RLP : Enhanced QoS Support for Real-Time Applications

Audrey Marchand and Maryline Silly-Chetto, *member IEEE*
LINA (Laboratoire d'Informatique de Nantes Atlantique)
Rue Christian Pauc
44306 Nantes cedex 03 France
Email: {audrey.marchand, maryline.chetto}@univ-nantes.fr

Abstract

In this paper, we study the problem of scheduling periodic task sets defined under Quality of Service (QoS) constraints. In our approach, periodic tasks allow occasional skips of instances. A new algorithm, called RLP (Red tasks as Late as Possible) based on the Skip-Over model and the EDL (Earliest Deadline as Late as Possible) scheduling strategy, is proposed to enhance the QoS observed for periodic tasks, i.e, the ratio of periodic tasks which complete before their deadline. We prove that our results are never worse than those obtained in previous work. Experimental results also show significant improvement achieved by our algorithm over RTO and BWP.

1. Introduction

Real-time systems are those in which the time at which the results are produced is important. The correctness of the result of a task is not only related to its logic correctness, but also to when the results occur. The stringency of timing constraints distinguishes real-time applications from non-real-time applications. Traditional classification of real-time systems stands for three classes to characterize the real-time requirement of such systems : hard, soft and firm. In hard real-time systems, all instances must be guaranteed to complete within their deadlines. In those critical control applications, missing a deadline may cause catastrophic consequences on the controlled system. For soft systems, it is acceptable to miss some of the deadlines occasionally. It is still valuable for the system to finish the task, even if it is late. In firm systems, tasks are also allowed to miss some of their deadlines, but, there is no associated value if they finish after the deadline. Typical illustrating examples of systems with firm real-time requirements are systems involving repeated signals with redundancy, like telecommunication systems.

Much work has been conducted in scheduling analysis in hard real-time systems, where violating task deadlines must be avoided at all cost. However, in recent years, many new real-time applications have emerged in which it is not necessary to meet all the task deadlines as long as the deadline violations are adequately spaced. Not having to meet every deadline allows the capacity of the system's resources to be smaller that it would be for meeting all of them.

There have been some approaches to the specification and design of real-time systems that tolerate occasional losses of deadlines. Hamdaoui and Ramanathan in [5] introduced the concept of *(m,k)-firm* deadlines to model tasks that have to meet m deadlines every k consecutive invocations. It relies on best-effort algorithms so it does not provide any guarantee on the number of deadlines a task can miss. The Skip-Over model was introduced by Koren and Shasha [6]. It is a particular case of the *(m,k)-firm* model. They reduce the overload by skipping some task invocations, thus exploiting skips to increase the feasible periodic load.

In [2, 3], Caccamo and Buttazzo schedule hybrid task sets consisting of skippable periodic and soft aperiodic tasks. They propose and analyze an algorithm, based on a variant of Earliest Deadline First (EDF) scheduling, in order to exploit skips under the Total Bandwith Server (TBS). They also derived schedulability bounds to perform off-line feasibility tests.

In this paper, we address the problem of the dynamic scheduling of periodic task sets with skip contraints. The scope of the paper is to maximize the QoS of periodic tasks by maximizing the number of instances which complete before their deadline. The remainder of this paper is organized in the following manner. Section 2 presents relevant background material about both the Skip-Over model and the EDL scheduling algorithm. More particularly, we give the definition of RTO (Red Tasks Only) and BWP (Blue When Possible) scheduling algorithms, which are based on the Skip-Over model. The functioning and optimality of the EDL algorithm is also outlined. Further, we describe the proposed algorithm, namely RLP (Red tasks as Late as

Possible), as an enhancement of the BWP algorithm, based on the EDL scheduling mechanism. The performance analysis of RLP, in terms of task completions, is reported in section 4. Finally, in section 5, we summarize our contribution.

2. Theoretical background

2.1 The Skip-Over model

We are here interested in the problem of scheduling periodic tasks which allow occasional deadline violations (*i.e.*, skippable periodic tasks), on a uniprocessor system. We assume that tasks can be preempted and that they do not have precedence constraints. A task T_i is characterized by a worst-case computation time c_i, a period p_i, a relative deadline equal to its period, and a skip parameter s_i, which gives the tolerance of this task to missing deadlines. The distance between two consecutive skips must be at least s_i periods. When s_i equals to infinity, no skips are allowed and T_i is equivalent to a hard periodic task. One can view the skip parameter as a QoS metric (the higher s_i, the better the quality of service).

A task T_i is divided into instances where each instance occurs during a single period of the task. Every instance of a task can be red or blue. These are the colors used by Koren and Shasha in [6]. A red task instance must complete before its deadline; a blue task instance can be aborted at any time. However, if a blue instance completes successfully, the next task instance is still blue.

Red Tasks Only (RTO) algorithm. The first algorithm proposed by Koren and Shasha is the Red Tasks Only (RTO) algorithm. Red instances are scheduled according to EDF, while blue ones are always rejected. In the deeply red model where all tasks are synchronously activated and the first $s_i - 1$ instances of every task T_i are red, this algorithm is optimal. As illustrated in Figure 1, we can see that the distance between every two skips is exactly s_i periods.

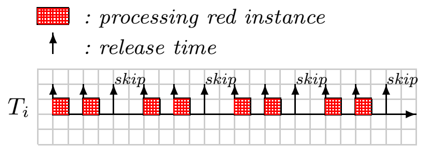

FIG. 1 – *RTO scheduling algorithm* $(s_i = 3)$

Blue When Possible (BWP) algorithm. The second algorithm studied is the Blue When Possible (BWP) algorithm which is an improvement of the first one. Indeed, BWP schedules blue instances whenever their execution does not prevent the red ones from completing within their deadlines. In that sense, it operates in a more flexible way. Figure 2 shows an example of the possible sequence of instances of a BWP task.

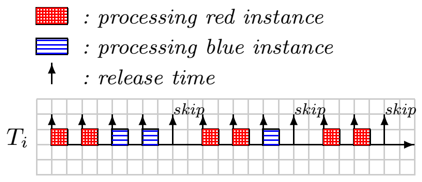

FIG. 2 – *BWP scheduling algorithm* $(s_i = 3)$

2.2 The EDL algorithm

The definition of the EDL algorithm makes use of some results presented by Chetto and Chetto in [4]. In this paper, two different implementations of EDF (Earliest Deadline First), namely *EDS (Earliest Deadline as Soon as possible)* and *EDL (Earliest Deadline as Late as possible)*, are studied. Under EDS, all ready tasks are ordered by increasing deadlines and are executed as soon as possible. Under EDL, periodic tasks are scheduled as late as possible, which needs an accurate characterization of the idle times during which the processor is not occupied. Determination of the latest start time for every request of the periodic tasks requires preliminary construction of the schedule by the EDL algorithm. Let us introduce the terminology used by the authors: with f_Y^X they denote the *availability function* defined with respect to a task set Y and a scheduling algorithm X.

$$\begin{aligned} f_Y^X(t) &= 1 \quad \textit{if the processor is idle at } t \\ &= 0 \quad \textit{otherwise} \end{aligned} \tag{1}$$

So, for any instants t_1 and t_2, value of $\int_{t_1}^{t_2} f_Y^X(t)dt$ denoted by $\Omega_Y^X(t_1,t_2)$ gives the total idle time in $[t_1,t_2]$.

The complexity of the EDL algorithm is $O(Kn)$ where n is the number of periodic tasks, and K is equal to $\lfloor \frac{R}{p} \rfloor$, where R is the longest deadline, and p is the shortest period [7]. We also recall the fundamental property relative to the optimality of EDL [4]:

Theorem 1 *Let X be any preemptive scheduling algorithm and $\mathcal{A}$ a set of independent aperiodic tasks. For any instant t,*

$$\Omega_{\mathcal{A}}^X(0,t) \leq \Omega_{\mathcal{A}}^{EDL}(0,t) \tag{2}$$

We give now an illustrative example of the computation of the idle times performed by EDL. Consider the periodic task set $\mathcal{T} = \{T_1,T_2\}$ consisting of two periodic tasks T_1 and T_2, with periods 10 and 6, and computation times 3 and 3 respectively. The $f_{\mathcal{T}}^{EDL}$ computation produced at time zero is described in Figure 3.

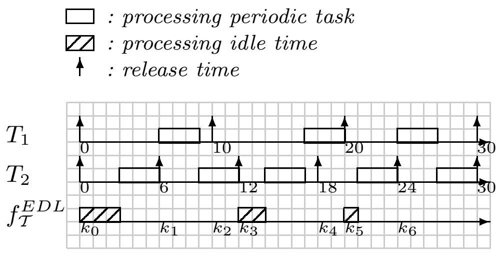

FIG. 3 – $f_{\mathcal{T}}^{EDL}$ *computation produced at time zero*

The authors in [4] described how the EDL algorithm can be applied, first to the decision problem that arises when a sporadic time critical task occurs and requires to be run at an unpredictable time and second, to the scheduling problem that arises in a fault tolerant system using the Deadline Mechanism for which each task implements primary and alternate algorithms. In next section, we are interested in using EDL only to simulate a schedule and to derive a measure required for deciding whether a blue task can be accepted. For this method to work, we propose an acceptance condition that can be easily tested in line whenever a blue task occurs and enables us to guarantee a feasible execution for it, each time it is possible.

3. Red tasks as Late as Possible (RLP) algorithm

3.1 Algorithm outline

Red tasks as Late as Possible (RLP) algorithm is designed to maximize the QoS (*i.e.*, the number of task completions) of periodic task sets defined under skip constraints. It is a dynamic scheduling algorithm which acts as follows: red tasks enter straight the system at their arrival time whereas blue tasks integrate the system upon acceptance. Once they have been accepted, blue tasks are scheduled as soon as possible together with red tasks. Note that, upon acceptance, blue tasks are of same importance beside red tasks (contrary to BWP which always assigns higher priority to red tasks).

Whenever a new blue task enters the system, the idle times of an EDL scheduler are computed using the current periodic tasks with skips, in order to determine the latest start time for every instances of the red task set. Note that this preliminary construction of the schedule is performed on the basis of red tasks only. Formulae for the computation of the idle time intervals are then adapted on the basis of a RTO sequence of instances, denoted EDL-RTO algorithm [1].

In fact, the policing mechanism of RLP allocates to the blue tasks only the idle times of an EDL schedule, without compromising the timeliness of the red tasks. A schedulability test checks whether there are enough idle times to accommodate the new blue task within its deadline, as described in the following section.

3.2 Acceptance test of blue tasks under RLP

Now, we are ready to present the new feasibility test algorithm for the RLP scheduling scheme which, given any occurring blue task B is capable of answering the question "Can B be accepted?". Notice that B will be accepted if and only if there exists a valid schedule, *i.e.*, a schedule in which B will execute by its deadline while red periodic tasks and blue tasks previously accepted, will still meet their deadlines. Let τ be the current time which coincides with the arrival of a blue task B. Upon arrival, task $B(r,c,d)$ is characterized by its release time r, its execution time c, and its deadline d, with $r+c \leq d$. We assume that the system supports several blue tasks at time τ. Each of them has been accepted before τ and has not completed its execution at time τ. Let denote by $\mathcal{B}(\tau) = \{B_i(c_i(\tau),d_i), i{=}1 \text{ to } blue(\tau)\}$ the blue task set supported by the machine at τ. Value $c_i(\tau)$ is called dynamic execution time and represents the remaining execution time of B_i at τ. A deadline occurs at d_i. We assume that $\mathcal{B}(\tau)$ is ordered such that $i < j$ implies $d_i \leq d_j$.

The acceptance test of blue tasks within a system involving RLP skippable tasks presented below in Theorem 2, is based on the one established by Silly and al. [8] for the acceptance of sporadic requests occurring in a system consisting of basic periodic tasks (*i.e.*, without skips).

Theorem 2 *Task B is accepted if and only if, for every task $B_i \in \mathcal{B}(\tau) \cup \{B\}$ such that $d_i \geq d$, we have $\delta_i(\tau) \geq 0$, with $\delta_i(\tau)$ defined as:*

$$\delta_i(\tau) = \Omega_{\mathcal{T}(\tau)}^{EDL-RTO}(\tau,d_i) - \sum_{j=1}^{i} c_j(\tau) \qquad (3)$$

$\delta_i(\tau)$ is called slack of task B_i at time τ, and represents the surplus processing power within the scheduling interval of B_i such that all red tasks and blue tasks with deadline less than or equal to d_i are feasibly scheduled. $\Omega_{\mathcal{T}(\tau)}^{EDL-RTO}(\tau,d_i)$ denotes the total units of time that the processor is idle in the time interval $[\tau, d_i]$. The total computation time required by blue tasks within $[\tau, d_i]$ is given by $\sum_{j=1}^{i} c_j(\tau)$

The procedure that implements the acceptance test calls for the EDL-RTO algorithm for the computation of the total idle times which will be used to compute the slack of blue tasks. Then, this slack is compared to zero. Thus, the acceptance test proposed in this paper runs in $O(\lfloor \frac{R}{p} \rfloor n + blue(\tau))$ in the worst-case, where n is the number of periodic tasks, R is the longest deadline, p is the shortest period, and $blue(\tau)$ denotes the number of active blue tasks at time τ, whose deadline is greater or equal to the deadline of the occurring task. Note that this acceptance test could be implemented in $O(n + blue(\tau))$ by considering and maintaining to update additional data structures using slack tables, as proved in [9]. The pseudo-code for the acceptance test performed by RLP is given below:

```
RLP feasibility algorithm;
Begin
Set $\mathcal{B}(\tau)$ = active blue task set at time $\tau$
While (a blue task $R$ occurs at time $\tau$) Do
      Set FEASIBLE $= 1$
      Select blue task $B_j$ in $\mathcal{B}(\tau) \cup \{B\}$
      with longest deadline
      Compute idle times between $\tau$ and $d_j$
      For all blue tasks $B_i$ in $\mathcal{B}(\tau) \cup \{B\}$
      with $d_i \geq d$
              Compute slack $\delta_i(\tau)$
              If ($\delta_i(\tau) < 0$)
                      Set FEASIBLE $= 0$ and stop
              EndIf
      EndFor
      return FEASIBLE
EndWhile
End
```

3.3 RLP vs. BWP

Consider the periodic task set $\mathcal{T} = \{T_1, T_2\}$ consisting of two periodic tasks T_1 and T_2, with periods 10 and 6, computation times 6 and 3, and skip parameters 2 and 2 respectively. Figure 4 and Figure 5 show the schedule produced by BWP and RLP respectively. It is easy to see that RLP improves on BWP. Observe that when BWP is used, two violations of deadline relative to blue task instances occur for T_1 and T_2 at time $t = 20$ and $t = 24$ respectively, thus reducing the QoS of the task set. In contrast, fewer deadline violations occur with the RLP scheduler. Red tasks and accepted blue tasks are jointly scheduled as soon as possible according to the EDF algorithm.

The acceptance test compensates for the time wasted in starting the execution of blue tasks which are not able to complete within their deadline, as we can observe in the BWP case for the first blue instance of T_1 released at time $t = 10$ and aborted at time $t = 20$ (5 units of time are indeed wasted).

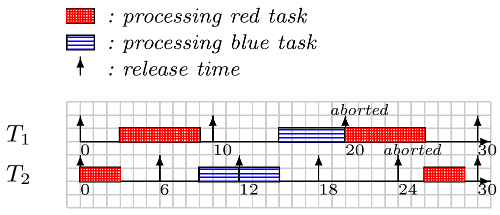

FIG. 4 – *BWP scheduling for task set $\mathcal{T}$*

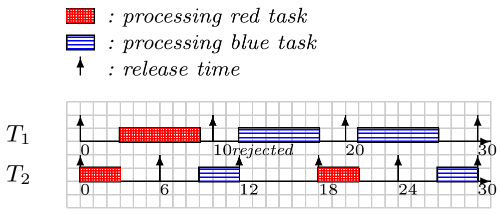

FIG. 5 – *RLP scheduling for task set $\mathcal{T}$*

The acceptance test processed by RLP at $t = 12$ when the blue instance of T_2 is released, is based on the computed sequence by EDL-RTO represented on Figure 6, where the execution of all red task instances is postponed.

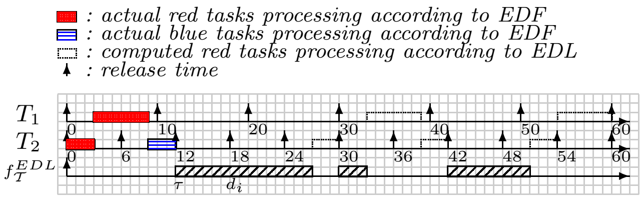

FIG. 6 – *Idle times computation for RLP at $t = 12$*

In the computed EDL-RTO sequence, we can note that T_1 task instance released at time $t = 20$ is blue (T_1 task instance released at time $t = 10$ was blue and has been accepted, so next instance of T_1 is still blue, according to the Skip-over definition). We apply the same reasoning to the occurring task T_2 which is supposed to be accepted at time $t = 12$. After, the EDL schedule is constructed on the basis of the RTO succession of instances where blue tasks are always rejected.

Let us compute the slack of T_1 and T_2 blue instances at time $\tau = 12$. We observe that the total units of time that the processor is idle in time intervals [12,18] and [12,20] is given by $\Omega^{EDL-RTO}_{\mathcal{T}(12)}(12{,}18) = 6$ and $\Omega^{EDL-RTO}_{\mathcal{T}(12)}(12{,}20) = 8$ respectively. The total computation time required by blue tasks within [12,18] is equal to $\sum_{j=1}^{i} c_j(12) = 3$ (T_2 blue instances require 3 units of time for completing its execution). Within [12,20], it is equal to $\sum_{j=1}^{i} c_j(12) = 9$ (T_1 and T_2 blue instances require respectively 3 and 6 units of time for completing their execution).

Consequently, the slack of T_2 blue instance at time $t = 12$ is $\delta_i(12) = \Omega^{EDL-RTO}_{\mathcal{T}(12)}(12{,}18) - \sum_{j=1}^{i} c_j(12) = 6-3 = 3$, which is correct. The slack of T_1 blue instance at time $t = 12$ is given by $\delta_i(12) = \Omega^{EDL-RTO}_{\mathcal{T}(12)}(12{,}20) - \sum_{j=1}^{i} c_j(12) = 8 - 9 = -1$. This negative value, which means that T_1 blue instance will miss one unit of time for completing within its deadline, implies that the occurring blue instance of T_2 must be rejected, if we want to avoid the deadline violation of T_1 blue instance which is observed at time $t = 20$ in the BWP case (see Figure 4). Note that this rejection, performed with RLP, contributes to save time used later for the successful completion of other blue instances.

In this illustrative example, RLP outperforms BWP in terms of the QoS observed for periodic tasks. In next section we quantify more precisely the gain of performance of RLP upon BWP.

4. Performance evaluation

In this section, we summarize the results of simulation studies which compare the performance of RLP to that of BWP. The objective is to maximize the QoS level of periodic tasks, i.e., the ratio of periodic tasks which complete before their deadline.

4.1 Simulation context

The simulation context includes a periodic task set with a least common multiple equal to 3360. Tasks are defined under QoS contraints with uniform s_i. Their worst-case execution time depends on the setting of the periodic load without skips $U_p = \sum_{i=1}^{n} \frac{c_i}{p_i}$. Deadlines are equal to the periods and greater than or equal to the computation times. Simulations have been processed over 10 hyperperiods. A fair comparaison needs also to study RLP and BWP according to different skip values.

4.2 Low skip parameters

Small skip parameters attached to periodic tasks means that many instances can be skipped. We considered here a uniform skip value $s_i = 2$ (one instance every two can be aborted). Measurements rely on the ratio of periodic tasks which complete before their deadline. The evaluation is done for a periodic load that varies from 90% to 160%. The results obtained for $s_i = 2$ are described on Figure 7 and Figure 8 for two periodic task sets, consisting of 5 and 15 tasks respectively.

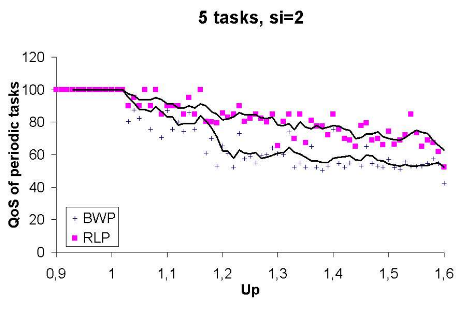

FIG. 7 – *QoS of tasks with low skips (5 tasks)*

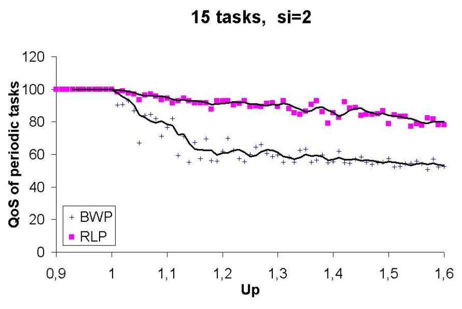

FIG. 8 – *QoS of tasks with low skips (15 tasks)*

From the graphs, we can say that the performance of BWP is dramatically worse than the one achieved by RLP. This result was expected because BWP attempts to schedule blue instances that have not enough time for completing within their deadlines. This wasted time is not saved for executing other blue instances with closer deadline. In contrast, RLP finds a way of saving this CPU time by implementing an acceptance test for blue instances. We can observe that this gain of performance is all the more significant as the periodic load U_p is higher. For instance, in Figure 8, for $U_p \geq 115\%$, RLP enjoys more than factor $\frac{1}{3}$ success rate advantage over BWP. Note that the advantage of RLP over BWP is wider when the number of tasks in the system is large.

4.3 High skip parameters

We constructed another set of experiments to evaluate the impact of the skip value for each algorithm, namely BWP and RLP. Simulation results reported on Figure 9 and Figure 10 are carried out for a uniform skip parameter $s_i = 6$, and for two periodic task sets, consisting of 5 and 15 tasks respectively. The periodic load was increased from 90% to 104%.

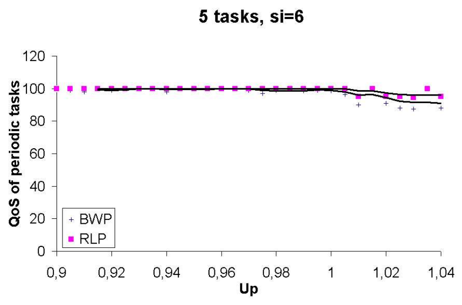

Fig. 9 – *QoS of tasks with high skips (5 tasks)*

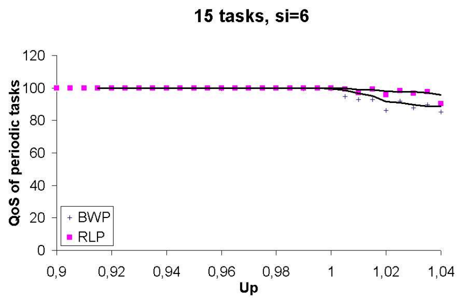

Fig. 10 – *QoS of tasks with high skips (15 tasks)*

In view of these results, we can say that, on the whole, RLP achieves better performance that BWP. We note that the higher the periodic load, the wider the performance advantage of RLP over BWP. The variation of the skip parameter value shows that, for wide loads, the QoS of periodic tasks is all the more improved with RLP that the QoS constraint is smaller (*i.e.*, skips of instances in quick succession). For instance, for $U_p = 104\%$, RLP applied to periodic tasks with $s_i = 2$ (see Figure 8) will successfully process more than twice as many periodic instances over BWP, as with periodic tasks with $s_i = 6$ (see Figure 10). As we can see, the major difference in the performance between RLP and BWP appears not only for heavy loads but also for small value of s_i. Finally, we note again, but to a lesser extent, that the advantage of RLP over BWP is all the more significant as there are lots of tasks involved in the system.

5. Conclusions

In many real-time embedded systems, *e.g.*, a video decoder, it is acceptable to miss task deadlines occasionally. In this paper, we addressed the problem of scheduling task sets with QoS constraints using the Skip-over model and the EDL server. Our main contribution was actually to provide a scheduling algorithm, namely RLP, which enhances the QoS of periodic tasks that allow skips. Similar to the BWP algorithm, the proposed approach manages red and blue tasks, with the difference that it implements an acceptance test of blue instances based on the EDL server's principle. Simulation results show that the improvements with this new scheduling algorithm are quite significant. Our future work includes extending the approach to hybrid task sets consisting of periodic tasks with skips together with soft and hard aperiodic tasks.

References

[1] A. Marchand and M. Silly-Chetto: *Dynamic Scheduling of Soft Aperiodic Tasks and Periodic Tasks with Skips*, In Proceedings of the 25th IEEE Real-Time Systems Symposium Work-In-Progress Session, Dec 2004

[2] G. C. Buttazzo, M. Caccamo: *Minimizing Aperiodic Response Times in a Firm Real-Time Environment*, IEEE Trans. Software Eng., Vol. 25, No. 1, pp 22-32, 1999

[3] M. Caccamo and G. Buttazzo: *Exploiting Skips in Periodic Tasks for Enhancing Aperiodic Responsiveness*, In Proceedings of the 18th IEEE Real-Time Systems Symposium (RTSS'97), Dec 1997

[4] H. Chetto and M. Chetto: *Some Results of the Earliest Deadline Scheduling Algorithm*, In Proceedings of the IEEE Transactions on Software Engineering, Vol. 15, No. 10, pp 1261-1269, Oct 1989

[5] M. Hamdaoui and P. Ramanathan: *A Dynamic Priority Assignment Technique for Streams with (m,k)-firm deadlines*, IEEE Transcations on Computers, Vol. 44, No. 4, pp 1443-1451, Dec 1995

[6] G. Koren and D. Shasha: *Skip-Over Algorithms and Complexity for Overloaded Systems that Allow Skips*, In Proceedings of the 16th IEEE Real-Time Systems Symposium (RTSS'95), Pisa, Italy, 1995

[7] M. Silly: *The EDL Server for Scheduling Periodic and Soft Aperiodic Tasks with Resource Constraints*, The Journal of Real-Time Systems, Kluwer Academic Publishers, Vol. 17, pp 1-25, 1999

[8] M. Silly, H. Chetto and N. Elyounsi: *An Optimal Algorithm for Guranteeing Sporadic Tasks in Hard Real-Time Systems*, In Proceedings of the IEEE Symposium on Parallel and Distributed Processing, pp 578-585, 1990

[9] T. Tia, J. Liu, J. Sun, and R. Ha: *A Linear-Time Optimal Acceptance Test for Scheduling of Hard Real-Time Tasks*, Technical report, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana-Champaign, IL, 1994