worst case behavior of real time systems more important than the average response time or user conveniences, which are important issues for general purpose computing systems. Due to reliability and simplicity, today, among fixed priority scheduling algorithms, Rate Monotonic (RM) scheduling has become the *de fecto* standard for real time systems and

Manuscript received March 1, 2007
Manuscript revised June 1, 2007

adopted by Boeing, General Dynamics, General Electric, Honeywell, IMB and NASA etc [7].

To fulfill timing constraints of periodic tasks, feasibility tests are performed over the entire task set to answer, whether the task set is RM schedulable. The first feasibility test for RM is proposed by Liu and Layland [8], called LL -bound. A recent attempt is made by Bini et.al. [9] that further improved the LL-bound. Both tests offer polynomial time complexity and can be performed for on-line guarantee of periodic task schedulability, however, they are sufficient conditions only and, do impose a bound on system utilization.

To overcome the utilization issue, Lehoczky, Sha and Ding provided an exact RM test that is both sufficient and necessary in [10]. Their work is extended by authors in [11, 12] and proposed tests based on response time analysis. All [10, 11, 12] are necessary and sufficient conditions but their time complexity is very high and again impede these tests to be performed online. In this paper we present a novel technique to tackle the complexity of time demand analysis [10] by reducing the number of points to be checked for analyzing task schedulability. Our technique first observe false points in current task feasibility and then avoid testing such points for lower priority task. A formal formulation is given and simulation results show the effectiveness of our algorithm, while no compromise is made on feasibility analysis.

The rest of the paper is organized as follows. Section 2 describes the task model and states our notations. Section3 explains the pervious work and apply results to an example task set. We extend the pervious work [10] in Section 4. Experimental results are discussed in Section 5, followed by conclusion and future work in Section 6.

2. Task Model and Assumptions

Let $\tau = \tau_1, \tau_2, ..., \tau_n$ represents a set of n hard periodic tasks. They are preemptive and non-idling (processor can not be left inactive if there exists some pending tasks).
In our model of a hard real-time tasks, each task τ_i generates a job at each integer multiple of p_i and each such job has an execution requirement of c_i time units that must be completed by the next integer multiple of τ_i. Task set is synchronous and all tasks initially arrive at $t = o$ (critical instant [8]). All tasks are independent: requests for any are not dependent upon the initiation or completion of any other task. Tasks can not be blocked and worst case execution time (WCET) is always in accordance with actual execution time (ACET). Furthermore, the overhead due to context switching, scheduling etc is included in the WCET of tasks.

The response time of the jobs are smaller than or equal to their respective periods. i.e. at most one instant/job of a task exit at any particular time. To determine weather a task can meet all its deadlines, we compute the total demand for processor time by task τ_i and check whether this demand can be met before its deadline. For verifying, timing constraints, feasibility tests are performed over the entire task set τ to make it RM schedulable. RM assigns static priorities on task activation rates (periods) such that for any two tasks τ_i and τ_j, Priority $(\tau_i) >$ Priority $(\tau_j) \Rightarrow$ Period $(\tau_j) >$ Period (τ_i), while ties are broken arbitrary. Determining whether an arbitrary periodic task system is feasible has been shown to be intractable - co-NP complete in the strong sense [13].

Based on above assumptions we describe a pseudo-polynomial time schedulability test developed in [10] for tasks scheduled according to a RM priority algorithm. We need the following additional notations for later convenience. The cumulative processor demand by $\tau_i, 1 < i \le n$, during the time interval $[0, P_i]$ is

$$W_i(t) = \sum_{j=1}^{n} c_j \left\lceil t / P_j \right\rceil \quad (1)$$

$$L_i(t) = W_i(t) / t \quad (2)$$

$$L_i = \min_{0<t\le p_i}(L_i) \quad (3)$$

$$L = \max_{1\le i\le n}\{L_i\} \quad (4)$$

3. Time Demand Analysis

Authors in [10], provide an exact formula to determine if a given set of periodic tasks can meet their deadlines when the Rate Monotonic algorithm is used. To carry out the time demand analysis on τ, we consider one task at a time, starting from the τ_i with the highest priority in decreasing priority order. The necessary and sufficient condition for the task schedulability as given by

Theorem 1 ([10])
Given a set of n periodic tasks $\{\tau_1, \tau_2, ..., \tau_n\}$, τ_i can be feasibly scheduled for all tasks phasing using RM iff $L_i \le 1$

The test is known as time-demand analysis [6]. Assuming the task phasings are all zero, task τ_i is feasible if we find some $'t'$ satisfying the above condition. In other words, task τ_i completes its computation requirements at time $t \in [0, P_i]$, if and only if the entire request from the higher priority tasks and computation time of τ_i, are completed at time t. As t is a continuous variable, there are infinite numbers of points to be tested. Authors in [10] noted that $W_i(t)$ is constant except at points, when task are released, called rate monotonic scheduling points. Consequently, to determine if τ_i is schedulable, we need to compute $W_i(t)$, only at multiples of $\tau_j < \tau_i, 1 \le j \le i$. Specifically, let

$$S_i = \{lP_j \mid j = 1,...,i; l = 1,...,\lfloor P_i / P_j \rfloor\} \quad (5)$$

Theorem 2 ([10])
Given a set of n periodic tasks, $\{\tau_1, \tau_2, ..., \tau_n\}$, τ_i can be feasibly scheduled for all tasks phasing using RM iff

$$L_i = \min_{t \in S_i} \{W_i(t) / t \le 1\} \quad (6)$$

It can be seen that set S_i has a finite number of points and limits the number of inequalities to be tested. We represent this test by Time Demand Analysis (TDA) in rest of the paper. We now apply results parented in Theorem.1 to a set of three tasks with parameters given in Table.1

Table: 1 A 3-task set example

Task	Execution Requirements	Time Period	RM-Scheduling Points
τ_1	c_1	3	3
τ_2	c_2	7	3,6,7
τ_3	c_3	20	3,6,7,9,12,14,15,18,20

In algebraic terms, we have:

Task τ_1 is RM-schedulable *iff*

$c_1 \le 3$

Task τ_2 is RM-schedulable *iff*

$c_1+c_2 \le 3$ or
$c_1+c_2 \le 6$ or
$3c_1+c_2 \le 7$

Task τ_3 is RM-schedulable *iff*

$c_1+c_2+c_3 \le 3$ or
$2c_1+c_2+c_3 \le 6$ or
$2c_1+c_2+c_3 \le 7$ or
$3c_1+2c_2+c_3 \le 9$ or
$4c_1+2c_2+c_3 \le 10$ or
$4c_1+2c_2+c_3 \le 12$ or
$4c_1+2c_2+c_3 \le 14$ or
$5c_1+3c_2+c_3 \le 15$ or
$6c_1+3c_2+c_3 \le 18$ or
$7c_1+4c_2+c_3 \le 20$

List. 1 Redundant list of points

4. Enhanced Rate Monotonic Analysis

From List 1, we obtain the following major points and propose solutions accordingly:

4.1 Complexity

The time complexity of Theorem 2 is $O(nq_{n,1})$, where $q_{n,1}$ denote the period ratio, i.e. the ratio of largest period p_n to the smallest period p_1 .For large task set, the number of points to be tested is huge and is equal to the sum of elements in S_i .The number of elements in S_i is high when $q_{n,1}$ is large. For any task τ_i, inequalities constraints are tested in ascending order, beginning with the smallest element $t \in S_i$, up to p_i (the largest value in S_i). However, there is a high probability that workload constituted by τ_i might not be fulfilled at smallest element of S_i which is p_1, while the probability is higher that this demand could be completed at the largest element of S_i, which is p_i, as the interval $[0, P_1] \ge [0, P_i]$.

4.2 Redundancy

Many points are redundant such as p_1, the time period of the highest priority task τ_1, which is equally important to be checked for lower priority task despite the fact that it is very likely that lower priority tasks might become unschedulable at this point. We avoid this redundancy by proposing that, when the time demand for task τ_i is not fulfilled at any point $'P_i'$, then it becomes unnecessary to test feasibility of lower priority tasks at the same point t as

$$\sum_{j=1}^{i} c_j \left\lceil t / p_j \right\rceil + c_{lower} > t \quad (7)$$

where c_{lower} is the execution demand of tasks whose priority is lower than τ_i, and $c_{lower} > 0, c_{lower} \in c_{i+1,...,}c_n$,

Definition 1(False Point):
RM scheduling point t, is said to be a false point for τ_i iff $W_i(t) > t$ at point $'t'$.

Based on the above observations we are now ready to present the necessary and suffecient condition for the fixed priority systems by extending the work of Lehoczky et al. discussed in Theorem 2. We implement an aggressive approach in Theorem 3, which unlike traditional approach, commence with largest point in $t \in S_i$ in order of decreasing value. Generally, our test needs a small number of inequalities to be tested, however, in worst case the complexity of our approach is the same, as that of TDA. We can immediately exit, when feasibility condition is satisfied at some point, otherwise that point is marked as false point, which further lowers the complexity of our approach.

Theorem 3
Given a set of n periodic tasks $\{\tau_1, \tau_2, ..., \tau_n\}$, τ_i can be feasibly scheduled for all tasks phasings using RM iff

$$L_i = \min_{t \in Z_i} \frac{W_i(t)}{t} \leq 1 \tag{8}$$

where $Z_i = (S_i - X_{i-1})$ is a list RM-scheduling points in descending order, and X_{i-1} is the set of false points for task τ_i; by definition $X_0 = \phi$. It can be seen that when task number is increased $(i \to n)$, more false points are added to the set of false points i.e. $X_0 \leq X_1 \leq X_2, ..., X_n$.

Proof:
According to the definition of false point $W_i(t) > t$, and $c_{i+1} > 0$, so:

$$W_{i+1}(t) = \sum_{j=1}^{i+1} \lceil t / p_j \rceil c_j = \sum_{j=1}^{i} \lceil t / p_j \rceil c_j + \lceil t / p_{i+1} \rceil c_{i+1}$$
$$= W_i(t) + c_{i+1} > W_i(t) \tag{9}$$

The entire task τ, is feasible for all tasks phasings using RM iff

$$L = \max_{1 \leq i \leq n} \{L_i \leq 1\} \tag{10}$$

We have replaced S_i given in Theorem 2 by Z_i in Theorem 3, which immediately reflects the reduced complexity of our algorithm $Z_i \subseteq S_i$: we need to test task feasibility at fewer points. Although the complexity class is the same, we have lowered the complexity of Theorem 2 by a factor of X_{n-1} (set of false points for τ_n). The results obtained with Theorem 3 can be directly applied to DVS-capable real-time systems without any modification to the fundamental constraints.

We use the term Enhanced Rate Monotonic Analysis (ERMA) hereafter, for our proposed technique. Below is the pseudo code of ERMA.

```
ERMA(τ )
{       X0=0;
        For each τi ∈ τ
        {
        Si = {lPj | j = 1,...,i; l = 1,...,⌊Pi / Pj⌋
        Zi = (Si − Xi−1)
        For each t∈Zi
                {
                If (Li<=1);
                τi is schedulable:break.
                else update Xi;
                }
        }
if (L<=1); τ is feasible.
else τ is infeasible.
}
```

Here we graphically illustratye ERMA in Fig 2. Let assume τ_i is infeasible at RM scheduling point t, or algebraically $\sum_{j=1}^{i} \lceil t / p_j \rceil c_j > t$. This makes it unnecessary to analyze feasibility of lower periodic task $\tau_{i+1}, ..., \tau_n$ at the same point $'t'$, as execution demand of lower priority task is added to the already false inequality. we mark such $'t'$ as false point and avoid this point to be tested during analyzing fesibility of lower priority tasks. The same treatment is applied to all false points that arise as ERMA proceeds. In the diagram symbol | shows the Rate Monotonic testing points while ↑ shows time periods of the current tasks. In Fig. 2, point p_2 is a false point, which is kipped for all lower priority tasks τ_3 up to the lowest priority task τ_n.

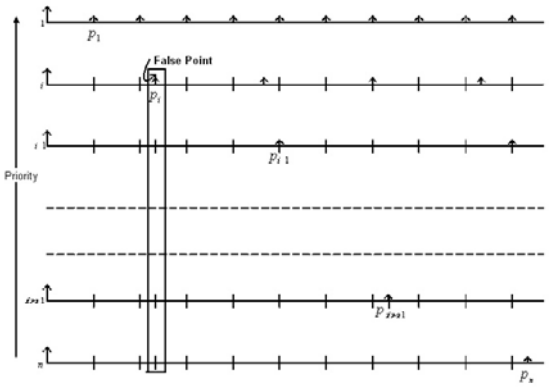
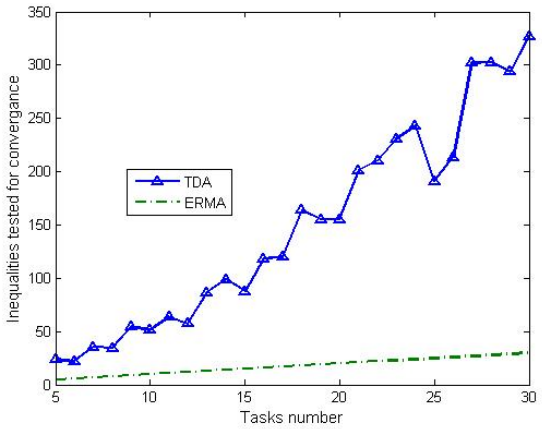


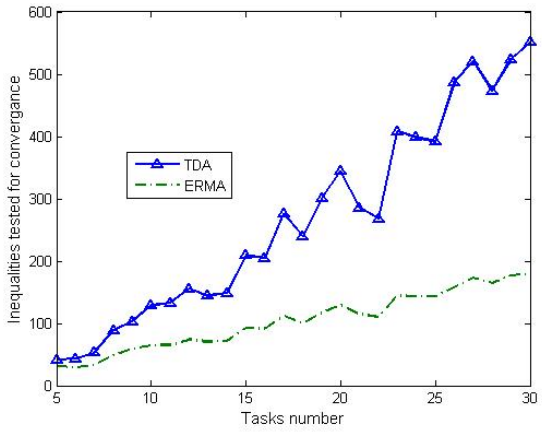
Fig. 1 Identification and propagation of false point

5. Experimental Results

In this section, we evaluate the performance of our scheme. In order to make a comparison of ERMA, Lehoczky, et al.'s work in [10] is also implemented. The performance of a test is evaluated in terms of number of inequalities tested. Together, a series of tasks were generated by varying the range from of 5 to 30. Task periods p_i are randomly extracted in [10,10000] (with uniform distribution) and task computation times c_i are computed as a random variable in $[1, P_i]$ (also with uniform distribution). Both tests begin with a common point p_1, which is the largest as well as the smallest point for highest priority task τ_1, however the advantage of ERMA becomes more promising when lower priority tasks are analyzed. As can be seen in Fig. 3(a) and 3(b), both tests are susceptible to total utilization i.e. more points are tested before feasibility is confirmed. Fig 3 shows the result of our experiment; ERMA has significant advantage over TDA. It can be easily seen that with time demand analysis the average number of inequalities checked increases dramatically, in contrast, ERMA exhibit a smaller increase with increased task numbers.



(a) System utilization ≤ 0.75



(b) System utilization ≤ 1.0

Fig. 3 Inequalities versus task number

6. Conclusions

This paper exploits the idea that tasks are usually unschedulable at points, which lie in beginning of their feasible interval. We also present the concept of false points, a point that fails to satisfy the workload presented by active tasks. The false points are propagated to lower priority tasks so that these points are skipped from their intended search space.

As a future work, we are intended to propose a bound on false points by adjusting tasks period.

Acknowledgments

This work is jointly supported by COMSATS Institute of Information Technology under faculty development program, the research collaboration between the Chinese Academy of Sciences and the Royal Society of the United Kingdom (Grant Number: 20030389, 20032006) and, the State Education Ministry's Scientific Research Foundation for the Returned Overseas Chinese Scholars (Grant Number: 406).

References

[1] G. E. Moore, "Cramming More Components onto Integrated Circuits", **Electronics**, Vol. 38, April 1965, pp.114-117.

[2] D. Linden, and T. Reddy, **Secondary Batteries-Introduction. In Handbook of Batteries**, NY: McGraw-Hill, 3rd edition, 2002, pp.22.3-22.24.

[3] N. Min-Allah, Y. Wang, J. Xing, W. Nisar, and A. Kazmi, "Towards Dynamic Voltage Scaling in Real Time Systems - A Survey", **International Journal of Computer Science and Engineering Systems,** Vol. 1, No. 2, 2007, pp.93-103.

[4] T. D. Burd, T. A. Pering, A. J. Stratakos, and R. W. Brodersen, "A Dynamic Voltage Scaled microprocessor System", **IEEE J. Solid-State Circuits**, Vol. 35, No. 11, 2000, pp. 1571-1580.

[5] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen, "Low Power CMOS Digital Design", In **IEEE Journal of Solid State Circuits**, 1992, pp.472-484.

[6] J. W. S. Liu, **Real Time Systems** , Prentice Hall, 2000.

[7] J. Obenza, "Rate Monotonic Analysis for Real Time Systems", **J.ACM**, Vol.26, No. 3, 1993, pp.73-74.

[8] C. L. Liu, and J.W. Layland, "Scheduling Algorithms for Multiprogramming in A Hard Real-time Environment", **Journal of the ACM**, Vol.20. No.1, 1973, pp.40-61.

[9] E. Bini, G. C. Buttazzo, and G. Buttazzo. "A Hyperbolic Bound for the Rate Monotonic Algorithm", In **IEEE 13th Euromicro Conf. on Real-Time Systems**, pp. 59-66, 2001.

[10] J. P. Lehoczky, L. Sha, and Y. Ding, "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior", in **IEEE Real-Time System Symposium**, 1989 , pp.166-171.

[11] N. C. Audsley, A. Burns, K. Tindell, and A. Wellings, "Applying New Scheduling Theory to Static Priority Preemptive Scheduling, **Software Engineering Journal,** Vol. 8, No. 2, 1993, pp: 80-89.

[12] M. Sjodin, and H. Hansson, "Improved Response-time Analysis Calculations", In **Proceedings of the 19th IEEE Real-Time Systems Symposium**, 1998, pp.399-409.

[13] P. R. Goossens1, **Overview of Real-time Scheduling Problems**, Universite Libre de Bruxelles, 2004