Acta Informatica 17, 31–41 (1982)

Scheduling with Slack Time*

C.L. Liu, Jane W.S. Liu, and Arthur L. Liestman
Department of Computer Science, University of Illinois, Urbana, Illinois 61801, USA

Summary. We consider a scheduling problem concerning a set of jobs $\{J_1, J_2, \ldots, J_n\}$ in which the job J_i requests C_i units of computation time every T_i units of time periodically. These jobs are to be executed by a time-shared single-processor computing system. It is assumed that the requests for each job arrive at the system at the beginning of the request periods and that deadline for completion of the requested computation in each period coincides with the beginning of the next period. For a set of jobs that is schedulable by a certain algorithm, the time span between the completion of a request and its deadline is referred to as the slack time of the request. It is shown here that when the set of jobs is scheduled according to the rate monotonic algorithm, the slack time of the first request of any job is no larger than the slack time of any subsequent request of that job. This result enables us to determine lower bounds to slack times of all requests.

1. Introduction

We consider here a scheduling problem in which a time-shared single-processor computing system is to execute a set of jobs, $J = \{J_1, J_2, \ldots, J_n\}$, each of which consists of a periodic sequence of requests. More specifically, a job J_i demands periodically, every T_i units of time, C_i units of computation time. We refer to $T_1, T_2, \ldots, T_n$ and $C_1, C_2, \ldots, C_n$ as the request periods and computation times of jobs $J_1, J_2, \ldots, J_n$, respectively. We assume here that the requests for each job arrive at the system at the beginning of request periods and that deadline for completion of the requested computation in each period coincides

* This work was partially supported by the Office of Naval Research under the Contract Number N00014-79-C-0775 and by NSF-MCS-8107647

0001-5903/82/0017/0031/$02.20

with the beginning of the next period. Hence, in our model, each job is characterized by an ordered pair of numbers (C_i, T_i). This model of processor work-load has been used extensively in the performance analysis of hard-real-time systems [1–5]. In such systems, requests for all jobs for which hard deadlines exist are usually periodic. Moreover, deadlines are imposed primarily to satisfy the run-ability constraints [3].

Throughout our discussion we assume that $T_1 \leqq T_2 \leqq \ldots \leqq T_n$ for convenience of notation. We use $u_1, u_2, \ldots, u_n$ to denote the ratios C_1/T_1, $C_2/T_2, \ldots, C_n/T_n$. Let u denote the sum $\sum_{i=1}^{n} C_i/T_i$. u is referred to as the utilization factor of the set of jobs $\{J_1, J_2, \ldots, J_n\}$.

By scheduling a set of jobs we mean, as usual, to determine the particular request to which the processor should be devoted at any time instant. A set of jobs is said to be *schedulable* by a certain scheduling algorithm if the computation of each request can be completed prior to the arrival of the next request of the same job. Throughout our discussion, we assume that a preemptive scheduling discipline is employed. Thus, a request of C_i units of computation time can be satisfied by one or more quanta of time which sum to C_i.

For a set of jobs that is schedulable by a certain algorithm, it is possible that a request be satisfied ahead of its deadline. We refer to the time span between the completion of a request and its deadline as the *slack time* of the request. In this paper, we investigate the problem of estimating the slack time of each request when the set of jobs is scheduled according to the rate monotonic scheduling algorithm. According to this algorithm higher priority is assigned to a request with shorter request period. Thus, the requests of a job with shorter request period will always preempt any request of another job with longer request period. It has been shown that [3] a set of n jobs with a utilization factor less than or equal to $n(2^{1/n}-1)$ is always schedulable by the rate monotonic scheduling algorithm.

The need to determine lower bounds to slack times arises in fault-tolerant real-time systems in which some form of deadline mechanism is used to guarantee reliability [6]. The deadline mechanism provides two algorithms for each job. The primarily algorithm, which produces good quality responses but requires longer and generally variable amounts of computation time, is normally not used. Rather, an alternate algorithm which produces acceptable responses and requires known computation time is used. (In our model, C_i corresponds to the time required to carry out the alternate algorithm for the job J_i.) However, if within a request period the slack time is sufficient and the processor is idle, the primary algorithm may be invoked to iteratively improve the response during that request period. In general, the knowledge of minimum slack time for all jobs in J allows the scheduler to selectively assign the processor to tasks other than those in J whenever the processor becomes idle.

In Sect. 2, we show that the slack time of the first request of any job is no larger than the slack time of any subsequent request of that job. This result enables us to determine lower bounds to slack times of all requests. These lower bounds are presented in Sects. 3 and 4.

2. A Fundamental Theorem

Without loss of generality, we assume that the first requests of all jobs in J arrive at $t=0$. As will be shown below, this assumption covers the worst possible case. In order to compare the slack times of requests in different request periods, we define the availability function $f(t)$ to be

$$f(t)=\begin{cases}0 & \text{if the processor is occupied at } t\\ 1 & \text{if the processor is not occupied at } t\end{cases}$$

with respect to a set of job, $J=\{J_1, J_2, \ldots, J_n\}$, scheduled according to some priority-driven scheduling algorithm in the time interval $[0, t]$. Clearly, for any Δ and τ, the integral

$$\int_{\Delta}^{\Delta+\tau} f(t)\,dt$$

gives the total units of time that the processor is not occupied in the time interval $[\Delta, \Delta+\tau]$. Consider a sequence of demands of computation time (not necessarily periodic) within the interval $[0, t_k]$ where d_1 units of computation time is demanded within the time interval $[0, t_1]$, and d_i units of computation time is demanded within the time interval $[t_{i-1}, t_i]$ for $i=2, 3, \ldots, k$. This sequence of demands can be denoted by

$$D=(d_1, [0, t_1]), (d_2, [t_1, t_2]), \ldots, (d_k, [t_{k-1}, t_k]).$$

We shall use D^{Δ} to denote the sequence of demands,

$$D^{\Delta}=(d_1, [\Delta, t_1+\Delta]), (d_2, [t_1+\Delta, t_2+\Delta]), \ldots, (d_k, [t_{k-1}+\Delta, t_k+\Delta])$$

which is D delayed by Δ.

Suppose that we are to schedule within the time interval $(0, t_k)$ not only requests of jobs in J but aiso the sequence of demands, D. Furthermore, suppose that all demands in D have lower priorities than requests of jobs in J. Clearly, if the demands in D can be satisfied as well for a given availability function $f(t)$, the demands in D will take up the first d_1 units of time within the time interval $[0, t_1]$ in $f(t)$, d_2 units of time in $[t_1, t_2]$, and so on. We shall use

$$f(t)-D(t)$$

to denote the available time that remains after all the demands in D are satisfied. (This is a slight abuse of notation, since straightly speaking a sequence of demands is not a function of time. Rather, D induces a corresponding $D(t)$ for a given $f(t)$, and will induce a different $D(t)$ for a different $f(t)$.) The following Lemmas are obvious:

Lemma 1. For any $f(t)$ and any D restricted to within the time interval $[0, \tau]$, if

$$\int_{\Delta_1}^{\Delta_1+\tau} f(t)\,dt \leqq \int_{\Delta_2}^{\Delta_2+\tau} f(t)\,dt$$

then

$$\int_{\Delta_1}^{\Delta_1+\tau} [f(t)-D^{\Delta_1}(t)]\,dt \leqq \int_{\Delta_2}^{\Delta_2+\tau} [f(t)-D^{\Delta_2}(t)]\,dt.$$

Let $D=(d_1,[0,t]),(d_2,[t_1,t_2]),\ldots,(d_{k-1},[t_{k-1},t_k])$ be a sequence of demands. Let E be a sequence of demands which is identical to D^{δ} in the time interval $[\delta,\delta+t_k]$ and no computation time is demanded in $[0,\delta]$. Then, we have

Lemma 2. For any τ,

$$\int_{\Delta}^{\Delta+\tau} [f(t)-D^{\Delta}(t)]\,dt \leqq \int_{\Delta}^{\Delta+\tau} [f(t)-E^{\Delta}(t)]\,dt.$$

We prove now a fundamental theorem which will be needed in our later discussion:

Theorem 1. For any set of n jobs $J_1,J_2,\ldots,J_n$ scheduled by the rate monotonic scheduling algorithm, we have

$$\int_0^{\tau} f(t)\,dt \leqq \int_{\Delta}^{\Delta+\tau} f(t)\,dt \tag{1}$$

for any Δ and τ.

Proof. The theorem is proved by induction on the number of jobs, n. As the basis of induction, we note that for $n=1$, $f(t)$ is a periodic function of period T_1 so that $f(t)=0$ for the first C_1 units of time and $f(t)=1$ for the next T_1-C_1 units of time in each period. Clearly,

$$\int_0^{\tau} f(t)\,dt \leqq \int_{\Delta}^{\Delta+\tau} f(t)\,dt.$$

To carry out the induction step, we assume that the theorem is true when $f(t)$ is the availability function after $n-1$ jobs have been scheduled. Let $\hat{f}(t)$ denote the availability function after n jobs have been scheduled. Consider first the simple case that Δ is a multiple of the longest request period, T_n. Let

$$D=(C_n,[0,T_n]),(C_n,[T_n,2T_n]),\ldots,(C_n,[(k-1)T_n,kT_n]) \tag{2}$$

such that $kT_n \geqq \tau$. According to the induction hypothesis

$$\int_0^{\tau} f(t)\,dt \leqq \int_{\Delta}^{\Delta+\tau} f(t)\,dt.$$

According to Lemma 1

$$\int_0^{\tau} [f(t)-D(t)]\,dt \leqq \int_{\Delta}^{\Delta+\tau} [f(t)-D^{\Delta}(t)]\,dt$$

which is

$$\int_0^{\tau} \hat{f}(t)\,dt \leqq \int_{\Delta}^{\Delta+\tau} \hat{f}(t)\,dt$$

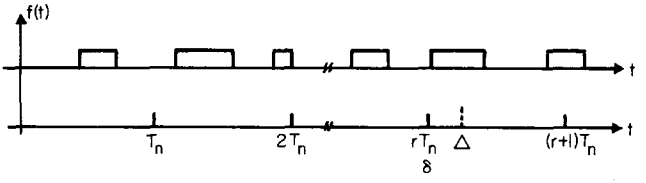

Fig. 1

Now consider the case that Δ is not a multiple of T_n as illustrated in Fig. 1. Let δ be a multiple of T_n, $\delta = r\,T_n$, such that $\delta < \Delta < (r+1)\,T_n$. We examine two cases:

Case 1. $\hat{f}(t) = 0$ for $\delta \leqq t \leqq \Delta$. In this case

$$\int_{\delta}^{\delta+\tau} \hat{f}(t)\,dt \leqq \int_{\Delta}^{\Delta+\tau} \hat{f}(t)\,dt. \tag{3}$$

According to the induction hypothesis

$$\int_{0}^{\tau} f(t)\,dt \leqq \int_{\delta}^{\delta+\tau} f(t)\,dt.$$

Moreover, according to Lemma 1

$$\int_{0}^{\tau} [f(t) - D(t)]\,dt \leqq \int_{\delta}^{\delta+\tau} [f(t) - D^{\delta}(t)]\,dt.$$

That is,

$$\int_{0}^{\tau} \hat{f}(t)\,dt \leqq \int_{\delta}^{\delta+\tau} \hat{f}(t)\,dt. \tag{4}$$

Combining (3) and (4), we obtain

$$\int_{0}^{\tau} \hat{f}(t)\,dt \leqq \int_{\Delta}^{\Delta+\tau} \hat{f}(t)\,dt.$$

Case 2. $\hat{f}(t) \neq 0$ for $\delta \leqq t \leqq \Delta$. That $\hat{f}(t) \neq 0$ for $\delta \leqq t \leqq \Delta$ implies the request of J_n within the time interval $[r\,T_n, (r+1)\,T_n]$ is satisfied at or prior to $t = \Delta$. Thus, within the time interval $[\Delta, (r+1)\,T_n]$, $\hat{f}(t) = f(t)$. Let λ denote $(r+1)\,T_n - \Delta$. Let

$$E = (0, [0, \lambda]), (C_n, [\lambda, T_n + \lambda]), \ldots, (C_n, [(k-1)\,T_n + \lambda, k\,T_n + \lambda]).$$

That is, E is the sequence D in (2) delayed by λ. Thus, we can write

$$\int_{\Delta}^{\Delta+\tau} \hat{f}(t)\,dt = \int_{\Delta}^{\Delta+\tau} [f(t) - E^{\Delta}(t)]\,dt. \tag{5}$$

According to Lemma 2

$$\int_{\Delta}^{\Delta+\tau} [f(t)-D^{\Delta}(t)]\,dt \leqq \int_{\Delta}^{\Delta+\tau} [f(t)-E^{\Delta}(t)]\,dt. \tag{6}$$

However, according to Lemma 1 and the induction hypothesis

$$\int_{0}^{\tau} \hat{f}(t)\,dt \leqq \int_{\Delta}^{\Delta+\tau} [f(t)-D^{\Delta}(t)]\,dt. \tag{7}$$

Combining (5)-(7), we obtain

$$\int_{0}^{\tau} \hat{f}(t)\,dt \leqq \int_{\Delta}^{\Delta+\tau} [f(t)-E^{\Delta}(t)]\,dt = \int_{\Delta}^{\Delta+\tau} \hat{f}(t)\,dt. \quad \square$$

We note that the inequality in (1) holds in particular for $\Delta = r T_i$, $i = 1, 2, \ldots, n$ and $r = 1, 2, 3 \ldots$. Thus, we have

Corollary 1. The slack time of the first request of any job is smaller than or equal to the slack times of all subsequent requests of that job.

3. Estimation of Slack Time

We now use the result in Theorem 1 to derive lower bounds on the slack time of a request. Theorems 2 and 3 are proven by induction on the number of jobs. We first show a lemma which is used as the basis of induction in these proofs.

Lemma 3. Let J_1 and J_2 be two jobs with $u \leqq 2(2^{1/2}-1)$ scheduled according to the rate monotonic scheduling algorithm. For an arbitrary request of J_2, let q denote the size of the last quantum of time allocated to that request and s denote the length of the slack time. Then $s \geqq 0.207\,q$.

Proof. There are two cases:

Case 1. The execution of the last quantum of the request begins earlier than T_1 units of time after the arrival of the request, as illustrated in Fig. 2(a). In this case, since $a \leqq T_1$, during this a units of time atmost C_1 units of computation time was devoted to J_1. Thus we have,

$$s \geqq T_2 - (C_1 + C_2) = T_2\left(1 - \frac{C_1 + C_2}{T_2}\right) \geqq T_2\left[1 - \left(\frac{C_1}{T_1} + \frac{C_2}{T_2}\right)\right]$$

$$= T_2(1-u) = C_2\left(\frac{1-u}{u_2}\right) \geqq q\left(\frac{1-u}{u_2}\right) \geqq q\left(\frac{1-u}{u}\right) \geqq q\left(\frac{1-0.83}{0.83}\right) = 0.207\,q$$

Case 2. The execution of the last quantum of the request begins later than T_1 units of time after the arrival of the request, as illustrated in Fig. 2(b). Let b denote $C_2 - q$. We have

$$u_2 = \frac{b+q}{a+q+s}$$

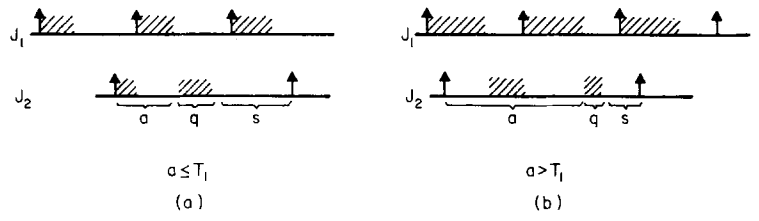

Fig. 2

or

$$s\,u_2 = q(1-u_2) + b - a\,u_2. \tag{8}$$

Consider a job that has a request period equal to a and computation time equal to b. Suppose that we want to schedule the two jobs (C_1, T_1) and (b, a) according to the rate monotonic scheduling algorithm. According to Fig. 2(b), and Theorem 1, the total processor time available for J_2 in the time interval $[0, a]$ is less than or equal to b after J_1 was scheduled. Consequently, either the set $\{(C_1, T_1), (b, a)\}$ fully utilizes the processor[1] or the set is not schedulable at all. In either case, we have:

$$u_1 + \frac{b}{a} \geqq 2(2^{1/2} - 1) \geqq u.$$

That is,

$$b - a\,u_2 \geqq 0.$$

Consequently, (8) yields

$$s\,u_2 \geqq q(1-u_2)$$

or

$$s \geqq q\left(\frac{1-u_2}{u_2}\right) \geqq 0.207\,q. \quad \square$$

Theorem 2. Let $J_1, J_2, \ldots, J_n$ be n jobs scheduled according to the rate monotonic algorithm. Let q denote the length of the last quantum of time allocated to the first request of J_n. If $u \leqq n(2^{1/n} - 1)$ then the slack time of *any* request of J_n is larger than or equal to $0.207\,q$.

Proof. According to Theorem 1, the slack time of the first request of J_n is less than or equal to the slack time of any request of J_n. Thus, it is sufficient to prove that the slack time of the first request is larger than or equal to $0.207\,q$.

The theorem is proved by induction on n. Lemma 3 provides the basis of induction. As to the induction step, we assume that the theorem is true for $n-1$ jobs. Let us use q and s to denote the length of the last quantum of time allocated to the first request of J_n and the slack time of this request. We consider two cases:

[1] A set of jobs is said to fully utilize the processor with respect to a particular scheduling algorithm if the set of jobs is schedulable by the algorithm; however, increasing the computation time of any of the jobs in the set will cause the set to become unschedulable

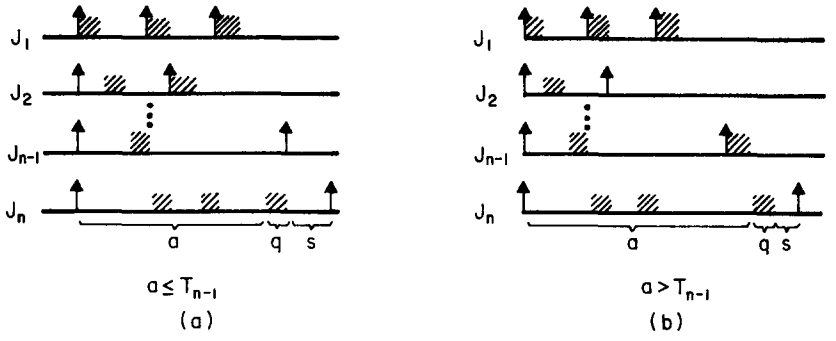

Fig. 3

Case 1. The execution of the last quantum of time allocated of the first request for J_n begins at no later than $t = T_{n-1}$ as illustrated Fig. 3(a).

Consider the set of $n-1$ jobs (C_1, T_1), $(C_2, T_2), \ldots, (C_{n-2}, T_{n-2})$, $(C_{n-1} + C_n, T_n)$. According to the diagram in Fig. 3(a) and Theorem 1, this set of $n-1$ jobs is schedulable by the rate monotonic scheduling algorithm. Furthermore, the first request of the job $(C_{n-1} + C_n, T_n)$ will occupy the processor during the time intervals in which the first request of J_{n-1} and the first request of J_n occupy the processor when the jobs $J_1, J_2, \ldots, J_{n-1}, J_n$ were scheduled. Let q' and s' denote the length of the last quantum of time allocated and the slack time of the first request of the job $(C_{n-1} + C_n, T_n)$. Then according to the induction hypothesis

$$s' \geqq 0.207\, q'.$$

However, since

$$s' = s \quad \text{and} \quad q' \geqq q$$

we have

$$s \geqq 0.207\, q.$$

Case 2. The execution of the last quantum of time allocated to the request begins later than $t = T_{n-1}$ as illustrated in the diagram in Fig. 3(b). Let b denote $C_n - q$. We have

$$u_n = \frac{b+q}{a+q+s}$$

which can be written as

$$s\, u_n = q(1-u_n) + b - a\, u_n. \tag{9}$$

Consider the n jobs (C_1, T_1), $(C_2, T_2), \ldots, (C_{n-1}, T_{n-1})$, (b, a). According to the diagram in Fig. 3(b), this set of jobs fully utilizes the processor with respect to the rate monotonic scheduling algorithm. (The processor was never left idle within the time interval $[0, a]$. Because if this was not the case, the first request

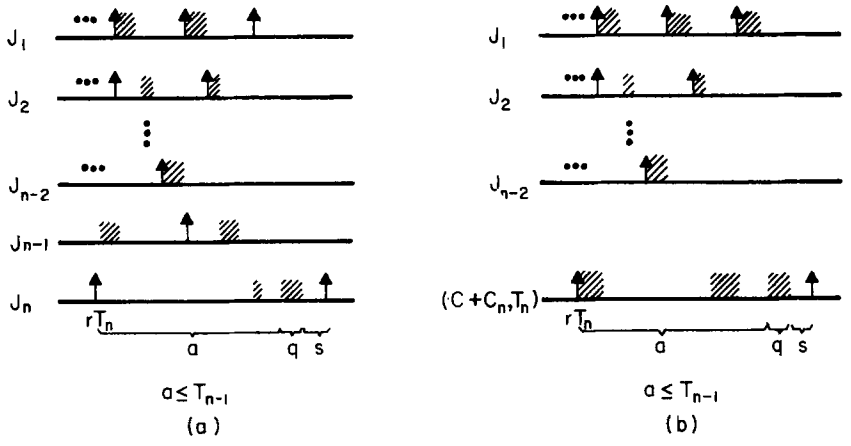

Fig. 4

of J_n will be completed earlier). It follows that

$$\frac{b}{a} \geqq u_n$$

or

$$b - a u_n \geqq 0.$$

It follows that (9) can be written as

$$s u_n \geqq q(1-u_n)$$

or

$$s \geqq q\left(\frac{1-u_n}{u_n}\right) \geqq q\left(\frac{1-u}{u}\right) \geqq 0.207\, q. \quad \square$$

Because of Theorem 1, we may expect that the last quantum of time allocated to any request of a job to be longer than the last quantum of time allocated to the first request of that job. In the following, we obtain a lower bound to the slack time of any request of job J_i in terms of the length of the last quantum allocated to that request.

Theorem 3. Let $J_1, J_2, \ldots, J_n$ be n jobs scheduled according to the rate monotonic algorithm. For any request of J_n, let q and s denote the length of the last quantum of time allocated to that request and the length of the slack time, respectively. If $T_n \geqq 2T_{n-1}$ and $u \leqq n(2^{1/n}-1)$, then $s \geqq 0.207\, q$.

Proof. The proof of the theorem is carried out by induction on n. Lemma 3 provides the basis of induction. As to the induction step we consider two cases:

Case 1. The execution of the last quantum of time allocated to the request begins at no more than T_{n-1} units of time after the arrival of the request as illustrated in Fig. 4(a).

Let C denote the total computation time allocated to J_{n-1} within the time interval $[rT_n, rT_n + a]$. Since $a \leqq T_{n-1}$, we have $C \leqq 2C_{n-1}$. Now consider the $n-1$ jobs $(C_1, T_1), (C_2, T_2), \ldots, (C_{n-2}, T_{n-2}), (C + C_n, T_n)$. Let q' and s' denote the length of the last quantum of time allocated and the slack time of any request of the job $(C + C_n, T_n)$, respectively. Since

$$\sum_{i=1}^{n-2} u_i + \frac{C + C_n}{T_n} \leqq \sum_{i=1}^{n-2} u_i + \frac{2C_{n-1}}{T_n} + \frac{C_n}{T_n} \leqq \sum_{i=1}^{n-2} u_i + \frac{2C_{n-1}}{2T_{n-1}} + \frac{C_n}{T_n} \leqq u$$

these $n-1$ jobs are schedulable by the rate monotonic scheduling algorithm. Moreover, the diagram for the time interval $[rT_n, (r+1)T_n]$ will be that shown in Fig. 4(b). By the induction hypothesis

$$s' \geqq 0.207 q'.$$

Since

$$s' = s \qquad q' \geqq q$$

we have

$$s \geqq 0.207 q.$$

Case 2. The execution of the last quantum of time allocated to the request begins no later than T_{n-1} units of time after the arrival of the request. The derivation of the bound for this case is similar to case 2 in the proof of Theorem 2, and will not be repeated here. □

4. Remarks

One of the advantages of the rate monotonic algorithm over other scheduling algorithms is its simplicity in practical implementation. In this paper, we also observe that the rate monotonic algorithm produces a schedule which is highly regular, and, consequently, it is possible to carry out a detailed analysis on the slack times of all requests. Corollary 1 provides a lower bound on the slack time of any request by the slack time of the first request of the same job. Note that the slack time of the first request of J_n can either be determined by an explicit construction or be estimated by

$$s \geqq T_n - C_n - \left(\left\lceil \frac{T_n}{T_1} \right\rceil C_1 + \left\lceil \frac{T_n}{T_2} \right\rceil C_2 + \ldots + \left\lceil \frac{T_n}{T_{n-1}} \right\rceil C_{n-1} \right).$$

Furthermore, our results in Theorem 2 and 3 give lower bounds of the slack time of a request as a function of the last quantum of computation time allocated to the first request or that particular request. The knowledge of these lower bounds allows the scheduler to determine dynamically during each request period whether further refinement of the result produced that period can be carried out (depending on the slack time of the request and the slack times of all requests of lower priorities.)

One might wonder about the reason to express the lower bound of the slack time of a request in terms of the last quantum of time allocated to that request. One explanation is that, depending on the values of the computation times and the periods of the jobs, the slack time of a request can be a very small but non-zero quantity. In our lower bound, this fact will be reflected in the size of the last quantum of computation time allocated to the request. From a practical point of view, it is quite possible that after the last quantum of computation time, a certain amount of "slack time" is needed for other activities (such as input/output operations) before the arrival of the next request. Our lower bound indicates that the larger the last quantum of computation time is, the more "breathing room" we can expect to have.

Clearly, one can modify our model such that each request has a deadline that is ahead of the arrival time of the next request. In this way, a "slack time" equal to the difference between the deadline of a request and the arrival time of the next request is guaranteed. However, there are two differences between such a model and our model. In the first place, the scheduling strategies might no longer be the same since in our model, priorities are assigned according to the request rates while in the other model priorities can be assigned according to both the request rates and the deadlines of the requests. In the second place, in the modified model, schedulability can be determined only when request rates and deadlines of the jobs are given. While in our model the slack time we determined can be considered as a bound on how much we can move the deadline ahead of the arrival time of the next request as in the modified model.

References

1. Dertouzos, M.L.: Control robotics: the procedural control of physical processes. IFIP Proc. 807–813 (1974)
2. Labetoulle, J.: Some theorems on real time scheduling. Computer architectures and networks, p. 285–298. E. Gelenbe and R. Mahl eds. Amsterdam: North-Holland Publ. Co. 1974
3. Liu, C.L., J.W. Layland: Scheduling algorithms for multiprogramming in hard-real-time environment. JACM, Jan. 1973, p. 46–61
4. Serlin, O.: Scheduling of time critical processes. Proc. of the Spring Joint Computer Conference, 1972, pp. 925–932
5. Dhall, S.K., C.L. Liu: On a real-time scheduling problem. Operations Research, Vol. 26, No. 1, 1978, pp. 127–140
6. Campbell, R.H., K.H. Horton, G.G. Belford: Simulation of a Fault-tolerant deadline mechanism. Proc. of 1979 International Symposium on Fault-Tolerant Computing, June, 1979

Received March 12, 1981/August 28, 1981