

# 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

## I. 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. For instance, in a cell phone, the user can run several applications at the same time (e.g., a music player, a map visualizer, a browser to download data from the internet, and a sound recorder). These applications are characterized by a highly variable computational demand and share common resources (e.g., processor, memory, touch screen, audio codec, and graphic display). Each application is typically implemented as a set of tasks with different priorities, and tested alone, as a component, on a computing platform under a given scheduling algorithm. When activating more of such applications on the same platform, however, if not properly handled, tasks can experience reciprocal interference and jerky behavior, due to long blocking delays. As a consequence, the user could experience a temporary motion stop on the movie or perceive an annoying jitter in the sound.

Resource reservation mechanisms [1], [2] can be effectively used to isolate the temporal behavior of the applications and limit their reciprocal interference [3]. To do that, the processor is partitioned into a number of reservations, each behaving as a slower virtual processor using a fraction of the full processor bandwidth. The advantage of this approach is that an overrun occurring in an application does not affect the other applications, but has only the effect of postponing the execution of those tasks in which the overrun is generated. Moreover, an 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.

Under Earliest Deadline First (EDF) scheduling, a reservation can efficiently be implemented by a Constant Bandwidth Server (CBS) [2], which has also been extended by Lipari and Baruah [4] to support hierarchical schedulers.

One of the main problems that arises in real-time 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*. To prevent such a problem, a number of protocols have been proposed, both under fixed priority (FP) assignments [5] and EDF scheduling [6]. Since in this work reservation servers are scheduled using the EDF scheduling algorithm [7], resources access is controlled by a slight variation of the Stack Resource Policy (SRP) [6]. 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 two distinct problems due to global resources.

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

**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 1 illustrates a situation in which a high priority task  $\tau_1$  shares a resource with another task  $\tau_2$  handled by a reservation server with budget  $Q_s = 4$  and period  $P_s = 12$ . At time  $t = 3$ ,  $\tau_1$  preempts  $\tau_2$  within its critical section, and at time  $t = 4$  it blocks on the locked resource. When  $\tau_2$  resumes, however, the residual budget is not sufficient to finish the critical section, and  $\tau_2$  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  $\tau_1$ . This example shows that suspending a task holding a resource leads to unacceptably long delays for tasks in other servers using the same resource.

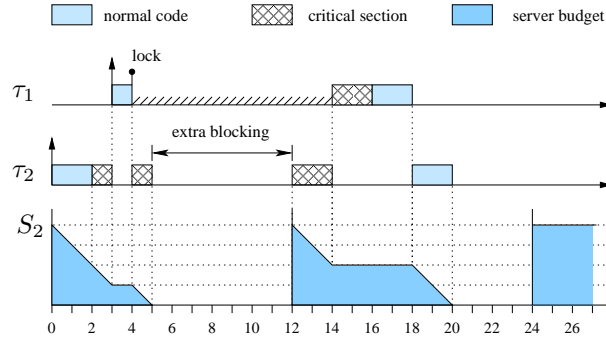


Figure 1: Problem caused when a server budget is exhausted inside a critical section.

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 [8], used by Abeni and Buttazzo under the Constant Bandwidth Server (CBS) [2], analyzed under fixed priorities by Davis and Burns [9] and later extended under EDF by Behnam et al. [10], [11]. 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 [12] 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. [13]. 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 [14], 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 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 [9] 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.

## A. Contributions

This work has five main contributions:

- 1) 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.
- 2) An extension to the HSRP protocol is defined to slightly reduce the blocking time experienced by the servers when accessing global resources shared among components.
- 3) A set of experiments is reported to evaluate the performance of the new tests with respect to the classical test for BROE, for different configuration parameters.
- 4) A comparative evaluation is presented to compare the performance of BROE and SIRAP, under both FP and EDF local scheduling, for different application parameters.
- 5) Since the tested approaches have also been implemented in the ERIKA Enterprise real-time kernel [15], practical implementation issues are also discussed for bandwidth preserving servers.

## B. Paper structure

The remaining of the paper is organized as follows. Section II presents the system model, the terminology, and the assumptions used throughout the paper. Section III briefly recalls the BROE algorithm for handling a reservation with bandwidth  $\alpha$  and maximum delay  $\Delta$  in the presence of shared resources. Section IV 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 V. Section VI introduces an improvement of the HSRP protocol that allows reducing the blocking time due to global resources. Section VII reports two sets of experiments aimed at showing the performance the new schedulability tests for BROE with respect to SIRAP and the original  $\alpha$ - $\Delta$  test, under FP and EDF, for different application parameters. Section VIII presents the implementation work of BROE and SIRAP on an existing RTOS, discussing practical implementation issues. Finally, Section IX states our conclusions and future work.

## II. SYSTEM MODEL

This paper considers a uniprocessor hierarchical system  $\mathcal{S}$  consisting of a number of subsystems  $S_k \in \mathcal{S}$ , each implemented by a BROE [13] 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 [16]. The system is schematically illustrated in Figure 2. The *global scheduler* is implemented by a hard Constant Bandwidth Server [2], [17], a bandwidth preserving mechanism that determines which subsystem can access the CPU at any given time. Each subsystem uses a *local scheduler* to select the running task on the subsystem. In this paper, both EDF and FP are considered as local scheduling policies for each subsystem.

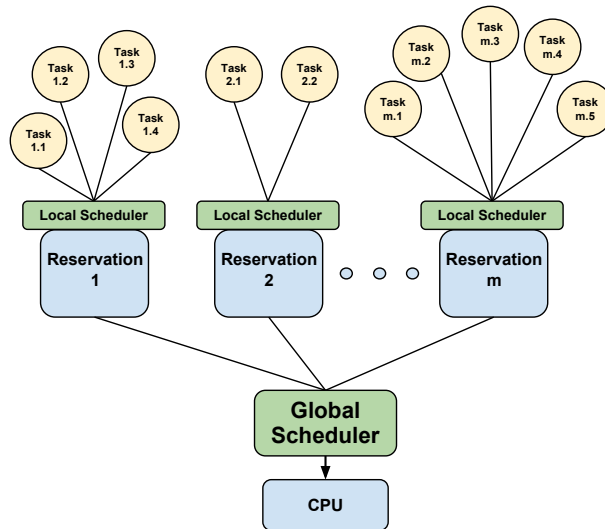


Figure 2: Two-level hierarchical scheduling framework.

### A. Task model

Each subsystem  $S_k$  runs an application  $\Gamma_k$  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  $\tau_i$  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  $\tau_1$  is the highest priority task.

### B. 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,  $Z_{i,j}$  denotes the longest critical section of  $\tau_i$  related to resource  $R_j$  and  $\delta_{i,j}$  denotes the WCET of  $Z_{i,j}$ .

*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  $\tau_i$ .

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

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

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

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

If local preemption is not disabled,  $H_{k,j}(i)$  must take into account the worst-case local interference experienced by  $\tau_i$  during the lock of  $R_j$  (details on how to compute  $H_{k,j}(i)$  can be found in [13]).

In addition, the maximum Resource Holding Time for an application  $\Gamma_k$  is defined as

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

and the *Level- $i$  maximum Resource Holding Time* for  $\Gamma_k$  is defined as

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

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  $\tau_i$  is assigned a local preemption level  $\pi_i$  and preemption levels are ordered inversely with respect to relative deadlines; that is,  $\pi_i > \pi_h \Leftrightarrow D_i < D_h$ . For each local or global resource  $R_j$  is assigned a local ceiling  $C_j^L$  according the classical SRP definition, that is

$$C_j^L = \max\{\pi_i \mid \tau_i \text{ holds local } R_j\}_0.$$

and a subsystem ceiling is defined for each server  $S_k$  as

$$\Pi_k^L = \max\{C_j^L \mid R_j \text{ is local and used by } S_k\}.$$

Then, a task  $\tau_i$  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  $\pi_k^S$  and server preemption levels are ordered inversely with respect to server periods; that is,  $\pi_h^S > \pi_k^S \Leftrightarrow P_h < P_k$ . For each global resource is assigned a global ceiling equal to

$$C_j^G = \max\{\pi_k^S \mid \exists \tau_i \in \Gamma_k \wedge \tau_i \text{ holds global } R_j\}_0$$

and a global system ceiling is defined as

$$\Pi^G = \max_j \{C_j^G\}.$$

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  $\Pi^G$  is incremented and a number of servers is prevented to execute, hence the blocking is extended to all tasks inside them.

Symbol	Description
$S_k$	$k^{th}$ subsystem (reservation server)
$\Gamma_k$	Application (task set) handled by $S_k$
$n_k$	Number of tasks in $\Gamma_k$
$Q_k$	Budget of server $S_k$
$P_k$	Period of server $S_k$
$\alpha_k$	Bandwidth of server $S_k$
$\Delta_k$	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 $\alpha - \Delta$ supply bound function of server $S_k$
$\tau_i$	$i^{th}$ task
$C_i$	Worst-case execution time (WCET) of $\tau_i$
$P_i$	Period (or minimum interarrival time) of $\tau_i$
$D_i$	Relative deadline of $\tau_i$
$R_j$	$j^{th}$ shared resource
$Z_{i,j}$	Longest critical section of $\tau_i$ related to $R_j$
$\delta_{i,j}$	WCET of $Z_{i,j}$
$H_{k,j}(i)$	Level- $i$ RHT of $R_j$ accessed by tasks in $\Gamma_k$
$H_k$	Maximum RHT for $\Gamma_k$
$H_k(i)$	Level- $i$ maximum RHT for $\Gamma_k$
$\pi_i$	Preemption level for $\tau_i$
$C_j^L$	Local resource ceiling for $R_j$
$\Pi_k^L$	Subsystem ceiling for $S_k$
$\pi_k^S$	Preemption level for $S_k$
$C_j^G$	Global resource ceiling for $R_j$
$\Pi^G$	Global system ceiling

Table I: Notation used throughout this paper.

### III. THE BROE SERVER

As stated in Section I, 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  $\tau_i$  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 (3).

If  $\tau_i$  wants to enter a critical section  $Z_{i,j}$  at time  $t$  and  $q < H$ , then a budget depletion may occur inside the critical section. Since  $\tau_i$  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. [13] proposed the BROE server, which is based on a hard Constant Bandwidth Server [17] 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).

#### IV. 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 [16], later extended by Baruah [18] to account for shared resources. According to this test, a task set  $\Gamma$  is schedulable by EDF on a reservation server  $S$  if

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

where  $\text{dbf}(\Gamma, t)$  is the *demand bound function* of the task set  $\Gamma$  (i.e., the maximum computational demand of  $\Gamma$  in any interval of length  $t > 0$ ),  $\text{sbf}(S, t)$  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,

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

In the original paper presented by Bertogna et al. [13], the supply bound function used in the test was actually a linear lower bound  $\text{sbf}^L(S, t)$  of a bounded-delay partition  $(\alpha, \Delta)$ , where  $\alpha$  is the server bandwidth and  $\Delta$  is the maximum service delay:

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

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

In this section we improve the effectiveness of the local schedulability test by deriving the actual supply bound function  $\text{sbf}^B(S, t)$  of BROE as a function of the maximum resource holding time  $H$  of the global resources accessed by  $\Gamma$ . 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.

To derive a more precise supply bound function for a BROE server, let us first consider a scenario in which no global resources are used in the server.

In this case, BROE behaves as a classical hard CBS server (like IRIS [17]), whose supply bound function is described as follows [19], [20]:

$$\text{sbf}^P(t) = \max \left\{ \begin{array}{l} 0, \\ (h(t) - 1)Q, \\ t - (h(t) + 1)(P - Q) \end{array} \right\} \quad (8)$$

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

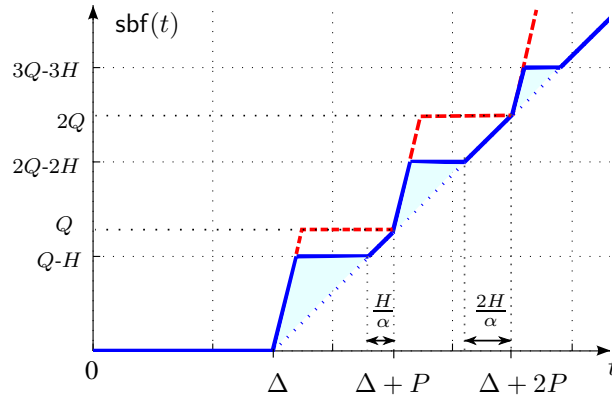


Figure 3: Supply bound functions: periodic (dashed line), linear  $\alpha - \Delta$  (dotted line), and new sbf proposed for BROE (continuous line).

When considering resource sharing, the worst-case supply can be found by considering Rule 4 of BROE in Section III, which reduces  $\text{sbf}^P(t)$  in some time intervals. Note that Rule 2 does not affect the  $\text{sbf}^B(t)$ , 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  $\text{sbf}^P(t)$ . 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  $\text{sbf}^L(t)$  (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  $\text{sbf}^B(t) = \text{sbf}^L(t)$  in the interval  $(\Delta + P - H/\alpha, \Delta + P]$ .

Note that, in the next interval  $[\Delta + P, \Delta + 2P]$ , the reduction of the  $\text{sbf}^B(t)$  with respect to  $\text{sbf}^P(t)$  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  $\text{sbf}^P(t)$  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  $\text{sbf}^L(t)$ .

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. A robust and sustainable [21] analysis should therefore consider the case in which the global lock is released an infinitesimal time after the budget replenishment and another lock request is made  $Q - H$  time-units after that, as correctly considered in the above discussion.

In general, the reduction of the  $\text{sbf}^B(t)$  with respect to  $\text{sbf}^P(t)$  increases period by period, until it reduces to  $\text{sbf}^L(t)$ . Considering the  $k^{\text{th}}$  period after  $\Delta$ , that is,

$$k = \max \left\{ 1, \left\lfloor \frac{t - \Delta}{P} \right\rfloor + 1 \right\}, \quad (9)$$

the intervals in which  $\text{sbf}^B(t) = \text{sbf}^L(t)$  are

$$\left[ \Delta + kP - \frac{kH}{\alpha}, \Delta + kP \right]. \quad (10)$$

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

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

which gives  $k \geq Q/H$ . Therefore, after  $\lceil Q/H \rceil$  periods, the supply bound function cannot be larger than  $\text{sbf}^L(t)$ . Note that the smaller  $H$  with respect to  $Q$ , the larger the improvement with respect to  $\text{sbf}^L(t)$ . In the extreme case in which no global resource is shared, we have  $H = 0$  and  $\text{sbf}^B(t)$  coincides with the supply bound function  $\text{sbf}^P(t)$  of a periodic server. Conversely, when  $H = Q$ ,  $\text{sbf}^B(t)$  is always equal to  $\text{sbf}^L(t)$ .

Figure 4 shows the  $\text{sbf}^B(t)$  in the  $k^{\text{th}}$  period after  $\Delta$ , for  $k < Q/H$ . The values of the timing parameters used in the figure are reported in Table II.

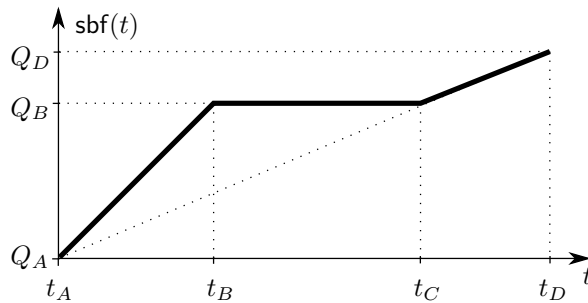


Figure 4:  $\text{sbf}^B(t)$  in the  $k^{\text{th}}$  period after the service delay.

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

Table II: Values for Figure 4.

Hence, the BROE supply bound function can be formally expressed as follows:

$$\text{sbf}^B(t) = \begin{cases} 0 & t \leq \Delta \\ t - \Delta - (k-1)(P-Q) & t_A < t \leq t_B \\ kQ - kH & t_B < t \leq t_C \\ \alpha(t - \Delta) & t_C < t \leq t_D \end{cases} \quad (11)$$

The BROE supply bound function defined above 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  $\text{sbf}^L(t)$ . Such areas are exploited by the schedulability test in Equation (5) for some application parameters.

Note that the proposed improvement does not affect the computational complexity of the schedulability test, which remains pseudo-polynomial as the original test based on the linear  $\text{sbf}^L(t)$ . In fact, computing the value of the novel supply bound function for a given time  $t$  just requires identifying the  $k^{\text{th}}$  period by Equation (9) and then computing the  $\text{sbf}^B(t)$  value by Equation (11). Hence, the total computation requires a modulo operation and a few additions and multiplications.

Also, the new proposed method preserves the modularity of the 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 [13]. Such a modular approach scales well with the number of tasks and servers, allowing an efficient integration of multiple applications in an open environment.

Finally, it is also worth observing that the schedulability test based on the improved supply bound function  $\text{sbf}^B(t)$  is strictly related to the value of  $H$ , improving with smaller values of  $H$ . Different methods have been proposed in the literature to reduce resource holding times by limiting (or disabling) local preemption when accessing global resources [22]. In particular, all critical sections having a length smaller than  $\Delta/2$  can be executed non-preemptively by BROE (Theorem 7 in [13]), 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 section.

## V. 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. [23] considering that, in any test interval  $[0, t]$ , only a fraction of time given by  $\text{sbf}(S, t)$  is available for the application. Hence, a task set  $\Gamma = \{\tau_i, \dots, \tau_n\}$  is schedulable under FP on a reservation server  $S$  if

$$\forall i = 1, \dots, n \exists t \in \text{tSet}_i \quad \text{rbf}_i(t) + B_i^L \leq \text{sbf}(S, t)$$

where  $\text{tSet}_i$  is the set of test points [23], [24], relative to task  $\tau_i$ , where the schedulability check has to be performed, and

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

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

As explained in Section IV, the  $\text{sbf}^B(S, t)$  for a BROE server depends on resource holding times in  $\Gamma$ : 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  $\tau_i$ , 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 (4).

Hence, to have a more precise schedulability test, the BROE supply bound function presented in Equation (11) has to be redefined by introducing the *Level- $i$*  supply bound function  $\text{sbf}_i^B(S, t)$ , obtained by replacing  $H$  with  $H(i)$ . Using such a refinement, a task set  $\Gamma$  is schedulable by FP on a BROE server if

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

## VI. 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  $\tau_i$ . A global resource  $R_1$  is shared between  $\tau_1$  and  $\tau_3$ , and another global resource  $R_2$  is shared between  $\tau_2$  and  $\tau_4$ , as illustrated in Figure 5.



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  $C_j^G = \pi_3^S = \pi_4^S$ . Suppose that  $\tau_1$  starts executing and locks  $R_1$ . After the lock operation, the global system ceiling will be  $\Pi^G = \pi_3^S$ . Hence, if  $\tau_4$  arrives when  $\tau_1$  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  $\tau_4$ , 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.

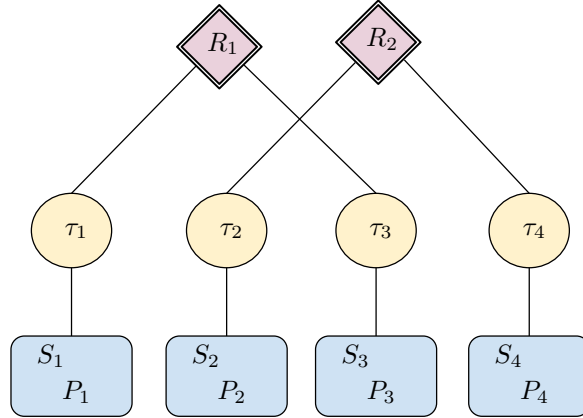


Figure 5: Example with global resources.

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  $\Gamma_k$  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 in Figure 5, 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:*

$$\left\| \begin{array}{l} \pi_k^S > \Pi^G \\ (\pi_k^S = \Pi^G) \wedge (\nexists R_j \text{ used by } S_k \mid C_j^G > 0). \end{array} \right.$$

#### A. 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  $S_1, \dots, S_m$  can be scheduled under an EDF-based global scheduler (such as IRIS [17]) if<sup>1</sup>

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

where  $\text{hep}(i)$  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  $S_\ell$  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  $S_\ell$  shares a resource with  $S_k$ . Formally,  $B_k^G$  can be expressed as follows:

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

with

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

<sup>1</sup>Note that the global feasibility test can also be performed using the processor demand criterion extended under resource sharing by Baruah [18].

## VII. 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. [13] and the schedulability test for SIRAP proposed by Behnam et. al. [25] in 2010, reformulated by van den Heuvel et al. [20] in 2011.

A comparison with overrun-based approaches [9] is not carried out here, considering that extensive experiments [20] 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 [13].

Note that, in the performance study reported by van den Heuvel et al. [20], the BROE schedulability is tested by using the original analysis based on the linear bound  $\alpha\text{-}\Delta$ .

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.

### A. 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 [26], 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  $\Gamma_k$ , consisting of  $n$  tasks, is generated with a total utilization  $\Psi_k U_k$ , where  $\Psi_k$  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  $\beta$  is a parameter of the algorithm, such that  $0 \leq \beta \leq 1$ .

For FP local scheduling policy we assume a deadline-monotonic priority assignment. Considering that the extra delay problem highlighted in Section I 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.

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:

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

In all the experiments reported below, all fixed parameters are reported in Table III and the following notation is used to refer to the compared algorithms:

### B. Experiments under local EDF

1) *Experiment 1*: In the first experiment we tested the schedulability ratio of feasible task sets as a function of the normalized application load  $\Psi$ , 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  $T_k^{max} = 12P_k$ . 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  $S_k$ .

<i>BROE-<math>\alpha\Delta</math></i>	Original schedulability test for BROE based on the linear supply bound function $\text{sbf}^L(t)$ , proposed by Bertogna et al. [13].
<i>BROE</i>	New test for BROE presented in Section IV (for EDF) and Section V (for FP).
<i>SIRAP</i>	Schedulability test for SIRAP [20], [25] proposed by Behnam et. al. in 2010.

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 III: Fixed parameters used in all experiments.

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 6 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 6a), SIRAP performs better than BROE- $\alpha\Delta$ , whereas for medium critical sections (see Figure 6b) BROE shows a huge improvements 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 6c), 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.

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  $\bar{H}/Q^*$ , setting  $H^{\max} - H^{\min} = 0.2Q^*$ ,  $\Psi = 0.6$ , and 5 global resources. Note that, as  $\bar{H}/Q^*$  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  $T_k^{\max} = 16P_k$ , so obtaining tasks with higher worst-case execution times.

The results shown in Figure 7 confirm the observed degradation. Again, SIRAP degrades more quickly, and for  $\bar{H}/Q^* = 0.4$  it guarantees only 20% of the load, while BROE reaches almost 80%.

It is worth observing that, although the  $\text{sbf}^P(t)$  used by SIRAP does not reduce while increasing  $\bar{H}$ , the observed degradation is due to the self-blocking phenomenon which significantly increases the blocking term [12]. On the other hand, BROE degradation is due to the cropping of its  $\text{sbf}^B(t)$ , as explained in Section IV. 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.

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  $T_k^{\max} = 12P_k$ . 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 6b, while the other two cases are reported in Figure 8. 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.

### C. 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 [13] the local schedulability analysis was derived only under EDF, we extended the BROE- $\alpha\Delta$  test by replacing  $\text{sbf}_i^B(S, t)$  with  $\text{sbf}_i^L(S, t)$  in Equation (13) in order to make a comparison under local FP.

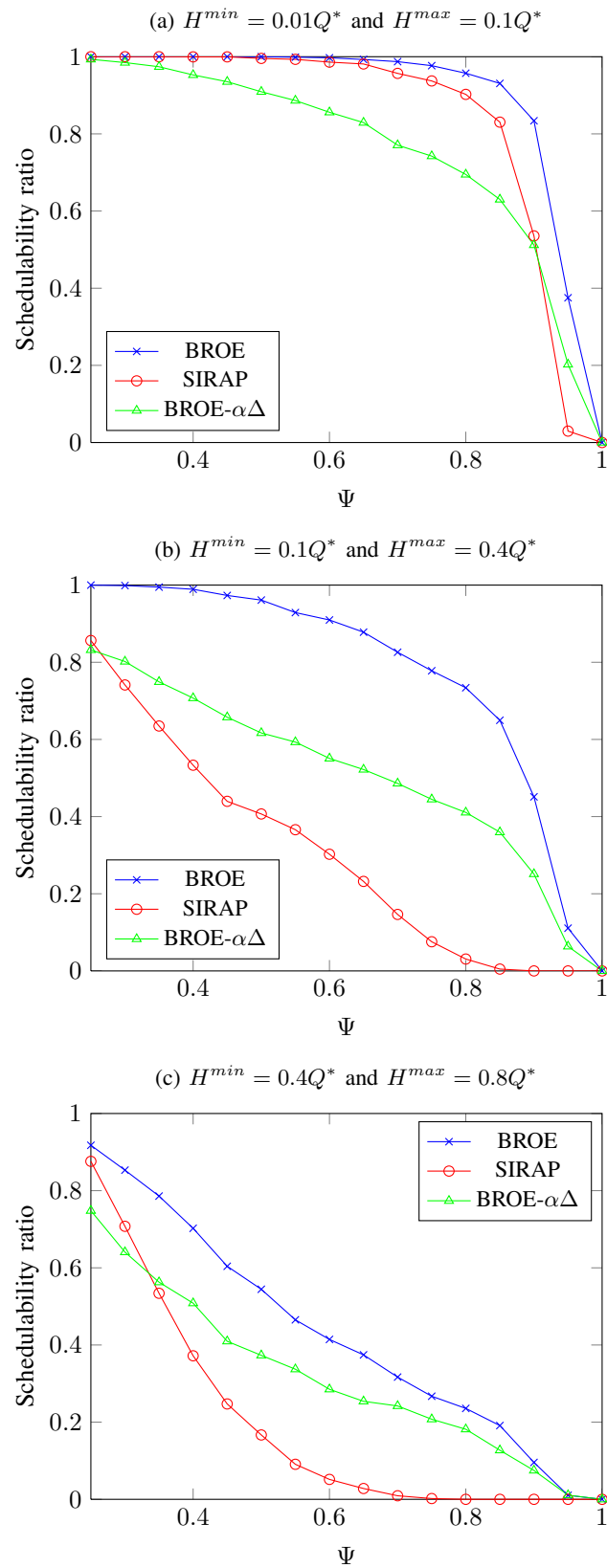


Figure 6: Schedulability under local EDF as a function of  $\Psi$ , for different values of  $H^{min}$  and  $H^{max}$ .

1) *Experiment 1*: The Experiment 1 made under EDF has been repeated under FP with the same configuration parameters: results are reported in Figure 9. The graphs show that, under FP, SIRAP performs better than BROE- $\alpha\Delta$

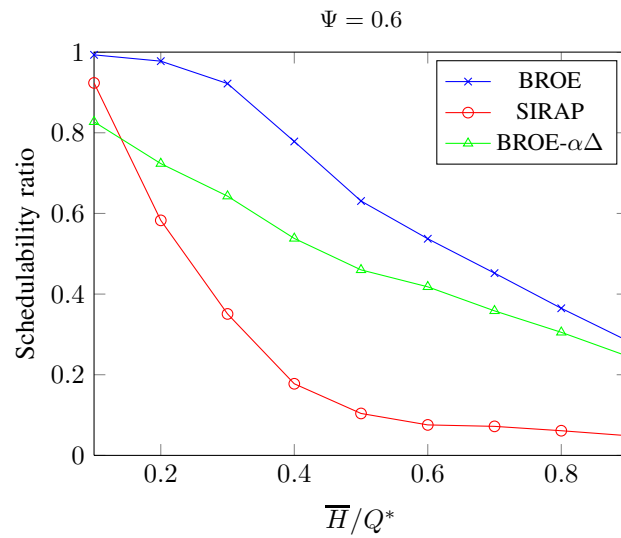


Figure 7: Schedulability under local EDF as a function of the average value  $\bar{H}/Q^*$ , for  $\Psi = 0.6$ .

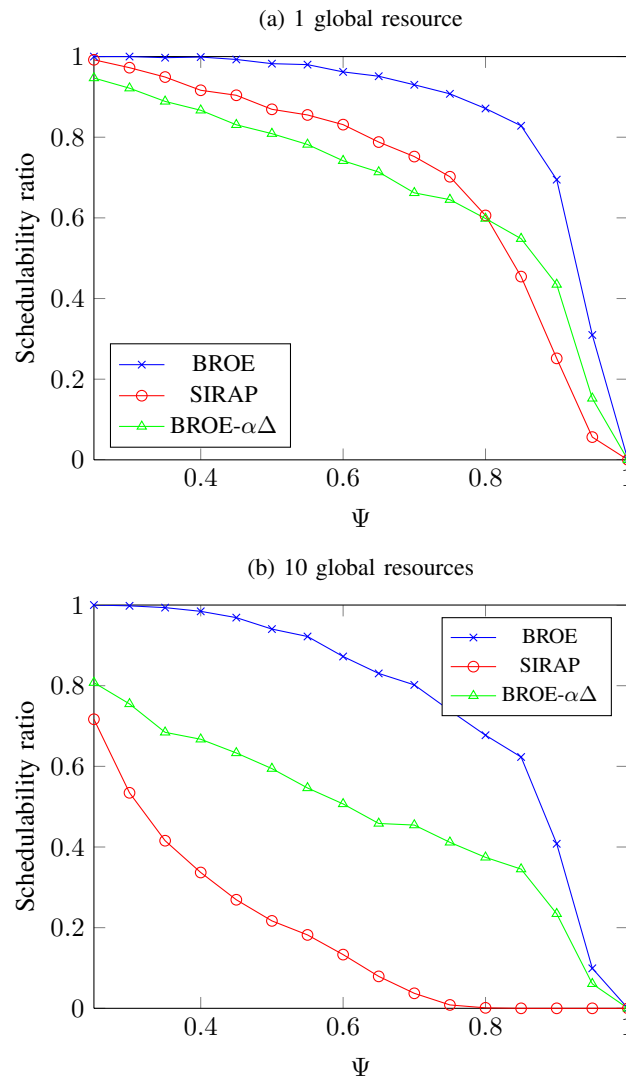


Figure 8: Schedulability under local EDF as a function of  $\Psi$ , for different numbers of global resources.

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.

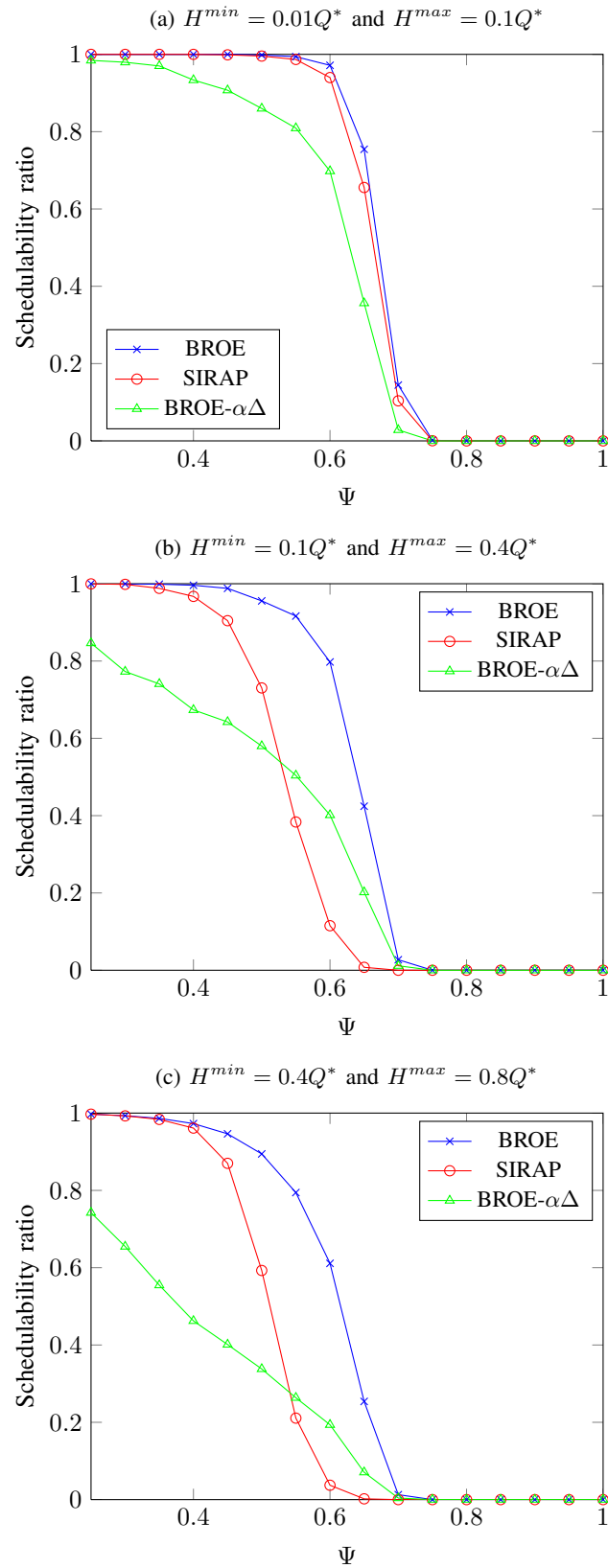


Figure 9: Schedulability under local FP as a function of  $\Psi$ , for different values of  $H^{min}$  and  $H^{max}$ .

2) *Experiment 2*: Experiment 2 has been repeated under FP with different configuration parameters. In fact, as  $\Psi$  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 using  $T_k^{min} = 2P_k$  and  $T_k^{max} = 18P_k$ , 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 10. In this case, SIRAP outperforms BROE- $\alpha\Delta$  for all values of  $\bar{H}/Q^*$ , but it is always dominated by the improved BROE.

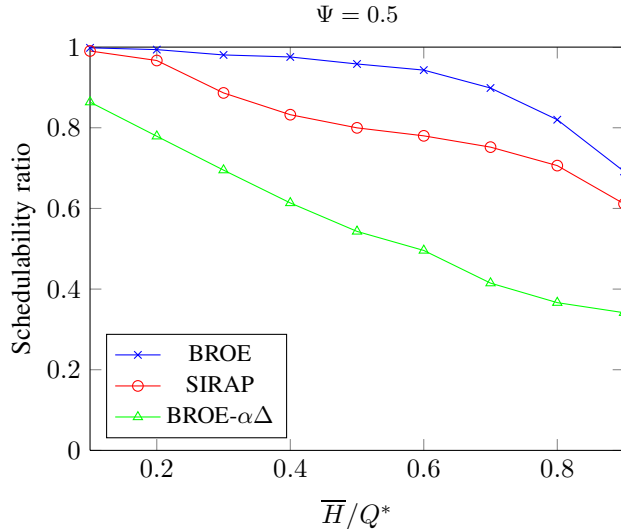


Figure 10: Schedulability under local FP as a function of the average value  $\bar{H}/Q^*$ , for  $\Psi = 0.5$ .

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 9b, while the other two cases are reported in Figure 11. 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.

## VIII. IMPLEMENTATION ISSUES

This section discusses some guidelines related to the implementation of the two resource sharing protocols SIRAP and BROE. These protocols have been implemented on the ERIKA Enterprise real-time kernel [15], using the Hard-CBS [17] 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  $\mu C/OS-II$  operating system by van den Heuvel et al. [27]. However, they used a periodic-idling server algorithm as a 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 [27] on the implementation complexity do not apply to bandwidth preserving server algorithms like the Hard-CBS.

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 [17], 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  $\tau_i$  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.

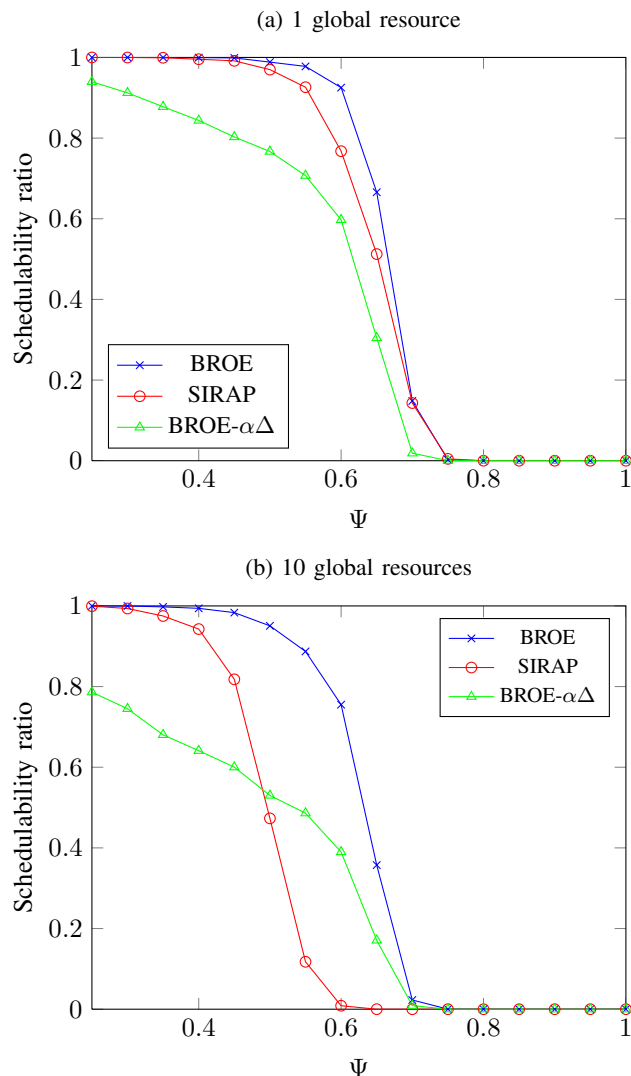


Figure 11: Schedulability under local FP as a function of  $\Psi$ , for different numbers of global resources.

## IX. 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  $\alpha\Delta$ . 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. 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 a suitable choice for implementing reservations in hierarchical systems under global resource sharing.

## REFERENCES

- [1] C. W. Mercer, S. Savage, and H. Tokuda, "Processor capacity reserves for multimedia operating systems," in *Proceedings of IEEE international conference on Multimedia Computing and System*, May 1994.
- [2] L. Abeni and G. Buttazzo, "Resource reservations in dynamic real-time systems," *Real-Time Systems*, vol. 27, no. 2, pp. 123–165, 2004.
- [3] 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.



- [4] 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.
- [5] 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.
- [6] T. P. Baker, "Stack-based scheduling for realtime processes," *Real-Time Systems*, vol. 3, no. 1, pp. 67–99, April 1991.
- [7] 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.
- [8] T. Ghazalie and T. Baker, "Aperiodic servers in a deadline scheduling environment," *Real-Time Systems*, vol. 9, 1995.
- [9] 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.
- [10] 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.
- [11] 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.
- [12] 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 13, 2007.
- [13] 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.
- [14] T.-W. Kuo and C.-H. Li, "A fixed priority driven open environment for real-time applications," in *Proceedings of the IEEE Real-Time Systems Symposium*, Phoenix, AZ, USA, December 1-3, 1999, pp. 256–267.
- [15] 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.
- [16] 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.
- [17] 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.
- [18] 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.
- [19] A. Easwaran, M. Anand, and I. Lee, "Compositional analysis framework using EDP resource models," in *Proceedings of the 28th IEEE Real-Time Systems Symposium (RTSS 2007)*, Tucson, Arizona, USA, December 3-6, 2007.
- [20] 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.
- [21] S. Baruah and A. Burns, "Sustainable scheduling analysis," in *Proceedings of the IEEE Real-time Systems Symposium (RTSS 2006)*, Rio de Janeiro, Brazil, December 5-8, 2006.
- [22] 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.
- [23] 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.
- [24] 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.
- [25] 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.
- [26] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Systems*, vol. 30, no. 1-2, pp. 129–154, 2005.
- [27] 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.