Schedulability analysis of hierarchical systems with arbitrary scheduling in the global level

Ana Guasque*, Patricia Balbastre*,Vicent Brocal† and Alfons Crespo*

*Institute of Control Systems and Industrial Computing (ai2), Valencia, Spain
anguaor, patricia, acrespo@ai2.upv.es

†Fent Innovative Software Solutions (FentISS)
vbrocal@fentiss.com

Abstract

Schedulability analysis of hierarchical real-time systems is based in the previous knowledge of the scheduling algorithms both in the local and the global levels.

In a partitioned system with safety and security issues and certification assurance levels, the global scheduling is usually generated using a static table. Therefore, each partition must allocate task jobs only in the temporal windows reserved for that partition. Even if the static table can come originally from a periodic server or other scheduling policy, the final plan can suffer from modifications due to changes in the system requirements. As a consequence, the CPU assignment to a partition does not have to correspond to any known policy. In this case, it is not possible to use existing scheduling analysis for hierarchical systems. This paper provides a schedulability analysis when the global level policy is not known but provided as a set of arbitrary time windows. The paper also provides a method to determine the more restrictive CPU assignment for a task set, as a mean of stablishing minimum temporal requirements for the partition.

I. Introduction

In the last years, modern computing systems have increased its processing capabilities and its computational resources in such a way that they are capable of executing several concurrent real-time applications that would formerly have required several embedded systems. Scheduling different real-time applications in a mono-processor not only achieves a cost reduction but it also offers the possibility of performance enhancement and closer integration of distinct applications [1].

In many domains such as avionics, space or industrial control systems, hard real-time constraints, safety and security issues and certification assurance levels are commonly required. Integrated Modular Avionics (IMA) was an architectural proposal emerged as a design concept to integrate in a hardware platform several applications with different levels of criticality. IMA approach proposes to encapsulate functions into partitions configuring a partitioned system. Partitioned architectures isolate software components into independent partitions whose execution shall not interfere with that of other partitions, preserving temporal and spatial isolation. Several projects have successfully developed using this approach in the avionic market.

For the last decade, the European space sector has adapted the initial IMA approach for the space requirements for the new satellite generation [2]. The IMA-SP project was focused on mono-processors [3]. The platform defines a virtualization layer (hypervisor) that permits the execution of several partitions. Each partition can contain a guest operating system and the application software. The hypervisor is in charge of assuring temporal and spatial isolation of partitions.

An IMA development process involves several roles as:

- System Architect (SA): The SA has responsibility to define the overall system requirements and the system design, including the optimal decomposition into hosted partitions jointly with the detailed resource allocation per partition.
- System Integrator (SI): The SI is responsible for verifying the feasibility of the system requirements defined by the SA, as well as responsible for the configuration and integration of all components.
- Application Suppliers (AS): An AS is responsible for the development of an application according to the

** This work has been partially funded by the Spanish Government in the frame of the project COBAMI: DPI2011-28507-C02-02
*** This paper has been submitted to IFAC Conference on Embedded Systems, Computational Intelligence and Telematics in Control CESCIT'15

overall requirements from the SA and the SI. AS shall verify that the allocated budget and safety parameters are respected. Assuming that each application is located in a partition and a partition can have only one application, an AS can also be called Partition Developer (PD).

There are other roles in the process but due to space restrictions we only detail those interesting for the purpose of the paper. For a complete description of the main roles and responsibilities see ([4]).

A key element in the development process and the final execution is the configuration of the system defined by the SA, which includes the components description and the resource allocation. This is identified as configuration data or configuration file. In order to preserve the confidentiality of the developing process, a configuration data is split and delivered to each PD with the required information for the application development.

Regarding the allocation of temporal resources, the SI is responsible for CPU allocation to applications while the PD shall manage the time budget assigned to its tasks by the SI.

Then, a partition does not have all the time assigned to schedule its tasks, but only certain slots throughout the hyper-period. To satisfy the temporal resource requirements of the partition, the system must supply sufficient computational resources.

The list of assigned slots is provided by the SI, that is in charge of assuring the feasibility in the global level. Thus, PD's give to the SI its temporal requirements, normally in the form of CPU bandwidth. The SI calculates and assigns this bandwidth to partitions using a well known bandwidth server or it can assign it using a cyclic scheduling. ARINC 653 standard [5] defines a hierarchical scheduling where a static cyclic executive scheduler is used in the global level.

If the assignment is made using a bandwidth algorithm, the corresponding feasibility tests are available in the literature so the PD can apply them to know if its tasks are schedulable with this slots assignment (see section V). On the contrary, if the assignment is made by the SI in an arbitrary way (i.e. not following any existing scheduling algorithm) the authors are not aware of any paper that addresses and solves this problem.

Moreover, schedulability tests for hierarchical systems are based on calculating the worst case response time of tasks in the local level and adding the worst case overhead due to the global level. But, if the scheduling policy is not known in the global level, existing schedulability tests can not apply. Our aim is to generalized the global level model characterizing it by a slots assignment (arbitrary CPU supply as we will comment later).

A. Contributions and outline

The problem to be addressed is concerned with the schedulability of a hierarchical system composed of two levels. The global level policy is not known but provided by the SI as a set of arbitrary time slots. By arbitrary we understand that are not derived from any known scheduling algorithm in the global level. We provide a schedulability analysis so the PD can accept the slots assignment. In the local level, we will assume EDF. Obviously, our results can be used even if the scheduling algorithm in the global level is known.

The results presented in this paper are also useful in the other direction. This paper provides a method to determine the more restrictive assignment slots for a task set, as a mean of minimum requirements that the SI assignment must meet. If the global level policy is known and based on a periodic server, our contributions will provide an alternative way of deducing the solution space for the server parameters.

The paper is organized as follows: Section II presents the model and notation used, while in section III the calculation of the minimum supply bound function is explained. The schedulability test is presented in section IV.Section V presents the most important works in the field of hierarchical scheduling. Finally, section VI summarizes the contributions of the paper and future lines of work.

II. System model and notation

Our model is concerned with the pre-emptive scheduling of real-time applications on an uniprocessor. Each application consists of a number of *partitions* $P_1, .., P_m$. Each partition comprises a number of tasks. Thus, our hierarchical system has two levels, the partition (or global) level and the task (or local) level, each of them with its own scheduling policy. In this work, we will assume that the local level is scheduled under EDF scheduling policy and the global level is scheduled under any scheduler. The information regarding the global scheduling is provided as temporal windows or slots in which a partition is allowed to execute.

In what follows, we will omit the sub index used to refer to a partition to simplify the notation. Therefore, formally, a partition P can be defined[1] as a tuple $P = \{\tau, R\}$ where:

- $\tau = \{\tau_1, \tau_2, .., \tau_n\}$ is a set of n tasks. A task τ_i is characterized by a tuple $\tau_i = \{\phi_i, C_i, D_i, T_i\}$ where ϕ_i is the offset, C_i is the worst case computation time, D_i is the relative deadline and T_i is the period. When all parameters to the system are integers, we may assume without loss of generality that all preemptions

[1]In the definition of the partition we omit all non-temporal resources

occur at integer time values. We then assume, for the remainder of the paper, that all parameters are indeed integers. Moreover, constrained deadlines are assumed so $D_i \leq T_i$.

- An arbitrary CPU supply R is represented by a sequence of intervals $I_1, I_2, ..., I_p$. Every I_i / $1 \leq i \leq p$ is a closed interval $I_i =: [s_i, e_i]$ repited every lcm_τ[2], so that $0 \leq s_i < e_i < s_{i+1}$ and $e_p \leq lcm_\tau$.
  Therefore, $\forall t$ exists a unique interval I_i so that $s_i \leq t \leq e_i$. The CPU supply R for a partition determines the *p* temporal slots in which tasks allocated to the partition are allowed to execute.

The problem to solve is concerned with the schedulability of the partition, that is, if task set τ can be scheduled without deadline misses in the slots defined by R.

A. Supply bound function

Although we have characterized R as a set of intervals, it can also be defined as a function.

Given a CPU supply R and interval of length t, the supply bound function gives the minimum amount of resource that model R is guaranteed to supply in any time interval of length t [6]. We can define the supply bound function of R, accordingly with the above definition.

Definition 1: The supply bound function ($sbf_R(t)$) of an arbitrary supply R expressed as a set of intervals is:

$$sbf_R(t) = \begin{cases} \sum_{i=0}^{j}(e_i - s_i) + t - e_j & \text{if } \exists j/t \in [s_j, e_j], \\ \\ \sum_{i=0}^{j}(e_i - s_i) & \text{if } \exists j/e_j < t < s_{j+1}. \end{cases}$$

Then, a CPU supply R can be characterized either by a set of intervals I_i or by its $sbf_R(t)$.

A basic schedulability condition is:

$$\forall t \quad sbf_R(t) \leq t$$

III. Minimum CPU supply for a task set

The goal of this section is to determine the minimum CPU supply for a task set τ while maintaining feasibility. To do this, we will base our method in the demand bound function for a task set.

If tasks are simultaneously activated at time t = 0 (i.e. $\phi_i = 0$ for all the tasks so teh task set is synchronous), then:

Definition 2: [7] The maximum cumulative execution time requested by jobs of τ whose absolute deadlines are less than equal to t is:

$$dbf_\tau(t) = \sum_{i=1}^{n} C_i \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor$$

[2]Least Common Multiple of $T_1, .., T_n$

It is a positive and increasing function that only increases in the so-called *scheduling points* i.e., when a deadline arrives.

The minimum supply bound function ($msbf_\tau(t)$) is the more restrictive set of temporal windows that can successfully schedule τ. We will show in the next results how to obtain the first two intervals of the function and, then, $msbf_\tau(t)$ will be generalized.

Theorem 1: Let $t_1 \in \mathbb{N}$ so that:

$$t_1 - dbf_\tau(t_1) = \min_t(t - dbf_\tau(t)) \quad \forall t \in (0, lcm_\tau]$$

Then,

$$msbf_\tau(t) = \begin{cases} t - t_1 + dbf_\tau(t_1) & \text{if } t \in [t_1 - dbf_\tau(t_1), t_1], \\ 0 & \text{if } t \in [0, t_1 - dbf_\tau(t_1)). \end{cases}$$

Proof: We will base the proof on adding a new task τ_{n+1}. This task will execute only where τ is not allowed to execute, that is, in $[0, t_1 - dbf_\tau(t_1))$ and we will demonstrate that the new set is schedulable. Then, we will increase the computation time of τ_{n+1} to see that the new set is not schedulable.

Let

$$\tau' = \tau \bigcup \tau_{n+1}$$

where

$$C_{n+1} = t_1 - dbf_\tau(t_1)$$

$$D_{n+1} = t_1 - dbf_\tau(t_1)$$

$$T_{n+1} = lcm_\tau$$

To prove schedulability of τ', the condition $dbf_{\tau'}(t) \leq t$ must hold in all the scheduling points in $[0, t_1]$. Let's suppose that a is a scheduling point in $[0, t_1]$. If $a < D_{n+1}$ clearly $dbf_{\tau'}(a) = dbf_\tau(a) \leq a$. If $a \geq D_{n+1}$ the demand bound function of the new task set τ' is:

$$\begin{aligned} dbf_{\tau'}(a) &= dbf_\tau(a) + C_{n+1} \\ &= dbf_\tau(a) + t_1 - dbf_\tau(t_1) \end{aligned}$$

As $t_1 - dbf_\tau(t_1) = \min_t(t - dbf_\tau(t))$ then

$$t_1 - dbf_\tau(t_1) \leq (a - dbf_\tau(a))$$

So,

$$\begin{aligned} dbf_{\tau'}(a) &\leq dbf_\tau(a) + a - dbf_\tau(a) \\ &\leq a \end{aligned}$$

Now, let's suppose

$$\tau'' = \tau \bigcup \tau_{n+1}$$

and

$$C_{n+1} = t_1 - dbf_\tau(t_1) + \epsilon$$

$$D_{n+1} = t_1 - dbf_\tau(t_1) + \epsilon$$

$$T_{n+1} = lcm_\tau$$

being ϵ a small positive number such that $0 < \epsilon \leq 1$.
Following the same reasoning:

$$dbf_{\tau'}(t_1) = dbf_\tau(t_1) + t_1 - dbf_\tau(t_1) + \epsilon$$
$$= t_1 + \epsilon$$

so τ'' is not schedulable.

■

As a result of the previous theorem, $msbf_\tau(t)$ until t_1, expressed as a set of intervals, is $msbf_\tau(t) = I_0 = [t_1 - dbf_\tau(t_1), t_1]$. Using a similar approach we will derive the next interval.

Lemma 1: Let $t_2 \in \mathbb{N}$, $t_1 < t_2$ so that:

$$t_2 - dbf_\tau(t_2) = \min_t(t - dbf_\tau(t)) \quad \forall t \in (t_1, lcm_\tau]$$

Then

$$msbf_\tau(t) = \begin{cases} t - t_1 + dbf_\tau(t_1) & \text{if } t \in [t_1 - dbf_\tau(t_1), t_1], \\ dbf_\tau(t_1) & \text{if } t \in (t_1, \\ & t_2 - dbf_\tau(t_2) + dbf_\tau(t_1)) \\ t - t_2 + dbf_\tau(t_2) & \text{if } t \in [t_2 - dbf_\tau(t_2) \\ & +dbf_\tau(t_1), t_2], \\ 0 & \text{if } t \in [0, t_1). \end{cases}$$

Proof:
Schedulability in $[0, t_1]$ is assured due to Theorem 1. Following the same reasoning than Theorem 1, we are going to add a task which computation time coincides with the idle time between I_0 and SI_1 and a deadline equal to the start time of I_1 (s_1).

Let
$$\tau' = \tau \bigcup \tau_{n+1} \bigcup \tau_{n+2}$$

where
$$C_{n+1} = t_1 - dbf_\tau(t_1)$$
$$D_{n+1} = t_1 - dbf_\tau(t_1)$$
$$T_{n+1} = lcm_\tau$$

$$C_{n+2} = t_2 - dbf_\tau(t_2) + dbf_\tau(t_1) - t_1$$
$$D_{n+2} = t_2 - dbf_\tau(t_2) + dbf_\tau(t_1)$$
$$T_{n+2} = lcm_\tau$$

Let's suppose that a is a scheduling point in $(t_1, t_2]$. If $a < D_{n+2}$, then $dbf_\tau(a) = dbf_{\tau'}(a)$, so the new task set is schedulable. If $a \geq D_{n+2}$, following the same reasoning than in Theorem 1, we add to $sbf_{\tau'}(t)$ the computation time of τ_{n+1} and τ_{n+2}:

$$dbf_{\tau'}(a) = dbf_\tau(a) + t_1 - dbf_\tau(t_1) +$$
$$+ t_2 - dbf_\tau(t_2) + dbf_\tau(t_1) - t_1$$
$$= dbf_\tau(a) + t_2 - dbf_\tau(t_2)$$

As,
$$t_2 - dbf_\tau(t_2) \leq (a - dbf_\tau(a))$$

then

$$dbf_{\tau'}(a) \leq dbf_\tau(a) + a - dbf_\tau(a)$$
$$\leq a$$

■

Theorem 1 and Lemma 1 provide a method to obtain the first two intervals of $msbf_\tau(t)$ function. Figure 1 shows graphically how the function is obtained.

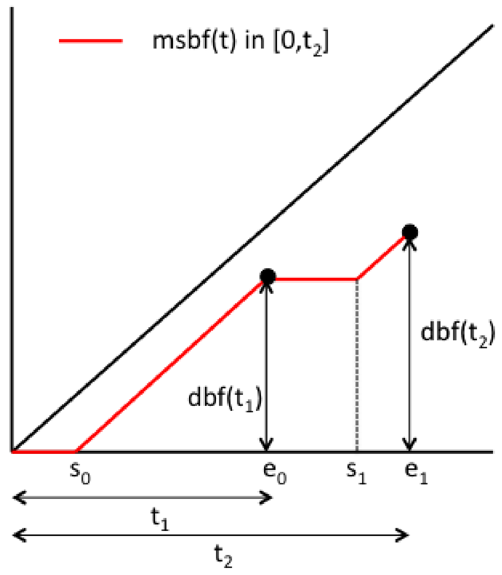

Figure 1. Calculation of $msbf_\tau(t)$ in $[0, t_2]$

It is straightforward to recursively construct all the minimum supply slots needed by τ to maintain feasibility by finding t_j points in where it holds that $t_j - dbf_\tau(t_j)$ is the minimum value in $(t_{j-1}, lcm_\tau]$. We will call this points the *minimum scheduling points* t_j. Therefore, the $msbf_\tau(t)$ is defined as in definition 2 but now we can give specific values to s_j and e_j:

$$msbf_\tau(t) = \begin{cases} \sum_{i=0}^{j}(e_i - s_i) + t - e_j & \text{if } \exists j / t \in [s_j, e_j], \\ \\ \sum_{i=0}^{j}(e_i - s_i) & \text{if } \exists j / e_j < t < s_{j+1}. \end{cases}$$

where $s_j = t_j - dbf_\tau(t_j) + dbf(t_{j-1})$ and $e_j = t_j$.

Replacing the values of s_j and e_j in the previous definition we find that:

$$\sum_{i=0}^{j}(e_i - s_i) = t_1 - t_1 + dbf_\tau(t_1) - dbf(t_0) +$$
$$+t_2 - t_2 + dbf_\tau(t_2) - dbf(t_1) + ...$$

Assuming that $t_0 = 0$ and $dbf_\tau(0) = 0$:

$$\sum_{i=0}^{j}(e_i - s_i) = dbf_\tau(t_j)$$

Therefore, we can provide a more compact definition for $msbf_\tau(t)$:

$$msbf_\tau(t) = \begin{cases} t - t_j + dbf_\tau(t_j) & \text{if } \exists j/t \in [s_j, e_j], \\ \\ dbf_\tau(t_j) & \text{if } \exists j/e_j < t < s_{j+1}. \end{cases}$$

The previous function is obtained from the $dbf_\tau(t)$ in definition 2, particularized for synchronized tasks. If we assume the possibility of asynchronism (i.e., exists any $\phi_i \neq 0$) between tasks, another function $dbf_\tau(t)$ must be defined.

IV. Schedulability analysis

Once $msbf_\tau(t)$ is obtained the following theorem provides the schedulability condition of $\{\tau, R\}$.

Theorem 2: A task set τ is schedulable under a CPU supply R if and only if:

$$\forall t \quad sbf_R(t) \geq msbf_\tau(t)$$

Proof:

We will prove that for any time point a :

$$msbf_\tau(a) \geq dbf_\tau(a) \quad \forall a \in [0, lcm_\tau]$$

We will suppose two cases:

- Case 1: $a \notin [s_j, e_j]$.
- Case 2: $a \in [s_j, e_j]$.

Case 1: If $a \notin [s_j, e_j]$, then $\exists j$ so $e_j < a < s_{j+1}$. Applying the second case in the $msbf_\tau(t)$ definition:

$$msbf_\tau(a) = dbf_\tau(a)$$

Case 2: If $a \in [s_j, e_j]$, applying the first case of the msbf definition:

$$msbf_\tau(a) = a - t_j + dbf_\tau(t_j)$$

As $dbf_\tau(t)$ is a positive and monotonic increasing function it holds that [8]:

If $a \leq t_j$ then $dbf_\tau(a) \leq dbf_\tau(t_j)$

And, as τ is schedulable then $dbf_\tau(t_j) - t_j \leq 0$.

Therefore:

$$\begin{aligned} msbf_\tau(a) &\geq dbf_\tau(a) - dbf_\tau(t_j) - t_j \\ &\geq dbf_\tau(a) \end{aligned}$$

In any case: $msbf_\tau(a) \geq dbf_\tau(a)$, so:

$$sbf_R(a) \geq dbf_\tau(a)$$

■

V. Related work

In a partitioned architecture, partitions can be viewed as components that consist of a real-time workload and a scheduling policy for the workload. This definition coincides with a compositional system hierarchically organized. Different works in compositional scheduling have been proposed for a variety of real-time task models ([9], [10], [11], [6]).

Hierarchical scheduling has been a topic of research interest in recent years. One of the first works on this area is the one presented by Deng and Liu [12], based on a two-level real-time scheduling framework. Kuo and Li [13] presented an exact schedulability condition for this framework assuming fixed priority pre-emptive scheduling. Lipari and Baruah [14] presented a similar work with EDF. Saewong et al. [15] provided a response time analysis for fixed-priority hierarchical systems. This analysis was pessimistic as it was showed by Davis and Burns in [16]. However, Davis and Burns gave an exact response time calculation only for the partition with the highest priority. Almeida and Pedreiras [17] improved the analysis by Saewong et al. but, again it was not an exact analysis. In [18], Bril et al. show that the worst case response time of a task is not necessarily assumed for the first job with the conditions of critical instant provided by Davis and Burns. They claim that the existing analysis could be improved but they do not provide an alternative analysis. The exact response time was provided by Balbastre et al. in [19]. Lipari and Bini [20] provide a kind of sensitivity analysis of the global level for a two-level hierarchical system, providing a methodology for calculating the domain parameters that makes the task set feasible. Lorente and Palencia [21] presented a worst-case response time analysis for the tasks in a two level hierarchical EDF systems.

VI. Conclusions

This paper has considered the case of a two level hierarchical real-time system, in which tasks in the local level are scheduled under EDF policy but the global level does not follow a known scheduling policy but the only information is provided as a set of CPU slots. The first contribution is a schedulability analysis to know if given a certain sequence of CPU slots the task set is schedulable. The second contribution is the calculation of the more restrictive sequence of slots (minimum supply bound function) of a task set in the local level. Our model can also be used with any existing server in the global level, however our model is a generalization of any scheduling in the global level. Although we initially apply our results to static scheduling, further work will be focused on providing on-line algorithms to calculate $msbf_{\tau}(t)$ in a specific window. This could be specially useful in flexible environments, where even if the scheduling algorithm is known a priori, jitter or latencies makes the final slots execution difficult to predict.

Improvements in the $msbf_{\tau}(t)$ calculation are also foreseen. The proposed algorithm needs all the hyperperiod space to be exact but we expect to dramatically reduce the calculation cost by obtaining an upper bound for $msbf_{\tau}(t)$.

References

[1] F. Zhang and A. Burns, “Analysis of hierarchical edf preemptive scheduling,” in *28th IEEE International Real-Time Systems Symposium, 2007. RTSS 2007.*, Dec 2007, pp. 423–434.

[2] J. Windsor and K. Hjortnaes, “Time and space partitioning in spacecraft avionics,” in *IEEE Conference on Space Mission Challenges for Information Technology*, July 19-23. Pasadena (USA) 2009.

[3] “IMA-SP Integrated Modular Avionics for Space. ESA project 4000100764,” 2011-13.

[4] IMA-SP. Integrated Modular Avionics for Space, “IMA Development Process, Roles and Tools,” IMA-SP D08-11, 2011-13.

[5] *Avionics Application Software Standard Interface (ARINC-653).*, March 2006 2006, Airlines Electronic Eng. Committee.

[6] A. Easwaran, I. Lee, I. Shin, and O. Sokolsky, “Compositional schedulability analysis of hierarchical real-time systems,” in *ISORC*, 2007, pp. 274–281.

[7] S. Baruah, A. Mok, and L. Rosier, “Preemptively scheduling hard real-time sporadic tasks on one processor,” in *IEEE Real-Time Systems Symposium*, 1990, pp. 182–190.

[8] I. Ripoll, A. Crespo, and A. Mok, “Improvement in feasibility testing for real-time tasks,” *Journal of Real-Time Systems*, vol. 11, pp. 19–40, 1996.

[9] S. Marimuthu and S. Chakraborty, “A framework for compositional and hierarchical real-time scheduling,” in *12th IEEE Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2006, pp. 91 –96.

[10] G. Lipari and E. Bini, “Resource partitioning among real-time applications,” in *Real-Time Systems, 2003. Proceedings. 15th Euromicro Conference on*, July 2003, pp. 151–158.

[11] I. Shin and I. Lee, “Periodic resource model for compositional real-time guarantees,” in *Real-Time Systems Symposium, 2003. RTSS 2003. 24th IEEE*, Dec 2003, pp. 2–13.

[12] Z. Deng and J. W.-S. Liu, “Scheduling real-time applications in an open environment,” in *IEEE Real-Time Systems Symposium*, 1997.

[13] T.-W. Kuo and C.-H. Li, “A fixed priority driven open environment for real-time applications,” in *IEEE Real-Time Systems Symposium*, 1998.

[14] G. Lipari and S. Baruah, “Efficient scheduling of real-time multi-task applications in dynamic systems,” in *IEEE Real-Time Tecnology and Applications Symposium*, 2000.

[15] S. Saewong, R. Rajkumar, and J. Lehoczky, “Analysis of hierarchical fixed-priority scheduling,” in *Euromicro Conference on Real-Time Systems*, 2002.

[16] R. I. Davis and A. Burns, “Hierarchical fixed priority preemptive scheduling,” in *IEEE Real-Time Systems Symposium*, 2005.

[17] L. Almeida and P. Pedreiras, “Scheduling within temporal partitions: response-time analysis and server design,” in *Fourth ACM International Conference on Embedded Software (EMSOFT)*, 2004.

[18] R. J. Bril, W. F. J. Verhaegh, and C. C. Wust, “A cognac-glass algorithm for conditionally guaranteed budgets,” in *IEEE Real-Time Systems Symposium*, 2006.

[19] P. Balbastre, I. Ripoll, and A. Crespo, “Exact response time analysis of hierarchical fixed-priority scheduling,” in *15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 2009. RTCSA ’09.*, Aug 2009, pp. 315–320.

[20] G. Lipari and E. Bini, “A methodology for designing hierarchical scheduling systems,” *Journal of Embedded Computing*, vol. 1, no. 2, pp. 257–269, 2005.

[21] J. Lorente and J. Palencia, “An edf hierarchical scheduling model for bandwidth servers,” in *Proceedings. 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 2006.*, 2006, pp. 261–266.