

Online Control Optimization Using Load Driven Scheduling

Lui Sha, Xue Liu
lrs, xueliu@cs.uiuc.edu
University of Illinois
Dept. of CS
Urbana, IL61801

Marco Caccamo
caccamo@sssup.it
Scuola Superiore S. Anna
Pisa (Italy)

Giorgio Buttazzo
giorgio@sssup.it
University of Pavia (Italy)
INFM research unit

Abstract

In many real-time control applications, the task periods are typically fixed and worst-case execution times are used in schedulability analysis. With the advancement of robotics, flexible visual sensing using cameras becomes a popular alternative to the use of embedded sensors. Unfortunately, the execution time of visual tracking varies greatly. In this paper, we integrate load driven online scheduling with direct digital designs to optimize control performance as a function of varying workload.

1 Introduction

In many real-time control applications, the task periods are typically fixed and worst-case execution times are used in schedulability analysis. This is a fine model for many classical control applications. However, with the advance of robotics, flexible visual sensing using cameras has become popular, and the execution time of visual tracking may vary greatly.

For example, consider a visual tracking laboratory device using a ball and plate control. A ball on the plate has to follow a specified circle on the plate, which can be controlled by acting on roll and pitch rotations. To speed up the visual tracking process, predictive techniques are typically used to search for the ball in a small mobile window centered in the estimated ball position rather than searching the whole image. Normally, the ball is found in the small window and its position can be computed quickly. However, if occasional disturbances makes the ball move outside of the predicted window, searching has to be extended in a larger area or even the entire plate, this may cost more computation time. In this example, the visual tracking task's computation time can be modeled as having a constant normal execution time with a bounded worst-case computation time.

It is worth noting that the search time is a part of the control loop, since the position information is needed in control computations. With a normally short (control

loop) computation time but an occasional long computation time, the use of worst-case computation times is inefficient. Nevertheless, to guarantee the control stability, we still need to close the control loop in time, even if the worst case arises.

If we reserve for the worst-case time, then most of the time there is a large amount of reserved but unused execution time budget. Although such an unused reserved time can be reclaimed for soft real-time aperiodic applications [7, 8, 9], such an aperiodic application may or may not exist in a given application environment. An alternative is to fully utilize the reserved budget to optimize the control performance in spite of the variation in computation time, while still guarantee the stability.

2 Related Work

How to handle large variations in execution times is an active research area. In [3], Gardner and Liu proposed a method that ensure tasks' deadlines that uses only normal execution times. Stankovic et al. in [10, 11] used feedback to minimize the percentage of deadline missing in multi-media applications.

The problem of selecting the set of control task frequencies to optimize the system control performance subject to schedulability constraint was addressed by Seto, Lehoczky, Sha and Shin[6]. In this formulation, each control task τ_i is characterized by a *Performance Loss Index* (PLI¹) as a function of the sampling frequency. The PLI can be approximated (use curve fitting) as the general form of $\Delta J_i(f_i) = \alpha_i e^{-\beta_i f_i}$. Here f_i is the frequency of the control task, α_i is a magnitude coefficient and β_i is the decay rate. Also, for each task, the lower bound on the sampling frequency is denoted by f_i^{min} . The PLI of the overall system which contains n tasks is defined as follows:

$$\Delta J(f_1, \dots, f_n) = \sum_{i=1}^n w_i \Delta J_i(f_i), \quad (1)$$

¹In the original formulation, the performance loss index was simply called performance index or PI. In the following, it will be called PLI for more clarity.

where w_i is a design parameter determined from the application. For instance, it can be the importance of the associated control system to have a better performance than the others.

Given the minimum frequency f_i^{min} and the worst-case execution time ($WCET_i$) of each task τ_i , the solution for f_i^{opt} , which minimizes the PLI of the system while guaranteeing the schedulability constraint, was developed in [6]. However, f_i^{opt} were computed based on $WCET_i$ s. If the normal computation time c_i^n are much less than $WCET_i$, then f_i^{opt} can be too low. To overcome this problem, Caccamo, Buttazzo and Sha proposed a new elastic control approach[13]. This approach integrates the continuous design with digitization (CDD) with adjustable frequencies. However, CDD does not optimize the control gains as a function of the selected control frequencies. In this paper, we propose the elastic control scheduling method using direct digital design (DDD), and ensure the stability of the resulting hybrid controllers.

2.1 Elastic control scheduling

In this subsection we review the elastic control model [13] developed for improving the performance of a real-time control system with large variations in computation time. We assume that the task set is scheduled by the Earliest Deadline First (EDF) algorithm.

For each task, there are worst-case computation and a normal computation time $c_i^n \leq WCET_i$ computed by analyzing the probabilistic distribution of the task computation time. Hence, each task is described by $\tau_i(c_i^n, WCET_i, f_i^{min}, \Delta J_i(f_i))$, where $WCET_i$ is the worst-case execution time, f_i^{min} is the minimum frequency τ_i can execute at, and $\Delta J_i(f_i)$ is the PLI(get either from DDD or CDD) of τ_i . Each task must complete at or before its hard deadline $D_i^{hd} = 1/f_i^{min}$. Notice, however, that task τ_i will be normally scheduled using a dynamic deadline D_i less than or equal to D_i^{hd} .

The goal of this approach is to compute the optimal frequency f_i^{opt} , using the normal computation time c_i^n instead of using the $WCET_i$, while guaranteeing that each task will never miss its hard deadline D_i^{hd} .

The algorithm we developed in [13] gives each task a computation budget as a percentage of the CPU. This budget will guarantee that every task will always meet its deadline. In addition, given the budgets, task frequencies can be adjusted in such a way that the control performance is optimal. The computation budgets can be implemented as a server that maintains the assigned budgets for each task.

In this model, each task can change its frequency dynamically depending on the current load. If $\tilde{d}_{i,j}$ is the normal deadline used by the server to schedule the job

$J_{i,j}$ and its overrun (whenever it occurs), the next job $J_{i,j+1}$ will start at time:

$$a_{i,j+1} = \max(a_{i,j} + \frac{1}{f_i^{opt}}, \tilde{d}_{i,j}). \quad (2)$$

Hence, each task instance (job $J_{i,j}$) has a variable period $T_{i,j} = a_{i,j+1} - a_{i,j}$. Using this formalism and assuming that each instance has to complete by its fictitious deadline $\tilde{d}_{i,j}$, we need to guarantee that

$$\forall i, j \quad T_{i,j} \leq \frac{1}{f_i^{min}}. \quad (3)$$

We now introduce a overrun management policy that allows each task to handle its overruns locally without affecting the frequencies of the other tasks. In the following we assume that the optimal frequency f_i^{opt} has already been computed for each task τ_i , based on c_i^n , using the algorithm proposed in [6]. The method is based on reserving a bandwidth $U_i = f_i^{opt} c_i^n$ for each task τ_i . In normal conditions (without overruns), each task has a constant period $T_i = 1/f_i^{opt}$ and each task behaves as a classic periodic task². When a job $J_{i,j}$ generates an overrun, a new deadline is computed for $J_{i,j}$ to avoid that the bandwidth consumed by τ_i exceeds its reserved bandwidth U_i .

Let $\tilde{d}_{i,j}$ be the normal deadline assigned by the server before the overrun occurs. That is,

$$\tilde{d}_{i,j} = a_{i,j} + \frac{1}{f_i^{opt}}.$$

If $J_{i,j}$ tries to execute more than c_i^n , the new deadline $\tilde{d}'_{i,j}$ of $J_{i,j}$ becomes:

$$\tilde{d}'_{i,j} = \tilde{d}_{i,j} + \frac{WCET_i - c_i^n}{U_i}. \quad (4)$$

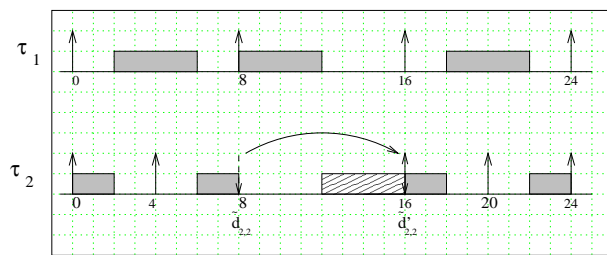


Figure 1: Example of overrun handled locally.

²Periodic tasks consist of an infinite sequence of identical activities, called instances or jobs, that are regularly activated at a constant rate.

The following example illustrates how an overrun is handled by the scheduling algorithm. The task set consists of two periodic tasks, τ_1 and τ_2 , with minimum frequencies $1/20$ and $1/12$, worst case execution times 5 and 6, normal execution times 4 and 2, respectively. Moreover, let us suppose that the optimal frequencies computed by the Seto et al. algorithm are $f_1^{opt} = 1/8$ and $f_2^{opt} = 1/4$. Therefore, each server has assigned a bandwidth $U_1 = U_2 = 0.5$.

Figure 1 shows what happens when τ_2 generates an overrun at time $t = 8$. Initially, the server assigns a normal deadline $\tilde{d}_{2,2} = 8$ to job $J_{2,2}$. At time $t = 8$, an overrun occurs and the server computes the new deadline $\tilde{d}'_{2,2} = \tilde{d}_{2,2} + (WCET_2 - c_2^n)/U_2 = 16$. After handling the overrun, the next job $J_{2,3}$ of task τ_2 arrives at time $a_{2,3} = 16$ and it will be executed again at its optimal frequency.

A simple necessary and sufficient condition, Theorem 1, states how it is possible to handle overruns locally.

Theorem 1 *Given a set Γ of periodic tasks $\tau_i(c_i^n, WCET_i, f_i^{min})$ where each task is handled by a dedicated server with bandwidth U_i , then each task instance (job $J_{i,j}$) has a period $T_{i,j} \leq 1/f_i^{min}$ if and only if:*

$$\forall \tau_i \quad U_i \geq f_i^{min} WCET_i. \quad (5)$$

Proof: See [13]. ■

Using the result of Theorem 1, we can compute the optimal frequencies f_i^{opt} and guarantee a minimum frequency f_i^{min} for each task τ_i even in the presence of overruns. Similar to the Seto et al. algorithm, in this approach a task set is guaranteed if and only if:

$$\sum_i f_i^{min} WCET_i \leq 1. \quad (6)$$

If the task set is feasible, a new lower-bound \tilde{f}_i^{min} of frequency can be computed for each task τ_i in order to take overruns into account. In fact the parameter \tilde{f}_i^{min} is defined as follows:

$$\forall \tau_i \quad \tilde{f}_i^{min} = \frac{f_i^{min} WCET_i}{c_i^n}. \quad (7)$$

Finally, the Seto et al. algorithm will be used to compute the optimal frequencies using c_i^n as computation time (instead of $WCET_i$) and \tilde{f}_i^{min} as minimum frequency (instead of f_i^{min}) for each task τ_i .

Note that, when overruns are handled locally and tasks have their normal computation times, this technique

can have a performance penalty; however, requiring that local handling is needed, the Seto et al. algorithm maintains its optimality even if the value of \tilde{f}_i^{min} is used as minimum frequency instead of f_i^{min} .

This is easy to prove, because the value \tilde{f}_i^{min} represents the minimum frequency permitted to each task τ_i in order to handle overruns locally. Hence, in this model, each optimal frequency f_i^{opt} must be greater than or equal to \tilde{f}_i^{min} .

3 Improving Performance Using Direct Digital Design

In [13] continuous design with digitization (CDD) approach was used for elastic control. Using the direct digital design (DDD), we can get smaller PLI, i.e., better control performance. Furthermore, DDD can typically maintain system stability at low frequencies which CDD can not. This implies the DDD approach can adapt to a wider work load fluctuation.

We illustrate the effect of the proposed DDD approach on a double Inverted Pendulum (IP) system, which is a simplified model for the study and testing of Simplex Architecture. Simplex Architecture[12] is a software architecture which can accommodate the on-line upgrade of control softwares. A web based version of Simplex that allows users to reliably change the control software of an IP without shutting it down and to observe the new control software's effect can be downloaded from www-drii.cs.uiuc.edu. In this paper, we use a modified version of the double Inverted Pendulum to illustrate the achievable performance improvements.

This example system consists of two inverted pendulums with different physical parameters, each is erected on a cart which was laid on the track. They are controlled by one on-board processor. In this example, a camera monitors the IP as sensor for getting its cart and angle position. Hence, each control task is characterized by two different functions: the first one reads the image memory filled by the camera frame grabber and determines the actual position of the IP, and the second one computes the next value of the control input. The first function of each control task is characterized by a variable computation time which depends on the current position of each IP. So, we can assume that each task is characterized by a worst-case execution time ($WCET_i$) and a normal computation time c_i^n .

The two IP models are shown in Figure 2 and Figure 3, respectively.

The task set parameters are shown in Table 1, where, for each $IP_i (i = 1, 2)$, $WCET_i$ (ms) is the control

$$\dot{x} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -2.7528 & -10.9526 & 0.0043 \\ 0 & 28.5812 & 24.9179 & -0.0441 \end{pmatrix} x + \begin{pmatrix} 0 \\ 0 \\ 1.9432 \\ -4.4385 \end{pmatrix} u$$

$$x(0) = [0.2, 0.1745, 0, 0]^T$$

$$Q = \text{diag}([100.25, 0.01, 0, 0]), \quad R = 3$$

Figure 2: IP1 model.

$$\dot{x} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -2.40 & -9.10 & 0 \\ 0 & 40.10 & 29.80 & 0 \end{pmatrix} x + \begin{pmatrix} 0 \\ 0 \\ 2.03 \\ -6.66 \end{pmatrix} u$$

$$x(0) = [0.3, 0.2618, 0, 0]^T$$

$$Q = \text{diag}([5.25, 4, 0, 0]), \quad R = 0.3$$

Figure 3: IP2 model.

task's worst-case execution time, f_i^{min} (Hz) is the lower bound on sampling frequency, and w_i is the weight assigned to system i . $\alpha_i^{CDD}, \beta_i^{CDD}$ are the fitted exponential decay function parameter of the CDD (ΔJ_{CDD}); $\alpha_i^{DDD}, \beta_i^{DDD}$ are the fitted exponential decay function parameter of the DDD (ΔJ_{DDD}).

Figure 4 shows the result of fitting the two exponential functions in Table 1 to the two real performance loss index. Here ΔJ_{CDD}^a and ΔJ_{DDD}^a are the corresponding exponential decay function approximation of ΔJ_{CDD} and ΔJ_{DDD} .

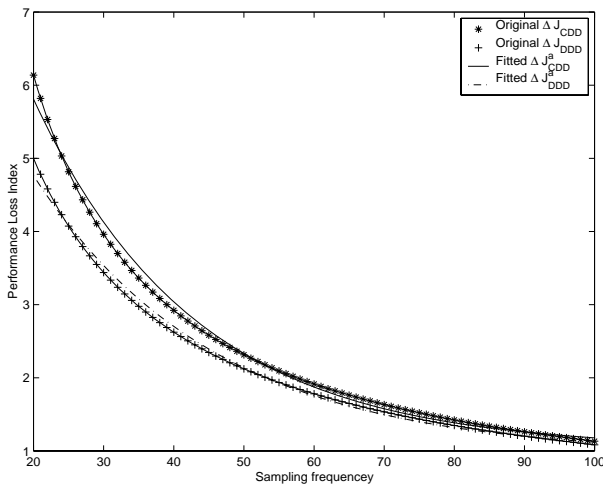


Figure 4: Exponential function approximation of the performance loss index.

A simple computation shows that the total CPU utilization of the overall system is 77.5% when the minimum

task frequencies are assigned. Now, we are supposing the total CPU utilization available for the double IP control system is 100%. Table 2 shows, at different values of c^n , the optimal frequencies computed using the Seto et al algorithm [6] and the resulting performance loss index of the overall system. Note that system performance increases as the performance loss index decreases. Moreover, $c^n = kWCET$ means that all normal computation times are reduced by that fraction. So, for instance, $c^n = 0.9WCET$ means that $c_1^n = 0.9WCET_1$ and $c_2^n = 0.9WCET_2$.

The results are reported in Table 2, it demonstrates: First, in both situations of CDD elastic control approach and DDD elastic control approach, the control system's performance significantly improves as the normal computation time is decreased with respect to the $WCET$, because tasks can run at higher frequencies. Please refer to Figure 5 for the relation between the performance loss index and the value of c^n . In the graph, the normal computation time on the x-axis is expressed as a fraction of $WCET$. For instance, a value of 0.9 means that $c_1^n = 0.9WCET_1$ and $c_2^n = 0.9WCET_2$. Note that a little difference between c^n and $WCET$ gives a significant gain in performance. This means that the elastic control approach can give better control performance.

Second, the performance of the DDD elastic control approach is better than the performance of CDD elastic control approach. This means if we can use DDD for the control system and at the same time, using elastic feedback control, we can get improved performance. The comparison of the optimal ΔJ_{CDD}^a and ΔJ_{DDD}^a is shown in Figure 5.

Task	$WCET_i$ (ms)	f_i^{min} (Hz)	w_i	α_i^{CDD}	β_i^{CDD}	α_i^{DDD}	β_i^{DDD}
IP_1	20	20	2	22.8072	0.0737	16.4821	0.0706
IP_2	15	25	1	3.7848	0.0723	1.4284	0.0269

Table 1: Task parameters for the double IP system.

c^n	$f_1^{opt}_{CDD}$ (Hz)	$f_2^{opt}_{CDD}$ (Hz)	ΔJ_{CDD}^a	$f_1^{opt}_{DDD}$ (Hz)	$f_2^{opt}_{DDD}$ (Hz)	ΔJ_{DDD}^a
$WCET$	31.25	25.00	5.18	31.25	25.00	4.36
$0.9WCET$	34.72	27.78	4.04	34.72	27.78	3.52
$0.8WCET$	39.06	31.25	2.96	39.06	31.25	2.71
$0.7WCET$	44.64	35.71	1.99	44.64	35.71	1.96

Table 2: Optimal frequencies and corresponding ΔJ for different values of c^n .

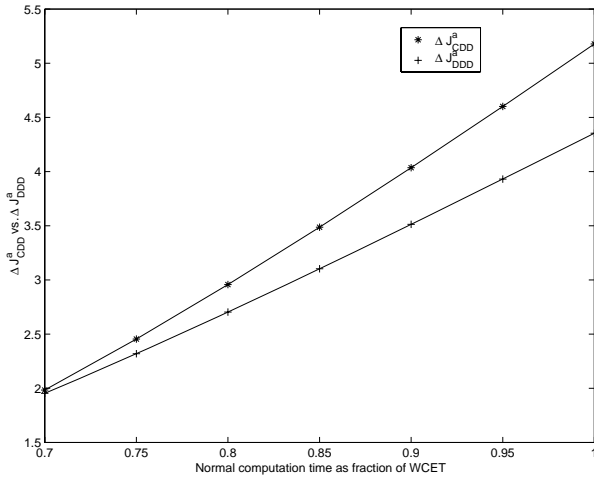


Figure 5: Comparison of ΔJ_{CDD}^a and ΔJ_{DDD}^a using Elastic Feedback Control.

From the system performance perspective, Figure 5 indicates the average performance difference decreases as the normal computation time becomes a smaller fraction of the WCET. This is because the overruns are infrequent, hence most of the time the system runs with a high frequency. Finally, we note that the difference between performance of DDD and CDD are small for high frequencies.

On the other hand, from system stability perspective, it is important to point out that DDD can maintain the stability at lower frequencies. We calculate the minimum frequency that can maintain system stability for the corresponding discretized system using eigenvalue criteria for IP1 in the above example. The result shows for CDD, the minimum frequency is 5 Hz, and for DDD, it is 0.3 Hz, which is much lower than that of CDD.

Readers who are interested in the CBS server mechanism which allows us to efficiently implement the overrun handling algorithm can referred to [13].

4 System stability under controller switching

In this paper, we introduced the DDD elastic control approach. This approach in fact is a hybrid control since we **switch** between different control gains as a function of the workload. It is well known that under some pathological conditions, the hybrid control system can become unstable even though each individual controller itself is stable. Thus, for any system, before applying the DDD elastic control approach, we must test whether the system can maintain stability under control gain switching. Here, we use the results in [14].

Consider a time varying discrete time system

$$x(k+1) = A(k)x(k), \quad k = 0, 1, 2, \dots \quad (8)$$

where $A(k) \in \mathcal{A} = \{A_1, A_2, \dots, A_N\}$, $A_i \in \mathcal{R}^{n \times n}$. Assume that in the discrete time system above, state matrices are chosen from a certain set, \mathcal{A} . The stability definition and result are given below.

Definition 1 A set of matrices, \mathcal{A} , is called asymptotically stable if discrete time system (8) is asymptotically stable for all possible sequences of matrices in \mathcal{A} .

Corollary 1 \mathcal{A} is asymptotically stable iff there exist a positive definite matrix $P \in \mathcal{R}^{n \times n}$ such that

$$A^T P A - P < 0, \quad \forall A \in \mathcal{A}^k, \quad \text{for some } k \geq 1 \quad (9)$$

where the inequality is in the sense of negative definiteness.

We now show how to determine stability under switching for our example of double IP by using the corollary above. Suppose the normal computation time $c^n = 0.7WCET$ for both inverted pendulum. For IP1, from Table 2, we know it normally runs at optimal frequency of 44.64 (Hz), which corresponds to a DDD control gain of $K_{normal} = -[5.1228, 39.5843, 13.1831, 8.0636]$; when

overflow occurs, it will switch to frequency of 31.25 (Hz), which corresponds to a DDD control gain of $K_{wcet} = -[4.8643, 38.5505, 12.8239, 7.8517]$. Let A_{normal} and A_{wcet} denote the discrete system matrix under these two conditions respectively, so $\mathcal{A} = \{A_{normal}, A_{wcet}\}$. Using the LMI control toolbox of Matlab, we can get

$$\text{when } k = 1, P = \begin{pmatrix} 149.59 & 139.47 & 52.20 & 27.00 \\ 139.47 & 394.10 & 115.80 & 67.57 \\ 52.20 & 115.80 & 97.29 & 51.56 \\ 27.00 & 67.57 & 51.56 & 30.36 \end{pmatrix}$$

will satisfy equation 9. So we know IP1 will guarantee to be stable even under the control gain switching. We can get the same result for IP2.

5 Conclusions

Traditionally, the task periods are typically fixed and worst case execution times are used in schedulability analysis. This is a fine model for many classical control application where the execution time variations are small.

However, with the advancement of robotics, flexible visual sensing using cameras becomes popular, and the execution time of visual tracking may vary greatly.

The DDD elastic control approach we proposed integrates direct digital design, the optimization of control performance subject to schedulability analysis and the CBS server algorithms – to create an innovative approach to optimize control performance online.

6 Acknowledgement

The authors would like to thank ONR and EPRI for their support. And also wish to thank Anton Cervin for his suggestion on the stability analysis.

References

[1] L. Abeni and G. Buttazzo, “Integrating Multimedia Applications in Hard Real-Time Systems”, *Proc. of the IEEE Real-Time Systems Symposium*, Madrid, Spain, December 1998.

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

[3] M. K. Gardner and J. W.S. Liu, “Performance of algorithms for scheduling real-time systems with overrun and overload”, *IEEE Proceedings of the 11th Euromicro Conference on Real-Time Systems*, York, UK, June 1999.

[4] J. Leung and J. Whitehead, “On the complexity of fixed-priority scheduling of periodic, real-time tasks”, *Performance Evaluation*, 2:237-250, 1982.

[5] 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.

[6] D. Seto, J.P. Lehoczky, L. Sha and K.G. Shin “On Task Schedulability in Real-Time Control System”, *Proc. of the IEEE Real-Time Systems Symposium*, December 1996.

[7] M. Spuri and G.C. Buttazzo, “Efficient Aperiodic Service under Earliest Deadline Scheduling”, *Proc. of the IEEE Real-Time Systems Symposium*, San Juan, Portorico, December 1994.

[8] M. Spuri, G. Buttazzo and F. Sensini, “Robust Aperiodic Scheduling Under Dynamic Priority Systems”, *Proc. of the 16th IEEE Real-Time Systems Symposium*, Pisa, Italy, December 1995.

[9] M. Spuri and G.C. Buttazzo, “Scheduling Aperiodic Tasks in Dynamic Priority Systems”, *The Journal of Real-Time Systems*, 10(2), 1996.

[10] J. A. Stankovic, C. Lu, S. Son and G. Tao, “The Case for Feedback Control Real-Time Scheduling”, *IEEE Proceedings of the 11th Euromicro Conference on Real-Time Systems*, York, UK, June 1999.

[11] C. Lu, J. A. Stankovic, G. Tao and S. H. Son, “Design and Evaluation of a Feedback Control EDF Scheduling Algorithm”, *Proceedings of the IEEE Real-Time Systems Symposium*, Phoenix, Arizona, December 1999.

[12] Lui Sha, “Dependable System Upgrade”, *IEEE Real-Time System Symposium*, Madrid, Spain, December 1998.

[13] Marco Caccamo, Giorgio Buttazzo, Lui Sha, “Elastic Feedback Control”, to appear in *Euromicro 2000, 12th International Conference on Realtime Systems*, Stockholm, Sweden, June 2000.

[14] Murat Dogruel, Umit Ozguner, “Stability of a Set of Matrices: A Control Theoretic Approach”, *Proceedings of the 34th Conference on Decision and Control*, New Orleans, LA, December 1995.