2011 17th IEEE Real-Time and Embedded Technology and Applications Symposium

Multi-Moded Resource Reservations

Luca Santinelli[1,2], Giorgio Buttazzo[1] Enrico Bini[1]

[1]*Scuola Superiore Sant'Anna, Pisa, Italy; email:* {name.surname}@sssup.it
[2]*INRIA Nancy Grand Est, Nancy, France; email:* luca.santinelli@inria.fr

***Abstract*—Often real-time systems can run in different modes depending on the external environment or their internal state. Each operational mode is characterized by a set of tasks with different computational demand, resource requirements, and resource availability. When resource reservation is used to achieve temporal isolation among applications, the reservation parameters may need to change from mode to mode. Hence, an additional guarantee is required to ensure feasibility not only of the applications, but also of the reservations.**

This paper presents a schedulability analysis to predict the timing behavior of a multi-moded resource reservation, whose parameters may change due to a mode transition. Resource provisioning is analyzed in all the operational modes and also during mode-changes in order to guarantee a minimum amount of resources and derive a feasibility condition for real-time applications and reservations. Theoretical results are also illustrated with examples and test cases.

I. Introduction

Nowadays, real-time systems are becoming highly dynamic, running applications that can adapt their behavior at run-time by changing their functionality. Often, real-time systems are characterized by different operational modes, designed to achieve different functionalities or to respond to changes in the environment. Each mode specifies functional and non-functional characteristics and consists of specific computational demands, resource requirements and resource availabilities.

As a consequence, the overall computational load and the allocated resources may change over time depending on the operational mode selected for the system. For example, adding a new task into the system at run-time may result in a reduction of the computational resources allocated to the other tasks. Other examples comes form the energy consumption policy where in order to save energy, the system may be required to discard some of its functionalities and redistribute the resources among its components at run-time.

A change in the system state, e.g., from start-up to normal, or from normal to energy saving, or from normal to shut-down, may also require re-allocating the computational resources among the tasks composing the application. Changing one or more parameters in the system components at run-time must also be considered a *mode change*, because it affects the system load and hence modifies the timing behavior of the application and the system itself.

Multi-moded real-time systems require a more accurate analysis with respect to classical single-mode systems, because of the criticality of mode transitions. In fact, there are situations in which, although timing constraints can be guaranteed to be met within each individual mode in steady state conditions, deadlines can still be missed during mode transitions. It is therefore essential to analyze the system in order to guarantee feasibility not only within each mode, but also across modes.

If a resource reservation approach [1], [2] is adopted in the system to achieve temporal isolation between different application components, then a mode transition may also require changing reservation parameters, e.g., to re-distribute the available resources whenever a new component is dynamically activated. In this situation, achieving predictability means not only providing guarantee *between components* (i.e., at the reservation level) before, after, and during mode transitions, but also *within each component*, before, after, and during mode transitions.

Whereas resource reservation manages applications by supplying the resource they require, adaptive applications must rely on adaptive resource reservations to meet their changing resource requirements. Changing a reservation means changing the corresponding resource reservation parameters to adapt the resource provisioning to the new requirements demanded by the system and its applications. Consequently, the reservation passes from one mode to the new one with a consequent mode transition stage. The resource reconfigurations need to be performed online without jeopardizing schedulability. It is therefore essential to develop appropriate resource reconfiguration criteria and algorithms to manage the criticality of the transition stage.

A. Related Work

The problem of timing analysis across mode changes has been addressed in the real-time literature under different assumptions and system models [3], [4], [5], [6]. For instance, Fohler [7] investigated the problem of mode changes in the context of pre run-time scheduled hard real-time systems, where a table-driven schedule is constructed for each operational mode and an appropriate time must be selected to

This research has been supported by the European Commission under the ArtistDesign Network of Excellence (214373) and the PREDATOR project (FP7/2008/ICT/216008).

1080-1812/11 $26.00 © 2011 IEEE
DOI 10.1109/RTAS.2011.12

start a new mode and avoid deadline misses. Crespo et al. [8] presented a survey of mode change protocols for uniprocessor systems under fixed-priority scheduling and proposed a new protocol along with their own schedulability analysis. Guangming [9] computed the earliest time at which a new task can be safely added to the system under the Earliest Deadline First (EDF) scheduling, without jeopardizing the feasibility of the task set. The underline idea behind such solutions is to wait for a certain amount of time before changing the schedule, identifying a safe time instant where the new mode can be activated without causing deadline misses.

A considerable amount of work has been recently addressed to the analysis of resource reservation mechanisms for achieving temporal protection in real-time systems. The concept of reservations was originally introduced by Mercer, Savage, and Tokuda [10], and later formalized by Rajkumar et al. [11] as a generic kernel mechanism to allocate a fraction of a computational resource to a set of tasks. Feng and Mok [12] adopted resource reservations for achieving hierarchical partitioning of computational resources.

Resource reservation is typically implemented through a server mechanism, which allocates a budget Q every period P to the served application. Several resource provisioning algorithms have been proposed in the literature, both under fixed priority systems, like the Polling Servers (PS) [13], the Deferrable Server (DS) [14] and the Sporadic Server (SS) [15], and under EDF, like the Dynamic Sporadic Server (DSS) [16] and the Constant Bandwidth Server (CBS) [2].

Fixed reservation paradigms (static reservations) are not appropriate to achieve the desired performance with applications in which the computational demand is highly variable. To cope with such dynamic systems, Buttazzo et al. [17] proposed an elastic scheduling methodology for adapting the rates of a periodic task set to different workload scenarios, without affecting the system schedulability. Abeni et al. [18] presented a framework for dynamically allocating the CPU to tasks whose execution times are not known a priori. Adaptive reservation techniques based on feedback scheduling have also been investigated by the same authors [19].

To the best of our knowledge, however, none of the proposed reservation mechanisms has been analyzed to predict the timing behavior of the served application during a reconfiguration process. Clearly, a safe approach could be to delay the mode change at the next idle time in the system, as done in the FRESCOR framework [20]. However, the delay could be too long and it is highly unlikely that the idle time occurs at the same time for all the applications.

Recently, some mechanisms have been proposed to dynamically change the server models at run-time. For instance, de Olivera et al. [21] addressed the problem of dynamically reconfiguring reservation parameters, offering support for multi-moded and adaptive real-time applications. Valls et al. [22] presented an adaptation protocol based on the definition of a contract model for filtering peaks in resource demands. However, in both frameworks no schedulability guarantee is provided during reconfigurations. The FRESCOR project [23] has proposed mode change protocols for sporadic servers, but they are not as general as the results presented in this paper, which can cope with arbitrary activation patterns.

Finally, in [24] Stoimenov et al. have tackled with the problem of adaptive resource reservation mechanisms in case of TDMA servers. Resource provisioning guarantees are investigated during the mode changes of the TDMA server paradigm.

Contributions: This paper addresses the problem of modifying reservation parameters to comply with different system requirements, specified through a set of operational modes. Provided that, the schedulability can be guaranteed within each mode in the steady state condition, our objective is to extend the schedulability analysis during mode transitions, under EDF scheduling. Thus, we first derived resource reservation functions to bound the server resource provisioning before, after, and during mode transitions. Then, such functions are used to guarantee the resource provisioning in all the working conditions and to check the application feasibility.

Organization of the paper: Section II states the problem addressed in this work and illustrates some examples showing possible anomalies that could occur during server reconfigurations. Section III presents the basic concepts used in our analysis in terms of generic supply and demand bound functions. Section IV classifies some of the server mechanisms and the resource they provide through a common model. Section V illustrates the mode change problem with servers and mostly the critical mode change transition. Section VI shows how to apply the analysis to guarantee real-time constraints in case of mode transitions. Finally, Section VII states our conclusions.

II. Problem Statement

The objective of the mode change analysis with resource reservation mechanisms is to guarantee that the server provides enough resource to its application in order to keep it feasible not only during the steady states, but also *along the transition stages*.

The next motivational examples illustrate situations in which the system is feasible within each mode, but during the mode transition either a reservation server cannot use all its available budget within its period or the served application cannot meet its timing constraints.

Example 2.1: A multi-moded real-time system is composed by two servers S_A and S_B scheduled by EDF (with deadlines equal to periods). In mode I, S_A has a period of 10 time units and it provides a budget of 5 time units each period, that is $S_A^I(5, 10)$, whereas $S_B^I(3, 6)$. In mode II, we

have $S_A^{II}(4,8)$ and $S_B^{II}(3,6)$. The first server S_A changes from mode I to mode II at time $t = 8$.

The two servers can provide all their computational resource to the application they manages in both their modes because the total utilization $\frac{5}{10} + \frac{3}{6}$ is less than 1. During the mode change the second server cannot provide entirely its budget during a transition period, as showed in Figure 1.

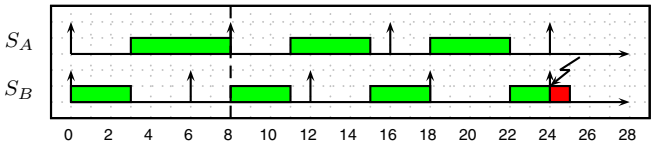


Figure 1. EDF server scheduling: unfeasible mode change at $t = 8$ from $S_A = (5, 10), S_B = (3, 6)$ to $S_A = (4, 8), S_B = (3, 6)$.

This example illustrates that reconfiguring a server may cause capacity miss of other servers, which means that the budget is not delivered to the applications within the server period.

Example 2.2: Consider a polling server S_A that can operate in two modes, $S_A^I(3, 10)$ and $S_A^{II}(1, 4)$, and must handle an application consisting of a single aperiodic task τ_A, with computation time of 3 ms and relative deadline of 15 ms. Figures 2 illustrates the worst-case scenario that can occur when τ_A arrives at time $t = 5$ ms. Note that, although both the application and the server are schedulable in each mode (in steady state conditions), the application can miss its deadline during a mode change occurring at time $t = 10$ ms.

III. System Model and Backgrounds

A real-time application is a set $\Gamma = \{\tau_1, \cdots \tau_n\}$ of n periodic or sporadic tasks. Every task τ_i is characterized by a worst-case execution time C_i, a period (or minimum inter-arrival time) T_i, and a relative deadline D_i smaller than or equal to the period. In this paper, tasks are assumed to be independent and scheduled by EDF. We consider also as resource the computation requested by the applications in order to execute. Nonetheless, our reasoning applies to any kind of resource such as the communication bandwidth, etc..

A. Demand Bound Function

The computational demand of a task set can be precisely described by the *demand bound function* (dbf), introduced by Baruah et al. [25]. It expresses the total computation that must be executed by the processor in each interval of time when tasks are scheduled by EDF. For any given periodic task τ_i activated at time $t = 0$, its demand bound function $\mathsf{dbf}_{\tau_i}(t)$ in any interval [0, t] is given by

$$\mathsf{dbf}_{\tau_i}(t) = \max\left\{0, \left(\left\lfloor \frac{t - D_i}{T_i} + 1 \right\rfloor C_i\right)\right\}.$$

The computational demand of a task set Γ of periodic tasks synchronously activated at time $t = 0$ can be computed as

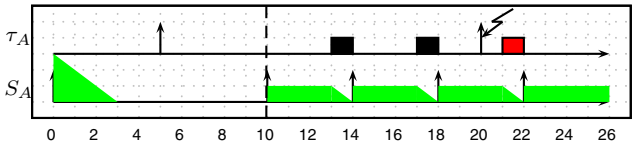


Figure 2. Unfeasible server and task scheduling with $\tau = (3, 15)$, $S^I = (2, 4)$ and $S^{II} = (1, 4)$; mode change at $t_{\mathsf{req}} = 10$.

the sum the individual demand bound functions of each task, that is

$$\mathsf{dbf}_\Gamma(t) = \sum_{\tau_i \in \Gamma} \mathsf{dbf}_{\tau_i}(t).$$

B. Supply Bound Function

The computational resources are provided by reservation servers. In this paper, we consider the class of server algorithms that can be described by a periodic server abstraction.

Definition 3.1 (Periodic Server): A periodic server S is characterized by two parameters (Q, P) where Q is the maximum budget (or server capacity), and P is the server period. A server must guarantee that Q units of time are allocated in each period P to the served application, with $Q \leq P$.

Given server S, its *supply bound function* $\mathsf{sbf}_S(t)$ is the minimum amount of time provided by S in any interval of length $t \geq 0$ [12], [26], [27].

The resource provided by a reservation server can also be described by the *bounded-delay function* (bdf) [12], [26], [27] characterized by the pair (α, Δ), where α is the resource provisioning rate of the server and Δ is the longest interval with no resource provisioning itself. The bdf is defined in the interval domain as

$$\mathsf{bdf}(t) = \max\{0, \alpha(t - \Delta)\} \tag{1}$$

with

$$\alpha = \lim_{t \to \infty} \frac{\mathsf{sbf}(t)}{t}$$
$$\Delta = \inf\{q \mid \alpha(t - q) \leq \mathsf{sbf}(t)\ \forall t\}.$$

The bounded-delay function bdf_S of a server S is defined as a linear approximation of the resource provisioning that lower bounds the resource provisioning, $\forall t \quad \mathsf{bdf}_S(t) \leq \mathsf{sbf}_S(t)$.

C. Feasibility analysis

Using the former abstractions the EDF schedulability of a task set Γ within a server S can be guaranteed if:

$$\forall t \quad \mathsf{dbf}_\Gamma(t) \leq \mathsf{sbf}_S(t). \tag{2}$$

Note that the schedulability can also be tested using the linear bounded-delay function bdf, which however provides a more pessimistic condition, because of the approximation applied. An example of demand bound function, supply

bound function and its bounded-delay approximation is illustrated in Figure 3. In that example the application is feasible under EDF, since in each interval of time the amount of computational resource always exceeds the processor demand of the application.

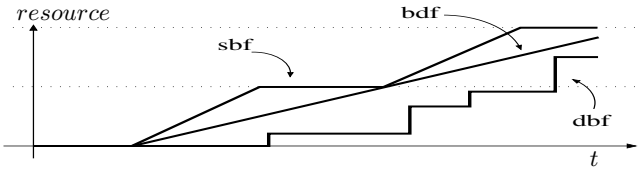


Figure 3. Demand bound function, supply bound function, and its bounded-delay approximation in the interval domain.

D. Mode-Change

Servers can change their parameters at run-time to cope with the system changing conditions. Whenever a server S switches from an *old mode* $S^I = (Q^I, P^I)$ to a *new mode* $S^{II} = (Q^{II}, P^{II})$, the equivalent supply bound function changes from $\mathsf{sbf}_S^I(t)$ to $\mathsf{sbf}_S^{II}(t)$. The supply bound function that describes the computational resource provided by the server across the mode transition is denoted by $\mathsf{sbf}_S^T(t)$, the transition supply bound function. The parameters of the corresponding bounded-delay functions are denoted by the pairs (α^I, Δ^I), $(\alpha^{II}, \Delta^{II})$, and (α^T, Δ^T) respectively.

In the following, t_{req} denotes the time instant at which the mode change is requested. From this time on, all the required changes in the system are initiated, while the new mode begins at t_{go} after a delay $\delta \geq 0$ from the mode change initiation, hence $t_{\mathsf{go}} = t_{\mathsf{req}} + \delta$. The mode transition is the stage between the two steady modes, starting at time t_{req} and ending when the new mode becomes effective, at t_{go}.

A mode changing server can abort its resource provisioning any time during the mode transition $[t_{\mathsf{req}}, t_{\mathsf{go}}]$. We focus our attention on two extreme transition cases:

- *transitionA*: at t_{req} the old mode aborts, interrupting the resource provisioning until the new mode is effective, see Figure 4(a).
- *transitionB*: the old mode server keeps on providing its service until the new mode is effective, see Figure 4(b).

The two cases results in a different behavior of the server and then of the whole system. The differences will be explained in the next sections. All the intermediate cases, with abortion time t_{ab} such that $t_{\mathsf{req}} < t_{\mathsf{ab}} < t_{\mathsf{go}}$ can be easily inferred from the transitionB case.

We assume that the system is feasible in each mode, that is in both steady states conditions, and we want to derive the condition in which feasibility can also be preserved during mode transitions.

IV. Resource Reservation

Although with different peculiarities, a lot of servers can be modeled as periodic servers because they guarantee to

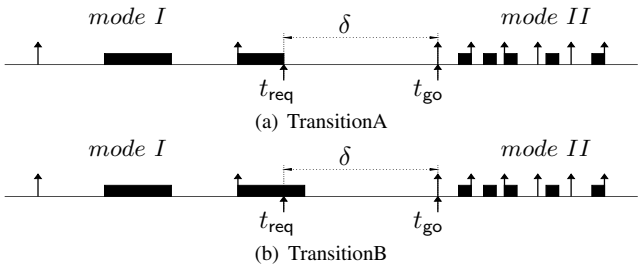


Figure 4. TransitionA and TransitionB; the service provisioning aborts or continues after the mode change t_{req}.

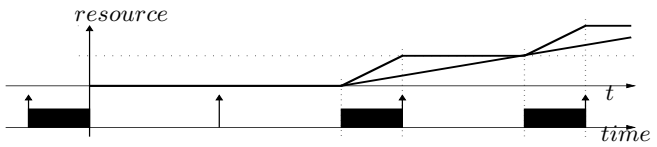


Figure 5. Supply bound function and bounded-delay function for a periodic server in the interval domain t and scheduling in the time domain $time$.

provide Q (and no more than Q) units of time in each period P. Examples of periodic servers include the polling server [13], the deferrable server [14] and the sporadic server [15], which are scheduled using fixed priority. Example of dynamic priority periodic servers include the Constant Bandwidth Servers (CBS) [28] or the dynamic versions of the Polling, Deferrable and Sporadic servers [13].

To achieve a more general result, we perform the analysis in a worst-case scenario where the processor is allocated at the beginning of the first period and then at the end of all subsequent periods, as illustrated in Figure 5. Under this condition, all the servers mentioned above have the same supply bound function, thus we refer to them as generic periodic servers. In the interval domain, the longest interval where no resource is provided is $2(P-Q)$, while after $2(P-Q)$ the server supplies the resource at a constant rate of $\frac{Q}{P}$.

The sbf of a periodic server, defined as the worst-case resource supply in time interval $[0, t)$, is

$$\mathsf{sbf}_S(t) = \max\{0, (k-1)Q, t-(k+1)(P-Q)\} \quad (3)$$

with $k = \left\lceil \frac{t-(P-Q)}{P} \right\rceil$. Such a resource supply bound can be lower bounded by the bounded-delay function of (1), with

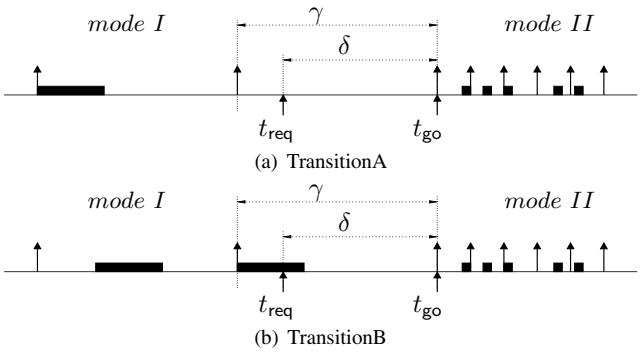
$$\alpha = \frac{Q}{P}, \quad \Delta = 2(P-Q), \quad (4)$$

as illustrated in Figure 5.

The computational resource the server provides is used by the application of the server.

Definition 4.1 (Server Schedulability): A server is said to be schedulable if its budget is provided on time within each period.

Definition 4.2 (Application Schedulability): An application is said to be schedulable by a server if all its tasks are able to meet their deadlines.



(a) TransitionA

(b) TransitionB

Figure 6. Server mode change and mode transition cases. Worst-case scenario resulting in the worst-case service provisioning during the mode transition stage.

Definition 4.3 (System Schedulability): A system consisting of multiple applications managed by a set of reservations is said to be schedulable if both servers and applications are schedulable.

V. Server Mode Change

With multi-moded servers it is important to identify the minimum amount of resource provided in any stage of the server. This includes the transition between different modes, where the service provisioning is affected from both the old and the new mode. The transition supply bound function sbf^T is the resource provisioning during the transition stage, while the transition bounded-delay function bdf^T lower bounds the transient service provisioning.

A. Transition Guarantees

The following theorems provide the sbf^T functions for the two types of transitions introduced in Section III-D (transitionA and transitionB).

Theorem 5.1 (Mode Change sbf *- transitionA):* Let S be a periodic server with steady states parameters (Q^I, P^I) and (Q^{II}, P^{II}) and with corresponding supply bound functions sbf^I and sbf^{II}. Let t_{req} be the time at which the mode change is requested and let t_{go} be the time at which the new mode is started, after a delay δ, such that $t_{\mathsf{go}} = t_{\mathsf{req}} + \delta$. If the server aborts the old mode at time t_{req} (transitionA case), then the resource provisioning during the transition stage is lower bounded by

$$\mathsf{sbf}^T(t) = \inf_{0\le\lambda\le t}\{\mathsf{sbf}^I(t-\lambda-\gamma+P^I-Q^I) + \mathsf{sbf}^{II}(\lambda+P^{II}-Q^{II})\}, \quad (5)$$

being $\gamma = t_{\mathsf{req}} - t^{\mathsf{last}} + \delta$ and t^{last} the initial instant of the last period in the old mode.

Proof: Since the mode change can occur any time during the period interval of the first mode, the worst-case service provisioning occurs when the t_{req} is at the beginning of P^I. The worst-case resource supply is when the old mode provides its budget at the beginning of the last period before the change, while the new mode provides resources at the end of the subsequent periods. This leaves a hole of $\gamma + P^I - Q^I + P^{II} - Q^{II}$ in the service provisioning. If $R(t)$ is the amount of computation (in this case is the computational resource considered) that has been delivered up to time t (along any interval of length t), we are looking upon a bound of such a resource in a generic interval $[r, s)$ with $s = r + t$ centered in $[t_{\mathsf{req}}, t_{\mathsf{go}}]$.

$$\begin{aligned} R[r,s) &= R^I[r,t_{\mathsf{go}}) + R^{II}[t_{\mathsf{go}},s) \\ &\quad \lambda := s - t_{\mathsf{go}} \\ &= R^I[r,s-\lambda) + R^{II}[s-\lambda,s) \\ &= R^I[s-t,s-\lambda) + R^{II}[s-\lambda,s) \end{aligned}$$

and each R ($R^I$is lower bounded by its sbf. In this case, $R^I[s-t, s-\lambda) \ge \mathsf{sbf}^I\left(t-\lambda-(\gamma+P^I-Q^I-2(P^I-Q^I))\right)$ where the service provisioning has a delay of $\gamma + P^I - Q^I$. Instead, $R^{II}[s-\lambda, s) \ge \mathsf{sbf}^{II}\left(\lambda-(P^{II}-Q^{II}-2(P^{II}-Q^{II}))\right)$ with a delay of $P^{II} - Q^{II}$. Both cases are represented in Figure 6(a). Thus

$$\begin{aligned} R[r,s) &\ge \mathsf{sbf}^I(t-\lambda-\gamma+P^I-Q^I) \\ &\quad +\mathsf{sbf}^{II}(\lambda+P^{II}-Q^{II})\ \forall\lambda \\ &\le \inf_{0\le\lambda\le t}\{\mathsf{sbf}^I(t-\lambda-\gamma+P^I-Q^I) \\ &\quad +\mathsf{sbf}^{II}(\lambda+P^{II}-Q^{II})\}, \end{aligned}$$

that proves the theorem. ∎

Theorem 5.2 (Mode Change sbf *- TransitionB):* Let S be a periodic server with steady states parameters (Q^I, P^I) and (Q^{II}, P^{II}) and with corresponding supply bound functions sbf^I and sbf^{II}. Let t_{req} be the time at which the mode change is requested and let t_{go} be the time at which the new mode is started, after a delay δ, such that $t_{\mathsf{go}} = t_{\mathsf{req}} + \delta$. If the server continues to provide its old mode service until time t_{go} (transitionB case), then the resource provisioning during the transition stage is lower bounded by

$$\mathsf{sbf}^T(t) = \inf_{0\le\lambda\le t}\{\mathsf{sbf}^I(t-\lambda-\gamma+2P^I-Q^I) + \mathsf{sbf}^{II}(\lambda+P^{II}-Q^{II})\}, \quad (6)$$

with $\gamma = t_{\mathsf{req}} - t^{\mathsf{last}} + \delta$ and t^{last} the initial of the last period of the old mode.

Proof: The worst-case service provisioning occurs when the old mode provisions its resource at the beginning of its last period, while the new mode provisions at the end of its period. This leaves a hole in the service provisioning of $\gamma - Q^I + P^{II} - Q^{II} \le \delta + P^{II} - Q^{II}$. For any interval $[r, s)$ such that $r \le t_{\mathsf{req}} \le t_{\mathsf{go}} \le s$ the amount of computational

resource provided $R(t)$ is

$$\begin{aligned} R[r,s) &= R^I[r,t_{\mathsf{go}}) + R^{II}[t_{\mathsf{go}},s) \\ &\quad \lambda := s - t_{\mathsf{go}} \\ &= R^I[r,s-\lambda) + R^{II}[s-\lambda,s) \\ &= R^I[s-t,s-\lambda) + R^{II}[s-\lambda,s) \end{aligned}$$

and each R is lower bounded by its sbf. In this case $R^I[s-t,s-\lambda) \geq \mathsf{sbf}^I\left(t-\lambda-(\gamma-Q^I-2(P^I-Q^I))\right)$ where the service provisioning has a delay of $\gamma - Q^I$. Instead, $R^{II}[s-\lambda,s) \geq \mathsf{sbf}^{II}\left(\lambda-(P^{II}-Q^{II}-2(P^{II}-Q^{II}))\right)$ with a delay of $P^{II}-Q^{II}$. Both cases are represented in Figure 6(b). Thus

$$\begin{aligned} R[r,s) &\geq \mathsf{sbf}^I(t-\lambda-\gamma+2P^I-Q^I) \\ &\quad +\mathsf{sbf}^{II}(\lambda+P^{II}-Q^{II})\ \forall\lambda \\ &\leq \inf_{0\leq\lambda\leq t}\{\mathsf{sbf}^I(t-\lambda-\gamma+2P^I-Q^I) \\ &\quad +\mathsf{sbf}^{II}(\lambda+P^{II}-Q^{II})\}, \end{aligned}$$

that proves the theorem. ∎

The obtained sbf^Ts can be used to guarantee the service provisioning during the mode change transitions.

1) Transition Bounded-Delay Function: With the bounded-delay modeling it is possible to derive the transition bounded-delay functions. First, the same idea of Equation (5) and Equation (6) can be applied with bounded-delay functions obtaining

$$\begin{aligned} \mathsf{bdf}^T_A(t) = \inf_{0\leq\lambda\leq t}\{&\mathsf{bdf}^I(t-\lambda+2P^I-\gamma-Q^I)+ \\ &\mathsf{bdf}^{II}(\lambda+P^{II}-Q^{II})\}, \\ \mathsf{bdf}^T_B(t) = \inf_{0\leq\lambda\leq t}\{&\mathsf{bdf}^I(t-\lambda+2P^I-\gamma-Q^I)+ \\ &\mathsf{bdf}^{II}(\lambda+P^{II}-Q^{II})\}, \end{aligned}$$

for transitionA and transitionB, respectively.

The transitionA resource supply results in (α^T_A,Δ^T_A) where

$$\begin{aligned} \alpha^T_A &= \min\{\alpha^I,\alpha^{II}\} \\ \Delta^T_A &= \max\{P^I-Q^I+\gamma+P^{II}-Q^{II},0\}. \end{aligned} \tag{7}$$

For transitionB, we have:

$$\begin{aligned} \alpha^T_B &= \min\{\alpha^I,\alpha^{II}\} \\ \Delta^T_B &= \max\{\gamma-Q^I+P^{II}-Q^{II},0\}. \end{aligned} \tag{8}$$

Both the transition supply bound function and the transition bounded-delay functions depends on the transition delay δ, $\mathsf{sbf}^T(t,\delta)$ and $\mathsf{bdf}^T(t,\delta)$.

Equation (7) and (8) define the relationship between the transition delay δ and the transition service provisioning delay Δ^T.

B. Server Schedulability

In multi-mode systems, an application can change its resource demand from one mode to another, thus dbf^I and dbf^{II} denote the resource demand of the application in the two modes, and dbf^T denotes its resource demand during the mode transition [29]. An application managed by a server is schedulable if and only if the resource demanded by the application does not exceed the resource provided by the server, in any possible stage of both the server and the application. Since we are assuming feasibility for each mode in a steady state condition, the following lemma states the condition for a guaranteed transition in the case both the application and the server change their mode at the same time.

Lemma 5.3 (Mode Change EDF Schedulability): Given a server S handling an application Γ that is feasible in each in a steady state condition, if both S and Γ change their at the same time, then the application is feasible during the transition stage if

$$\forall\ t\ \ \mathsf{dbf}^T_\Gamma(t) \leq \mathsf{sbf}^T_S(t). \tag{9}$$

Using the the bounded-delay linear approximation, the feasibility condition becomes:

$$\forall\ t\ \ \mathsf{dbf}^T_\Gamma(t) \leq \mathsf{bdf}^T_S(t), \tag{10}$$

Note that when the application increases its resource request, a short transition delay in the server adaptation is good for the application. However, a too short delay could result in a service over-provisioning that would steal bandwidth from the other servers, so jeopardizing the schedulability of the other applications during the transient. On the other hand, a large delay in the server adaptation would affect the schedulability of the application itself. In general, there is a trade-off between schedulability of the application and the schedulability of the servers.

1) Intra-Server Schedulability: By intra-server schedulability we refer to the schedulability of the application based on the resources provided by the server. From the analysis performed by applying Condition (9) or Condition (10), we can derive a maximum delay $\delta^\flat$ after which the new mode can safely start. For all $\delta \leq \delta^\flat$ the server mode change is feasible for the application because it will result in a larger resource supply.

2) Inter-Server Schedulability: By inter-server schedulability we refer to the schedulability of the servers when they adapt their parameters, independently of the behavior of the served applications. Such an analysis can be performed using the results achieved for multi-mode task sets, which is well known in the real-time literature (see Section I-A). The analysis can easily be applied to periodic servers by considering them as periodic tasks that must receive Q units of computational resource every period P. Therefore, the effect of a mode change in a server has to be investigated

with respect to the entire set of servers composing the system.

Applying the results achieved in [9], [6] or [17], it is possible to derive a minimum mode change delay $\delta^\sharp$ that does not affect the schedulability of the other servers. Any change performed with a delay $\delta \geq \delta^\sharp$ keeps the system feasible because it reduces the amount of resource required by the server during its transition.

Example 5.4: Consider two servers such that $S_1^I = (2, 4)$ and $S_1^{II} = (4, 8)$, whereas $S_2^I = S_2^{II} = (5, 10)$. If a mode change is requested at time $t_{\mathsf{req}} = 2$ and applied with a delay $\delta = 0$ (with S_2^{II} starting at t_{req}), the set of server results unfeasible during the transition. By applying the utilization-based approach proposed in [17], the minimum delay the first server has to wait before changing parameters is $\delta^\sharp = 2$. For all delays larger than $\delta^\sharp$ the server set results feasible, while $\forall \delta < \delta^\sharp$ the server set may suffer deadline misses.

A real-time system with servers and applications is feasible during mode transitions if the delay is less than or equal to a threshold derived from the intra-server analysis and greater than or equal to a threshold $\delta^\flat$ derived from the inter-server analysis; that is, $\delta \leq \delta^\flat \wedge \delta \geq \delta^\sharp$. The resulting feasibility region for δ is then

$$\Phi = \begin{cases} 0 & \text{if } \delta^\sharp > \delta^\flat \\ \forall \delta \mid \delta^\sharp \leq \delta \leq \delta^\flat & \text{if } \delta^\sharp \leq \delta^\flat \end{cases} \tag{11}$$

This result is stated in the following theorem.

Theorem 5.5: Consider a multi-mode system with m servers S_i, each managing an application Γ_i, such that the system is feasible in any of its working mode. If server S_i performs a mode transition with a delay δ, the system remains feasible if $\delta \in \Phi_i$, where Φ_i is the feasibility region of server S_i from Equation (11). If Φ_i is empty, the transition is unfeasible under any condition.

Proof: The theorem directly comes form the construction of the region Φ_i. Since the δs in Φ_i have been obtained by guaranteeing the feasibility of the set of servers in the system, the theorem is easily proved. ■

Algorithm 1 describes how to compute a transition delay that keeps the system feasible.

Algorithm 1 Algorithm to compute the transition delay for the mode changing server S_i. The policy is the strategy adopted by the system to handle mode transitions.

Input: Γ, $\Gamma_S = \{S_1, \ldots, S_m\}$ and the mode changing server S_i, $S_i \in \Gamma_S$;
Output: transition delay $\delta \in \Psi$;
$\delta^\flat$ = intra-serverSchedulability(Γ_i, S_i);
$\delta^\sharp$ = inter-serverSchedulability(Γ_S, S_i);
δ = policy($\delta^\flat, \delta^\sharp$);

VI. Resource Reservation Analysis

The server and the application requirements during a mode transition are encoded into the feasibility region Φ in terms of transition delays that the set of servers and the applications can tolerate.

A. (α, Δ)-*Space*

A server S_i can be described by the tuple (α_i, Δ_i), where α_i represents its bandwidth and Δ_i the worst-case delay in supplying the computational resource to the application. An application can also be mapped into the (α, Δ)-space, as its feasibility region, considered as the set of all service supply pairs that guarantee the application timing constraints. Note that such a region depends on the applied scheduling policy.

In the case of EDF, the application feasibility region is defined by

$$\forall t \in \mathrm{D} \ : \ \mathsf{dbf}(t) \leq \alpha(t - \Delta),$$

meaning that

$$\forall t \ \in \ \mathrm{D} \ : \ \Delta \leq t - \frac{\mathsf{dbf}(t)}{\alpha}$$
$$\Delta \leq \min_{t \in \mathrm{D}} \left\{ t - \frac{\mathsf{dbf}(t)}{\alpha} \right\}, \tag{12}$$

where D is the set of deadlines in which the application schedulability has to checked.

Equation (12) describes the feasibility region $\Omega_{\Gamma, sched}$ of the application in the (α, Δ)-space, which depends on the application Γ and the scheduling algorithm $sched$, that in our case is EDF.

An example of feasibility region for an application under EDF scheduling is illustrated in Figure 7. The feasibility region is the area below the bold curve, while the other curves are those used to derive the region as the minimum among them. All points within the feasibility region represent server parameters that can guarantee the application timing constraints.

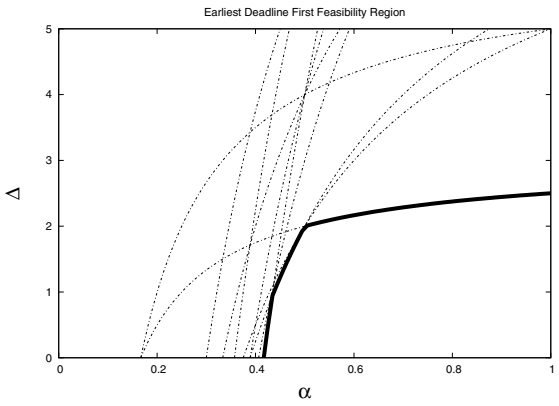


Figure 7. Example of EDF feasibility region for the application $\Gamma = \{\tau_1 = (0.5, 3), \tau_2 = (2, 8)\}$.

In this space, a mode change can be represented by three points, corresponding to mode I, mode II, and the transition. In particular, the transition point is computed

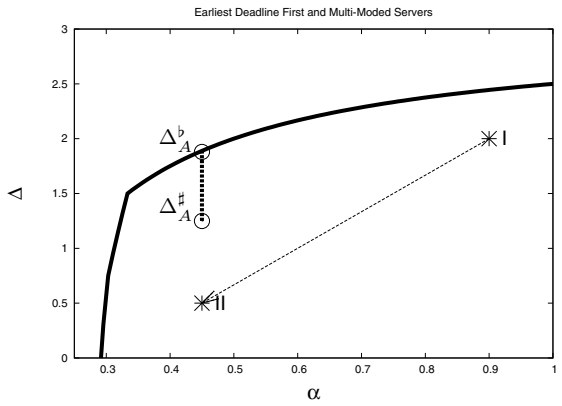


Figure 8. Feasibility region: application scheduled with EDF and server changing from $S^I = (\alpha^I, \Delta^I) = (1, 2)$ to $S^{II} = (\alpha^{II}, \Delta^{II}) = (0.45, 0.5)$. The application has two tasks $\tau_1 = (0.5, 3)$ and $\tau_2 = (1, 8)$. The transitionA is represented, and the segment describes Φ_A for the Δs.

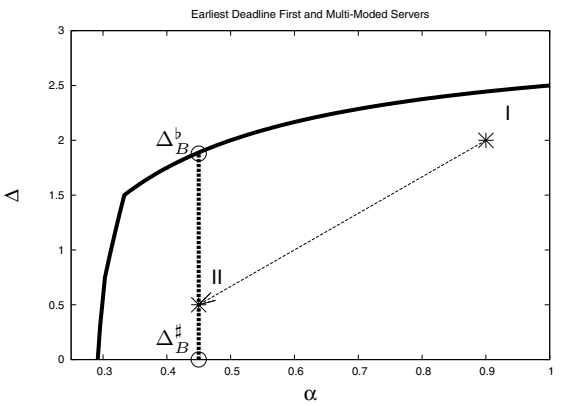


Figure 9. Feasibility region: application scheduled with EDF and server changing from $S^I = (\alpha^I, \Delta^I) = (1, 2.5)$ to $S^{II} = (\alpha^{II}, \Delta^{II}) = (0.45, 0.5)$. The application has two tasks $\tau_1 = (0.5, 3)$ and $\tau_2 = (1, 8)$. The transitionB is represented, and the segment describes Φ_B for the Δs

through Equation (7) or (8), depending on the type of transition. It is worth observing that it depends on the delay δ with which the server adaptation is performed.

The problem addressed in this paper considers the case where the two points corresponding to the steady state modes fall in the feasibility region, and the proposed analysis allows verifying the feasibility of the application also during the transition stage, assuming it is performed with a given delay δ. In the case the transition point falls outside the feasibility region, the delay delta can be used as a design parameter to reach feasibility during the transition, if possible.

B. Mode Change Resource Reservation

An example of mode change is illustrated in Figure 8 and Figure 9 for an application with two periodic tasks, $\tau_1 = (0.5, 3)$ and $\tau_2 = (1, 8)$, handled by a server changing from $S^I = (\alpha^I, \Delta^I) = (0.9, 2)$ to $S^{II} = (\alpha^{II}, \Delta^{II}) = (0.45, 0.5)$. Supposing that the inter-server schedulability analysis proves a minimum transition delay $\delta^\sharp = 0$, which means $\Delta^\sharp_A = 1.25$ and $\Delta^\sharp_B = 0$ respectively for transitionA and transitionB. From the feasibility analysis of the server and its application we obtain $\Delta^\flat = 1.88$ that in case of transitionA correspond to $\delta^\flat_A = 0.63$, while for transitionB is $\delta^\flat_B = 10.63$. The transitionA case results in a region $\Phi_A = [0, 0.63]$ while, the transitionB has $\Phi_B = [0, 10.63]$. Figure 8 shows the case of transitionA and the feasible region for the delays δ in terms of bounded-delay function delays Δ. Figure 9 shows the case of transitionB together with the fact that transitionB is the best possible transition because it provides the widest interval of feasible δ inside the application feasibility region. As a matter of fact the whole Φ_B in terms of Δs stays inside the application feasibility region.

The figure also reports the two values $\Delta^\flat$ and $\Delta^\sharp$ (derived from $\delta^\flat$ and $\delta^\sharp$ through Equation (7) or (8)) which identify the range of transition delays that make the application feasible. For this application, the maximum Δ that can guarantee a feasible transition is $\Delta = 1.88$, corresponding to a maximum transition delay of either $\delta = 0.63$ or $\delta = 10.63$, respectively for transitionA or transitionB . The delays have been computed considering a mode change starting at the beginning of old mode period, $t_{\mathsf{req}} = t_{\mathsf{last}}$.

The proposed analysis can be used to find the best δ in accordance to a desired policy that reflects the requirements of the system. The policy included in Algorithm 1 affects the selection of δ in Ψ. In particular, if the goal is to minimize the application response time, then the minimum delay can be selected as $\delta = \min\{\delta \mid \delta \in \Psi\}$. If the goal is to minimize the resource required during the transition (which could be relevant for saving energy), then the maximum delay can be selected as $\delta = \max\{\delta \mid \delta \in \Psi\}$.

Summarizing, a mode change framework for resource reservations has different degrees of freedom for making an unfeasible transition feasible.

- *Resource reservation parameters*: the resource reservation parameters before and after the transition affect the transition itself and can be set to make an unfeasible transition feasible.
- *Kind of transition*: the different transition types determine different transition delays because of their resource supply. In particular, transitionB provides a larger service than transitionA, which means that a larger delay can be afforded by the application when transitionB is applied. By selecting the abortion time during the transition it is possible to modulate the solution in order to satisfy the delay requirements of a system: if the allowed delays are small enough, then transitionA has to be preferred; otherwise (if intra-server analysis does not allow such small delays) transitionB has to be applied.
- *The server mechanism*: the particular server mechanism

adopted for implementing a reservation affects the resource provisioning, thus affecting the Φ region for the delay. Hence an appropriate server can be chosen to keep the system feasible during the transition and satisfy the constraints of the problem.

C. Case Study

We now present a case study of a multi-moded resource reservation. Let us consider two periodic servers $S = (Q, P)$ characterized by the following initial modes: $S_1^I = (2, 4)$ and $S_2^I = (5, 10)$. The first server manages an application Γ_1 composed by two tasks $\tau_{1,1} = (2, 20)$, $\tau_{1,2} = (5, 30)$ with relative deadlines equal to periods. The second server manages a single task $\tau_{2,1} = (6, 12)$. At time $t = t_{\mathsf{req}} = 2$, S_1 is required to change its parameters from $S_1^I = (2, 4)$ to $S_1^{II} = (4, 8)$, while S_2 remains unchanged $S_2^{II} = S_2^I$. In this case, the system is asking server S_1 to provide more resource with a larger period, but with the same bandwidth. The second server guarantees the same resource provisioning to its application during the system changes.

The transition bounded-delay function for the server S_1 results to have the delay $\Delta_{B,1}^T = \gamma + 2$ for transitionB.

The inter-server analysis provides a minimum delay $\delta^\sharp = 2$ for guaranteeing the feasibility of the other server S_2, see Example 5.4 for more details about the computation of $\delta^\sharp$. In other words, adapting S_1 no earlier than 2 time units from the mode change request, the second server is not affected by the mode change and its resource provisioning is guaranteed also during transitions. This translates into bounded-delay functions with a delay $\Delta_{B,1}^\sharp = 6$ for transitionB.

The intra-server analysis provides the maximum transition delay $\delta^\flat$ that keeps the served application feasible during the transition. That is, $\delta^\flat = \max\{\gamma \mid \mathsf{bdf}^T(t, \gamma) \geq \mathsf{dbf}^T(t)\} - t_{\mathsf{req}} + t^{\mathsf{last}}$. For a transitionB it is $\delta_{B,1}^\flat = 12$.

Hence, the feasibility regions is $\Phi_{B,1} = [2, 12]$ for transitionB. The optimal transition delay δ can then be chosen within these intervals, based on the adopted system policy. In terms of Δ the interval is $[6, 16]$ for transitions like transitionB, as represented by Figure 11. Figure 10 shows that Γ_1 is feasible in both modes of server S_1 and the bounded-delay functions requirements during the transitions.

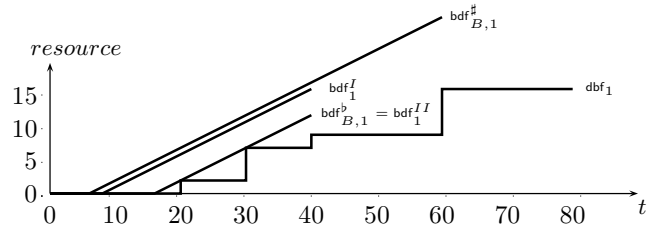


Figure 10. Demand curves (dbf) of the application Γ_1 and resource curves (bdf) for both the transitions and the steady states of server S_1.

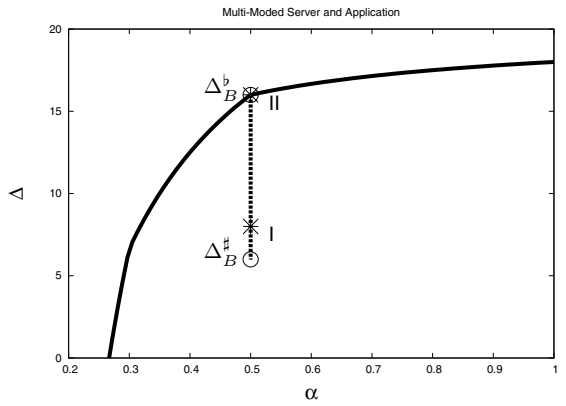


Figure 11. Feasibility region with EDF: bounds to server S_1 parameters in case of transition from mode I, $S^I = (2, 4)$ to mode II, $S^{II} = (4, 8)$. The feasibility region for the delay δ is derived from the maximum Δ^T allowed, represented by the segment. TransitionB is represented.

VII. Conclusions

In this paper we have presented an analysis framework that tackles multi-moded servers. Such a framework has been developed in order to investigate the feasibility of real-time applications in case of dynamic resource reservation policies.

The analysis defines the guarantees to the resource that dynamic and periodic servers provide in any of their possible modes and during their mode transitions. We apply two representations for resource reservations: the periodic model (Q, P) and the bounded-delay model (α, Δ), and by them we have developed schedulability conditions during the critical mode transition stage. The bounded-delay model allows also translating the schedulability of resource reservations into the (α, Δ)-space. The analysis in the (α, Δ)-space outlines the properties that multi-moded applications and multi-moded resource reservation have to guarantee to keep the system feasible in any condition.

The work represents a further step toward the full comprehension of dynamic systems by including multi-moded resource reservations. In the future our framework will be improved with the development of more accurate server models and mostly by considering multiple transitions. Further improvements have to take into account a resource reservation manager that can define complex and dynamic reservation strategies in order to cope with dynamic systems. This component will allow to investigate complex server transitions and the modification of multiple servers inside the real-time system.

References

[1] C. Mercer, R. Rajkumar, and J. Zelenka, "Temporal protection in real-time operating systems," in *Proceedings of the 11th IEEE Workshop on Real-Time Operating Systems and Software (RTOSS'94)*, May 1994, pp. 79–83.

[2] L. Abeni and G. Buttazzo, "Resource reservation in dynamic real-time systems," *Real-Time Syst.*, vol. 27, no. 2, pp. 123–167, 2004.

[3] K. W. Tindell, A. Burns, and A. J. Wellings, "Mode changes in priority pre-emptively scheduled systems," in *RTSS*, 1992, pp. 100–109.

[4] P. Pedro and A. Burns, "Schedulability analysis for mode changes in flexible real-time systems," in *ECRTS*, 1998, pp. 172–179.

[5] L. Sha, R. Rajkumar, J. Lehoczky, and K. Ramamritham, "Mode change protocols for priority-driven preemptive scheduling," *Real-Time Systems*, vol. 1, no. 3, pp. 243–264, 1989.

[6] N. Stoimenov, S. Perathoner, and L. Thiele, "Reliable mode changes in real-time systems with fixed priority or edf scheduling," in *DATE*, 2009.

[7] G. Fohler, "Changing operational modes in the context of pre run-time scheduling (special issue on responsive computer systems)," *IEICE transactions on information and systems*, vol. 76, no. 11, pp. 1333–1340, 1993.

[8] J. Real and A. Crespo, "Mode change protocols for real-time systems: A survey and a new proposal," *Real-Time Systems*, vol. 26, no. 2, pp. 161–197, 2004.

[9] Q. Guangming, "An earlier time for inserting and/or accelerating tasks," *Real-Time Systems*, 2009.

[10] C. W. Mercer, S. Savage, and H. Tokuda, "Processor capacity reserves for multimedia operating systems," Carnegie Mellon University, Pittsburg, Tech. Rep. CMU-CS-93-157, May 1993.

[11] 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, Vol. 3310*, 1998, pp. 150–164.

[12] X. Feng and A. Mok, "A model of hierarchical real-time virtual resources," in *Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS 2002)*, December 2002, pp. 26–35.

[13] G. C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, Springer, Ed. Springer, 1998.

[14] J. P. Lehoczky, L. Sha, and J. K. Strosnider, "Enhanced aperiodic responsiveness in hard real-time environments," in *Proceedings of the IEEE Real-Time System Symposium (RTSS 1987)*, December 1997.

[15] B. Sprunt, L. Sha, and J. P. Lehoczky, "Aperiodic task scheduling for hard real-time systems," *Journal of Real-time Systems*, 1987.

[16] M. Spuri and G. Buttazzo, "Scheduling aperiodic tasks in dynamic priority systems," *Real-Time Systems*, vol. 10, pp. 179–210, 1996.

[17] G. Buttazzo and L. Abeni, "Adaptive rate control through elastic scheduling," in *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, vol. 5, 2000, pp. 4883–4888 vol.5.

[18] L. Abeni and G. Buttazzo, "Adaptive bandwidth reservation for multimedia computing," in *RTCSA '99: Proceedings of the Sixth International Conference on Real-Time Computing Systems and Applications*. Washington, DC, USA: IEEE Computer Society, 1999, p. 70.

[19] ——, "Hierarchical qos management for time sensitive applications," in *RTAS '01: Proceedings of the Seventh Real-Time Technology and Applications Symposium (RTAS '01)*. Washington, DC, USA: IEEE Computer Society, 2001, p. 63.

[20] T. Cucinotta, L. Palopoli, and G. Lipari, "FRESCOR Delivarable D-AQ2v2: control algorithms for coordinated resource-level and application-level adaptation v2," 2008.

[21] A. B. de Oliveira, E. Camponogara, and G. Lima, "Dynamic reconfiguration in reservation-based scheduling: An optimization approach," in *15th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2009, pp. 173–182.

[22] M. G. Valls, A. Alonso, and J. A. de la Puente, "Mode change protocols for predictable contract-based resource management in embedded multimedia systems," *Embedded Software and Systems, Second International Conference on*, vol. 0, pp. 221–230, 2009.

[23] M. G. Harbour, D. Sangorrn, and M. T. de Esteban, "FRESCOR Delivarable D-AT2: schedulability analysis techniques for distributed systems," 2009.

[24] N. Stoimenov, L. Thiele, L. Santinelli, and G. Buttazzo, "Resource adaptations with servers for hard real-time systems," in *International Conference On Embedded Software (EMSOFT)*, 2010.

[25] S. Baruah, R. R. Howell, and L. E. Rosier, "Algorithms and complexity concerning the preemptive scheduling of periodic real-time tasks on one processor," *Real-Time Systems*, vol. 2, 1990.

[26] G. Lipari and E. Bini, "Resource partitioning among real-time applications," in *ECRTS'03, IEEE Computer Society*, 2003, pp. 151–158.

[27] I. Shin and I. Lee, "Periodic resource model for compositional real-time guarantees," in *Proceedings of the* 24^{th} *Real-Time Systems Symposium*, Cancun, Mexico, Dec. 2003, pp. 2–13.

[28] L. Abeni and G. Buttazzo, "Integrating multimedia applications in hard real-time systems," in *Proceedings of the* 19^{th} *IEEE Real-Time Systems Symposium*, Madrid, Spain, dec 1998, pp. 4–13.

[29] S. Perathoner, N. Stoimenov, and L. Thiele, "Reliable mode changes in real-time systems with fixed priority or edf scheduling," Computer Engineering and Networks Laboratory, ETH Zurich, ftp://ftp.tik.ee.ethz.ch/pub/publications/TIK-Report-292.pdf, TIK Report 292, Sep. 2008.