Feedback scheduling for pipelines of tasks [⋆]

Tommaso Cucinotta[1], Luigi Palopoli[2]

[1] *Scuola Superiore S. Anna* author_secondname@sssup.it- *Pisa (Italy),*
[2] *University of Trento - Trento (Italy)* author_secondname@dit.unitn.it

Abstract. The problem analysed in this paper is how to effectively share a pool of resources amongst software applications consisting of pipelines of communicating tasks. The goal is to guarantee that specified Quality of Service (QoS) requirements are met. To this end, we advocate the use of a scheduling mechanism able to reserve fraction of the different resources to the competing tasks. Our work is focused on a feedback controlled adaptation of these fractions based on measurements of the QoS experienced by the application.

1 Introduction

In this paper, we consider embedded software applications consisting of multiple tasks, which run on different and networked computing nodes. Significant examples include (but are not limited to) MPEG streaming, video-surveillance and Voice-over-IP. The particular problem we deal with is how to effectively *share* resources without compromising the real-time behaviour of the applications. In fact, while resource sharing is a cost effective and flexible solution enabled by modern operating systems and middleware infrastructures, it also introduces non-deterministic scheduling delays affecting the Quality of Service of the application.

In the past few years, researchers have been confronted with the problem of constructing a real-time software infrastructure matching the temporal guarantees of a dedicated solution with the efficiency of resource sharing. A large body of results stemming from this activity has focused on predictable scheduling mechanisms [1,2,3,4]. The idea is to reserve a certain fraction of the resources to the competing tasks guaranteeing that this allocation will be respected in time (within a specified granularity). In the Resource Kernels project [3], the resource reservation approach has been successfully applied to different types of resources (including disk and network).

More recently, this technique has been complemented with adaptive mechanisms able to dynamically track the resource requirements of each task: the idea of *feedback scheduling*. In this framework, the parameters of the schedulers are used as "actuators" to adjust the QoS measured by appropriate *sensors* within acceptable bounds. In particular, Stankovic et al. [5] propose a similar mechanism

[⋆] This work has been supported by the FP6/2005/IST/5-034026 European Commission Project named FRESCOR

using the deadline of an EDF scheduler as an *actuation* mechanism. In [6,7,8], the use of a better suited scheduler enables the design of the feedback law based on a precise dynamic model of the "plant" to be controlled, thus making for a better founded application of control theory to this problem.

The results cited above do not offer a general solution to the problem of QoS management, since do not support real-time applications using multiple resources. In general, the need for different types of resources may generate undesired effects of unexpected harshness, unless the interaction of different allocation mechanisms is adequately accounted for. In particular, Rajikumar et al. [9] developed a framework (called QRAM) that decides the fraction of different resources to be allocated to the applications by solving an optimisation (NP-hard) problem. However, the resulting allocation is static and it does not allow the scheduler to accommodate applications with dynamically varying or scarcely known resource requirements. What we need is an appropriate combination of feedback control and multiple resource management.

This paper offers a first contribution in this direction. We consider applications consisting of pipelines of tasks which communicate by means of intermediate buffers, where tasks use resources of different type. For each resource, we use a reservation-based scheduler [3] that allows one to reserve a specified fraction (bandwidth) of the resource to each task using it. This technique allows to define a *dynamic model* that describes the temporal evolution of the system. In particular, we introduce a performance metric (called scheduling error) that is a good indicator for both the QoS offered by the application and its efficiency in utilising resources. These issues are discussed in Section 2. Based on this model, we define a control strategy that dynamically changes the bandwidth of each task. This technique is called *adaptive reservations* and has been proposed in previous work for a single resource [10,6]. The central contribution of this paper is to extend it to pipeline of tasks. To this end, we use a decentralised control algorithm, in which each stage of the pipeline is associated to a local *task controller* while a global supervisor enforces consistency on the total allocated bandwidth for each resource. Each task controller is based on a combination of a predictor and of a controller that counters the fluctuations of the resource requirements. Contrary to previous work on the control of pipelined applications [11], we can prove practical stability of our algorithm, based on the dynamic model of the plant. The control design is shown in Section 3. Finally in Section 4, we show simulations proving the effectiveness of the approach (an implementation of the technique in the Linux kernel is under way).

2 Task model and scheduling

We consider a set of applications $\mathcal{A}^{(1)}, \ldots, \mathcal{A}^{(L)}$ sharing a pool of resources $\mathcal{R} = \{\mathcal{R}^1, \ldots, \mathcal{R}^R\}$. Resources can be of potentially different kind (e.g., CPU, disks, network links etc). An application $\mathcal{A}^{(i)}$ consists of a pipeline of $n^{(i)}$ tasks $\mathcal{A}^{(i)} = (\tau^{(i)[1]}, \ldots, \tau^{(i)[n^{(i)}]})$, which are pairwise connected by uni-directional buffers. For notational convenience, we will henceforth omit the (i) superscript whenever the discussion refers to a single application. Each resource is allocated by a scheduler that operates using scheduling parameters decided for the different tasks.

2.1 The task model

Consider an application $\mathcal{A}$ associated to a pipeline $\tau^{[1]}, \ldots, \tau^{[n]}$. Task $\tau^{[1]}$ acquires its input from a data source, which produces data blocks (tokens) at a regular rate. Tokens are processed in sequence by each task in the pipeline and arrive to $\tau^{[n]}$ which sends the result to an output device (e.g. a screen if we are dealing with a video stream). The first task in the pipeline is periodically activated at the production period T of the data source (*time-triggered* activation). The remaining tasks of the pipeline are activated as soon as a new element is pushed into the input buffer by the task in the previous stage of the pipeline (*event-triggered* activation).

When task $\tau^{[j]}$ receives the k^{th} token, it instantiates a job $J_k^{[j]}$, which cannot start before the termination of $J_{k-1}^{[j]}$ (e.g. the decoding of a frame in a MPEG stream cannot start if the previous frame is still being decoded). The job consumes one input token and produces one output token, which is placed on the output buffer. When a job is terminated, in absence of a new token to process, the task is blocked on a read operation. On the contrary, write operations are assumed to be non-blocking. As discussed next, our adaptive scheduling solution allows us to respect this semantic with a finite number of buffers. We denote by $s_k^{[j]}$ the start time of $J_k^{[j]}$ and by $f_k^{[j]}$ its finishing time.

As a simplifying assumption, we require that each task $\tau^{(i)[j]}$ uses *only one* resource, denoted by $r^{(i)[j]}$. The resource requirement of the k^{th} job of task $\tau^{[j]}$ is denoted by $c_k^{[j]}$. Since we are interested in real-time applications, job $J_k^{(i)[j]}$is associated a deadline $d_k^{(i)[j]}$, which is a *soft* execution constraint, i.e., occasional failures are tolerated provided that the problem be kept in check. Since the activation pattern of the pipeline is periodic, it is reasonable to consider periodically spaced out deadlines: $d_{k+1}^{[j]} = d_k^{[j]} + T$. Concerning the initial values of the absolute deadlines $\left\{d_1^{[j]}\right\}$, we set $d_1^{[j]} = d_1^{[j-1]} + T$, resulting into $d_k^{[j]} \triangleq (k + j - 1)T$.

Example. Figure 1 shows an example of the activation pattern of the jobs. The bottom line reports the execution of the first task in the pipeline, whose job activations are periodically activated. For the second task in the pipeline (top line), the job $J_k^{[2]}$ cannot start until job $J_k^{[1]}$ finishes and produces the k^{th} token. On the contrary, $J_{k+1}^{[2]}$ starts right after $J_k^{[2]}$ finishes because $J_{k+1}^{[1]}$ has already terminated by that time and the $(k+1)^{th}$ token is already available.

2.2 The scheduler

The scheduler plays the role of a “plant” to be controlled. Therefore, we need an algorithm exposing *sensors* and *actuators*. Moreover, a sound design for a

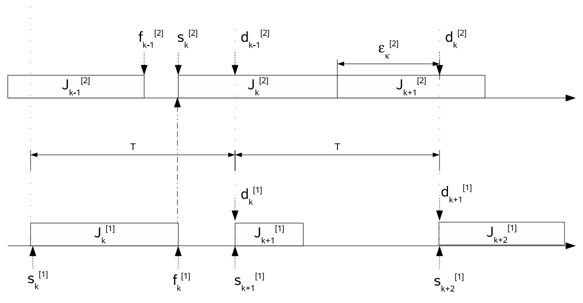


Fig. 1. Example showing the activation pattern of the jobs and the notation.

feedback scheduler has to be founded upon a realistic dynamic model relating the QoS evolution to the control choices.

To attain these goals, we *restricted* our choice to scheduling algorithms that closely approximate a *fluid allocation* of the resource (see [12]). In simple terms, it means that each task $\tau^{(i)[j]}$ executes as if using a dedicated resource whose speed is a fraction (which we call *bandwidth*) of the actual resource $r^{[j]}$.

Actuators. In our framework, the bandwidth $b_k^{(i)[j]}$ can be set for each job of $\tau^{(i)[j]}$. Therefore, it can be used as an *actuator* to control the evolution of the QoS. However, the sum of the bandwidths reserved for a resource $\mathcal{R}^r$ must never exceed its total capacity C^r:

$$\forall r \in \mathcal{R},\, \forall t \qquad \sum_{i,j,k\,:\, r^{(i)[j]}=r \,\wedge\, s_k^{(i)[j]} \le t < f_k^{(i)[j]}} b_k^{(i)[j]} \le C^r, \tag{1}$$

Sensors. We introduce the scheduling error $\epsilon_k^{(i)[j]} \triangleq f_k^{(i)[j]} - d_k^{(i)[j]}$ as a metric to quantify the violation of a deadline, which is measured by *appropriate* sensors inside the operating system. This quantity is a good indicator for both the QoS experienced by the task and its efficiency in utilising resources. Indeed, if task $\tau^{(i)[i]}$ produces an output to the end-users, a large positive scheduling error corresponds to an increased latency for the output of the output. In a real-time application, this degradation has to be coped with by by either dropping tokens or increasing the size of the buffers (which is an expensive). On the other hand, a negative value for the scheduling error is associated to an excessive allocation of the resource, as the job would have completed on-time with a smaller resource allocation as well. Therefore, it is required that the value of the scheduling error be kept as near as possible to zero.

Dynamic model: The evolution of the pipeline can be described by a discrete event model. Consider the k^{th} token produced by the data source, which determines the subsequent activations of the k^{th} jobs for all the tasks in the application pipeline $J_k^{[1]}, \ldots, J_k^{[n]}$. The state of the pipeline can be described by a vector of state variables $\boldsymbol{\epsilon}_k$, where each component is the scheduling error $\epsilon_k^{[i]}$ that the i^{th} element of the pipeline experiences when it processes the token.

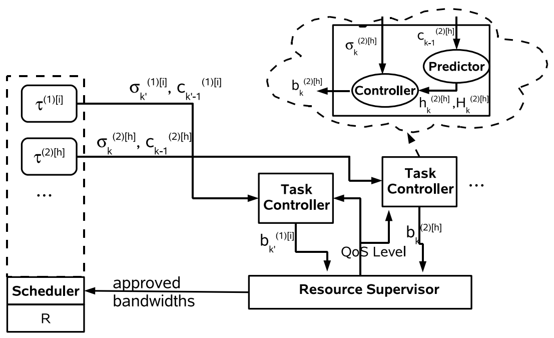


Fig. 2. The control Scheme adopted in this paper. We zoom in one task controller to show its internal structure.

Due to lack of space; in this context we omit a technical discussion on the derivation of the dynamic model (see [13] for details). For our purposes it is sufficient to say that the evolution of $\epsilon_k^{[i]}$ is given by:

$$\epsilon_k^{[j]} = \sigma_k^{[j]} + \frac{c_k^{[j]}}{b_k^{[j]}} - T, \text{ with } \sigma_k^{[j]} = \begin{cases} \max\left\{\epsilon_k^{[j-1]}, \epsilon_{k-1}^{[j]}\right\} & j \geq 2, k \geq 2 \\ \epsilon_k^{[j-1]} & j \geq 2, k = 1 \\ \max\left\{\epsilon_k^{[j-1]}, 0\right\} & j = 1, k \geq 2 \\ 0 & j = 1, k = 1 \end{cases}, \quad (2)$$

where, for notational convenience, the symbol $\sigma_k^{[j]} = s_k^{[j]} - d_{k-1}^{[j]}$ has been introduced to denote the start time of $J_k^{[j]}$ relative to the "ideal" value $d_{k-1}^{[j]}$.

3 Control design

The system described in this paper is comprised of different software applications that evolve independently and asynchronously from each other. Moreover, referring to a single application, the different components of its state vectors are asynchronously collected at different times (in the state vector $\epsilon_{\mathbf{k}}$, index k refers to a token and not to a time instant). These considerations dictate a decentralised control scheme, as shown in Figure 2, in which each resource controller consists of a supervisor and of a collection of task controllers.

There is a strong separation of concerns between the task controllers and the supervisor. Roughly speaking, the role of task controllers is to track the resource requirements of each task to maintain or recover an equilibrium condition where the QoS level is regarded as acceptable. On the contrary, the role of the supervisor is to ensure that consistency Condition (1) is never violated. To this end, the supervisor is allowed to change the working conditions of the task controllers (i.e. the QoS level of the task) to lower their bandwidth requests. An important remark regarding both the task controllers and the supervisor is that, since the entire machinery is activated at every scheduling decision, the control algorithms are bound to take a few dozens of numeric operations.

Control decisions are taken at the *start time of a new job* $J_k^{[j]}$ according to the following scheme: 1) the task controller acquires information about the state of the task (in particular $\sigma_k^{[j]}$ and the computation time of the previous job $c_{k-1}^{[j]}$); 2) it computes the new bandwidth $b_k^{[j]}$ to be used for $J_k^{[j]}$ and submits the request to the supervisor; 3) the supervisor grants the request if it does not violate Condition (1), otherwise it changes the working mode of some of the task controllers to reduce the cumulative required bandwidth. For the sake of brevity, in this paper we will restrict the focus only to the design of the task controllers and discuss their properties. For what concerns the supervisor, we will simply offer some insight into how a controlled QoS degradation (leading to diminished bandwidth requirements) can actually be obtained. For a complete description of the supervisor the reader is referred [13].

3.1 Task controllers

A task controller operates on a single task $\tau^{[j]}$ of the application $\mathcal{A}$, but the coordinated action of the different task controllers for $\mathcal{A}$ aims at attaining stability properties for the entire application. To this regard, a perfect allocation could be one where the each task experiences a null scheduling error and it receives, at each step, a bandwidth equal to $b_k^{[j]} = \frac{c_k^{[j]}}{T}$. This goal is not attainable, since it would entail predictive knowledge of $c_k^{[j]}$. A more realistic situation is one where the task controller uses a predictor able to produce, at the beginning of each job, a range $P_k^{[j]} = [h_k^{[j]}, H_k^{[j]}]$ such that $c_k^{[j]} \in P_k^{[j]}$(see Figure 2). The design of the predictor is largely application dependent (see [6,14]) and is out of the scope of the present paper. Because of the resource constraints, it is important to quantify a saturation level for the control laws, which is associated to the maximum bandwidth reserved to the tasks.

Considering this control scheme, a natural notion of practical stability is the following (see [15]):

Definition 1. *Consider the system defined by Equation* (2) *and let* $\mathcal{H}, \mathcal{G}$ *be two sets such that* $\mathcal{H} \subseteq \mathcal{G} \subseteq \mathbb{R}^{\mathrm{n}}$. *Let* $\varepsilon_{\mathbf{k}} \triangleq [\varepsilon_k^{[1]} \dots \varepsilon_k^{[n]}]$ *and* $\mathbf{P}_k \triangleq P_k^{[1]} \times \dots \times P_k^{[n]}$. *The system is said* $(\mathcal{H}, \mathcal{G})$*-stabilisable in* M *steps iff there exists a control such that:*

1. $\mathcal{H}$ *is a Robust Controlled Invariant Set (RCIS):* $\forall k_0 : \boldsymbol{\epsilon}_{k_0} \in \mathcal{H} \wedge \mathbf{c_k} \in \mathbf{P}_k \forall k > k_0$ *implies* $\boldsymbol{\epsilon}_k \in \mathcal{H} \forall k > k_0$*;*
2. $\mathcal{H}$ *is robustly attractive from* $\mathcal{G}$ *in* M *steps:* $\forall k_0 : \boldsymbol{\epsilon}_{k_0} \in \mathcal{G} \wedge \mathbf{c_k} \in \mathbf{P}_k \forall k \in [k_0, \dots k_0 + M]$ *implies* $\boldsymbol{\epsilon}_k \in \mathcal{G} \forall k \in [k_0 + 1, \dots k_0 + M] \wedge \boldsymbol{\epsilon}_{k_0+M} \in \mathcal{H}$*.*

The first property in the definition above refers to the equilibrium condition: we require that the state evolves in a small set countering the possible fluctuations of $c_k^{[j]}$ within $P_k^{[j]}$. As far as the geometry of the $\mathcal{H}$ set is concerned, we are interested in a hypercube where the state of each task is constrained in the interval $\mathcal{I} = [-e, E]$ with $e, E \in \mathbb{R}^+$, thus $\mathcal{H} = \mathcal{I}^n = \mathcal{I} \times \dots \times \mathcal{I}$. E quantifies the maximum delay a task can suffer, e quantifies the efficiency in utilising resources. Moreover, it can be shown [13] that the number of buffer elements required for intertask communications between two stages of the pipeline can be bounded by $\lfloor \frac{e+E}{T} \rfloor + 1$; therefore, with this number of elements, it never happens that a task finds a full buffer in its write operation.

The second property in Definition 1 relates to the evolution of the state when it is initially outside of $\mathcal{I}^n$. This situation occurs as a result of a perturbation such as a temporary system overload which prevents the task controller to use all the bandwidth it needs. After a perturbation of the equilibrium has terminated, k_0 is the first job entering the first task of the pipeline. Generally, the scheduling errors experienced by the task instances $k_0 - 1$ on all the pipeline stages $\left\{ J^{[j]}_{k_0 - 1} \right\}_{j=1,2,\ldots}$ deviate from $\mathcal{I} = [-e,\, E]$ and are in the set $\mathcal{J} =[-e,\, L]$, with $e, L \in \mathbb{R}^+$ and $L \geq E$. In this case, L quantifies the maximum delay that tasks in the pipeline will suffer. The definition requires that we are able to reduce the state from $\mathcal{G} = \mathcal{J}^n$ to $\mathcal{H} = \mathcal{I}^n$ in at most M steps. The requirements of a fixed number of steps (as opposed to an asymptotic definition) makes the property of practical interest for system design.

In the sequel, we will separately look at control schemes that attain Robustly Controlled Invariance and Robust attractivity.

Maintaining the equilibrium The following offers necessary and sufficient conditions for the existence of a RCIS, and describes a family of control laws attaining it.

Theorem 1. *A control law attaining robust controlled invariance of $\mathcal{I}^N$ with $\mathcal{I} =[-e,\, E]$ exists iff the following conditions hold*

$$\tilde{H} \leq T \wedge \begin{cases} e + \alpha^{[1]} E \geq T \left(1 - \alpha^{[1]}\right) \\ e + E \geq T \left(\frac{1-\alpha}{\alpha}\right), \end{cases} \tag{3}$$

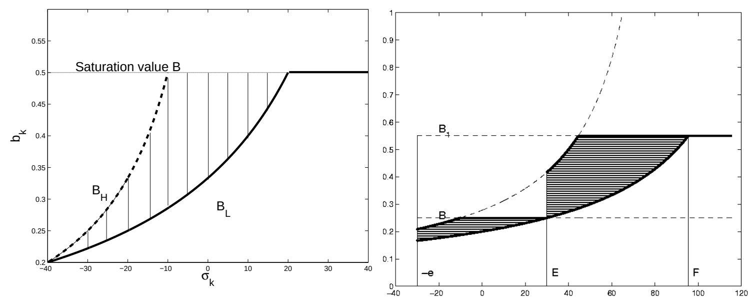
where $\tilde{H} \triangleq \max_j \left\{ \frac{\sup_k \{H^{[j]}_k\}}{B^{[j]}} \right\}$, $\alpha^{[1]} \triangleq \inf_k \left\{ \alpha^{[1]}_k \right\}$, $\alpha \triangleq \min_{j \geq 2} \left\{ \inf_k \left\{ \alpha^{[j]}_k \right\} \right\}$, *with* $\alpha^{[j]}_k \triangleq \frac{h^{[j]}_k}{H^{[j]}_k}$. *Furthermore, the control laws meeting the goal requirements have to be chosen in the range* $b^{[j]}_k \in [\mathcal{B}^{[j]}_L(\sigma^{[j]}_k), \mathcal{B}^{[j]}_H(\sigma^{[j]}_k)]$, *where:*

$$\mathcal{B}^{[j]}_L(\sigma) = \frac{H^{[j]}_k}{T+E-\sigma^{[j]}_k},\ \mathcal{B}^{[j]}_H(\sigma) = \min \left\{ \frac{h^{[j]}_k}{T - e - \sigma^{[j]}_k}, B^{[j]} \right\} \tag{4}$$

Proof. See Appendix.

The following remarks are very useful in the design of the supervisor.

Remark 1. Condition (4) identifies a family of controllers attaining the control goal (See Figure 3.(A)). The choice of one element of this family depends on the



(A) (B)

Fig. 3. (A) Family of control laws ensuring controlled invariance; (B) Family of control laws used to restore the equilibrium.

different trade-offs sought in the application. For instance, the lower bound $\mathcal{B}_L^{[j]}$ is clearly the most conservative in terms of used bandwidth. On the contrary, picking a control value closer to the upper bound $\mathcal{B}_H^{[j]}$ is the most robust choice for the QoS against possible un-modelled effects. Moreover, it is easy to show that robust controlled invariance of $\mathcal{I}$ is preserved if we change the value of the bandwidth during the job, as long as its value always belongs to the $[\mathcal{B}_L^{[j]}(\sigma), \mathcal{B}_H^{[j]}(\sigma)]$ range. Therefore, the supervisor is allowed to change the bandwidth allocated to the task up to $\mathcal{B}_L^{[j]}$ without compromising the invariance of $\mathcal{I}^N$.

It is possible for a task to specify different values for the upper-bound E of the RCIS, which correspond to different QoS levels. In a normal situation, the task aims at the maximum QoS level, but it can switch to a degraded QoS (i.e., a larger value of E) if an overload occurs. Indeed, by doing so, its minimum bandwidth requirement $\mathcal{B}_L^{[j]}(\sigma)$ decreases. This type of degradation is another degree of freedom exposed to the supervisor to manage overload situations.

Restoring the equilibrium A control law that attains $(\mathcal{J}^n,\mathcal{I}^n)-$attractivity in M steps can be built using, for each task controller j, two saturation levels $B'^{[j]}$ and $B^{[j]}$ with $B'^{[j]} > B^{[j]}$. $B'^{[j]}$is used to recover the equilibrium and $B^{[j]}$ to maintain it. Let $F_k^{[j]}$ be a number such that $\frac{H_k^{[j]}}{T+E-F_k^{[j]}} = B'^{[j]}$. A control strategy is built as follows:

$$\begin{aligned}
&b_k^{[j]} = B'^{[j]} \,\mathrm{if}\, \sigma_k^{[j]} \geq F_k^{[j]},\\
&b_k^{[j]} \in [\mathcal{B}_L^{[j]}(\sigma), \min B'^{[j]}, \frac{h_k^{[j]}}{T-e-\sigma_k^{[j]}}], \,\mathrm{if}\, E \leq \sigma_k^{[j]} \leq F_k^{[j]},\\
&b_k^{[j]} \in [\mathcal{B}_L^{[j]}(\sigma_k^{[i]}), \mathcal{B}_H^{[j]}(\sigma_k^{[i]})], \mathrm{if} -e \leq \sigma_k^{[j]} \leq E.
\end{aligned} \tag{5}$$

The rational is very simple. When we are far off from the target equilibrium, we use the maximum available bandwidth B' to quickly reduce the scheduling error. Then we reach a zone ($\sigma_k \in [E, F]$) where we can reach the equilibrium in one step. Finally, when we are inside the target we can switch to the control laws that allow us to maintain the equilibrium (which requires a lower saturation value). The family of control laws constructed in this way are those in the striped

Predictor	$E[\alpha_k^{[1]}]$	$\Pr\{\epsilon_k^{[1]} \in \mathcal{I}\}$	$E[b_k^{[1]}]$	*Predictor*	$E[\alpha_k^{[2]}]$	$\Pr\{\epsilon_k^{[2]} \in \mathcal{I}\}$	$E[b_k^{[2]}]$
Fix. 0.15/0.70	-	*15.71%*	=	*Fix. 0.15/0.70*	-	*25.34%*	=
Fix. 0.17/0.75	-	*23.41%*	=	*Fix. 0.17/0.75*	-	*57.22%*	=
MMA[3,3]	*0.767*	*72.42%*	*16.9%*	*MMA[3,3]*	*0.900*	*89.78%*	*73.7%*
MMA[12,3]	*0.796*	*86.31%*	*15.4%*	*MMA[12,3]*	*0.916*	*89.31%*	*73.5%*

Table 1. Experimental probability for a scheduling error in the target RCIS.

area delimited by the tick lines in Figure 3.(B). The effectiveness of this control policy is shown in the next Theorem.

Theorem 2. *Under the assumption of Theorem 1, the family of control laws in Equation* (5) *attains* $(\mathcal{J}^n, \mathcal{I}^n)-$*attractivity (with* $\mathcal{J} = [-e, L]$ *and* $\mathcal{I} = [-e, E]$*) in* $M + 1$ *steps for the system in Equation* (2) *if* $\hat{H} \leq \frac{T+E-L+MT}{M+1}$, *where* $\hat{H} = \max_j \left\{ \frac{\sup_{k>k_0}\{H_k^{[j]}\}}{B'^{[j]}} \right\} \leq \tilde{H}$.

Proof. See Appendix.

Remark 2. As one would expect, the saturation level required for attractivity is higher than the one required for mere controlled invariance. Indeed, the two values coincide if $M \to \infty$.

4 Experimental Results

We applied the control techniques shown so far to an MPEG-2 decoder, whose behaviour has been simulated by using execution traces measured from a real application running on Linux. The application consists of a pipeline of two tasks. The first task loads the frames from the disk; therefore its resource requirements are proportional to the frame size. The second task decodes the buffered frames; the resource requirements are given, in this case, by the decoding times measured from the real application.

We ran the simulations experiment considering different scenarios. In the first scenario, we used a fixed bandwidth allocation, in which the bandwidth chosen for the two stages were slightly above the mean value of the resource requirements. In the second scenario, we considered a fixed allocation with a greater bandwidth value for the two stages. In the third and in the fourth scenario, we used our control scheme with different predictors, based on multiple moving averages (MMA(3,3), and MMA(12,3), meaning respectively 3 and 12 independent averages of 3 samples). The use of these predictors is motivated by the periodic coding scheme used for the considered MPEG2 stream [6]. The mean time required to decode the frame was $\mu_{c^{[2]}} = 26.85ms$. The mean time required to load a frame for the disk and its standard deviation were respectively $\mu_{c^{[1]}} = 4.941ms$. For both tasks we specified a desired RCIS $[-e, E] = [-16ms, 16ms]$. The application has a period of $T = 40ms$ (corresponding to 25 frames-per-second stream).

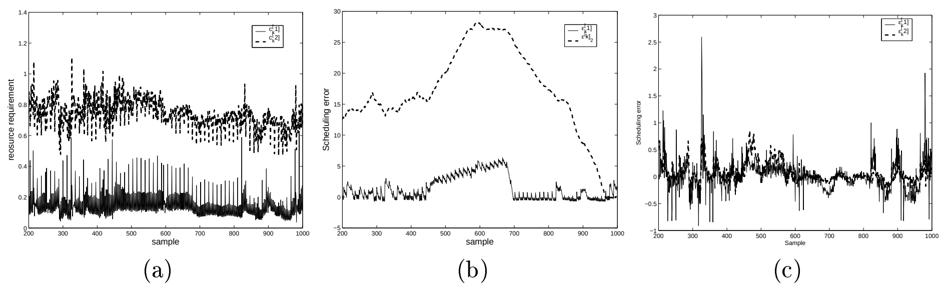


Fig. 4. Segments of execution traces. (a) Resource requirements, (b) Evolution of the scheduling error in the case of fixed bandwidth (0.17/0.75). (c) Evolution in case of feedback control with MA[12,3] predictor.

Figure 4.(a) shows the temporal evolution of the resource requirements in the two stages for 800 frames (as a percentage of the task period). In this segment, the $c_k^{[2]}$ value is often above the average. Moreover, $c_k^{[1]}$ displays several peaks above the average. As a result, if we look at the scheduling error for fixed bandwidth (Figure 4.(b)), we can experience large delays on the second stage (which suffers for both the peaks of $c_k^{[1]}$and of $c_k^{[2]}$). The feedback scheme, although based on a predictor that occasionally fails, is able to compensate this effect. If we want to evaluate the system performance on the entire stream, it is useful to compute the experimental probability of having a scheduling error inside the target RCIS. This information is reported in Table 1, where the first column reports the mean value of the $\alpha_k^{[i]}$ parameter (i.e. $mean \frac{h_k^{[i]}}{H_k^{[i]}}$) produced by the predictors for the two stages. This is related to the quality of the prediction and to the variability of the trace (a lower value corresponds to a higher variability). The last column reports the average bandwidth allocated by the controllers. The improvement achieved by using feedback control is evident. The last column of the table also highlights that the average bandwidth allocated by the controller is very close to the average value of the computation requirements.

5 Conclusions and future work

In this paper, we have shown the application of a feedback controller to the problem of dynamical allocation of resources to a time-sensitive application consisting of a pipeline of tasks. The use of a scheduling mechanism that approximates a fluid partitioning of the resources enables the definition of precise dynamic model for the system, which can be used in the design of a the feedback controller providing guarantees on its closed loop performance. Most of the future work will be concentrated in evaluating different strategies for the supervisor and on the application of stochastic control techniques to the design of the feedback scheduler.

References

1. Stoica, I., Abdel-Wahab, H., Jeffay, K., Baruah, S.K., Gehrke, J.E., Plaxton, C.G.: A proportional share resource allocation algorithm for real-time, time-shared systems. In: Proceedings of the IEEE Real-Time Systems Symposium. (1996)
2. Mercer, C.W., Rajkumar, R., Tokuda, H.: Applying hard real-time technology to multimedia systems. In: Workshop on the Role of Real-Time in Multimedia/Interactive Computing System. (1993)
3. Rajkumar, R., Juvva, K., Molano, A., Oikawa, S.: Resource kernels: A resource-centric approach to real-time and multimedia systems. In: Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking. (1998)
4. Abeni, L., Buttazzo, G.: Integrating multimedia applications in hard real-time systems. In: Proceedings of the IEEE Real-Time Systems Symposium, Madrid, Spain (1998)
5. C. Lu, J. Stankovic, G.T., Son, S.: Feedback control real-time scheduling: Framework, modeling and algorithms. Ppecial issue of RT Systems Journal on Control-Theoretic Approaches to Real-Time Computing **23**(1/2) (2002)
6. Abeni, L., Cucinotta, T., Lipari, G., Marzario, L., Palopoli, L.: Adaptive reservations in a linux based environment. In: Proceeding of the Real-Time Application Symposium (RTAS 04), Toronto (Canada), IEEE (2004)
7. Goel, A., Walpole, J., Shor, M.: Real-rate scheduling. In: Proc.of rtas04. (2004) 434
8. Eide, E., Stack, T., Regehr, J., Lepreau, J.: Dynamic cpu management for real-time, middleware-based systems. In: Proc. of 10th IEEE Real-Time and Embedded Technology and Applications Symposium, Toronto, Canada (2004)
9. Rajkumar, R., Lee, C., Lehoczky, J.P., Siewiorek, D.P.: Practical solutions for QoS-based resource allocation. In: RTSS. (1998) 296–306
10. Palopoli, L., Cucinotta, T., Bicchi, A.: Quality of service control in soft real-time applications. In: Proc. of the IEEE 2003 conference on decision and control (CDC02), Maui, Hawai, USA (2003)
11. Steere, D., Shor, M.H., Goel, A., Walpole, J., P, C.: Control and modeling issues in computer operating systems: Resource management for real-rate computer applications. In: Proceedings of 39th IEEE Conference on Decision and Control (CDC00). (2000)
12. Abeni, L., Palopoli, L., Lipari, G., Walpole, J.: Analysis of a reservation-based feedback scheduler. In: Proc. of the Real-Time Systems Symposium, Austin, Texas (2002)
13. Luigi Palopoli, T.C.: Feeback scheduling for pipelines of tasks. Technical report, Uiversity of Trento (2006)
14. Calafiore, G., Camp, M.: Interval predictors for unknown dynamical systems: an assessment of reliability. In: 41st IEEE Conference on Decision and Control (cdc02), 2002 (2002)
15. Blanchini, F.: Set invariance in control. Automatica (1999)

A Proofs of the stability theorems

Robust controlled invariance. In order to show Theorem 1, we need some preliminary Lemmas.

Lemma 1. *Consider the job $J_k^{[j]}$ of task $\tau^{[j]}$ with $j \geq 2$. A control law exists guaranteeing that $\epsilon_k^{[j]} \in \mathcal{I}$, $\forall \sigma_k^{[j]} \in \mathcal{I}$, $\forall c_k^{[j]} \in [h_k^{[j]}, H_k^{[j]}]$ if and only if $\frac{H_k^{[j]}}{T} \leq B^{[j]} \wedge e + E \geq T\left(\frac{1-\alpha_k^{[j]}}{\alpha_k^{[j]}}\right)$. Moreover, the bandwidth guaranteeing such property has to be chosen in the range identified by Equation* (4).

Proof. For the sake of brevity, the proof is given only for the case $e + E < T$. Consider job $J_k^{[j]}$, with $k \geq 2$, of $\tau^{[j]}$, with $j \geq 2$. Under the stated hypotheses $\sigma_k^{[j]} \in [-e, E]$, in view of Equation (2), we can have $\epsilon_k^{[j]} \in [-e, E]$ iff $T - e - \sigma_k^{[j]} \leq \frac{c_k^{[j]}}{b_k^{[j]}} \leq T + E - \sigma_k^{[j]}$. Since $e + E < T$, it is possible to re-write the condition as: $\frac{c_k^{[j]}}{T+E-\sigma_k^{[j]}} \leq b_k^{[j]} \leq \frac{c_k^{[j]}}{T-e-\sigma_k^{[j]}}$. As $c_k^{[j]}$ is not known, and it may take any value in the range $[h_k^{[j]}, H_k^{[j]}]$, the only possibility for the controller not to violate last equation is the choice of a bandwidth value belonging to the intersection of all the ranges corresponding to any possible value of $c_k^{[j]}$, i.e. $\frac{H_k^{[j]}}{T+E-\sigma_k^{[j]}} \leq b_k^{[j]} \leq \frac{h_k^{[j]}}{T-e-\sigma_k^{[j]}}$. Such a choice exists if and only if $e + \alpha_k^{[j]} E \geq (1 - \alpha_k^{[j]})(T - \sigma_k^{[j]})$. The latter condition must hold for any possible value of the start time $\sigma_k^{[j]} \in [-e, E]$, leading to $e + E \geq T\left(\frac{1-\alpha_k^{[j]}}{\alpha_k^{[j]}}\right)$. Furthermore, the chosen control value must be legal, i.e. $b_k^{[j]} \leq B^{[j]}$. This is possible if and only if $\frac{H_k^{[j]}}{T+E-\sigma_k^{[j]}} \leq B^{[j]}$. This must hold for each possible value of $\sigma_k^{[i]} \leq E$, which leads to the existence condition for the controller: $\frac{H_k^{[j]}}{B^{[j]}} \leq T$.

With simular arguments, we can prove the following for the first stage.

Lemma 2. *Consider job $J_k^{[1]}$ of task $\tau_k^{[1]}$ with $k \geq 2$. A control law exists guaranteeing that $\epsilon_k^{[1]} \in \mathcal{I}$, $\forall \sigma_k^{[j]} \in [0, E]$, $\forall c_k^{[1]} \in [h_k^{[1]}, H_k^{[1]}]$, if and only if $\frac{H_k^{[1]}}{T} \leq B^{[1]}$ and $e + \alpha_k^{[1]} E \geq T\left(1 - \alpha_k^{[1]}\right)$.*

Proof. (of Theorem 1). First we focus on sufficiency of the theorem condition, which may be proved by proving the following statement $S(k)$ inductively on $k > k_0$:

$$\begin{cases} \max_{k \in]k_0, k]} \max_j \left\{\frac{H_k^{[j]}}{B^{[j]}}\right\} \leq T \\ e + \alpha_k^{[1]} E \geq T\left(1 - \alpha_k^{[1]}\right) \\ e + E \geq T\left(\frac{1-\alpha_{k_0}}{\alpha_{k_0}}\right) \; j \geq 2 \end{cases} \quad \text{implies} \begin{cases} \varepsilon_{\mathbf{k_0}} \in \mathcal{H} \\ \mathbf{c_h} \in \mathbf{P}_h \forall h \in]k_0, k] \end{cases} \quad \Longrightarrow \varepsilon_{\mathbf{k+1}} \in \mathcal{H}.$$

Pick a value for k_0 and consider the base inductive case $k = k_0$. Assume $\varepsilon_{\mathbf{k_0}} \in \mathcal{H} \wedge \mathbf{c_{k+1}} \in \mathbf{P}_{k+1}$. For Lemma 2, $\frac{H_{k_0}^{[1]}}{T} \leq B^{[1]}$ and $e + \alpha_{k_0}^{[1]} E \geq T\left(1 - \alpha_{k_0}^{[1]}\right)$

guarantees that $\varepsilon^{[1]}_{k_0+1} \in \mathcal{I}$. Furthermore, for Lemma 1, $\frac{H^{[j]}_{k_0}}{T} \le B^{[j]}$ and $e + E \ge T\left(\frac{1-\alpha^{[j]}_{k_0}}{\alpha^{[j]}_{k_0}}\right)$ guarantees that $\varepsilon^{[j]}_{k_0+1} \in [-e, E]$. Therefore, $\varepsilon_{\mathbf{k_0+1}} \in \mathcal{H}$ is derived from the intersection of all these conditions, i.e. the left-hand side of $S(k_0)$.

For a generic k, assume $S(k-1)$ holds, $\varepsilon_{\mathbf{k_0}} \in \mathcal{H}$ and $\mathbf{c_h} \in \mathbf{P}_h \forall h \in]k_0, k]$. Then, under the conditions of the left-hand side of $S(k-1)$, we have: $\varepsilon_{\mathbf{k}} \in \mathcal{H}$. For the two cited lemmas, $\varepsilon_{\mathbf{k+1}} \in \mathcal{H}$ is thus derived from the intersection of the conditions $\frac{H^{[1]}_k}{T} \le B^{[1]}$, $e+\alpha^{[1]}_k E \ge T\left(1-\alpha^{[1]}_k\right)$, $\frac{H^{[j]}_k}{T} \le B^{[j]}$ and $e+E \ge T\left(\frac{1-\alpha^{[j]}_k}{\alpha^{[j]}_k}\right)$, plus the ones in the left-hand side of $S(k-1)$. These constitute exactly the left-hand side of $S(k)$. Therefore, $S(k)$ is true for any $k > k_0$. Sufficiency of the theorem condition is obtained by observing that it is obtained as $\lim_{k\to+\infty} S(k)$.

Concerning necessity of the theorem condition, we have to observe that in the definition of RCIS (1) the stated property is required to hold for each k_0. This means that, if we consider only a single evolution step of the state vector from k_0 to k_0+1, Lemma 2 and 1 require $\frac{H^{[1]}_{k_0}}{T} \le B^{[1]}$, $e+\alpha^{[1]}_{k_0} E \ge T\left(1-\alpha^{[1]}_{k_0}\right)$, $\frac{H^{[j]}_{k_0}}{T} \le B^{[j]}$ and $e+E \ge T\left(\frac{1-\alpha^{[j]}_{k_0}}{\alpha^{[j]}_{k_0}}\right)$ as necessary conditions to guarantee that $\varepsilon_{\mathbf{k_0+1}} \in \mathcal{H}$. Therefore, by considering any possible value for k_0, we obtain that all of these conditions (at varying k_0) must hold true, leading to the theorem proof.

$(\mathcal{H}, \mathcal{G})$ **attractivity.** We recall that $F^{[j]}_k$ has been defined as a number such that $\frac{H^{[j]}_k}{T+E-F^{[j]}_k} = B'^{[j]}$. Before showing Theorem 2, we need the following.

Lemma 3. *Under the assumptions of Theorem 2, any control law chosen in the family in Equation* (5) *guarantees that: 1) if* $\sigma^{[j]}_k \ge -e$*, then* $\epsilon^{[j]}_k \ge -e$ *, 2) if* $\sigma^{[j]}_k \in [-e, E]$*, then* $\epsilon^{[j]}_k \in [-e, E]$*, 3) if* $\sigma^{[j]}_k \in [E, F^{[j]}_k]$*, then* $\epsilon^{[j]}_k \in [-e, E]$.

Proof. The Lemma immediately derives from the way the family of controllers is constructed and from the evolution of the system, condensed in Equation (2). For instance, to violate the first claim, we should choose a value for $b^{[1]}_k \ge \frac{h^{[1]}_k}{T-e-\sigma^{[j]}_k}$, which is never done as shown in Figure 3.B. Similar arguments can be used for the other claims.

Proof. (of Theorem 2) Consider k_0 as defined above. As a preliminary step, we prove the following property:

$$\forall j,\, \forall n \in [1,\, M+1],\, -e \le \epsilon^{[j]}_{k_0+n} \le \max\left\{L + n(\hat{H} - T),\, E\right\}. \qquad (6)$$

The proof is by induction on the stages of the pipeline. Let's focus on the first stage. In this case, $\sigma^{[1]}_{k_0+n} = \max\{0,\, \epsilon^{[1]}_{k_0+n}\}$. In view of the first property in Lemma 3 and of the evolution of the system in Equation (2), $\epsilon^{[1]}_{k_0+n} \ge -e$ is easily verified by induction on n. As far as the upper bound below E is concerned, we observe that, if for any $k \in [k_0, k_{0+n}]$ $\epsilon_k^{[1]} \leq F_k^{[1]}$, the claim is an immediate result of the second and third claims in Lemma 3. On the contrary, assuming that for all $k \in [k_0, k_{0+n}]$ $\epsilon_k^{[1]} > F_k^{[1]}$, we use the saturation value $B'^{[1]}$ as a control value; therefore we can write $\epsilon_{k_0+n}^{[1]} \leq \epsilon_{k_0}^{[1]} + \frac{\sum_{h=k_0+1}^{k_0+n} H_h^{[1]}}{B'^{[1]}} - nT \leq L + \frac{\hat{H} n B'^{[1]}}{B'^{[1]}} - nT = L + n(\hat{H} - T)$.

Now, let's consider stage j of the pipeline assuming that the property holds for stage $j-1$. In this case, $\sigma_k^{[j]} = \max\{\epsilon_k^{[j-1]}, \epsilon_{k-1}^{[j]}\}$.By inductive hypotheses, $\epsilon_k^{[j-1]} \geq -e$, which leads to $\sigma_k^{[j]} \geq -e$. Hence, in view of the first property of Lemma 3, we have $\epsilon_k^{[j]} \geq -e$.

As far as the upper bound is concerned, the dynamic evolution is described in Equation (2) and it can lead to one of the following cases:

$$\epsilon_{k_0+n}^{[j]} = \begin{cases} \epsilon_{k_0}^{[j]} + \sum_{h=k_0+1}^{k_0+n} \frac{c_h^{[j]}}{B'^{[j]}} - nT & \text{if } \epsilon_{k-1}^{[j]} > \epsilon_k^{[j-1]} \, \forall k \in [k_0+1, , k_0+n] \\ \epsilon_{k_0+1}^{[j-1]} + \sum_{h=k_0+1}^{k_0+n} \frac{c_h^{[j]}}{B'^{[j]}} - nT & \text{if } \epsilon_{k-1}^{[j]} > \epsilon_k^{[j-1]} \, \forall k \in [k_0+2, k_0+n] \wedge \epsilon_{k_0+1}^{[j-1]} > \epsilon_{k_0}^{[j]} \\ \cdots & \\ \epsilon_{k_0+n}^{[j-1]} + \frac{c_{k_0+n}^{[j]}}{B'^{[j]}} - T & \text{if } \epsilon_{k_0+n}^{[j-1]} > \epsilon_{k_0+n-1}^{[j]} \end{cases}$$

The first case is dealt with exactly as shown for the previous stage of the pipeline and it leads to Equation (6). Consider the generic case $\epsilon_{k_0+n}^{[j]} = \epsilon_{k_0+m}^{[j-1]} + \sum_{h=k_0+m}^{k_0+n} \frac{c_h^{[j]}}{B'^{[j]}} - (n-m+1)T$, with $m \geq 1$. The application of the inductive hypothesis $\epsilon_{k_0+m}^{[j-1]} \leq \max\left\{L + m(\hat{H} - T), E\right\} \equiv \rho(m)$ leads us to $\epsilon_{k_0+n}^{[j]} \leq \rho + \sum_{h=k_0+m}^{k_0+n} \frac{c_h^{[j]}}{B'^{[j]}} - (n-m+1)T \leq L + (n+1)(\hat{H} - T) \leq L + n(\hat{H} - T)$.

If $\rho = L + m(\hat{H} - T)$, then we can write $\epsilon_{k_0+n}^{[j]} \leq L + (n+1)(\hat{H} - T) \leq L + n(\hat{H} - T)$. If $\rho = E$, then we can write $\epsilon_{k_0+n}^{[j]} \leq E + (n-m+1)(\hat{H} - T) \leq E$ (since $\hat{H} \leq \tilde{H} \leq T$ by hypotheses). This terminates the proof of property in Equation (6).

As a consequence of Lemma 3, the claim of the theorem is proved if: 1) $\sigma_k^{[j]} \geq -e$ for $k = k_0+1, \ldots, k_0+M+1$ and 2) $\sigma_{k_0+M+1}^{[j]} \leq F_{k_0+M+1}^{[j]}$. Condition 1) is part of Equation (6). Because of Equation (6), we can also write (considering the only relevant case with $L + M(\hat{H} - T) > E$) : $\sigma_{k_0+M+1}^{[j]} = \max\{\varepsilon_{k_0+M+1}^{[j-1]}, \varepsilon_{k_0+M}^{[j]}\} \leq L + M(\hat{H} - T)$. On the other hand, $F_k^{[j]} = T + E - \frac{H_k^{[j]}}{B'^{[j]}} \geq T + E - \hat{H}$, thus condition 2) is satisfied if:$L + M(\hat{H} - T) \leq T + E - \hat{H}$, which can be written as $\hat{H} \leq \frac{T+E-L+MT}{M+1}$, which completes the proof.