A Solution to the Time Jitter Removal in Deadline Based Scheduling of Real-Time Applications

L. David & F. Cottet

LISI-ENSMA
BP 40109
1, Av. Clément Ader
86961 Futuroscope Cedex
{david, cottet}@ensma.fr

SUPERVISOR : Francis COTTET
KEYWORDS : Temporal jitter, on line scheduling, scheduling algorithms

Abstract. One typical feature of real-time systems is concurrent processing with timing constraints. These timing constraints, for example, can deal not only with the deadline meeting but also with the task execution regularity. For application in hard real-time systems, repetitive requests are usually designed with the periodic task model from Liu and Layland [1]. However, in this model, each task execution on any period is independent in term of scheduling from the same task execution on the other periods: the task execution regularity is hardly carried out. Yet, unpredictable delays or jitter in the task periodic executions may be quite disturbing for some real-time applications such as process controls (sampled data systems, P.I.D. control, etc.). We propose in the following a method based on the modification of the task temporal parameters in the well known DM and EDF algorithm context, in order to master the jitter in particular conditions.

I. Introduction

A real-time computer controlling a physical device or a process often involves following basic functions: sampling readings from sensors at periodic intervals, and responses to different scenarios by sending signals to actuators with a time-bound. Respect to the process requirement, those periodic intervals must be sometimes as periodic as possible. In fact, we know that is not necessary the case with real-time computers, for which priorities, preemptiveness, shared resources, etc. must be taken into account. In many real-time applications however, there are regularity execution constraints, e.g., in multimedia data transmission [2], in systems for which fault-tolerance property is a paramount parameter [3], in control systems for which the main feature is the execution smoothness (robots that carry hazardous stuff) [3], in A.T.M. (Asynchronous Transfer Mode) networks [4], etc. Di Natale and Stankovic presented in [5] an off-line scheduling method with jitter control issue in distributed context. One other off-line method, solution to the jitter control problem, was addressed with an exhaustive research of all valid sequences [6], [7]. Regarding to regular execution constraints, Han and Lin also addressed a method to master the jitter but proposed some specific scheduling algorithms [8]. In this paper, we address a jitter control issue for the on-line scheduling context, applied to periodic constraint

tasks, that we call indifferently regular tasks. In the first section, we deal with the jitter calculus formalization attached to a periodic task. Then we present a method for jitter control of regular task with a unit execution duration. Finally we illustrate this method through a six task configuration example.
In this study, we care about mono processor environment with periodic and independent tasks. We use the Liu and Layland model [1], for which each task is characterized with four parameters: ($r_{i,1}$, C_i, R_i, P_i). $r_{i,1}$ is the first request time, C_i the maximum execution duration, D_i the relative deadline, P_i the task period. (see fig. 1). C_i is supposed to be equal to 1, which corresponds to the temporal granularity of the scheduling process. Let denote with X^* the new value of the variable X after a modification, X can be "r" or "D" variables.

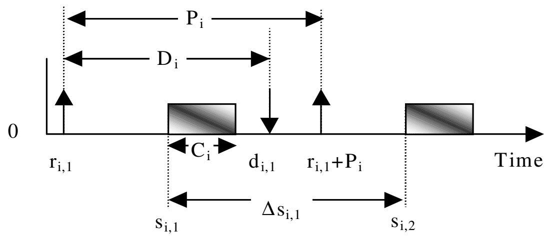

Fig. 1. The task model and the starting date definition

II. The jitter

The execution irregularity described above can be assessed in different ways. In the following, we determine it on the scheduling period with the use of statistical tools. This scheduling period for a periodic task configuration is given by [9] [10][1]:

$$P_{sched} \leq \max_i(r_i) + 2LCM(P_i) \quad \textbf{(1)}$$

for not synchronized start tasks, and by:

$$P_{sched} = LCM(P_i) \quad \textbf{(2)}$$

otherwise.

Let's now formalize this assessment: let $s_{i,k}$ be the k^{th} starting execution date of the i^{th} task from a periodic task set (see fig. 1). The duration between 2 successive starting dates of the same task are defined by the difference:

$$\Delta s_{i,k} = s_{i,k+1} - s_{i,k} \quad \textbf{(3)}$$

[1] LCM is the least common multiple

where k denotes the k^{th} difference within the scheduling period. $\Delta s_{i,k}$ spreads from 0 limit to $2P_i$ limit (characterized by deadline equal to period with negligible execution time). Let's introduce then:

$$g_{S_{i,k}} = \frac{\left|\Delta s_{i,k} - P_i\right|}{P_i} 100\,\% \qquad \textbf{(4)}$$

which assesses, in percentage, the starting execution date irregularity between the k^{th} instance and the $k+1^{th}$ instance:

- if the difference equals P_i (regular tasks), then $g_{S_{i,k}} = 0\,\%$.
- if the difference tends to 0 or to $2P_i$ (maximum irregularity in the execution), then $g_{S_{i,k}} = 0\,\%$.

Given T_i a task, the arithmetic mean of variable $g_{S_{i,k}}$ defines the mean temporal jitter:

$$G_{moy}(T_i) = \frac{1}{N_i} \sum_{k=1}^{N_i} \frac{\left|\Delta s_{i,k} - P_i\right|}{P_i} 100\,\% \qquad \textbf{(5)}$$

where N_i is the interval numbers between two successive instances on the same task within the scheduling period.

III. The jitter control issue

Designing a real-time application, the P_i parameter depends on process dynamic characteristics and thus can not be altered. C_i may be interpreted as the worst case execution time of task T_i and may include a bounded computation time due to context switching, scheduling, etc. Therefore, in order to modify the execution sequence of an application, we can alter only the two other parameters $r_{i,1}$ and D_i. In order to not contradict the initial requirements, note that the modified parameters $r_{i,1}^*$ and D_i^* must satisfy the following inequalities : $r_{i.1}^* \geq r_{i.1}$ and $D_i^* \leq D_i$.

Our proposed way of constructing a real-time system including regularity constraint tasks is based on three main steps. The first step consists of determining asynchronized release times of all the regular task frames. In fact, if two regular tasks have conflicting execution requests, undoubtedly, one of them will get a time jitter. The second step deals with fixing the deadlines of these tasks in order to assure the highest priority when they require the processor. An implication of that is to use a scheduling algorithm based on task deadline such as DM and EDF. Since the task parameters have been modified during the previous two steps, it is necessary in the last phase to check whether the derived task set is schedulable. This schedulability analysis can be carried out either by applying analytic criteria or by simulating over a simulation time P_{Sched}.

II.1 Request Date Modification

Let T_1, T_2, …, T_n denote a set of regular periodic tasks belonging to a task system of m tasks, with m > n. The other tasks T_j with j ∈ $[n+1, m]$ are only periodic tasks.

Moreover, the values of execution times C_1, C_2, …, C_n are equal to 1. First we desire to find $r_{1,1}$, $r_{2,1}$, …, $r_{n,1}$ of those regular tasks so that no activation dates of all their frames are identical within the scheduling period. This specification can be formally written for 2 tasks as:
Given two periods (P_i, P_j) from the set of regular tasks with $i \neq j$, we choose $r_{i,1}$ and $r_{j,1}$ so that (see fig. 2):

$$(\forall k \in \mathbb{N})(\forall k' \in \mathbb{N}) \; r_{i,1} + k \cdot P_i \neq r_{j,1} + k' \cdot P_j \qquad \textbf{(6)}$$

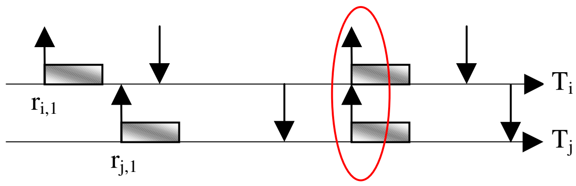

Fig. 2. Conflicting execution requests of T_i and T_j

Introducing the modulo mathematics notation (mod) and the greatest common divisor (Λ), and according to the work of [11], [12] and [13], this problem can be reduced to the seeking of a natural integer couple ($r_{i,1}$,$r_{j,1}$) verifying:

$$r_{j,1} - r_{i,1} \neq 0 \bmod (P_i \wedge P_j) \qquad \textbf{(7)}$$

where P_i and P_j are necessary prime at each others

Coming back to the set of n tasks, the problem consists in seeking ($r_{1,1}$,$r_{2,1}$,…,$r_{n,1}$) satisfying:

$$\forall (i,j) \in [\![1;n]\!]^2 \text{ with } i \neq j, \; r_{j,1} - r_{i,1} \neq 0 \bmod (P_i \wedge P_j) . \qquad \textbf{(8)}$$

with however a necessary condition on periods:
Remark 1. Before any research of $r_{i,1}$, we have to check whether all the periods are pairwise relatively prime.
This arithmetic problem (8) is a quite intricate one, but, in some particular conditions, we can characterize some solutions in a straightforward way. For example:
Proposition 1. if $\min_{i,j}(P_i \wedge P_j) \geq n$, then there are some solutions to the problem. The set denoted by $r_{1,1}=0$, …, $r_{n,1}=n-1$ is a solution as well as any permutation of this set.

Yet, this problem (8) remains complex. When analytical tools do not help us to characterize a solution, but do prove the solution existence, we have to look closer to enumeration techniques with a methodical research of $r_{i,1}$.

II.2 Priority modification

Then we have to be concerned with the instantaneous execution of the task. This is done by giving the highest priority to it by the means of the deadline parameter. Let us separately analyze the two scheduling algorithms :

- case DM. The derived deadline parameter is then determine such that:

$$(C_i = 1) \le D_i^* \le \min_{j \ne i}(D_i, D_j - 1) \quad \textbf{(9)}$$

where j describes the set of non necessary regular tasks. So the task T_i gets a relative deadline strictly smaller that all the D_j, and so gets the higher priority. This must be applied to each regular task considered in any order because their activation requests are always separated according to the specification (8).

• case EDF. In this case, in order to obtain the highest priority for a task, it is achieved with setting minimum:

$$D_i^* = C_i = 1 \quad \textbf{(10)}$$

It should be pointed out that we do not get one solution but a set of modified parameters from the equations (8) and (9) or (10), and therefore a newly derived set of tasks. Given such a derived task set, we have to check the schedulability.

IV. Example

In order to illustrate this methodology, let's take a 6 task configuration in which two are sampling tasks with regular execution constraints (see table 1). We choose the execution duration equal to one.

Table 1. A 6 task configuration example and the acquisition task jitters.

Tasks	$r_{i,1}$	**Ci**	**Ri**	**Pi**	**Jitters** DM - EDF	**Comments**
Acquisition 1	0	1	8	8	9,4% - 10,9%	Rapid control (ex.: fluid stream control)
Treatment 1	0	2	8	8		
Control 1	0	1	7	8		
Acquisition 2	0	1	18	18	18,5% - 11,1%	Slow control (ex. : temperature)
Treatment 2	0	4	17	18		
Control 2	0	1	6	6		Fast control (ex. warning)

Those parameters derives a scheduling period of 72 processor units, a processor load of 94.5 % and some jitter release for the two data acquisition tasks reported in table 1. Let's mention that should we had to oversample the signals, the configuration would have not be schedulable. Then we apply the above method on the two regular tasks in DM and EDF context. Defining the new temporal parameters with those values: ($r_{1,1}$=0, $r_{2,1}$=1) and ($1 \le D_1^* \le 5$; $1 \le D_2^* \le 5$) in DM context, and with ($r_{1,1}$=0, $r_{2,1}$=1), (D_i^*=1) in EDF context, we get the desired results after schedulability verification: the jitters are cancelled.

V. Conclusion

In the context of real-time applications involving regularity constraints tasks, we propose a scheduling method which keeps the deadline based scheduling algorithms (DM , EDF). Indeed, the method is divided into three steps: asynchronization of request dates for all the regular tasks, redefinition of priorities so that those tasks are prior from the others, and finally schedulability verification. Then, we get a task configuration where the regular task periods are strictly respected.

In a further work, we shall investigate the problem of regular tasks for any execution duration. We shall also be less restricted on the constraints so that we shall specify a maximum jitter in which the regular tasks are allowed to swing. Dependent tasks with or without shared resources, as well as distributed environment will be also the next step parts of this study.

References

1. Liu, C.L. and J.W. Layland, *Scheduling algorithms for multiprogramming in real-time environment.* Journal of the ACM, 1973. 20(1): p. 46-61.
2. Elsaadany, A. *Priority Communication Schemes on Local Computer Networks for Multimedia Traffic*. in: *Proc. of the 19th Conf. Local Computer Networks*. 1994: IEEE.
3. Lin, K.-J. and A. Herkert. *Jitter Control in Time-Triggered Systems*. in: *Proc. of the 29th Annual Hawaii International Conference on System Sciences*. 1996.
4. Boudec, J.-Y.L., *The Asynchronous Transfer Mode: a Tutorial.* Computer Networks and ISDN Systems, 1992. 24: p. 279-309.
5. DiNatale, M. and A. Stankovic. *Applicability of Simulated Annealing Methods to Real-Time Scheduling and Jitter Control*. in: *Proc. of Real-Time Systems Symposium*. 1995. Pisa (Italy).
6. Cheng, S.-T. and C.-M. Chen. *A Cyclic Scheduling Approach for Relative Timing Requirements*. in: *Proc. of the 3rd IEEE Real-Time Applications Worshop*. 1996.
7. Choquet-Geniet, A., D. Geniet, and F. Cottet. *Exhaustive Computation of the Scheduled Task Execution Sequences of a Real-Time Application*. in: *Proc. of the 4th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*. 1996. Uppsala, Sweden.
8. Han, C.-C., K.-J. Lin, and C.-J. Hou, *Distance-constrained scheduling and its applications to real-time systems.* IEEE Transactions on Computers, 1996. 45(7): p. 814-826.
9. Leung, J. and M. Merrill, *A note on preemptive scheduling of periodic real-time tasks.* Information Processing Letters, 1980. 11(3): p. 115-118.
10. Grolleau, E., A. Choquet-Geniet, and F. Cottet, *Decreasing schedule length in real-time system validation*, . 1997, Internal Report 97007: LISI-ENSMA Futuroscope, France.
11. Cottet, F., M. Courtès, and M. Holle. *Traitement de la gigue temporelle pour les ordonnancements par échéance*. in: *Proc. of RTS Conferences*. 1998.
12. David, L., F. Cottet, and E. Grolleau, *Maîtrise de la gigue temporelle avec les algorithmes d'ordonnancement DM et ED*, . 2000, Internal Report 00 003: LISI-ENSMA Futuroscope, France.
13. David, L., F. Cottet, and E. Grolleau. *Gigue temporelle et ordonnancement par échéance dans les applications temps réel*. in: *Proc. of Conf. Inter. Francophone d'Automatique (IEEE)*. 2000. Lilles, France.