

Stochastic feedback-based control of QoS in soft real-time systems

T. Cucinotta, L. Palopoli, L. Marzario

Abstract—This paper investigates application of feedback based control mechanisms to the problem of scheduling soft real-time tasks on a shared CPU. Each task has an execution time stochastically varying in time and it has to be provided with a specified level of Quality of Service (QoS). The problem of feedback control is formalised in the stochastic domain, by expressing QoS requirements in terms of properties to be satisfied by the stochastic process describing the evolution of the system state. We present several control schemes and address the fundamental problem of stochastic stability. Experimental results collected by a modified version of the Linux operating system show the effectiveness of the approach and its practical feasibility.

I. INTRODUCTION

The ubiquitous presence of networked computing systems in consumer electronics has emphasised the importance of appropriate resource allocation policies. The general situation is one where different activities require the same resources in the same time. Each activity has to provide the user with a specified level of Quality of Service (QoS) but its resource requirements change in time and a static allocation may prove unfeasible or inefficient: This is a perfect scenario for the application of feedback control.

This type of applications has gained momentum in the last few years with particular focus on the control of computer networks and queues [1]. An emerging trend is the application of feedback control techniques to the management of resources inside the Operating System (OS) of a computer, which is a manager for hardware resources shared between multiple software tasks. Whenever an application implemented by a task is endowed with real-time constraints, the OS scheduler has to allocate enough resources “in the right time” so that such constraints can be met. For many applications real-time constraints are *soft*, i.e., occasional violations entail only QoS degradations without any danger for the integrity of the system or for the people safety. As an example, for a video streaming application it is not necessary to decode every frame within a fixed interval as long as fluctuations in the decoding rate do not overcome the threshold of human perception. In this context properly designed schedulers offer considerably better performance than conventional solutions. As a representative example of this class of problems, in this paper we will specifically consider the problem of CPU scheduling. A wealth of soft

real-time algorithms have been proposed in the last few years for the CPU. In particular, two classes of algorithms known as Resource Reservation (RR) [2] and Proportional Share [3] offer close approximations of a “fluid” CPU allocation: each task executes as if on a dedicated virtual processor whose speed is a fraction of the actual processor. Even though the use of such techniques proves beneficial for many reasons, the problem of how to choose the fraction of CPU (bandwidth) allocated to each task is left unsolved. The main problem is that the execution requirements of each task that are hardly known before executing it and they may dramatically change in time. This is a strong motivation for the application of feedback control to adjust the bandwidth based on QoS measures. It is perfectly conceivable that in response to a QoS degradation the amount of CPU dedicated to the task be increased, while it can be decreased if the task is receiving too much jeopardising the execution of other activities. In order for such a feedback mechanism to be acceptable in the context of the OS scheduler, its introduced overhead must be negligible as compared to the normal bookkeeping activities of the scheduler. This inhibits the use of scheduling algorithms that could be computationally expensive.

The idea outlined above has been explored in several works. Indeed, feedback control techniques have been recently applied to real-time scheduling [4], [5], [6] and multimedia systems [7], [8]. However, owing to the difficulties in modelling schedulers as dynamic systems, these works could offer little evidence of their theoretical soundness. The use of a RR scheduling algorithm allowed us to fill in this gap. Based on a precise dynamic model for the system’s evolution, a switching PI strategy was developed [9] and formal proofs on the stability of the resulting schemes were offered in [10]. In [11], ad-hoc nonlinear feedback laws were proposed for the same system and the sequence of execution times for a task have been characterised in terms of worst case effects on the system stability. A significant drawback with this approach is that any knowledge on the stochastic properties of the execution times process is dissipated. In fact, whenever such a knowledge is available, it can be used to improve the system’s performance. To achieve this goal, the control design problem has to be attacked directly in the stochastic domain, which is the goal of this paper. In particular we will first show how the problem can be cast into the framework of a stochastic control problem. After taking this step, we are able to design control laws whose goals and requirements can be expressed

This work has been partially supported by the European OCERA IST-2001-35102 and RECSYS IST-2001-32515 projects, and by the Italian Ministry of Education (MIUR) PRIN project 2002 “Software Solutions for Embedded Platforms”

in terms of the stochastic properties of the process representing the evolution of the system state. A fundamental concern on the soundness of the approach regards stochastic stability: does the evolution of the system lead to unique steady state distribution of the system state? This complex problem is addressed in the paper and some analytical criteria are presented to decide whether a given feedback law is endowed with this property. Such criteria can be successfully applied to the control schemes presented in this paper concluding their mathematical well-formedness. The technological feasibility of the proposed approach by means of a minimally invasive set of modifications to the Linux OS is shown in [12]. Using this modified OS, we are able to show in this paper the applicability of the approach in real-life situations.

II. PROBLEM DESCRIPTION

In this section, after a quick introduction on basic concepts concerning real-time scheduling, we describe the problem of QoS control in soft real-time systems.

A. Basic definitions on real-time processor scheduling

A real-time computing system consists of a set of concurrently running software activities (tasks) $\{\tau^{(i)}, i = 1, \dots, n\}$. Each task $\tau^{(i)}$ is a program consisting of a stream of jobs (execution instances) $\{J_k^{(i)}\}_{k=1,2,\dots}$. The k^{th} job $J_k^{(i)}$ of the i^{th} task arrives (i.e. becomes executable) at time $r_k^{(i)}$ and requires a variable computation time $e_k^{(i)}$; the finishing time of $J_k^{(i)}$, depending on the scheduling performed by the OS, is denoted as $f_k^{(i)}$. For real-time tasks, each job $J_k^{(i)}$ is associated a deadline $d_k^{(i)}$, which is said to be *respected* if $f_k^{(i)} \leq d_k^{(i)}$, *violated* otherwise ($f_k^{(i)} > d_k^{(i)}$). While in the hard real-time setting all deadlines must be respected, for soft real-time tasks occasional deadline misses can be tolerated provided that this deviation be kept in check. We will be more formal on this issue later. For the scope of this paper, we will consider only periodically activated tasks, i.e. $r_k^{(i)} = kT^{(i)}$, where $T^{(i)}$ is the task's activation period. Moreover, for the sake of simplicity, the deadline of a job will be set equal to the activation instant of the next one: $d_k^{(i)} = r_{k+1}^{(i)}$.

B. Resource Reservation scheduling

The present work is focused on the analysis and control of a set of tasks scheduled according to a RR algorithm, first proposed in [2] then developed in [13], [14], [15]. Using such techniques, each task $\tau^{(i)}$ is associated a pair $[Q^{(i)}, P^{(i)}]$ meaning that the task is guaranteed a *budget* of $Q^{(i)}$ execution time units for every allocation period $P^{(i)}$, whenever in need. Thus the ratio $b^{(i)} \triangleq \frac{Q^{(i)}}{P^{(i)}}$ is the utilisation fraction (bandwidth) of the CPU allocated to the task. Even though the allocation period $P^{(i)}$ may be arbitrarily chosen in RR techniques, we will assume that $P^{(i)}$ be small as compared to the task period $T^{(i)}$. Under this assumption RR scheduling approximates a fluid

allocation of the processor. Indeed, introducing the *virtual finishing time* $v_k^{(i)}$ as the time a job $J_k^{(i)}$ would finish if it were running on a dedicated processor of speed $b^{(i)}$ times the real CPU speed, it is possible to prove [15] that, if the RR paradigm is strictly applied, then $v_k^{(i)}$ approximates $f_k^{(i)}$ with a maximum error of $P^{(i)}(1 - b^{(i)})$. For this reason, we will henceforth assume a perfectly fluid allocation of the CPU, neglecting the technological problems introduced by an overly small choice of $P^{(i)}$ (i.e. OS overhead due to the frequent CPU switches between the tasks).

Clearly, when multiple tasks populate a system at the same time, the total sum of the allocated bandwidths $b^{(i)}$ cannot exceed the total processor's availability of computing power: $U \sum_i b^{(i)} \leq U$. The constant $U \leq 1$ accounts for the efficiency of the scheduling algorithm.

C. Quality of Service metrics and dynamic model

In order to properly adjust the scheduling parameters (e.g. $Q^{(i)}$ and $P^{(i)}$) of each task, it is of paramount importance to quantify the Quality of Service (QoS) that the task experiences during its execution. Since in our model we tolerate occasional violations of the deadlines, it is reasonable to use, as a QoS metric, the *scheduling error* (s.e.) defined as the difference between the finishing time of a job and its deadline, measured relatively to the task period. Referring to the fluid allocation assumption, the definition of this quantity is $\varepsilon_k^{(i)} = \frac{v_k^{(i)} - d_k^{(i)}}{T^{(i)}}$. An ideal bandwidth allocation would be one for which $\varepsilon_k^{(i)} = 0$ for all k and i . Indeed, both $\varepsilon_k^{(i)} > 0$ and $\varepsilon_k^{(i)} < 0$ are undesirable situations, since in the former case the job does not respect its timing constraint, whilst in the latter one it is given more bandwidth than strictly required.

For notational convenience, the task superscript (i) will be dropped whenever all cited quantities refer to the same task. Furthermore, the symbol c_k will be used as a shorthand for e_k/T . The dynamics of ε_k will be characterised by a function $\mathfrak{S}_C(\cdot)$, representing the s.e. value at the next step based on: 1) the s.e. at the current step, 2) the allocated bandwidth for the next job execution, and 3) its computation time. As shown in [9], such a function can be expressed as as follows

$$\varepsilon_{k+1} = \mathfrak{S}_C(\varepsilon_k, c_k, b_k) = s(\varepsilon_k) + \frac{c_k}{b_k} - 1, \quad (1)$$

where $s(\cdot)$ is defined as $s(x) \triangleq x$ if $x > 0$ and $s(x) \triangleq 0$ otherwise.

D. The Stochastic control problem

The model in Equation 1 describes a discrete-event systems whose evolution is observed on the job finishing times $\{f_k\}$.

For each task, given the impossibility of exact predictive knowledge of the job execution times, the sequence $\{c_k\}$ is modelled as the discrete-time, continuous-state stochastic process (s.p.) $\{C_k\}$, where the k^{th} job execution time is

modelled as the stochastic variable (s.v.) C_k . The scheduling error ε_k is assumed to be a measurable quantity (i.e. the OS is endowed with an appropriate “sensor” that allows us to measure ε_k upon the termination of the k -th job). This information allows us to “reconstruct” the computation time of the Job J_{k-1} . Therefore a component controller can be used to decide the bandwidth b_k to be used for job J_k by means of a feedback function $\mathfrak{B}(\cdot)$ depending on the current system state and on the sequence of past job execution times. The assigned value for the bandwidth cannot exceed a saturation limit $B_H \in]0, 1]$ that is fixed *a-priori* for each task. Therefore $b_k = \mathfrak{B}(\varepsilon_k, B_H, \{c_{k-1}, c_{k-2}, \dots\})$.

Starting from a null scheduling error ($\varepsilon_1 \triangleq 0$) and considering the closed loop evolution of the system, both the sequence of the scheduling errors and of the assigned bandwidth can be modelled as stochastic processes that will be respectively denoted as $\{B_k\}$ and $\{\mathcal{E}_k\}$. The evolution of the s.e. process is stated in terms of the relation between stochastic variables given by the \mathfrak{S}_C function introduced above:

$$\mathcal{E}_{k+1} = \mathfrak{S}_C(\mathcal{E}_k, B_k, C_k) = s(\mathcal{E}_k) + \frac{C_k}{B_k} - 1. \quad (2)$$

E. Assessing control performance

After providing a representation of the scheduling error as a s.p., we can describe the QoS experienced by a task in terms of its stochastic properties.

A qualitative assessment of the controller performance can be made by plotting the first order probability density of the s.e. $f_{\mathcal{E}_k}(\cdot)$, when k tends to infinity. Namely, we would wish that such function be tightly centred around the value 0 for the scheduling error (ideally a Dirac δ). A quantitative assessment may be done on the basis of the achieved s.e. expected value $\mu_{\mathcal{E}_k} = E[\mathcal{E}_k]$ and standard deviation $\sigma_{\mathcal{E}_k} = E\left[(\mathcal{E}_k - \mu_{\mathcal{E}_k})^2\right]$. Another metric of interest can be defined in terms of the probability that the s.e. resides within a prefixed interval I near the origin $\Pr\{\mathcal{E}_k \in I\}$.

Such metrics, will assume a particular relevance for $k \rightarrow \infty$ whenever the resulting process reaches stochastic equilibrium, representing the steady state expected behaviour of the closed loop system.

III. CONTROL SCHEMES

The stochastic properties of the process $\{C_k\}$ play a role of paramount importance in our problem set-up. Such properties are to a large extent application specific. In this section, we will provide a general framework for designing such controllers, where application dependencies are contained in a specific component of the controller. The proposed architecture for the controller is comprised of two blocks and it is shown in Figure III. The first block (predictor) is used to predict the evolution of C_k . In our view, this block should be tailored on specific classes of applications. The second block - which is on the contrary fairly independent from the specific application - implements a feedback law based on the measured value of

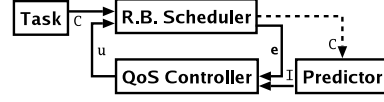


Fig. 1. Control loop. Assigned bandwidth depends on the current s.e. value and on the predicted behaviour of the input process.

ε_k and on the information received from the predictor. In this work we will restrict our attention particularly to this second block.

In particular we will present two possible control schemes that aim at different trade-offs between the following requirements: 1) implementation overhead, 2) simplicity of use, 3) QoS experienced by the application, 4) “minimality” of the bandwidth allocated to the task.

The discussion below refers to the case of i.i.d. $\{C_k\}$ process, but it is easily adaptable to the case of a correlated input process by introducing the proper conditioning events within the stochastic operators. This is not explicitated here for the sake of brevity and notational simplicity.

A. Stochastic invariant set

In the first control scheme, it is assumed that, at step k , the predictor provides an interval $[h_k, H_k]$ where C_k is expected to fall with a high probability. The prediction algorithm for constructing the interval $[h_k, H_k]$ is application specific. As an example, for a video decoder, it could be possible to perform a quick estimation of the upper and lower bounds for a frame decoding time by looking at the frame structure before starting the decoding itself. Because for the present discussion, we are restricting to the case of i.i.d. $\{C_k\}$, we will assume a fixed prediction interval $\forall k, [h_k, H_k] = [h, H]$. The control goal is constraining the evolution of the s.e. within a target interval $[-e, E]$ whenever the prediction of the $\{C_k\}$ is respected (i.e. $C_k \in [h, H]$). In particular, we require that if at some k the s.e. is in the set \mathcal{I} , it remains confined therein with a probability at least equal to the probability of producing a right prediction. This is a stochastic generalisation of the notion of controlled invariant set, which was introduced adopting a deterministic “worst-case” viewpoint in [11]. The following easy result provides a family of controllers guaranteeing stochastic invariance:

Proposition 1: Assume that, at step, 1) $\Pr\{C_k \in [h, H]\} \geq p$, 2) $e + \frac{h_k}{H_k}E \geq 1 - \frac{h_k}{H_k}$, 3) $B_H \geq H_k$. Then a controller choosing a bandwidth value $\mathfrak{B}(\varepsilon)$ in the range

$$\left[\frac{H}{1 + E - s(\varepsilon_k)}, \min \left\{ \frac{h}{1 - e - s(\varepsilon_k)}, B_H \right\} \right], \quad (3)$$

guarantees that

$$\Pr\{\mathcal{E}_{k+1} \in \mathcal{I} \mid \mathcal{E}_k = \varepsilon_k \wedge \varepsilon_k \in \mathcal{I}\} \geq p$$

With the introduced controller, a *recovery policy* must be undertaken when the state falls outside of the invariant set $\mathcal{I} = [-e, E]$. A reasonable policy is a minimum time strategy; it corresponds to choosing, for $\varepsilon_k > E$, the

maximum available bandwidth that preserves $\varepsilon_{k+1} \geq -e$ whenever $C_k \geq h_k$. This can be formulated as: $\mathfrak{B}(\varepsilon_k) =$

$$\begin{cases} \min \left\{ B_H, \frac{h_k}{1-e-\varepsilon_k} \right\} & \text{if } E < \varepsilon_k < 1 - e \\ B_H & \text{if } \varepsilon_k \geq 1 - e \end{cases} \quad (4)$$

Different trade-offs can be achieved by different choices of e and E . In particular, if the only concern is the probability of a deadline miss, one can simply set $\mathcal{I} = [-1, E]$, whereas a *narrow* \mathcal{I} ensures that the assigned bandwidth remains always close to the strict necessary.

B. Stochastic dead-beat (SDB)

The invariant-based technique presented above provides the user with a relatively fine control of the system performance by different choices of the interval extremes. However, this freedom for some applications can also be a source of unrequested complexity (e.g. in parameter tuning). In some cases, an effective goal can just be limited to the control of the next s.e. expected value. Since we want to control the expected value of the s.e. to a target value in one step, we call this technique stochastic dead-beat.

The following easy to prove result yields a control law satisfying such a requirement.

Proposition 2: Assume that, at step k , the predictor produces the expected value μ_{C_k} of the next job execution time C_k . Then a controller setting $\mathfrak{B}(\varepsilon_k) =$

$$\begin{cases} \frac{\mu_{C_k}}{1+\varepsilon^*-s(\varepsilon_k)} & \text{if } \varepsilon_k < 1 - \frac{\mu_{C_k}}{B_H} \\ B_H & \text{if } \varepsilon_k \geq 1 - \frac{\mu_{C_k}}{B_H} \end{cases}. \quad (5)$$

ensures that $E[\mathcal{E}_{k+1} \mid \mathcal{E}_k = \varepsilon_k] = \varepsilon^*$. where ε^* is the target value.

Clearly, dealing with i.i.d. $\{C_k\}$ the expected value μ_{C_k} is fixed (i.e., it does not depend on k). A particular case of interest is obtained by setting $\varepsilon^* = 0$ in Equation 5, achieving, at each step, a null expected scheduling error for the next job. As compared to the invariant based controller, the SDB requires a looser knowledge of the input stochastic process, i.e. just the μ_{C_k} parameter, thus simplifying the structure of the predictor.

IV. STOCHASTIC STABILITY OF THE PROPOSED CONTROLLERS

When evaluating correctness of the QoS control strategies introduced above, two properties are of great interest: stochastic stability, i.e., the existence and uniqueness of an invariant probability function for $\{\mathcal{E}_k\}$ when $k \rightarrow \infty$, and ergodicity of the resulting process. In the following, we will provide two general sufficient conditions for the existence and uniqueness of an invariant p.d.f. for the controlled systems that will serve as criteria to test such properties.

The theory involved in the presentation below is quite technical and involved, and it requires concepts borrowed from [16]. For the sake of simplicity and brevity, we will keep the discussion at an intuitive level, omitting some technical details that would enhance its mathematical rigour.

A. Basic definitions and Markov operator

The fundamental concept that we will work on is a generalised version of a Markov Chain (MC), for which the state space is not required to be finite or countably infinite (see Sec. 2.2 for a precise definition). The classical transition probability matrix is then replaced by a continuous integral operator, which is called *Markov Operator* (MO). Namely, let $f_k(\cdot)$ represent the p.d.f. of the scheduling error at step k , and let $f_0(\cdot)$ be the scheduling error p.d.f. at step 0. The MO $\mathcal{P} : D \rightarrow D$, where D denotes the set of probability density functions on the $[-1, +\infty[$ range, is such that $f_{k+1} = \mathcal{P}f_k$.

The MO for our system can be easily explicitated by considering the probability $f_{k+1}(y) dy = Pr\{y \leq \mathcal{E}_{k+1} < y + dy\}$, the relation 2 between the stochastic variables $\mathcal{E}_k, C_k, \mathcal{E}_{k+1}$, and conditioning to all possible values of the scheduling error at the current step, leading to

$$\mathcal{P}f(y) = \int_{-1}^{+\infty} f_{C_k}[\mathfrak{B}(x)(1+y-s(x))] \mathfrak{B}(x)f(x)dx. \quad (6)$$

Another quantity of interest, when dealing with properties of our system at stochastic equilibrium, is the *stochastic kernel* associated to the MO \mathcal{P} , i.e. the function $K(x, y)$ such that: $\mathcal{P}f(x) = \int K(x, y)f(y)dy$. From the explicit representation of \mathcal{P} of Equation 6, it easily follows

$$K(x, y) \triangleq f_C[\mathfrak{B}(y)(1+x-s(y))] \mathfrak{B}(y). \quad (7)$$

The presence of the control function $\mathfrak{B}(\cdot)$ in the stochastic kernel and in the MO makes our system a controlled MC. We require that the control action ensures the existence of an invariant p.d.f. f , independent by $f_0(\cdot)$ and given by $f = \lim_{k \rightarrow +\infty} f_k$, which is the fixed point of the \mathcal{P} operator, i.e. $f = \mathcal{P}f$. Finally, a technical property of interest for our discussion is the weak-Feller property of a MC. Roughly speaking, if a MC is endowed with this property, its probability transition function preserves continuity and real-boundedness of the functions it is applied on (see Def. 4.4.2 of the cited book for a more precise definition).

B. Sufficient conditions for the existence and uniqueness of an invariant p.d.f.

Theorem 1: Consider the system defined by Equation 2, evolving under the action of an input process $\{C_k\}$ i.i.d., for which the mean value and standard deviation exist, under the control of a generic control function $\mathfrak{B}(\cdot)$ satisfying the following requirements:

- (b₁) $\mathfrak{B}(\cdot)$ is continuous
- (b₂) $\mathfrak{B}(\cdot)$ is upper bounded by $B_H > \mu_C$, and $\forall \varepsilon \geq \varepsilon_H, \mathfrak{B}(\varepsilon) = B_H$
- (b₃) $\mathfrak{B}(\cdot)$ is lower bounded by a $B_L > 0$.

Then, an invariant p.d.f. exists for the scheduling error evolution.

Proof: The proof is given applying theorem 7.2.4 in [16]. This theorem states that sufficient conditions for the existence of an invariant probability function are: (a) the

MC respects the weak-Feller property, and (b) fixed an arbitrary, continuous, strictly positive function $f_0(\cdot)$, with the property that $\lim_{x \rightarrow \infty} f_0(x) = 0$, a nonnegative number b and a nonnegative measurable function $V(\cdot)$ not identically null exist, s.t. $\forall x, E[\mathcal{E}_{k+1} | \mathcal{E}_k = x] \leq V(x) - 1 + bf_0(x)$.

(a). As shown in the note at page 58 of [16], a MC of the form $\mathcal{E}_{k+1} = F(\mathcal{E}_k, C_k)$, with $\{C_k\}$ i.i.d., is weak-Feller if $F(\cdot, y)$ is continuous $\forall y$. In our case, $F(x, y) \equiv s(x) + \frac{y}{\mathfrak{B}(x)} - 1$ is always continuous, under the assumptions that the $\mathfrak{B}(\cdot)$ function be continuous and strictly positive, as from the assumptions.

(b). Define V as $V(x) \triangleq x^2$. Then, $E[V(\mathcal{E}_{k+1}) | \mathcal{E}_k = x] = E\left[\left(s(x) + \frac{C_k}{\mathfrak{B}(x)} - 1\right)^2\right] = [s(x) - 1]^2 + \frac{\mu_C^2 + \sigma_C^2}{\mathfrak{B}^2(x)} + 2[s(x) - 1]\frac{\mu_C}{\mathfrak{B}(x)} \leq x^2 - 1 + bf_0(x)$, where last inequality must be verified. For positive values of x , we have: $2(1-x)\left(1 - \frac{\mu_C}{\mathfrak{B}(x)}\right) + \frac{\mu_C^2 + \sigma_C^2}{\mathfrak{B}^2(x)} \leq bf_0(x)$. Consider the value x^* defined as $x^* \triangleq \max\{\varepsilon_H, 1\}$. It is clear that, if $\mu_C < B_H, \forall x > x^*$, the left member becomes negative, and diverges to $-\infty$ as $x \rightarrow \infty$. Thus, due to the strictly positive lower bound $B_L > 0$ on $\mathfrak{B}(x)$, a b value satisfying the inequality always exists: $b \geq \max_{x \in [-1, x^*]} \left\{ \frac{1}{f_0(x)} \left[2(1-x) \left(1 - \frac{\mu_C}{\mathfrak{B}(x)} \right) + \frac{\mu_C^2 + \sigma_C^2}{\mathfrak{B}^2(x)} \right] \right\}$. For negative values of x , instead, we have: $2\left(1 - \frac{\mu_C}{\mathfrak{B}(x)}\right) + \frac{\mu_C^2 + \sigma_C^2}{\mathfrak{B}^2(x)} \leq x^2 + bf_0(x)$. As $x \in [-1, 0]$, it is sufficient to take: $b \geq \max_{x \in [-1, 0]} \left\{ \frac{1}{f_0(x)} \left[2\left(1 - \frac{\mu_C}{\mathfrak{B}(x)}\right) + \frac{\mu_C^2 + \sigma_C^2}{\mathfrak{B}^2(x)} - x^2 \right] \right\}$. Thus, the final b value is the maximum between the two cases. ■

Given the p.d.f. of an input process modelling a real application, and a control function $\mathfrak{B}(\cdot)$, the stated result allows one to check the existence of an invariant p.d.f. for the closed loop system dynamics. In addition to that, we require that such an invariant p.d.f. be unique (hence, it is reached starting from any initial state). The following result offers a criterion that allows us to conclude uniqueness of the invariant p.d.f., if some additional assumption on the input process is made.

Theorem 2: Suppose, in addition to the assumptions made in Theorem 1, that the input process p.d.f. $f_C(\cdot)$ is strictly positive in an interval $[a, b]$ such that $a < B_L < B_H < b$, and is null if $c < a$. Then, a unique stationary p.d.f. exists for the s.e. evolution, and the system is ergodic.

Proof: Due to Theorem 1, a stationary p.d.f. exists. Then, it is sufficient to apply Proposition 4.2.2 in [16], stating that if a MC is irreducible, and admits an invariant p.d.f., then the invariant p.d.f. is unique and the MC is ergodic. In order to prove irreducibility, we have to show that the expected number of visits to any set A (with non null measure) is positive, given any start state x , which may also be written as: $\forall x \in [0, +\infty[, \forall A$ s.t. $\phi(A) > 0, \sum_{n=1}^{\infty} P^n(x, A) > 0$, with the transition probability function P given by: $P(x, A) = \int_A K(y, x) dy$. We will prove that, starting from any state x , there exist a sequence of expanding ranges $\{R_n \equiv [a_n, b_n]\}$ of y values where

$K^n(y, x)$ is certainly strictly positive, and that this sequence converges to the entire state space, for $n \rightarrow \infty$. In fact, $K(y, x) = f_C[\mathfrak{B}(x)(1 + y - s(x))] \mathfrak{B}(x)$, and $\mathfrak{B}(x) \leq B_H$ implies $\mathfrak{B}(x)(1 + y - s(x)) \leq B_H(1 + y - s(x)) \leq b$ which is implied by any $y \leq \frac{b}{B_H} + s(x) - 1 > s(x)$. Furthermore, $\mathfrak{B}(x) \geq B_L$ implies $\mathfrak{B}(x)(1 + y - s(x)) \geq B_L(1 + y - s(x)) \geq a$ which is implied by any $y \geq \frac{a}{B_L} + s(x) - 1 < s(x)$. This means that $a_1 = s(x) - (1 - \frac{a}{B_L})$, $b_1 = s(x) + (\frac{b}{B_H} - 1)$; $a_{n+1} = s(a_n) - (1 - \frac{a}{B_L})$, $b_{n+1} = s(b_n) + (\frac{b}{B_H} - 1)$, and the upper bound of each set grows of a value $(\frac{b}{B_H} - 1) > 0$ for each set in the sequence. Proof is concluded by observing that the region $\{\varepsilon < -(1 - \frac{a}{B_L})\}$ is never reachable, in any number of steps, thus it is sufficient to exclude it from the state space, and also from the possible x values, in the proof. Thus, on the remaining range $[-(1 - \frac{a}{B_L}), +\infty[$, the MC is completely reachable, thus it is irreducible. ■

Using the above criteria it is immediate to show that both the SDB controller and any continuous invariant-based function, among the ones allowed by Equation 3 and Equation 4, guarantee existence and uniqueness of an invariant probability function, and that the resulting $\{\mathcal{E}_k\}$ process is ergodic.

V. EXPERIMENTAL RESULTS

In this section we report experimental results collected using a modified version of the Linux Kernel, developed as a part of the OCERA E.U. project (see [12] for more architectural details). In particular, we focused on the context of multimedia applications, and applied our feedback based scheduling mechanism to the decoding thread of the *Xine* MPEG video player. All experiments shown in this section have been done using an AMD Athlon(tm) XP 1800+ based platform, running the Linux OS with kernel 2.4.18.

In the first experiment, a constant bandwidth equal to the average required bandwidth was used (15.5% for this specific movie). This choice results into unacceptable occasional degradations of the experienced QoS levels during the play. This is visible in Figure 2 (a) where for a long time-span the scheduling error experiences high positive deviations. On the other hand, increasing the static bandwidth allocation to 18%, the decoder behaves much better during the play, but the decoding thread most times uses a bandwidth that is much higher than strictly required. This is reflected in a scheduling error most times being negative and high in absolute value (not reported for the sake of brevity).

In a third experiment, we activated feedback based scheduling of the decoding thread. The controller used a predictor based on 3 independent moving averages, each based in its turn on 4 samples. This very simple strategy, based on multiple moving averages, is to be considered as pure proof of concept. In the future, we plan to experiment more sophisticated prediction techniques based on the theory of stochastic identification. The control algorithm we used is a dead-beat controller (with $\varepsilon^* = 0$), with a

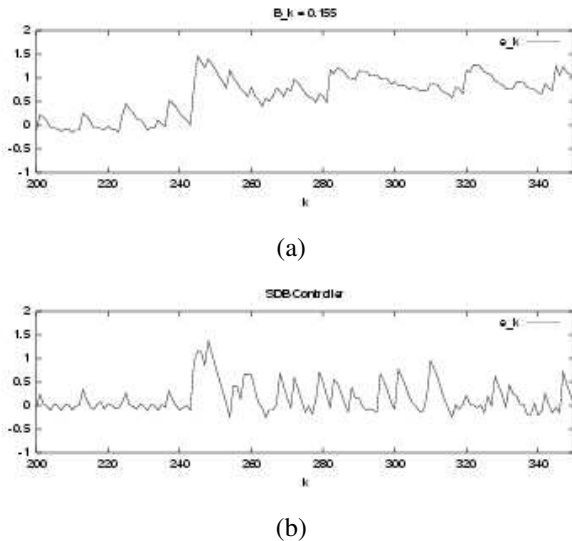


Fig. 2. Scheduling error experienced with (a) a bandwidth statically fixed to $b_k = 0, 155$ and (b) a SDB controller with a saturation value of 18.

saturation value of $B_H = 18\%$. This resulted in a movie played substantially as fluidly as with the static allocation of 18%, but an average allocation of the bandwidth of 16% during play. Figure 2 (b) shows the evolution of the s.e. under the action of the feedback based controller. The scheduling quickly recovers from occasional spikes, and its evolution remains most often constrained in a narrow interval.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, the problem of controlling the QoS experienced during execution of a soft real-time task has been attacked in the stochastic domain. QoS requirements were formalised in terms of stochastic properties of the scheduling error process. We showed different ways for formulating QoS requirements and presented two control schemes complying with such requirements. Furthermore, we provided a general criterion for assessing stochastic stability of the closed loop system dynamics, whenever a generic control action is in place.

Finally, experimental results have been reported, collected by an implementation of the introduced techniques into the Linux kernel, showing evident advantages in the application of feedback scheduling techniques to the decoding stage of a MPEG player.

Further investigations are possible around the research topics covered by this paper. First, we aim at relaxing stability conditions stated in Theorem 2. Most importantly, stochastic stability in the more general case of non-i.i.d. input process needs be investigated (by including a stochastic model for the $\{C_k\}$ process. Another interesting area of research is the application of the proposed techniques to the case tasks requiring access to multiple resources, which frequently occurs in many important multimedia applications.

REFERENCES

- [1] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.
- [2] C. W. Mercer, S. Savage, and H. Tokuda, "Processor capacity reserves for multimedia operating systems," Tech. Rep. CMU-CS-93-157, Carnegie Mellon University, Pittsburg, May 1993.
- [3] K. Jeffay, F. Smith, A. Moorthy, and J. Anderson, "Proportional share scheduling of operating system services for real-time applications," in *IEEE Real Time System Symposium*, (Madrid, Spain), December 1998.
- [4] T. Nakajima, "Resource reservation for adaptive qos mapping in real-time mach," in *Sixth International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*, April 1998.
- [5] 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.
- [6] C. Lu, J. A. Stankovic, T. F. Abdelzaher, G. Tao, S. H. Son, and M. Marley, "Performance specifications and metrics for adaptive real-time systems," in *Proceedings of the 21th IEEE Real-Time Systems Symposium*, (Orlando, FL), December 2000.
- [7] D. Steere, A. Goel, J. Gruenberg, D. McNamee, C. Pu, and J. Walpole, "A feedback-driven proportion allocator for real-rate scheduling," in *Proceedings of the Third usenix-osdi*, pub-usenix, feb 1999.
- [8] 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.
- [9] L. Abeni, L. Palopoli, G. Lipari, and J. Walpole, "Analysis of a reservation-based feedback scheduler," in *Proc. of the Real-Time Systems Symposium*, (Austin, Texas), November 2002.
- [10] L. Palopoli, L. Abeni, and G. Lipari, "On the application of hybrid control to cpu reservations," in *Hybrid systems Computation and Control (HSCC03)*, (Prague), april 2003.
- [11] L. Palopoli, T. Cucinotta, and A. Bicchì, "Quality of service control in soft real-time applications," in *Conference on Decision and Control (CDC)*, 2003.
- [12] L. Abeni, T. Cucinotta, G. Lipari, L. Marzario, and L. Palopoli, "Adaptive reservations in a linux environment," in *To appear in Proceedings of RTAS 2004*, 2004.
- [13] 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.
- [14] 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.
- [15] G. Lipari and S. Baruah, "Greedy reclaimation of unused bandwidth in constant bandwidth servers," in *IEEE Proceedings of the 12th Euromicro Conference on Real-Time Systems*, (Stokholm, Sweden), June 2000.
- [16] J. Lasserre and O. Hernandez-Lerna, *Markov chains and invariant probabilities*, vol. Progress in mathematics: Volume 211. Birkhaeuser Verlag, second ed., 2003.