Schedulability Analysis of Hierarchical Real-Time Systems under Shared Resources

Alessandro Biondi, Giorgio C. Buttazzo, *Fellow, IEEE,* and Marko Bertogna, *Senior Member, IEEE*

Abstract—Sharing resources in hierarchical real-time systems implemented with reservation servers requires the adoption of special budget management protocols that preserve the bandwidth allocated to a specific component. In addition, blocking times must be accurately estimated to guarantee both the global feasibility of all the servers and the local schedulability of applications running on each component. This paper presents two new local schedulability tests to verify the schedulability of real-time applications running on reservation servers under fixed priority and EDF local schedulers. Reservation servers are implemented with the BROE algorithm. A simple extension to the SRP protocol is also proposed to reduce the blocking time of the server when accessing global resources shared among components. The performance of the new schedulability tests are compared with other solutions proposed in the literature, showing the effectiveness of the proposed improvements. Finally, an implementation of the main protocols on a lightweight RTOS is described, highlighting the main practical issues that have been encountered.

Index Terms—Real-time systems, Resource reservation, Resource sharing, Hierarchical scheduling

1 Introduction

With the rapid performance enhancement of modern computer architectures, a computer system is typically required to execute several applications concurrently, often independently developed by different teams, but sharing the same resources (e.g., processor, memory, radio transceiver, and other peripheral devices). For instance, in automotive systems, the current trend is to confine the exponential growth of the electronic control units (ECUs) by integrating several software components into a reduced number of more powerful hardware platforms [1].

When running multiple components in the same platform, however, computational activities belonging to different components can affect each others. In particular, the misbehavior occurring in a component could impact on the performance of the entire system. Also, if not properly handled, computational activities can experience reciprocal interference and jerky behavior due to long blocking delays on shared resources. Such delays could degrade the overall control performance, or even jeopardize the system stability.

A possible solution to prevent these problems would be a suitable kernel infrastructure capable of providing temporal isolation among different components, thus allowing the analysis of independently developed applications.

Resource reservation mechanisms [2], [3] can effectively be used to isolate the temporal behavior of concurrent applications and limit their reciprocal interference [4]. The basic idea behind this mechanism is to partition the processor into a number of reservations, each behaving as a slower virtual processor using a fraction of the full processor bandwidth. A reservation can be efficiently implemented by a reservation server S_k, providing a budget Q_k for the application every period P_k. In this case, the bandwidth reserved to an application results to be $\alpha_k = Q_k/P_k$. The advantage of this approach is that an overrun occurring in an application does not interfere with the other applications, but only affects the application experiencing the overrun. Moreover, the application can be designed and analyzed independently of the others, because its execution behavior only depends on its own computational demand and the allocated bandwidth.

A further step to support modularity and temporal isolation on a single platform shared by multiple applications is to provide a hierarchical scheduling framework, where a system $\mathscr{S}$ consists of a number of subsystems (or components), each implemented by reservation server, as schematically illustrated in Figure 1.

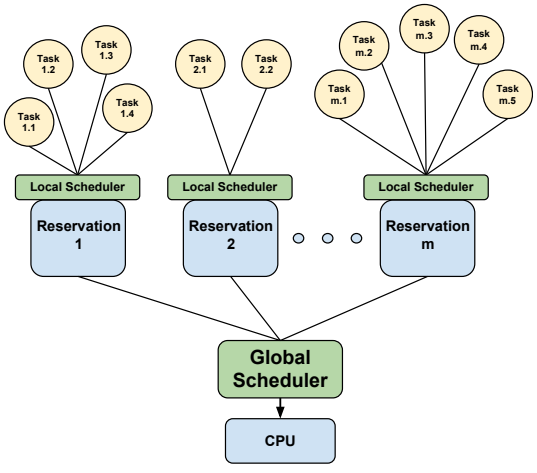


Figure 1: Two-level hierarchical scheduling framework.

- *A. Biondi is with the TeCIP Institute of the Scuola Superiore Sant'Anna, Pisa, Italy. E-mail: alessandro.biondi@sssup.it*
- *G. Buttazzo is with the TeCIP Institute of the Scuola Superiore Sant'Anna, Pisa, Italy. E-mail: giorgio.buttazzo@sssup.it*
- *M. Bertogna is with the University of Modena and Reggio Emilia, Modena, Italy. E-mail: m.bertogna@unimore.it*

In this approach, a *global scheduler* determines which subsystem can access the CPU at any given time, whereas a *local scheduler* selects the running task within the subsystem. In a general hierarchical system, a component can in turn consist of a number of subsystems, by partitioning the component bandwidth through lower-level reservation servers, and so on. In this paper, both EDF and FP are considered as local scheduling policies for each subsystem. For the sake of simplicity, this paper considers a two-level hierarchical system, although the proposed methodology is valid for a generic n-level hierarchical system considering the compositional real-time scheduling framework proposed by Shin and Lee [5].

Under Earliest Deadline First (EDF) scheduling [6], a reservation can efficiently be implemented by a Constant Bandwidth Server (CBS) [3], which has also been extended by Lipari and Baruah [7] to support hierarchical schedulers.

One of the main problems that arises in reservation-based systems comes from the blocking time experienced by tasks when accessing shared resources. Unfortunately, the use of classical synchronization mechanisms, such as semaphores or monitors, may result in a well known phenomenon called *priority inversion* [8]. To bound such a problem, a number of protocols have been proposed, both under fixed priority (FP) assignments [8] and EDF scheduling [9]. Since in this work reservation servers are scheduled using the EDF scheduling algorithm, resources access is controlled by the Stack Resource Policy (SRP) [9], which has been extended to be used in the presence of reservation servers. For example, applying SRP under a two-level hierarchical system requires the definition of two types of resources: those shared among tasks within the same reservation (*local resources*) and those shared among tasks belonging to different reservations (*global resources*). Integrating SRP with such a hierarchical scheme requires addressing the following two problems due to global resources.

Problem 1. When global resources are used by tasks handled within a reservation server, a problem occurs when the server budget is exhausted inside a critical section. In this case, the served task cannot continue the execution, in order to prevent other tasks from missing their deadlines; thus, an extra delay is added to the blocked tasks to wait until the next budget replenishment. Figure 2 illustrates a situation in which a high priority task τ_1 shares a resource with another task τ_2 handled by a reservation server with budget $Q_s = 4$ and period $P_s = 12$. Tasks τ_1 and τ_2 execute upon different reservation servers, S_1 and S_2, respectively. The figure reports in the bottom timeline the budget consumption of S_2. At time $t = 3$, τ_1 preempts τ_2 within its critical section, and at time $t = 4$ it blocks on the locked resource. When τ_2 resumes, however, the residual budget is not sufficient to finish the critical section, and τ_2 must be suspended until the budget will be replenished at time $t = 12$, so introducing an extra delay of 7 units in the execution of τ_1. This example shows that *suspending a task holding a resource leads to unacceptably long delays for tasks in other servers using the same resource*.

To solve this problem various approaches have been proposed in the literature. One of the first solutions is based on a budget overrun: when the budget is exhausted inside a resource, the server is allowed to consume some extra budget until the end of the critical section. This approach was first proposed by Ghazalie and Baker [10], used by Abeni and Buttazzo under the Constant Bandwidth Server (CBS) [3], analyzed under fixed priorities by Davis and Burns [11] and later extended under EDF by Behnam et al. [12], [13]. Davis and Burns proposed two versions of this mechanism:

- *Overrun with payback*, where the server pays back in the next execution instant, in that the next budget replenishment is decreased by the overrun value;
- *Overrun without payback*, where no further action is taken after the overrun.

Note that the budget overrun technique does not increase the response time of the served task, but implies a greater bandwidth requirement for the reservation. Such an extra bandwidth requirement leads to a violation of the temporal isolation property of resource reservation, unless the server is assigned a smaller budget, subtracting the largest possible overrun.

Another proposed solution consists of introducing a budget check before granting the access to a resource: if the budget is sufficient to complete the critical section, the task can access the resource, otherwise the access to the resource is postponed until the next budget replenishment. This mechanism is used in the SIRAP protocol [14] to share resources among reservations. This approach does not affect the execution of tasks in other reservations, but penalizes the response time of the served tasks.

Another approach, named BROE (Bounded-Delay Resource Open Environment), has been proposed by Bertogna et al. [15]. According to this mechanism, when a task wants to enter a critical section and the budget is not sufficient for its completion, a full budget replenishment is planned at the earliest possible time that preserves both the server bandwidth and the maximum service delay. The server is blocked until the budget replenishment.

As highlighted by Kuo and Li [16], since local resources are used only by tasks within a server, no extra delay is added to the blocked tasks that are waiting for a local resource, hence the classical SRP can be adopted to access

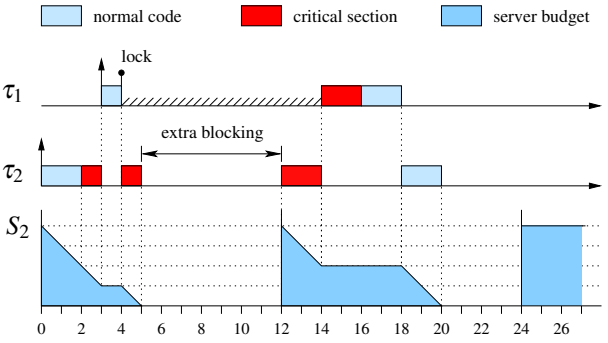


Figure 2: Problem caused when a server budget is exhausted inside a critical section.

local resources within each server. Note that the SRP parameters used in a server for local resources are defined independently of the parameters used in the others servers.

Problem 2. When SRP is used in a hierarchical framework, preemption rules need to be carefully defined when tasks lock global resources. In fact, since tasks within a component may have preemption levels unrelated to those assigned in another component, there is the problem of assigning the ceilings of global resources in a context where *there is no global reference for preemption levels*.

Davis and Burns [11] proposed the Hierarchical Stack Resource Policy (HSRP) extending SRP for hierarchical systems. Their solution consists in defining a preemption level for each server and use it to compute a resource ceiling for each global resource. Similarly to the classical SRP, the ceiling of a global resource is equal to the highest preemption level of any server including tasks that can be blocked on the global resource. Then, ceilings of global resources are used to vary a global system ceiling during execution.

1.1 Contributions

This work provides the following novel contributions:

1) A new and more efficient method is proposed to analyze the schedulability of real-time applications running in a reservation, by incorporating resources constraints directly into the supply bound function describing the worst-case service time of a component. The effectiveness of this approach is evaluated through a set of experiments against existing tests for different algorithms and configuration scenarios.
2) Two new local guarantee tests are proposed to verify the schedulability of real-time applications running on reservation servers implemented with the BROE algorithm, under both fixed priority and EDF local scheduling, in the presence of local and global shared resources.
3) A comparative evaluation is presented to compare the performance of BROE and SIRAP, under both FP and EDF local scheduling, for different application parameters.

It is also worth knowing that there exists an optimal design algorithm [17] for computing the reservation parameters that minimizes the server bandwidth while guaranteeing the application schedulability. The design algorithm is freely available on [18].

1.2 Paper structure

The remainder of the paper is organized as follows. Section 2 presents the system model, the terminology, and the assumptions used throughout the paper. Section 3 briefly recalls the BROE algorithm for handling a reservation in the presence of shared resources. Section 4 derives a new supply bound function for a BROE server taking into account resource holding times, and presents the local schedulability test for real-time task sets scheduled with EDF on a BROE reservation server. A similar test for task sets locally scheduled with fixed priority is presented in Section 5. Section 6 introduces an improvement of the HSRP protocol that allows reducing the blocking time due to global resources. Section 7 reports two sets of experiments aimed at showing the performance of the new schedulability tests for BROE with respect to SIRAP and the original BROE test, under FP and EDF, for different application parameters. Section 8 presents the implementation work of BROE and SIRAP on an existing RTOS, discussing practical implementation issues. Finally, Section 9 states our conclusions and future work.

2 System model

This paper considers a uniprocessor hierarchical system $\mathscr{S}$ consisting of a number of subsystems $S_k \in \mathscr{S}$, each implemented by a BROE [15] reservation server (also denoted as S_k), characterized by a budget Q_k and a period P_k. For the sake of simplicity we consider a two-level hierarchical system, although our contributions can be extended to a generic n-level hierarchical system considering the compositional real-time scheduling framework proposed by Shin and Lee [5]. The *global scheduler* is implemented by a hard Constant Bandwidth Server [19], [20], whereas a *local scheduler* can use either EDF and FP as scheduling policies for each subsystem.

2.1 Task model

Each subsystem S_k runs an application Γ_k consisting of n_k periodic or sporadic tasks. Each task generates a potentially infinite sequence of instances (jobs), executed on different data. They may be activated periodically, at fixed intervals of time, or sporadically, with a minimum interarrival time between consecutive jobs. Each task τ_i is characterized by a worst-case execution time (WCET) C_i, a period (or minimum interarrival time) T_i, and a relative deadline D_i. Under local EDF scheduling, tasks are ordered by increasing relative deadlines, whilst under local FP scheduling tasks are ordered by decreasing priorities, so that τ_1 is the highest priority task.

The *Level-i* notation is used in this paper to generalize the application parameters under both EDF and FP local scheduling. A *Level-i* parameter provides an aggregate information among all the tasks τ_k with $k \leq i$. This notation comes useful for the FP schedulability analysis, which requires computing a schedulability test for each task τ_i (i^{th} level). On other hand, the EDF schedulability analysis is based on a single test covering all the tasks: this requirement fits also well with the *Level-i* notation by simply considering the application parameters for the n^{th} level ($i = n$).

2.2 Resource model

Two types of resources can be defined:

- *Local resource*: a resource shared among tasks within the same subsystem;

- *Global resource*: a resource shared among tasks belonging to different subsystems.

In the following, $\delta_{i,j}$ denotes the WCET for the longest critical section of τ_i related to resource R_j.

Definition 1: The *Resource Holding Time* $RHT_{k,j}(i)$ of a global resource R_j accessed by a task $\tau_i \in S_k$ is the maximum amount of budget consumed by S_k between the lock and the corresponding release of R_j performed by τ_i.

We also define the *Level-i Resource Holding Time* of a global resource R_j accessed by $\tau_h \in \Gamma_k$ as

$$H_{k,j}(i) = \max_{h \le i} \{RHT_{k,j}(h)\} \quad (1)$$

Note that if global resources are accessed by disabling local preemption, $H_{k,j}(i)$ can be expressed as

$$H_{k,j}(i) = \max_{h \le i} \{\delta_{h,j} \mid \tau_h \in \Gamma_k\}. \quad (2)$$

If local preemption is not disabled, $H_{k,j}(i)$ must take into account the worst-case local interference experienced by τ_i during the lock of R_j (details on how to compute $H_{k,j}(i)$ can be found in [15]).

In addition, the *Level-i* maximum Resource Holding Time for an application Γ_k is defined as

$$H_k(i) = \max_j \{H_{k,j}(i)\}, \quad (3)$$

and the maximum Resource Holding Time for Γ_k is defined as

$$H_k = \max_j \{H_{k,j}(n_k)\}. \quad (4)$$

To access shared resources in such a hierarchical framework, the SRP can be used as it is for local resources, while it has to be extended for global resources. In the following, the local and global version of SRP is denoted as SRP-L and SRP-G, respectively, and it is summarized below. In the following, the notation $\{x\}_0$ is used to denote $\{0\} \cup \{x\}$.

Local SRP (SRP-L). Within a server S_k, each task τ_i is assigned a local preemption level π_i and preemption levels are ordered inversely with respect to relative deadlines; that is, $\pi_i > \pi_h \Leftrightarrow D_i < D_h$. Each local or global resource R_j is assigned a local (static) ceiling $\mathscr{C}_j^L$ equal to

$$\mathscr{C}_j^L = \max_i \{\pi_i \mid R_j \text{ is used by } \tau_i\}_0.$$

A subsystem ceiling is defined for each server S_k as

$$\Pi_k^L = \max_j \{\mathscr{C}_j^L \mid R_j \text{ is locked and used by } S_k\}.$$

Then, a task τ_i running in a server S_k can preempt another task in S_k only if $\pi_i > \Pi_k^L$.

Global SRP (SRP-G). To handle global resources, like in HSRP, each server S_k is assigned a preemption level π_k^S and server preemption levels are ordered inversely with respect to server periods; that is, $\pi_h^S > \pi_k^S \Leftrightarrow P_h < P_k$. Each global resource is assigned a global (static) ceiling equal to

$$\mathscr{C}_j^G = \max_k \{\pi_k^S \mid \exists \tau_i \in \Gamma_k \wedge R_j \text{ is used by } \tau_i\}_0.$$

A global system ceiling is defined as

$$\Pi^G = \max_j \{\mathscr{C}_j^G \mid R_j \text{is locked}\}.$$

Then, a server S_k can preempt the currently scheduled server only if $\pi_k^S > \Pi^G$.

Note that, when a global resource is locked, the system ceiling Π^G is incremented and a number of servers is prevented to execute, hence the blocking is extended to all the tasks executing upon the blocked servers.

Symbol	Description
S_k	k^{th} subsystem (reservation server)
Γ_k	Application (task set) handled by S_k
n_k	Number of tasks in Γ_k
Q_k	Budget of server S_k
P_k	Period of server S_k
α_k	Bandwidth of server S_k
Δ_k	Maximum of service delay of server S_k
$\text{sbf}(S_k,t)$	Generic supply bound function of server S_k
$\text{sbf}^B(S_k,t)$	New proposed supply bound function of server S_k
$\text{sbf}^P(S_k,t)$	Periodic supply bound function of server S_k
$\text{sbf}^L(S_k,t)$	Linear α-Δ supply bound function of server S_k
τ_i	i^{th} task
C_i	Worst-case execution time (WCET) of τ_i
T_i	Period (or minimum interarrival time) of τ_i
D_i	Relative deadline of τ_i
R_j	j^{th} shared resource
$\delta_{i,j}$	WCET of the longest critical section of τ_i on R_j
$H_{k,j}(i)$	Level-i RHT of R_j accessed by tasks in Γ_k
H_k	Maximum RHT for Γ_k
$H_k(i)$	Level-i maximum RHT for Γ_k
π_i	Preemption level for τ_i
$\mathscr{C}_j^L$	Local resource ceiling for R_j
Π_k^L	Subsystem ceiling for S_k
π_k^S	Preemption level for S_k
$\mathscr{C}_j^G$	Global resource ceiling for R_j
Π^G	Global system ceiling

Table 1: Notation used throughout this paper.

3 THE BROE SERVER

As stated in Section 1, suspending a task that holds a global resource for a budget exhaustion would lead to unacceptably long delays in tasks in other subsystems wishing to access the same resource. More generally, consider a task τ_i belonging to a server S and let q be the residual budget of S at time t. For the sake of simplicity, in this and in the following section we refer to a single server and thus remove the index k, therefore H will denote the maximum Resource Holding Time of server S, as defined in Equation (4).

If τ_i wants to enter a critical section at time t and $q < H$, then a budget depletion may occur inside the critical section. Since τ_i cannot continue the execution to prevent other tasks from missing their deadlines, an extra delay is added to the blocked tasks to wait until the next budget replenishment.

To address the problem above, Bertogna et al. [15] proposed the BROE server, which is based on a hard Constant Bandwidth Server [19], [20] with period P and maximum budget Q (the bandwidth is $\alpha = Q/P$). At any time t, the server is characterized by an absolute deadline

d and a remaining budget q. When a job executes, q is decreased accordingly. The rules of a BROE server are summarized below:

1) Initially, $q = 0$ and $d = 0$.
2) When BROE is idle and a job arrives at time t, a replenishment time is computed as $t_r = d - q/\alpha$:
   a) if $t < t_r$, the server is suspended until time t_r. At time t_r, the budget is replenished to Q and $d \leftarrow t_r + P$.
   b) otherwise the budget is immediately replenished to Q and $d \leftarrow t + P$;
3) When $q = 0$, the server is suspended until time d. At time d, the server budget is replenished to Q and the deadline is postponed to $d \leftarrow d + P$.
4) When a pending task wishes to access a global resource at a time t, a budget check is performed: if $q \geq H$, there is enough budget to complete the critical section, hence the access is granted. Otherwise a replenishment time is computed as $t_r = d - q/\alpha$:
   a) if $t < t_r$, the server is suspended until time t_r. At time t_r, the budget is replenished to Q and $d \leftarrow t_r + P$.
   b) otherwise the budget is immediately replenished to Q and $d \leftarrow t_r + P$.

According to the above rules, a server running ahead with respect to its guaranteed processor utilization will self-suspend in two cases: when reactivating after an idle time (Rule 2), and when trying to enter a global critical section with insufficient budget (Rule 4). In both cases, it will self-suspend until the guaranteed processor utilization is matched (time $t_r = d - q/\alpha$). At time t_r, the server budget is replenished to Q and the deadline is set to $d \leftarrow t_r + P$. When instead the server consumed less processor resources than its allowed share, it will immediately replenish its budget in the two mentioned cases. However, the deadline set when reactivating after an idle ($d \leftarrow t + P$, according to Rule 2) differs from the one set when trying to enter a global critical section with insufficient budget ($d \leftarrow t_r + P$, according to Rule 4).

4 Local Schedulability Test for BROE under EDF

The local schedulability analysis of a reservation server can be performed using the test proposed by Shin and Lee [5], later extended by Baruah [21] to account for shared resources. According to this test, a task set Γ is schedulable by EDF on a reservation server S if

$$\forall t > 0 \quad B^L(t) + \mathsf{dbf}(\Gamma, t) \leq \mathsf{sbf}(S, t) \tag{5}$$

where $\mathsf{dbf}(\Gamma, t)$ is the *demand bound function* of the task set Γ (i.e., the maximum computational demand of Γ in any interval of length $t > 0$), $\mathsf{sbf}(S, t)$ is the *supply bound function* of the server S, (i.e., the minimum amount of service time provided by the server in any interval of length $t > 0$), and $B^L(t)$ is the blocking time in interval $(0, t]$, computed as the maximum critical section of tasks having deadline $> t$, accessing resources common to at least one task with deadline $\leq t$, that is,

$$B^L(t) = \max_{i,j} \{\delta_{i,j} \mid D_i > t \ \wedge \ \exists \tau_\ell \text{ accessing } R_j \text{ with } D_\ell \leq t\}. \tag{6}$$

As shown in [21], it is possible to limit the number of test points for Equation (5) to a discrete set for an efficient implementation.

In the original paper presented by Bertogna et al. [15], the supply bound function used in the test was actually a linear lower bound $\mathsf{sbf}^L(S, t)$ of a bounded-delay partition (α, Δ), where α is the server bandwidth and Δ is the maximum service delay:

$$\mathsf{sbf}^L(S, t) = \alpha(t - \Delta), \tag{7}$$

where $\alpha = Q/P$ and $\Delta = 2(P - Q)$.

In the following theorem, we improve the effectiveness of the local schedulability test by deriving the actual supply bound function $\mathsf{sbf}^B(S, t)$ of BROE as a function of the maximum resource holding time H of the global resources accessed by Γ. Note that the use of H for the local analysis is compliant with BROE, since resource holding times are also required for the global schedulability analysis, as well as used by the resource access policy to avoid budget depletion within critical sections. The improved supply bound function is illustrated in Figure 3 (continuous line) together with the linear bound (dotted line) proposed in the original work. As clear from the figure, the new supply bound function introduces some additional areas over the linear bound $\mathsf{sbf}^L(t)$, for time intervals in $[\Delta, \Delta + (\lceil Q/H \rceil - 1)P]$. Such areas may be efficiently exploited by the schedulability test in Equation (5) to improve the schedulability analysis. For time intervals outside this domain, the original $\mathsf{sbf}^L(t)$ can be used instead.

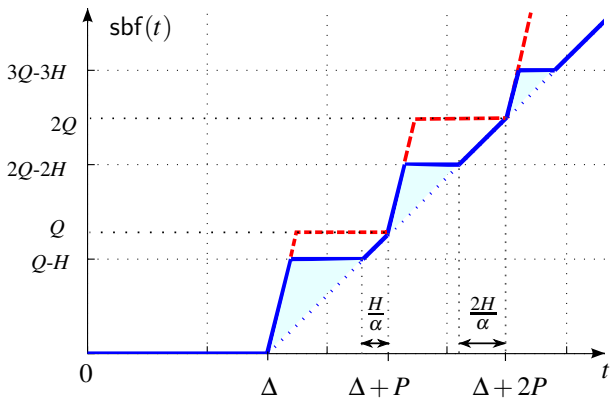


Figure 3: Supply bound functions: periodic (dashed line), linear α-Δ (dotted line), and new sbf proposed for BROE (continuous line).

Theorem 1: In any interval of length $t \in [\Delta, \Delta + (\lceil Q/H \rceil - 1)P]$, the supply provided by a BROE server with a maximum resource holding time of H cannot be lower

than the following function:

$$\mathsf{sbf}^B(t) = \begin{cases} t-\Delta-(k-1)(P-Q) & t_A < t \le t_B \\ kQ-kH & t_B < t \le t_C \\ \alpha(t-\Delta) & t_C < t \le t_D \end{cases} \quad (8)$$

where

$$k = \left\lceil \frac{t-\Delta}{P} \right\rceil, \quad (9)$$

and

$$\begin{cases} t_A = \Delta+(k-1)P \\ t_B = \Delta+(k-1)P+(Q-kH) \\ t_C = \Delta+kP-kH/\alpha \\ t_D = \Delta+kP. \end{cases}$$

Proof: When no global resource is shared, BROE behaves as a classical hard CBS server (see [20]), whose supply bound function is described as [22], [23]:

$$\mathsf{sbf}^P(t) = \max \left\{ \begin{matrix} 0, \\ (h(t)-1)Q, \\ t-(h(t)+1)(P-Q) \end{matrix} \right\} \quad (10)$$

where $h(t) = \left\lceil \frac{t-P+Q}{P} \right\rceil$. Function $\mathsf{sbf}^P(t)$ is shown in Figure 3 as a dashed line.

When considering resource sharing, the worst-case supply can be found by considering Rule 4 of BROE in Section 3, which reduces $\mathsf{sbf}^P(t)$ in some time intervals. Note that Rule 2 does not affect the $\mathsf{sbf}^B(t)$, since it is applied only when the server is resumed from an idle state, and therefore will never be invoked in the busy period considered in the worst-case scenario by the schedulability test of Equation (5).

After its worst-case delay $t = \Delta$, the server will be able to execute for at least $Q-H$ units of time. After time $t = \Delta + Q - H$, a pending task wishing to access a global resource R_j can experience the condition $q < H_j$. According to Rule 4 of BROE, such a condition causes a deadline shift to $t_r + P$, suspending the server H units earlier than in the more favorable $\mathsf{sbf}^P(t)$. Then, the latest time the server can resume execution is at time $t_r + P - Q$. Since $t_r = \Delta + Q - H/\alpha$, then the server can restart executing at $t_r + P - Q$, which, rephrasing the terms, is equal to $\Delta + P - H/\alpha$. Such a point lies at the intersection of the original $\mathsf{sbf}^L(t)$ (dotted line in Figure 3).

Since the same condition imposed by Rule 4 can occur at any time in $(\Delta + Q - H,\ \Delta + Q]$, then $\mathsf{sbf}^B(t) = \mathsf{sbf}^L(t)$ in the interval $(\Delta + P - H/\alpha,\ \Delta + P]$.

Note that, in the next interval $[\Delta + P,\ \Delta + 2P]$, the reduction of the $\mathsf{sbf}^B(t)$ with respect to $\mathsf{sbf}^P(t)$ is more significant. The reason is that the server, when resuming the execution at time $\Delta + P - H/\alpha$, can be suspended again after executing for $Q-H$ time units, thus the $\mathsf{sbf}^P(t)$ is "cropped" earlier than in the previous period (i.e., at $t = \Delta + P + (Q-2H)$). By computing $t_r + P - Q$ using Rule 4, the server is resumed at time $\Delta + 2P - 2H/\alpha$. This point lies again at the intersection of the original $\mathsf{sbf}^L(t)$.

In general, the reduction of the $\mathsf{sbf}^B(t)$ with respect to $\mathsf{sbf}^P(t)$ increases period by period, until it reduces to $\mathsf{sbf}^L(t)$. Considering the k^{th} period after Δ, the intervals in which $\mathsf{sbf}^B(t) = \mathsf{sbf}^L(t)$ are

$$[\Delta + kP - \frac{kH}{\alpha},\ \Delta + kP]. \quad (11)$$

The larger k, the larger such intervals. The first period in which, for all t, $\mathsf{sbf}^B(t) = \mathsf{sbf}^L(t)$ can be derived by finding the smallest k that satisfies the following inequality:

$$\Delta + kP - \frac{kH}{\alpha} \le \Delta + (k-1)P,$$

which gives $k \ge Q/H$. Therefore, from the $\lceil Q/H \rceil$-th period on, the supply bound function cannot be larger than $\mathsf{sbf}^L(t)$.

Figure 4 shows the $\mathsf{sbf}^B(t)$ in the k^{th} period after Δ, i.e., for $t \in [\Delta + kP,\ \Delta + (k+1)P]$, when $k < \lceil Q/H \rceil$. The values of the timing parameters used in the figure are reported in Table 2.

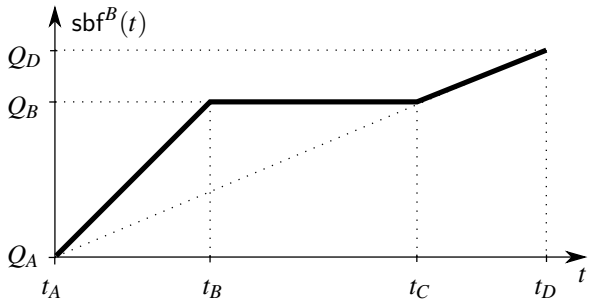


Figure 4: $\mathsf{sbf}^B(t)$ in the k^{th} period after the service delay.

t_A	$\Delta+(k-1)P$
t_B	$\Delta+(k-1)P+(Q-kH)$
t_C	$\Delta+kP-kH/\alpha$
t_D	$\Delta+kP$
Q_A	$(k-1)Q$
Q_B	$kQ-kH$
Q_D	kQ

Table 2: Values for Figure 4.

The supply provided between t_A and t_B can be computed as $t - t_A + Q_A$, giving $t - \Delta - (k-1)(P-Q)$. The supply between t_B and t_C is equal to $Q_B = kQ - kH$. Finally, the supply between t_C and t_D coincides with the linear supply bound function $\mathsf{sbf}^L(t) = \alpha(t-\Delta)$.

The theorem follows noting that for any $t \le \Delta + (\lceil Q/H \rceil - 1)P$, k is smaller than $\lceil Q/H \rceil$. □

Considering that the sbf^B has been derived as the minimum supply provided by BROE for all actual worst-case scheduling scenarios, the resulting schedulability test is *tight* by construction.

Moreover, it is worth observing that H is just an upper bound of the resource holding time, while the effective locking time can be significantly smaller, depending on the interference caused by higher priority jobs preempting the critical section. Note that the presented analysis is robust and sustainable, according to the criteria outlined in [24].

In fact, the analysis does not make any assumption on the *actual* duration of the resource holding time. One may be tempted to improve the supply bound function by assuming that a budget replenishment due to a global lock request (Rule 4) will then cause the corresponding server to execute for H time units. In this case, the supply provided would be higher than in our sbf^B. However, an analysis using such an improved supply function would be not sustainable, i.e., it would not provide sufficient guarantees to tasks that may execute for less than their worst-case execution times. Our supply bound function correctly considers also the case in which a global lock is released an infinitesimal time after the budget replenishment, and another lock request is made QH time units after that, as correctly considered in the proof above.

Analyzing the improved supply bound function, it becomes apparent that the improvement with respect to the original $\mathsf{sbf}^L(t)$ is magnified when H is much smaller than Q. In the extreme case in which no global resource is shared, we have $H = 0$ and $\mathsf{sbf}^B(t)$ coincides with the supply bound function $\mathsf{sbf}^P(t)$ of a periodic server. Conversely, when $H = Q$, $\mathsf{sbf}^B(t)$ is always equal to $\mathsf{sbf}^L(t)$. Different methods have been proposed in the literature to reduce resource holding times by limiting (or disabling) local preemption when accessing global resources [25]. In particular, all critical sections having a length smaller than $\Delta/2$ can be executed non-preemptively by BROE (Theorem 7 in [15]), leading to a minimal resource holding time equal to the critical section length, so magnifying the improvement allowed by the supply bound function presented in this paper.

Note that the improved supply bound function can be simply plugged in the schedulability test of Equation (5). The obtained improvement does not affect the computational complexity of the schedulability test, which remains pseudo-polynomial as the original test based on the linear $\mathsf{sbf}^L(t)$. In fact, computing the value of the novel supply bound function for a given time t just requires identifying the k^{th} period by Equation (9) and then computing the $\mathsf{sbf}^B(t)$ value by Equation (8). Hence, the total computation requires a modulo operation and a constant number of additions and multiplications.

Also, the new proposed method preserves the modularity of BROE's original approach, where the local schedulability of each application can be validated in isolation, without requiring the knowledge of the parameters of the other applications. The global schedulability of the various applications on open environments can then be verified based on simple application interfaces [15]. Such a modular approach scales well with the number of tasks and servers, allowing an efficient integration of multiple applications in an open environment.

5 Local schedulability test for BROE under FP

This section presents the local schedulability analysis of a real-time application scheduled under FP within a BROE server. The guarantee test can be derived from the FP-test proposed by Lehoczky et al. [26] considering that, in any test interval $[0,t]$, only a fraction of time given by $\mathsf{sbf}(S,t)$ is available for the application. Hence, a task set $\Gamma = \{\tau_i, ..., \tau_n\}$ is schedulable under FP on a reservation server S if

$$\forall i = 1, ..., n \; \exists t \in \mathrm{tSet}_i \;\; \mathsf{rbf}_i(t) + B_i^L \leq \mathsf{sbf}(S,t)$$

where tSet_i is the set of test points [26], [27], relative to task τ_i, where the schedulability check has to be performed, that is

$$\mathrm{tSet}_i = \left\{ rT_j \mid j = 1 \ldots i, r = 1 \ldots \left\lfloor \frac{T_i}{T_j} \right\rfloor \right\},$$

and

$$\mathsf{rbf}_i(t) = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_i} \right\rceil C_j,$$

$$B_i^L = \max_{\ell,j} \{ \delta_{\ell,j} | \pi_\ell < \pi_i \wedge \exists R_j \text{ used by } \tau_h : \pi_h \geq \pi_i \} \quad (12)$$

As explained in Section 4, the $\mathsf{sbf}^B(S,t)$ for a BROE server depends on resource holding times in Γ: in particular, for EDF local schedulability, it depends on their maximum value H. Since, under FP, the schedulability test has to be computed for each task τ_i, the BROE supply bound function only depends on the maximum among resource holding times of the i highest priority tasks, that is, the *Level-i* maximum resource holding time $H(i)$ defined in Equation (3).

Hence, to have a more precise schedulability test, the BROE supply bound function presented in Equation (8) can be redefined by introducing the *Level-i* supply bound function $\mathsf{sbf}_i^B(S,t)$, obtained by replacing H with $H(i)$. Using such a refinement, a task set Γ is schedulable by FP on a BROE server if

$$\forall i = 1, ..., n \; \exists t \in \mathrm{tSet}_i \;\; \mathsf{rbf}_i(t) + B_i^L \leq \mathsf{sbf}_i^B(S,t). \quad (13)$$

6 Improving SRP-G

The next example shows a particular situation in which the global SRP-G rule causes an unnecessary blocking. Consider four servers S_1, S_2, S_3, and S_4, with periods $P_1 > P_2 > P_3 = P_4$. Each server S_i runs a single task τ_i. A global resource R_1 is shared between τ_1 and τ_3, and another global resource R_2 is shared between τ_2 and τ_4.

Server preemption levels are defined according to SRP-G, thus $\pi_1^S < \pi_2^S < \pi_3^S = \pi_4^S$, and when any resource is locked, its ceiling will be $\mathscr{C}_j^G = \pi_3^S = \pi_4^S$. Suppose that τ_1 starts executing and locks R_1. After the lock operation, the global system ceiling will be $\Pi^G = \pi_3^S$. Hence, if τ_4 arrives when τ_1 is locking R_1, it cannot preempt, because its server (S_4) is blocked by SRP-G ($\Pi^G \geq \pi_4^S$). However, such a blocking is not necessary for τ_4, because no task in S_4 uses R_1. A similar situation can occur if S_4 does not use global resources. In the described example, only a single task experiences the unnecessary blocking, but in a general system configuration several served tasks could be affected.

To avoid the blocking situation described above, a new preemption test is proposed in this section to manage preemptions among servers. The proposed improvement allows global preemption when a server S_k has $\pi_k^S = \Pi^G$ and its task set Γ_k does not use resources that are currently locked. Note that the improvement is effective only when two or more servers have the same preemption level, like in the example presented above, hence it can be disabled (to save runtime overhead) whenever it is not needed.

Considering such a new rule, the new SRP-G preemption test can then be formally expressed as follows.

A server S_k can preempt another server only if one of the following conditions is verified:

$$\left\| \begin{array}{l} \pi_k^S > \Pi^G \\ (\pi_k^S = \Pi^G) \wedge (\nexists R_j \text{ used by } S_k \mid R_j \text{ is locked }). \end{array} \right.$$

6.1 Global schedulability analysis

The SRP-G extension introduced above affects the calculation of the blocking time due to global resources, hence it affects the global schedulability analysis.

A set of reservation servers $S_1, \ldots, S_m$ can be scheduled under an EDF-based global scheduler (such as IRIS [19]) if[1]

$$\forall k = 1, \ldots, m \quad \sum_{hep(i)} \alpha_i + \frac{B_k^G}{P_k} \leq 1 \tag{14}$$

where $hep(i)$ denotes the set of servers with period higher than or equal to P_k and B_k^G is the blocking factor of server S_k. Note that the improvement proposed in this paper for SRP-G can reduce the blocking factor with respect to the classical formulation.

In particular, the maximum blocking time that server S_k can experience is equal to the maximum resource-holding-time among all the servers S_ℓ with period $P_\ell > P_k$ that share a global resource with some server S_h with period $P_h < P_k$, or with period $P_h = P_k$ when S_ℓ shares a resource with S_k. Formally, B_k^G can be expressed as follows:

$$B_k^G = \max_{P_\ell > P_k} \{H_{\ell,j} \mid R_j \text{ used by } S_h \in \Omega(k,j)\}_0 \tag{15}$$

with

$$\Omega(k,j) = \{S_h \mid (P_h < P_k) \vee ((P_h = P_k) \wedge R_j \text{ used by } S_k)\}.$$

7 Experimental results

This section presents a set of experiments carried out to evaluate the performance of the new local schedulability test proposed for BROE, under local EDF and FP, for different configuration parameters. Performance results are also compared with the original schedulability test for BROE proposed by Bertogna et al. [15] and the schedulability test for SIRAP proposed by Behnam et. al. [28] in 2010, reformulated by van den Heuvel et al. [23] in 2011.

A comparison with overrun-based approaches [11] is not carried out here, considering that extensive experiments [23] have shown that SIRAP clearly outperforms such methods.

Since we assume global EDF-based scheduling of reservation servers, we use as global schedulability test the same test proposed by Bertogna et al. in [15].

Note that, in the performance study reported by van den Heuvel et al. [23], the BROE schedulability is tested by using the original analysis based on the linear bound α-Δ.

In all the experiments, the performance of each resource sharing algorithm is evaluated by measuring the ratio of the number of feasible task sets and the total number N of randomly generated task sets, for a given configuration parameter. In each graph, for each value of the configuration parameter the ratio is computed over $N = 2500$ task sets.

7.1 Task set generation

Given m servers with a total system utilization U, server parameters are generated as follows:

- server utilization U_k is randomly generated by the UUniFast algorithm [29], which guarantees a uniform distribution, limiting the minimum server utilization to U_S^{min}.
- server budget Q_k is randomly generated with uniform distribution in a given range $[Q^{min}, Q^{max}]$;
- server period P_k is computed as $P_k = Q_k / U_k$.

Within each subsystem S_k, the task set Γ_k, consisting of n tasks, is generated with a total utilization $\Psi_k U_k$, where Ψ_k represents the application load normalized with respect to the server bandwidth. Note that such a normalized load represents a crucial parameter affecting the performance of the schedulability test.

The other task set parameters are generated as follows:

- the task utilization U_i is randomly generated by UUniFast;
- the task period T_i (or minimum interarrival time) is randomly generated with uniform distribution in a given range $[T_k^{min}, T_k^{max}]$, set for the considered server S_k;
- the task WCET C_i is computed as $C_i = T_i U_i$;
- the relative deadline D_i is randomly generated with uniform distribution in $[C_i + \beta(T_i - C_i), T_i]$, where β is a parameter of the algorithm, such that $0 \leq \beta \leq 1$.

Considering that the extra delay problem highlighted in Section 1 is not caused by local resources, and that the protocols compared in this study are explicitly designed to solve such a problem, only global resources are taken into account. Global resources are randomly assigned to tasks following an exponential distribution, in order to simulate a more realistic resource sharing among tasks. In this way we

1. Note that the global feasibility test can also be performed using the processor demand criterion extended under resource sharing by Baruah [21].

obtain a high probability that a resource is shared among a small number of tasks and, viceversa, a low probability that is shared by a high number of tasks.

To simplify the computation of $H_{k,j}(i)$ and the local blocking time $B_k^L(t)$, global resources are accessed using a non-preemptive scheme. For each global resource R_j, the corresponding resource holding time $H_{k,j}$ is randomly generated with uniform distribution in the range $[H^{min}, H^{max}]$. Since both BROE and SIRAP require $H_k < Q_k$, H^{min} and H^{max} are computed as a fraction of the actual minimum budget Q^* deriving from the random generation of the server parameter:

$$Q^* = \min_k \{Q_k\}.$$

During our experiments we noticed that some parameters do not significantly affect the schedulability ratio, so we decided not to present the corresponding experiments. In all the experiments reported below, all fixed parameters are reported in Table 4 and the following notation is used to refer to the compared algorithms:

BROE-$\alpha\Delta$	Original schedulability test for BROE based on the linear supply bound function $\mathsf{sbf}^L(t)$, proposed by Bertogna et al. [15].
BROE	New test for BROE presented in Section 4 (for EDF) and Section 5 (for FP).
SIRAP	Schedulability test for SIRAP [23], [28] proposed by Behnam et. al. in 2010.

For each scheduling policy (EDF and FP) we performed three experiments: Table 3 reports a short description for each experiment.

Experiment 1	Schedulability as a function of the application load.
Experiment 2	Schedulability as a function of the maximum average resource holding time.
Experiment 3	Schedulability as a function of the application load varying the number of global resources.

Table 3: Descriptions of the performed experiments.

Number of servers	$m = 5$
Total utilization	$U = 0.8$
Minimum budget	$Q^{min} = 300$
Maximum budget	$Q^{max} = 1000$
Minimum server utilization	$U_S^{min} = 0.08$
Number of tasks per server	8
Deadline constraint	$\beta = 1$

Table 4: Fixed parameters used in all experiments.

7.2 Experiments under local EDF

7.2.1 Experiment 1

In the first experiment we tested the schedulability ratio of feasible task sets as a function of the normalized application load Ψ, varied in the range [0.25, 1], with step 0.05, for all the servers. In addition, the number of global resources is set to 5 and task periods are varied between $T_k^{min} = 2P_k$ and $T^{max} = 12P_k$. Note, in fact, that applications including tasks with a period smaller than $\Delta_k = 2(P_k - Q_k)$ cannot be guaranteed on the reservation server S_k.

In order to show the influence of the resource holding time $H_{k,j}$ on the performance of the algorithms, three different graphs are reported in Figure 5 for different ranges $[H^{min}, H^{max}]$.

As it is clear from the graphs, the new schedulability test for BROE proposed in this paper always outperforms the other two tests for all configuration parameters. For very small critical sections (see Figure 5a), SIRAP performs better than BROE-$\alpha\Delta$, whereas for medium critical sections (see Figure 5b) BROE shows a significant improvement with respect to SIRAP. For instance, for $\Psi = 0.6$, BROE schedules three times more task sets than SIRAP.

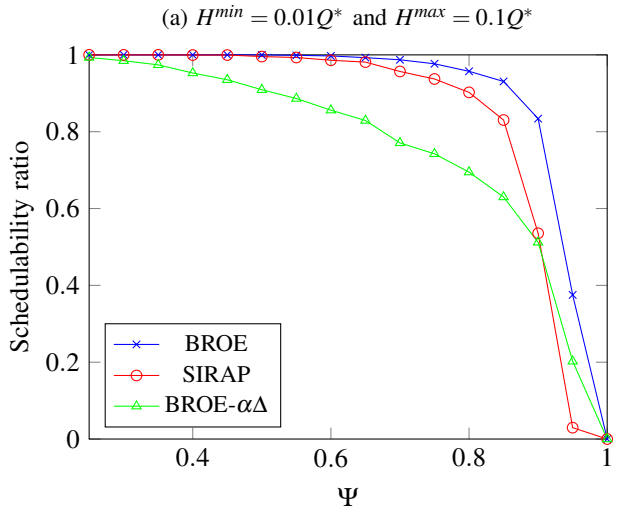
It is worth observing that, as critical sections get larger, all the algorithms tend to degrade (see Figure 5c), but SIRAP degrades more quickly, and for application loads higher than 60% it is not able to guarantee a significant load, while BROE shows a more graceful degradation.

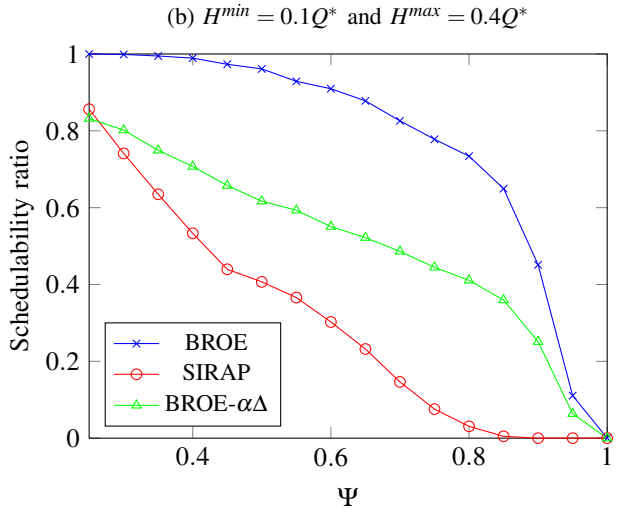
7.2.2 Experiment 2

To better illustrate the dependency of the tests on the resource holding time $H_{k,j}$, we carried out another experiment by monitoring the schedulability ratio as a function of the average maximum resource holding time normalized with respect to Q^*, that is $\overline{H}/Q^*$, setting $H^{max} - H^{min} = 0.2Q^*$, $\Psi = 0.6$, and 5 global resources. Note that, as $\overline{H}/Q^*$ increases, the average number of critical sections that can be inserted in the task code decreases. To limit the effect of such a phenomenon, in this experiment we varied the task periods in a larger interval by setting $T_k^{min} = 2P_k$ and $T^{max} = 16P_k$, so obtaining tasks with higher worst-case execution times.

The results shown in Figure 6 confirm the observed degradation. Again, SIRAP degrades more quickly, and for $\overline{H}/Q^* = 0.4$ it guarantees only 20% of the load, while BROE reaches almost 80%.

It is worth observing that, although the $\mathsf{sbf}^P(t)$ used by SIRAP does not reduce while increasing $\overline{H}$, the observed degradation is due to the self-blocking phenomenon which significantly increases the blocking term [14]. On the other hand, BROE degradation is due to the cropping of its $\mathsf{sbf}^B(t)$, as explained in Section 4. As a consequence, it appears that self-blocking has a higher negative impact on local schedulability than the cropping effect present in BROE. Note that this behavior becomes even more apparent when global resources are accessed without using a non-preemptive scheme, causing larger blocking times that significantly penalize SIRAP.





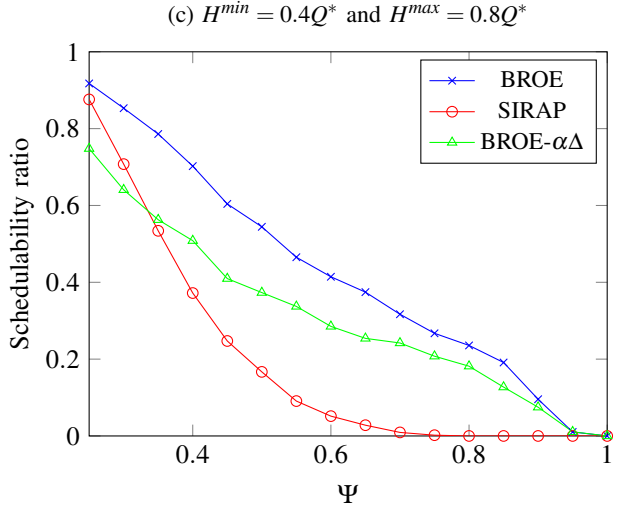


Figure 5: Schedulability under local EDF as a function of Ψ, for different values of H^{min} and H^{max}.

7.2.3 Experiment 3

A final experiment has been carried out to show the dependency of the schedulability tests on the number of global resources used by the task set. Here, resource holding times have been generated using the medium case, with $H^{min} = 0.1Q^*$ and $H^{max} = 0.4Q^*$, and task periods have been varied between $T_k^{min} = 2P_k$ and $T^{max} = 12P_k$. The number of global resources has been set to 1, 5, and 10. The case with 5 global resources has already been shown in Figure 5b, while the other two cases are reported in Figure 7. Although all the tests exhibit a degradation as the number of global resources increases, the SIRAP test degrades more significantly, while the new test proposed for BROE is much less sensitive to such a variation.

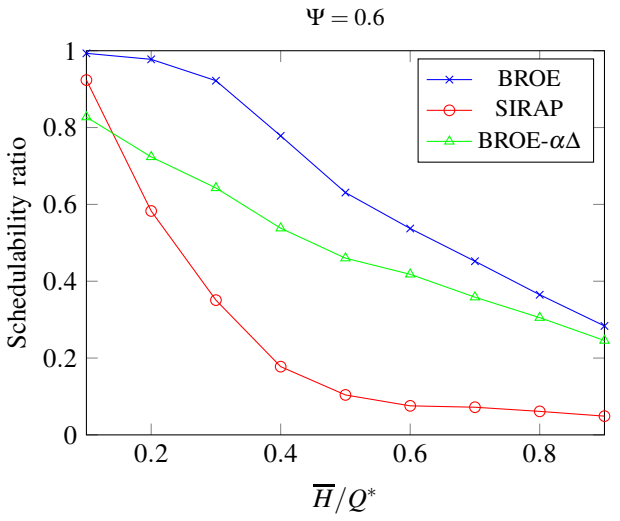


Figure 6: Schedulability under local EDF as a function of the average value $\overline{H}/Q^*$, for $\Psi = 0.6$.

7.3 Experiments under local FP

This section compares the performance of BROE, BROE-$\alpha\Delta$ and SIRAP schedulability tests under local FP scheduling. Since in the original BROE paper [15] the local schedulability analysis was derived only under EDF, we extended the BROE-$\alpha\Delta$ test by replacing $\mathrm{sbf}_i^B(S,t)$ with $\mathrm{sbf}^L(S,t)$ in Equation (13) in order to make a comparison under local FP.

7.3.1 Experiment 1

The Experiment 1 made under EDF has been repeated under FP with the same configuration parameters: results are reported in Figure 8. The graphs show that, under FP, SIRAP performs better than BROE-$\alpha\Delta$ with respect to the EDF case, while the new BROE schedulability test proposed in this paper again outperforms the other two tests, confirming the relevance of our contribution.

7.3.2 Experiment 2

Experiment 2 has been repeated under FP with different configuration parameters. In fact, as Ψ increases, the performance under local FP degrades more quickly than under EDF. We therefore ran the experiment setting $\Psi = 0.5$ (instead of $\Psi = 0.6$). To allow inserting a sufficient number of critical sections in the tasks, periods have been generated

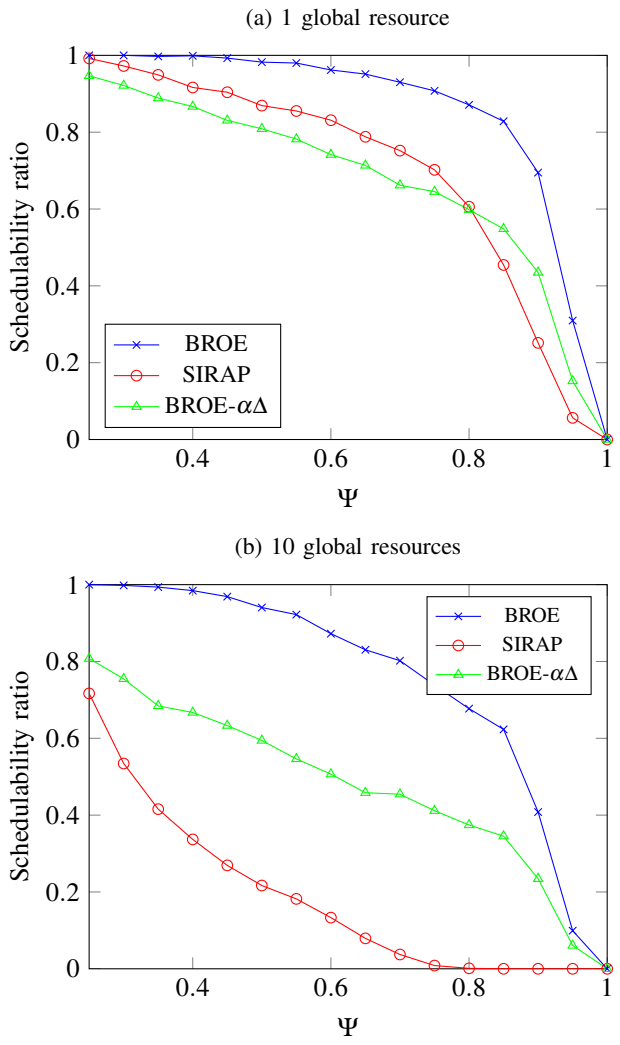


Figure 7: Schedulability under local EDF as a function of Ψ, for different numbers of global resources.

using $T_k^{min} = 2P_k$ and $T^{max} = 18P_k$, while resource holding times have been generated using the same constraints as in the EDF case. The number of global resources has been also kept to 5. Results are reported in Figure 9. In this case, SIRAP outperforms BROE-$\alpha\Delta$ for all values of $\overline{H}/Q^*$, but it is always dominated by the improved BROE.

7.3.3 Experiment 3

A last experiment has been carried out under FP to show the dependency of the schedulability tests on the number of global resources, using the same parameter ranges used in the third experiment for EDF. The number of global resources has been set to 1, 5, and 10. The case with 5 global resources has already been shown in Figure 8b, while the other two cases are reported in Figure 10. The results of this experiment show that, under local FP, both BROE and SIRAP are less sensitive to the number of global resources with respect to the EDF case, although BROE still dominates the other two tests for all ranges of parameters.

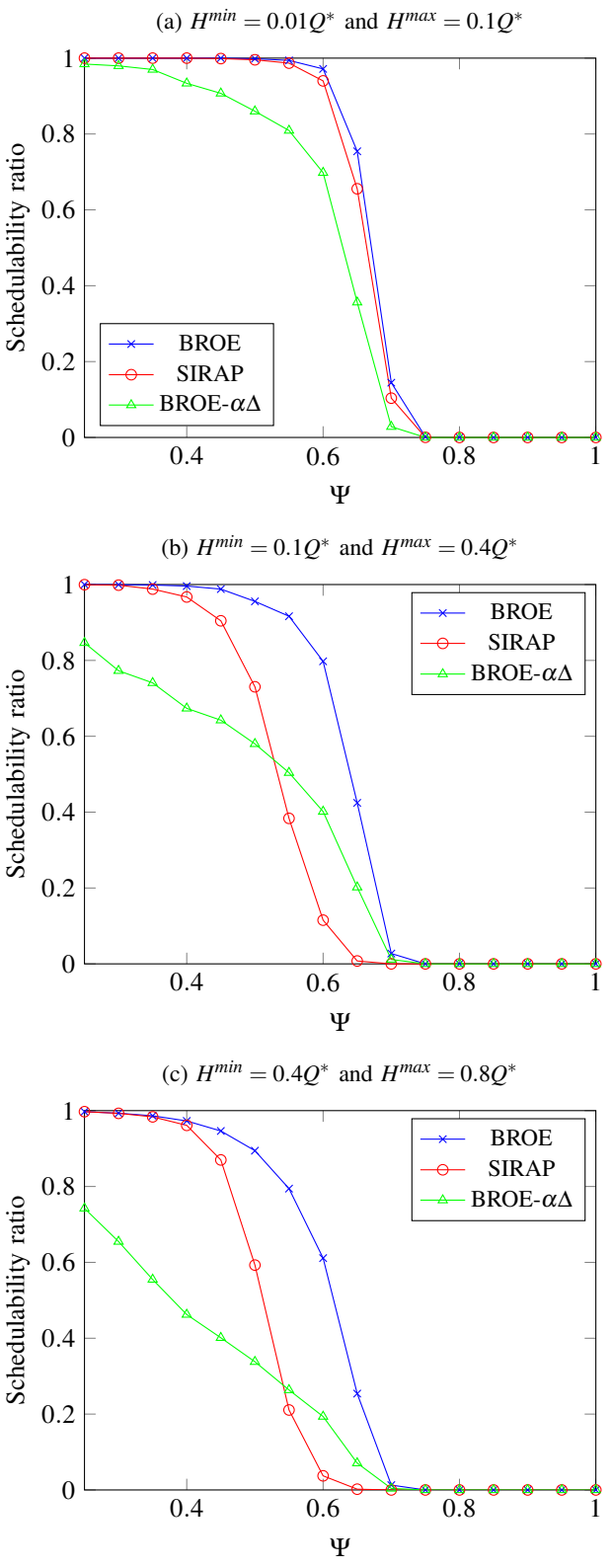


Figure 8: Schedulability under local FP as a function of Ψ, for different values of H^{min} and H^{max}.

8 Implementation Issues

This section discusses some guidelines related to the implementation of the two resource sharing protocols SIRAP

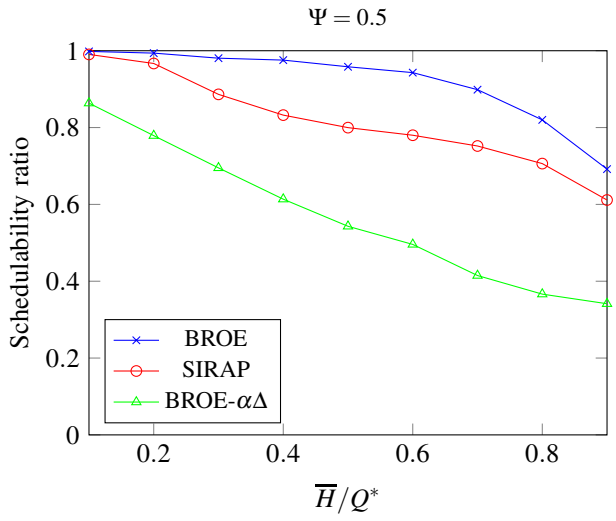
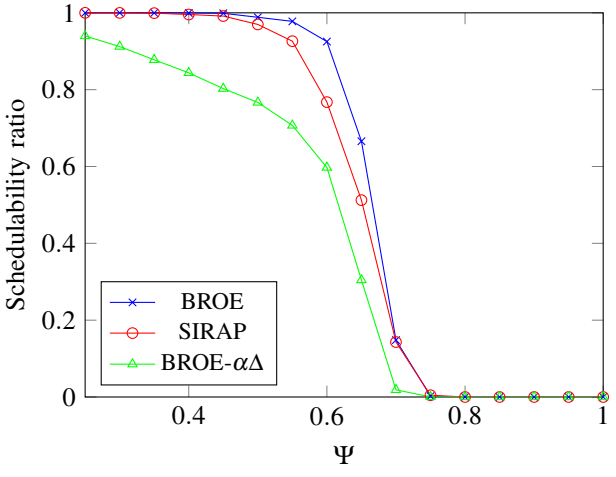


Figure 9: Schedulability under local FP as a function of the average value $\overline{H}/Q^*$, for $\Psi = 0.5$.

(a) 1 global resource



(b) 10 global resources

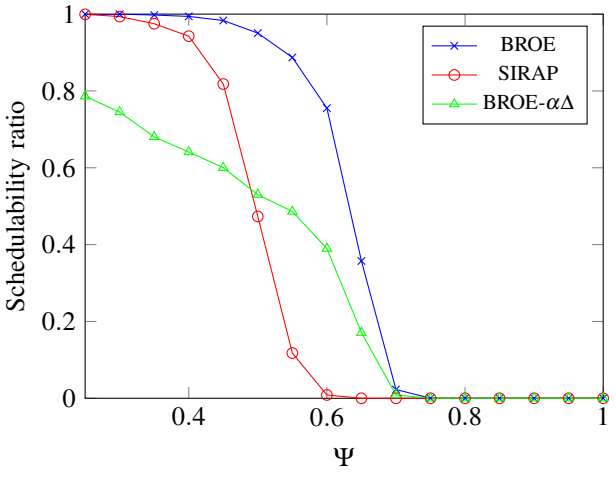


Figure 10: Schedulability under local FP as a function of Ψ, for different numbers of global resources.

and BROE. These protocols have been implemented on the ERIKA Enterprise real-time kernel [30], using the Hard-CBS [19], [20] as a global scheduler. The Hard-CBS is a bandwidth preserving server algorithm that has been extended in this work to support resource sharing.

It is worth noting that both SIRAP and BROE have also been implemented on the μC/OS-II operating system by van den Heuvel et al. [31]. However, their SIRAP implementation relies on a periodic-idling server algorithm as global scheduler, which discharges the budget every time the server becomes idle. Viceversa, bandwidth preserving servers preserve the server budget as long as possible without violating the guaranteed reservation bandwidth. Due to such a difference, the considerations in [31] on the implementation complexity do not apply to bandwidth preserving server algorithms like the Hard-CBS. In summary, the implementation comparison presented in [31] considers different global schedulers for BROE and SIRAP, hence the reported results do not represent a consistent evaluation between these two resource sharing protocols.

The implementation carried out for this work has shown that BROE can be easily implemented on a bandwidth preserving server, because it does not require additional data structures in the kernel and its behavior is realized just by acting on two state variables: the current budget q_k and the absolute deadline d_k. In particular, the BROE suspension required by Rule 4-a can be implemented in a transparent fashion without modifying the Hard-CBS implementation: when a server experiences such a suspension, its remaining budget q_k can be discarded and the Recharging Time defined by the Hard-CBS can be set to $t_r = d_k - q_k/\alpha_k$. The same considerations are valid for IRIS [19], which is a Hard-CBS algorithm that also includes a reclaiming mechanism for exploiting available idle times.

On the other hand, SIRAP requires specific data structures, one for each server, to manage self-blocked tasks. In terms of implementation, handling such data structures increases both the kernel footprint and the runtime overhead. As a side note, implementing SIRAP over the Hard-CBS implies an additional complication, needed for handling a situation in which all served tasks τ_i that are active and eligible for execution ($\pi_i > \Pi^L$) experience a self-blocking. This situation is not compliant with the Hard-CBS rules, because the Hard-CBS does not handle a server with active tasks that is waiting for a budget replenishment. Solving such an inconsistency requires additional coding and timer operations that introduce extra overhead.

9 Conclusions

In this paper we presented two local schedulability tests (one under local FP and one under local EDF schedulers) to verify the schedulability of real-time applications in a two-level hierarchical system under an EDF-based global scheduler, where resource sharing among reservations is performed by the BROE service algorithm. A simple extension of the global SRP protocol has been also proposed to possibly reduce the blocking time of the servers while accessing global resources in certain conditions.

The performance of the new BROE schedulability tests has been compared with the ones of SIRAP and the original

test proposed for BROE based on the linear bound α-Δ. Since both algorithms have been implemented on the Erika Enterprise real-time kernel, practical issues related to implementation complexity and runtime overhead have also been discussed for bandwidth preserving servers.

Experimental results showed that the new BROE tests outperform the others for all configuration parameters. For several configuration parameters, the new BROE test is able to accept 2-3 times more task sets than the older BROE test and up to 8 times more than SIRAP.

Although the schedulability of all tests degrades as critical sections get larger, BROE exhibits a more graceful degradation with respect to the other tests, making it the best choice for implementing reservations in hierarchical systems under global resource sharing. This choice is also justified by the existence of an optimal design algorithm [17] that exploits the novel analysis methodology proposed in this paper. The design method allows computing the reservation parameters that minimize the server bandwidth for guaranteeing the schedulability of the real-time application. A proof-of-concept implementation of the server design algorithm is freely available on [18].

REFERENCES

[1] M. D. Natale and A. S. Vincentelli, "Moving from federated to integrated architectures in automotive: The role of standards, methods and tools," *Proc. of the IEEE*, vol. 98, no. 4, pp. 603–620, April 2010.

[2] C. W. Mercer, S. Savage, and H. Tokuda, "Processor capacity reserves for multimedia operating systems," in *Proc. of IEEE international conf. on Multimedia Computing and System*, May 1994.

[3] L. Abeni and G. Buttazzo, "Resource reservations in dynamic real-time systems," *Real-Time Systems*, vol. 27, no. 2, pp. 123–165, 2004.

[4] E. Bini, G. Buttazzo, J. Eker, S. Schorr, R. Guerra, G. Fohler, K.-E. Arzen, V. R. Segovia, and C. Scordino, "Resource management on multicore systems: The ACTORS approach," *IEEE Micro*, vol. 31, no. 3, pp. 72–81, May-June 2011.

[5] I. Shin and I. Lee, "Compositional real-time scheduling framework," in *Proceedings of the 25th IEEE Real-Time Systems Symposium*, Lisbon, Portugal, December 5-8, 2004, pp. 57–67.

[6] C. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the Association for Computing Machinery*, vol. 20, no. 1, pp. 46–61, January 1973.

[7] G. Lipari and S. Baruah, "A hierarchical extension to the constant bandwidth server framework," in *Proceedings of the 7th Real-Time Technology and Application Symposium*, Taipei, Taiwan, June 2, 2001, pp. 26–35.

[8] L. Sha, R. Rajkumar, and J. Lehoczky, "Priority inheritance protocols: An approach to real-time synchronization," *IEEE Transactions on Computers*, vol. 39, no. 9, pp. 1175–1185, September 1990.

[9] T. P. Baker, "Stack-based scheduling for realtime processes," *Real-Time Systems*, vol. 3, no. 1, pp. 67–99, April 1991.

[10] T. Ghazalie and T. Baker, "Aperiodic servers in a deadline scheduling environment," *Real-Time Systems*, vol. 9, 1995.

[11] R. I. Davis and A. Burns, "Resource sharing in hierarchical fixed priority pre-emptive systems," in *Proceedings of the IEEE Real-time Systems Symposium (RTSS 2006)*, Rio de Janeiro, Brazil, December 5-8, 2006, pp. 257–268.

[12] M. Behnam, I. Shin, T. Nolte, and M. Nolin, "Scheduling of semi-independent real-time components: Overrun methods and resource holding times," in *Proceedings of 13th IEEE Int. Conference on Emerging Technologies and Factory Automation (ETFA'08)*, Hamburg, Germany, September 15-18, 2008.

[13] M. Behnam, T. Nolte, , M. Sjödin, and I. Shin, "Overrun methods and resource holding times for hierarchical scheduling of semi-independent real-time systems," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 1, pp. 93–104, February 2010.

[14] M. Behnam, I. Shin, T. Nolte, and M. Nolin, "SIRAP: a synchronization protocol for hierarchical resource sharing in real-time open systems," in *Proceedings of the 7th ACM & IEEE International Conference on Embedded Software (EMSOFT 2007)*, Salzburg, Austria, October 1Ű3, 2007.

[15] M. Bertogna, N. Fisher, and S. Baruah, "Resource-sharing servers for open environments," *IEEE Transactions on Industrial Informatics*, vol. 5, no. 3, pp. 202–219, August 2009.

[16] T.-W. Kuo and C.-H. Li, "A fixed prioriy driven open environment for real-time applications," in *Proc. of the IEEE Real-Time Systems Symposium*, Phoenix, AZ, USA, December 1-3, 1999, pp. 256–267.

[17] A. Biondi, A. Melani, M. Bertogna, and G. Buttazzo, "Optimal design for reservation servers under shared resources," in *Proceedings of the 26th Euromicro Conference on Real-Time Systems (ECRTS 14)*, Madrid, Spain, 9-11 July, 2014.

[18] *A MATLAB® optBROE algorithm implementation*, http://retis.sssup.it/∼a.biondi/optBROE.

[19] L. Marzario, G. Lipari, P. Balbastre, and A. Crespo, "IRIS: A new reclaiming algorithm for server-based real-time systems," in *Proc. of the IEEE Real-Time and Embedded Technology and Applications Symposium*, Toronto, Canada, May 2004.

[20] A. Biondi, A. Melani, and M. Bertogna, "Hard constant bandwidth server: Comprehensive formulation and critical scenarios," in *Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES 2014)*, Pisa, Italy, 18-20 June, 2014.

[21] S. Baruah, "Resource sharing in EDF-scheduled systems: a closer look," in *Proceedings of the 27th IEEE Real-Time Systems Symposium (RTSS'06)*, Rio de Janeiro, Brazil, December 5-8, 2006.

[22] A. Easwaran, M. Anand, and I. Lee, "Compositional analysis framework using EDP resource models," in *Proc. of the 28th IEEE Real-Time Systems Symposium (RTSS 2007)*, Tucson, Arizona, USA, December 3-6, 2007.

[23] M. van den Heuvel, M. Behnam, R. J. Bril, J. Lukkien, and T. Nolte, "Opaque analysis for resource sharing in compositional real-time systems," in *4th Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS'11)*, November 2011.

[24] S. Baruah and A. Burns, "Sustainable scheduling analysis," in *Proceedings of the IEEE Real-time Systems Symposium (RTSS 2006)*, Rio de Janeiro, Brasil, December 5-8, 2006.

[25] M. Bertogna, N. Fisher, and S. Baruah, "Resource holding times: Computation and optimization," *Real-Time Systems*, vol. 41, no. 2, pp. 87–117, February 2009.

[26] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: Exact characterization and average case behavior," in *Proceedings of the 10th IEEE Real-Time Systems Symposium (RTSS'89)*, Santa Monica, CA, USA, December 5-7, 1989, pp. 166–171.

[27] E. Bini and G. C. Buttazzo, "Schedulability analysis of periodic fixed priority systems," *IEEE Transactions on Computers*, vol. 53, no. 11, pp. 1462–1473, 2004.

[28] M. Behnam, T. Nolte, and R. J. Bril, "Bounding the number of self-blocking occurrences of SIRAP," in *Proceedings of the 31st IEEE Real-Time Systems Symposium (RTSS 2010)*, San Diego, California, USA, November 30 - December 3, 2010.

[29] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Systems*, vol. 30, no. 1-2, pp. 129–154, 2005.

[30] P. Gai, G. Lipari, L. Abeni, M. di Natale, and E. Bini, "Architecture for a portable open source real-time kernel environment," in *Proceedings of the Second Real-Time Linux Workshop and Hand's on Real-Time Linux Tutorial*, November 2000.

[31] M. van den Heuvel, R. Bril, and J. Lukkien, "Transparent synchronization protocols for compositional real-time systems," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, pp. 322–336, 2012.