Dynamic Scheduling of Soft Aperiodic Tasks and Periodic Tasks with Skips

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

Abstract

Soft real-time embedded systems have gained momentum with the emergence of new real-time applications dealing with multimedia and active monitoring. The traditional task model with hard timing constraints is not really suitable for these applications because of its lack of flexibility. For instance, missing a deadline once in a while decreases the Quality of Service (QoS) but will not cause any damage to multimedia communication systems. This paper deals with dynamic scheduling algorithms that handle both soft aperiodic tasks and periodic tasks with deadlines which can occasionally skip one instance. The ability to provide QoS is accomplished through the use of predefined skip parameters, which indicate the consecutive periods in which a task cannot miss a deadline. In this paper, we study services aperiodic tasks together with periodic tasks with skips by using variants of the EDL (Earliest Deadline as Late as possible) server, namely EDL-RTO (EDL-Red Tasks Only) and EDL-BWP (EDL- Blue When Possible). The objective is to minimize the average response time of soft aperiodic requests, while ensuring that the QoS of periodic tasks will never be less than a bound (i.e the skip factor) fixed by the application programmer. We performed some simulations to compare the performance of the EDL server to the BG (Back-Ground) server to determine the effect on average aperiodic response time and to compare the QoS of periodic tasks under EDL-RTO and EDL-BWP.

1. Introduction

Timeliness is the single most important aspect of a real-time system. Real-time systems are computer systems that monitor, respond to, or control an external environment. The computer system must meet various timing and other constraints that are imposed on it by the real-time behavior of the external world to which it is interfaced. The design of a real-time system must specify the timing requirements of the system and ensure that the system performance is both correct and timely.

There are three types of time constraints which can be attached to a task: *hard* (a late response implies a system failure), *soft* (the task has no deadline) and *firm* (the task may sometimes lose deadlines). 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. Applications and examples of such emerging systems are manufacturing systems with robots, telephone, radio and satellite communications, multimedia systems that provide text, graphic, audio, and video interfaces.

Hamdaoui and Ramanathan in [4] 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 [5]. 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 [1, 2], 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 constraints together with soft aperiodic tasks. The scope of the paper is to exploit the skips in order to minimize the average response time of soft aperiodic requests, using the EDL (Earliest Deadline as Late as possible) server, which is a dynamic slack stealing algorithm [3]. The remainder of this paper is organized in the following manner.

Section 2 presents relevant background material about dynamic scheduling of periodic tasks with skips and dynamic servicing of soft aperiodic requests. Further, we describe two dynamic scheduling algorithms, namely EDL-RTO and EDL-BWP, able to schedule soft aperiodic tasks together with periodic tasks with skips. The performance analysis of EDL-RTO and EDL-BWP, in terms of response time and ratio of task completions, is reported in section 4. Finally, in section 5, we summarize our contribution.

2. Theoretical background

2.1 Scheduling periodic tasks with skips

We look at the problem of uniprocessor scheduling of occasionally skippable periodic tasks. 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, $2 \leq s_i \leq \infty$, which gives the tolerance of this task to missing deadlines. The distance between two consecutive skips must be at least s_i periods. When $s_i = \infty$, 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 QoS).

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 [5]. A red task instance must complete before its deadline; a blue task instance can be aborted at any time. When a task misses its deadline, we say that the task (or deadline) instance was *skipped*. However, if a blue instance completes successfully, the next task instance is still blue.

The first algorithm proposed in [5] is the *Red Tasks Only (RTO)* algorithm. The red instances are scheduled as soon as possible according to EDF (Earliest Deadline First), while the 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. The second algorithm is the *Blue When Possible (BWP)* algorithm which is an improvement of the first one. 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.

2.2 Servicing soft aperiodic tasks with basic periodic tasks

We consider basic periodic tasks $T_i(c_i,p_i)$ without skips. The EDL *(Earliest Deadline as Late as possible)* server [3] processes the periodic tasks as soon as possible when no aperiodic activity is present. Whenever an aperiodic request occurs, it executes all periodic tasks as late as possible, while ensuring that all deadlines are met. In other words, the EDL algorithm takes advantage of the effective laxity (*i.e.*, the interval between the completion time and the deadline) of the periodic tasks, to minimize the aperiodic response time.

The fundamental property of EDL [3] is that it guarantees the maximum idle time in a given interval for any set of periodic tasks. In others words, the EDL server was proved to be optimal. In [3], Chetto and Chetto presented a simple way for on-line determining the localization and duration of idle times for the schedule produced by EDL. Note that the EDL sequence is constructed on-line only at instants corresponding to the arrival of a new aperiodic task. Moreover, as proved in [6], the on-line computation is performed in $O(\lfloor \frac{R}{p} \rfloor n)$ where n denotes the number of periodic tasks, R the longest deadline and p the shortest period.

3 Scheduling problem of interest

In this paper, we propose to merge the two approaches described in section 2.1 and 2.2 respectively, in order to deal with applications having both firm and soft real-time constraints. Let us consider periodic tasks with skips $T_i(c_i,p_i,s_i)$ where c_i represents the worst-case computation time, p_i the period, and s_i the skip factor of the task. The problem is to service soft aperiodic tasks together with periodic tasks with skips, by using the EDL server. We will show the EDL functioning with Skip-Over algorithms, namely RTO and BWP algorithms. The idea is to make use of the spare time saved by skipped instances in order to enhance the response time of aperiodic requests.

Let us assume that at time τ, an aperiodic task occurs. Determination of the idle times at time τ is performed on the basis of the *Red Tasks Only (RTO)* model (see Fig.1).

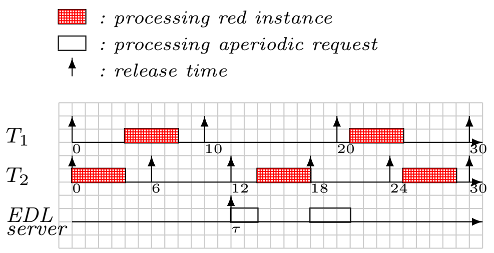
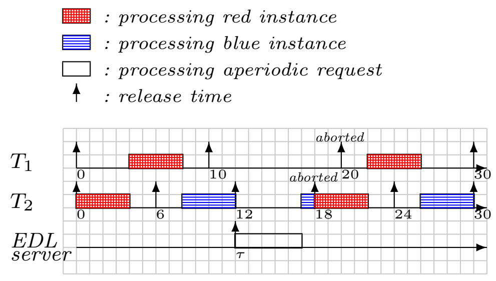


FIG. 1 – *The EDL server with the RTO task model*

Let P be the current hyperperiod equal to the least common multiple of the periods of the tasks. The determination of the idle times duration is performed in $O(\lfloor \frac{R}{p} \rfloor n)$ where n denotes the number of periodic tasks, R the longest deadline and p the shortest period.

If we want to adapt the EDL server to the BWP model, idle times determination must be performed at time τ considering that some task instances are blue. This is due to the fact that, when a blue instance completes successfully, the next task instance is still blue, thus introducing a shift in the RTO sequence used for the idle times determination at time τ (see Fig. 2). The EDL schedule must be constructed up to the end of the current hyperperiod P. The idle times determination is then in $O(N)$ where N denotes the total number of periodic requests released in this interval. Note that the on-line computation has always to be performed on the whole current hyperperiod, because of the unpredictable shifts in the RTO sequence, caused by blue instances completions.



: processing red instance
: processing blue instance
: processing aperiodic request
: release time
T_1
T_2
EDL server
aborted
aborted
0
10
20
30
0
6
12
18
24
30
τ

FIG. 2 – *The EDL server with the BWP task model*

Let us note that for the example, the aperiodic response time is respectively 9 and 5 with EDL-RTO and EDL-BWP.

4. Performance evaluation

Simulation results are presented to show variation of aperiodic response time and QoS of periodic tasks, with variable periodic and aperiodic offered load. The objective is to minimize aperiodic responsiveness while maintaining the QoS level of periodic tasks predefined by the application programmer. At this prospect, we compared the performance of the EDL server to that of the Background (BG) server which basically consists of scheduling aperiodic tasks when there is no red periodic instances to execute. Moreover, with the BG server, blue instances are executed as background tasks when there is neither red periodic instances nor aperiodic requests ready for execution.

The simulation context includes 5 periodic tasks under QoS contraints with uniform s_i. Their worst-case execution time depends on the setting of the total periodic load U_p. Aperiodic tasks are randomly generated according to the desired aperiodic load U_s.

4.1 Minimizing aperiodic tasks response time

First, the paper shows simulation results in terms of the average aperiodic response time, verifying that the EDL algorithm offers significant performance improvements over other conventional joint scheduling algorithms (like BG), especially under a heavy load. In the graphs, the average aperiodic response time is normalized with respect to the computation time.

Simulation results reported on Figure 3 are carried out, for two different values of s_i, varying the periodic load ($20\% \leq U_p \leq 110\%$), while the aperiodic load remains constant ($U_s = 40\%$).

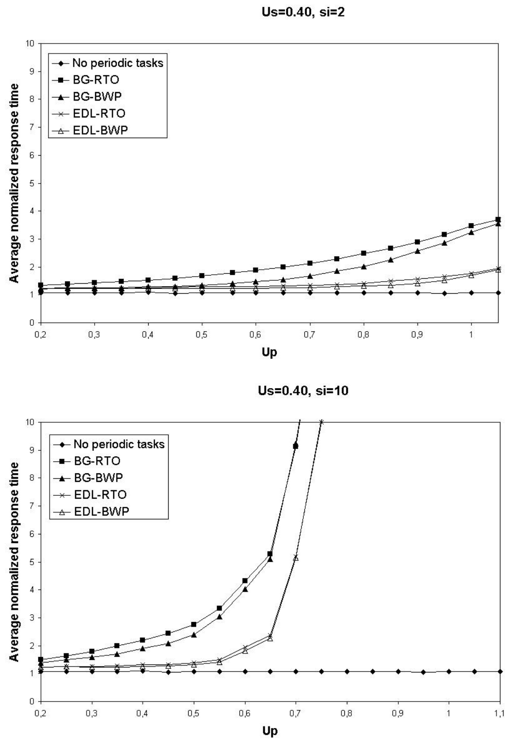


FIG. 3 – *Average aperiodic response time*

From the graphs, we can say that the EDL server offers better performance than the BG server. Moreover, we can note that this advantage is all the more significant as the periodic load U_p is higher. In addition, results are better when the two servers are used with the BWP model. Finally, for a high periodic load, variations of s_i parameter show that the response time is all the less as the QoS observed for periodic tasks is low.

4.2 Guaranteeing a QoS level for periodic tasks

As EDL-RTO and EDL-BWP give the same performance in terms of aperiodic responsiveness, we are now interested in the evaluation of the QoS observed for periodic tasks, for these two models. Measurements rely on the ratio of periodic tasks which complete before their deadline. The evaluation is done for a periodic load that varies from 20% to 110% while the aperiodic load is kept constant. The results obtained for $s_i = 2$ are described on Figure 4.

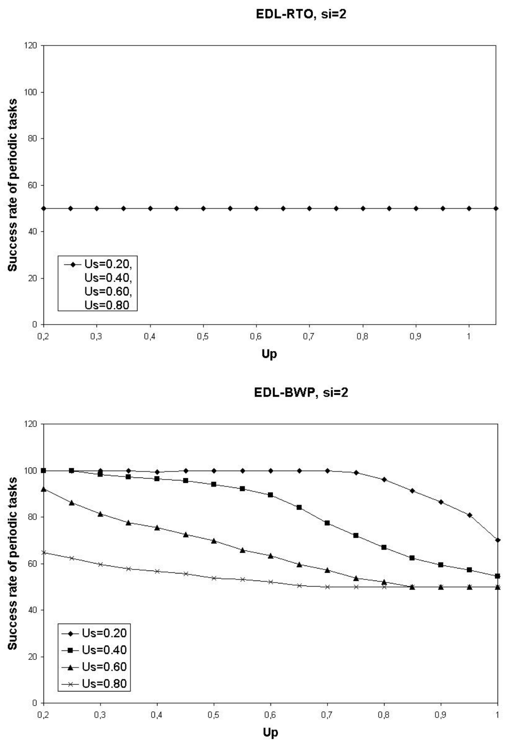


FIG. 4 – *QoS of periodic tasks*

From the graphs, we note that EDL-BWP provides better performance than EDL-RTO. Nevertheless, EDL-RTO offers an efficient solution to the problem of the idle times determination, with a low computation complexity. This is offset by a constant QoS observed for periodic tasks, directly depending on parameter s_i (here with $s_i = 2$, we observe that only 50% of periodic task instances are completed within their deadlines). EDL-BWP requires more computation time for the management of aperiodic requests, but provides better performance. In summary, EDL-BWP appears to perform better than EDL-RTO, as long as the total load remains low. For heavy loads, both algorithms tend to have the same behavior. Note that the advantage of EDL-BWP over EDL-RTO is wider when tolerated violations of deadlines are important (small value of s_i).

5. Conclusions

Real-time and embedded systems often need to provide their services within strict time deadlines. However, with the emergence of new applications ranging from multimedia to monitoring, real-time periodic tasks may present occasional deadline violations. In this paper, our main contribution was actually to merge two existing approaches and to evaluate it. We have considered the problem of jointly scheduling periodic tasks with skips and soft aperiodic tasks. Through the results, we show to what extend the merging of the Skip-Over and the EDL approaches minimize aperiodic responsiveness, while a QoS level (*i.e.*, skip parameter established by the application programmer) is always guaranteed for periodic tasks. This scheduling scheme is being under integration in the library of open-source components available with the CLEOPATRE[1] kernel [7].

References

[1] 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

[2] M. Caccamo and G. Buttazzo: *Exploiting skips in periodic tasks for enhancing aperiodic responsivess*, In Proceedings of the 18th IEEE Real-Time Systems Symposium (RTSS'97), Dec 1997

[3] 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

[4] 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

[5] 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

[6] 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

[7] *http://www.cleopatre-project.org*

1. work supported by the French research office, grant number 01 K 0742