The Rate Monotonic Scheduling Algorithm: Exact Characterization And Average Case Behavior

John Lehoczky[1], Lui Sha[2] and Ye Ding[1]
Department of Statistics[1]
Department of Computer Science[2]
Carnegie Mellon University

Abstract

This paper presents an exact characterization of the ability of the rate monotonic scheduling algorithm to meet the deadlines of a periodic task set. In addition, a stochastic analysis is presented which gives the probability distribution of the breakdown utilization of randomly generated task sets. It is shown that as the task set size increases, the task computation times become of little importance, and the breakdown utilization converges to a constant determined by the task periods. For uniformly distributed tasks, a breakdown utilization of 88% is a reasonable characterization of the breakdown utilization level. An interesting case is presented in which the average case breakdown utilization reaches the worst case Liu and Layland lower bound.[1]

1. Introduction

The problem of scheduling periodic tasks with hard deadlines equal to the task periods was first studied by Liu and Layland [3] in 1973. In their classic paper, they derived the optimal static priority and dynamic priority scheduling algorithms and determined their worst case behavior. This was accomplished by first identifying the worst case phasing of a task set and then deriving for each n the worst case task set of size n. This is the task set having the smallest processor utilization for which all deadlines are met, but if the processing requirement for any single task were increased by any amount, a deadline would be missed. The optimal fixed priority algorithm was shown to be the *rate monotonic algorithm* in which a task with a shorter period is given a higher priority than a task with a longer period. Ties are broken arbitrarily. The worst case utilization bound was shown to be $n(2^{(1/n)}-1)$, a quantity which decreases monotonically from .83 when n=2 to $\log_e 2$ = .693 as $n \rightarrow \infty$. This result shows that any periodic task set of any size will be able to meet all deadlines all of the time if the rate monotonic algorithm is used and the total utilization is not greater than .693.

Liu and Layland [3] also found that the optimal dynamic scheduling algorithm was the *nearest deadline algorithm*. This algorithm was shown to have a worst case bound of 1.000. Consequently, it can meet all the deadlines of all periodic tasks up to full processor utilization. In spite of the apparent dominance of the nearest deadline algorithm over the rate monotonic algorithm, the rate monotonic algorithm is of great practical importance [1, 2, 5, 6, 10]. First, it can be used to ensure that the timing requirements of the most important tasks are met when a transient overload occurs. Second, it provides a convenient way to offer fast response times to aperiodic tasks while still meeting the deadlines of the periodic tasks using the deferrable server algorithm [2], the sporadic server algorithm [10] or the extended priority exchange algorithm [8]. Third, it can be modified to handle task synchronization requirements by using the priority ceiling protocol [5]. Fourth, it can be conveniently used to schedule tasks where imprecise computation is permitted [4]. Finally, it is easy to implement in processors, in I/O controllers and in communication media [9, 11]. Consequently, it provides an approach to system-wide timing integration.

In common practice, the rate monotonic algorithm can often successfully schedule task sets having total utilization higher than .693. Indeed, it is not uncommon for large periodic task sets with total utilization around .90 to be schedulable using the rate monotonic algorithm. This suggests that the average case behavior is substantially better than the worst case behavior. The behavior is strongly dependent upon the relative values of the periods of the tasks comprising the task set. The purpose of this paper is to give an exact characterization of the ability of the rate monotonic algorithm to meet the deadlines of a given periodic task set. In addition, we present a stochastic analysis of the performance of the algorithm when task sets are generated randomly. Specifically, a task set is generated randomly, and the computation times are scaled to the point at which a deadline is first missed. The corresponding task set utilization is a breakdown utilization. This breakdown utilization is presented and an asymptotic analysis is given. We only consider the case in which task deadlines are given by the arrival of the next

[1]This work was sponsored in part by the Office of Naval Research under contract N00014-84-K-0734, in part by Naval Ocean Systems Center under contract N66001-87-C-0155, and in part by the Systems Integration Division of IBM Corporation under University Agreement YA-278067.

job of that task; however, the analysis presented here extends to the case of more general deadlines.

2. Problem Formulation

Consider a set of n periodic tasks $\tau_1,\ldots,\tau_n$. Task τ_i has a period T_i, a computation requirement, C_i, and a phasing relative to 0, I_i, with $0 \le I_i < T_i$. This means that jobs corresponding to task τ_i are initiated at times $I_i + kT_i$, $k \ge 0$. The job initiated at time $I_i + kT_i$ has $I_i + (k+1)T_i$ as its deadline, the initiation time of the next job. We label the tasks so that $T_1 \le T_2 \le \ldots \le T_n$. Consequently, τ_i receives priority i. We assume tasks are ready to run at their initiation times, tasks can be preempted instantly (and so ignore all blocking) and ignore all overhead such as task swapping times.

Liu and Layland proved that the worst case phasing occurs when $I_i = 0$ for $1 \le i \le n$. This is called a *critical instant* in which all tasks are simultaneously instantiated. Using this concept, Liu and Layland also proved that the task set is schedulable (all deadlines of all jobs of every task are met) using the rate monotonic algorithm if the first job of each task can meet its deadline when it is initiated at a critical instant. Theorem 1 gives a necessary and sufficient condition for the first job of each task to meet its deadline under the worst case phasing. To determine if a task can meet its deadline under the worst case phasing, we need to consider the processor demands made by the task set as a function of time. If we focus on tasks $\tau_1, \ldots, \tau_i$, then the expression

$$W_i(t) = \sum_{j=1}^{i} C_j \cdot \lceil t/T_j \rceil,$$

gives the cumulative demands on the processor made by these tasks over [0,t] when 0 is a critical instant. For later notational convenience, we define

$$L_i(t) = W_i(t)/t,$$

$$L_i = \min_{\{0<t\le T_i\}} L_i(t),$$

$$L = \max_{\{1\le i\le n\}} L_i.$$

Given this notation, we can conveniently express the exact criterion for a task or task set to be schedulable by the rate monotonic algorithm.

Theorem 1: Given periodic tasks $\tau_1, \ldots, \tau_n$,

1. τ_i can be scheduled for all task phasings using the rate monotonic algorithm if and only if $L_i \le 1$.
2. The entire task set can be scheduled for all task phasings using the rate monotonic algorithm if and only if $L \le 1$.

Proof: Given the worst case phasing was proved by Liu and Layland to be a critical instant and we require schedulability for all task phasings, we can restrict attention to the worst case. Given the assumptions of perfect preemption, no blocking, no overhead and jobs are ready at their initiation times, two conclusions follow. First, jobs of τ_i will be preempted only by higher priority jobs, and they will preempt any lower priority jobs. Thus only $\tau_1, \ldots, \tau_i$ need be considered to determine if τ_i can be scheduled. Second, starting at a critical instant, the processor will not idle before either the first job of τ_i is finished or it misses its deadline, whichever occurs first.

Task τ_i completes its computation requirement at time $t \in [0,T_i]$, if and only if all the requests from all the jobs with priority higher than i and C_i, the computation requirement of τ_i, are completed at time t. The total of such requests is given by $W_i(t)$, and they are completed at t if and only if $W_i(t) = t$ and $W_i(s) > s$ for $0 \le s < t$. Dividing by t, we find equivalently $L_i(t) = 1$. It follows that a necessary and sufficient condition for τ_i to meet its deadline is the existence of a $t \in [0,T_i]$ such that $L_i = 1$. Consequently, a necessary and sufficient condition for τ_i to meet its deadline under all phasings is $L_i \le 1$. This proves the first assertion.

The second assertion follows directly from the first. The entire task set is schedulable if and only if each of the tasks is schedulable. This means that $L_i \le 1$ for each i. Equivalently, $L \le 1$ is necessary and sufficient for the task set to be schedulable.

The characterization of schedulability given in Theorem 1 requires a minimization over the continuous variable $t \in [0, T_i]$. In fact, only a finite number of times t need to be checked. The function $L_i(t)$ is piecewise monotonically decreasing, that is $\lceil t/T_i \rceil / t$ is strictly decreasing except at a finite set of values called the rate monotonic scheduling points. When t is a multiple of one of the periods T_j, $1 \le j \le i$, the function has a local minimum, is left continuous and jumps to a higher value to the right. Consequently, to determine if τ_i can meet its deadline one need only search over these local minima, the multiples of $T_j \le T_i$ for $1 \le j \le i$. Specifically, let

$$S_i = \{k \cdot T_j \mid j=1,\ldots,i; k=1,\ldots,\lfloor T_i/T_j \rfloor\}.$$

The elements of S_i are the scheduling points for task i, the deadline of task i and the arrival times of tasks of priority higher than i before the deadline of task i assuming a critical instant phasing. For example, suppose we consider three tasks with periods $T_1 = 5$, $T_2 = 14$, and $T_3 = 30$. The scheduling points are given by $S_1 = \{5\}$, $S_2 = \{5, 10, 14\}$ and $S_3 = \{5, 10, 14, 15, 20, 25, 28, 30\}$. It follows that

$$L_i = \min_{\{t \in S_i\}} L_i(t).$$

One can modify the criteria of Theorem 1 to give

Theorem 2: Given periodic tasks $\tau_1, \ldots, \tau_n$,

1. τ_i can be scheduled for all task phasings using the rate monotonic algorithm if and only if

$$L_i = \min_{\{t \in S_i\}} W_i(t)/t \leq 1.$$

2. The entire task set is schedulable for all task phasings using the rate monotonic algorithm if and only if

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

We present a simple example to illustrate Theorem 2. Suppose we have a task set with three periodic tasks for which $U_i = C_i/T_i$.

- Task τ_1: $C_1 = 20$; $T_1 = 100$; $U_1 = 0.200$
- Task τ_2: $C_2 = 40$; $T_2 = 150$; $U_2 = 0.267$
- Task τ_3: $C_3 = 100$; $T_3 = 350$; $U_3 = 0.286$

The total utilization of these three tasks is 0.753, which is below the Liu and Layland bound for three tasks: $3(2^{1/3} - 1) = 0.779$. Hence, we know these three tasks are schedulable using the rate monotonic algorithm, i.e., they will meet their deadlines if τ_1 is given the highest priority, τ_2 the next highest, and τ_3 the lowest. The remaining 24.7% processor capacity can be used for low priority background processing. However, we can also use it for additional hard real-time computation.

Suppose we replace τ_1's computation with one that is more accurate but requires 40 rather than 20 units of time. The total processor utilization increases from 0.753 to 0.953. Since the utilization of the first two tasks is 0.667, which is below the Liu and Layland bound for two tasks, $2(2^{1/2} - 1) = 0.828$, the first two tasks cannot miss their deadlines. For task τ_3, we use Theorem 2 (part 1) to check whether the task set is schedulable. Here i=3 and $S_3 = \{100, 150, 200, 300, 350\}$.

Task τ_3 is schedulable if any of the following equations holds:

	$C_1 + C_2 + C_3 \leq T_1$	$40 + 40 + 100 > 100$
or	$2C_1 + C_2 + C_3 \leq T_2$	$80 + 40 + 100 > 150$
or	$2C_1 + 2C_2 + C_3 \leq 2T_1$	$80 + 80 + 100 > 200$
or	$3C_1 + 2C_2 + C_3 \leq 2T_2$	$120 + 80 + 100 = 300$
or	$4C_1 + 3C_2 + C_3 \leq T_3$	$160 + 120 + 100 > 350$

The analysis shows that task τ_3 is also schedulable, and in the worst-case phasing will meet its deadline exactly at time 300. Hence, we can double the utilization of the first task from 20% to 40% and still meet all the deadlines. The remaining 4.7% processor capacity can be used for either background processing or a fourth hard deadline task, which has a period longer than that of τ_3. Task τ_3 just meets its deadline at 300, hence we cannot add a task with a priority higher than that of task τ_3.

3. Stochastic Characterization

The necessary and sufficient conditions for schedulability of a task set given in either Theorem 1 or 2 give an exact condition for the rate monotonic algorithm to be able to schedule a given task set, but they do not, by themselves, offer any insight into the conditions under which the rate monotonic algorithm can exceed its worst case utilization bound. One approach is to generate a task set randomly and then use the conditions to determine the utilization level at which it fully utilizes the processor. In most average case analyses of algorithms, a simple probability model is used to generate the cases from which the average results are derived. For example, in bin packing, one generally considers unit size bins and uniformly distributed objects to pack. In the travelling salesman problem, one usually picks cities uniformly in a region, and in sorting algorithms, one usually assumes uniformly distributed permutations of objects to sort. In the case of the rate monotonic algorithm, it is possible to consider a very general model of randomly generated task sets. Specifically, we introduce a cumulative distribution function (c.d.f.) $F_T(t)$ for the task periods and a c.d.f. $F_C(c)$ for the task computation requirements. First, we draw a random sample of task periods, $T_1,...,T_n$, from $F_T(t)$. Next we label them so that $T_1 \leq T_2 \leq \ldots \leq T_n$. Finally, we draw a random sample of computation requirements, $C_1,...,C_n$, from $F_C(c)$. The resulting pairs (C_1,T_1), ... , (C_n,T_n) constitute a random task set.

We think of the C_is and T_is as providing *relative* computation requirements and periods. The task set must be scaled to create an appropriate utilization. In particular, we multiply each C_i by a small factor Δ and systematically increase Δ to a threshold value Δ^* at which some task deadline is missed assuming that the task phasing corresponds to a critical instant. The associated utilization of the task set at which a deadline is first missed is a measure of the threshold at which the rate monotonic algorithm misses a deadline. The quantity Δ is a random variable as is the breakdown utilization, U_n^*. We offer an exact formula for Δ^* and U_n^* and an approximate characterization of the random variable, U_n^*.

For the randomly generated and scaled task set, the criterion developed in Theorem 2 indicates that the task set is schedulable if and only if

$$L = \max_{\{1 \leq i \leq n\}} \min_{\{t \in S_i\}} \sum_{j=1}^{i} \Delta \cdot C_j \lceil t/T_j \rceil / t \leq 1 \quad (1)$$

or

$$\Delta \leq [\max_{\{1 \leq i \leq n\}} \min_{\{t \in S_i\}} \sum_{j=1}^{i} C_j \lceil t/T_j \rceil / t]^{-1}. \quad (2)$$

The largest possible scaling factor, Δ^*, is given by the right side of (2):

$$\Delta^* = [\max_{\{1 \leq i \leq n\}} \min_{\{t \in S_i\}} \sum_{j=1}^{i} C_j \lceil t/T_j \rceil / t]^{-1}. \quad (3)$$

If Δ is increased beyond Δ^*, a deadline will be missed. The utilization associated with the critical scaling factor is the breakdown utilization of the randomly generated task set. It is given by

$$U_n^* = [\sum_{i=1}^{n} U_i] / [\max_{\{1 \le i \le n\}} \min_{\{t \in S_i\}} W_i(t)/t \quad (4)$$

where $U_i = C_i/T_i$.

The numerator is merely the total utilization of the task set. The maximizing index i in the denominator picks the task in the task set whose deadline will be missed if the scaling factor is further increased, and the associated minimizing t picks the time at which that task will meet its deadline. The form for the breakdown utilization given in (4) is useful for conducting simulation studies of the average case behavior of the rate monotonic algorithm. Some simplifications of this formula are sometimes possible. For example, if $F_T(1) = 0$ and $F_T(2) = 1$, so $1 \le T_n/T_1 \le 2$, then the maximum in the denominator occurs when i = n. We formulate this as a theorem, one which offers a simplification of the calculation of the breakdown utilization in this case.

Theorem 3: Suppose $1 \le T_1 \le \ldots \le T_n \le 2$. Then

$$U_n^* = [\sum_{i=1}^{n} C_i/T_i] / \min_{\{t \in S_n\}} \sum_{j=1}^{n} C_j \cdot \lceil t/T_j \rceil / t.$$

Proof: We define

$$\delta_i = \min_{\{t \in S_i\}} \sum_{j=1}^{i} C_j \cdot \lceil t/T_j \rceil / t.$$

From the definition of S_i and the fact that $\lfloor T_n/T_1 \rfloor = 1$, the minimum in δ_i must occur at some T_j, $1 \le j \le i$. We prove $\delta_{i+1} \ge \delta_i$ for all i. This will allow us to conclude that $\delta_n \ge \delta_i$ for all i and will complete the proof of the theorem.

Suppose the minimum for δ_{i+1} is assumed at $t = T_j$ where $j \le i$. Then δ_{i+1}

$= [2(C_1 + \ldots + C_{j-1}) + C_j + \ldots + C_{i+1}] / T_j$

$\ge [2(C_1 + \ldots + C_{j-1}) + C_j + \ldots + C_i)] / T_j$

$\ge \delta_i$,

since the latter is the $t = T_j$ component of δ_i.

Suppose instead that the minimum for δ_{i+1} is assumed at $t = T_{i+1}$. Then δ_{i+1}

$= [2C_1 + \ldots + 2C_i + C_{i+1}] / T_{i+1}$

$\ge (C_1 + \ldots + C_i) / T_1$

$\ge \delta_i$,

since $T_{i+1} \le 2T_1$ and $C_{i+1} \ge 0$.

In either case, $\delta_{i+1} \ge \delta_i$. Consequently, the maximum occurs when i = n, and attention can be restricted to δ_n which completes the proof.

Before giving an asymptotic analysis of U_n^*, we present a simple example.

3.1. Example 1

Assume n = 2, $C_1 = C_2$, and the task periods are chosen according to a Uniform [1,2] distribution. Let $R = T_2/T_1$, so $1 \le R \le 2$. It follows that

$$U_n^* = (R+1)/\min(2R, 3). \quad (5)$$

The probability density function (p.d.f.) of R is given by

$$f_R(r) = (4/r^2) - 1, \quad 1 \le r \le 2 \quad (6)$$

It is straightforward to find the p.d.f. of U_n^* and its mean value. In particular, its density function is given by

$$f(x) = 12/(3x-1)^2 + 5 - 2/(2x-1)^2, \quad \frac{5}{6} \le x \le 1, \quad (7)$$

and this p.d.f. has mean value equal to .917, which is larger than the Liu and Layland bound of .828. Indeed, the assumption $C_1 = C_2$ increases the worst case bound to .833, larger than the usual bound of .828.

4. Asymptotic Approximation

The quantity U_n^* is a random variable, one which is a complicated function of the C_is and T_is. Liu and Layland proved $n(2^{1/n}-1) \le U_n^* \le 1$. We wish to offer insight into the size of the random variable U_n^* as a function of $F_T(t)$ and $F_C(c)$. In this section, we present an asymptotic approximation for this random variable as $n \to \infty$, that is for large task sets.

The expression for U_n^* given in (4) involves sums of random variables. To simplify the presentation, we will assume that the random variables C_i and T_i are bounded and the largest period ratio is also bounded. Consequently, we assume $1 \le T_i \le B$ for some finite B. In addition, we assume that F_T has a density function which is strictly positive over the interval [1,B].

The appearance of sums of random variables in the numerator and denominator of (4) allows for the application of convergence theorems from probability theory. In particular, the numerator of (4) , after dividing by n, is a sum of independent, identically distributed utilization ratios, $U_i = C_i/T_i$. It follows that

$$\sum_{i=1}^{n} U_i/n \to E(U_i) = E(C) \cdot E(1/T) \quad a.s. \quad (8)$$

The latter inequality holds by the presumed independence of C_i and T_i. Although the periods have been labeled to be in increasing order, since the sum contains all the terms, it is independent of the particular permutation used. Finally, the central limit theorem could be used to create a normal approximation to the numerator.

The denominator of (4) requires more care. After dividing by n, the denominator becomes

$$\max_{\{1 \le i \le n\}} \min_{\{t \in S_i\}} \sum_{j=1}^{i} C_j \cdot \lceil t/T_j \rceil / tn. \quad (9)$$

For fixed t and i, the T_j have been ordered to be non-

decreasing, so the sum in (9) involves the i smallest periods. As such, the T_i are no longer independent and identically distributed. Nevertheless, the behavior of the order statistics from a random sample is well understood (see for example [7]). In particular, if we let $i = \lfloor \alpha n \rfloor$, then

$$T_i \to F_T^{-1}(\alpha) \quad a.s.$$

where

$$G(\alpha)=F_T^{-1}(\alpha) = \inf\{x \mid F_T(x) \geq \alpha\}.$$

Moreover, $T_1,...,T_i$ is, for large n, approximately a random sample from the c.d.f. of T conditioned on the event $\{T \leq F_T^{-1}(\alpha)\}$.

4.1. Notation and Assumptions

The asymptotic behavior of U_n^* is derived using the law of large numbers and the asymptotic behavior of the order statistics of a random sample from F_T. In order to apply these theorems, we adopt a series of assumptions concerning F_T.

Assumptions

1. F_T has a strictly positive and continuous density f on [1, B], i.e. $F_T(1) = 0$, $F_T(B) = 1$.
2. The function $h(x) = (1 + x)/F_T^{-1}(x)$ has a unique minimum $x_0 \in [1,B]$ and is locally increasing on $[x_0 , x_0 + \delta]$ and locally decreasing on $[x_0 - \delta , x_0]$ for some $\delta > 0$.
3. The function h(x) defined in (2) above satisfies a Lipschitz condition,

   $$|h(x_1)-h(x_2)| < M|x_1-x_2|$$

   for some $M>0$.
4. Let

   $$w(\alpha) = \min_{\{1 \leq s \leq G(\alpha)\}} I(s)$$

   where

   $$I(s)=\int_0^{\alpha} \lceil s/G(t) \rceil dt/s.$$

   Moreover, $w(\alpha)$ has a unique maximum α_0 and is locally increasing on $[\alpha_0 - \beta , \alpha_0]$ and locally decreasing on $[\alpha_0 , \alpha_0 + \beta]$ for some $\beta > 0$. In addition, $w(\alpha)$ satisfies the Lipschitz condition

   $$|w(\alpha_1)-w(\alpha_2)| < K|\alpha_1-\alpha_2|$$

   for some $K>0$.

To simplify the notation, we define

$$h = \min_{\{0 \leq x \leq 1\}} h(x),$$

$$w = \sup_{\{0 \leq \alpha \leq 1\}} \inf_{\{G(0) \leq s \leq G(\alpha)\}} \int_0^{\alpha} \lceil s/G(u) \rceil du/s.$$

Using this notation, we have the following theorem.

Theorem 4:

1. Under Assumption 1 with $B \leq 2$,

   $$U_n^* \to E(1/T)/h \quad as \quad n \to \infty, \quad a.s.$$

2. Under Assumption 1 with $B \geq 2$,

   $$U_n^* \to E(1/T)/w \quad as \quad n \to \infty, \quad a.s.$$

3. For $B \geq 2$ and under Assumptions 1 and 4 or if F_T is uniform [1,B], then

   $$U_n^* - E(1/T)/w = o_p(n^{r-.5})$$

   for $r>0$.

4. For $B \leq 2$ and under Assumptions 1 and 3,

   $$U_n^* - E(1/T)/h = o_p(n^{r-.5})$$

   for $r>0$.

It is important to note the remarkable fact that the above theorem indicates that the threshold of schedulability, U_n^*, converges to a constant as the task set size increases. Furthermore, this constant depends only upon F_T, not on F_C! The distribution F_C does play a role when n is relatively small; however, asymptotically it drops out.

Theorem 4 also indicates that the convergence rate is nearly $n^{-(1/2)}$. More analysis is needed to fully characterize the asymptotic behavior of the breakdown untilization. In particular one would expect

$$n^{1/2}(U_n^* - E(1/T)/w) \to N(0,\sigma^2)$$

where σ^2 is a complicated function of F_T and F_C. This is commonly observed in simulations; however, it is not true in general. One need only consider Example 2 with B = 2 given in the next section. In this case, the limiting breakdown utilization is the Liu and Layland lower bound. Consequently, the random variable $[U_n^* - E(1/T)/w]$ is strictly positive and cannot be normally distributed.

In many circumstances, it is possible to prove that U_n^* has an approximate normal distribution. In general, however, the behavior can be more subtle and requires working with a Brownian bridge approximation to the empirical distribution function arising in the denominator [7] as well as taking proper account of the potential correlation between the numerator and denominator of (4).

4.2. Example 2

Suppose that the task periods are drawn from a uniform [1,B] distribution with $B \leq 2$. Both numerator and denominator are easy to compute. In particular, $E(1/T) = [\mathrm{Log}_e(B)]/(B-1)$ and $\min_{\{0 \leq x \leq 1\}} h(x)=1$. Consequently

$$U_n^* \rightarrow [Log_e(B)]/(B-1), \quad 1 \leq B \leq 2, \quad n \rightarrow \infty, \quad a.s.$$

If B > 2 , then the numerator is unchanged; however, the denominator must be recomputed. It becomes

$$d = [(B/\lfloor B \rfloor) + \sum_{i=2}^{\lfloor B \rfloor - 1} 1/i]/(B-1).$$

Consequently, for $B \geq 2$ and as $n \rightarrow \infty$,

$$U_n^* \rightarrow [\log_e(B)]/[B/\lfloor B \rfloor + \sum_{i=2}^{\lfloor B \rfloor - 1} (1/i)], \quad a.s.$$

The above example is particularly interesting, because it provides an illustration of a situation in which the asymptotic average case behavior coincides with the worst case behavior. Specifically, when B = 1, the average case asymptotic performance is 1 and decreases monotonically until it reaches the level $\mathrm{Log}_e 2$ when B = 2, the Liu and Layland worst case bound. It then slowly increases to 1 as $B \rightarrow \infty$. The following table presents the limiting performance values, i.e. breakdown utilization values for an assortment of values of B.

B	Asymptotic Performance
1.0	1.000
1.5	.811
2.0	.693
3.0	.732
5.0	.773
10.0	.814
20.0	.844
40.0	.867
80.0	.885
100.0	.889

As B increases without limit, the limiting value of the breakdown utilization converges to 1. Nevertheless, the rate of convergence is quite slow. For randomly generated task sets consisting of a large number of tasks whose periods are drawn from a uniform distribution with largest period ratio ranging from 50 to 100, 88% is a good approximation to the threshold of schedulability for the rate monotonic algorithm. The slow rate of convergence means that a large task set is required for the breakdown utilization to be essentially equal to the constant specified in theorem 4. This underscores the importance of developing normal approximations where possible to sharpen the results in Theorem 4.

5. Summary and Conclusion

In this paper, we have provided an exact characterization and a stochastic analysis for a randomly generated set of periodic tasks scheduled by the rate monotonic algorithm. The stochastic analysis has shown that the average scheduling bound is usually much better than the worst case behavior as suggested by simulations. The formula of exact characterization can be used to check if a given task set is schedulable. The analysis can be generalized to allow for more general task deadlines, in particular when task deadlines need not be at the end of the task period. The analysis can also be applied to the case in which task synchronization must be considered, and when the priority ceiling protocol is used. Finally, the analysis can be carried out when there is insufficient priority granularity, that is, when there are fewer priority levels available than there are task periods. This arises in real-time communication where the number of priority bits is inherently limited.

References

1. Lehoczky, J. P. and Sha, L. "Performance of Real-Time Bus Scheduling Algorithms". *ACM Performance Evaluation Review, Special Issue Vol. 14, No. 1* (May, 1986).

2. Lehoczky, J. P., Sha L. and Strosnider, J. "Aperiodic Scheduling in A Hard Real-Time Environment". *Proc. IEEE Real-Time Systems Symp.* (1987), 261-270.

3. Liu, C. L. and Layland J. W. "Scheduling Algorithms for Multiprogramming in a Hard Real Time Environment". *JACM 20 (1)* (1973), 46 - 61.

4. Liu, J. W. S., K. J. Lin and S. Natarajan. "Scheduling real-time, periodic jobs using imprecise results". *Proceedings of the IEEE Real-Time Systems Symposium* (1987), 252-260.

5. R. Rajkumar. *Task synchronization in real-time systems.* Ph.D. Th., Carnegie Mellon University, August, 1989.

6. Sha, L., Lehoczky, J. P. and Rajkumar, R. "Solutions for Some Practical Problems in Prioritized Preemptive Scheduling". *IEEE Real-Time Systems Symposium* (1986).

7. Shorack, G. and J. Wellner. *Empirical Processes with Applications to Statistics.* John Wiley, New York, 1986.

8. Sprunt, B., Lehoczky, J. and L. Sha. "Exploiting unused periodic time for aperiodic service using the extended priority exchange algorithm". *Proceedings IEEE Real-Time Systems Symposium* (1988), 251-258.

9. Sprunt, B., Kirk, D. and L. Sha. "Priority-driven preemptive I/O controllers for real-time systems". *Proceedings of 15th International Symposium on Computer Architecture* (1988), 152-159.

10. Sprunt, B., Sha, L., and J. Lehoczky. "Aperiodic task scheduling for hard-real-time systems". *Real-Time Systems* (1989), 27-60.

11. Strosnider, J. K., Marchok, T. and J. Lehoczky. "Advanced real-time scheduling using the IEEE 802.5 Token Ring". *Proceedings of the IEEE Real-Time Systems Symposium* (1988), 42-52.