On Adaptive Control Techniques in Real-Time Resource Allocation

Luca Abeni and Luigi Palopoli
Scuola Superiore S. Anna, Pisa
luca@sssup.it, palopoli@sssup.it

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

Abstract

A remarkable class of soft-real time applications exhibits a very dynamical behaviour due to the variations in the treated data. Moreover, such programs have to be able to run on hundreds of different platforms. As a consequence classical real-time scheduling algorithms are not flexible enough since they are based on the exact knowledge of the tasks' timing parameters. Some of the approaches proposed so far in the literature guarantee temporal isolation, but they make a static assignment of resources to each task, which, once again, is based on an a priori *knowledge. In this paper we propose a closed loop method for on-line adapting the fraction of assigned resource to the task requirements. The approach is based on adaptive control techniques and has resulted to be effective in a significant set of real-life experiments.*

1. Introduction

Real-time techniques, originally aimed at embedded control, are being extended to address a new kind of applications. In particular, multimedia is becoming an appealing field. This evolution introduces new classes of problems. First, this kind of programs typically run on hundreds of different machines. Secondly, they have to coexist together with many other programs which can cause sharp variations in the system's workload. Finally, the execution time of each task activation (from now on, *job*) strongly depends on the particular data being processed and not only on the program.

Classical real-time approaches [15, 6, 5] are essentially based on the exact knowledge of tasks' timing parameters (execution time and interarrival time), thus they are not adequate to multimedia applications. More recent systems tend to implement some form of temporal protection aimed at limiting the damages caused by misbehaved tasks. Such techniques are based on resource reservations [18] or some form of proportional share scheduling [22, 10].

According to the Resource Reservation (RR) approach, a system is able to enforce that each task τ_i will use at most an amount C_i of a resource in each time interval T_i: in this way a task will never compromise the other tasks' schedulability by requiring more than the reserved amount of a given resource.

Proportional Share (PS) scheduling permits to fairly share a resource between a set of tasks based on some weighting factors associated to each of them: each task τ_i receives a fraction $\frac{w_i}{\sum_j w_j}$ of the total amount of the resource, where w_i is the task weight.

PS and RR are two different ways to address the same problem of temporal protection, as shown in [3]. These approaches permit to isolate each task's temporal behaviour, by describing it through a weight w_i, and a couple $(C_i,\ T_i)$, respectively. However, at least an approximate estimate of every single task's demand on the available resources is needed in order to provide the required Quality of Service (QoS). Using RR or PS, the a priori knowledge is used to assign the scheduling parameters ((C_i, T_i) or w_i) [19, 21]. In a dynamic system, where no a priori knowledge is available, these parameters have to be adapted at run-time in order to keep the QoS stable. This problem can be solved using a feedback mechanism to dynamically adapt the reserved resources to the task requirements.

A dynamic QoS adaptation is tested in [17, 8], while feedback control schemes are widely used in multimedia applications to scale resource utilization (for example, see [14]). In [7] a mechanism for changing the tasks' utilization factor is presented. All these works present some form of dynamic QoS adaptation, but the utilization is scaled only by changing the tasks' rates/periods.

In [11] the authors describe a scheduler providing resource reservations and a method to dynamically adjust the reserved time. However, it is not clear how the system behaves in overload situations (when all the reservations require to be increased). This problem is common to other approaches concerning rate adaptation, such as [23].

Most of the methods cited so far propose a simple proportional feedback mechanism, showing experimental evidence of its effectiveness; the papers do not provide any theoretical consideration on the correctness of the adaptation mechanism.

In [14] the authors present a general framework to control the application requests for system resources using the amount of allocated resources as a feedback. In particular, they show that a Proportional Integrative Derivative (PID) controller (well known in control theory and practice) can be used to bound tasks' resource usage in a stable and fair way. Recently, in [20], a feedback control mechanism based on a PID controller 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. In these two works the control mechanism has been analyzed to prove its correctness and stability, but, in order to apply the classical control theory, only a simplified (and approximated) system model has been used. Moreover, in [20], the feedback mechanism is used to reject jobs in order to keep the total number of missed deadlines below a desired value, but the method cannot be used to control the deadline miss ratio of individual tasks.

In [2], the problem of managing the CPU resource has been addressed by defining some task models suitable for multimedia applications. In particular, two of them (the CM task model and the ED task model) use a feedback mechanism and a PI controller for adjusting the reserved fraction of CPU bandwidth. Although the approach proposed in [2] is promising, several issues are not treated in a formal way. For example, the controller parameters are experimentally tuned and a general proof of the system stability is missing.

In this paper, we decided to treat the system as "a black-box" trying to estimate an approximate model for tuning the controller's parameters while the system is being controlled. This technique has long been investigated in the control system community and is known as *adaptive control*. With this solution, the final user is not required to provide any controller parameter, since the controller gets automatically tuned on the system.

In order to study the applicability of adaptive control to real-time resource management, the framework presented in our previous work has been extended. An adaptive and extended form of the PID controller has proven to be an effective solution.

We remark that the system can surprisingly be controlled even with a little knowledge on its physical model. Moreover, adaptive control techniques for this class of problems are relatively easy to design and implement and have produced encouraging results.

The paper is organized as follows: in Section 2 we describe the system model and introduce the problem; in Section 3 we briefly describe the adaptive control techniques used in the rest of the paper; in Section 4 we present the implementation of the QoS manager used for our experiments and show the experimental results, while the conclusions are stated in Section 5.

2. Problem Description

As an example of the problems presented in the previous section, consider an MPEG player. Since frames are processed by a periodic activity, a deadline can naturally be associated with each job. However, the quality of the presentation is not significantly affected if the data of each frame are made available in a sufficiently small neighborhood of the deadline. The amplitude of this neighborhood can be used as a possible metrics for controlling the QoS associated with the data stream. Starting from an initial situation where the deviation from the deadline is large, our goal is to change the amount of resources assigned to the task, acting on the scheduler parameters, until the deviation from the deadline is acceptable (ideally 0).

Any scheduling algorithm endowed with temporal protection can be used to implement this idea. For this work, we chose to serve each task by a dedicated Constant Bandwidth Server (CBS) [1], scheduled by a global Earliest Deadline First (EDF) scheduler [15]. The CBS provides an intuitive output variable to be used as a feedback: the *CBS Scheduling Error*, which will be defined below.

In the remainder of this section, we will briefly describe the CBS mechanism. The CBS is a service mechanism first presented in [1], where it was used to implement a *bandwidth reservation strategy*. Readers interested to a complete description of the algorithm and of its properties are referred to the cited paper; in this context we recall a few basic concepts which can be useful for a better understanding of our feedback mechanism.

A CBS is described by two parameters Q_s and T_s, where Q_s is the *server maximum budget*, T_s can be thought of as a kind of *server period*, while $\frac{Q_s}{T_s}$ is the server bandwidth. The service mechanism can be summarized as follows: when a new request arrives at time $r_{i,j}$, the server checks whether it can be scheduled using the last assigned deadline, otherwise the request is assigned an initial deadline equal to $r_{i,j} + T_s$. If the job executes for more than Q_s time units, its scheduling deadline is postponed by T_s. In this way, it is guaranteed that each active task can execute for at least Q_s time units in every interval T_s, but can not demand more than Q_s time units in the same interval. If a task misbehaves (i.e., it requests more than expected), its execution is slowed down without compromising the QoS guaranteed to the other tasks (temporal isolation). It is worth pointing out that if the worst case execution time C_i and the minimum interarrival time T_i of task τ_i are exactly known, using a CBS with parameters $Q_s = C_i$ and $T_s = T_i$ a bandwidth $U_s = \frac{Q_s}{T_s} = \frac{C_i}{T_i}$ is reserved to the task and it is guaranteed that the scheduling deadline will not be postponed (hard guarantee).

We can now introduce the following definition:

Definition 1 *Given a task τ_i serviced by a CBS having parameters Q_s and T_s, if a job $J_{i,j}$ is issued for the task at time $r_{i,j}$, the* CBS scheduling error $\epsilon_{i,j}$ *is defined as follows:*

$$\epsilon_{i,j} = d^s_{i,j} - r_{i,j} - T_s \tag{1}$$

where $d^s_{i,j}$ is the last deadline assigned to $J_{i,\ j}$ (after possibly having been postponed).

We remark that if the deadline is not postponed the scheduling error is zero. Also, the scheduling error is meaningfully defined only after the first deadline assignment (i.e., the first job). When $\epsilon_{i,j} > 0$, the task is not assigned a sufficient bandwidth and its reserved budget Q_s has to be increased until $\epsilon_{i,j}$ becomes zero.

3. Adaptive Control

The CBS scheduling system can be viewed as a a "plant" to be controlled, represented by a difference equation. Considering a single task, the scheduling error as defined in Equation(1) is the output variable; following a traditional control notation we can denote such a variable with the $y(k)$ symbol. The input variable is the reserved bandwidth $B(k)$. The control problem can be stated as follows: find a control law for the input variable $B(k)$ such that

$$\lim_{k \to \infty} y(k) = 0.$$

The plant is very hard to treat for several reasons. Both the state and the output evolution follow a nonlinear law. Moreover, the system is excited by a random signal (the tasks computation time) which cannot be externally controlled. Designing a control law directly tailored on the nonlinear and stochastic system model is a hard task. A typical approach, in these cases, is to find an approximating linear system in the neighborhood of an equilibrium point and design the control law referring to the linearized system. This approach cannot be applied "as it is", since the system under analysis exhibits a very dynamical behaviour and can swiftly change its working points depending on the characteristics of the data stream.

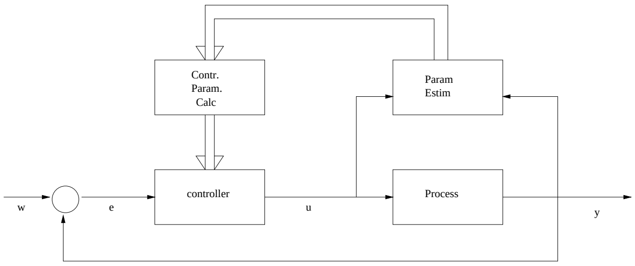

Figure 1. Scheme of a self-tuning regulator (STR)

One of the solutions proposed in the literature to cope with the plant's variations is adaptive control, in which the controller parameters are updated depending on the "situation" as it is identified on-line. Readers interested to a more complete understanding of adaptive techniques within linear control schemes is referred to [12] and [4] and to the citations therein.

The control scheme that we have used in our application belongs to the class of self-tuning regulators and is shown in Figure 1. The scheme is composed of two control loops. The inner loop includes the plant and an ordinary linear feedback regulator. The outer loop is aimed at estimating the plant model and adjusting the regulator parameters. This scheme can be viewed as an automated process for identifying the system and designing the controller, in which the plant "model" and the control scheme are updated at each sampling period. When designing this kind of controllers at least three points have to be addressed. The first point is what kind of mathematical model should be used for the plant. The second one is how to perform the online model estimation. Thirdly, a choice has to be made on what control law shall be used and how it can be adapted to the plant model being estimated.

As far as the system model is concerned, we have used the **ARMAX** model described by:

$$A(z^{-1})y(z) - B(z^{-1})z^{-d}u(z) = D(z^{-1})v(z). \tag{2}$$

where $y(z)$ is the *Z-transform* of the output signal, $u(z)$ is the *Z-transform* of the input signal, and $v(z)$ is the *Z-transform* of a statistically independent stochastic signal $v(k)$ having mean $E\{v(k)\} = 0$ and variance σ_v^2. A, B, D are polynomials in the variable z^{-1} having constant coefficients: The d parameter is an integer non-negative number called "deadtime".

The set of parameters to be identified are: $\theta^{\mathrm{T}} = [a_1, \ldots, a_m; b_1, \ldots, b_m; d_1, \ldots, d_m]$.

The experimental determination of the dynamical behaviour of a process and of its signals is called *identification*. Since our models are parametric we will use the term "parameters estimation" as a synonymous of "identification". Moreover, for our purposes, we are mostly interested in the identification algorithms based on a *real-time* processing, according to which at each sampling instant, input and output signals are collected and used to refine the estimation of the system's parameters. Such techniques will be called, following the current literature, *recursive parameter estimations*.

Classical methods belonging to this class are *least square estimation* and its variants. A comprehensive discussion of this approach is out of the scope of this paper and can be found in [16, 9].

The basic idea is the following. At every step, the current measurements for the inputs and the outputs are used to update the $\phi = [-y(k-1) \ \ldots - y(k-m)u(k-d-1)\ldots u(k-d-m)]^{\mathrm{T}}$ vector; then the following equation is applied:

$$
\begin{aligned}
K(k) &= P(k)\boldsymbol{\phi}(k) \\
\hat{\boldsymbol{\theta}}(k) &= \hat{\boldsymbol{\theta}}(k-1) + K(k)(y(k) - \boldsymbol{\phi}^{\mathrm{T}}(k)\hat{\boldsymbol{\theta}}(k-1)) \\
P(k) &= (I - K(k)\boldsymbol{\phi}^{\mathrm{T}}(k))P(k-1)/\lambda \qquad (3)
\end{aligned}
$$

where $\hat{\boldsymbol{\theta}}$ is the estimated parameters vector, and P is a covariance matrix, which for practical purposes, can be initialized with a diagonal matrix having a wide norm, and λ is a forgetting factor used to account for the parameters time variations.

This method's correctness is boring to prove but, at the same time, very intuitive. According to this method, at each step, the parameters estimate is corrected proportionally to "how far we missed the target" (Equation(3)) i.e., to the difference between the actual measurement $y(k)$ and the measurement based on the previous estimate $\hat{\boldsymbol{\theta}}(k-1)$. The other equations are used to compute the appropriate weighting factors.

The last problem to be addressed is the choice of the control law and of its adaptation mechanism. One of the best known approaches for realizing simple controllers is the PID controller, which provides acceptable solutions for a surprisingly large class of control problems. Recent works [14, 20] have proven the effectiveness of using a PID controller in the class of applications we are dealing with. However, none of the cited works offers solutions to the problem of tuning the controller parameters.

The basic PID controller was first conceived in the continuous time domain. For small sampling time, its behaviour can be approximated by the following recursive discrete expression:

$$
\begin{aligned}
u(k) - u(k-1) &= q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) \\
q_0 &= K(1 + \frac{T_D}{T_0}) \\
q_1 &= -K(1 + 2\frac{T_D}{T_0} - \frac{T_0}{T_I}) \\
q_2 &= K\frac{T_D}{T_0}
\end{aligned}
$$

where e is the difference between the desired output and the measured one, and K, T_I andT_D are the proportional gain, the integration time and the derivative time of the continuous time equation, respectively, while T_0 is the sampling period. In order to apply a PID algorithm in the context of an adaptive controller we have two specify the criteria for choosing on-line the q_i parameters, or, in other words, for solving on-line a control design problem. We have chosen the approach of assigning the closed-loop poles. Suppose the plant transfer function be $G_p(z)$, and the PID controller have the form $G_c(z) = \frac{q_0 + q_1 z^{-1} + q_2 z^{-2}}{1 - z^{-1}}$. The resulting characteristic equation of the closed loop system is $1 + G_c\ G_p$. The design technique is based upon assigning the pole of the closed loop system.

Since the proposed controller has only three free parameters (q_0, q_1, q_2) and the number of the resulting polynomial coefficients is $m + d + 2$, the method can be applied exactly only if $m + d + 2 = 3$. In our study we chose $d = 0$ and $m = 1$.

4. Experimental Results

The application of the presented adaptive control techniques has been tested on a meaningful case study consisting of a set of CBS served periodic tasks having a wide variance in the computation time requested by each job. We have identified such an application in a set of MPEG players processing sequences of frames in different windows on the screen. Since the streams are characterized by a variable bit rate (VBR), the frame decoding time are subject to large variations. Further stochastic variations in the execution time of each job are due to the possibility of dynamically opening other windows.

Our implementation framework has been based on the QoS manager described in [2], implemented on the HARTIK real-time kernel [13]. This kernel natively supports EDF scheduling and bandwidth reservation through the CBS. In this section, the execution environment is briefly described and then the experimental results are presented.

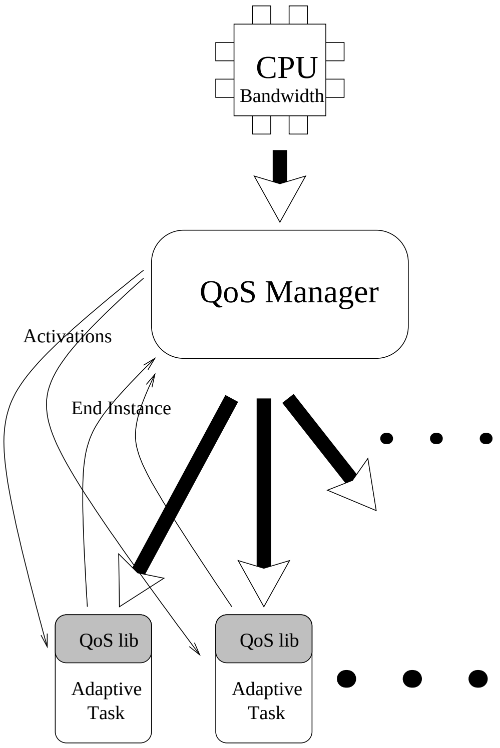

Figure 2. Scheme of the QoS Manager.

The system is composed of two different types of tasks: the regular tasks and the adaptive tasks. Regular tasks are characterized by *fixed* scheduling parameters, and are implemented as HARTIK (periodic or aperiodic) real-time tasks, while adaptive tasks have time-varying (adaptive) parameters, as described in the previous sections[1].

Since the visibility of the parameters relative to all of the adaptive tasks' is needed to perform the bandwidth adaptation, an active entity (the QoS manager) is used to adjust the parameters. The QoS manager is implemented ad a regular aperiodic task, called `qosman`, having the global system visibility (see Figure 2). This task is used to create adaptive tasks and to manage their bandwidth according to some user defined policy. The proposed solution is more flexible and portable than a kernel hardwired one (modification of the kernel scheduler to support QoS management), and can be easily implemented on other kernels that do not natively support scheduling parameters adaptation.

When the `qosman` task is created, it takes from the system all the available CPU bandwidth in order to distribute it among the demanding tasks.

When the application needs to create an adaptive task, it must issue a request to the `qosman` (through a library call), specifying the task period T_i. After this call the task is created and added to the set of managed tasks. The task is served by a CBS with server period $T_s = T_i$, and maximum budget Q_s starting from a small value.

At each period, the adaptive task is activated (i.e., a new job is created for that task). When the job finishes, the task has to notify this event to the `qosman` task (through another library call); in this way, `qosman` has the possibility of measuring a set of quantities of interest for its activities on the QoS management. For our purposes, the measured quantity is the *CBS scheduling error* $\epsilon_{i,j}$, which is our controlled variable $y_i(j)$. After measuring this quantity, `qosman` computes the new reserved bandwidth $B_i = u_i(j) = G(u_i(j-1), u_i(j-2), ..., y_i(j), y_i(j-1), ...)$ according to the feedback control law and adjusts the maximum budget of the CBS serving the task ($Q_s = B_i T_i$); then the task is deactivated.

When a new task is created or when the QoS manager decides to increase the bandwidth reserved to an adaptive task (as a response to a job termination), it is possible that $\sum_i B_i > 1$; in this case all the reserved bandwidths have to be compressed. In [2] we proposed a compression equation

$$\hat{B}_i = B_i w_i \frac{B_{max}}{\sum_j B_j w_j};$$

for the experiments on adaptive control we chose to give all tasks the same importance ($\forall \tau_i, \tau_j, w_i = w_j$), obtaining a simple proportional compression rule:

$$\hat{B}_i = B_i \frac{B_{max}}{\sum_j B_j}.$$

Using this framework, we implemented the PID adaptive control mechanisms. At the end of each job the QoS manager is activated and performs the following actions:

1. sample the scheduling error;
2. use the newly sampled data (and the stored old ones) $[u_i(j-1), u_i(j-2), ..., y_i(j), y_i(j-1), ...]$ to refine the a_i, b_i parameters estimation (using the RLS algorithm);
3. use the updated model to tune the controller parameters;
4. apply the control law to compute the new assigned bandwidth B_i;
5. adjust the scheduling parameter $Q_s = B_i T_i$.

[1] in this work we consider only periodic adaptive tasks

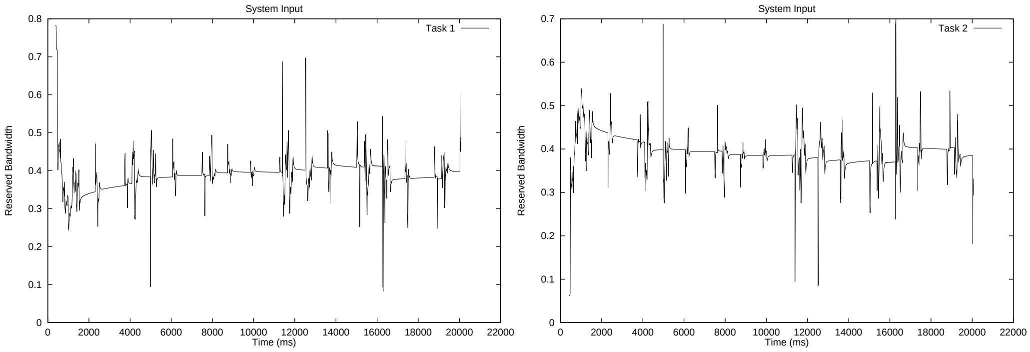

Figure 3. Evolution of the reserved bandwidth (two identical MPEG Players, Adaptive PID controller).

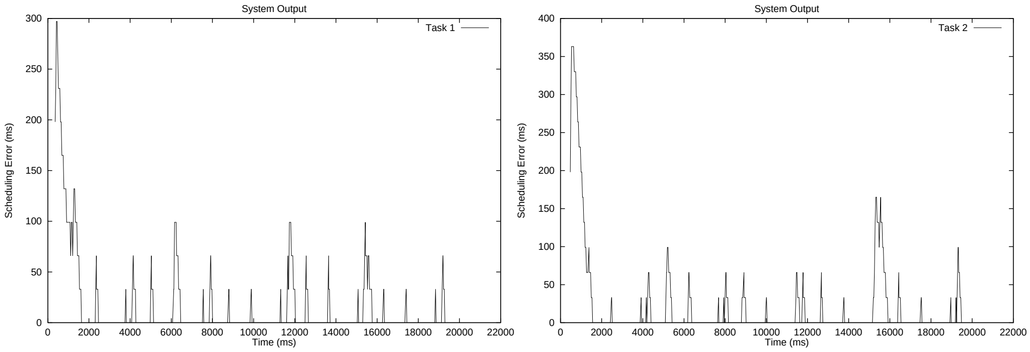

Figure 4. Evolution of the scheduling error (two identical MPEG Players, Adaptive PID controller).

The adaptive control schemes have been used to play multiple VBR MPEG streams, measuring the inter-frames times and the controlled CBS scheduling error.

We assumed the following plant model:

$$y(t) = -ay(t-1) + bu(t-1);$$

the assigned closed-loop poles were $Z_1 = 0.3$, $Z_2 = 0.2$, and $Z_3 = 0.1$.

The adaptive PID controller has been able to properly adjust the reserved bandwidth and to control the scheduling error to zero also in the case of multiple activated tasks. Experiments on a large number of task sets reveal that the controller needs an initial transient for "learning" the system model before its action becomes effective.

As an example, we report the controller behaviour on a specific experiment: the task set is composed of 2 adaptive tasks τ_1 and τ_2, with periods $T_1 = T_2 = 33ms$ decoding the same MPEG stream. Figure 3 plots the evolution of the reserved bandwidth over time, while Figure 5 plots the evolution of the estimated parameters. Figure 4 plots the evolution of the CBS scheduling error: as can be seen, after time $2000ms$ the controlled value is near the desired value 0. Since the system workload is near to one, the scheduling error cannot be constantly kept at the 0 value. A reasonable explanation can be given to some of the observed phenomena in the output dynamics. During the initial phase the most important signal driving the system identification process is the scheduling error, and the role played by the stochastic disturbances (the variations in the frame decoding time) is limited. When the scheduling error is finally reduced to very low values, the disturbances importance increase. As a consequence in the presence of frames requiring a big decoding time, the systems estimated parameters can experience some fluctuations which result in a ripple on the output variable. However, the obtained QoS is not seriously affected.

5. Conclusions and Open Issues

In this paper we have presented a novel approach for using feedback in real-time scheduling, based on adaptive control theory.

The proposed technique allows to adjust the resources assigned to every single task without requir-

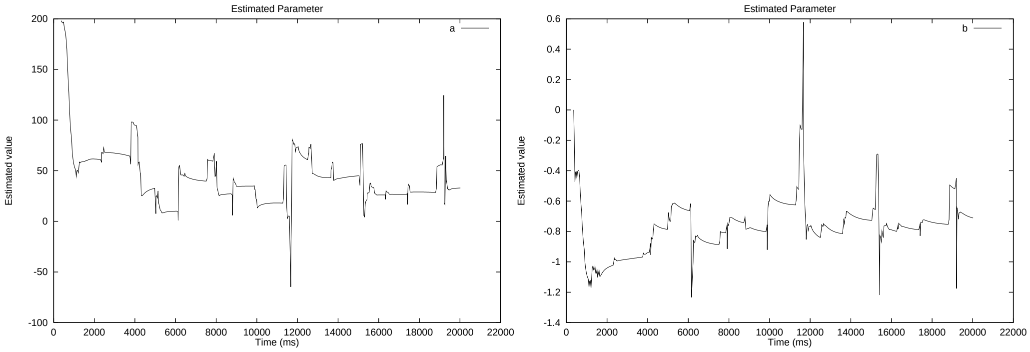

Figure 5. Evolution of the estimated ARMAX parameters for Task 1.

ing any a-priory knowledge about the task requirements and without any consideration about the system model. For this reason, this scheme is very general and can easily be applied in different contexts. The implementation solutions can be applied to different architectures with little efforts.

This work is an initial step towards a more systematic application of control theory to resource allocation problems, and proposes a completely new perspective. Many important issues, on this way, are still open fields for research.

The experimental results obtained using the PID controller are promising. Adaptive control turns out to be an effective choice for dynamically adjusting resource reservation. However, the control literature offers a wide set of solutions to the same problem. Restricting to the field of linear adaptive control, there are families of controllers other than PID. In particular an interesting alternative could be the minimum variance controllers which are able to explicitly keep into account stochastic disturbances. One of our future works will be testing these possibilities. As far as the PID controller is concerned, we are interested in exploring the assigned poles influence on the system performance, through a set of well-defined and significant metrics.

Another interesting work could be applying Kalman filtering techniques to account for the random factors derived from the job execution times. From the implementation point of view, we aim at extending our scheme to non periodic tasks and to port the QoS manager to platforms other than HARTIK.

References

[1] L. Abeni and G. Buttazzo. Integrating multimedia applications in hard real-time systems. In *Proceedings of the IEEE Real Time Systems Symposium*, Madrid, Spain, December 1998.

[2] L. Abeni and G. Buttazzo. Adaptive bandwidth reservation for multimedia computing. In *Proceedings of the IEEE Real Time Computing Systems and Applications*, Hong Kong, December 1999.

[3] L. Abeni and G. Buttazzo. Constant bandwidth vs proportional share resource allocation. In *Proceedings of the IEEE International Conference on Mutimedia Computing and Systems*, Florence, Italy, June 1999.

[4] K. Åstrom and B. Wittenmark. *Adaptive control.* Addison-Wesley, 1989.

[5] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. Wellings. Applying new scheduling theory to static priority preemptive scheduling. *Software Engineering Journal*, 8(5):284–292, September 1993.

[6] S. K. Baruah, A. K. Mok, and L. E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *IEEE Real Time System Symposium*, pages 182–190, Orlando, Florida, December 1990.

[7] G. Buttazzo, L. Abeni, and G. Lipari. Elastic task model for adaptive rate control. In *Proceedings of the IEEE Real Time Systems Symposium*, Madrid, Spain, December 1998.

[8] H. Fujita, T. Nakajima, and H. Tezuka. A processor reservation system supporting dynamic qos control. In *2nd International Workshop on Real-Time Computing Systems and Applications*, October 1995.

[9] G. C. Goodwin and K. S. Sin. *Adaptive Filtering, Prediction and Control.* Prentice Hall, Englewood Cliffs, N.J, 1984.

[10] P. Goyal, X. Guo, and H. M. Vin. A hierarchical cpu scheduler for multimedia operating systems. In *2nd OSDI Symposium*, October 1996.

[11] H. hua Chu and K. Nahrstedt. CPU service classes for multimedia applications. In *Proceedings of the IEEE International Conference on Mutimedia Computing and Systems*, Florence, Italy, June 1999.

[12] R. Isermann. *Digital Control Systems.* Springer-Verlag, 1989.

[13] G. Lamastra, G. Lipari, G. Buttazzo, A. Casile, and F. Conticelli. Hartik 3.0: A portable system for developing real-time applications. In *Proceedings of the IEEE Conference on Real-Time Computing Systems and Applications*, October 1997.

[14] B. Li and K. Nahrstedt. A control theoretical model for quality of service adaptations. In *Proceedings of Sixth International Workshop on Quality of Service*, 1998.

[15] C. L. Liu and J. Layland. Scheduling alghorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1), 1973.

[16] L. Ljung and T. Söderström. *Theory and Practice of Recursive Identification.* MIT Press, Cambridge: Mass, 1983.

[17] T. Nakajima and H. Tezuka. A continuous media application supporting dynamic qos control on real-time mach. In *ACM Multimedia*, 1994.

[18] R. Rajkumar, K. Juvva, A. Molano, and S. Oikawa. Resource kernels: A resource-centric approach to real-time and multimedia systems. In *Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking*, January 1998.

[19] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek. A resource allocation model for qos management. In *Proceedings of the IEEE Real Time Systems Symposium*, 1997.

[20] J. A. Stankovic, C. Lu, and S. H. Son. The case for feedback control in real-time scheduling. In *Proceedings of the IEEE Euromicro Conference on Real-Time*, York, England, June 1998.

[21] I. Stoica, H. Abdel-Wahab, and K. Jeffay. On the duality between resource reservation and proportional share resource allocation. In *SPIE Multimedia Computing and Networking*, volume 3020, pages 207–214, San Jose, CA, February 1997.

[22] I. Stoica, H. Abdel-Wahab, K. Jeffay, S. K. Baruah, J. E. Gehrke, and C. G. Plaxton. A proportional share resource allocation algorithm for real-time, time-shared systems. In *Proceedings of the IEEE Real Time Systems Symposium*, December 1996.

[23] D. K. Y. Yau and S. S. Lam. Adaptive rate controlled scheduling for multimedia applications. *IEEE/ACM Transactions on Networking*, August 1997.