

Weighted Feedback Reclaiming for Multimedia Applications *

Luigi Palopoli ^{#1}, Luca Abeni ^{#2}, Tommaso Cucinotta ^{&3}, Giuseppe Lipari ^{&4}, Sanjoy K. Baruah ^{^5}

[#] *University of Trento, Italy*

[&] *Scuola Superiore Sant'Anna, Italy*

[^] *University of North Carolina at Chapel Hill, USA*

{¹palopoli, ²abeni}@disi.unitn.it, {³t.cucinotta, ⁴g.lipari}@sssup.it, ⁵baruah@cs.unc.edu

Abstract

Resource reservations are a very popular choice to schedule multimedia tasks. However, the high variability of the resource requirements hinders a static choice of the scheduling parameters. In this paper we address this problem by a combination of two strategies: adaptive reservations and resource reclaiming. The first one operates "locally" (using the information of a single task), the second one operates "globally" distributing unused bandwidth between the tasks. In this paper, we show by analytical results and by extensive simulations that the two techniques can be safely and usefully combined.

1. Introduction

Applications processing multimedia content (such as compressed audio or video) are characterised by implicit temporal constraints [11, 21]. As an example, a video player should decode and reproduce each video frame within a specified deadline (given by the frame's Presentation TimeStamp - PTS). Occasional violations of these deadlines cause a degradation in the perceived Quality of Service (QoS), but do not invalidate the computation. Moreover, the time needed to process a single frame exhibits large variations, and is highly dependent on the input (think about the decoding times in an MPEG decoder) [11, 21]. Important examples of this behaviour can be found in such multimedia applications as players or streaming programs, Voice Over IP applications, Video on Demand clients, etc.

The high variability of the computation times makes the application of classical real-time scheduling theory [15] troublesome, because assuming worst case execution times leads to a very conservative design. This problem can be addressed by implementing some form of QoS adaptation in the applications, or by using more flexible and dynamic resource allocation policies. The first solution [25, 24, 11, 12] can be based on scalable video or audio processing mecha-

nisms [10], which allow one to reduce the execution times at the cost of a reduced quality. A similar idea is quality adaptation [23, 18, 6]. In response to time-varying application requirements and availability of shared resources, the behaviour of the applications is changed at run-time to make their requirements fit in the instantaneous resource availability. Although a very effective technique, application level adaptation relies on a particular structure for the applications (e.g., the presence of discrete quality options that can be switched online). Therefore, in this paper we adopt a complementary strategy that operates on resource allocation.

In the context of multimedia applications, Resource Reservations [17, 19] are commonly regarded as an effective tool for resource allocations which allow to control the QoS of a task independently from the others. These scheduling mechanisms enable a fine-grained control, reserving a fraction of the resource to each application. A point of crucial importance is an appropriate selection of the fraction of the total resources that each application receives. A static choice is only possible if the resource usage of the application can be characterised before its execution [20]. A possible strategy to alleviate this problem is *resource reclaiming* [9, 8, 14]. One can make a conservative allocation, and by using a reclamation strategy the unused resources are redistributed between the remaining applications. The policy used for the redistribution can be of different type (e.g., greedy [9] or weight based) but it does not take into account the current situation of the tasks. Moreover, this approach is not very useful when the system is heavily overloaded. A second thread of papers advocates the use of *resource level adaptivity*, i.e., by changing the scheduling parameters (actuators) based on the observation of the timing behaviour of the applications (sensors), which is obviously related to the QoS. The application is unaware of the adaptation mechanisms and it can be implemented using standard algorithms. A promising technique of this type is referred to as Adaptive Resource Reservations [2, 4, 3]. Contrary to other feedback schedulers [22, 16, 7], adaptive reservations allow one to control the QoS experienced by each single task, and

*This work has been partially supported by the European FRESOR FP6/2005/IST/5-034026 and IRMOS FP7/2008/ICT/214777 projects.

not only some global QoS metric (such as the total number of missed deadlines in the system). To attain this goal, an adaptive reservation considers the evolution of a single task in isolation and produces a resource request to accommodate its execution requirements. Using the mathematical model developed in [4], convergence and stability of the resulting feedback schemes are relatively easy to show for a single application [3], neglecting the interference generated by the bandwidth requests of the other tasks.

In this paper we make the point that there is a natural complementarity between adaptive reservations and resource reclaiming techniques. Indeed, the former can be used to identify online the resource needed by the tasks and to manage overload conditions, the latter can be used to correct possible over-allocations of bandwidth. As shown in the experimental section, there is an evident performance improvement in using the combination of the two techniques with respect to either of them taken in isolation. From the theoretical point of view, we show that an appropriate design can lead us to prove *system-wide* stability properties, combining the guarantees that can be offered to each single task.

2 Basic Definitions and System Model

A multimedia system is modelled as a set of real-time tasks. A real-time task τ_i is a stream of jobs $J_{i,j}$. Each job $J_{i,j}$ arrives (becomes executable) at time $r_{i,j}$, and finishes at time $f_{i,j}$ after executing for a time $c_{i,j}$. Moreover, $J_{i,j}$ is characterised by a deadline $d_{i,j}$, that is respected if $f_{i,j} \leq d_{i,j}$, and is missed if $f_{i,j} > d_{i,j}$. A task is said periodic if $r_{i,j+1} = r_{i,j} + T_i$, where T_i is the *task period*, and $d_{i,j} = r_{i,j} + T_i = r_{i,j+1}$.

In this paper, multimedia tasks are modelled as *soft* real-time tasks, for which a few deadline violations are deemed acceptable provided that the anomaly is kept in check. Reasonable performance metrics can be related to the frequency (or the probability) of a deadline miss or to the maximum deviation from the deadline (note that for a video player a deadline miss corresponds to a skipped frame).

Each task τ_i is scheduled through a CPU reservation $RSV_i = (Q_i, P_i)$. Informally speaking, the task is allowed to execute for Q_i time units every reservation period P_i . The fraction of CPU time (sometimes mentioned as “bandwidth”) reserved to the task is $B_i = Q_i/P_i$. It is important not to confuse the reservation period P_i with the task period T_i . A task can be associated to a CPU reservation even if it is not periodic at all. Many algorithms have been proposed in the literature that provide the abstraction of resource reservations, both for fixed and dynamic priority schedulers. This paper focuses on algorithms based on dynamic priorities, such as CBS [1] and GRUB [9].

It can be shown that if $\sum_i B_i \leq U^{lub}$ where $U^{lub} =$

1 for the considered class of algorithms, then the CBS or GRUB algorithm reserve a fraction B_i of the CPU time to each task τ_i regardless of the behaviour of the other tasks.

In this paper we will consider an adaptive reservation scheme, whereby the bandwidth B_i can be regarded as a controllable variable and changed for each job (hence the notation $B_{i,j}$). The execution time $c_{i,j}$ can be considered as an external (uncontrollable) input, although we record its value *a posteriori* (i.e., after the execution of the jobs) to the purpose of predicting its evolution. Feedback based adjustments are made based on the observations of a quantity $\epsilon_{i,j}$ (called *scheduling error*). This quantity has been introduced in our previous work (see [4]) and, roughly speaking, quantifies the adherence of the reserved bandwidth to the task needs. A positive value for $\epsilon_{i,j}$ means that job $J_{i,j}$ received too little and suffered a delayed termination while a negative value means that it received too much.

As shown in the cited paper, the evolution of the model can be approximated by a *fluid model*, plus a quantisation error. Namely, the fluid model is described by:

$$\epsilon_{i,j+1} = S_i(\epsilon_{i,j}) + \frac{c_{i,j+1}}{B_{i,j+1}} - T_i, \text{ with } S_i(x) = \begin{cases} x & \text{if } x \geq P_i \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Based on our definition, the scheduling error is lower bounded by $-T$ and the deadline is respected if $\epsilon_{i,j} \leq 0$.

3. Control architecture

The solution proposed in this paper is comprised of two basic elements: 1) a set of local controllers associated to each task (controllers), and 2) a compression function which is embedded into the scheduling mechanism. The purpose of the local controller is to formulate a minimum bandwidth request that allows the task to respect its timing constraints. Because the controllers have only a local visibility, they could formulate bandwidth requests exceeding U^{lub} . The compression function is then used to reduce the bandwidth requests. Additionally, if the requests of the tasks do not saturate the constraint, the compression function can also be used to redistribute the bandwidth in excess (reclaiming). The conceptual link between the two components is a minimum guaranteed bandwidth B_i^{min} that has to be granted to job $J_{i,j}$, whenever the task controller formulates a request $B_{i,j}^{req} \geq B_i^{min}$. In the rest of this section, we will describe the task controllers and show the closed loop stability properties that can be guaranteed with an appropriate minimum bandwidth B_i^{min} . From now on, whenever the discussion refers to a single task, we will drop the subscript i for notational simplicity.

The scheduling error ϵ_j can be used to quantify an upper bound of the deviation of the finishing time of job J_j from its deadline. Therefore, an ideal goal is to have $\forall j, \epsilon_j \leq 0$. However, we can allow for a small deviation from the

deadline, thus the ideal control law can be one that restrains ϵ_j inside the region $\mathcal{R}_T \triangleq [-T, \delta] \subset \mathbb{R}$, with $\delta < 0$ if we want to add robustness, or $\delta \geq 0$ if the application tolerates small deadline violations. To allow for an effective sharing of a single system between a multitude of applications, this goal should be achieved by using the minimum resource allocation that is sufficient for the purpose.

For the system described by Equation (1), such an “ideal” control law would simply be one that assigns $B_{j+1} = \frac{c_{j+1}}{T-S(\epsilon_j)}$. Unfortunately, this ideal controller is not implementable for two reasons: first, this controller can easily lead to overload conditions ($\sum_i B_i > U^{lub}$). Second, we lack an *a-priori* knowledge of the computation time c_{j+1} . The first problem is solved by the compression mechanism described in the next section.

As far as the second problem is concerned, we adopt a stochastic approach in which the evolution of the computation time c_j and of the scheduling error ϵ_j , are regarded as stochastic processes and denoted as \mathcal{C}_j and \mathcal{E}_j . A specialised *predictor* component in our architecture is used to estimate stochastic parameters on the evolution of the computation time. Such estimation may be done, for example, by appropriate filtering of the past computation times observed within a moving window [13] or by application specific predictors [21]. For our purposes, the predictor produces an upper bound \tilde{H}_{j+1} for the computation time of the next job c_{j+1} . The predictor can fail. Therefore, we can characterise its performance in terms of two parameters, p and ρ , defined as follows:

$$\forall j, \Pr \left\{ \mathcal{C}_{j+1} \leq \tilde{H}_{j+1} \right\} \geq p, \rho = \sup_j \frac{c_j}{H_j}. \quad (2)$$

The parameter p quantifies the probability of an exact prediction, whereas ρ quantifies the maximum error that the predictor can make.

The design of our controller can be specified in terms of the following goals:

1) If the scheduling error is in a region $\mathcal{R} = [-T, R]$ enclosing the target region $\mathcal{R}_T = [-T, \delta]$ ($R \geq \delta$), then it is steered to \mathcal{R}_T with at least the probability p :

$$\Pr \{ \mathcal{E}_{j+1} \in \mathcal{R}_T \mid \mathcal{E}_j \in \mathcal{R} \} \geq p. \quad (3)$$

In simpler terms, if we are sufficiently close to the target set, we will reduce the scheduling error into the target set with at least the same probability of making an exact prediction. We will refer to \mathcal{R} as *attractivity region*.

2) If the scheduling error is initially inside \mathcal{R}_T , then it cannot go beyond a certain limit and it will return into the attractivity region \mathcal{R} within a maximum number of steps. In other words, the scheduling error is allowed to occasionally leave the target set \mathcal{R}_T but the anomaly is bounded both in space and in time.

A control law that achieves the first goal can be constructed as shown in the following result:

Proposition 1 Consider the system (1) and assume that it is controlled by a feedback controller using a predictor that generates an upper bound \tilde{H}_{j+1} with a probability p as specified in Equation (2). Moreover, assume that the task is guaranteed at least a minimum bandwidth of $B^{min} \geq \frac{\sup_j \tilde{H}_j}{T+\delta-R}$, then the control law: $B^{req}(\epsilon_j) = \frac{\tilde{H}_{j+1}}{T+\delta-S(\epsilon_j)}$, fulfills the requirement in (3).

Proof. In the proposed control architecture, at the time of computation of the control law $B^{req}(\epsilon_j)$, the value ϵ_j of the stochastic variable \mathcal{E}_j is known to the controller. Therefore, it is easy to verify that:

$$\Pr \left\{ \mathcal{E}_{j+1} \leq \delta \mid \mathcal{E}_j \leq T + \delta - \frac{\tilde{H}_{j+1}}{B^{min}} \right\} \geq \Pr \left\{ \mathcal{C}_{j+1} \leq \tilde{H}_{j+1} \right\}$$

if $B_{j+1}(T + \delta - S(\epsilon_j)) \geq \tilde{H}_{j+1}$, where B_{j+1} is the bandwidth used for the next job. Such condition is verified under the assumption that the controller requests a bandwidth $B^{req}(\epsilon_j)$ as detailed in the theorem statement and that the minimum reserved bandwidth B_i^{min} is not below any such value, for $\epsilon_j \leq R$. \square

As a corollary of the theorem above, for a given B^{min} , the controller fulfils the requirement if $\epsilon_j \leq R_j \triangleq T + \delta - \frac{\tilde{H}_j}{B^{min}}$ and the maximum value of R for which the requirement can always be guaranteed is: $R^{max} = \sup_j R_j = T + \delta - \sup_j \frac{\tilde{H}_j}{B^{min}}$. We can extend the control law proposed above by saturating it to B^{min} for values of the scheduling error greater than R_j :

$$B^{req}(\epsilon_j) = \begin{cases} \frac{\tilde{H}_{j+1}}{T+\delta-S(\epsilon_j)}, & \text{for } \epsilon_j \leq R_j \\ B^{min} & \text{otherwise.} \end{cases} \quad (4)$$

With this extension, we can address the second design goal as shown in the following:

Proposition 2 Consider the system (1) and assume that it is controlled by the control law (4), using a predictor characterised by p and ρ . Let M be an integer number and P the server period. Assume that $R^{max} \geq P$ and that

$$B^{min} > \max \left\{ \sup_{j_0} \frac{1}{MT} \sum_{i=1}^M c_{j_0+i}, \sup_j \frac{\tilde{H}_j}{T+\delta} \right\}. \quad (5)$$

Define $\phi \triangleq 1 - \frac{\sup_{j_0} \frac{1}{MT} \sum_{i=1}^M c_{j_0+i}}{B^{min}} > 0$ and $L \triangleq 1 + \left\lceil \frac{\rho(T+\delta) - T - R^{max}}{MT\phi} \right\rceil M$. For any j_0 , for any $\epsilon_{j_0} \leq R_{j_0}$, the two following statements hold true:

- I) $\exists \bar{j} \in \{1, \dots, L\}$ such that $\epsilon_{j_0+\bar{j}} \leq R_{j_0+\bar{j}}$
- II) $\forall i \geq 1$ we have $\epsilon_{j_0+i} \leq \rho(T+\delta) - T + (M-1)T - MT\phi$

Proof. We start by proving the first claim. First, assuming that $-1 \leq \epsilon_{j_0} \leq R_{j_0}$, it is very easy to show that $\epsilon_{j_0+1} \leq \rho(T + \delta) - T$. Now, focus on the path followed by the system in the subsequent M samples. If for any $i \in \{1, \dots, M\}$ we have that $\epsilon_{j_0+1+i} \leq R_{j_0+1+i}$ the claim is proved. In the opposite case, the control law is always saturated to B^{min} . Because $R^{max} \geq P$, $S(\epsilon_{j_0+1+i}) = \epsilon_{j_0+1+i} \forall i \in \{1, \dots, M\}$. Therefore, we can write:

$$\begin{aligned} \epsilon_{j_0+1+M} &= \epsilon_{j_0+1} + \sum_{i=1}^M \frac{c_{j_0+1+i}}{B^{min}} - MT \leq \\ &\leq \epsilon_{j_0+1} + MTB^{min} \frac{1-\phi}{B^{min}} - MT = \epsilon_{j_0+1} - MT\phi \end{aligned}$$

where the last step is an application of the assumption on B^{min} . In other words, in a horizon of M steps the scheduling error is reduced by $MT\phi$. We can iterate the reasoning until we are able to cover the distance between $\rho(T + \delta) - T$ and R_{j_0+j} which is upper-bounded by the distance between $\rho(1 + \delta) - 1$ and R^{max} and this proves the first claim.

To prove the second claim, consider that the maximum possible scheduling error is upper-bounded by $\epsilon_{j_0+1} + \max_{h \in \{1, \dots, M\}} \{ \sum_{i=1}^h \frac{c_{j_0+1+i}}{B^{min}} - hT \}$. An upper bound to this value can be found considering the computation times as free decision variables of an optimisation problem constrained by $MTB^{min}(1 - \phi) \geq \sum_{i=1}^M c_{j_0+1+i}$. It can be easily seen that this maximum is attained for $c_{j_0+1+i} = MTB^{min}(1 - \phi)$ and $c_{j_0+1+i} = 0$ for $i > 1$, which leads to the claim. \square

The result stated above corresponds to our second design goal. Indeed, the first claim tells us that if the scheduling error goes outside of the attractivity region, it returns inside it in at most L steps, while the second claim provides a maximum bound for the possible deviation of the scheduling error from the attractivity region. Because the proposed controller bounds the probability of not violating the deadline it will be denoted as PDNV.

4. Reclaiming Mechanism

This section describes a possible implementation of the compression function based on the requirements described in Section 3. In particular, the following function will be implemented:

$$B_i = B'_i + \frac{w_i}{\sum_j w_j} \left(1 - \sum_k B'_k \right) \quad (6)$$

$B'_i = \min\{B_i^{req}, B_i^{min}\}$, B_i^{req} is the bandwidth requested by the controller, and B_i is the resulting bandwidth after the compression.

Although the compression function has been generally implemented in a dedicated software component (called *supervisor*), it can also be directly embedded in the scheduling algorithm, by using a proper reclaiming strategy, based on

GRUB. In particular, the SHRUB algorithm is used in this paper. SHRUB is a variant of the GRUB [9] algorithm, which in turn is based on the CBS [1]. In SHRUB each reservation is also assigned a positive *weight* w_i , and execution time is reclaimed based on w_i (the reclaimed time is distributed among active tasks proportionally to the reservation weights).

The main idea behind GRUB and SHRUB is that if $B_{act}(t)$ is the sum of the bandwidths of the reservations active at time t , a fraction $(1 - B_{act}(t))$ of the CPU time is not used and can be re-distributed among needing reservations. The re-distribution is performed by acting on the *accounting rule* used to keep track of the time consumed by each task. In GRUB all the reclaimed bandwidth is greedily assigned to the current executing reservation, and time is accounted to each reservation at a rate that is proportional to the current reserved bandwidth in the system. If $B_{act} < 1$, this is equivalent to temporarily increasing the maximum budget of the currently executing reservation for the current period. In the limit case of a fully utilised system, $B_{act} = 1$ and the execution time is accounted as in the CBS algorithm. In the opposite limit case of only one active reservation, time is accounted at a rate B_i (so, a time Q_i is accounted in a period P_i). SHRUB, instead, fairly distributes the unused bandwidth among all active reservations, by using the weights. In the two limit cases (fully utilised system and only one reservation), the accounting mechanism for GRUB and SHRUB work in the same way. However, when there are many reservations in the system and there is some spare bandwidth, SHRUB effectively distributes the spare bandwidth to all needing reservations in proportion to their weights. Unlike GRUB, SHRUB uses the weights to assign more spare bandwidth to reservations with higher weights, implementing Equation 6. Due to space constraints, the details of the scheduling algorithms are omitted, and interested readers can refer to [1, 9, 5].

5. Simulation Results

To show the effectiveness of the proposed solution, some simulations have been performed using an *Adaptive Reservation simulation framework* (ARSim), which simulates a set of reservations described by Equation 1 controlled through the algorithm described in Section 1, and scheduled through SHRUB.

In particular this section reports the results obtained by simulating two MPEG decoders. As far as the execution times are concerned, we used the traces measured on a real application¹ and scaled the computation times in order to produce a significant workload on the system. For task τ_1 , we computed a mean computation time $\mu_{c_1, j} = 0.42T_1$, a standard deviation $\delta_{c_1, j} = 0.059T_1$, a maximum and a

¹The decoding times are courtesy of Philips Research.

minimum computation time equal in their turn to $0.65T_1$ and $0.25T_1$. For task τ_2 we found $\mu_{c_{2,j}}q = 0.433T_2$, $\delta_{c_{1,j}} = 0.061T_2$ while the maximum and the minimum computation time were $1.11T_2$ and $0.165T_2$. An interesting parameter are also the maximum values for the moving averages, which were $0.708T_1$ for τ_1 and $0.688T_2$ for τ_2 .

The following scenarios have been considered: 1) use of the Shrub algorithm along with the PDNV control algorithm (Shrub-PDNV); 2) use of the Shrub algorithm with a static configuration (Shrub-only); 3) use of a hard reservation scheduler (without any reclaiming), with a PDNV bandwidth controller for each task: in this case, we included a bandwidth compression strategy in case of requests violating capacity (PDNV-only).

For each one of the above scenarios, we ran a set of simulations varying B_1^{min} and B_2^{min} , which correspond to the minimum guaranteed bandwidth for the Shrub-PDNV and PDNV configurations and to the assigned bandwidth for the Shrub-only configuration. We ranged B_1^{min} between 30% and 70%, increasing it by 5%, and the bandwidth B_1^{min} was chosen equal to $1 - B_2^{min}$. Among the considered configurations, the first and last ones respect Condition (5) in Proposition 2 with $N = 3$, within a good approximation, for the second task and for the first one respectively.

The PDNV controller introduced in Section 3 was used to guarantee a conditional probability of p for the scheduling error to respect the deadline. We used a very simple predictor consisting of a cascade of 12 interleaved moving averages to de-correlate the sequence and a block that estimates the 83^{rd} percentile of the error computed over a moving average of 12 samples.

Figure 1 displays the average ϵ_j^{av} (top) and the maximum ϵ_j^{max} (bottom) scheduling errors (each point in the figure is for a different choice of B_1^{min} and B_2^{min}). The points closest to the left-upper corner are those associated to bigger values for B_1^{min} , resulting in a worse performance for τ_2 . Moving toward the right lower corner, the performance of τ_1 decreases, and the performance of τ_2 improve (in the points associated to lower values for B_1^{min}). In the PDNV-only configuration the system achieves a value for ϵ_j^{av} very close to zero for all choices of the minimum guaranteed bandwidths, proving that the feedback mechanism provides an average allocation perfectly matching the task requirements. The price to be paid is that the probability of respecting the deadline is not very high (as shown below), but the maximum value of the scheduling error ϵ_j^{max} is moderate proving that the system is under control (see Figure 1 (bottom)). The Shrub-only and Shrub-PDNV configurations clearly achieved a lower value for $\epsilon_j^{(av)}$ because they reclaim the available bandwidth. Notably, the difference is not very evident on $\epsilon_j^{(av)}$ because when the task is delayed, the feedback mechanism tends to use all the available bandwidth reducing the impact of the reclaiming

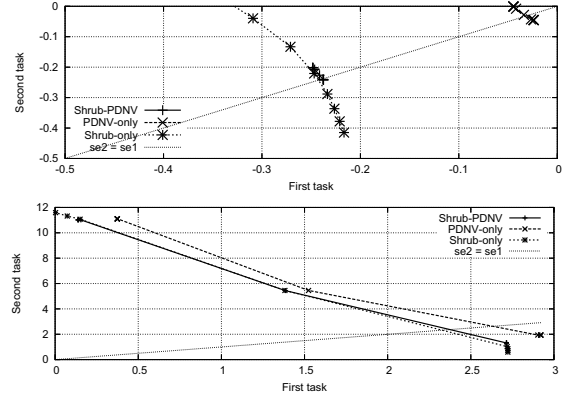


Figure 1. Average (top) and maximum (bottom) scheduling error for the two tasks.

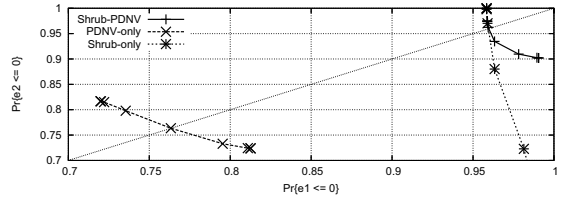


Figure 2. Probability of respecting the deadline for the two tasks.

mechanism. Note that if B^{min} is chosen according to Condition (5), then the maximum deviation of the scheduling error is below the upper bound provided by the theoretical result. Indeed, it is relatively easy to get the upper bound as in the condition, which is $4T_1$ for the first task and $8T_2$ for the second one. If the left extreme point of the curve is the one in which τ_1 receives the minimum bandwidth dictated by the theorem ($B_1^{min} = 0.7$), we get $\epsilon_1^{max} \approx 0 \leq 4$; conversely, in the right extreme the correct minimum bandwidth is assigned to τ_2 , for which $\epsilon_2^{max} \approx 2 \leq 8$.

Figure 2 reports the performance achieved by the three different configurations in terms of probability of respecting the deadline. Specifically, the x and y coordinates of each point in Figure 2 represent the experimental probability for, respectively, the first and the second task to respect the deadline. In this case, the tasks have the same weights. Therefore, the best performance is associated to points close to the upper-right corner (meaning that the two probabilities are close to 1 and equal).

The PDNV-only approach achieves an experimental probability of respecting the deadline close to the probability for the predictor to produce a correct guess. Indeed, $\Pr\{C_j \leq \tilde{H}_j\}$ is 81.3% for the first task predictor, and 81.7% for the second, while the experimental probability of respecting the deadline ranges from 74% up to 81.3% for the first task, and from 81.7% down to 73.9% for the second one. This fact shows that the results found in Section 3 on conditional probabilities (Proposition 1) are representative of the average system behavior.

Some additional experiments have been ran assigning different weights to the various multimedia tasks, but cannot be reported here due to space limits. Anyway, such experiments showed that the weights w_i can be effective in assigning different importance to the various applications during transient overloads.

6. Conclusions

This paper presents a novel approach for allocating resources to multimedia tasks, based on a combination of adaptive reservations and a reclaiming policy. Simulation and theoretical results have been proposed that show the advantages of cobining the two approaches.

In a near future, we plan to repeat some of the experiments on a real implementation the SHRUB algorithm (based on the AQuoSA framework for Linux).

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, T. Cucinotta, G. Lipari, L. Marzario, and L. Palopoli. Adaptive reservations in a linux based environment. In *Proceeding of the Real-Time Application Symposium (RTAS 04)*, Toronto (Canada), May 2004. IEEE.
- [4] 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.
- [5] S. Baruah, G. Lipari, and L. Abeni. Shrub: Shared reclamation of unused bandwidth. Technical report, Scuola Superiore Sant'Anna, July 2008. http://retis.sssup.it/~lipari/papers/shrub_tech_report_jul_08.pdf.
- [6] S. Brandt and G. Nutt. Flexible soft real-time processing in middleware. *Real-time systems journal, Special issue on Flexible scheduling in real-time systems*, 22(1-2):77–118, January-March 2002.
- [7] G. T. C. Lu, J. Stankovic and S. Son. Feedback control real-time scheduling: Framework, modeling and algorithms. *Special issue of RT Systems Journal on Control-Theoretic Approaches to Real-Time Computing*, 23(1/2), September 2002.
- [8] M. Caccamo, G. C. Buttazzo, and D. C. Thomas. Efficient reclaiming in reservation-based real-time systems with variable execution times. *IEEE Transactions on Computers*, 54(2):198–213, Feb. 2005.
- [9] G. Lipari and S. Baruah. Greedy reclamation of unused bandwidth in constant bandwidth servers. In *IEEE Proceedings of the 12th Euromicro Conference on Real-Time Systems*, Stockholm, Sweden, June 2000.
- [10] C. Hentschel, R. Bril, M. Gabrani, L. Steffens, K. van Zon, and S. van Loo. Scalable video algorithms and dynamic resource management for consumer terminals. In *Proceedings of the International Conference on Media Futures (ICMF)*, May 2001.
- [11] D. Isovich and G. Fohler. Quality aware mpeg-2 stream adaptation in resource constrained systems. In *Proceedings of the Euromicro Conference on Real-Time Systems*, Catania, Italy, 2004.
- [12] T. Lan, Y. Chen, and Z. Zhong. Mpeg2 decoding complexity regulation for a media processor. In *Fourth IEEE Workshop on Multimedia Signal Processing*, Cannes, France, 2001.
- [13] J. Liang, K. Nahrstedt, and Y. Zhou. Adaptive multi-resource prediction in distributed resource sharing environment. In *CCGRID '04: Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid*, pages 293–300, Washington, DC, USA, 2004. IEEE Computer Society.
- [14] C. Lin and S. Brandt. Improving soft real-time performance through better slack reclaiming. In *Proceedings of the Real-Time Systems Symposium*, December 2005.
- [15] C. L. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1), 1973.
- [16] 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.
- [17] C. W. Mercer, S. Savage, and H. Tokuda. Processor capacity reserves for multimedia operating systems. Technical Report CMU-CS-93-157, Carnegie Mellon University, Pittsburgh, May 1993.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] M. Roitzsch and M. Pohlack. Principles for the prediction of video decoding times applied to mpeg-1/2 and mpeg-4 part 2 video. In *Proceedings of the IEEE Real-Time Systems Symposium*, 2006.
- [22] 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.
- [23] H. Tokuda and T. Kitayama. Dynamic QoS control based on real-time threads. In *NOSSDAV'93: Proceedings of the 4th International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 114–123, London, UK, 1993. Springer-Verlag.
- [24] C. C. Wüst, L. Steffens, W. F. J. Verhaegh, R. J. Bril, and C. Hentschel. Qos control strategies for high-quality video processing. *Real-Time Systems*, 30(1-2):7–29, 2005.
- [25] C. C. Wüst and W. F. J. Verhaegh. Quality control for scalable media processing applications. *J. Scheduling*, 7(2):105–117, 2004.