Non-preemptive Fixed Priority Scheduling of Hard Real-Time Periodic Tasks^{*}

Moonju Park

Ubiquitous Computing Lab., IBM Korea, Seoul, Korea
mjupark@kr.ibm.com

Abstract. This paper addresses the problem of scheduling periodic tasks on a uniprocessor using static priority assignment without preemption. The problem of non-preemptive fixed priority scheduling has received little attention until recently, while real-life applications are often based on non-preemptive systems especially in case of embedded systems. In this paper, we show that Rate Monotonic priority assignment is optimal for non-preemptive scheduling when each task's relative deadline is equal to its period. We have derived a schedulability bound for non-preemptive Rate Monotonic scheduling by using period ratio of the tasks to provide a guarantee that tasks will meet their deadlines. Since the obtained bound is relatively small comparing with the preemptive scheduling, we also propose a method for designing high-utilization non-preemptive system to enhance the utilization bound.

Keywords: real-time scheduling, non-preemptive, periodic tasks, rate monotonic, utilization bound.

1 Introduction

In this paper, we address the problem of fixed priorities non-preemptive scheduling of hard real-time periodic tasks. In many real-time scheduling problems such as I/O scheduling, preemption is often impossible or prohibitively expensive due to properties of hardware device and software handling the device [5]. For example, [10] showed that non-preemptive fixed priority scheduling can be used for real-time signal processing applications because the amount of processor state including the stack to be stored can be reduced. Also non-preemptive scheduling on uni-processor systems eliminates the synchronization overhead of resource protecting mechanism. Since we can expect much lower overhead and smaller memory requirement at run-time with non-preemptive scheduling, non-preemptive scheduling is widely used in embedded systems, especially devices such as cell phones whose behaviour largely depends on network I/O.

Non-preemptive schedulers have received less attention than the preemptive ones, which have been extensively studied for both fixed and dynamic priority assignments since well-known results of Liu and Layland [8]. It is known that the

^{*} This work was supported in part by MIC & IITA through IT Leading R&D Support Project.

general problem of finding a feasible schedule in an idling non-preemptive context is NP-complete. But some results are known for non-idling dynamic priority non-preemptive scheduling. [5] showed that Earliest Deadline First (EDF) is optimal among non-idling schedulers, non-preemptive scheduling of concrete periodic tasks is NP-Hard in the strong sense, and a necessary and sufficient condition for schedulability under non-idling EDF. It was shown in [4] that the worst case response time can be calculated using the results of [12].

Contrary to the preemptive context and dynamic priority scheduling, less results are known about fixed priority non-preemptive scheduling. It was shown that Deadline Monotonic is not optimal when relative deadline differs from period [4], but we do not know yet whether there is an optimal fixed priority scheduler when relative deadline is equal to period. Since there is little guidance for designing fixed priority non-preemptive systems, engineers in such an environment as small embedded devices are developing their systems by trial-and-error. They build systems with a certain priority assignment, and if the system crashes, they add more resources, change periods, or change priorities. The main purpose of this paper is to establish results on schedulability and optimal schedulers, and to give a guidance of designing fixed priority non-preemptive systems, for example, utilization bound.

This paper is organized as follows. Section 2 summarizes assumptions and terms used in this paper. Schedulability conditions and optimal schedulers will be discussed in section 3. Section 4 gives a utilization bound for non-preemptive Rate Monotonic scheduler, and some guides for achieving higher utilization. Finally, section 5 concludes our work.

2 Computational Model and Assumptions

A periodic task is denoted by τ_i . A periodic task set is represented by the collection of periodic tasks, $\tau = \{\tau_i\}$. Each τ_i is associated with (T_i, D_i, C_i) ; period, relative deadline, and the worst case computation time respectively. The total utilization of a periodic task set τ is given by $U_\tau = \sum_{\tau_i \in \tau} \frac{C_i}{T_i}$. For a given task τ_i , we define $hp(\tau_i)$ as the subset of τ consists of tasks with priority equal to or higher than τ_i . On the other hand, $lp(\tau_i)$ is the set of tasks with lower priority than τ_i .

A concrete task has a specified release time, or the time of the first invocation. The difficulty of scheduling tasks can be affected by the release time. A periodic task set is said to be schedulable if and only if all concrete task sets that can be generated from the periodic task set are schedulable. We will consider only periodic task sets in this paper.

The following is a summary of assumptions used in this paper.

- There is only one processor.
- Scheduling overhead can be ignored.
- $C_i \leq D_i \leq T_i$
- Tasks are sorted in non-decreasing order by period. That is, for any pair of tasks τ_i and τ_j , if $j < i$ then $T_j \leq T_i$.

- Tasks are all independent and cannot be suspended by themselves.
- Tasks become ready when arrive. (*i.e.*, there is no inserted idle time.)

3 Schedulability of Scheduling Non-preemptive Periodic Tasks

The following lemma from [4] shows the concept of level- i busy period [7] is also useful in non-preemptive scheduling.

Lemma 1. *A periodic task τ_i has the largest response time in a level- i busy period obtained by releasing all tasks τ_j with $\tau_j \in hp(\tau_i)$ simultaneously at time $t = 0$, and the task τ_k with $\tau_k \in lp(\tau_i)$ and $C_k = \max\{C_l | l > i\}$ just before time $t = 0$.*

Readers may notice that the level- i busy period given in Lemma 1 is basically similar to the critical interval of dynamic priority driven scheduler in [5]. The interval given in Lemma 1 is an extension of the concept of the critical instant in [8].

Based on Lemma 1, the following theorem establishes a necessary and sufficient condition for schedulability for any periodic tasks with fixed priorities.

Theorem 1. *A periodic task set τ is schedulable using a fixed priority scheduler if and only if*

$$C_{\max}^i + \sum_{\tau_j \in hp(\tau_i)} \left\lfloor \frac{D_i}{T_j} \right\rfloor C_j \leq D_i \quad (1)$$

for $\forall \tau_i \in \tau$ where $C_{\max}^i = \max\{C_k | k > i\}$.

Proof. Theorem 1 is derived directly from Lemma 1 and non-preemptibility of tasks.

(if-part) Since a level- i busy period starts from time $t = 0$, the maximum processor demand of the tasks with priority higher than or equal to τ_i before the deadline of τ_i is given by:

$$\sum_{\tau_j \in hp(\tau_i)} \left\lfloor \frac{D_i}{T_j} \right\rfloor C_j$$

Note that we use floor function rather than ceiling function, because tasks are non-preemptive. Since tasks are non-preemptive, we do not need to examine the last instances released after the level- i busy period. Because a task with lower priority than τ_i is already executing just before time $t = 0$, the execution time of the task $C_{\max}^i$ should be accounted. Thus the total processor demand in the level- i busy period is less than or equal to:

$$C_{\max}^i + \sum_{\tau_j \in hp(\tau_i)} \left\lfloor \frac{D_i}{T_j} \right\rfloor C_j$$

By Lemma 1, if τ_i meets its deadline in the level- i busy period, it will never miss its deadline. Therefore if all tasks satisfy equation (1), the task set is schedulable by a fixed priority scheduler.

(only-if-part) Let us assume that τ is schedulable but the equation (1) is not satisfied for some τ_i . Using Lemma 1, we can make a concrete periodic task set such that τ_i has the processor demand equal to lefthand side of the equation (1). Since the processor demand exceeds its period, τ_i should miss its deadline, which leads to a contradiction.

The condition in Theorem 1 is similar to the results of Priority Ceiling Protocol (PCP) [11] or Stack Resource Policy (SRP) [3] by considering the processor as a shared resource. However, using Theorem 1, we can determine the schedulability problem of non-preemptive fixed priority scheduling in polynomial time, while the results for preemptive scheduling has pseudo-polynomial time complexity.

Now we will investigate the problem of optimal priority assignment for non-preemptive scheduling. Suppose that each task τ_i has the same relative deadline as its period, i.e., $D_i = T_i$. In this case, it is well known that Rate Monotonic (RM) scheduling algorithm [8] is optimal for preemptive scheduling. The following theorem shows RM scheduling is also optimal for non-preemptive scheduling.

Theorem 2. *Rate Monotonic is an optimal priority assignment for non-preemptively scheduling periodic task sets with $D_i = T_i$.*

Proof. Let $\tau = \{\tau_1, \tau_2, \dots, \tau_i, \tau_{i+1}, \dots, \tau_n\}$ be a set of n tasks sorted in non-decreasing order by period. Suppose that a scheduling algorithm assigns τ_{i+1} higher priority than τ_i and the task set is schedulable using the priority assignments. Let us consider Rate Monotonic priority assignment with τ_i being the higher priority one by interchanging the priorities of τ_i and τ_{i+1} . For task τ_i , the interference due to higher priority task is reduced by C_{i+1} and $C_{\max}^i$ is given by $\max\{C_j | j = i+1, i+2, \dots, n\}$. And $C_{\max}^i$ for τ_i when it has a lower priority than τ_{i+1} was $\max\{C_j | j = i+2, i+3, \dots, n\}$. The interference τ_i experience after priority exchange is

$$\max\{C_j | j = i+1, i+2, \dots, n\} \leq \max\{C_j | j = i+2, i+3, \dots, n\} + C_{i+1}. \quad (2)$$

Because the total interference for τ_i is reduced, τ_i will remain schedulable.

Now let us consider task τ_{i+1} . It is easy to see that lowering the priority does not increase $C_{\max}^{i+1}$. The only increased interference due to the priority exchange is τ_i 's execution, $\left\lfloor \frac{T_{i+1}}{T_i} \right\rfloor C_i$. As pointed out in [8], if when τ_{i+1} has higher priority than τ_i execution time of $C_{i+1} + C_i \leq T_i$ can be serviced during T_i , the increased interference time due to priority exchange, $\left\lfloor \frac{T_{i+1}}{T_i} \right\rfloor C_i$, can be successfully handled during T_{i+1} because $\left\lfloor \frac{T_{i+1}}{T_i} \right\rfloor T_i \leq T_{i+1}$. Therefore τ_{i+1} will meet its deadlines also with Rate Monotonic priority assignment. As stated in [8], in non-preemptive scheduling also, since the Rate Monotonic priority assignment can be obtained from any priority ordering by a sequence of pair-wise priority reordering as above, we prove the theorem.

For tasks with $D_i \leq T_i$, it is known that Deadline Monotonic priority assignment [2] is optimal for preemptive scheduling. But for non-preemptive scheduling, it was shown in [4] that the Deadline Monotonic is no longer optimal. They also showed that the optimal priority assignment method for preemptive scheduling described in [1] is still optimal in non-preemptive systems. Therefore, since the priority assignment method runs in $O(n^2)$, we can obtain a valid priority assignment method for tasks with $D_i \leq T_i$ in polynomial time by using Theorem 1 and the assignment method of [1].

4 Utilization Bound

We can find a task set with arbitrarily small utilization which is unschedulable by non-preemptive fixed priority scheduling. However, by restricting the ratio of minimum and maximum period (*i.e.*, T_n/T_1), we can achieve relatively higher utilization bound. The following theorem gives us a utilization bound with which task sets can be successfully scheduled by Rate Monotonic non-preemptive scheduler.

Theorem 3. *A set of n periodic tasks with $D_i = T_i$ is schedulable by the non-preemptive Rate Monotonic scheduler if*

$$\sum_{i=1}^n \frac{C_i}{T_i} \leq \frac{1}{r} \quad (3)$$

where $T_n = rT_1$.

Proof. From equation (3), $T_1 \geq T_n \sum_{i=1}^n \frac{C_i}{T_i} = \sum_{i=1}^n \frac{T_n}{T_i} C_i$. For a given task τ_i ,

$$C_{\max}^i + C_i + \sum_{j=1}^{i-1} \left\lfloor \frac{T_i}{T_j} \right\rfloor C_j \leq C_{\max}^i + C_i + \sum_{j=1}^{i-1} \frac{T_n}{T_j} C_j \leq T_n \sum_{j=1}^n \frac{C_j}{T_j} \leq T_1 \leq T_i$$

The task set is schedulable by Theorem 1 since above equation holds for $\forall \tau_i$.

As shown in Theorem 3, the utilization of a schedulable task set decreases as the ratio of the maximum and the minimum period increases. It was shown in [6] that a preemptive task set is schedulable by RM scheduler if the utilization of the task set is less than or equal to $\ln r + 2/r - 1$ where $r = T_n/T_1$ and $1 \leq r \leq 2$. This utilization bound is further enhanced in [9] using additional information of T_n/T_{n-1} , but we will use the result of [6] in this paper for comparison purpose. Fig. 1 compares the utilization bound for preemptive and non-preemptive scheduling using these results. As shown in Fig. 1, the utilization bound for non-preemptive fixed priority scheduling becomes smaller in comparison with preemptive scheduling as the period ratio increases.

To further enhance the utilization bound, we can make use of the following theorem.

Fig. 1. Utilization bound for preemptive and non-preemptive scheduling for different values of r

Theorem 4. *A task set τ of n tasks with $D_i = T_i$ is schedulable by the non-preemptive Rate Monotonic scheduler if*

$$\max_{\tau_i \in \tau} \left\{ \frac{C_i}{T_i} \right\} \leq \frac{1}{r+n} \quad (4)$$

where $T_n = rT_1$ and $n \geq 2$.

Proof. For a given task τ_i , let us suppose $C_{\max}^i = C_k$. By dividing the lefthand side of equation (1) by T_i , we have

$$\begin{aligned} \frac{C_{\max}^i}{T_i} + \frac{C_i}{T_i} + \frac{1}{T_i} \sum_{j=1}^{i-1} \left\lfloor \frac{T_i}{T_j} \right\rfloor C_j &\leq \frac{C_k}{T_k} \cdot \frac{T_k}{T_i} + \frac{C_i}{T_i} + \sum_{j=1}^{i-1} \frac{C_j}{T_j} \leq \\ \frac{r}{r+n} + \sum_{j=1}^i \frac{C_j}{T_j} &\leq \frac{r}{r+n} + \sum_{j=1}^i \frac{1}{r+n} \leq \frac{r}{r+n} + \frac{n}{r+n} = 1 \end{aligned}$$

Therefore if $\max_{\tau_i \in \tau} \left\{ \frac{C_i}{T_i} \right\} \leq \frac{1}{r+n}$, since the condition in Theorem 1 is satisfied, the task set is schedulable.

When it is hard to limit each task's utilization as given in Theorem 2, we can use the utilization bound calculated using the maximum task utilization as in the following corollary derived from Theorem 4.

Corollary 1. *A set of n periodic tasks with $D_i = T_i$ is schedulable by the non-preemptive Rate Monotonic scheduler if*

$$\sum_{i=1}^n \frac{C_i}{T_i} \leq 1 - \alpha r \quad (5)$$

where $T_n = rT_1$ and $\alpha = \max_{\tau_i \in \tau} \left\{ \frac{C_i}{T_i} \right\}$.

Proof. For a given task τ_i , let us suppose $C_{\max}^i = C_k$. By dividing the lefthand side of equation (1) by T_i , we have

$$\frac{C_{\max}^i}{T_i} + \frac{C_i}{T_i} + \frac{1}{T_i} \sum_{j=1}^{i-1} \left\lfloor \frac{T_i}{T_j} \right\rfloor C_j \leq \frac{C_k}{T_k} \cdot \frac{T_k}{T_i} + \frac{C_i}{T_i} + \sum_{j=1}^{i-1} \frac{C_j}{T_j} \leq \alpha r + \sum_{j=1}^n \frac{C_j}{T_j} \leq 1$$

which proves the corollary.

If we limit each task's utilization by the condition given in Theorem 4, by combining Theorem 4 and Corollary 1 (*i.e.*, letting $\alpha = \frac{1}{r+n}$) we can achieve utilization bound as large as $\frac{n}{r+n}$.

As shown in Theorem 4, since tasks are non-preemptive, dividing tasks into several tasks can enhance the utilization bound. In this sense, Theorem 4 gives us a guide for designing real-time systems with non-preemptive fixed priority scheduling. To achieve higher system utilization, tasks should be divided so that we have large number of tasks, and the ratio of the maximum and the minimum period should be kept as small as possible. For example, when $r = 2$, the utilization bound given in Theorem 3 is only 50%. But by dividing some tasks to limit each task's utilization by 10% and to have 8 tasks, the possible utilization bound can be achieve is $\frac{8}{(2+8)} = 80\%$. Note that dividing a task does not help enhancing the utilization bound for preemptive scheduling.

Fig. 2 compares the utilization bound of non-preemptive and preemptive fixed priority scheduling when $r = 2$. The utilization bound for a given number of tasks is calculated using the result of [6], that is $(n-1)(2^{1/(n-1)} - 1) + 2/r - 1$. As the number of tasks increases, the utilization bound of preemptive scheduling approaches to the Liu and Layland's bound, $\ln 2$, while that of non-preemptive scheduling approaches 100%.

Fig. 2. Utilization bound for preemptive and non-preemptive scheduling for different number of tasks

5 Conclusions

In this paper, we have shown Rate Monotonic priority assignment is optimal for non-preemptive scheduling as in preemptive scheduling and schedulability for periodic tasks can be determined in polynomial time. Utilization bound for Rate Monotonic scheduling is presented using the ratio of the maximum and the minimum period. To overcome the small utilization bound, we also presented a method to enhance the utilization bound by limiting the each task's utilization. It was shown that by keeping the period ratio small and dividing a task into tasks with the same period and smaller execution time, we can achieve relatively higher utilization. The optimal fixed-priority scheduling policy for non-preemptive tasks with relative deadline smaller than period, and schedulability of concrete periodic tasks will be our future work.

References

1. N.C. Audsley. Optimal priority assignment and feasibility of static priority tasks with arbitrary start times. Technical report, Department of Computer Science, University of York, 1991.
2. N.C. Audsley, A. Burns, M. Richardson, and A. Wellings. Hard real-time scheduling: The deadline monotonic approach. In *Proceedings of IEEE Workshop on Real-Time Operating Systems and Software*, pages 133–137, May 1991.
3. T.P. Baker. A stack-based allocation policy for real-time processes. In *Proceedings of IEEE Real-Time Systems Symposium*, pages 191–200, December 1990.
4. L. George, N. Riviere, and M. Spuri. Preemptive and non-preemptive real-time uniprocessor scheduling. Technical report, INRIA, 1996.
5. K. Jeffay, D.F. Stanat, and C.U. Martel. On non-preemptive scheduling of periodic and sporadic tasks. In *Proceedings of IEEE Real-Time Systems Symposium*, pages 129–139, December 1991.
6. S. Lauzac and D. Mossé R. Melhem. An improved rate-monotonic admission control and its applications. *IEEE Transactions on Computers*, 52(3):337–350, 2003.
7. J.P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Proceedings of IEEE Real-Time Systems Symposium*, pages 201–209, 1990.
8. C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of ACM*, 20(1):46–61, 1973.
9. W.-C. Lu, H.-W. Wei, and K.-J. Lin. Rate monotonic schedulability conditions using relative period ratios. In *Proceedings of IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 3–9, August 2006.
10. T. M. Parks and E. A. Lee. Non-preemptive real-time scheduling of dataflow systems. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 3225–3238, May 1995.
11. L. Sha, R. Rajkumar, and J. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Transactions on Computers*, 39(9):1175–1185, 1990.
12. M. Spuri. Analysis of deadline scheduled real-time systems. Technical report, INRIA, 1996.