Manuscript Number:

Title: Rate Monotonic Schedulability Tests Using Period-Dependent Conditions

Article Type: Manuscript

Section/Category:

Keywords: Real-time systems; Periodic scheduling; Rate monotonic analysis; Schedulability test.

Corresponding Author: Ms. Wan-Chen Lu,

Corresponding Author's Institution:

First Author: Wan-Chen Lu

Order of Authors: Wan-Chen Lu; Kwei-Jay Lin; Hsin-Wen Wei; Wei-Kuan Shih

Manuscript Region of Origin:

Abstract: Feasibility and schedulability problems have received considerable attention from the real-time systems research community in recent decades. Since the publication of the Liu and Layland bound, many researchers have tried to improve the schedulability bound of the RM scheduling. The LL bound does not make any assumption on the relationship between any of the task periods. In this paper we consider the relative period ratios in a system. By reducing the difference between the smallest and the largest virtual period values in a system, we can show that the RM schedulability bound can be improved significantly. This research has also proposed a system design methodology to improve the schedulability of real time system with a fixed system load.

Cover Page

Paper Title: Rate Monotonic Schedulability Tests Using Period-Dependent Conditions

Author List: Wan-Chen Lu, Kwei-Jay Lin, Hsin-Wen Wei, Wei-Kuan Shih

Addresses and Affiliations of Authors:

W.-C. Lu, H.-W. Wei, W.-K. Shih
Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, ROC; e-mail: {wanchen, bertha, wshih}@rtlab.cs.nthu.edu.tw

K.-J. Lin
Department of Electrical Engineering and Computer Science, University of California, Irvine, CA; e-mail: klin@uci.edu

Rate Monotonic Schedulability Tests Using Period-Dependent Conditions

Abstract Feasibility and schedulability problems have received considerable attention from the real-time systems research community in recent decades. Since the publication of the Liu and Layland bound, many researchers have tried to improve the schedulability bound of the RM scheduling. The LL bound does not make any assumption on the relationship between any of the task periods. In this paper we consider the relative period ratios in a system. By reducing the difference between the smallest and the largest virtual period values in a system, we can show that the RM schedulability bound can be improved significantly. This research has also proposed a system design methodology to improve the schedulability of real time system with a fixed system load.

Keywords Real-time systems **.** Periodic scheduling **.** Rate monotonic analysis **.** Schedulability test

1. Introduction

The design of single-processor real-time systems remains an important issue due to the development of many embedded controllers and sensor devices (Liu and Lee, 2002; Tan and Mooney, 2004). Embedded devices usually have a limited computing power yet must deliver a real time response. They often use simple schedulers such as fixed priority scheduling. How to achieve the best performance under a known resource constraint and how to keep the cost as low as possible are important for many embedded devices designed for the large consumer

This paper is an extended version of the paper "Rate Monotonic Schedulability Conditions Using Relative Period Ratios" that appeared in the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications.

or sensor market.

For a periodic task set, each periodic task τ_i is defined with two parameters (c_i, p_i), where c_i and p_i are the worst case computation time and the period of task τ_i respectively. The *utilization* factor of task τ_i is defined by $u_i = c_i/p_i$. The schedulability of periodic real-time tasks using the Rate Monotonic (RM) fixed priority scheduling algorithm can be checked by summing the utilization factors of all tasks in a system. If the *load* (i.e. total utilization) of the system is less than or equal to a specified bound, the system is guaranteed to be schedulable. Liu and Layland (1973) have proposed the first and the most well-known RM bound (referred as the *LL bound* in this paper). The LL bound is $\sum_{i=1}^{n} u_i \leq n(2^{1/n}-1)$ where n is the number of periodic tasks in the system. When the number of tasks n approaches ∞, the LL bound is $ln\ 2 \approx 0.693$.

The LL bound provides a simple guideline when we design real-time systems. Given a real-time task set, if the load is larger than the LL bound, the schedulability of the task set may not be guaranteed. For example, if the total system load is 0.8 and the number of tasks in the system is more than 2, RM scheduling may not guarantee that all jobs can meet their deadlines. However, it is well known that the LL bound is pessimistic and may be lower than the actual achievable bound given a specific system. For example, the task set $\mathbf{T}$ = {(4, 16), (3, 17) (3, 18), (2, 19), (2, 20)} has a total load of 0.798, which is higher than the LL bound for 5 tasks (0.743), yet it is schedulable.

Since the publication of the LL bound in Liu and Layland (1973), many researchers have tried to improve the schedulability bound of the RM algorithm. One known result is that if all periods are multiples of any smaller period in a system (called the *simply periodic* system), the schedulability bound can be as high as 1 (Liu, 2000). It has also been observed that the number of tasks can be reduced by forming groups of tasks with the harmonic period property; in that case, it is the number of groups that decides the LL bound (Kuo et al., 2003). The schedulability bound is also improved by making conditions on the relationship between any of the periods in Lauzac et al. (2003). A polynomial time checking algorithm has also been designed to compare a system being designed with its simply periodic counterpart (Han and Tyan, 1997). An exact analysis has been proposed in Audsley et al. (1993) using the worst case response time recurrence analysis. However, the exact analysis is time consuming and does not provide a simple methodology on how to improve the schedulability of a system. Many others have studied the RM scheduling bounds extensively (Lehoczky et al., 1989; Burchard et al., 1995; Bini et al., 2003).

Therefore, the first and probably the most important issue for designing an efficient real-time system is to select the best applicable schedulability bound. In this paper, we propose a method to improve the RM schedulability bound by considering the relative period values in a system as in Lauzac et al. (2003). Suppose z_1 and z_2 are the smallest and largest period ratios in the system respectively. When the number of tasks in a system approaches infinity, the schedulability bound derived in Lauzac et al. (2003) is $2z_1 - \ln z_1 - 1$. In this paper, we improve the bound to $2z_1 + 1/z_2 + (\ln z_2 - \ln z_1) - 2$. Moreover, when the number of tasks in a system is n, the schedulability bound derived in Lauzac et al. (2003) is $2z_1 - 1 + (n-1)((1/z_1)^{1/(n-1)} - 1)$, and that derived by our method is $z_1 + 1/z_2 - 2 + (n-2)((z_2/z_1)^{1/(n-2)} - 1)$.

For the general case, we define virtual period v_i for each task τ_i as $\lfloor p_n/p_i \rfloor p_i$. Assume there are n periodic tasks in a system; let the *smallest virtual period ratio* $z_1 = \boldsymbol{min}_{1 \le i \le n-1}\{v_i/p_n\}$ and the *largest virtual period ratio* $z_2 = \boldsymbol{max}_{1 \le i \le n-1}\{v_i/p_n\}$. By reducing the *difference* between the smallest and largest virtual period ratios in a system, we can show that the RM schedulability bound can be improved significantly. Therefore, this paper presents a bound $q = f(z_1, z_2)$ which is a function of the smallest and largest virtual period ratios. If the condition $\boldsymbol{min}_{1 \le i \le n-1}\{v_i/p_n\} \ge z_1$ and $\boldsymbol{max}_{1 \le i \le n-1}\{v_i/p_n\} \le z_2$ is satisfied and the total system load is less than or equal to q, it can be shown that all tasks in the system are RM schedulable. For example, when z_1 is 0.6 and z_2 is 0.65, q is 0.818504. When z_1 is 0.9 and z_2 is 0.95, q is as large as 0.906699 regardless of the number of tasks in a system. These q bounds are significantly better than the LL bound of $ln\ 2$.

The main contribution of this paper is as follows:

1. We have provided the RM bound as a function of z_1 and z_2, where z_1 is the ratio of the smallest virtual period to the largest period and z_2 is the ratio of the largest virtual period to the largest period in a system;
2. Given a desirable schedulability bound and a known longest period in a system, we can identify the acceptable period range for all other tasks.

The remainder of this paper is organized as follows. In Section 2, we define some terminology and provide a motivation for the proposed RM bound. Section 3 presents a RM bound according to the ratios of the smallest and largest virtual periods to p_n. Section 4 provides a methodology for real-time system design by adjusting the periods of all tasks except τ_n. The paper is concluded in Section 5.

2. Definitions and Motivation

Before we discuss our new result, we first present some formal definitions about RM scheduling.

Definition 1. (Liu and Layland, 1973) Let $\mathbf{T} = \{\tau_1, \tau_2, \ldots, \tau_n\}$ be a set of n periodic tasks. Each task τ_i ($i = 1, \ldots, n$) is a tuple (c_i, p_i), where c_i and p_i are the maximum computation time and the period of task τ_i, respectively. The utilization of τ_i, denoted by u_i, is equal to c_i/p_i. The total utilization of $\mathbf{T}$ is denoted as U($\mathbf{T}$). Without loss of generality, we may assume that tasks in $\mathbf{T}$ are indexed in the order of increasing periods, and hence task τ_n has the longest period of all tasks.

Definition 2. (Liu, 2000) *Job $J_{i,j}$*: $J_{i,j}$ is the jth job of periodic task τ_i. That is, $J_{i,j}$ is released at time $(j\text{-}1)p_i$ and has a deadline at time $j\mathrm{p}_i$, for $j \geq 1$. Note that the feasible scheduling interval of $J_{i,j}$ is the time period $((j\text{-}1)\mathrm{p}_i, j\mathrm{p}_i)$, and therefore, the first job of task τ_i has period $(0,\mathrm{p}_i)$.

Definition 3. (Liu, 2000) *Current Period at time t*: At any time t, the current period of a task is the period that begins before t and ends at or after t. That is, for task τ_i, the current period of τ_i is the time interval $(\lceil t/\mathrm{p}_i \rceil \mathrm{p}_i - \mathrm{p}_i, \lceil t/\mathrm{p}_i \rceil \mathrm{p}_i]$. Note that the period starting at time t is not a current period of time t.

Definition 4. (Liu, 2000) *Current Jobs at time t*: At any time t, the current job of a task is the job that is released at the beginning of its current period.

Here is an example to illustrate the definition of current jobs. Let $\mathbf{T} = \{\tau_1 = (1,2), \tau_2 = (2,7)\}$ be a set of periodic tasks. At time 7, the current periods of τ_1 and τ_2 are (6,8] and (0,7], respectively, and their current jobs are $J_{1,4}$ and $J_{2,1}$, respectively, as illustrated in Fig. 1.

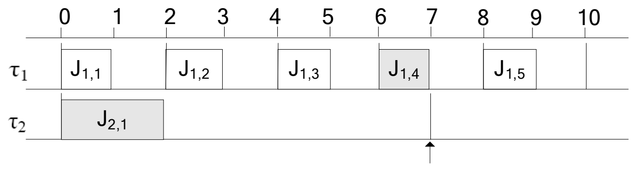


Fig. 1 Current jobs of τ_1 and τ_2 at time 7

Definition 5. Critical Jobs: The critical jobs are defined specifically at time p_n, the largest period in a system. At p_n, the current jobs of all tasks, excluding $J_{n,1}$, are called the critical jobs of the system. Please note that every task except τ_n has a critical job. Also note that the critical jobs are identified on time p_n.

Definition 6. Saturated Schedule and Saturated Task Set: Let **S** be an RM schedule corresponding to a specific task set. We say that **S** is a saturated schedule if no idle time exists in the time interval $(0,\mathrm{p}_n]$ in **S** and all jobs meet deadlines. The task set running on a saturated schedule is called a saturated task set.

For a saturated schedule, if the utilization of task τ_n is increased by any (however small) value, τ_n will miss its deadline.

Definition 7. RM Bound (*RMB*) *and the RMB Task Set*: Any task set with a total utilization less than or equal to RMB is guaranteed schedulable. A saturated task set with a total utilization equal to RMB is called the RMB task set.

Suppose that Γ is the universal set of task sets, formed by all periodic task sets with all possible utilizations and all possible combinations of periods. $\Gamma_{\textit{schedulable}}$ is a subset of Γ, including all task sets that are schedulable under the RM policy. The conditions that a task set is a RMB task set, denoted by $\mathbf{T}_{\mathrm{rmb}}$, are as follows:

1. $\mathbf{T}_{\mathrm{rmb}}$ is a saturated task
2. $\mathbf{T}_{\mathrm{rmb}}$ belongs to $\Gamma_{\textit{schedulable}}$
3. $\mathbf{T}_{\mathrm{rmb}}$ has the smallest total utilization among all task sets belonging to $\Gamma_{\textit{schedulable}}$

The LL bound is shown in Liu and Layland (1973) to be the RMB defined in Definition 7. This RMB is the most general RMB since all possible task sets are considered. However, if we constrain our consideration to some task sets with a specific condition, we can define the conditional RMB as follows.

Definition 8. Conditional RM Bound under Condition C (*CRMB(*C*)*) and *the CRMB(*C*) Task Set*: Given a condition C about a periodic task set (e.g. "$p_1 \geq 0.6p_n$ and $p_{n-1} \leq 0.8p_n$"), the CRMB(C) is the RMB of all task sets that satisfy condition C. A saturated task set meeting condition C and with a total utilization equal to CRMB(C) is called the CRMB task set.

For example, CRMB($p_1 > 0.5p_n$) is equal to the LL bound. In this paper, we will first present our study on deriving CRMB($p_1 \geq z_1 p_n$, $p_{n-1} \leq z_2 p_n$). With that new bound and given a fixed system load, we then present a real-time system design methodology to meet the condition C so that CRMB(C) $\geq$ the system load. In other words, the system may be refined to satisfy the condition C in order to guarantee that the CRMB(C) is larger than or equal to the given system load.

3. The Derivation of the CRMB Formula

3.1. CRMB for $p_1>p_n/2$

Just like the derivation of the LL bound, we also need the worst case (or hard-to-schedule) task set to derive our CRMB(C). For the LL bound, the hard-to-schedule case happens on the tasks set with $p_1>p_n/2$. In this paper, we will show that the hard-to-schedule case happens when $p_1 \geq z_1 p_n$ and $p_{n-1} \leq z_2 p_n$ where $z_1>1/2$ and $z_2 \leq 1$. That is, to find the CRMB(C), we only need to consider the task set with $p_1 \geq z_1 p_n$ and $p_{n-1} \leq z_2 p_n$. In the next subsection, we will show how to handle the case when some periods are relatively small (i.e. $p_i \geq z_1 p_n$ where $z_1 \leq 1/2$ for some task τ_i).

Our study is motivated by the observation that many jobs actually use more processor time than they are supposed to (according to their utilizations) in RM schedules. For task τ_n with a deadline p_n, all other tasks' current jobs at time p_n may take some extra time and utilization. (The current job of τ_n does not consume extra processor time.) These are called the *critical* jobs (defined earlier) at time p_n. We define another concept, called the *gross* utilization, to reflect how the total CPU time before p_n is distributed among all tasks.

Definition 9. Gross Utilization: For a task set **T**, let **S** be the actual RM schedule. The gross utilization of task τ_i, denoted by y_i, is defined by e_i/p_n where e_i is the total scheduled time of τ_i during $(0,p_n]$ in **S**.

For example, let **T** = $\{\tau_1, \tau_2, \ldots, \tau_n\}$ be a set of periodic tasks. The computation time of τ_1 is 20, and the period of τ_1 and τ_n are 80 and 100, respectively. Using the RM scheduling, τ_1 is scheduled for 40 time units before time 100. Therefore, $y_1 = 40/100 = 0.4$ while $u_1 = 0.25$ (less than y_1).

Definition 10. Excessive Utilization: For a task set **T**, the excessive utilization w_i of task τ_i is equal to $y_i - u_i$.

The excessive utilization w_1 of task τ_1 in the last example is $0.4 - 0.25 = 0.15$.

Let **S** be a RM schedule for a task set **T**. In **S**, each task τ_i has two types of utilization u_i and y_i. u_i is the actual utilization of τ_i and is independent to the actual schedule. y_i is the gross utilization of τ_i and depends on the actual schedule. When the excessive utilization w_i of all tasks is zero, the total utilization can be as high as 1 and no task will miss its deadline.

If the critical job of a task τ_i is scheduled at or after p_n, or the critical job scheduled in $(0,p_n]$ is not larger than $(p_n - \lfloor p_n/p_i \rfloor p_i)u_i$, the excessive utilization w_i of this task may be zero, or even less than 0. Since we are only interested in the schedulability bound, we assume that the excessive utilization is 0 whenever it is less than 0.

Suppose the *smallest system period ratio* is z_1, i.e. $z_1 \le p_1/p_n$. Let us now assume $1/2 < z_1 \le 1$. (z_1 with a smaller value will be discussed in the next subsection). Since $p_1 \ge z_1 p_n$, $p_{n-1} \le z_2 p_n$ and $p_{n-1} \ge \dots \ge p_2 \ge p_1$, all critical jobs will be scheduled after the beginning of the 2nd period of τ_1 (i.e. after $z_1 p_n$), either in $(z_1 p_n, p_n]$ or after p_n. We will show that when z_1 increases or z_2 decreases, CRMB($p_1 \ge z_1 p_n$, $p_{n-1} \le z_2 p_n$) increases. For simplicity, we will use CB(z_1, z_2) to denote CRMB($p_1 \ge z_1 p_n$, $p_{n-1} \le z_2 p_n$) for the rest of this paper.

We first assume that the number of tasks in a task set approaches infinity. We will remove this condition later in the section. Without loss of generality, we assume that each task has a unit-length computation time. (The unit-length computation time can be set to any small value for discussion purposes.)

Lemma 1. *Suppose a saturated task set **T** is a CB(z_1, z_2) task set and **T** satisfies the condition $p_1 \ge z_1 p_n$ and $p_{n-1} \le z_2 p_n$. Then, in the saturated schedule of **T**, jobs scheduled in $(p_1,p_n]$ must be critical jobs of tasks in **T**.*

Proof: Assume this lemma is incorrect. That is, there exists an x in $(p_1,p_n]$, and the job scheduled in x is not a critical job. So, we can increase the deadlines of jobs scheduled in (p_1,

x-1] by one and this task set is also a saturated task set. After this deadline modification, we have a task set with a smaller utilization. In this way, we have found a new saturated task set with a smaller total utilization and we get a new CB(z_1, z_2) with a lower total utilization. This is a contradiction. Therefore, jobs scheduled in (p_1,p_n] must be critical jobs of tasks in **T**. □

Lemma 2. *Suppose the saturated task set **T** is a CB(z_1, z_2) task set and **T** meets the condition $p_1 \geq z_1 p_n$ and $p_{n-1} \leq z_2 p_n$. In a saturated schedule of **T**, if the critical job of τ_k at p_n is scheduled at x in (p_1,p_n], the period of τ_k is x.*

Proof: According to Lemma 1, we know that all critical jobs of **T** at p_n are executed between $z_1 p_n$ and p_n. (Fig. 2 shows the RM schedule of critical jobs.) In the saturated schedule of **T**, if the period of τ_k is not equal to x, p_k must be less than x; otherwise, the critical job of τ_k cannot be scheduled in x. If we set p_k be equal to x, the schedule remains the same; but u_k will be decreased, and we find a new saturated task set with a smaller total utilization. That is, we get a new CB(z_1, z_2) with a lower total utilization. This is a contradiction. Therefore, the period of τ_k must be x. □

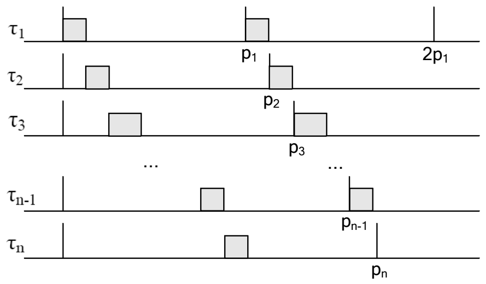


Fig. 2 Critical jobs scheduling

Theorem 1. *Given that $z_1 = p_1/p_n$ and $z_2 = p_{n-1}/p_n$, CB(z_1, z_2) = $2z_1+1/z_2+(\ln z_2 - \ln z_1)-2$.*

Proof: We know that $p_1 \geq z_1 p_n$, and in a saturated task set, each task has unit-length computation time. According to Lemma 1 and Lemma 2, we know that in the schedule of the CB(z_1, z_2) task set, jobs scheduled in $(p_1,p_n]$ must be critical jobs, and if the critical job of τ_k at p_n is scheduled in x, x belongs to $(p_1,p_n]$, then, the period of τ_k is x. Since $p_1 \geq z_1 p_n$ and $p_{n-1} \leq z_2 p_n$, in the worst case, $p_1 = z_1 p_n$ and $p_{n-1} = z_2 p_n$. Therefore, in the worst case, there is one unit critical job of τ_i scheduled in $(z_1 p_n, z_2 p_n]$ where i = 1 to n-2. For τ_i, there are two units scheduled in the time interval $(0,p_n]$. Therefore, $u_i = 1/p_i$, but $y_i = 2/p_n$. The excessive utilization is $y_i - u_i = 2/p_n - 1/p_i$. When the unit-length of computation time is sufficiently small (that is, the number of tasks n approaches infinity), the total excessive utilization for τ_1, …, τ_{n-2} in the saturated task set is:

(the total gross utilization of τ_1, τ_2, …, τ_{n-2}) - (the total utilization of τ_1, τ_2, …, τ_{n-2}), i.e.

$$2(z_2 - z_1) - \int_{x=z_1}^{z_2} (1/x)dx = 2(z_2 - z_1) - (\ln z_2 - \ln z_1).$$

And the excessive utilization for τ_{n-1} in the saturated task set is

(the total gross utilization of τ_{n-1}) - (the total utilization of τ_{n-1})

$= 2(1-z_2) - (p_n - p_{n-1})/p_{n-1} = 3 - 2z_2 - 1/z_2$.

Thus, CB(z_1, z_2) is 1- (the total excessive utilization), i.e.

$$1-((2(z_2 - z_1)-(\ln z_2 - \ln z_1))+(3-2z_2-1/z_2)) = 2z_1+1/z_2+(\ln z_2 - \ln z_1)-2. \quad \square$$

From Theorem 1, if z_1 = 0.55 and z_2 = 0.6, the CB(0.55, 0.6) is equal to $2(0.55)+1/0.6+(\ln 0.6 - \ln 0.55)-2 = 0.853678$, or if z_1 = 0.55 and z_2 = 1, the CB(0.55, 1) is equal to $2(0.55)+1/1+(\ln 1 - \ln 0.55)-2 = 0.697837$. Table 3 shows the values of CB(z_1, z_2) where z_1 and z_2 = 0.55, 0.6, …, 1. It is obvious that on condition that z_1 is fixed, if z_2 decreases, CB(z_1, z_2) increases. Note that the gray area in Table 1 shows the schedulability bound proposed in Lauzac et al. (2003) which only considers the condition of z_1 (i.e. 0.55, 0.6, …, 1) on large number of tasks. Therefore, a higher schedulability bound could be achieved by adjusting the value of z_2.

Table 1 CB(z_1, z_2) on large number of tasks

	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95	1
0.55	0.918182	0.853678	0.805516	0.769733	0.743488	0.724693	0.711789	0.703588	0.699175	0.697837
0.6		0.866667	0.818504	0.782722	0.756477	0.737682	0.724777	0.716576	0.712164	0.710826
0.65			0.838462	0.802679	0.776434	0.757639	0.744735	0.736534	0.732121	0.730783
0.7				0.828571	0.802326	0.783531	0.770627	0.762426	0.758013	0.756675
0.75					0.833333	0.814539	0.801634	0.793433	0.789020	0.787682
0.8						0.850000	0.837095	0.828894	0.824482	0.823144
0.85							0.876471	0.868270	0.863857	0.862519
0.9								0.911111	0.906699	0.905361
0.95									0.952632	0.951293
1										1.000000

The bound is higher when the number of tasks is $n < \infty$. The CB(z_1, z_2) for n tasks is given in Theorem 2. Note that when $n = 2$, $z_1 = z_2 = p_1/p_2$.

Theorem 2. *Given that $z_1 = p_1/p_n$, $z_2 = p_{n-1}/p_n$, $CB(z_1, z_2) = 2z_1+1/z_2 -2 + (n-2)((z_2/z_1)^{1/(n-2)}-1)$ where n is the number of tasks in a system.*

Proof: For a hard-to-schedule task set (Liu and Layland, 1973), it has been shown that for $k = 1, 2, \ldots, n-2$, $p_{k+1} = p_k(1 + u_k)$ (see Fig. 2). Moreover, since $p_1 \geq z_1 p_n$ and $p_{n-1} \leq z_2 p_n$, we know that $p_n \leq p_1/z_1$ and $p_{n-1} \leq z_2 p_n$. Combining these two expressions, we have the following inequality:

$$p_1(1+u_1)p_2(1+u_2)\ldots p_{n-2}(1+u_{n-2})p_{n-1}p_n \leq p_2 p_3 \ldots p_{n-1}(z_2 p_n)(p_1/z_1)$$

$$\Rightarrow (1+u_1)(1+u_2)\ldots(1+u_{n-2}) \leq z_2/z_1$$

Treating $u_1, u_2, \ldots, u_{n-2}$ as variables, it has been shown in Liu (2000) that

$$(1 + (u_1+u_2+\ldots+u_{n-2})/(n-2))^{n-2} \leq (1+u_1)(1+u_2)\cdots(1+u_{n-2})$$

Thus we can obtain the following result

$(1 + (u_1+u_2+\ldots+u_{n-2})/(n-2))^{n-2} \le z_2/z_1$

$\Rightarrow u_1+u_2+\ldots+u_{n-2} \le (n-2)((z_2/z_1)^{1/(n-2)}-1)$

Moreover, the utilization of τ_{n-1} in a hard-to-schedule task set is $(p_n-p_{n-1})/p_{n-1}$, that is, $1/z_2-1$. Since z_1 is always larger than 1/2, all tasks except τ_n have two jobs scheduled in $(0, p_n]$. The utilization of τ_n in a hard-to-schedule task set is 1 - (the total gross utilization of $\tau_1, \tau_2, \ldots, \tau_{n-1}$) $= 1 - 2(1 - z_1) = 2z_1 -1$. Therefore,

$$\begin{aligned} CB(z_1, z_2) &= u_1+u_2+\ldots+u_{n-2}+u_{n-1}+u_n \\ &\le (n-2)((z_2/z_1)^{1/(n-2)}-1)+(1/z_2-1)+(2z_1-1) \\ &= 2z_1+1/z_2-2 + (n-2)((z_2/z_1)^{1/(n-2)}-1) \quad \square \end{aligned}$$

Table 2 shows the values of $CB(z_1, z_2)$ (where z_1 and $z_2 = 0.55, 0.6, \ldots, 1$) when the number of periodic tasks in a system is 3. The gray area in Table 2 is close to the schedulability bound derived in Lauzac et al. (2003) which only considers the condition of z_1 (i.e. 0.55, 0.6, …, 1).

Table 2 $CB(z_1, z_2)$ on the system of 3 tasks

	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95	1
0.55	0.918182	0.857576	0.820280	0.801299	0.796970	0.804545	0.821925	0.847475	0.879904	0.918182
0.6		0.866667	0.821795	0.795238	0.783333	0.783333	0.793137	0.811111	0.835965	0.866667
0.65			0.838462	0.805495	0.787179	0.780769	0.784163	0.795726	0.814170	0.838462
0.7				0.828571	0.804762	0.792857	0.790756	0.796825	0.809774	0.828571
0.75					0.833333	0.816667	0.809804	0.811111	0.819298	0.833333
0.8						0.850000	0.838971	0.836111	0.840132	0.850000
0.85							0.876471	0.869935	0.870279	0.876471
0.9								0.911111	0.908187	0.911111
0.95									0.952632	0.952632
1										1.000000

In Theorem 1, we have proved the $CB(z_1, z_2)$ bound when the number of tasks approaches ∞. In the following, we prove that if the number of tasks $n < \infty$, and the total utilization is less than the CB(z_1, z_2) bound, the task set is also schedulable.

Lemma 3. *Given a task set **T** with n tasks, let **T'** be the task set created by dividing each task in **T** into subtasks with unit-length computation time. The period of each subtask remains the same as that of its corresponding tasks in **T**. **T'** is schedulable if and only if **T** is schedulable.*

Proof. The RM scheduling policy is a fixed priority policy. Since τ_1 in **T** has the smallest period among all tasks in **T**, it has the highest priority. All subtasks corresponding to τ_1 in **T'** also have the highest priority among all tasks in **T'**. Therefore, the time interval assigned to τ_1 in the schedule of **T** is the same as the time interval used by all subtasks of τ_1 in the schedule of **T'**. This is also true for τ_2, τ_3, …, τ_n. Therefore, there exists a schedule of **T** such that **T** is schedulable if and only if there exists a schedule of **T'** such that **T'** is schedulable. □

Although in our proof, each task is divided into small subtasks; it is for the proof discussion only, not for actually scheduling these tasks.

Theorem 3. *Let CB(z_1, z_2) be the RM bound for the case that the number of tasks n approaches infinity. **T** is a task set with the number of tasks being a fixed value n. If U(**T**) $\leq$ CB(z_1, z_2), then **T** is schedulable.*

Proof*:* We create a new task set **T'** by dividing each task in **T** into small subtasks with unit-length computation time (or we can set the unit to be a sufficiently small value). After **T'** is constructed, the number of tasks n approaches infinity and we perform the schedulability test on **T'**. Since U(**T'**) = U(**T**)**,** if the U(**T**) $\leq$ CB(z_1, z_2), then U(**T'**) $\leq$ CB(z_1, z_2). Therefore, **T'** is schedulable. From Lemma 3, we know that **T** is also schedulable. □

From Theorem 3, we know that the schedulability bound for the case that n approaches infinity is also a schedulability bound for any given n, the number of tasks. Therefore, if the number of tasks in the system is unknown, to simplify the on-line schedulability testing, we can apply the schedulability bound for the case that the number of tasks n approaches infinity.

3.2. CRMB for the General Case

For the case that some tasks have their periods less than or equal to $p_n/2$, we can show that $CB(z_1, z_2)$ is actually the same. Since $p_1 \leq p_n/2$, the definition of the ratios z_1 and z_2 needs to be changed to reflect the actual schedule. Under this condition, we use the critical jobs defined in Section 2 to define the z_1 and z_2. Let k_i be $\lfloor p_n/p_i \rfloor$ and $k_i \geq 2$. It is obvious that, the ready time of the critical job of τ_i is $k_i p_i$. Therefore, we define the virtual period of each task as follows.

Definition 11. Virtual Period of Task τ_i: A virtual period of τ_i, denoted by v_i, is the ready time of the critical job of τ_i. That is, $v_i = \lfloor p_n/p_i \rfloor p_i$.

Note that for the case discussed in Section 3.1, all tasks have a virtual period equal to their real period. Therefore, all p_i (for $i = 1, \dots, n$) used in the discussions of previous sections may be replaced by their virtual period (i.e. v_i). After the periods replacement, we need to find the smallest and the largest values among all virtual periods and the values of z_1 and z_2 are modified to be:

$$z_1 = \boldsymbol{min}_{1 \leq i \leq n-1}\{v_i/p_n\}$$

$$z_2 = \boldsymbol{max}_{1 \leq i \leq n-1}\{v_i/p_n\}$$

The above formula provides us a new insight when scheduling real-time systems. That is, we only need to consider the ready times of all critical jobs, or the virtual periods. If the actual periods of tasks are small, only the ready times of all critical jobs need to be examined. All of them must satisfy the constraint defined in Theorem 1 and Theorem 2. This idea will be discussed in more details in Section 4.

In this section, we consider the case that $p_1 \le p_n/2$. $p_1 \le p_n/2$ implies that it is possible that $p_2 \le p_n/2$. Under this condition, we need to make a transformation for all tasks with relative small periods in the system to find the CB(z_1, z_2). In the following theorem, we will derive the formula for the general task sets.

Theorem 4. *Let **T** be a task set with n periodic tasks. Suppose* $z_1 = \boldsymbol{min}_{1 \le i \le n-1}\{v_i/p_n\}$ and $z_2 = \boldsymbol{max}_{1 \le i \le n-1}\{v_i/p_n\}$. ***T** is RM schedulable if* $U(\mathbf{T}) \le CB(z_1, z_2) = 2z_1+1/z_2+(\ln z_2 - \ln z_1)-2$.

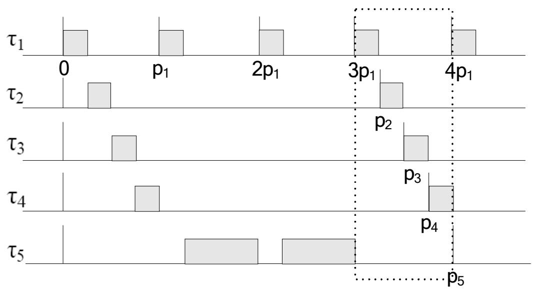
Proof: For the case that all tasks have virtual periods equal to their actual periods, the discussion in Section 3.1 provides the proof. Here we only consider the task sets where some tasks have relatively small periods so that their virtual periods are not their actual periods.

Assume the theorem is not correct. That is, there is a task set **T** with $p_1 \le p_2 \le \ldots \le p_j \le p_n/2$, $U(\mathbf{T}) \le CB(z_1, z_2) = 2z_1+1/z_2+(\ln z_2 - \ln z_1)-2$ where $z_1 = \boldsymbol{min}_{1 \le i \le n-1}\{v_i/p_n\}$, $z_2 = \boldsymbol{max}_{1 \le i \le n-1}\{v_i/p_n\}$ and **T** is not schedulable. Since **T** is not schedulable, p_n must have missed its deadline in a saturated schedule from 0 to p_n.

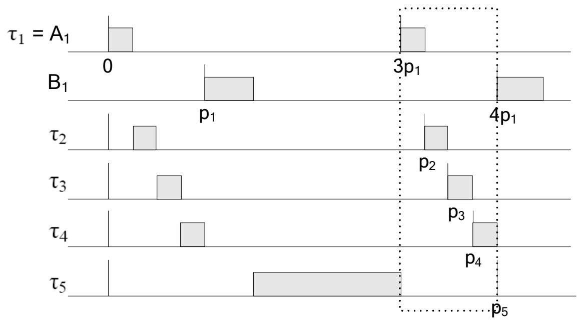
We will transform **T** by increasing those small task periods so that all periods $p_i > p_n/2$. The transformation of $\tau_1 = (c_1, p_1)$, for example, is done as follows. Let $k = \lfloor p_n/p_1 \rfloor$ and $k \ge 2$. We divide τ_1 into two tasks, denoted by A_1 and B_1 where $A_1 = (c_1, kp_1)$ and $B_1 = ((k-1)c_1, kp_1)$. Furthermore, we delay the ready time of B_1 to time p_1. Let us call the new task set created by this transformation as $\mathbf{T}_{new}$. In $\mathbf{T}_{new}$, $u_{A1}+u_{B1} = c_1/(kp_1)+ (k-1)c_1/(kp_1) = c_1/p_1 = u_1$. Both A_1 and B_1 have a period that is the virtual period v_1 of τ_1 and $U(\mathbf{T}_{new}) = U(\mathbf{T})$. Fig. 3 shows the transformation when $k = 3$. In Fig. 3(a), we can see the virtual period v_1 of τ_1 is $3p_1$. In Fig. 3(b), τ_1 is partitioned into A_1 and B_1. Now, the period of A_1 is $v_1 > p_n/2$. The critical job of A_1 is the same as τ_1. However, the critical job of B_1 is not scheduled in the time interval $(v_1, p_n]$

so that the excessive utilization of B_1 is zero. If **T** is not schedulable, $\mathbf{T}_{\text{new}}$ still is not schedulable since the total execution time needed by all critical jobs at p_n remains the same.

In the new task set $\mathbf{T}_{\text{new}}$, if there is another task with a period less than or equal to $p_n/2$, we perform the same transformation on the task and the newly created task set will remain unschedulable. In this way, we can perform the transformation one by one on all tasks τ_i with $p_i \le p_n/2$, and the new task set created by each transformation remains unschedulable. The final task set after all transformations are done, denoted by $\mathbf{T}_{\text{new-complete}}$, is still unschedulable. Moreover, all tasks in $\mathbf{T}_{\text{new-complete}}$ have a period $v_i > p_n/2$ and z_1 is equal to $\boldsymbol{min}_{1 \le i \le n-1}\{v_i/p_n\}$, z_2 is equal to $\boldsymbol{max}_{1 \le i \le n-1}\{v_i/p_n\}$. But according to Theorem 1, we know that $\mathbf{T}_{\text{new-complete}}$ is schedulable if $U(\mathbf{T}_{\text{new-complete}}) = U(\mathbf{T}) \le CB(z_1, z_2) = 2z_1 + 1/z_2 + (\ln z_2 - \ln z_1) - 2$, where $1/2 < z_1 \le z_2 \le 1$. This is a contradiction. Therefore, the theorem is correct. □



(a) The saturated task set under $p_1 \le p_n/2$



(b) The transformation of for τ_1

Fig. 3 CB under $p_1 \le p_n/2$

Here is an example to show what the value of $CB(z_1, z_2)$ is for a task set under $p_1 \le p_n/2$. Let

T = {τ_1, τ_2, τ_3} be a set of periodic tasks. The period of τ_1, τ_2 and τ_3 are 3, 4, and 10 respectively. Since p_1/p_3 and p_2/p_3 are less than 1/2, the virtual period of τ_1 is set to $3p_1 = 9$ and the virtual period of τ_2 is set to $2p_1 = 8$. To calculate the CB(z_1, z_2) of **T**, the virtual periods, 9 and 8, are used. Therefore, the CB(z_1, z_2) of **T** is equal to CRMB($\min(9,8) \geq 10z_1$, $\max(9,8) \leq 10z_2$) = CRMB($8 \geq 10z_1$, $9 \leq 10z_2$) = CB(0.8, 0.9) = 2(0.8)+1/0.9+ (*ln*(0.9)- *ln* (0.8))-2 = 0.8288. Hence, if U(**T**) $\leq$ 0.8288, all tasks in **T** are schedulable.

4. System Design Methodology

The schedulability bound CB(z_1, z_2) described in the previous section allows us to achieve a higher schedulability bound if a system has a high period ratio z_1 and a low period ratio z_2 (when the number of tasks in a task set approaches infinity). Therefore, we can use the CB(z_1, z_2) bound to check for schedulability in those systems. In addition, it also provides system designers the possibility to explore and to adjust task periods in a system in order to achieve a higher schedulability bound.

Suppose there are n tasks in a system and q is the current system load. The maximum period among all tasks, p_n, is also known. If the periods of all other tasks can be modified, our goal is to set them in such a way that the schedulability of all tasks is guaranteed. For designing the system efficiently, we assume the value of z_2 is equal to 1 when finding the CRMB for a system. It should be obvious that as long as the threshold z_1p_n can be determined and if all task periods are set to be larger than or equal to the threshold, the task set is guaranteed RM schedulable.

Unfortunately, the inverse function for the CB(z_1, z_2) function in Theorem 2 is very difficult to derive. We thus use a binary search algorithm to find the threshold z_1p_n where $1/2 < z_1 \leq 1$. The binary search algorithm FIND-THRESHOLD is shown as follows.

Algorithm 1: FIND-THRESHOLD

```
Input: $q$ and $p_n$
Output: threshold
$L = 1/2$; $R = 1$;
while $(R-L)>1/p_n$ do
    $z_1 = (L+R)/2$;
    if $CB(z_1, 1) = 2z_1+1/z_2+(ln\ z_2 - ln\ z_1)-2 < q$ then
        $L = z_1$;
    else
        $R = z_1$;
    end if
end while
return $threshold = Rp_n$;
```

In this algorithm, the system load q and the period of τ_n (i.e. p_n) are given. The interval (L, R) is the current range for the binary search. When $z_1 = (L+R)/2$ has a CB(z_1, 1) smaller than the required system load, we should find a larger z_1 so that the guaranteed CB(z_1, 1) can be increased. The search range will be reduced to ((L+R)/2, R). Otherwise, the current search range will be reduced to (L, (L+R)/2). The search continues until the range is smaller than $1/p_n$. The threshold produced by the algorithm is the minimum value for all task periods except τ_n. Please note that in the binary search process, we always keep CB(R, 1) $\geq q$.

For example, assume $\mathbf{T} = \{\tau_1, \tau_2, \tau_3\}$ is a set of periodic tasks and the total system load of **T** is 0.8. If p_3 is equal to 100, we can design the periods of τ_1 and τ_2 when **T** needs to be guaranteed schedulable by setting a suitable period threshold. The threshold of **T** produced by the search is 77.34. If the virtual periods of τ_1 and τ_2 are selected to be larger than 77.34 but smaller than 100, all tasks in **T** will be schedulable.

5. Conclusions

In this paper, we present the RM schedulability bound as a function of z_1 and z_2, the ratios of the smallest and the largest virtual periods to the largest period in a periodic real-time system. By reducing the difference between the smallest virtual period and the largest virtual period, a periodic task system can achieve a higher RM schedulability bound. This new insight provides a new system design perspective to adjust task periods. We plan to use this result to improve the schedulability bound for multiprocessor real time systems.

References

Audsley NC, Burns A, Richardson M, Tindell K, Wellings A (1993) Applying new scheduling theory to static priority pre-emptive scheduling, Software Engineering Journal, 8(5), pp. 284-292

Bini E, Buttazzo GC, Buttazzo GM (2003) Rate monotonic analysis: the hyperbolic bound, IEEE Transactions on Computers, Vol. 52(7), pp. 933-942

Burchard A, Liebeherr J, Oh Y, Son SH (1995) Assigning real-time tasks to homogeneous multiprocessor systems, IEEE Transactions on Computers, 44(12), pp. 1429-1442

Han CC, Tyan HY (1997) A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithms, 18th IEEE Real-Time Systems Symposium, pp. 36-45

Kuo TW, Chang LP, Liu YH, Lin KJ (2003) Efficient on-line schedulability tests for real-time systems, IEEE Transactions on Software Engineering, 29(8), pp. 734-751

Lauzac S, Melhem R, Mossé D (2003) An improved rate-monotonic admission control and its applications, IEEE Transactions on Computer, 52(3), pp. 337-350

Lehoczky J, Sha L, Ding Y (1989) The rate-monotonic scheduling algorithm: Exact characterization and average behavior, Proceedings of the IEEE Real-Time Systems Symposium, pp. 166-171

Liu CL, Layland JW (1973) Scheduling algorithms for multiprogramming in a hard real-time environment, Journal of the ACM, 20(1), pp. 40-61

Liu J, Lee EA (2002) Timed multitasking for real-time embedded software, IEEE Control System Magazine, 23(1), pp. 65-75

Liu JW-S (2000) Real-Time Systems, Prentice Hall, Englewood, Cliffs, NJ

Tan Y, Mooney V (2004) Timing Analysis for Preemptive Multi-tasking Real-Time Systems with Caches, Proceedings of Design, Automation and Test in Europe, pp. 1034-1039