TR-94-008

Minimum Delay of Nonpreemptive Real-Time Schedulings

Chia-Hsiang Chang, Jan-Ming Ho,
Ming-Tat Ko, Kuo-Hui Tsai, Da-Wei Wang

TR-94-005

Minimum Delay of Nonpreemptive Real-Time Schedulings *

Chia-Hsiang Chang, Jan-Ming Ho, Ming-Tat Ko, Kuo-Hui Tsai
Da-Wei Wang
Institute of Information Science
Academia Sinica, Taiwan
R.O.C.

Abstract

In traditional hard realtime scheduling problems on a set of periodical tasks, denoted as $\{\tau_i\}$, deadline for the CPU to complete the computation of a particular request of task τ_i is usually defined as being the same as its period. A feasible schedule satisfying this type of deadline constraint tends to under-utilize CPU bandwidth especially when the tasks are non-preemptive. For realtime applications with less stringent deadline constraints, e.g., a worst-case delay guarantee rather than a hard deadline constraint, better utilization of CPU bandwidth is achievable.

In this paper, we study a family of non-preemptive scheduling algorithms in which no inserted idle time are allowed, i.e., CPU is activated as long as computation requests are pending. We show that a request always receives CPU service in a finite delay, denoted as queuing delay, if and only if CPU utilization is no greater than 1. The FCFS discipline is shown to minimize maximum queuing delay. Queuing delay of other scheduling policies, e.g., rate-monotonic, fixed priority, and earliest deadline first, etc., are also analyzed.

1 Introduction

In this paper, we consider nonpreemptive periodic task scheduling problem. A periodic task is an infinite series of requests for the same computation with a constant inter-arrival time.

*This work has been supported in part by National Science Council under Grant NSC83-0408-E-001-001.

A periodic task τ is denoted by a triple (T, c, s), where T is its period or the inter-arrival time of requests, c is its computation time for a request and s is its initial request time. Let $\tau_i = (T_i, c_i, s_i), i = 1, 2, \cdots, n$ be a set of n periodic tasks. In traditional realtime scheduling problems, a deadline d_i is given for each τ_i and the problem is to determine if the task set is schedulable, i.e., there exist a schedule of tasks such that any request of τ_i released at time t is processed and completed before time $t + d_i$. Deadline d_i is typically defined as the period T_i of task τ_i.

In the hard real-time applications [4], the schedulers must guarantee the given deadline of each periodic task at the cost of sacrificing CPU untilization. However, in some soft real-time applications, such as continuous media communication and video-on-demand services [6, 5, 1, 2], any request missing its deadline just causes unpleasant results. It is proper to refer deadline as delay for soft real-time applications. For soft real-time applications, a less strict bounded delay is tolerable and usually these delay bounds are several times larger than the periods of the tasks. These two types of realtime applications will be referred to as realtime applications with *hard deadlines*, and with *soft deadlines*, respectively. For the soft-deadline applications, we are going to show that a utilization factor of 1 is achievable. Note that an obvious necessary condition for a task set to be schedulable is that the *utilization factor*, $\sum_{i=1}^{n} c_i/T_i$, be no greater than one; otherwise, it can be shown that the queue of requests waiting for processing grows indefinitly. It implies that at least one of the tasks can not meet any finite deadline. For the hard-deadline realtime scheduling problem, one wants to find a feasible schedule satisfying respective deadlines for each task such that CPU utilization is maximized; while for the soft-deadline problems, we regard deadline as a minimization criteria.

In this paper, we study a family of non-preemptive scheduling algorithms in which no inserted idle time are allowed, i.e., CPU is activated as long as computation requests are pending. We show that a request always receives CPU service in a finite delay, denoted as queuing delay, if and only if CPU utilization is no greater than 1. The FCFS discipline is shown to minimize maximum queuing delay. Queuing delay of other scheduling policies, e.g., rate-monotonic, fixed priority, and earliest deadline first, etc., are also analyzed. In practice, these algorithms are preferred because they are simple and are easy to compute on-line.

In section 2, we show that a request always receives CPU service in a finite delay, denoted as queuing delay, if and only if CPU utilization is no greater than 1. In sections 3, 4, and 5, delay bounds of several popular scheduling policies are given. Specifically, we show that the delay bound of the FCFS policy is a global minimum. In section 6, concluding remarks are given.

2 Upper Bound

To guarantee fully utilization of the processor, scheduling strategy not allowing inserted idle time is prefered. For any scheduling strategy not allowing inserted idle time, we have the following general upper bound of the minimum delay.

Theorem 1 *The minimum delay is less that or equal to the least common multiple (LCM) of the periods.*

Proof: Suppose the contrary. Let r be a request of τ_i released at time t which is the first request of response time longer than LCM. Without loss of generality, we may assume that before time t there is no idle time and the initial time is 0. Let $kLCM \leq t < (k+1)LCM$ and r is the sth request of τ_i. Thus, $s \leq (k+1)LCM/T_i$. By the assumption, in the time interval $[0,(k+1)LCM)$, there are at most $(k+1)LCM/T_j$ requests of τ_j are completed for $j \neq i$, at most $s-1$ requests of τ_i are completed and the Sth request of τ_i is processed partially. Thus, the total used time of the processor in the time interval $[0,(k+1)LCM)$ is less than $\sum_{j\neq i} \frac{(k+1)LCM}{T_j} c_j + sc_i \leq (k+1)LCM$. It means that there is idle time of the processor in $[0,(k+1)LCM)$ which is a contradiction. Q.E.D.

3 First-Come-First-Serve Strategy

Theorem 2 *For the first-come-first-serve strategy, the minimum delay is $\sum_{i=1}^{n} c_i$ for all τ_i.*

Proof: Let d_i denote the minimum delay of τ_i. By considering the case that all the tasks come to give a request at the same instant, t, the latest processed request is completed at time $t + \sum_{i=1}^{n} c_i$. Thus, we obtain that $d_i \geq \sum_{i=1}^{n} c_i$. In the following, let us prove that each request can be completed in $d = \sum_{i=1}^{n} c_i$ time. In other words, if d is set to be the deadline, the task system is schedulable. Suppose the contrary. Let τ_j be the first task misses the deadline at time t. We may assume that from the beginning, there is no idle time. Otherwise, we consider all the requests from the end of the latest idle time. Without loss of generality, assume the starting time is 0. Let J be all the requests issued at or before time $t-d$. Since the first task misses the deadline at time t, thus the missed request is issued at time $t-d$ and the scheduling rule is first-come-first-serve, only the requests in J are processed during time 0 to time d. From time 0 to time $t-d$, a task τ_i issues at most $\lfloor\frac{t-d}{T_i}\rfloor+1$ requests. Thus, the computation time for the requests in J is at most

$$\sum_{i=1}^{n}(\lfloor\frac{t-d}{T_i}\rfloor+1)c_i \leq \sum_{i=1}^{n}(\frac{t-d}{T_i}+1)c_i \leq t-d+\sum_{i=1}^{n}c_i \leq t.$$

Since the last request of task τ_j misses deadline and there is no idle time from time 0 to time t, we obtain a contradiction. Q.E.D.

4 Earliest-Next-Request-First Strategy

In the earliest-next-request-first strategy (ENRF), a request of higher priority if its next request time is earlier. For a periodic task system in which the deadline of each task is equal to its respective period, the ENRF is the same as the well-known earliest deadline first scheduling (EDF) [4].

Let tasks $\tau_i, i=1,2,\cdots,n$, be indexed such that $T_i \leq T_{i+1}$. For the special case that the deadline, d_i equal to T_i, Jeffay et al proved the following theorem [3].

Theorem 3 ([3]) *The task system is schedulable if and only if*

for all $i, 1 \leq i \leq n$; *for all* $L, T_1 < L < T_i$: $L \geq c_i + \sum_{j=1}^{i-1}\lfloor\frac{L-1}{T_j}\rfloor c_j$.

In addition, it is proved that if the task system is schedulable, then the earliest deadline first strategy can always give a proper schedule [3]. However, it is also shown that there are unschedulable task systems of arbitrary small utilization factor. Even for the task system of constant computation time, the utilization factor needs to be less than or equal to 1/2 to guarrantee schedulability.

Theorem 4 *Let $\{\tau_i\}_{i=1}^n$ be a task system of constant computation time. If the utilization factor, $\sum \frac{c_i}{T_i} \leq \frac{1}{2}$ then the task system is schedulable by EDF.*

Proof: Let c denote the computation time. Since the utilization factor is less than or equal to $\frac{1}{2}$, $2c \leq T_1$. The theorem is followed by verifying the condition in Theorem 3. Let $u = \sum_{j=1}^{n-1} \frac{c(L-1)}{T_j}$ and let L_0 be the solution of equation $L = c + \sum_{j=1}^{n-1} \frac{c(L-1)}{T_j}$. Since $u \leq \frac{1}{2}$, $L_0 = \frac{c-u}{1-u} \leq 2c \leq T_1$. It is obvious that for $L \geq L_0$, $L \geq c_i + \sum_{j=1}^{i-1} \lfloor \frac{L-1}{T_j} \rfloor c_j$. Thus, the condition in Theorem 3 is satisfied. Q.E.D.

An easy example shows that 1/2 is an upper bound of the utilization factor of a constant computation time task system to be schedulable. For a number $u = 1/2 + x, x > 0$, consider a constant computation time task system of two tasks. The computation time is $1 + x$ and the first task is of period 2 and the second is of a very long period. When the second task release a request just before the first task does, it is obvious that the first task will miss its deadline.

Theorem 5 *The minimum delay d_j for τ_j is less than or equal to the maximum, m_j, of T_j*

and

$$\max_{k>j} \max_{T_j<T<T_k} \{\sum_{p<k} \lfloor \frac{T-1}{T_p} \rfloor c_p + c_k + T_j - T\}.$$

Proof: For any request r of task τ_j, we will prove that its response time is less than or equal to m_i for all $i = 1, \cdots, n$. Let $[t_b, t_e]$ be the maximum time interval having no idle time in which the request r is processed.

Case 1: There is no request of next request later than $r + T_j$ is processed before r is processed . All the computation time required by the requests with their next requests earlier than $r + T_j - t$ is

$$\sum_{i=1}^{n} \lfloor \frac{r + T_j - t_b}{T_i} \rfloor c_i.$$

Thus, the response time of r is less than

$$\sum_{i=1}^{n} \lfloor \frac{r + T_j - t_b}{T_i} \rfloor c_i + t_b - r.$$

Since $\sum_{i=1}^{n} \frac{c_i}{T_i} \leq 1$, the response time is less than T_j.

Case 2: There are some requests of their next requests later than $r + T_j$ are processed before r is processed. Suppose that request r' of task τ_k is the last such request and let s be its starting time. Notice that all the requests processed later than $s + c_k$ are released after time s and all the requests of task $\tau_i, i > k$ are not processed before r is processed. Thus, the response time of r is

$$\sum_{p<k} \lfloor \frac{r + T_j - s - 1}{T_p} \rfloor c_p + c_k + s - r.$$

Let $T = r + T_j - s$. The response time becomes to be

$$\sum_{p<k} \lfloor \frac{T-1}{T_p} \rfloor c_p + c_k + T_j - T.$$

Since r is later than s and there are only one request of task τ_k is processed, $T_j < T < T_k$. We have that the response time is less than or equal to m_j. Q.E.D.

5 Rate Monotonic Scheduling

The rate monotonic scheduling is a fixed priority scheduling strategy [4]. For a set of periodic tasks, in the rate monotonic scheduling, a task of longer period is of lower priority. Let tasks $\tau_i, i = 1, 2, \cdots, n$, be indexed such that $T_i \leq T_{i+1}$ and m is the largest index that $\{\tau_i\}_{i=1}^{m}$ is rate monotonic schedulable as a preemptive task system. Let $M_i = \max_{j>i}\{c_j\}$ for $i \neq n$ and $M_n = 0$.

Theorem 6 *The minimum delay , d_i, of τ_i, for $i = 1, \cdots, m$, is the smallest solution of the following equation*

$$t - c_i = M_i - 1 + \sum_{j=1}^{i-1} (\lfloor \frac{t - c_i}{T_j} \rfloor + 1) c_j. \quad (1)$$

Proof: For any request r of task τ_i, we will prove that its response time is less than d_i for all $i = 1, \cdots, n$. Let $[t_b, t_e]$ be the maximal time interval in which the request r is processed. has no idle time. Let r be released at time t. Time t_b is the end of the last idle time before time t and t_e is the start of the first idle time after time t.

Case 1: There are requests of lower priority processed before t in $[t_b, t_e]$.

Let r', a request of τ_k, be the latest request of lower priority than τ_i processed before time t. For simplicity of notation, assume that the execution of r' starts at time $t_0 = -1$. The requests processed from time $c_k - 1$ to the start time of r are of priority higher than or equal to τ_i. Let the request r is the sth request of τ_i processed in $[t_0, t_e]$. In the following, we will prove the theorem by induction on s. When $s = 1$, the completion time of r is the smallest solution of the following equation

$$t - c_i = c_k - 1 + \sum_{j=1}^{i-1}(\lfloor\frac{t - c_i - s_j}{T_j}\rfloor + 1)c_j,$$

where $s_j > 0$ is the first release time of τ_j in the time interval. Since $c_k \leq M_i$ and $s_j > 0$, the smallest solution, t_1, of the above equation is obviously less than d_i. If the release time of r is later than $t_1 - c_i$, a request of lower priority or an idle time interval starts at T_1, which is contradicts the assumption. Thus, the release time of r is at or earlier than $t_1 - c_i$ and its response time is less than d_i. Let t_m be the smallest solution of

$$d - c_i = c_k - 1 + (m-1)c_i + \sum_{j=1}^{i-1}(\lfloor\frac{d - c_i - s_j}{T_j}\rfloor + 1)c_j.$$

By induction assumption, time t_{s-1} is the completion time of the $(s-1)th$ request of τ_i and the response time is less than d_i. We proceed to prove that the response time of the sth request of τ_i is also less than d_i. In fact, we will prove that $t_s \leq t_{s-1} + T_i$. Let $t_s = t_{s-1} + d$. Thus, $d > 0$ is the smallest solution of the following equation

$$c_k - 1 + sc_i + \sum_{j=1}^{i-1}(\lfloor\frac{t_{s-1} + d - c_i - s_i}{T_j}\rfloor + 1)c_j - t_{s-1} - d = 0. \quad (2)$$

The left side of equation 2 is equal to

$$c_k - 1 + (s-1)c_i + \sum_{j=1}^{i-1}(\lfloor\frac{t_{s-1}-c_i-s_i}{T_j}\rfloor + 1)c_j + \sum_{j=1}^{i-1}\lfloor\frac{d+\Delta_j^s}{T_j}\rfloor c_j + c_i - d \quad (3)$$

$$= \sum_{j=1}^{i-1}\lfloor\frac{d+\Delta_j^s}{T_j}\rfloor c_j + c_i - d, \quad (4)$$

where $0 \leq \Delta_j^s < T_j$ is the remainder of $t_{s-1} - c_i - s_i$ divided by T_j. Let $\overline{\Delta_j^s} = T_j - \Delta_j^s$. Notice that $\frac{d+\Delta_j^s}{T_j}$ is not an integer, since d is the smallest solution of equation 2. We obtain that $\lfloor\frac{d+\Delta_j^s}{T_j}\rfloor = \lceil\frac{d-\overline{\Delta_j^s}}{T_j}\rceil$. The equation 4 is equal to

$$\sum_{j=1}^{i-1}\lceil\frac{d-\overline{\Delta_j^s}}{T_j}\rceil c_j + c_i - d \quad (5)$$

Since $\{\tau_i\}_{i=1}^s$ as a preemptive task system is rate monotonic schedulable, the smallest solution of equation 5 is less than or equal to T_i. Thus, the proof of the case 1 is complete.

Case 2: There is no request of lower priority processed before t in $[t_b, t_e]$.

Notice that in the proof of case 1, the argument is true for all $c_k > 0$. The proof of case 2 follows the same argument of that of case 1 with $c_k = 1$. Q.E.D.

6 Concluding Remarks

In this paper, the minimum delay objective was proposed in the discussion of the scheduling strategies for nonpreemptive soft real-time periodic task systems. In the soft real-time scheduling, high utilization bounded delay is preferred rather than the strict deadlines which usually causes low untilization of the processors. We have given an upper bound for arbitrary scheduling strategies of no inserted idle times and showed the formulas of the minimum delay of first-come-first-serve, rate monotonic and early-next-request first scheduling strategies.

References

[1] D.D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an intergrated services packet network: architecture and mechanism. *ACM Computer Comm. Review*, 22(4):14–26, October 1992.

[2] H.Schulzrinne. A transport protocol for audio and video conferences and other multiparticipant real-time applications. Technical report, 1992. Internet Draft.

[3] D. F. Stanat K. Jeffay and C. U. Martel. On non-preemptive scheduling of periodic and sporadic tasks. In *Proceedings of IEEE Symposium on Real-Time Systems*, pages 129–139, 1991.

[4] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in the hard real time environment. *Journal of ACM*, 20(1):46–61, 1973.

[5] P. Venkat Rangan and Harrick M. Vin. Designing an on-demand multimedia service. *IEEE Communications Magazine*, July 1992.

[6] W. D. Sincoskie. System architecture for a large video on demand service. *Computer Networks and ISDN Systems*, (22), 1991.