IEICE TRANS. INF. & SYST., VOL.E92–D, NO.10 OCTOBER 2009

LETTER

Utilization Bound of Non-preemptive Fixed Priority Schedulers

Moonju PARK[†a)] ***and*** **Jinseok CHAE**[†b)]**,** ***Members***

SUMMARY It is known that the schedulability of a non-preemptive task set with fixed priority can be determined in pseudo-polynomial time. However, since Rate Monotonic scheduling is not optimal for non-preemptive scheduling, the applicability of existing polynomial time tests that provide sufficient schedulability conditions, such as Liu and Layland's bound, is limited. This letter proposes a new sufficient condition for non-preemptive fixed priority scheduling that can be used for any fixed priority assignment scheme. It is also shown that the proposed schedulability test has a tighter utilization bound than existing test methods.

key words: *real-time system, non-preemptive scheduling, fixed priority*

1. Introduction

Although the Earliest Deadline First (EDF) algorithm is optimal for preemptive [1] and non-preemptive [2] real-time scheduling on a uniprocessor, fixed priority scheduling such as the Rate Monotonic (RM) algorithm has been more widely used in real-time systems because it is very easy to implement [3]. Generally, preemptive schedulers provide better schedulability than non-preemptive schedulers. However, it can be shown that in the context of fixed priority scheduling, preemptive schedulers do not dominate non-preemptive schedulers [4]. Non-preemptive schedulers are useful for I/O-bound applications, they can avoid synchronization problems, and can be used for DSP systems [5].

Unlike dynamic priority scheduling, there is no optimal algorithm for non-preemptive fixed priority scheduling; RM is optimal for preemptive scheduling, but not optimal for non-preemptive scheduling [6]. George *et al.* showed in [6] that the worst case response time can be computed using the concept of the level-i busy period. The test based on the worst case response time in [6] can determine the schedulability of a task set exactly, but the complexity of the exact test is pseudo-polynomial, and thus it is not applicable if the number of tasks is large [3].

In many cases, schedulability tests based on utilization bounds in polynomial time are used because of their simplicity and efficiency. For non-preemptive scheduling, we can apply utilization bounds of preemptive scheduling, such as Liu and Layland's bound, by regarding the processor as a shared resource [3], [7]. However, they can only be applied to RM priority assignment. Since RM is not optimal for non-preemptive scheduling, we need to develop a utilization bound for non-RM priority assignments. This letter presents a new schedulability test for non-preemptive scheduling. The present test can be used for fixed priority scheduling (RM and others), and it is shown to be less pessimistic than the existing tests.

The rest of the letter is organized as follows. Section 2 summarizes the assumptions and notations that are used in this letter. Section 3 proposes a new schedulability test for non-preemptive fixed priority scheduling. Simulation results for evaluating the performance of the proposed method are presented in Sect. 4. Finally Sect. 5 concludes this work.

2. Task Model and Assumptions

A periodic task is denoted as τ_i. A periodic task set is represented by the collection of periodic tasks, $\tau = \{\tau_i\}$. Each τ_i is a two-tuple (T_i, C_i) where T_i is the period and C_i is the worst case execution time. Note that $T_i \geq C_i > 0$. This requires that if the first instance of τ_i is invoked at time t_x, the following instances are invoked periodically at $t_x + kT_i$, where $k = 1, 2, 3, \ldots$, and τ_i must be allocated C_i units of processor time in the interval $[t_x + (k-1)T_i, t_x + kT_i]$.

When an instance of τ_i is invoked at time t and finishes at time t', $t' - t$ is called "response time" of the instance. The worst case response time of τ_i is defined as the maximum possible response time among all instances.

A concrete task has a specified release time, which is the time of the first activation. A concrete task set is schedulable if and only if the worst case response time of any concrete task τ_i in the set is not larger than T_i. A periodic task set is defined as schedulable if and only if all concrete task sets that are generated from the periodic task set are schedulable. In this letter, we consider only periodic task sets.

A summary of the assumptions used in this letter is given. Scheduling overhead can be ignored. Tasks are sorted in non-increasing order according to priority. For any pair of tasks τ_i and τ_j, if $i < j$, then, τ_i has higher priority than τ_j. Time is represented by an integer; therefore, time is discrete and clock cycles are indexed according to integers, as in [2]. Tasks are all independent and cannot suspend themselves.

For a given task τ_i, we define $hp(\tau_i)$ as the subset of τ consisting of tasks with higher priority than τ_i. On the other hand, $lp(\tau_i)$ is the set of tasks with lower priority than τ_i.

Manuscript received May 12, 2009.
Manuscript revised June 11, 2009.

[†]The authors are with the Department of Computer Science and Engineering, University of Incheon, Korea.
a) E-mail: mpark@incheon.ac.kr
b) E-mail: jschae@incheon.ac.kr

DOI: 10.1587/transinf.E92.D.2152

3. Utilization Bound of Non-preemptive Tasks

As described in Sect. 1, we can apply utilization bounds of preemptive scheduling by regarding the processor as a shared resource. A higher-priority task that becomes ready when a non-preemptible lower-priority task is executing is blocked until the lower-priority task completes. The delay due to the non-preemptible lower-priority task is called "blocking time." The worst case blocking time of τ_i is the longest delay caused by the tasks in $lp(\tau_i)$.

Based on the well-known Liu and Layland's bound (called the LL bound in this letter), it was shown in [7] that a task set is schedulable by RM if it satisfies the following condition:

$$\sum_{j=1}^{i-1} \frac{C_j}{T_j} + \frac{C_i + B_i}{T_i} \leq i\left(2^{\frac{1}{i}} - 1\right) \tag{1}$$

for $\forall \tau_i \in \tau$ where B_i is the worst case blocking time of τ_i.

Similarly, the hyperbolic bound [3] can be used to determine whether non-preemptive tasks can be scheduled by RM. A periodic task set τ is schedulable by RM if

$$\left(1 + \frac{C_i + B_i}{T_i}\right) \prod_{j=1}^{i-1} \left(1 + \frac{C_j}{T_j}\right) \leq 2 \tag{2}$$

for $\forall \tau_i \in \tau$. The above tests run in polynomial time ($O(n^2)$) when the number of tasks is n, and they can only be used for RM priority assignment.

In [6], George et al. showed that the concept of the level-i busy period is also useful in non-preemptive scheduling. The level-i busy period is a processor busy period in which only instances of tasks with priority higher than or equal to that of τ_i are processed.

Lemma 1: (From [6]) If tasks are non-preemptive, a periodic task τ_i has the worst case response time in a level-i busy period obtained by releasing all tasks τ_j with $\tau_j \in hp(\tau_i)$ and τ_i simultaneously at time $t = 0$, while releasing task τ_k such that $\tau_k \in lp(\tau_i)$ with $C_k = \max\{C_l | \tau_l \in lp(\tau_i)\}$, at time $t = -1$.

Lemma 1 implies that the maximum interference for task τ_i caused by lower priority tasks is given by $\max\{0, \max\{C_k | \tau_k \in lp(\tau_i)\} - 1\}$. Based on Lemma 1, George et al. also developed an exact schedulability test in [6] that can be performed in pseudo-polynomial time, but the complexity is high if the number of tasks is large.

To compute the upper bound of the maximum interference caused by higher priority tasks, we introduce the following function:

$$G_i(t) = \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{t}{T_j} \right\rceil C_j. \tag{3}$$

The function $G_i(t)$ represents the execution time requested by tasks with priority higher than that of τ_i, from time 0 to time t, if these tasks are released at time 0. The following lemma presents an upper bound of the maximum interference due to higher priority tasks using the above function.

Lemma 2: For a non-preemptive task τ_i, the maximum interference caused by a task $\tau_j \in hp(\tau_i)$ is less than or equal to;

$$\left\lceil \frac{T_i}{T_j} \right\rceil C_j \ , \ if\ G_i\left(\left\lfloor \frac{T_i}{T_j} \right\rfloor T_j\right) + B_i \geq \left\lfloor \frac{T_i}{T_j} \right\rfloor T_j$$
$$\left\lfloor \frac{T_i}{T_j} \right\rfloor C_j \ , \ \text{otherwise}$$

where $B_i = \max\{0, \max\{C_k | \tau_k \in lp(\tau_i)\} - 1\}$.

Proof: The last activation of the task τ_j prior to T_i occurs at time $T_i - (T_i \ mod \ T_j)$, which is given by $\left\lfloor \frac{T_i}{T_j} \right\rfloor T_j$. Let L_{ij} represents this value. By Lemma 1, the worst case response time is achieved by releasing all tasks in $hp(\tau_i)$ and τ_i simultaneously at time $t = 0$, while releasing the task with the largest execution time at time $t = -1$. Thus, the maximum interference for the execution of τ_i is less than or equal to $G_i\left(\left\lfloor \frac{T_i}{T_j} \right\rfloor T_j\right) + B_i$.

If the total execution time requested by $hp(\tau_i)$ prior to $L_{ij} + B_i$ is greater than or equal to L_{ij}, τ_i might not run prior to L_{ij} because the processor must serve higher priority tasks. In this case, the last instance of task τ_j may be executed prior to τ_i, but, if $T_i \ mod \ T_j = 0$, the last instance of τ_j interferes with the subsequent instance of τ_i. Thus, the maximum interference caused by τ_j is less than or equal to $\left\lceil \frac{T_i}{T_j} \right\rceil C_j$.

When the total execution time requested by $hp(\tau_i)$ prior to $L_{ij} + B_i$ is less than L_{ij}, τ_i must have a chance of being scheduled prior to L_{ij}. Since tasks are non-preemptive, once task τ_i is executed it runs until completion, so, the last activation of τ_j does not interfere with τ_i. Thus, the maximum interference caused by τ_j is given by $\left\lfloor \frac{T_i}{T_j} \right\rfloor C_j$. ■

In the following theorem, we present a sufficient condition for non-preemptive fixed priority scheduling.

Theorem 1: A non-preemptive periodic task set τ is schedulable if

$$B_i + C_i + \sum_{\tau_j \in hp(\tau_i)} I_{ij} \leq T_i \tag{4}$$

for $\forall \tau_i \in \tau$, where $B_i = \max\{0, \max\{C_k | \tau_k \in lp(\tau_i)\} - 1\}$ and

$$I_{ij} = \begin{cases} \left\lceil \frac{T_i}{T_j} \right\rceil C_j, & \text{if } G_i\left(\left\lfloor \frac{T_i}{T_j} \right\rfloor T_j\right) + B_i \geq \left\lfloor \frac{T_i}{T_j} \right\rfloor T_j \\ \left\lfloor \frac{T_i}{T_j} \right\rfloor C_j, & \text{otherwise} \end{cases}$$

Proof: By Lemma 2, the maximum interference caused by tasks with higher priority than τ_i is less than or equal to $\sum_{\tau_j \in hp(\tau)} I_{ij}$. By Lemma 1, the maximum interference caused by tasks with lower priority than τ_i is given by B_i. Therefore if $B_i + C_i + \sum_{\tau_j \in hp(\tau_i)} I_{ij} \leq T_i$, τ_i meets its deadline. ■

The schedulability test in Theorem 1 runs in polynomial time with time complexity $O(n^3)$ when the number of tasks is n. But, it can be applied to priority assignments other than RM. Furthermore, the following theorem shows the proposed test can accept more task sets than the other two tests.

Theorem 2: Non-preemptive task sets with utilization less than or equal to the LL bound or hyperbolic bound satisfy the condition in Theorem 1.

Proof: Since Eqs. (1, 2) are applicable to RM scheduling, we only need to consider the RM priority ordering. In RM priority ordering, a higher priority task has shorter period than a lower priority task. So if $i < j$, then $T_i < T_j$. Let us assume that the number of tasks is n. The ratio between the maximum and minimum period, T_n/T_1, is called "period ratio."

The proof consists of two parts. First, we show the theorem holds when period ratio is less than 2. Then we show that general task sets can be converted to have period ratio less than 2, thus the theorem holds for all task sets.

(Part 1) Task sets with period ratio less than 2.

The conditions in Eqs. (1, 2) are based on the fact that the following task set (the hard-to-schedule task set) has the minimum utilization and fully utilizes the processor if tasks are scheduled by RM in a preemptive context [1]:

- $T_1 < T_i < 2T_1$
- $C_i = T_{i+1} - T_i$ for $i < n$, $C_n = T_n - 2\sum_{j=1}^{n-1} C_j$

Equations (1, 2) are obtained by applying above conditions to each task τ_i by replacing $n = i$ and $C_i = \{C_i + B_i\}$ [7]. Thus the above conditions can be modified for each task τ_i:

- $T_1 < T_j < 2T_1$
- $C_j = T_{j+1} - T_j$ for $j < i$, $C_i = T_i - 2\sum_{k=1}^{i-1} C_k - B_i$

If a subset $\{\tau_1, \ldots, \tau_i\}$ has higher utilization than the hard-to-schedule task set, the task set does not satisfy Equation either (1) or (2). From the above conditions, it is clear that $\sum_{k=1}^{j} C_k = T_{j+1} - T_1$ and $\sum_{k=1}^{i} C_k = T_i - B_i - \sum_{k=1}^{i-1} C_k = T_1 - B_i$.

From Eq. (1),

$$B_i + C_i + \sum_{\tau_j \in hp(\tau_i)} I_{ij} \le B_i + C_i + 2\sum_{j=1}^{i-1} C_j$$

$$= B_i + \sum_{j=1}^{i} C_j + \sum_{j=1}^{i-1} C_j = B_i + T_1 - B_i + T_i - T_1 = T_i.$$

Thus, we can see that the hard-to-schedule task set satisfies the condition given by Theorem 1.

Now consider a task set $\{\tau_i\} = \{(T_i, C_i)\}$ where $i = 1, \ldots, n$ and $T_i < 2T_1$, which satisfies Eq. (1) or (2). For a task subset $\{\tau_1, \ldots, \tau_i\}$, we can make a corresponding hard-to-schedule task set $\{\tau'_j\} = \{(T_j, C'_j)\}$, where $C'_j = T_{j+1} - T_j$ for $j < i$ and $C'_i = T_i - 2\sum_{j=1}^{i-1} C'_j - B_i$.

Note that tasks $\tau_1, \ldots, \tau_{i-1}$ execute twice before T_i at most. Since the processor utilization of $\{\tau_i\}$ is lower that that of hard-to-schedule task set, it must be $C_i + 2\sum_{j=1}^{i-1} C_j \le T_i - B_i$ (if $C_i + 2\sum_{j=1}^{i-1} C_j > T_i - B_i$, τ_i misses its deadline). By definition, $C'_i + 2\sum_{j=1}^{i-1} C'_j = T_i - B_i$. So, $C_i + 2\sum_{j=1}^{i-1} C_j \le C'_i + 2\sum_{j=1}^{i-1} C'_j$. Then, we have

$$B_i + C_i + \sum_{\tau_j \in hp(\tau_i)} I_{ij} \le B_i + C_i + 2\sum_{j=1}^{i-1} C_j$$

$$= B_i + C'_i + 2\sum_{j=1}^{i-1} C'_j \le T_i$$

since $\sum_{j=1}^{i-1} C'_j = T_i - T_1$ and $\sum_{j=1}^{i} C'_j = T_1 - B_i$. Therefore, Eq. (4) is satisfied.

(Part 2) Task sets with period ratio larger than or equal to 2.

As in Chapter 6 of [8], we show that a task set with a period ratio larger than or equal to 2 can be converted to a task set with a period ratio less than 2.

For a task τ_j with priority higher than or equal to τ_i, let $v_{ij} = \left\lfloor \frac{T_i}{T_j} \right\rfloor T_j$. Then task set $\{(v_{ij}, \frac{v_{ij}}{T_j} C_j)\}$, where $j = 1, \ldots, i$, has the same utilization as the task set $\{(T_j, C_j)\}$, $j = 1, \ldots, i$. Note that this transformation does not affect B_i because tasks $(T_{i+1}, C_{i+1}), \ldots, (T_n, C_n)$ remain unchanged. If we apply the task set $\{(v_{ij}, \frac{v_{ij}}{T_j} C_j)\}$ to Eq. (1), we have $\sum_{j=1}^{i-1} \frac{v_{ij} C_j / T_j}{v_{ij}} + \frac{v_{ii} C_i / T_i + B_i}{v_{ii}} = \sum_{j=1}^{i-1} \frac{C_j}{T_j} + \frac{C_i + B_i}{T_i}$ since $v_{ii} = T_i$. Thus the schedulability test results using Eq. (1) on task sets $\{(v_{ij}, \frac{v_{ij}}{T_j} C_j)\}$ and $\{(T_j, C_j)\}$ are the same for a given i. We can easily see that the task set $\{(T_j, C_j)\}$, $j = 1, 2, \ldots, i$, satisfies Eq. (2) if and only if task set $\{(v_{ij}, \frac{v_{ij}}{T_j} C_j)\}$ satisfies Eq. (2). Since it is trivial that $\min\{v_{ij}\} < T_i < 2\min\{v_{ij}\}$, the task set $\{(v_{ij}, \frac{v_{ij}}{T_j} C_j)\}$ is a transformation of $\{(T_j, C_j)\}$ with period ratio less than 2. Therefore, if a task set satisfies Eq. (1) or (2), the task set must have utilization lower than or equal to the hard-to-schedule task set, which is shown to satisfy Eq. (4). ■

On the other hand, we can find a task set that does not satisfy either Eq. (1) or (2), but, satisfies the condition given in Theorem 1. Let us consider the task set $\{T_1 = (35, 7), T_2 = (45, 29), T_3 = (46, 3)\}$ as an example. For this task set, $B_1 = 28$, $B_2 = 2$, and $B_3 = 0$. The worst case response times are $R_1 = 35$, $R_2 = 38$, and $R_3 = 46$, so the task set is schedulable. However, this task set has higher utilization than the LL bound and the hyperbolic bound;

(1) LL bound
- T_1: $\frac{7+28}{35} \le 1\,(2-1)$
- T_2: $\frac{7}{35} + \frac{29+2}{45} = 0.888 > 2\left(2^{\frac{1}{2}} - 1\right) = 0.828$
- T_3: $\frac{7}{35} + \frac{29}{45} + \frac{3}{46} = 0.909 > 3\left(2^{\frac{1}{3}} - 1\right) = 0.779$

(2) Hyperbolic bound
- T_1: $\left(1 + \frac{7+28}{35}\right) \le 2$
- T_2: $\left(1 + \frac{7}{35}\right)\left(1 + \frac{29+2}{45}\right) = 2.026 > 2$
- T_3: $\left(1 + \frac{7}{35}\right)\left(1 + \frac{29}{45}\right)\left(1 + \frac{3}{46}\right) = 2.102 > 2$

Table 1 Schedulable task sets by non-preemptive RM.

Average utilization	Exact test	LL-bound	Hyperbolic bound	Proposed test
10%	100%	100%	100%	100%
20%	98%	98%	98%	98%
30%	96%	96%	96%	96%
40%	92%	92%	92%	92%
50%	90%	90%	90%	90%
60%	93%	92%	93%	93%
70%	78%	73%	78%	78%
80%	74%	0%	26%	71%
90%	34%	0%	13%	19%

But, by Theorem 1, we can derive;

T_1: $28 + 7 \leq 35$
T_2: $G_2(35) + 2 = 9 < 35$. So $2 + 7 + 29 = 38 \leq 45$
T_3: $G_3(35) = 36 \geq 35$,
$G_3(45) = 2 \times 7 + 29 = 43 < 45$
Thus $2 \times 7 + 29 + 3 = 46 \leq 46$.

Therefore, the task set is determined to be schedulable.

4. Simulation Results

To show the effectiveness of the proposed test method, we performed a simulation using randomly generated tasks. Task parameters were generated using the UNIX random() function. Periods were smaller than 100,000 and the worst case execution times were restricted to be smaller than 10,000. Any particular task does not have a utilization exceeding 70%, and all tasks have at least 0.5% utilization. We generated and tested a total of 1,300 task sets, containing 8,337 tasks. We compared the percentage of task sets that were determined to be schedulable by non-preemptive RM scheduling. The results are summarized in Table 1.

As shown in Table 1, while all polynomial time tests show a good acceptance ratio if the average utilization is less than or equal to 70%, the schedulability test given in Theorem 1 performs better than the other test overall. For task sets with a processor utilization of less than or equal to 70%, the proposed test method determines the schedulability of all task sets exactly. For task sets with 80% of the average utilization, the proposed method shows an error rate of only 4% in determining the schedulability (4% of task sets are determined as "not schedulable," even though they are schedulable). For task sets with high utilization, the error rate in determining the schedulability becomes larger, but it shows a better ratio than that of other methods.

5. Conclusion

In this letter we proposed a new schedulability test for non-preemptive fixed priority scheduling. The proposed test can be used for any fixed priority assignment, including RM. It was shown that the proposed test method is less pessimistic than existing methods. The proposed test is useful for systems with a large number of tasks, while exact tests are not.

References

[1] C. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," J. ACM, vol.20, no.1, pp.46–61, 1973.
[2] K. Jeffay, D.F. Stanat, and C.U. Martel, "On non-preemptive scheduling of periodic and sporadic tasks," Proc. IEEE Real-Time Systems Symposium, pp.129–139, Dec. 1991.
[3] E. Bini, G.C. Buttazzo, and G.M. Buttazzo, "Rate monotonic analysis: The hyperbolic bound," IEEE Trans. Comput., vol.52, no.7, pp.933–942, 2003.
[4] Y. Wang and M. Saksena, "Scheduling fixed-priority tasks with preemption threshold," Proc. 6th International Conference on Real-Time Computing Systems and Applications, pp.328–335, Dec. 1999.
[5] T.M. Parks and E.A. Lee, "Non-preemptive real-time scheduling of dataflow systems," Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, pp.3225–3238, May 1995.
[6] L. George, N. Riviere, and M. Spuri, "Preemptive and non-preemptive real-time uniprocessor scheduling," Tech. Rep., INRIA, RR-2966, 1996.
[7] L. Sha, R. Rajkumar, and J.P. Lehoczky, "Priority inheritance protocols: An approach to real-time synchronization," IEEE Trans. Comput., vol.39, no.9, pp.1175–1185, 1990.
[8] J.W.S. Liu, Real-Time Systems, Prentice-Hall, 2000.