Results on the Slack of a Periodic Task Set

Maryline Chetto

CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

Rapport interne IRCCyN

RI 2008_5

Results on the Slack of a Periodic Task Set

Maryline CHETTO

Juin 2008

Institut de Recherche en Communications et en Cybernétique de Nantes

IRCCyN - 1, rue de la Noë - BP 92 101 - 44321 Nantes CEDEX 03 - Fax : 02 40 37 69 30

Results on the slack of a periodic task set

Maryline Chetto

IRCCyN / CNRS - University of Nantes

1 Rue de la Noë, F-44321 Nantes FRANCE

Abstract

Key words: real time systems, scheduling, periodic tasks, slack, earliest deadline

1 Introduction

In many real-time applications, the computer is required to execute preemptable periodic tasks (e.g sensory processing) with strict deadlines.

A periodic task set can be denoted as follows: $\mathcal{T} = \{T_i(C_i, R_i, P_i), i = 1 \; to \; n\}$. In this characterization, every task T_i makes its initial request at time 0 and its subsequent requests at times kP_i, $k = 1, 2, ...$ The execution time required for each request of T_i is C_i time units and a deadline for T_i occurs R_i units after each request by which task T_i must have completed its execution. We assume that $0 < C_i \leq R_i \leq P_i$ for each $1 \leq i \leq n$. A schedule Γ for $\mathcal{T}$ is said to be valid if the deadlines of all tasks of $\mathcal{T}$ are met in Γ. A task set is said to be feasible on one processor if there exists a valid schedule for $\mathcal{T}$ on one processor. A scheduling algorithm is said to be optimal if it produces a valid schedule for every task set which is feasible.

The problem of scheduling periodic tasks on one processor has been an active area of research (see, e.g., [1]). Liu and Layland have shown that ED is optimal [3]. ED schedules at each instant of time t, the ready task (i.e the task that may be processed and is not yet completed) whose deadline is closest to t. Deciding if a task set $\mathcal{T}$ is feasible requires to construct the ED schedule and to see if the deadlines of all the requests are met from 0 to $\mathcal{P}$ where $\mathcal{P}$ (called the hyperperiod) is the least common multiple of $P_1, P_2, \ldots, P_n$ since the processor does exactly the same thing at time t ($t \geq 0$) that it does at times $t + k\mathcal{P} \quad (k = 2, 3, \ldots)$ [4]. We define the slack of $\mathcal{T}$ at current time t, denoted by $\delta(t)$, as the maximum time for defering the execution of the periodic requests from t without comprimising the validity of the ED schedule. Computation at run time of the slack can be used to authorize or forbid the execution of a nonpreemptable task that requires to be run upon arrival from start to completion without interruption (e.g. interrupt handling or exception handling).

At this point, two fundamental questions arise:

(1) how to determine the slack at run time?

(2) are there lower bounds to the slack?

2 Background materials

Two versions of ED, namely EDS and EDL are proposed by the author [2]. Under EDS, the ready tasks are processed as soon as possible, whereas under EDL they are processed as late as possible. Let $\mathcal{S}$ be a sporadic task set defined as follows: $\mathcal{S} = \{S_i(r_i, C_i, d_i), i = 1 \; to \; m\}$. In this characterization, task S_i becomes ready at time r_i, requires C_i units of time and a deadline occurs at d_i.

Let $D = max\{d_i; S_i \in \mathcal{S}\}$. For any instants t_1 and t_2, let denote by $\Omega_S^X(t_1, t_2)$ the total processor idle time available in $[t_1, t_2]$ when $\mathcal{S}$ is scheduled according to algorithm X. We now recall fundamental properties of EDS and EDL.

Theorem 1 *For any instant t such that $t \leq D$,*

$$\Omega_S^{EDS}(0,t) \leq \Omega_S^X(0,t) \leq \Omega_S^{EDL}(0,t) \tag{1}$$

Proof: See [2]

Theorem 1 says that applying EDS (respectively EDL) to a task set $\mathcal{S}$ guarantees the minimum (respectively maximum) available idle time within any time interval $[0, t]$, $0 \leq t \leq D$. Given that at current time t, the set of periodic requests available from t up to the end of the current hyperperiod behaves like a sporadic task set, Theorem 1 gives us theoretical basis of an algorithm for computing the slack. This is done by mapping out the EDL schedule which can be represented by means of two arrays. The first, $\mathcal{K} = (k_0, k_1, ...k_p)$, represents the times at which idle times occur, necessarily after the deadline of a periodic request. The second, $\mathcal{D} = (\Delta_0, \Delta_1, ...\Delta_p)$ represents the lengths of these idle times. Details of computation are given in [5].

The complexity of the algorithm is O(K.n) where K is equal to $\lfloor R/p \rfloor$, where R and p are respectively the longest deadline and the shortest period of current ready tasks. So, it may vary from O(n) to O(N) where N is the number of distinct periodic requests that occur in the hyperperiod. As the overhead can be very large, we are interested in providing properties on the variation in slack over time so as to avoid unnecessary on-line computations of the exact value.

3 Slack of a periodic task set

First, let compute the length of the initial slack, $\delta(0)$, obtained by applying EDL to $\mathcal{T}$ at time 0. Consequently, $\delta(0) = \Delta_0$. Let $x_j = P_j - R_j$ for j=1 to n.

Proposition 1 $$\Delta_0 = \min_{0 \le i \le p} k_i - \sum_{j=1}^{n} \lceil \frac{k_i + x_j}{P_j} \rceil C_j \tag{2}$$

Proof: Consider the schedule produced by EDL for $\mathcal{T}$ from 0 to $\mathcal{P}$. Let k be the first instant between 0 and P such that there is no idle time within $[0 + \Delta_0, k]$ and all the tasks with a deadline greater than k are entirely processed within $[k, \mathcal{P}]$. It follows that Δ_0 is equal to the length of the time interval $[0, k]$ minus the total quantity of processor time assigned to the requests with a deadline less than or equal to k. All the requests of every task T_j whose ready time is greater than $k+x_j$ must then be rejected. It comes that $\Delta_0 = k - \sum_{j=1}^{n} \lceil \frac{k+x_j}{P_j} \rceil C_j$ Let t be any time instant in $\mathcal{K}$ and $\alpha(t) = t - \sum_{j=1}^{n} \lceil \frac{t+x_j}{P_j} \rceil C_j$. Let us prove that $\forall t \neq k, \alpha(t) \geq \Delta_0$.

Case 1: $t < k$. From definition of k, we know that there exist some requests with a deadline posterior to t which are processed within $[0, t]$. Let $Q(t)$ be the processor time reserved to these tasks. It comes that $\Delta_0 = \alpha(t) - Q(t)$ and consequently $\Delta_0 < \alpha(t)$.

Case 2: $t > k$. There may exist some idle time within $[0 + \Delta_0, t]$ and there may exist some requests with a deadline posterior to t which are processed within $[0 + \Delta_0, t]$. Then, $\Delta_0 = \alpha(t) - Q(t) - \varphi(t)$ where $\varphi(t)$ denotes the total idle time within $[0 + \Delta_0, t]$. Consequently, $\Delta_0 < \alpha(t)$. Finally, it comes that $\Delta_0 = min\{\alpha(t); t \in \mathcal{K}\}$ which corresponds to (2). □

We show first a lemma which is used later to a derive lower bound to the slack at any time.

Lemma 1 *For any released time e that concides with the end of an idle time interval, $\delta(e) \geq \Delta_0$*

Proof: At time e, all the available tasks released before e have been processed. Consider the set of requests available from time e to time $\mathcal{P}$ and form the associated set of sporadic tasks. Let $\mathcal{S}$ be this set and consider time e as a new time zero. From theorem 1, applying EDL to $\mathcal{S}$ from e will produce a schedule where the total idle time that follows e is maximized and corresponds to the slack, $\delta(e)$. Now, we show that Δ_0 provides a lower bound to the length of this idle time. For this purpose, let t be the first deadline after e such that t is followed by an idle time interval and the processor is fully utilized between $e + \delta(e)$ and t. Assume that time t coincides with the deadline of a request, for every task in $\mathcal{T}$. In particular, t coincides with the deadline of the request that is released at time e. Let T_l be this task. Then, $\exists k, k' \in \mathbf{N}$; $e = kP_l$ and $t = k'P_l - x_l$. This assumption takes care of the worst possible case in the sense that the processor is required to provide maximum service in the time interval $[kP_l, k'P_l - x_l]$. Let $\zeta = (k' - k)P_l - x_l$.It comes that $\delta(e) \geq \zeta - \sum_{j=1}^{n} \lceil \frac{\zeta + x_j}{P_j} \rceil C_i$. Since $\zeta \in \mathcal{K}$, proposition 1 enables us to conclude that $\delta(e) \geq \Delta_0$. □

Theorem 2 *For any time t, $\delta(t) \geq \Delta_0$*

Proof: Without loss of generality,, we assume that $t \in \{0, 1, 2, ..., \mathcal{P}\}$ since the schedule is periodic, and consequently $\delta(t) = \delta(t + k\mathcal{P}), k = 1, 2, \ldots$ The theorem is proved by induction on the units of time t. The basis of induction corresponds to $t = 0$. To carry out the induction step, we assume that the theorem is true at t i.e $\delta(t) \geq \Delta_0$ and prove that $\delta(t+1) \geq \delta_0$. Introduce $\Gamma(t)$

to be the schedule produced by EDS from 0 to t and by EDL from t to $\mathcal{P}$ on $\mathcal{T}$. $\delta(t)$ is then given by the length of the idle time that follows time t in $\Gamma(t)$. Now, consider the schedule $\Gamma(t+1)$. We examine three cases.

Case 1: The processor is processing a task T with a current deadline d in $[t, t+1]$ and there is no task released after t with a deadline anterior to d. As tasks are scheduled by EDS, T has the earliest deadline among the ready tasks at t. This implies that this task is the first one to be scheduled after t in $\Gamma(t)$, according to EDL. It follows that $\Gamma(t+1)$ is obtained from $\Gamma(t)$ by a permutation of the quantum of idle time between t and $t+1$ and the quantum of processor time for T between $t+\delta(t)$ and $t+\delta(t)+1$. Then, $\delta(t) = \delta(t+1)$ and consequently $\delta(t+1) \geq \Delta_0$.

Case 2: The processor is processing a task T with a current deadline d and there is at least one task released after t with a current deadline anterior to d.

Let e_i and d_i be the release time and the deadline of the first request after t that verifies $e_i > t$ and $d_i \leq d$.We have to examine two subcases:

Subcase 1: T is not scheduled before time d_i in $\Gamma(t+1)$. Consequently, T is not scheduled within $[e_i, d_i]$ in $\Gamma(e_i)$. As there is no ready task with a deadline less than d between $t+1$ and e_i , this means that there is no task scheduled by EDL within $[t+1, e_i]$ and so, $\delta(t+1) = \delta(e_i) + (e_i - (t+1))$. Since $\delta(e_i) \geq \Delta_0$ from Lemma 1, then $\delta(t+1) \geq \Delta_0$.

Subcase 2: T is partially scheduled before time d_i in $\Gamma(t+1)$. As T is scheduled after T_i in $[e_i, d_i]$, $\Gamma(t+1)$ is obtained from $\Gamma(t)$ by a permutation of the quantum of idle time between t and $t+1$ and the quantum of processor time for T_i between $t+\delta(t)$ and $t+\delta(t)+1$.

Case 3: The processor is not occupied in $[t, t+1]$. The processor will remain inactive until the next release time. Let e_i be this time instant. It comes

that $\delta(t+1) = e_i - t + \delta(e_i)$. Since $\delta(e_i) \geq \Delta_0$ from Lemma 1, then $\delta(t+1) \geq \Delta_0$. □

From demonstration of theorem 2, we conclude that $\delta(t)$ has local maximum at the beginning of every idle time interval, is linear decreasing within any idle time interval,and non increasing in any busy time interval. Besides, $\delta(t)$ is never less than Δ_0.

4 Slack with additional tasks

Consider once again the periodic task set $\mathcal{T}$ defined previously. And assume that sporadic tasks have been accepted for execution between 0 and t without comprimising the validity of the resulting ED schedule. We have now established the two following theorems:

Theorem 3 *For any time t such that $\delta(t) \geq \Delta_0$ and for any length q such that no additional task has been accepted within $[t, t+q]$, then $\delta(t+q) \geq \Delta_0$.*

Proof: We prove that the existence of a time instant t such that $\delta(t) = \Delta_0$ implies $\delta(t+1) \geq \Delta_0$. Let $\Gamma(t+1)$ be defined as in the proof of theorem 2 and examine the two possible situations:

Case 1: the processor is idle within $[t, t+1]$ in $\Gamma(t+1)$. So, there is no ready task to be processed at time t. Time interval $[t, t+1]$ is then included in an idle time interval and schedule $\Gamma(t+1)$ from $t+1$ does not depend on the execution of sporadic tasks within $[0, t+1]$. From theorem 2, $\delta(t) \geq \Delta_0$.

Case 2: the processor is active within $[t, t+1]$ in $\Gamma(t+1)$. Let d be the deadline of the sporadic task or the request of the periodic task which is processed

between t and $t+1$. Then, two situations are possible:

Case 2-a: there is at least one idle time in $[t+1+\delta(t+1), d]$. The end of this idle time coincides with a deadline which necessarily belongs to a request of a periodic task, ready to be executed at or after $t+1$. The existence of an idle time after d_j means that the schedule $\Gamma(t+1)$ restricted to $[t+1, d_j]$ does not depend on the execution of sporadic tasks within $[0, t+1]$. Therefore, according to theorem 2, $\delta(t+1) \geq \Delta_0$

Case 2-b: there is no idle time in $[t+1+\sigma(t+1), d]$ Consequently, there is no idle time in $[t+\delta(t+1), d]$. The quantity of processor idle time between t and d respectively in $\Gamma(t)$ and $\Gamma(t+1)$ are identical. As a result, we have $\delta(t+1) = \delta(t) - ((t+1)-t)+1$, i.e $\delta(t+1) = \delta(t)$. Since $\delta(t) = \Delta_0$ by hypothesis, it follows that $\delta(t+1) = \Delta_0$. □

Theorem 3 says that if periodic tasks are jointly scheduled with sporadic tasks and if at a given time instant, the slack is greater than or equal to Δ_0 (notably whenever the processor is idle), the slack will never decrease below Δ_0 as long as no additional task is accepted. Consequently, without any test, we may accept in the future any preemptable or nonpreemptable sporadic task whose execution time is less than or equal to Δ_0.

Theorem 4 *For any time t such that $\delta(t) < \Delta_0$ and for any length q such that no additional task has been accepted within $[t, t+q]$, then $\delta(t+q) \geq \delta(t)$.*

Proof: Let us prove that, if at time t, $\delta(t) < \Delta_0$ then $\delta(t+1) \geq \delta(\tau)$. This proof is obvious when one considers the different cases of Proposition 8:
In cases 1 and 2.1, we have $\delta(t+1) \geq \Delta_0$. Since $\delta(t) < \Delta_0$ by hypothesis, then $\delta(t+1) \geq \delta(t)$.
In case 2.2, we have $\delta(t+1) = \delta(t)$. Consequently, $\delta(t+1) \geq \delta(t)$. □

Theorem 4 states the non decreasing variation of the slack from any time where its value is less than Δ_0.

5 Summary

A first result was to show that a lower bound to the slack of a periodic task set is the slack at time zero, obtained by an off-line computation in O(N). Properties on the slack stated in theorems 3 and 4 are used to improve efficiency and predictivity of the scheduler in the on-line acceptance of sporadic tasks. It will be interesting to determine similar properties under resource constraints.

References

[1] G.C. Buttazzo, Hard real-time computing systems, Springer (2005)

[2] H. Chetto and M.Chetto, Some Results of the Earliest Deadline Scheduling Algorithm, IEEE Transactions on Software Engineering, 15 (10), pp 1261-1269 (1989).

[3] C.L. Liu and J.W. Layland, Scheduling algorithms for multiprogramming in a hard-real-time environment, J.ACM 20 (1) (1973).

[4] J.Y.K. Leung and M.L. Merril, A note on preemptive scheduling of periodic real-time tasks, Information Processing Letters, 20 (30), pp 115-118 (1980).

[5] M. Silly-Chetto, The EDL server for scheduling periodic and soft aperiodic tasks with resource constraints, Journal of Real-Time Systems, 17 (1), pp 1-25 (1999).