Smooth Rate Adaptation through Impedance Control

Giorgio Buttazzo
Dept. of Computer Science
University of Pavia
buttazzo@unipv.it

Luca Abeni
RETIS Lab
Scuola Superiore S. Anna, Pisa
luca@sssup.it

Abstract

In many real-time applications involving human-computer interactions, the quality of service also depends on the way performance is changed during workload variations. When human factors affect performance measurements, smooth rate transitions are always preferred with respect to abrupt parameter changes.

In this paper, we propose a new methodology for automatically achieving smooth rate adaptation of a periodic task set during workload variations due to abrupt environmental or systems changes. Load balancing is performed using an elastic task model, according to which tasks utilizations are treated as damped springs with given elastic and damping coefficients. The model has been implemented on top a real-time kernel and experimental results are reported to show the effectiveness of the proposed approach.

1 Introduction

In real-time applications that involve human-computer interactions, the quality of a delivered service depends not only on the absolute level of performance, but also on the way performance is changed during workload variations. For example, while watching a movie, a continuous transition between color and black/white mode is considered much more annoying than watching the entire movie in black and white. In multimedia applications (e.g., sound reproduction, video playing, or graphical animations), transient degradations of the quality due to overload conditions are accepted much better when they occur gradually. In general, when human factors are involved in measuring the quality of a computing system, smooth rate transitions in periodic activities are always preferred with respect to abrupt period changes.

Typically, a rate change in a periodic activity may be caused either by the task itself, as a response to a variation occurred in the environment, or by the system, as a way to cope with an overload condition.

For example, in a visual tracking system, the sampling rate of a tracking task can be changed on line based on the target velocity. Also, in automotive applications, engine control tasks run with a rate proportional to the motor angular velocity.

In other situations, the possibility of varying tasks' rates increases the flexibility of the system in handling overload conditions, providing a more general admission control mechanism. For example, whenever a new task cannot be guaranteed by the system to meet its timing constraints, instead of rejecting the task, the system can try to reduce the utilizations of the other tasks (by increasing their periods in a controlled fashion) to decrease the total load and accommodate the new request.

The problem of rate adaptation during overload conditions has been widely considered in the real-time literature, although most of the proposed solutions do not address the issue of achieving smooth period adaptation.

For example, Kuo and Mok [10] propose a load scaling technique to degrade the workload of a system by adjusting the periods of processes. In this work, tasks are assumed to be equally important and the objective is to minimize the number of fundamental frequencies to improve schedulability under static priority assignments. In [16], Nakajima and Tezuka show how a real-time system can be used to support an adaptive application: whenever a deadline miss is detected, the period of the failed task is increased. In [18], Seto et al. describe a method for computing tasks' periods to minimize a performance index defined over the task set. This approach is effective at a design stage to optimize the performance of a discrete control system, but it cannot be used for on-line load adjustment. In [12], Lee, Rajkumar and Mercer propose a number of policies to dynamically adjust the tasks' rates in overload conditions. In [1], Abdelzaher, Atkins, and Shin present a model for QoS negotiation to meet both predictability and graceful degradation requirements in cases of overload. In this model, the QoS is specified as a set of negotiation options, in terms of rewards and rejection penalties. In [17], Nakajima shows how a multimedia activity can adapt its requirements during transient overloads by scaling down its rate or its computational demand. However, it is not clear how the QoS can be increased when the system is underloaded. Recently, in [19, 20], a feedback control mechanism has been considered for observing and adjusting the system workload to reduce the number of deadline misses when tasks' execution times are not precisely known. However, this approach does not permit to control each task's utilization individually. In [15], the previous approach has been extended to enable system designers to specify a desired behavior in terms of a set of performance metrics and apply existing control methods to design an adaptive real-time system to achieve the specified requirements. In [4], Beccari et al. propose several policies for handling overload through period adjustment. The authors, however, do not address the problem of increasing the task rates when the processor is not fully utilized.

Although these approaches provide solutions for changing task rates during overload conditions, they do not address the problem of controlling the quality of service during transitions.

In this paper, we propose a new methodology for automatically achieving smooth rate adaptation of a periodic task set during workload variations due to abrupt environmental or systems changes. Load balancing is performed using the elastic task model presented in [6], according to which tasks utilizations are treated as springs with given elastic coefficients. To achieve smooth rate transitions, however, the model is extended by coupling each spring with a damping device which prevents abrupt period changes. The model has been implemented on top a real-time kernel and experimental results are reported to show the effectiveness of the proposed approach.

The rest of the paper is organized as follows. Section 2 presents the task model and the basic assumptions. Section 3 briefly recalls the elastic approach. Section 4 presents the impedance control method and illustrates the approach used to achieve smooth rate transitions. Section 5 illustrates some experimental results achieved on the HARTIK kernel. Finally, Section 6 contains our conclusions and future work.

2 Task model

In our framework, each task is considered as flexible as a spring with a given rigidity coefficient and length constraints. In particular, the utilization of a task is treated as an elastic parameter, whose value can be modified by changing the period within a specified range. To achieve smooth rate transitions, however, the spring operates together with a damping device, characterized by a given damping coefficient.

Each task is characterized by five parameters: a worst-case computation time C_i, a minimum period T_{i_0} (considered as a nominal period), a maximum period $T_{i_{max}}$, an elastic coefficient E_i, and a damping coefficient B_i. The elastic coefficient specifies the flexibility of the task to vary its utilization for adapting the system to a new feasible rate configuration: the greater E_i, the more elastic the task. The coefficient B_i specifies the damping with which task τ_i performs a period transition: the greater B_i, the higher the damping effect, and hence the smoother the transition. Thus, an elastic task is denoted as:

$$\tau_i(C_i, T_{i_0}, T_{i_{max}}, E_i, B_i).$$

From a design perspective, elastic coefficients can be set equal to values which are inversely proportional to tasks' importance, whereas damping coefficients can be set equal to values which are directly proportional to the level of quality specified by the user during transients phases.

In the following, T_i will denote the actual period of task τ_i, which is constrained to be in the range $[T_{i_0}, T_{i_{max}}]$. Moreover, we define $U_0 = \sum_{i=1}^{n} C_i/T_{i_0}$ and $U_{min} = \sum_{i=1}^{n} C_i/T_{i_{max}}$.

Any period variation is always subject to an *elastic* guarantee and is accepted only if there exists a feasible schedule in which all the other periods are within their range. In our framework, tasks are scheduled by the Earliest Deadline First algorithm [14]. Hence, if $U_0 \leq 1$, all tasks can be created at the minimum period T_{i_0}, otherwise the elastic algorithm is used to adapt the tasks' periods to T_i such that $\sum \frac{C_i}{T_i} = U_d \leq 1$, where U_d is some desired utilization factor. It can easily been shown (see [8] for details) that a solution can always be found if $U_{min} \leq 1$.

3 Task compression algorithm

In this section we will briefly recall the compression algorithm originally presented in [6]. For a more general description of the elastic approach see

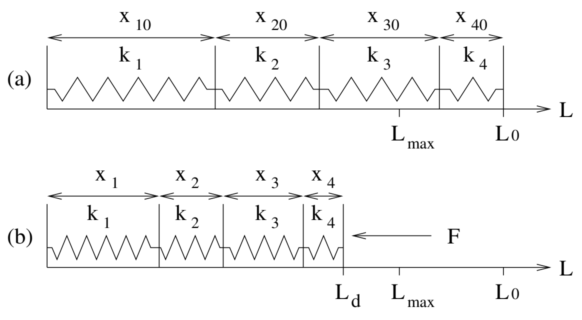

Figure 1. Compression of a linear spring system: the total length is $L_0 > L_{max}$ when springs are uncompressed (a); and $L_d < L_{max}$ when springs are compressed (b).

[8], where the algorithm has also been extended to be used in the presence of resource constraints.

3.1 The elastic model

The basic idea behind the elastic approach is to consider the utilizations of tasks as flexible as springs, which can be compressed and expanded (by acting on the periods) to conform to a desired workload. In particular, given a set of n periodic tasks with $U_0 > U_{max}$, the objective of the guarantee is to compress tasks' utilization factors in order to achieve a desired utilization $U_d \le U_{max}$ such that all the periods are within their ranges. In a linear spring system of total length L_0, this is equivalent to compressing the springs by a force F, so that the new total length becomes $L_d < L_0$. This concept is illustrated in Figure 1.

As proved in [8], if each period cannot exceed a maximum value T_{max}, the problem of finding a feasible period configuration requires an iterative solution. In fact, if while compressing tasks' utilizations one or more tasks reach their minimum utilization (i.e., their maximum period), the additional compression force must only deform the remaining tasks. Thus, at each instant, the set Γ of tasks can be divided into two subsets: a set Γ_f of fixed tasks having maximum period, and a set Γ_v of variable tasks that can still be compressed.

If $U_{i_0} = C_i/T_{i_0}$ is the nominal utilization of task τ_i, and U_0 is the sum of all the nominal utilizations, then to achieve a desired utilization $U_d < U_0$ each task has to be compressed up to the following utilization:

$$\forall \tau_i \in \Gamma_v \quad U_i = U_{i_0} - (U_{v_0} - U_d + U_f)\frac{E_i}{E_v} \tag{1}$$

where

$$U_{v_0} = \sum_{\tau_i \in \Gamma_v} U_{i_0} \tag{2}$$

$$U_f = \sum_{\tau_i \in \Gamma_f} U_{i_{min}} \tag{3}$$

$$E_v = \sum_{\tau_i \in \Gamma_v} E_i. \tag{4}$$

If there exist tasks for which $U_i < U_{i_{min}}$, then the period of those tasks has to be fixed at its maximum value $T_{i_{max}}$ (so that $U_i = U_{i_{min}}$), sets Γ_f and Γ_v must be updated (hence, U_f and E_v recomputed), and equation (1) applied again to the tasks in Γ_v. If there exists a feasible solution, that is, if the desired utilization U_d is greater than or equal to the minimum possible utilization $U_{min} = \sum_{i=1}^{n} \frac{C_i}{T_{i_{max}}}$, the iterative process ends when each value computed by equation (1) is greater than or equal to its corresponding minimum $U_{i_{min}}$.

All tasks' utilizations that have been compressed to cope with an overload situation can return toward their nominal values when the overload is over. Let Γ_c be the subset of compressed tasks (that is, the set of tasks with $T_i > T_{i_0}$), let Γ_a be the set of remaining tasks in Γ (that is, the set of tasks with $T_i \le T_{i_0}$), and let U_d be the current processor utilization of Γ. Whenever a task in Γ_a decreases its rate, all tasks in Γ_c can expand their utilizations according to their elastic coefficients, so that the processor utilization is kept at the value of U_d.

Now, let U_c be the total utilization of Γ_c, let U_a be the total utilization of Γ_a, and let U_{c_0} be the total utilization of tasks in Γ_c at their nominal periods. It can easily be seen that if $U_{c_0} + U_a \le U_{max}$ all tasks in Γ_c can return to their nominal periods. On the other hand, if $U_{c_0} + U_a > U_{max}$, then the release operation of the tasks in Γ_c can be viewed as a compression, where $\Gamma_f = \Gamma_a$ and $\Gamma_v = \Gamma_c$. Hence, it can still be performed by using equations (1), (2), (3) and (4).

4 Impedance control

When dealing with damped springs, each elastic element can be modeled as shown in Figure 2.

For the sake of completeness, a damped spring is a special case of a system which behaves as a mechanical impedance, with stiffness k, damping b, and mass m, as shown in Figure 3.

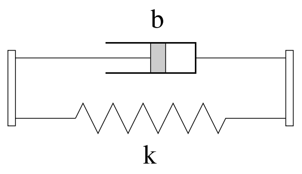

Figure 2. A damped elastic element.

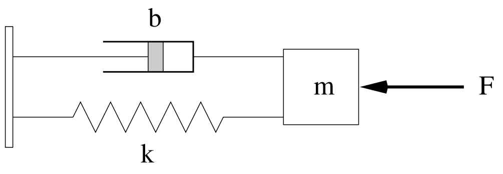

Figure 3. A generic mechanical impedance.

Such a system responds to an external force F as a second-order system according to the following equation:

$$F = m\ddot{x} + b\dot{x} + kx.$$

For linear, time-invariant continuous systems, the impedance Z is defined as the ratio of the Laplace transform of the effort (F) to the Laplace transform of the flow ($\phi = \dot{x}$), hence

$$Z(s) = \frac{F(s)}{\Phi(s)} = \frac{F(s)}{sX(s)} = ms + b + k/s.$$

From a system point of view, the input/output behavior of a linear system like this is described by the ratio of two variables: the effort and the flow [3]. For a mechanical system, effort is represented by force and torque, and flow is represented by linear and angular velocity. Motors and batteries are equivalent from a system point of view, both being effort sources. Similarly, a current generator or a rotating shaft are both flow sources.

Hence, using an electrical comparison, a mass (inertial element) is equivalent to an inductive element, a damper (dissipative element) is equivalent to a resistor, whereas a spring (conservative element) is equivalent to a capacitor. In this comparison, a force corresponds to a voltage generator, whereas the speed corresponds to a current. The equivalent electrical circuit is depicted in Figure 4 and the impedance $Z(s)$ is expressed as

$$Z(s) = Ls + R + \frac{1}{Cs}.$$

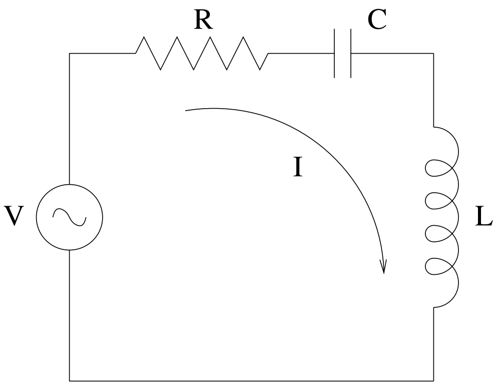

Figure 4. An electrical impedance.

Thus, a damped spring is equivalent to an RC circuit, for which we can write:

$$I(s) = \frac{V(s)}{Z(s)} = \frac{V(s)}{R + 1/Cs} = \frac{Cs}{RCs + 1}V(s).$$

Hence, the position of the damping device can be computed as

$$X(s) = \frac{RI(s)}{s} = \frac{RC}{RCs + 1}V(s).$$

Therefore, the transfer function of our system can be rewritten as

$$G(s) = \frac{X(s)}{V(s)} = \frac{a}{s + a}$$

where $a = \frac{1}{RC}$ is the system's pole for the RC circuit, and $a = \frac{k}{b}$ in the case of a damped spring. Using Z-transform, the transfer function can be expressed in a discrete time domain, as follows:

$$\begin{aligned} G(z) &= \mathcal{Z}\left[\left(\frac{1 - e^{-sT}}{s}\right)\frac{a}{s + a}\right] \\ &= (1 - z^{-1})\mathcal{Z}\left[\frac{a}{s(s + a)}\right] \\ &= \frac{z - 1}{z}\mathcal{Z}\left[\frac{1}{s} - \frac{1}{s + a}\right] \\ &= \frac{z - 1}{z}\left(\frac{z}{z - 1} - \frac{z}{z - p}\right) \\ &= \frac{1 - p}{z - p} = \frac{(1 - p)z^{-1}}{1 - pz^{-1}}. \end{aligned}$$

where T is the sampling period and $p = e^{-aT}$.

From $G(z)$, the discrete time equation expressing the position $x(t)$ of the damped spring as a function of the force $F(t)$ becomes:

$$x(t) = (1 - p)F(t - 1) + px(t - 1). \qquad (5)$$

We observe that any transient law can be used to perform a transition from a period to another. The one expressed by equation (5) is just the one which describes the change occurring in a damped spring, which is exponential. In the experiments we performed on the kernel, we also evaluated the effects caused by a linear period transition.

4.1 Implementation notes

To test the effectiveness of the adaptation algorithm, the proposed approach has been implemented on top of the HARTIK kernel [5, 11, 2], as a middleware layer. In particular, the elastic guarantee mechanism has been implemented as an aperiodic task, the `Elastic Manager` (EM), activated by the other tasks when they are created or when they want to change their period. Whenever activated, the EM calculates the new periods and changes them atomically. The overhead caused by the elastic mechanism can easily be taken into account since the EM is handled using a CBS server with a bounded utilization ($U_{EM} = 0.05$).

The graceful rate adaptation mechanism has been implemented on top of the EM, as a periodic task, the `Damping Manager` (DM). The purpose of the DM is to perform the rate transition according to the transition law set by the user. To bound the transition time, the DM runs with a period T_{DM} and performs a full transition in N steps, requiring an interval of NT_{DM} time units.

The DM task can be in two states: active or idle; when the system is started, the DM is idle, and it becomes active when some other task wants to change its period. When the DM task becomes active at time t_0, instead of changing the periods immediately it gracefully changes them during a transient of size $T = NT_{DM}$.

After N activations, the periods arrive to their final values and the period adapter returns to its idle state, waiting for the next request. More specifically, the gracefull adaptation mechanism works as follows:

- When a task τ_i wants to change its period from T_i to T_i^{new}, it posts a request to the DM.
- When the DM is idle and receives a new request, it becomes active and computes the next period value $T_i(k)$ according to the transition law set by the user. We note that $T_i(0) = T_i$, and after N steps $T_i(N) = T_i^{new}$.
- At each period T_{DM}, the DM updates the period T_i to the next value $T_i(k)$ and invokes the EM to achieve a feasible configuration.
- After N activations, the periods are adjusted to their final values, and the period adapter returns to its idle state.

There are some details to be considered in the implementation of the Damping Manager.

1. Transient values can be generated using a generic law $T_i(k) = f(k, T_i(k-1), T_i^{new})$. In this paper, we considered a linear function $T_i(k) = T_i(k-1) + \frac{T_i^{new} - T_i(0)}{N}$ and an exponential function derived from a typical RC circuit as the one shown by equation (5): $T_i(k) = (1-p)T_i^{new} + pT_i(k-1)$, where $p = exp(-aT_{DM})$, and $a = \frac{1}{E_i B_i}$.
2. The periods converge to their final values in a transient time $T = NT_{DM}$ that can be specified by the user. If $T = 0$ ($N = 0$), the period change is immediate and the system is equivalent to the classical spring system presented in [6]. Hence, the original elastic model is a special case of the generalized model presented here.
3. If a task requests a period change while the adapter is still active, the request is enqueued, and will be served only when all the periods will be stabilized. This has been done in order to simplify the adapter, but we are planning to remove this constraint in a future work.

5 Experimental Results

To verify the behavior of the Damping Manager, we performed an experiment with 4 tasks having the parameters shown in Table 1. To avoid deadline misses during transitions, periods are changed at the next release time of the task whose period is decreased. If more tasks ask to decrease their period, the EM will change them, if possible, at their next release time. See [8] for a theoretical validation of the compression algorithm.

Considering the utilization reserved for the EM, the DM and other device handlers in the system, the effective total utilization U_{max} available for the task set is 0.782. Since $23/100 + 23/100 + 23/100 + 23/100 > 0.782$, the periods are initially expanded by the elastic law, and the tasks start with current periods different from their nominal periods: $T_1 = 107$, $T_2 = 107$, $T_3 = 122$, and $T_4 = 143$. At time $t = 5$, τ_1 issues a request to change its period to $T_1^{new} = 50$, and the DM starts to gracefully adapts the periods. At time $t = 15$, τ_1 issue a request to change its period to 250, and all the other periods

Task	C_i	T_{i_0}	$T_{i_{max}}$	E_i	B_i
τ_1	23	100	500	1	1
τ_2	23	100	500	1	1
τ_3	23	100	500	3	1
τ_4	23	100	500	5	1

Table 1. Task set parameters used for the second experiment. Times are expressed in milliseconds.

can gracefully go to their nominal values. In this experiment, the transient periods $T_i(k)$ for τ_1 were generated using a linear law.

The result of this test is illustrated in Figure 5, which shows how task periods evolve during the transition. It is worth observing that, although T_1 is modified using a linear transition law, the other periods vary according to a non-linear function. This happens because, when T_1 is decreased, the total processor utilization increases, so the elastic manager performs a compression of the other tasks (enlarging their periods) to keep the total load constant. Given the non linear relation between total utilization and periods ($U = C_1/T_1 + \ldots + C_n/T_n$), the other periods change in a non-linear fashion. Figure 6 shows the period evolution when an exponential law is used for τ_1 to modify its period.

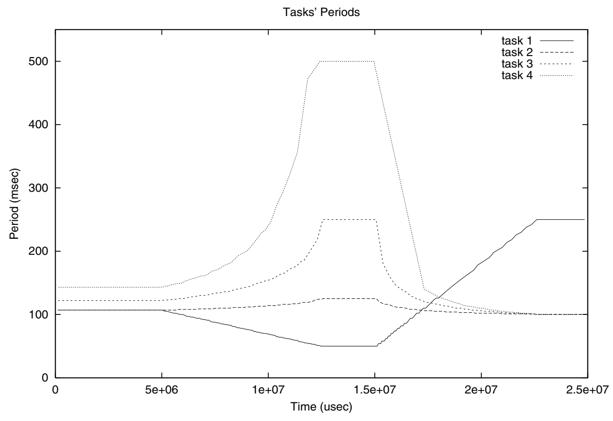

Figure 5. Periods evolution as a function of time when τ_1 changes its period with a linear law.

A different experiment has been performed using the task set shown in Table 2 to test the behavior of the DM in the presence of dynamic task activations. In this case, when a new task τ_h needs to enter the system with period T_h^* and there exists a feasible elastic schedule for it (see [8] for details on the elastic guarantee), it cannot be immediately activated with that period. In fact, the other tasks have to gradually reduce their utilizations (according to the damping law) to decrease the load and create space for the new task. As a consequence, the new task is activated with a large period (theoretically infinite, practically equal to MAXINT), which is gradually reduced to the final T_h^* value by applying the damping law. We note that the time required to the transition is always bound to NT_{DM}, where N is the number of steps fixed for the transition and T_{DM} is the period of the Damping Manager.

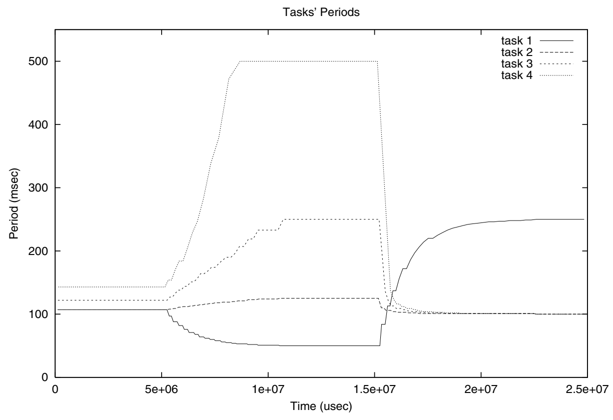

Figure 6. Periods evolution as a function of time when τ_1 changes its period with an exponential law.

In the specific experiment we performed, task τ_2 was added at time $t = 5sec$, and the DM started to decrease its period towards the final value $T_{2_0} = 100ms$. Figure 7 shows the result of this experiment when a linear transition law was used. It is worth noticing that, as a consequence of τ_2 arrival, all the task periods begin to gracefully expand to create space for τ_2, thus τ_2 effectively begins to execute only when $T_2(k) < T_2^{max}$. From Figure 7 it is also interesting to observe that, as T_2 is decreased linearly, the other periods increase exponentially, based on their elastic coefficients, for the same reason explained in the previous experiment.

Figure 8 shows the result achieved on the same task set using an exponential transient law. In this case, the activation delay experienced by τ_2 is smaller with respect to the linear case. Moreover, the other periods reach their final values with a much smoother transition.

Task	C_i	T_{i_0}	$T_{i_{max}}$	E_i	B_i
τ_1	40	100	500	1	1
τ_2	40	100	500	1	1
τ_3	40	100	500	1.5	1
τ_4	40	100	500	2	1

Table 2. Task set parameters used for the third experiment. Times are expressed in milliseconds.

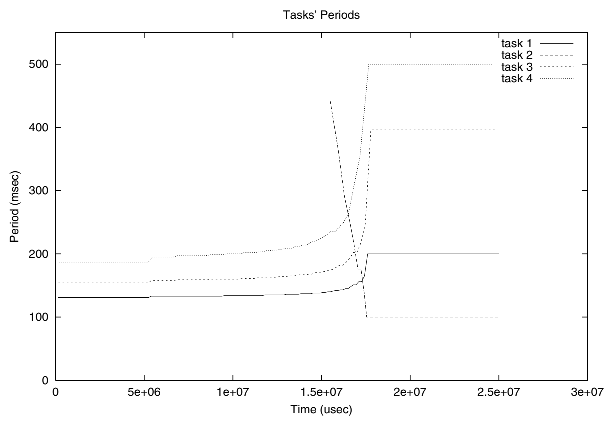

Figure 7. Periods evolution as a function of time when task τ_2 is dynamically activated and its period is changed using a linear transition law.

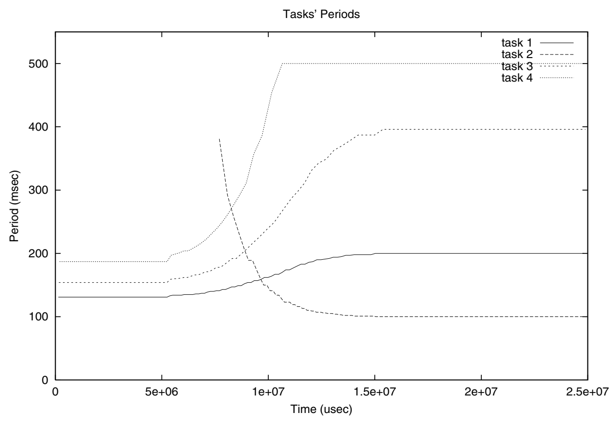

Figure 8. Periods as a function of time when T_2 is changed using an exponential transition law.

6 Conclusions

In this paper we presented a method to gracefully perform rate adaptation during transient overload conditions. Tasks are treated as elastic springs whose utilizations can be adjusted (through proper period variations) to create a desired workload. Smooth rate change is achieved by adding a damping device to each spring. The proposed approach has been implemented on the HARTIK kernel [5, 11], where some experiments have been performed to test the effectiveness of damping for different period transition laws.

From the experiments performed on the kernel, we observed that when an active task wants to change its period (either lower or higher than the current one), a linear transition law is able to achieve smoother periods variation on the other tasks. On the other hand, when a new task needs to be activated in the system, an exponential law (for reducing its period to the required final value) is able to vary the other periods more gracefully and it also allows to reduce the activation delay of the newly arrived task.

As a future work, we plan to implement this methodology on top of different kernels (e.g., Real-Time Linux or Linux-RK) and develop a multimedia application to verify the influence of the elastic and damping coefficients on the quality perceived by the users.

References

[1] T. F. Abdelzaher, E. M. Atkins, and K. G. Shin, "QoS Negotiation in Real-Time Systems and Its Applications to Automated Flight Control," *Proc. of the IEEE Real-Time Technology and Applications Symposium*, June 1997.

[2] L. Abeni and G. Buttazzo, "Support for Dynamic QoS in the HARTIK Kernel," *Proc. of the 7th IEEE Real-Time Computing Systems and Applications*, December 2000.

[3] R. J. Anderson and M. W. Spong, "Hybrid Impedance Control of Robotic Manipulators," *IEEE Journal of Robotics and Automation*, Vol. 4, No. 5, October 1988.

[4] G. Beccari, S. Caselli, M. Reggiani, F. Zanichelli, "Rate Modulation of Soft Real-Time Tasks in Autonomous Robot Control Systems," *IEEE Proc. of the 11th Euromicro Conference on Real-Time Systems*, June 1999.

[5] G. Buttazzo, "HARTIK: A Real-Time Kernel for Robotics Applications", *Proc. of the 14th IEEE Real-Time Systems Symposium*, Raleigh-Durham, pp. 201–205, December 1993.

[6] G. Buttazzo, G. Lipari, and L. Abeni, "Elastic Task Model for Adaptive Rate Control", *Proc. of the IEEE Real-Time Systems Symposium*, December 1998.

[7] G. Buttazzo and L. Abeni, "Adaptive Rate Control through Elastic Scheduling," *Proc. of the 39th IEEE Conference on Decision and Control*, December 2000.

[8] G. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni, "Elastic Scheduling for Flexible Workload Management", *IEEE Transactions on Computers*, Vol. 51, No. 3, March 2002.

[9] H. Fujita, T. Nakajima, and H. Tezuka, "A Processor Reservation System supporting Dynamic QOS control," *Proc. of the 2nd International Workshop on Real-Time Computing Systems and Applications*, October 1995.

[10] T.-W. Kuo and A. K, Mok, "Load Adjustment in Adaptive Real-Time Systems," *Proc. of the 12th IEEE Real-Time Systems Symposium*, December 1991.

[11] G. Lamastra, G. Lipari, G. Buttazzo, A. Casile, and F. Conticelli, "HARTIK 3.0: A Portable System for Developing Real-Time Applications," *Proc. of the IEEE Real-Time Computing Systems and Applications*, October 1997.

[12] C. Lee, R. Rajkumar, and C. Mercer, "Experiences with Processor Reservation and Dynamic QOS in Real-Time Mach," *Proc. of Multimedia Japan 96*, April 1996.

[13] G. Lipari, G. Buttazzo, and L. Abeni, "A Bandwidth Reservation Algorithm for Multi-Application Systems", *Proc. of IEEE Real Time Computing Systems and Applications*, October 1998.

[14] C.L. Liu and J.W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard real-Time Environment," *Journal of the ACM* 20(1), 1973, pp. 40–61.

[15] C. Lu, J. Stankovic, T. Abdelzaher, G. Tao, S. Son, and M. Marley,"Performance Specifications and Metrics for Adaptive Real-Time Systems," *Proceedings of the IEEE Real-Time Systems Symposium*, December 2000.

[16] T. Nakajima and H. Tezuka, "A Continuous Media Application supporting Dynamic QOS Control on Real-Time Mach," *Proceedings of the ACM Multimedia '94*, 1994.

[17] T. Nakajima, "Resource Reservation for Adaptive QOS Mapping in Real-Time Mach," *Sixth International Workshop on Parallel and Distributed Real-Time Systems*, April 1998.

[18] D. Seto, J.P. Lehoczky, L. Sha, and K.G. Shin, "On Task Schedulability in Real-Time Control Systems," *Proc. of the IEEE Real-Time Systems Symposium*, December 1997.

[19] J. A. Stankovic, C. Lu, and S. H. Son, "The Case for Feedback Control in Real-Time Scheduling", *IEEE Proc. of the Euromicro Conference on Real-Time Systems*, June 1998.

[20] C. Lu, J. A. Stankovic, G. Tao, and S.H. Son, "Design and Evaluation of a Feedback Control EDF Scheduling Algorithm", *Proc. of the 20th IEEE Real-Time Systems Symposium*, December 1999.