

# Towards exploiting the preservation strategy of sporadic servers

Reinder J. Bril and Pieter J.L. Cuijpers

*Technische Universiteit Eindhoven (TU/e), Department of Mathematics and Computer Science,  
Den Dolech 2, 5600 AZ Eindhoven, The Netherlands  
r.j.bril@tue.nl, p.j.l.cuijpers@tue.nl*

## Abstract

*Worst-case response time analysis of hard real-time tasks under hierarchical fixed priority pre-emptive scheduling has been addressed in a number of papers. By means of an example, we show that the existing analysis can be improved when a sporadic server is applied at highest priority and that server is exclusively used for hard real-time tasks. Improving the analysis is not straightforward, however, because the worst-case response time of a task is not necessarily assumed for the first job when released at a critical instant. Moreover, our example illustrates that the provision of the capacity of the server may be fragmented. The paper includes a brief investigation of best-case response times and response jitter for the example.*

## 1. Introduction

Today, fixed-priority pre-emptive scheduling (FPPS) is a de-facto standard in industry for scheduling systems with real-time constraints. A major shortcoming of FPPS, however, is that temporary or permanent faults occurring in one application can hamper the execution of other applications. To resolve this shortcoming, the notion of *resource reservation* [8] has been proposed. Resource reservation provides *isolation* between applications, effectively protecting an application against other, malfunctioning applications.

In a basic setting of a real-time system, we consider a set of independent applications, where each application consists of a set of periodically released, hard real-time tasks that are executed on a shared resource. We assume two-level hierarchical scheduling, where a *global* scheduler determines which application should be provided the resource and a *local* scheduler determines which of the chosen application's tasks should execute. Although each application could have a dedicated scheduler, we assume FPPS for every application. For temporal protection, each application is associated a dedicated reservation. We assume a *peri-*

*odic resource model* [11] for reservations. Conceivable implementations include FPPS for global scheduling using a specific type of server, such as the *periodic server* [5], the *deferrable server* [13], or the *sporadic server* [12].

Worst-case response time analysis of real-time tasks under hierarchical FPPS using deferrable servers and sporadic servers to implement reservations has been addressed in [1, 5, 6, 10], where the analysis presented in [5] improves on the earlier work. In [2, 4], we have shown that the analysis in [5] can be improved for a deferrable server when that server is exclusively used for hard real-time tasks. Essentially, the absence of soft real-time tasks allows for an exploitation of the preservation strategy of the deferrable server. In this paper, we will show that the analysis in [5] can also be improved for a sporadic server in the absence of soft real-time tasks. Improving the existing analysis is not straightforward, however, because the worst-case response time of a task is not necessarily assumed for the first job when released at a critical instant. Moreover, the provision of the capacity of the server may be fragmented, potentially giving rise to high context switch costs. For illustration purposes, we consider a specific class of subsystems  $\mathcal{S}$  and an example subsystem  $S \in \mathcal{S}$ . The paper includes a brief investigation of best-case response times and response jitter.

This paper is organized as follows. In Section 2, we briefly recapitulate existing analytical results for our class of subsystems  $\mathcal{S}$  and introduce our example subsystem  $S \in \mathcal{S}$ . This example clearly illustrated the potential for improvement. We investigate response times and response jitter for our example in Section 3. In Section 4, we discuss the differences found between our new results and the existing approaches. We conclude the paper in Section 5.

## 2. A recapitulation of existing analysis

In this section, we briefly recapitulate existing analysis. We start with a description of a scheduling model and then present our example subsystem  $S$ . Next, we recapitulate the analysis for a periodic resource model [11], a periodic

server [5], and a deferrable server [2], which we illustrate by means of  $S$ . We conclude this section with an overview.

## 2.1. A scheduling model

We assume FPPS for global scheduling, and consider a class of subsystems  $S$  consisting of an application with a single, periodic hard real-time task  $\tau$  and an associated server  $\sigma$  at highest priority. The server  $\sigma$  is characterized by a *replenishment period*  $T^\sigma$  and a *capacity*  $C^\sigma$ , where  $0 < C^\sigma \leq T^\sigma$ . Without loss of generality, we assume that  $\sigma$  is replenished for the first time at time  $\phi^\sigma = 0$ . The task  $\tau$  is characterized by a *period*  $T^\tau$ , a *computation time*  $C^\tau$ , and a *relative deadline*  $D^\tau$ , where  $0 < C^\tau \leq D^\tau \leq T^\tau$ . We assume that  $\tau$  is released for the first time at time  $\phi^\tau \geq \phi^\sigma$ , i.e. *at or after* the first replenishment of  $\sigma$ . The *worst-case response time*  $WR^\tau$  of the task  $\tau$  is the longest possible time from its arrival to its completion. The utilization  $U^\tau$  of  $\tau$  is given by  $\frac{C^\tau}{T^\tau}$  and the utilization  $U^\sigma$  of  $\sigma$  by  $\frac{C^\sigma}{T^\sigma}$ . A *necessary* schedulability condition for  $S$  is given by [4]

$$U^\tau \leq U^\sigma \leq 1. \quad (1)$$

## 2.2. An example subsystem

For illustration purposes, we use an example subsystem  $S \in \mathcal{S}$  with characteristics as described in Table 1. Note that  $\tau$  is an *unbound* task [5], because its period  $T^\tau$  is not an integral multiple of the period  $T^\sigma$  of the server. In this section, we are interested in minimum capacity  $C_{\min}^\sigma$  for the various types of servers, where  $C_{\min}^\sigma = \min\{C^\sigma \mid WR^\tau \leq D^\tau\}$ . Given (1),  $C_{\min}^\sigma \geq U^\sigma \cdot T^\sigma = 1.2$ .

	$T = D$	$C$
$\sigma$	3	$C^\sigma$
$\tau$	5	2

**Table 1. Characteristics of subsystem  $S$ .**

## 2.3. Analysis for periodic resource model

Based on [11], we merely postulate the following lemma. Without further elaboration, we mention that we can postulate similar lemmas for the analysis of  $S$  based on a deferrable server in [10] and the abstract server model in [6] (and therefore also on the sporadic and deferrable server).

**Lemma 1** *Assuming a periodic resource model for  $S$ , the worst-case response time  $WR^\tau$  of task  $\tau$  is given by*

$$WR^\tau = C^\tau + \left( \left\lceil \frac{C^\tau}{C^\sigma} \right\rceil + 1 \right) (T^\sigma - C^\sigma). \quad (2)$$

Given (2), we derive for our example  $S$  that the minimum capacity for a periodic resource model is given by  $C_{\min}^\sigma = 2$ . For this capacity, we find  $WR^\tau = 4$ .

## 2.4. Analysis for a periodic server

Strictly spoken, our class of subsystems  $S$  does not satisfy the model described in [5], because that article assumes that every set of tasks associated with a server contains at least one soft real-time task. Fortunately, a periodic server provides its resources irrespective of demand. As a result, the soft real-time tasks of a task set do not hamper the execution of the hard real-time tasks with which they share a periodic server. The analysis presented in [5] therefore equally well applies to  $S$  in general and  $S$  in particular. For an unbound task, we derive from [5] that  $WR^\tau$  is given by

$$WR^\tau = C^\tau + \left\lceil \frac{C^\tau}{C^\sigma} \right\rceil (T^\sigma - C^\sigma). \quad (3)$$

Without further elaboration, we mention that (3) also holds for the analysis of  $S$  based on a deferrable server in [1] and on a sporadic server in [1, 10]. Given (3), we derive that  $C_{\min}^\sigma = 1.5$ , giving rise to  $WR^\tau = 5$ .

## 2.5. Analysis for a deferrable server

The following theorem for  $S$  is proven in [4].

**Theorem 1** *Consider a highest-priority deferrable server  $\sigma$  with period  $T^\sigma$  and capacity  $C^\sigma$ . Furthermore, assume that the server is associated with a periodic task  $\tau$  with period  $T^\tau$ , worst-case computation time  $C^\tau$ , and deadline  $D^\tau = T^\tau$ , where the first release of  $\tau$  takes place at or after the first replenishment of  $\sigma$ . The deadline  $D^\tau$  is met when the respective utilizations satisfy the following inequality*

$$U^\tau \leq U^\sigma \leq 1. \quad (4)$$

Note that (4) is a necessary and sufficient (i.e. exact) schedulability condition for both the task and the server.

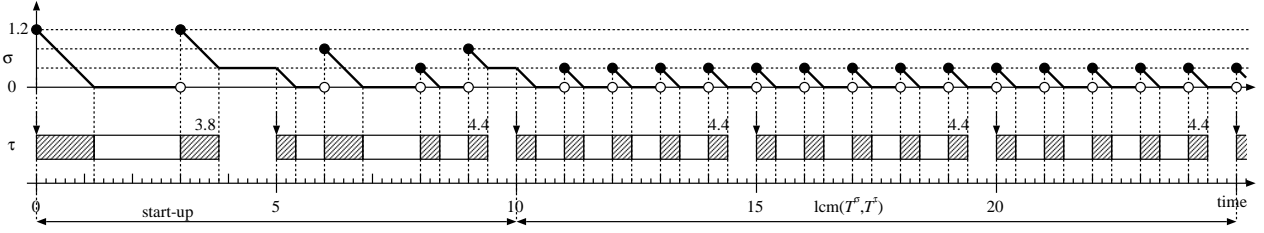
According to Theorem 1,  $S$  is schedulable using a deferrable server for a capacity  $C_{\min}^\sigma = U^\tau \cdot T^\sigma = 1.2$ . As illustrated in [2], the worst-case response time  $WR^\tau$  of task  $\tau$  for  $C^\sigma = 1.2$  is equal to 4.4.

## 2.6. Overview

Table 2 gives an overview of the minimum capacities  $C_{\min}^\sigma$  and minimum server utilities  $U_{\min}^\sigma$  that guarantee schedulability of  $S$  for existing analytical approaches and different types of servers. The table includes the worst-case response time  $WR^\tau$  of  $\tau$  for these approaches. The analysis for a sporadic server is the topic of the next section.

## 3. Analysis for a sporadic server

We will now explore the example in more detail by considering the worst-case response time, best-case response



**Figure 1. Timeline for  $S$  with a simultaneous release of task  $\tau$  and sporadic server  $\sigma$ , including a graph with the remaining capacity of  $\sigma$ . The numbers at the top right corner of the boxes denote the response times of the respective releases.**

approach (including server or model)	$C_{\min}^{\sigma}$	$U_{\min}^{\sigma}$	$WR^{\tau}$
periodic resource model [11]	2.0	5/6	4.0
abstract server model [6]			
deferrable server [6, 10]			
sporadic server [6]	1.5	1/2	5.0
periodic server [5]			
deferrable server [1]			
sporadic server [1, 10]	1.2	2/5	4.4
deferrable server [2, 4]			
sporadic server (this paper)			

**Table 2. A comparison of approaches for  $S$ .**

time, and response jitter of task  $\tau$  of  $S$  for a sporadic server with a capacity  $C^{\sigma} = 1.2$ .

Figure 1 shows a timeline with the available processing time of sporadic server  $\sigma$  and the executions of task  $\tau$  with a first release of  $\tau$  at  $\varphi^{\tau} = 0$ . From this figure, we conclude that  $S$  is schedulable under a sporadic server for  $\varphi^{\tau} = 0$ . Moreover, we derive that the worst-case response time  $WR^{\tau}(\varphi^{\tau})$  and best-case response time  $BR^{\tau}(\varphi^{\tau})$  of  $\tau$  for  $\varphi^{\tau} = 0$  are given by  $WR^{\tau}(0) = 4.4$  and  $BR^{\tau}(0) = 3.8$ , respectively. Because the processing time of a sporadic server is never lost, both the worst-case response time  $WR^{\tau}(\varphi^{\tau})$  and best-case response time  $BR^{\tau}(\varphi^{\tau})$  of  $\tau$  are independent of the first release  $\varphi^{\tau}$  of the task, hence  $WR^{\tau}(\varphi^{\tau}) = 4.4$  and  $BR^{\tau}(\varphi^{\tau}) = 3.8$ . This has the following consequences. Firstly, both a *critical instant* [7] and an *optimal* (or *favorable*) *instant* [3, 9] occurs for every value of  $\varphi^{\tau}$ . Next, the response jitter  $RJ^{\tau}$  of task  $\tau$  is constant, i.e.

$$\begin{aligned}
 RJ^{\tau} &= \sup_{\varphi^{\tau}} (WR^{\tau}(\varphi^{\tau}) - BR^{\tau}(\varphi^{\tau})) \\
 &= WR^{\tau} - BR^{\tau} = 4.4 - 3.8 = 0.6.
 \end{aligned}$$

When we ignore the initial start-up phase, the response time of the task  $\tau$  is constant and equal to 4.4. In such a case, the response jitter becomes equal to zero.

From  $WR^{\tau} = 4.4$ , we conclude that  $S$  is schedulable under a sporadic server with a capacity of  $C^{\sigma} = 1.2$ . Moreover, we conclude that capacity suspension of  $\sigma$  is a prerequisite

for schedulability of  $S$  with a capacity of  $C^{\sigma} = 1.2$ , and  $S$  is therefore not schedulable with a periodic server with that capacity. Hence, the worst-case response time analysis presented in [5] can be improved when a sporadic server is exclusively used for hard real-time tasks.

We conclude this section with three observations. Firstly, the worst-case response time is not assumed for the first job of the task  $\tau$ . Secondly, the execution of any job of  $\tau$  after the 1<sup>st</sup> job in Figure 1 is dependent on the execution of a previous job of  $\tau$ . Thirdly, the provision of the capacity of the server becomes fragmented with a size 0.4, which is equal to the greatest common divisor of the computation time  $C^{\tau}$  of task  $\tau$  and the capacity  $C^{\sigma}$  of the server  $\sigma$ , i.e.  $\gcd(C^{\tau}, C^{\sigma}) = \gcd(1.2, 2.0) = 0.4$ .

## 4. Discussion

In Table 2, we partitioned the various approaches for  $S$  into three main categories based on the minimum capacity  $C_{\min}^{\sigma}$  of the periodic resource or server and the worst-case response time  $WR^{\tau}$  of the task  $\tau$ . Notably, both the deferrable server and the sporadic server are dealt with in all three categories. The differences between these results originate from the differences in the assumptions made for the three categories. We briefly consider the assumptions for the three categories and conclude the section with a remark.

### 4.1. Assumptions of approaches

For the first category, no assumptions are made about the characteristics of other servers nor about the priority of other servers. As a result, the specific preservation strategy of a server cannot be exploited.

For [1, 5] of the second category, the fact that  $S$  has highest priority is taken into account, and without potential interference of higher priority servers the minimum capacity of  $\sigma$  can therefore be reduced significantly, i.e. with 25%. These existing approaches do not exploit the preservation strategy of a server to improve the results of a specific server, however. We observe that because every task

set is assumed to have at least one (unspecified) soft real-time task in [5], the preservation strategy of a server cannot be exploited. We consider the sporadic server approach of [10] in the next section.

Similar to the second category, the third category also takes the fact that  $S$  has highest priority into account. Moreover, the approaches in this category exploit the specific preservation strategy of both the deferrable server and the sporadic server. As a result, the minimum capacity of  $\sigma$  can again be reduced, in this case with an additional 20%.

## 4.2. Concluding remark

Considering the approaches of [10] in Table 2, the deferrable server falls into the first category and the sporadic server falls into the second category. This is surprising, because [10] does not make any assumptions about the characteristics of other servers nor about the priority of other servers. Hence, one would expect that the results for the approach for the sporadic server in [10] would also fall into the first category.

In [5], it is claimed that periodic servers dominate both deferrable servers and sporadic servers when a task set contains at least one soft real-time task. In [2, 4] and this paper, we have shown by means of an example that in the absence of soft real-time tasks that claim no longer holds.

## 5. Conclusion

In this paper, we considered response times and response jitter of hard real-time tasks under H-FPPS using sporadic servers. We showed by means of an example that existing worst-case response time analysis can be improved when a sporadic server is used at highest priority and that server is exclusively used for hard real-time tasks. For our example, the utilization of the server can be significantly reduced when a sporadic server is used rather than a periodic server or a general periodic resource model is assumed. Given these initial results, application of a sporadic server at highest priority can be an attractive alternative for resource-constrained systems with stringent timing requirements for a specific application when no appropriate period can be selected for its associated server. Unfortunately, improving the existing analysis is not straightforward, because the worst-case response time of a task is not necessarily assumed for the first job when released at a critical instant due to a start-up phenomenon. Moreover, the provision of the capacity of the server may be fragmented, as illustrated by the example, potentially giving rise to high context switch costs.

Using the same example, we briefly investigated best-case response times and response jitter. Unlike existing best-case response times of tasks under FPPS [3, 9], we did

not assume infinite repetitions towards both ends of the time axis. As a result, the best-case response time of a task is determined by a start-up phase. When the start-up phase can be ignored, the best-case response time becomes equal to the worst-case response time, and the resulting response jitter therefore becomes equal to zero.

Improved response time analysis of H-FPPS using sporadic servers is a topic of future work.

## References

- [1] L. Almeida and P. Peidreiras. Scheduling with temporal partitions: response-time analysis and server design. In *Proc. 4<sup>th</sup> ACM International Conference on Embedded Software (EMSOFT)*, pp. 95 – 103, September 2004.
- [2] R. Bril and P. Cuijpers. Towards exploiting the preservation strategy of deferrable servers. In *Proc. WiP session of the 14<sup>th</sup> IEEE RTAS*, pp. 13–16, April 2008.
- [3] R. Bril, E. Steffens, and W. Verhaegh. Best-case response times and jitter analysis of real-time tasks. *Journal of Scheduling*, 7(2):133–147, March 2004.
- [4] P. Cuijpers and R. Bril. Towards budgetting in real-time calculus: deferrable servers. In *Proc. 5<sup>th</sup> International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS), LNCS-4763*, pp. 98 – 113, October 2007.
- [5] R. Davis and A. Burns. Hierarchical fixed priority preemptive scheduling. In *Proc. 26<sup>th</sup> IEEE RTSS*, pp. 389–398, December 2005.
- [6] G. Lipari and E. Bini. Resource partitioning among real-time applications. In *Proc. 15<sup>th</sup> ECRS*, pp. 151–158, July 2003.
- [7] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a real-time environment. *Journal of the ACM*, 20(1):46–61, January 1973.
- [8] R. Rajkumar, K. Juvva, A. Molano, and S. Oikawa. Resource kernels: A resource-centric approach to real-time and multimedia systems. In *Proc. SPIE, Vol. 3310, Conference on Multimedia Computing and Networking (CMCN)*, pp. 150–164, January 1998.
- [9] O. Redell and M. Sanfridson. Exact best-case response time analysis of fixed priority scheduled tasks. In *Proc. 14<sup>th</sup> ECRS*, pp. 165–172, June 2002.
- [10] S. Saewong, R. Rajkumar, J. Lehoczky, and M. Klein. Analysis of hierarchical fixed-priority scheduling. In *Proc. 14<sup>th</sup> ECRS*, pp. 152–160, June 2002.
- [11] I. Shin and I. Lee. Periodic resource model for compositional real-time guarantees. In *Proc. 24<sup>th</sup> IEEE RTSS*, pp. 2–13, December 2003.
- [12] B. Sprunt, L. Sha, and J. Lehoczky. Aperiodic task scheduling for hard real-time systems. *Real-Time Systems*, 1(1):27–60, June 1989.
- [13] J. Strosnider, J. Lehoczky, and L. Sha. The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments. *IEEE Transactions on Computers*, 44(1):73–91, January 1995.