Stability and Robustness Issues in Scheduling Periodic Tasks with Firm Real-Time Requirements

Audrey Marchand and Maryline Silly-Chetto
IRCCyN (Institut de Recherche en Communications et Cybernétique de Nantes)
UMR CNRS 6597 - 1, rue de la Noë - BP 92101
44321 Nantes Cedex 03, France
Email : {audrey.marchand, maryline.chetto}@univ-nantes.fr

Abstract

This paper is concerned with dynamic scheduling of firm periodic tasks on a uniprocessor system. Periodic tasks are defined under Quality of Service (QoS) constraints. More precisely, tasks can occasionally skip one instance according to the definition of the Skip-Over model. Under this mechanism, each task implements two types of instances, respectively called red and blue. The former must complete within its deadline whereas the latter can be aborted at any time. The work presented here focuses on one of the two classical skip-over algorithms, namely BWP. The intent of the paper is to underline the impact of the scheduling of blue tasks on the stability of the system, here defined in terms of individual success balancing. We present two scheduling strategies, called BWP-LF and BWP-MS. Simulation results are reported in order to show performance improvement obtained with the proposed scheduling algorithms.

1. Introduction

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.

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. For soft systems, it is acceptable to miss some of the deadlines occasionally. In firm systems, tasks are also allowed to miss some of their deadlines. Typical illustrating examples of systems with firm real-time requirements are multimedia and automotive control systems.

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. These new scheduling techniques can deal with the problem of maintaining satisfactory performance under overload conditions.

There have been some approaches to the specification and design of real-time systems that tolerate occasional losses of deadlines. Hamdaoui and Ramanathan in [2] 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. The Skip-Over model was introduced by Koren and Shasha [3]. 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 this paper we address the problem of analyzing performance of real-time systems that feature the Skip-Over model for specifying tolerated deadline violations. In this context the objective is to show, to what extent, classical Skip-Over scheduling algorithms are not stable in terms of task success balancing, and then to provide stable variants.

The remainder of the paper is organized as follows : Section II presents relevant background materials about the Skip-Over model. In Section III, we perform an analysis of classical Skip-Over algorithms in terms of stability. Further, we describe two dynamic scheduling algorithms, namely BWP-LF and BWP-MS, based on stability requirements. The performance analysis of BWP-LF and BWP-MS, in terms of individual success ratio of tasks, is reported in Section V. Finally, in Section VI, we summarize our contribution.

2. Background and terminology

2.1. The Skip-Over Model

We are here interested in the problem of scheduling periodic tasks which allow occasional deadline violations, on a uniprocessor system. 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 [3]. 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.

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.

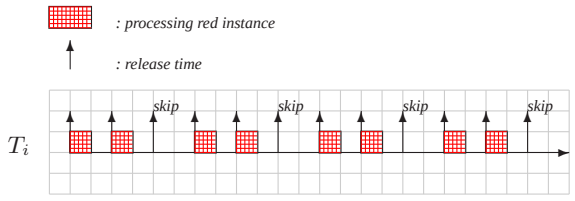

Figure 1 – RTO scheduling algorithm ($s_i = 3$)

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.

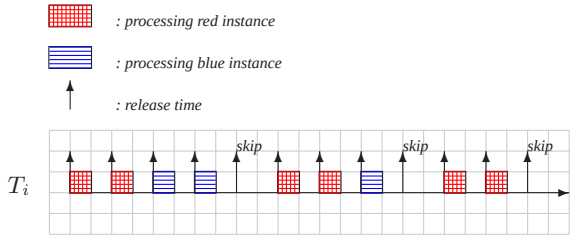

Figure 2 – BWP scheduling algorithm ($s_i = 3$)

2.2. Robustness Criterion

Robustness is one of the important characteristics of real-time scheduling strategies since it refers to the global success ratio (the total number of task completions) for a given task set, i.e. to the absolute global performance of the system. Let us consider the following definition of robustness for a computer system, proposed in [6] :

Definition 1 *A scheduling algorithm X is more robust than a scheduling algorithm Y if the global success ratio with X is greater than the global success ratio with Y.*

What is interesting here is not quantifying this term but using it in a relative manner in order to compare several scheduling strategies. For that purpose, a dynamic scheduling algorithm, called RLP (Red as Late as Possible) based on the EDL (Earliest Deadline as Late as Possible) strategy [1, 7] has been recently proposed in order to improve the robustness of basic Skip-Over algorithms. Results are reported in [5].

2.3. Stability Criterion

Whereas stability is a very precise concept in the control theory, it may have various definitions in the real-time context. For example, Stankovic [8] defined stability of a distributed scheduling algorithm in terms of load balancing (i.e. a system is said to be stable if the load on any two nodes does not differ by more than x percent). We give here the definition in terms of success balancing proposed in [6] which resembles that of fairness :

Definition 2 *A scheduling algorithm X is more stable than a scheduling algorithm Y if the greatest difference in success ratio of any tasks with X is less than the greatest difference in success ratio of any tasks with Y.*

Note that stability does not refer to the ability of the system to maintain a certain level of performance. This characteristic would still permit aberrant behavior of the system such as degrading the global success ratio but keeping individual success balanced nevertheless.

3. Analyzing Skip-Over Algorithms Stability

By definition, RTO scheduling algorithm is very stable since it never attempts to schedule blue task instances. Consequently, tasks have always the same individual success ratio related to red task completions, but this is coupled with a poor global performance (for $s_i = 2$ where one instance every two has to complete within its deadline, the global success ratio is equal to 50%).

That's why we will focus here on the behavior of the system in terms of individual success ratios, using the BWP scheduling strategy. The objective is to underline the lack of stability of this algorithm where ready blue task instances are ordered by increasing deadlines, i.e. according to Earliest Deadline (ED). That is why, we will denote this strategy by BWP-ED.

3.1. Simulation context

The simulation context includes 50 periodic task sets, each consisting of 10 tasks with a least common multiple equal to 3360. Tasks are defined under QoS contraints with uniform $s_i = 2$. Their worst-case execution time depends on the setting of the periodic load U_p and is randomly generated in order to reflect the greatest number of applications. Deadlines are equal to the periods and greater than or equal to the computation times. Simulations have been processed over 10 hyperperiods.

3.2. Varying periodic load

The behavior of BWP-ED is studied for a periodic load U_p varying from 90% to 160%. Both individual and global success ratios are depicted on Figure 3. Tasks T_i are ordered such that $i < j$ implies task periods $P_i > P_j$.

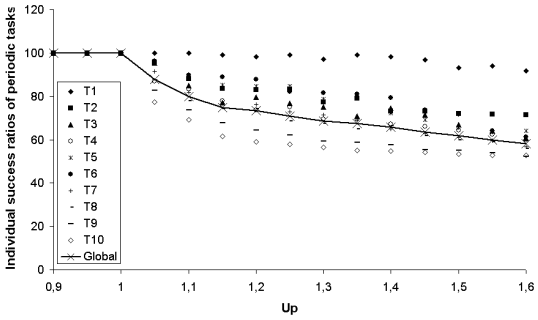

Figure 3 – Individual success ratios under BWP-ED

Results show that BWP-ED algorithm functioning related to blue tasks scheduling leads to give more importance to some tasks than to others. Individual success ratios are really scattered around the curve depicting the global success ratio. As a matter of fact, BWP-ED leads to provide preferential treatment to tasks with long periods, which is mainly due to the fact that deadline ties are broken in favour of the task with the earliest release time. Consequently, when two tasks have the same absolute deadline, the task with the longest period is chosen for execution.

This phenomena is illustrated by simulation results : for instance, for $U_p = 1.40$, we observe that more than 98% of the T_1 instances succeed in completing within their deadline while less than 55% do so for task T_{10}, thus involving a high standard deviation.

4. Proposed Scheduling Algorithms

As mentioned above, performance evaluation of a firm scheduling strategy must be performed not only on the basis of the global success ratio but rather must describe the behavior of every task which is characterized by its individual success ratio. The need of designing systems which are as stable as possible has led us to study new blue tasks scheduling algorithms. As the previous analysis brought out the fact that BWP-ED lacks stability, two novel scheduling strategies have been defined in order to improve this criterion.

The proposed scheduling algorithms are inspired by the work described in [6] which deals with the problem of analyzing the performance of real-time control systems that feature the Deadline Mechanism [4] for on-line recovery from timing faults.

4.1. BWP-LF algorithm

BWP-LF (Blue When Possible - Last Failure) algorithm schedules at each time instant, the ready blue task whose number of successive successes from the last failure is least. The earliest deadline rule is used to break ties between blue tasks of equal priorities.

4.2. BWP-MS algorithm

BWP-MS (Blue When Possible - Minimum Success) algorithm schedules at each time instant, the ready blue task whose individual success ratio computed from the initialization time is least. As for the BWP-LF case, ties are broken in favor of the task with the earliest deadline.

These two variants of BWP-ED scheduling algorithm ensure that any task will have the highest priority within finite time and no task will indefinitely keep the highest priority.

5. Performance Evaluation

BWP-LF and BWP-MS scheduling algorithms have been evaluated using the same simulation context as the one described in Section III.A for BWP-ED. Figure 4 and Figure 5 show the results obtained with BWP-LF and BWP-MS algorithms respectively.

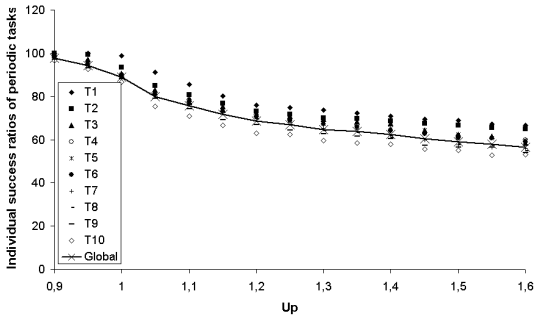

Figure 4 – Individual success ratios under BWP-LF

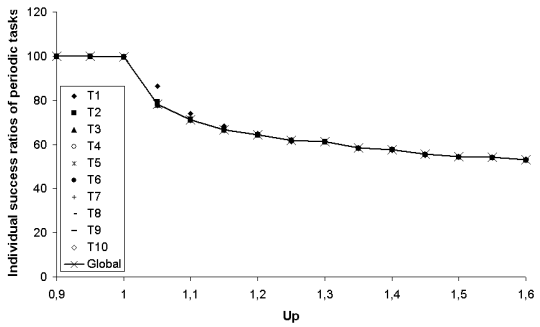

Figure 5 – Individual success ratios under BWP-MS

These experimental results indicate the feasibility of achieving good stability by use of BWP-LF and BWP-MS. The distribution of the individual success ratios has been significantly improved by the proposed algorithms. Individual success ratio of tasks with short period has been increased by 10% on average.

The main criteria selected for the stability performance evaluation of the different strategies are summarized un

Table 1, namely the mean difference d_{mean}, the maximal difference d_{max} and the standard deviation σ computed on the individual success ratios.

The standard deviation is the most common measure of statistical dispersion. Simply put, standard deviation measures how spread out the values in a data set are. If the data points are all similar, then the standard deviation will be low (closer to zero).

Algorithms	d_{mean}	d_{max}	σ
BWP-ED	30.88	43.86	8.89
BWP-LF	12.65	15.77	3.73
BWP-MS	1.24	8.71	0.38

Table 1 – Relevant stability criteria

Whereas the mean distance between two individual success ratios is respectively equal to 30.88% for BWP-ED and 12.65% for BWP-LF, it is reduced to 1.24% for BWP-MS. Similarly, as regards the maximal difference of individual success ratios, we observe a big gap between the different strategies. The evaluation of the standard deviation σ reinforces the fact that BWP-MS is the most stable algorithm. Results obtained for BWP-LF are also acceptable compared with the great dispersion of individual success ratio of the basic BWP-ED algorithm.

6. Conclusions

This paper has considered the problem of adding stability to firm real-time systems whose tolerance to deadline violations are specified according to the Skip-Over model. The work presented here underlined the impact of the scheduling of blue tasks on the stability of the system, here defined in terms of success balancing. As the classical BWP-ED algorithm does not perform well, we were led to design and test new heuristics. A performance evaluation of them was presented, with respect to our specific stability issues. Future work includes extending this approach to a new scheduling strategy, RLP (Red as Late as possible) proposed in [5] which provides a better quality of service than BWP-ED while guaranteeing stability and robustness.

7. References

[1] 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, 1989.

[2] M. Hamdaoui and P. Ramanathan, "A dynamic priority assignment technique for streams with (m,k)-firm deadlines", *IEEE Transactions on Computers*, Vol. 44, No. 4, pp 1443-1451, 1995.

[3] 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.

[4] A-L. Liestman and R-H. Campbell, "A fault tolerant scheduling problem", *In Proceedings of the IEEE Transactions on Software Engineering*, Vol. 12, No. 10, pp 1089-1095, 1986.

[5] A. Marchand and M. Silly-Chetto, "RLP : Enhanced QoS Support for Real-Time Applications", *In Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems ans Applications*, Hong-Kong, 2005.

[6] M. Silly-Chetto, "On the stability of scheduling algorithms for real-time control", *IMACS/IEEE-SMC Computational Engineering in Systems Applications Multiconference*, Lille, France, 1996.

[7] M. Silly-Chetto, "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] J. Stankovic, "Stability and distributed scheduling algorithms", *IEEE Transactions on Software Engineering*, Vol. SE-11, No.10, pp 1141-1152, 1985.