Analysis of hierarchical fixed-priority pre-emptive scheduling revisited

Analysis of hierarchical fixed-priority pre-emptive scheduling revisited

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

This paper revisits worst-case response time analysis of real-time tasks under hierarchical fixed-priority pre-emptive scheduling. Using an example consisting of a single server and a single hard real-time task, we show that existing worst-case response time analysis can be improved for deferrable servers and sporadic servers when a server is exclusively used for hard real-time tasks. Moreover, we show that improving the existing analysis in this setting 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. Finally, we briefly investigate best-case response times of tasks.

1. Introduction

One of the main scheduling approaches in real-time computing systems is given by fixed-priority preemptive scheduling (FPPS) [10, 12]. The approach is founded on a fixed-priority scheduling theory, supported by a suite of open software standards, commercially available schedulability analysis tools and real-time operating systems, and adopted by leading companies and institutions world-wide.

Scheduling a set of real-time tasks sharing a resource gives rise to the so-called *temporal interference* problem, i.e. a malfunctioning task may cause other tasks to fail to meet their time constraints. A standard solution for this problem is to introduce resource budgets for tasks, which provide temporal protection between tasks by guaranteeing a minimal amount of resources [16]. Those budgets can be implemented using so-called *servers* that dispatch the available resources to the tasks that are appointed to them [9, 21]. Appointing servers to tasks gives rise to *hierarchical* scheduling. For two-level hierarchical scheduling, a *global* scheduler is used to determine which set of tasks is allowed to use the resource at any given moment in time and a *local* scheduler is used to determine which of those tasks should actually execute. In this paper, we assume two-level hierarchical scheduling using FPPS at both levels.

In general, a server for a shared processing resource, such as a CPU, is characterized by a *capacity* and a *replenishment period* [6]. The capacity is the maximum amount of resources (i.e. the maximum amount of processing time) that a server can provide to its associated tasks during its replenishment period. The replenishment period is the minimum time between replenishments of the capacity. Servers typically differ with respect to the amount and moment in time of the replenishments and to the preservation of the remaining capacity when their associated tasks are not ready to use it. In this paper, we consider *deferrable servers* [24], *sporadic servers* [23], and *periodic servers* [9].

Analysis of real-time tasks under hierarchical FPPS (H-FPPS) has been addressed in [1, 3, 9, 21, 22], where the analysis presented in [9] improves on the earlier work. In this paper, we show that the existing worst-case response time analysis can be improved for deferrable servers and sporadic servers when a server is exclusively used for hard real-time tasks. To this end, we consider an example consisting of a single server and a single hard real-time task. We show that 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. We merely illustrate best-case response times.

This paper is organized as follows. In Section 2 we briefly describe a real-time scheduling model for the classes of systems considered in this paper. Next, in Section 3, we recapitulate existing worst-case response time analysis. We illustrate the existing analysis for an example using a periodic server in Section 4. We revisit the analysis for deferrable servers and sporadic servers in Sections 5 and 6, respectively. We discuss our findings in Section 7 and conclude the paper in Section 8.

2. A real-time scheduling model

This section describes a basic real-time scheduling model for H-FPPS. Most of the definitions of this model are taken from [4, 9], and originate from [15]. We start this section with a description of a general scheduling model in terms of tasks and servers. We subsequently characterise the classes of systems which we consider in this paper. Next, we introduce specific terminology, concepts, and assumptions for tasks and servers for the systems under consideration.

2.1. A general scheduling model

We assume a single processor, a set of n servers σ_1, σ_2, ..., σ_n, and a collection of n disjunct sets $\mathcal{T}_1$, $\mathcal{T}_2$, ..., $\mathcal{T}_n$ of tasks, where the server σ_i is appointed to the set of tasks $\mathcal{T}_i$. Each set $\mathcal{T}_i$ consists of m_i periodically released, independent hard real-time tasks and m_i' soft real-time tasks. At any moment in time, the processor is used to execute the highest priority task that has work pending of the highest priority server that still has processing time remaining.

In this paper, we assume that the tasks of $\mathcal{T}_i$ only execute under the server σ_i. Moreover, we assume that the m_i hard real-time tasks in a task set $\mathcal{T}_i$ execute at higher priorities than the m_i' soft real-time tasks. We also assume that servers and tasks have fixed priorities and are fully pre-emptive. For notational convenience we assume that the servers are given in order of decreasing priority, i.e. server σ_1 has highest priority and server σ_n has lowest priority.

2.2. Systems under consideration

In this paper, we consider three main classes of systems:

- Γ, consisting of a single server σ and a single hard real-time task τ, i.e. $n = 1$, $m_1 = 1$, and $m_1' = 0$;
- Γ', with a single hard real-time task τ running on a highest priority server σ, i.e. $n \geq 1$, $m_1 = 1$, and $m_1' = 0$;
- $\widetilde{\Gamma}$, consisting of a single server σ, a single hard real-time task τ, and one or more soft real-time tasks, i.e. $n = 1$, $m_1 = 1$, and $m_1' \geq 1$.

We observe that Γ is a special case of the class Γ'. Note that $\widetilde{\Gamma}$ only differs from Γ by its constituent non-empty set of soft real-time tasks. Finally, we mention that $\widetilde{\Gamma}$ can be transformed into Γ' by appointing all soft real-time tasks to a dedicated server σ' which receives a lower priority than σ.

2.3. A task model

The hard real-time task τ is characterized by a *period* (or *inter-arrival time*) $T^\tau \in \mathbb{R}^+$, a *worst-case computation time* $C^\tau \in \mathbb{R}^+$, a (*relative*) *deadline* $D^\tau \in \mathbb{R}^+$, where $C^\tau \leq \min(D^\tau, T^\tau)$, and a *phasing* (or *first activation time*) $\varphi^\tau \in \mathbb{R}$. For the time being, we do not constrain the deadline, i.e. we assume an arbitrary deadline. An *activation* (or *release*) *time* is a time at which task τ becomes ready for execution. A release of a task is also termed a *job*. The job of task τ with release time φ^τ is referred to as job one. The release of job k of τ therefore takes place at time $a_k^\tau = \varphi^\tau + (k-1)T^\tau$, $k \in \mathbb{N}^+$. The (absolute) deadline of job k of τ taskes place at $d_k^\tau = a_k^\tau + D^\tau$.

The *active* (or *response time*) *interval* of job k of τ is defined as the time span between the activation time of that job and the completion time c_k^τ, i.e. $[a_k^\tau, c_k^\tau)$. The response time r_k^τ of job k of τ is defined as the length of its active interval, i.e. $r_k^\tau = c_k^\tau - a_k^\tau$.

The *worst-case response time* WR^τ of task τ is the largest reponse time of any of its jobs. A *critical instant* of task τ is defined to be an (hypothetical) instant that leads to its worst-case response time. Typically, such an instant is described as a point in time with particular properties. As an example, a critical instant for tasks under FPPS is given by a point in time for which all tasks have a simultaneous release. Because the deadline of task τ is hard, τ can be scheduled iff, i.e. if and only if

$$WR^\tau \leq D^\tau. \tag{1}$$

The *best-case response time* BR^τ of task τ is the shortest response time of any of its jobs. An *optimal instant* of task τ is defined to be an (hypothetical) instant that leads to its best-case response time.

The fraction of processor time spent on executing task τ is $\frac{C^\tau}{T^\tau}$, and is termed the *utilization* (*factor*) U^τ of task τ, i.e.

$$U^\tau = \frac{C^\tau}{T^\tau}. \tag{2}$$

We assume that we do not have control over the phasing φ^τ, for instance since the task is released by external events, so we assume that any arbitrary phasing may occur. This assumption is common in real-time scheduling literature [11, 12, 15]. We observe that, although we assume arbitrary phasing, we do assume that τ is released *at* or *after* the first replenishment of the server σ. We also assume other standard basic assumptions [15], i.e. task τ is ready to run at the start of each period and does not suspend itself, a job of task τ does not start before its previous job is completed, soft real-time tasks will be preempted instantaneously when task τ becomes ready to run, and the overhead of context switching and task scheduling is ignored.

2.4. Server models

The server σ is characterized by a *replenishment period* $T^\sigma \in \mathbb{R}^+$, a *capacity* (or *maximum processing time*) $C^\sigma \in \mathbb{R}^+$, a (*relative*) *deadline* $D^\sigma \in \mathbb{R}^+$, and a *phasing* (or *first replenishment time*) $\varphi^\sigma \in \mathbb{R}$.

We assume that the server's period and deadline are equal, i.e. $T^\sigma = D^\sigma$. Without loss of generality, we assume that the server σ is replenished for the first time at time $\varphi^\sigma = 0$.

The *worst-case response time* WR^σ of server σ is the longest possible time from the server being replenished to its capacity being exhausted, assuming the tasks of $\mathcal{T}_1$ are ready to use all of its capacity. The server is schedulable iff

$$WR^\sigma \leq T^\sigma. \tag{3}$$

For the systems under consideration, the (highest priority) server σ is schedulable when $C^\sigma \leq T^\sigma$.

The fraction of processor time spent by server σ is $\frac{C^\sigma}{T^\sigma}$, and is termed the *utilization* (*factor*) U^σ of server σ, i.e.

$$U^\sigma = \frac{C^\sigma}{T^\sigma}. \tag{4}$$

We also assume most other basic assumptions for tasks, i.e. the server σ is ready to provide processing time to the tasks of $\mathcal{T}_1$ when it still has processing time available and a task of $\mathcal{T}_1$ has work pending, a lower priority server (and the task consuming its processing time) will be preempted instantaneously when σ becomes ready, and the overhead of context switching and server scheduling is ignored.

As mentioned before, servers differ with respect to the amount and moment in time of the replenishments and to the preservation of the remaining capacity when their associated tasks are not ready to use it. A *periodic server* is replenished periodically. Moreover, it is always ready to provide processing time when it still has processing time available. Just like the hard real-time task, a periodic server never suspends itself, and it therefore does not preserve processing time. When none of its associated tasks has work pending, it behaves as if a background task is ready to consume its processing time.

A *deferrable server* is also replenished periodically. Unlike a periodic server, a deferrable server suspends itself when it still has processing time available and none of its associated tasks has work pending. The remaining processing time is therefore preserved and becomes ready for consumption upon a release of any of its associated tasks before the end of the period. The processing time that remains at the end of the period is lost.

Unlike the periodic server and the deferrable server, a *sporadic server* is not replenished periodically. A sporadic server preserves its processing time when its associated tasks are not ready to use it, but its processing time is never lost. Consumed processing time is replenished an interval of time corresponding with the replenishment period after the server became ready [23].

3. Recapitulation of worst-case analysis

In this section, we briefly recapitulate existing worst-case response time analysis of hard real-time tasks under H-FPPS as presented in [9]. The analysis is claimed to be exact by the authors of that paper for periodic servers, deferrable servers and sporadic servers. Given our scheduling model, the systems considered in that paper can be characterized by $n \geq 1$, $m_i \geq 1$, and $m'_i \geq 1$. As a result, $\widetilde{\Gamma}$ is covered by that paper, but Γ and Γ' are not.

Deadlines of tasks are not explicitly constrained in [9]. Given a comment on page 18 of [8], a document on which [9] is based, the analysis assumes $D^\tau \leq T^\tau$, however. A task τ in [9] can be either *bound* or *unbound*. A task τ is bound if it has a period that is an exact multiple of the server's period, i.e. $T^\tau = nT^\sigma$ for $n \in \mathbb{N}^+$, and an arrival time that coincides with the replenishment of the server's capacity. Otherwise τ is unbound. Because we assume an arbitrary phasing for the hard real-time task τ, we consider τ to be unbound.

3.1. Analysis for the class of systems $\widetilde{\Gamma}$

We will now recapitulate the analysis of [9] for a class of systems $\widetilde{\Gamma}$ consisting of a single server σ, a single hard real-time task τ with $D^\tau \leq T^\tau$ and one or more soft real-time tasks. The critical instant for τ under the server σ occurs when:

1. The server's capacity has been exhausted by the soft real-time tasks as early in the period as possible.
2. The task τ arrives just after the server's capacity has been exhausted.

Moreover, the worst-case response time WR^τ of the task τ is given by

$$WR^\tau = C^\tau + \left\lceil \frac{C^\tau}{C^\sigma} \right\rceil (T^\sigma - C^\sigma). \tag{5}$$

3.2. An example

For illustration purpose, let's assume that the deadline D^τ of task τ is equal to its period T^τ. Using Equation (1), this leads to the following condition for schedulability for τ

$$C^\tau + \left\lceil \frac{C^\tau}{C^\sigma} \right\rceil (T^\sigma - C^\sigma) \leq T^\tau. \tag{6}$$

Now, as an example, we fix C^τ and T^τ and plot the minimum utilization U^σ_{min} of the server as a function of T^σ, i.e. we plot

$$U^\sigma_{min}(T^\sigma) = \min \left\{ \frac{C^\sigma}{T^\sigma} \mid T^\tau \geq C^\tau + \left\lceil \frac{C^\tau}{C^\sigma} \right\rceil (T^\sigma - C^\sigma), C^\sigma \geq 0 \right\}.$$

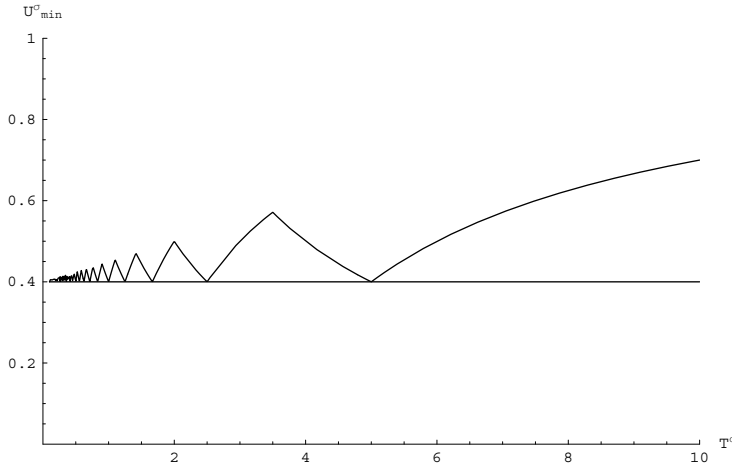

Figure 1. Minimum server utilization U^{σ}_{min} for $C^{\tau} = 2$ and $T^{\tau} = 5$ as a function of T^{σ}.

The result for $C^{\tau} = 2$ and $T^{\tau} = 5$, obtained using Mathematica, is depicted in figure 1. The horizontal line in this figure, shows the utilization $U^{\tau} = \frac{C^{\tau}}{T^{\tau}}$ of the task.

The figure illustrates that according to [9] only for values of T^{σ} equal to an integral fraction of T^{τ}, i.e. $T^{\sigma} = \frac{T^{\tau}}{n}$ for $n \in \mathbb{N}^{+}$, the minimum server utilization U^{σ}_{min} needed for schedulability of task τ is equal to the task utilization U^{τ}. For other values of T^{σ}, U^{σ}_{min} is higher than U^{τ}.

4. Analysis for periodic servers

Based on Equation (6), we derive that for $\widetilde{\Gamma}$ with $C^{\tau} = 2$, $T^{\tau} = D^{\tau} = 5$, and $T^{\sigma} = 3$ the minimum capacity C^{σ}_{min} required for a periodic server σ^{P} is 1.5. Using Equation (5), we derive $WR^{\tau} = 5$. A critical instant for τ under σ^{P} is shown in Figure 2, i.e. for $\varphi^{\tau} = C^{\sigma} = 1.5$.

From the description of periodic servers in Section 2.4, it is clear that the capacity of σ^{P} is provided irrespective of either hard or soft real-time tasks being ready for execution. This is also illustrated in Figure 2. This behavior of periodic servers has the following implications for the execution of the hard real-time task τ. Firstly, the execution of the hard real-time task τ is independent of (the absence or presence of) soft real-time tasks. As a consequence, the timeline for $\widetilde{\Gamma}$ shown in Figure 2 is also valid for Γ. Secondly, the execution of a job of τ is independent of the execution of a previous job of τ. The response time of a job of τ can therefore be determined in isolation. Thirdly, the behavior of the server σ^{P} and the task τ becomes *stable* after an initial start-up phase of 0.5. In particular, the schedule in $[0.5, 15.5)$ is repeated in the intervals $[0.5 + hH, 0.5 + (h+1)H)$, $h \in \mathbb{N}$ and $H = 15$, where H is the *hyperperiod* of the server and the task, which is equal to the least common multiple (lcm) of their periods. Stated in other words, the schedule is periodic with period $H = 15$ from time 0.5.

We will now explore the example in more detail by determining the best-case response times and worst-case response times of task τ as functions of φ^{τ}. To this end, we first introduce the auxiliary notion of *relative phasing* φ_{R} of task τ with respect to the server σ, which is given by

$$\varphi_{\mathrm{R}} = (\varphi^{\tau} - \varphi^{\sigma}) \mod T^{\sigma}. \tag{7}$$

Note that $0 \leq \varphi_{\mathrm{R}} < T^{\sigma}$. In general, we can therefore restrict φ^{τ} to values in the interval $[0, T^{\sigma})$ for Γ. Because the execution of a job of τ is independent of the execution of a previous job of τ, the length of the start-up phase is at most equal to φ^{τ} and the start-up phase does not contain any executions of τ. As a result, we can restrict φ^{τ} even further to values in the interval $[0, \gcd(T^{\sigma}, T^{\tau}))$, where gcd is the greatest common divisor of T^{τ} and T^{σ}, i.e. $\gcd(T^{\sigma}, T^{\tau}) = \gcd(3,5) = 1$. Timelines with a first release of task τ at $\varphi^{\tau} \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ are shown in Figure 14 in Appendix B.1. The resulting functions for the best-case and worst-case response times of τ are illustrated in Figure 3. Notably, the function for the best-case response time is constant, i.e. $BR^{\tau}(\varphi^{\tau}) = 3.5$.

5. Analysis for deferrable servers revisited

A deferrable server has the ability to defer the provision of processing time in case no associated task is ready to use it. Unfortunately, this relative advantage of a deferrable server compared to perioric server can not be exploited for $\widetilde{\Gamma}$ because of the presence of soft real-time tasks. Based on a novel schedulability theorem, we show that existing worst-case response time analysis can be improved when a deferrable server is exclusively used for hard real-time tasks.

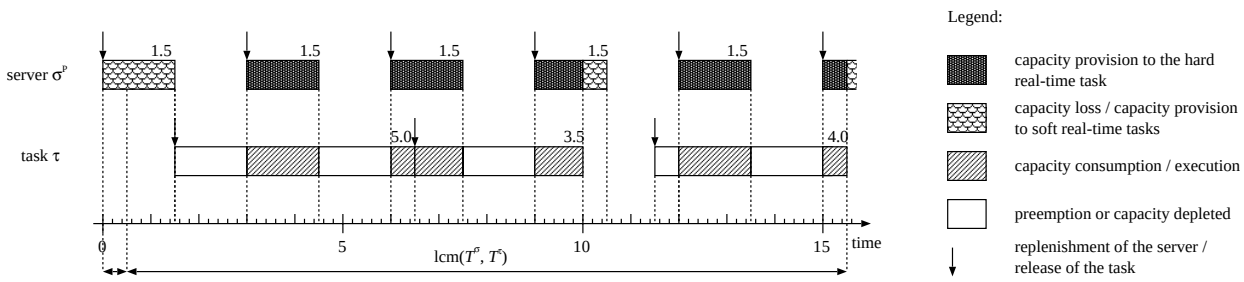

Figure 2. Timeline with a release of task τ at time $t = 1.5$ using a periodic server σ^P with capacity $C^\sigma = 1.5$. The numbers at the top right corner of the boxes denote the reponse times of the respective releases. The behavior becomes stable after an initial start-up phase of 0.5.

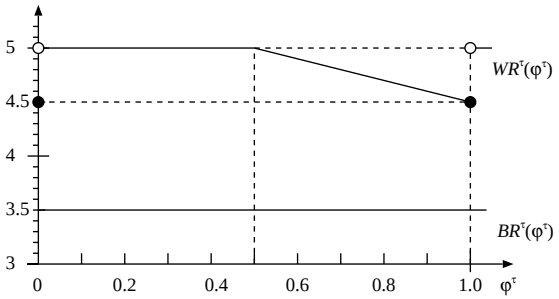

Figure 3. Worst-case and best-case response times of task τ as a function of the first release time φ^τ using a periodic server σ^P with capacity $C^\sigma = 1.5$.

This will be illustrated for an example system belonging to the class Γ. We will explore the example in detail by determining worst-case response times and best-case response times of task τ as a function of φ^τ.

5.1. A schedulability theorem

Given our scheduling model in Section 2, the following theorem holds for Γ; see [7], in which the theorem is derived from a formal model of a deferrable server with a single task, using the real-time calculus of [25].

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

$$U^\tau \leq U^\sigma \leq 1. \tag{8}$$

Note that our theorem holds for both a bound task and an unbound task. Furthermore, note that (8) is a necessary and sufficient (i.e. exact) schedulability condition for both the task and the server. Finally, note that

$$U^{\mathcal{T}} \leq U^\sigma \leq 1, \tag{9}$$

is a *necessary* schedulability condition for any set $\mathcal{T}$ of hard real-time tasks with utilization $U^{\mathcal{T}}$ and an associated server σ with utilization U^σ.

5.2. Schedulability of task τ

According to our theorem, the server utilization U^σ may be chosen equal to the task utilization U^τ for the class of systems Γ, irrespective of T^σ. From this example, we conclude that the surplus of U^σ_{min} compared to U^τ in Figure 1 is needed for $\widetilde{\Gamma}$ using a deferrable server to compensate for the affect of the soft real-time tasks on the worst-case response time of the hard real-time task τ. Notably, there is no need for compensation when T^σ is an integral fraction of T^τ. Moreover, we conclude that the schedulability condition expressed by (6) is sufficient but not necessary for Γ assuming task τ to be unbound. As a result, the worst-case response time analysis presented in [9] can be improved when a deferrable server is exclusively used for hard real-time tasks.

5.3. Improving the worst-case analysis

We now consider improving the worst-case response time analysis for deferrable servers. To this end, we consider an example system $S \in \Gamma$ with characteristics as described in Table 1. According to Theorem 1, S is schedulable. Conversely, S is considered unschedulable according to the existing analysis, i.e. a worst-case response time equal to 5.6 is derived for τ using equation (5), a value which is larger than τ's deadline. Hence, S is not schedulable with a periodic server.

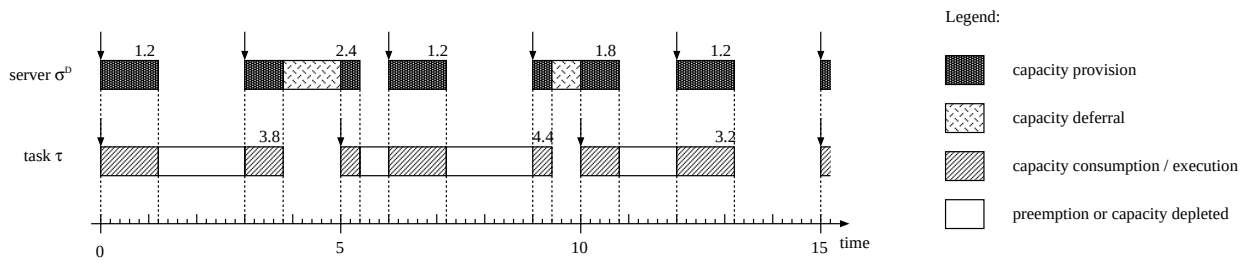

Figure 4. Timeline for S with a simultaneous release of task τ and deferrable server σ^D.

	$T = D$	C
σ	3	1.2
τ	5	2

Table 1. Characteristics of example system S.

Figure 4 shows a timeline with the executions of the server and the task for $\varphi^\tau = 0$ in an interval of length 15, i.e. equal to the hyperperiod H of the server and the task. The schedule in $[0,15)$ is repeated in the intervals $[hH,(h+1)H)$, with $h \in \mathbb{Z}$, i.e. the schedule is periodic with period H. From this figure and the fact that S is not schedulable with a periodic server, we conclude that capacity deferral of σ is a prerequisite for schedulability of S. Unlike the systems considered in [9], periodic servers therefore do not dominate deferrable servers with respect to schedulability of tasks when the set of tasks appointed to a server consists of hard real-time tasks only.

Considering Figure 4, we observe that the execution of the 2^{nd} job of τ, which is released at time $t = 5$, is influenced by the first job. In particular, an amount of just 0.4 processing time is provided to the 2^{nd} job rather than a maximum amount of 1.0 processing time that still could have been available at the release of the 2^{nd} job. Unlike the example given in Section 4 for the periodic server, the executions of jobs of task τ are therefore not necessarily independent when a deferrable server is used. Moreover, we observe that the worst-case response time of the task is assumed for the 2^{nd} rather than the 1^{st} job. Hence, we need to revisit the notion of *busy period* [13] in the context of hierarchical FPPS to take account of this fact.

We will now explore the example in more detail by considering the worst-case response times for the task τ for specific values of the first release time φ^τ. Because the executions of jobs of task τ are not necessarily independent, we cannot straight away restrict φ^τ to values in the interval $[0, \gcd(T^\sigma, T^\tau))$. Fortunately, the response times of jobs of τ in the start-up phase are at most equal to the largest response time of jobs in the stable phase. As a result, it is sufficient to consider values for φ^τ in the interval $[0, \gcd(T^\sigma, T^\tau))$ when considering the worst-case response times of τ. Timelines for τ and σ^D are shown for $\varphi^\tau \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ in Figure 11 in Appendix A.

The worst-case response time WR^τ of task τ is shown as function of φ^τ in Figure 5. WR^τ is equal to 4.4 and assumed for $\varphi^\tau = 0$, i.e. when τ is released at the start of the period of the deferrable server σ. Hence, a *critical instant* occurs for $\varphi^\tau = 0$.

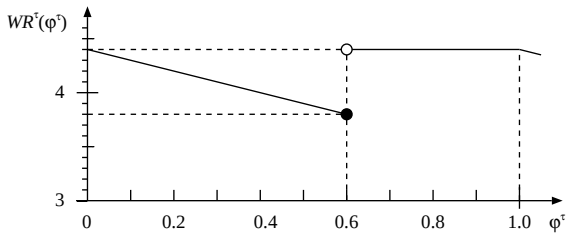

Figure 5. Worst-case response time of task τ as a function of the first release time φ^τ using a deferrable server σ^D for system S.

5.4. Investigating best-case response times

We will now explore the example by considering the best-case response times for the task τ for specific values of the first release time φ^τ. Unlike worst-case response times, we cannot restrict φ^τ to values in the interval $[0, \gcd(T^\tau, T^\sigma))$, but have to consider values in the interval $[0, T^\sigma)$ instead. This is caused by the fact that the response time of τ in the start-up phase can be smaller than the response time in the stable phase, as illustrated for $\varphi^\tau = 0.8$ in Figure 11 in Appendix A. Although the relative phasing of the 1^{st} job of τ compared to the 1^{st} replenishment of σ^D is identical to that of the 4^{th} job of τ compared to the 6^{th} replenishment of σ^D, the response time of the 1^{st} job $R_1^\tau = 3.0$ and of the 4^{th} job $R_4^\tau = 3.2$. These differences in response times are caused by the fact that the execution of the 1^{st} job is *not* influenced by earlier jobs, whereas the execution of

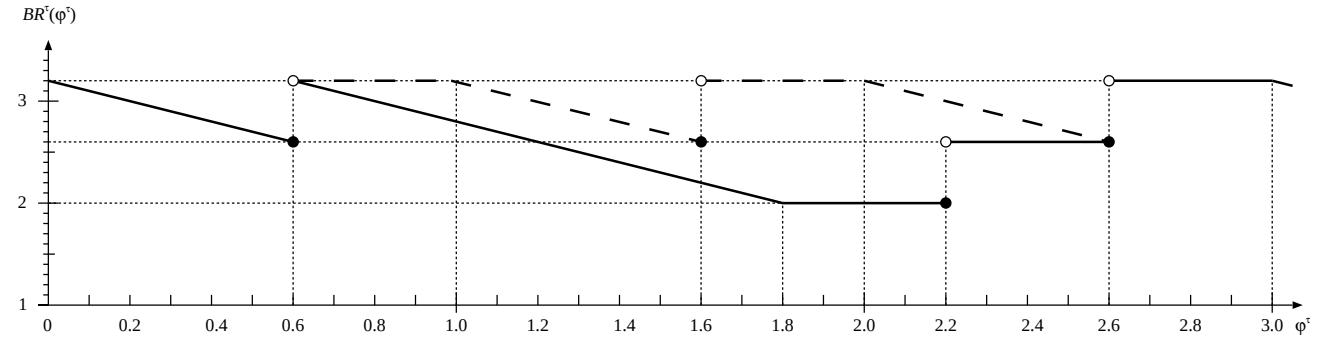

Figure 6. Best-case response time of task τ during its lifetime as a function of the first release time φ^τ using a deferrable server σ^D for system S. The dashed line shows the shortest response time in the stable phase.

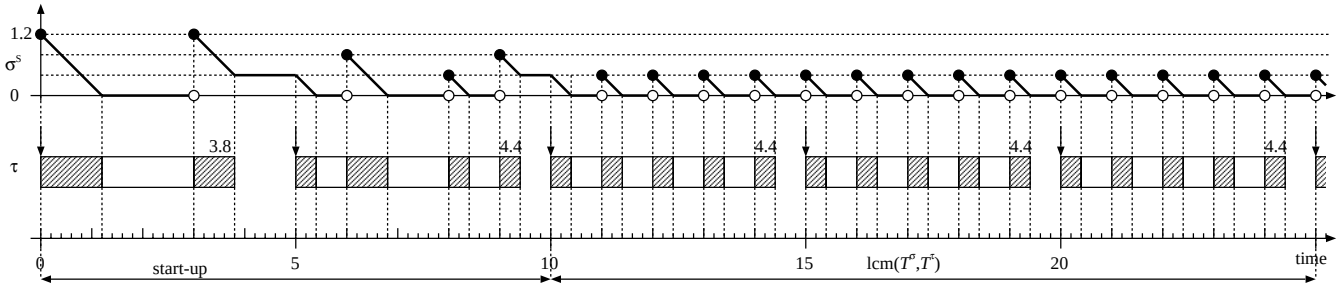

Figure 7. Timeline for S with a simultaneous release of task τ and sporadic server σ^S.

the 4^{th} job is. We merely observe that because the executions of the jobs of τ are independent under a periodic server in Section 4 we could restrict φ^τ to $[0, \gcd(T^\tau, T^\sigma))$ for the best-case response time analysis in that section.

The best-case response times of τ are shown as a function of φ^τ in Figure 6. The dashed line in this figure shows for which values of φ^τ the shortest response time in the stable phase is larger than the shortest response time in the start-up phase. From this figure, we draw the following conclusions. Firstly, the best-case response time under arbitrary phasing is 2.0, which is equal to the computation time C^τ of τ. Secondly, if we only consider response times of τ in the stable phase, the shortest response time becomes 2.6. Finally, $BR^\tau(\varphi^\tau)$ is determined by the start-up phase for phasings $\varphi^\tau \in (0.6, 2.6)$.

6. Analysis for sporadic servers revisited

Similarly to a deferrable server, a sporadic server suspends itself in case no associated task is ready to use it. Moreover, this relative advantage of a sporadic server compared to a periodic server also cannot be exploited for $\widetilde{\Gamma}$ because of the presence of soft real-time tasks. Using the same example system S as used for deferrable servers, we will show that the existing worst-case response time analysis can be improved when a sporadic server is exclusively used for hard real-time tasks.

Figure 7 shows a timeline with the available processing time of sporadic server σ^S and the executions of task τ with a first release of τ 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 τ 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 τ are independent of the first release φ^τ 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* and an *optimal instant* occurs for every value of φ^τ. Next, the end-jitter EJ^τ of task τ is constant, i.e.

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

When we ignore the initial start-up phase, the response time

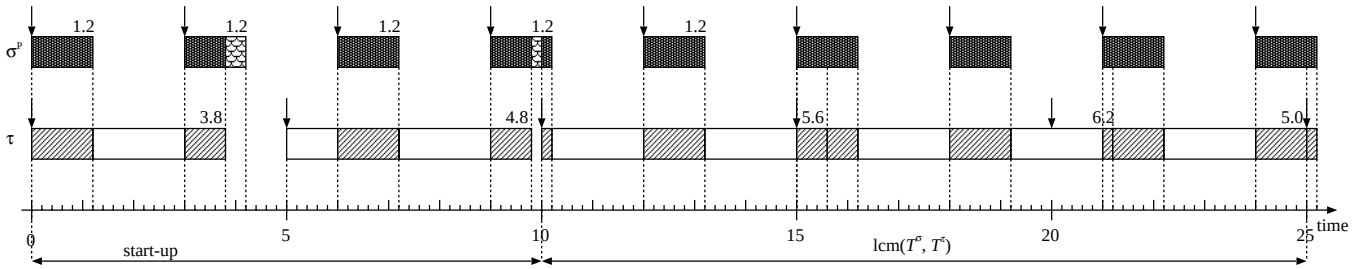

Figure 8. Timeline of task τ using a periodic server σ^{P}.

of the task τ is constant and equal to 4.4.

From $WR^{\tau} = 4.4$, we conclude that S is schedulable under a sporadic server. Hence, we can draw the same conclusions for a sporadic server as we did for a deferrable server in Section 5.2. In particular, the worst-case response time analysis presented in [9] can be improved when a sporadic server is exclusively used for hard real-time tasks.

We conclude this section with two observations. Firstly, the execution of any job of τ after the 1^{st} job in Figure 7 is dependent on the execution of a previous job of τ. Secondly, 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^{τ} of task τ and the capacity C^{σ} of the server σ^{S}, i.e. $\gcd(C^{\tau}, C^{\sigma}) = \gcd(1.2, 2.0) = 0.4$.

7. Discussion

In Section 4 it was shown that for Γ with $C^{\tau} = 2$, $T^{\tau} = D^{\tau} = 5$, and $T^{\sigma} = 5$ a periodic server would need at least a capacity $C^{\sigma}_{min} = 1.5$ to make the task schedulable. In this section, we will remove the constraint on the deadline to investigate the behavior of the system for a capacity $C^{\sigma} = 1.2$. This allows us to compare results for the periodic server with those of the deferrable server in Section 5 and the sporadic server in Section 6. This section includes an architectural consideration and a brief discussion on the notion of busy period for the hard real-time task τ for the class of systems Γ.

7.1. Periodic server revisited

Figure 8 shows a timeline with a simultaneous release of task τ and a periodic server σ^{P}. From this figure, we draw the following conclusions. Firstly, the start-up phase has a length of 10 and contains the executions of two jobs of τ. Secondly, the worst-case response time for a phasing $\varphi^{\tau} = 0$ is equal to $WR^{\tau}(0) = 6.2$, which is larger than the period of the task, and assumed for a job in the stable phase. Because $WR^{\tau}(0) > T^{\tau}$, the executions of jobs of τ are not necessarily independent. Similar to the case of deadlines larger than periods for FPPS [13], a job of a task may delay the start-time of a next job of that task. As a result, the worst-case response time of a job is not necessarily found for the first job upon a critical instant. Thirdly, the best-case response time for a phasing $\varphi^{\tau} = 0$ is equal to $BR^{\tau}(0) = 3.8$, and assumed for a job in the start-up phase.

We will now explore the example in more detail by considering the worst-case and best-case response times of task τ as a function of φ^{τ}; see also the timelines shown in Figures 15 and 16 in Appendix B.2. For the worst-case response times, we can restrict φ^{τ} to values in the interval $[0, \gcd(T^{\sigma}, T^{\tau}))$; see Figure 9. From this figure, we conclude that the worst-case response time WR^{τ} of task τ is equal to 6.2 and assumed for $\varphi^{\tau} = 0$. Hence, a critical instant occurs for $\varphi^{\tau} = 0$, and Figure 8 therefore illustrates a critical instant.

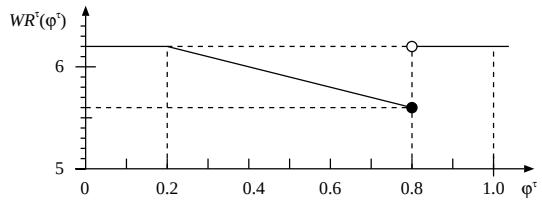

Figure 9. Worst-case response times of task τ as a function of the first release time φ^{τ} using a periodic server σ^{P} for system S.

Because the executions of the jobs of τ are not independent, we have to consider values of φ^{τ} in the interval $[0, T^{\sigma})$ for best-case response times; see Figure 10. From this figure, we conclude that the best-case response time BR^{τ} of task τ is equal to 3.8 and also assumed for $\varphi^{\tau} = 0$. Hence, an optimal instant occurs for $\varphi^{\tau} = 0$, and Figure 8 therefore also illustrates an optimal instant.

7.2. A comparison of servers

As already became clear from the description of periodic servers, the processing time of a periodic server is provided

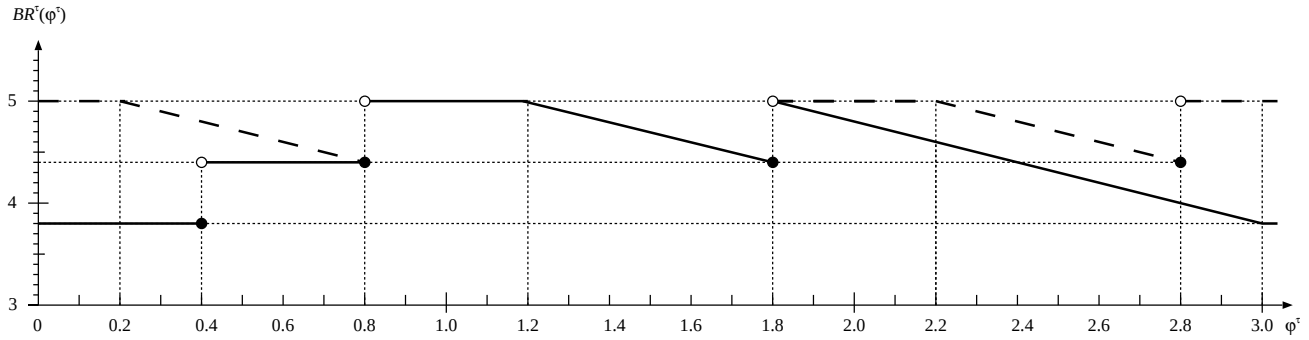

Figure 10. Best-case response time of task τ during its lifetime as a function of the first release time φ^τ using a periodic server σ^P for system S. The dashed line shows the best-case response time in the stable phase.

irrespective of tasks associated with the server being ready for execution. Stated in other words, a periodic server provides its processing time independent of the execution of its associated tasks. As a result, the execution of a task is independent of (the absence or presence of) lower priority tasks. Unlike periodic servers, the execution of a task can be influenced by lower priority tasks for sporadic and deferrable servers. As a result, the existing worst-case response time analysis of hard real-time tasks as presented in [9] can be improved for both deferrable servers and sporadic servers when a server is exclusively used for hard real-time tasks. In particular, we have shown for a specific example of a class of systems Γ that a periodic server needs more capacity to make the system schedulable and results in a larger worst-case response time for a task than a deferrable server and a sporadic server. Without soft real-time tasks, the relative merits of both deferrable servers and sporadic servers compared to periodic servers can therefore be exploited for hard real-time tasks.

Using examples of a class of systems Γ with $D^\tau \leq T^\tau$, we have shown that the execution of a job of task τ is not affected by the execution of a previous job of that task for a periodic server. This is an immediate consequence of the task independent provision of processing time by a periodic server. Conversely, the provision of processing time of a deferrable server and a sporadic server does depend on the execution of associated tasks, and the execution of a job of a task can therefore be affected by a previous job of that task. As a result, the worst-case response time of a task is not necessarily assumed for the first job when released at a critical instant when using a deferrable server or sporadic server. Moreover, the response time of a job of a task during the start-up phase can be smaller than the shortest response time of a job during the stable phase. Hence, for both deferrable servers and sporadic servers, we have to take the start-up phase into account for best-case response time analysis. This can be an important concern for systems with mode-changes [19] in a distributed context where jitter has to be taken into account [5, 17, 20].

For sporadic servers, we observed that the provision of processing time can become heavily fragmented.

Finally, we considered the behavior of a task τ under a periodic server assuming the same characteristics for the server as we did for the deferrable server and the sporadic server by removing the constraint on the deadline of the task. We observed that the worst-case response time of the task became larger than the period, giving rise to similar complexities from an analytical perspective for the periodic server as for the deferrable server and sporadic server. An overview of the response times and jitter for the task for the different types of servers is given in Table 2.

	WR^τ	BR^τ	EJ^τ
periodic server	6.2	3.8	2.4
deferrable server	4.4	2.0	2.4
sporadic server	4.4	3.8	0.6

Table 2. Response times and jitter for τ for different types of servers.

7.3. An architectural consideration

Above we have shown that without soft real-time tasks, the relative merits of both deferrable servers and sporadic servers compared to periodic servers can be exploited for hard real-time tasks for a specific example of a class of systems Γ. To better understand the relative merits of the various types of servers, the following two topics require further

investigation:

- exact worst-case response time analysis of hard real-time tasks under H-FPPS for deferrable servers and sporadic servers;
- the impact of the selection of a particular type of server on the schedulability of lower priority servers, given the required characteristics of that server to guarantee schedulability of its associated tasks.

We observe that schedulability of periodic servers and sporadic servers can be determined by means of classical worst-case response time analysis for FPPS and that schedulability of deferrable servers has been addressed in [2].

The result of this research may have an interesting implication for resource management from a software architecture point of view. To this end, consider the following two complementary approaches. On the one hand, an application may be viewed to consist of a set of tasks, and a server can be appointed to the entire application. Schedulability of the application is subsequently determined by means of the schedulability analysis of the hard real-time tasks of the application. As illustrated in [9], periodic servers are to be preferred for this approach. On the other hand, an application may be viewed to consist of multiple disjunct sets of tasks, where all tasks in a set have the same type, e.g. a set of hard real-time tasks, a set of firm real-time tasks, and a set of soft real-time tasks. A dedicated server is subsequently appointed to each set of tasks. Schedulability of the application is once again determined by the schedulability of the set of hard real-time tasks. When periodic servers do not dominate deferrable servers and sporadic servers when servers are exclusively used for hard real-time tasks, the schedulability will play a role in the selection of the server in this approach. As a consequence, this latter approach may be preferred from a schedulability point of view, and therefore also from an architectural point of view.

This latter subdivision of the set of tasks of applications can be complemented with dedicated spare capacity allocation strategies and mechanisms. As an example, [18] distinguishes different types of reservations (e.g. hard, firm, and soft) based on the way they compete for spare capacity. Alternatively, one can decide to allocate the gain time of a server appointed to the hard real-time tasks of an application to the sets of firm and soft real-time tasks of that application. Further elaboration of these alternatives are a topic of future work.

7.4. Towards a definition of busy period

For H-FPPS, we are investigating the notion of busy period. In this section, we briefly consider a busy period for the hard real-time task τ for the class of systems Γ.

Based on the timelines in Appendix A and B.2 we consider a busy period to start at time t_S when

- a task τ is activated;
- deletion of the activations of all jobs of τ priori to t_S does not change the execution of the job activated at time t_S.

Similarly, we consider the busy period to end at time t_E with the latest completion of a job of task τ before the start of the next busy period. The busy period thus identified is given by $[t_S, t_E)$.

For Figure 2 in Section 4 with a timeline using a periodic server, the busy periods are given by the response intervals $[a_k^\tau, c_k^\tau)$ of the jobs of τ.

Now consider Figure 11 in Appendix A with timelines using a deferrable server. Given the description above, the timelines contain a start of a busy period at

- time $t = \varphi^\tau + k \cdot \text{lcm}(T^\sigma, T^\tau)$ with $k \in \mathbb{N}$ for $\varphi^\tau \in \{0, 0.2, 0.4\}$;
- time $t = \varphi^\tau + k \cdot \text{lcm}(T^\sigma, T^\tau)$ with $k \in \mathbb{N}$ and time $t = \varphi^\tau + T^\tau + k \cdot \text{lcm}(T^\sigma, T^\tau)$ with $k \in \mathbb{N}$ for $\varphi^\tau = 0.6$;
- time $t = \varphi^\tau$ and time $t = \varphi^\tau + T^\tau + k \cdot \text{lcm}(T^\sigma, T^\tau)$ with $k \in \mathbb{N}$ for $\varphi^\tau = 0.8$.

Considering Figure 7 in Section 6 with timelines using a sporadic server, we conclude that the timeline contains a single start of a busy period at time $t = 0$, but that period never ends. Hence, we have to revert to other means to restrict the number of jobs to be considered to determine the worst-case response time of a task, e.g. by exploiting the periodic behavior in the stable phase, similar to the approach as described in, for example, [14].

Finally, consider Figure 15 in Appendix B.2. This figure illustrates busy periods $[0, 3.8)$, $[5, 9.8)$, $[10, 25)$, and the start of a busy period at time $t = 25$.

8. Conclusion

In this paper, we revisited worst-case response time analysis of real-time tasks under hierarchical FPPS. We showed by means of an example that existing worst-case response time analysis can be improved for deferrable servers and sporadic servers when a server is exclusively used for hard real-time tasks. Moreover, we showed that 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. Finally, we briefly investigated best-case response times of a task.

Exact worst-case response time analysis of hierarchical FPPS using deferrable servers and sporadic servers is a topic of future work, and we are currently re-investigating the notions of critical instant and busy period in this context.

Acknowledgement

We thank Robert I. Davis and Alan Burns from the University of York, UK, for their feedback on a preliminary version of a related document, and for helping us to better understand their work.

References

[1] L. Almeida. Response-time analysis and server design for hierarchical scheduling. In *Proc. Work-in-Progress (WiP) session of the* 24^{th} *IEEE Real-Time Systems Symposium (RTSS)*, December 2003.

[2] G. Bernat and A. Burns. New results on fixed priority aperiodic servers. In *Proc.* 20^{th} *IEEE Real-Time Systems Symposium (RTSS)*, pp. 68–78, December 1999.

[3] E. Bini and G. Lipari. Resource partitioning among real-time applications. In *Proc.* 15^{th} *Euromicro Conference on Real-Time Systems (ECRTS)*, pp. 151–158, July 2003.

[4] R.J. Bril. *Real-time scheduling for media processing using conditionally guaranteed budgets*. PhD thesis, Technische Universiteit Eindhoven (TU/e), The Netherlands, July 2004. http://alexandria.tue.nl/extra2/200412419.pdf.

[5] R.J. Bril, E.F.M. Steffens, and W.F.J. Verhaegh. Best-case response times and jitter analysis of real-time tasks. *Journal of Scheduling*, 7(2):133–147, March 2004.

[6] G.C. Buttazzo. *Hard real-time computing systems - predictable scheduling algorithms and applications (*2^{nd} *edition)*. Springer, 2005.

[7] P.J.L Cuijpers and R.J. Bril. Towards periodic budgeting in real-time calculus. CS-Report 06-22, Technische Universiteit Eindhoven (TU/e), The Netherlands, July 2006.

[8] R. Davis and A. Burns. Hierarchical fixed priority pre-emptive scheduling. Technical Report YCS-385, University of York, Department of Computer Science, UK, April 2005. http://www.cs.york.ac.uk/ftpdir/reports/YCS-2005-385.pdf.

[9] R.I. Davis and A. Burns. Hierarchical fixed priority pre-emptive scheduling. In *Proc.* 26^{th} *IEEE Real-Time Systems Symposium (RTSS)*, pp. 389–398, December 2005.

[10] M. Joseph, editor. *Real-Time Systems: Specification, Verification and Analysis*. Prentice Hall, 1996.

[11] M. Joseph and P. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395, 1986.

[12] M.H. Klein, T. Ralya, B. Pollak, R. Obenza, and M. González Harbour. *A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems*. Kluwer Academic Publishers, 1993.

[13] J.P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Proc.* 11^{th} *IEEE Real-Time Systems Symposium (RTSS)*, pp. 201–209, December 1990.

[14] J.Y.T. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance Evaluation*, 2(4):237–250, December 1982.

[15] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a real-time environment. *Journal of the ACM*, 20(1):46–61, January 1973.

[16] C. Mercer, R. Rajkumar, and J. Zelenka. Temporal protection in real-time operating systems. In *Proc.* 11^{th} *IEEE Workshop on Real-Time Operating Systems and Software (RTOSS)*, pp. 79–83, May 1994.

[17] J.C. Palencia Gutiérrez, J.J. Gutiérrez García, and M. González Harbour. Best-case analysis for improving the worst-case schedulability test for distributed hard real-time systems. In *Proc.* 10^{th} *EuroMicro Workshop on Real-Time Systems*, pp. 35–44, June 1998.

[18] 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.

[19] J. Real and A. Crespo. Mode change protocols for real-time systems: a survey and a new protocol. *Real-Time Systems*, 26(2):161–197, March 2004.

[20] O. Redell and M. Sanfridson. Exact best-case response time analysis of fixed priority scheduled tasks. In *Proc.* 14^{th} *Euromicro Conference on Real-Time Systems (ECRTS)*, pp. 165–172, June 2002.

[21] S. Saewong, R. Rajkumar, J.P. Lehoczky, and M.H. Klein. Analysis of hierarchical fixed-priority scheduling. In *Proc.* 14^{th} *Euromicro Conference on Real-Time Systems (ECRTS)*, pp. 152–160, June 2002.

[22] I. Shin and I. Lee. Periodic resource model for compositional real-time guarantees. In *Proc. 24th IEEE Real-Time Systems Symposium (RTSS)*, pp. 2–13, December 2003.

[23] B. Sprunt, L. Sha, and J.P. Lehoczky. Aperiodic task scheduling for hard real-time systems. *Real-Time Systems*, 1(1):27–60, June 1989.

[24] J.K. Strosnider, J.P. 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.

[25] L. Thiele, S. Chakraborty, and M. Naedele. Real-time calculus for scheduling hard real-time systems. In *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 4, pp. 101–104, May 2000.

A. Timelines for S using a deferrable server

Figures 11 till 13 show timelines for S of task τ for various values of the first release φ^τ using a deferrable server σ^D. As discussed in Section 5.3, it is sufficient to consider values of φ^τ in the interval $[0, \gcd(T^\sigma, T^\tau)) = [0,1)$ to determine the worst-case response time WR^τ as a function of φ^τ, because the response times of jobs in the start-up phase are at most equal to the response time of jobs in the stable phase. Figure 11 has been used to determine $WR^\tau(\varphi^\tau)$. Because the response time of τ in the start-up phase can be smaller than the response time in the stable phase, we need to consider values of φ^τ in the interval $[0, T^\sigma) = [0,3)$ to determine the best-case response time BR^τ as a function of φ^τ. We therefore also included Figures 12 and 13.

For this particular example with $U^\tau = U^\sigma$, the completion of the latest interval with capacity loss coincides with the end of the start-up phase and therefore the start of the stable phase. We merely observe that the stable phase for $\varphi^\tau = 0$ is simply shifted compared to the stable phase for $\varphi^\tau = 1.0$ and $\varphi^\tau = 2.0$. Similar observations hold for $\varphi^\tau = 0.2$, $\varphi^\tau = 0.4$, and $\varphi^\tau = 0.8$. There happens to be an exception for $\varphi^\tau = 0.6$, i.e. the stable phase is not simply shifted compared to the stable phase for $\varphi^\tau = 2.6$. This anomaly happens upon a sudden decrease in the number of jobs in the start-up phase during the increase of φ^τ when φ^τ becomes 2.6.

B. Timelines for a periodic server

This appendix shows timelines for an example system belonging to the class of systems Γ, where the single hard real-time task τ has characteristics $C^\tau = 2$ and $T^\tau = 5$ and the periodic server σ^P has a period $T^\sigma = 3$. The capacity of the server $C^\sigma = 1.5$ in Section B.1 and $C^\sigma = 1.2$ in Section B.2.

B.1. Capacity $C^\sigma = 1.5$

Figures 14 show timelines for task τ and periodic server σ^P with period $T^\sigma = 3$ and capacity $C^\sigma = 1.5$ for various values of the first release φ^τ.

B.2. Capacity $C^\sigma = 1.2$

Figures 15 till 18 show timelines of task τ for various values of the first release φ^τ using a periodic server σ^P with period $T^\sigma = 3$ and capacity $C^\sigma = 1.2$. Similar to the figures in Section A, the start-up phase ends when the latest interval with capacity loss is completed. Also similar to the figures in Section A, the stable phase is typically simply shifted. In this case, there happens to be an exception for $\varphi^\tau = 0.8$, i.e. the stable phase is not simply shifted compared to the stable phase for $\varphi^\tau = 1.8$. This anomaly happens upon a sudden decrease in the number of jobs in the start-up phase during the increase of φ^τ when φ^τ becomes 0.8.

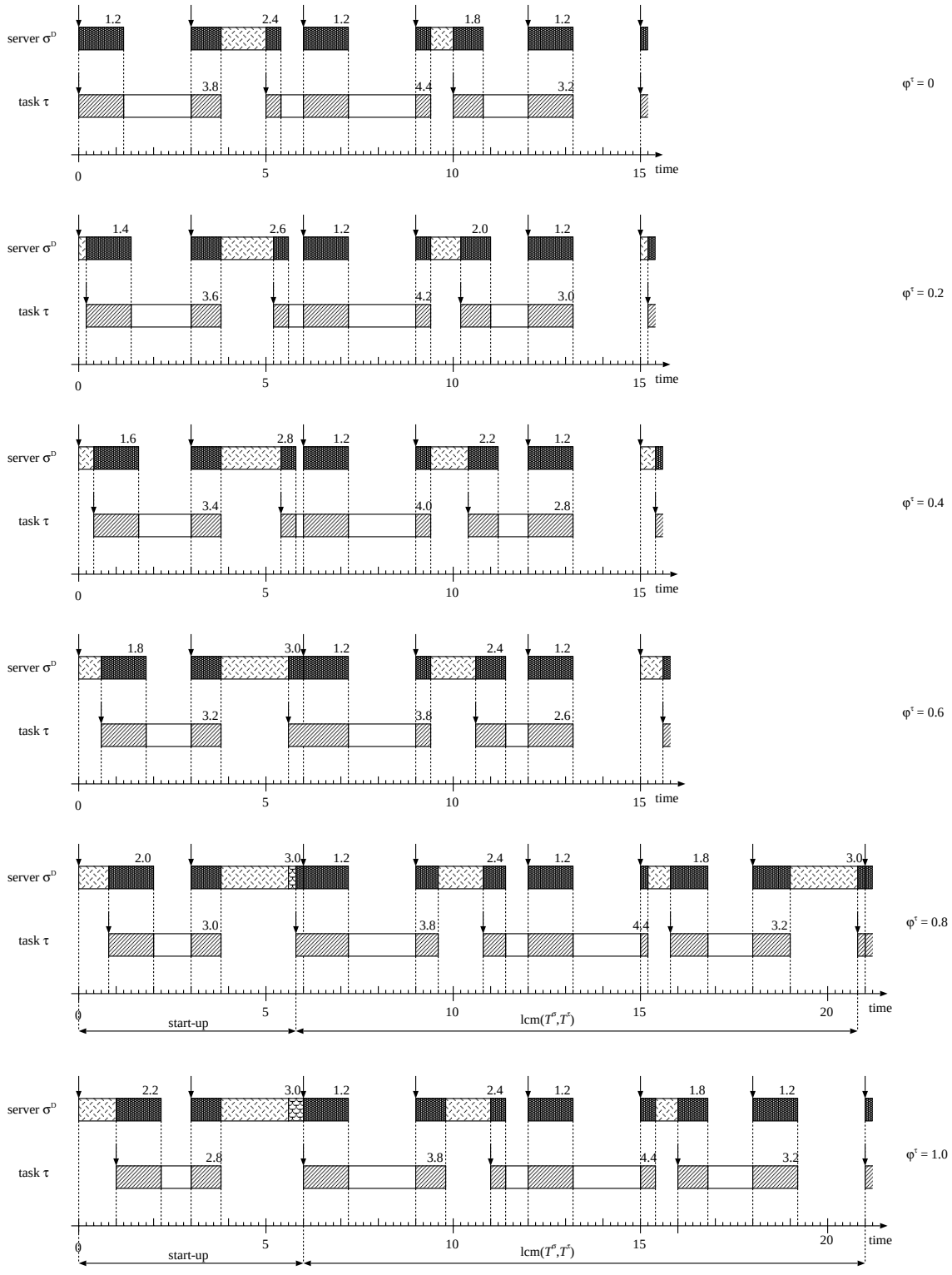

Figure 11. Timelines for S with a first release of task τ at $\varphi^{\tau} \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ using a deferrable server σ^{D}.

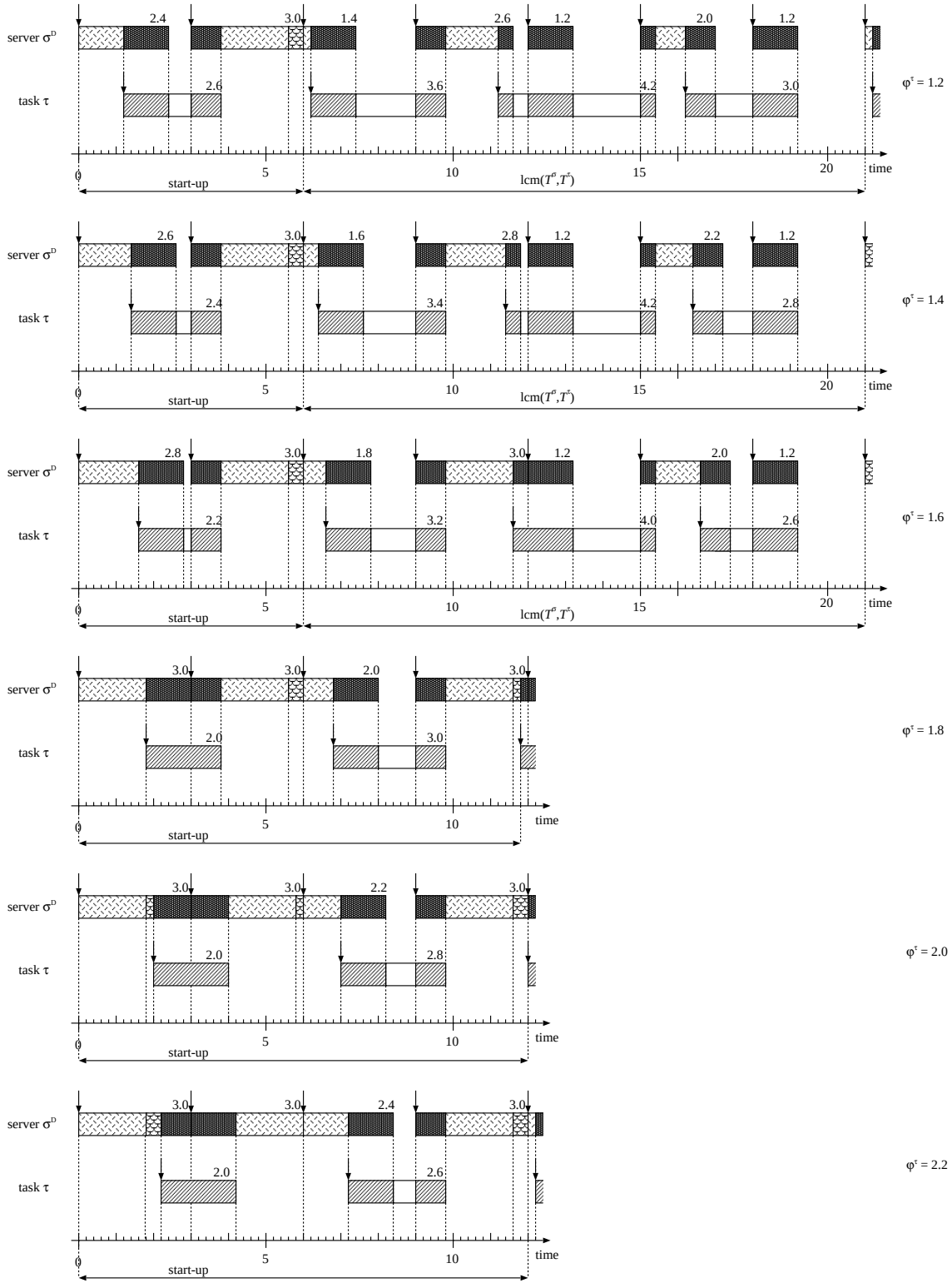

Figure 12. Timelines for S with a first release of task τ at $\varphi^{\tau} \in \{1.2, 1.4, 1.6, 1.8, 2.0, 2.2\}$ using a deferrable server σ^{D}.

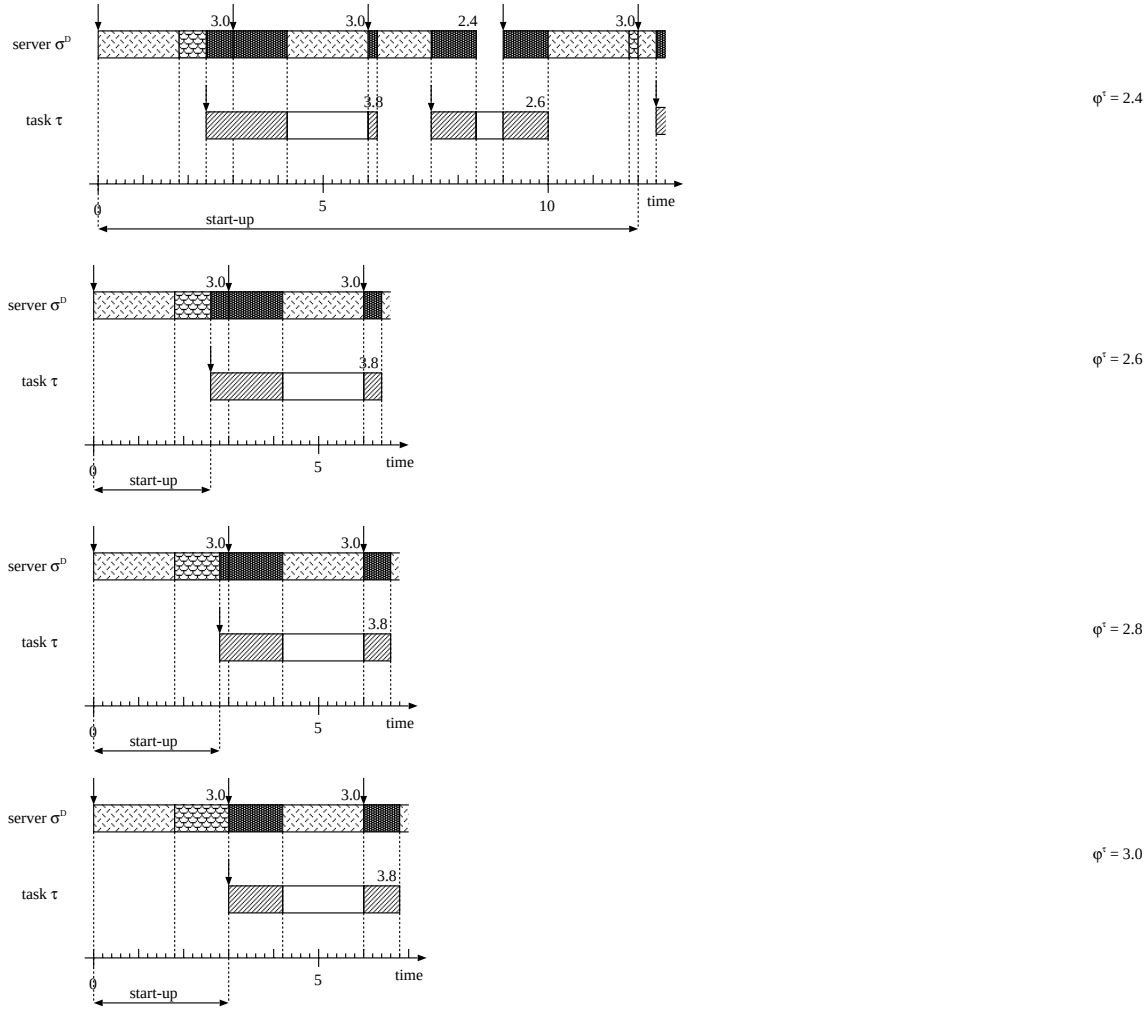

Figure 13. Timelines for S with a first release of task τ at $\varphi^{\tau} \in \{2.4, 2.6, 2.8, 1.0\}$ using a deferrable server σ^{D}.

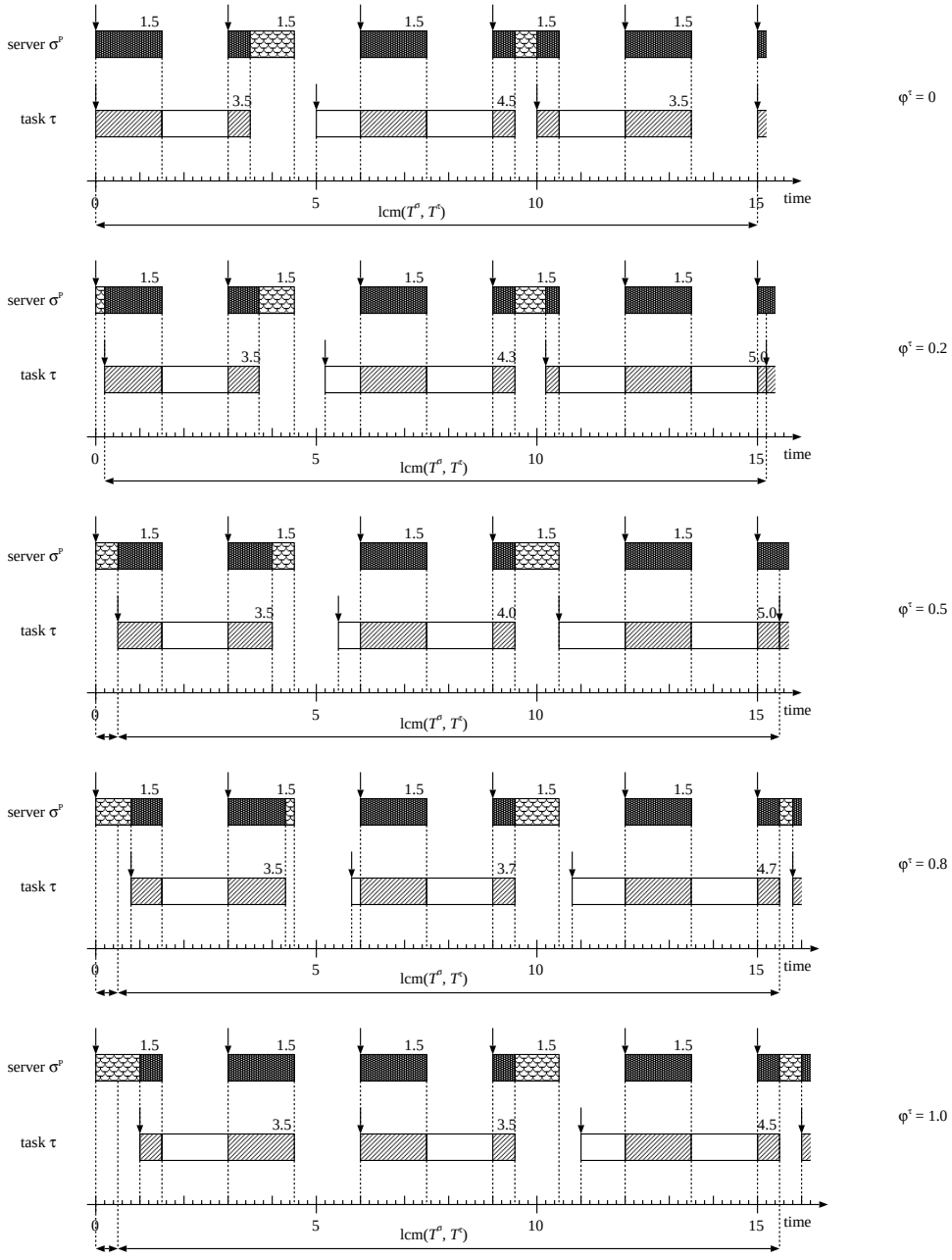

Figure 14. Timelines with a first release of task τ at $\varphi^{\tau} \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ using a periodic server σ^{P} with capacity $C^{\sigma} = 1.5$.

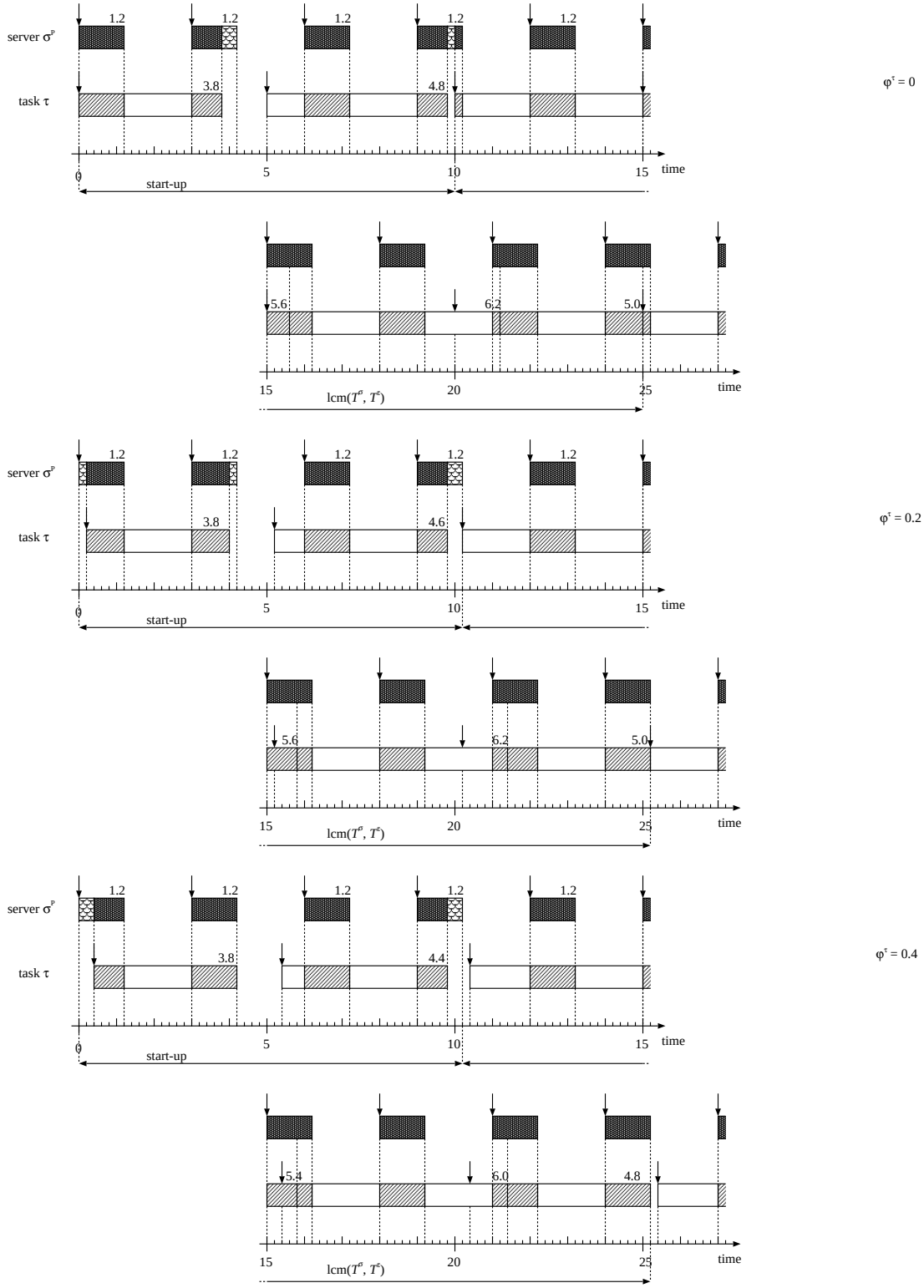

Figure 15. Timelines for task τ with a first release at $\varphi^{\tau} \in \{0, 0.2, 0.4\}$ using a periodic server σ^{P}.

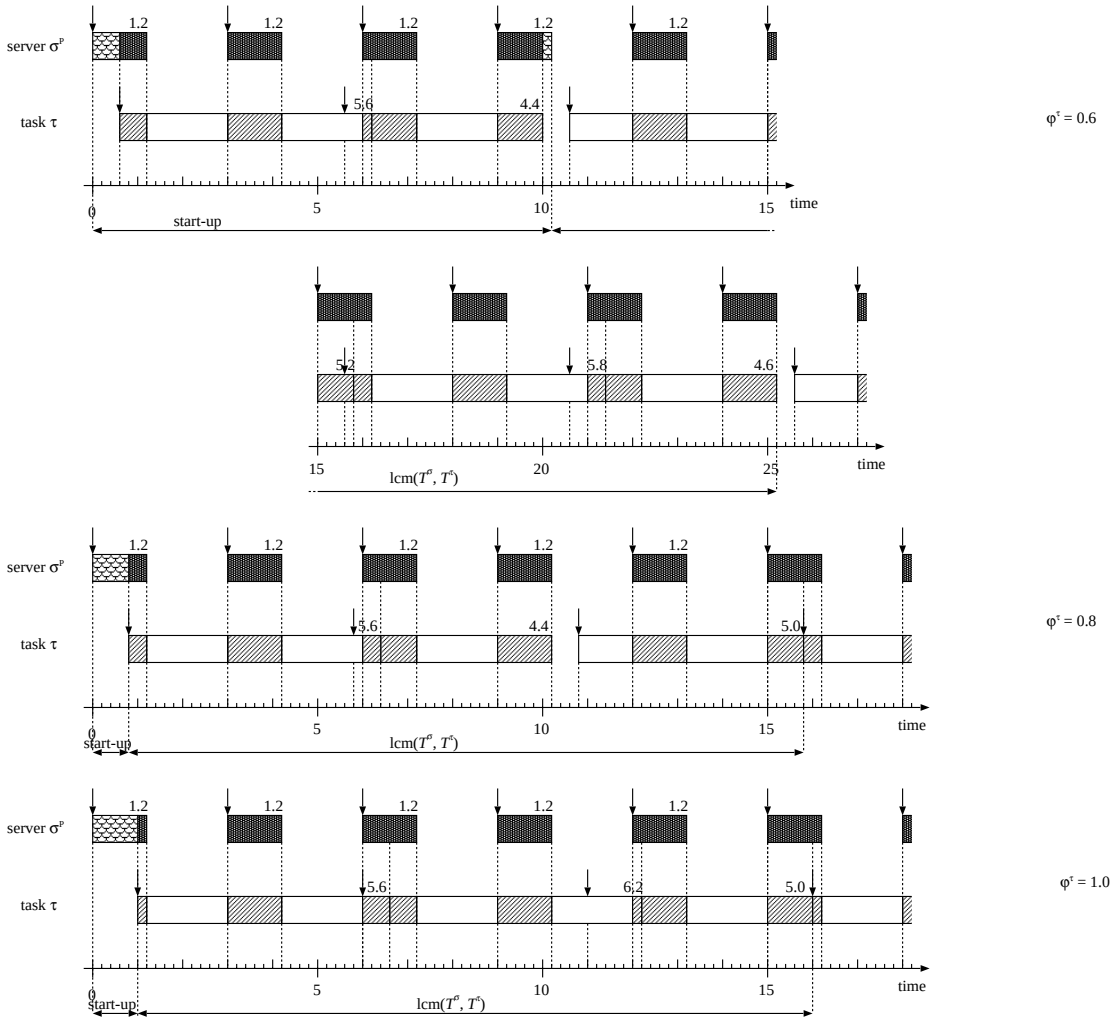

Figure 16. Timelines for task τ with a first release at $\varphi^{\tau} \in \{0.6, 0.8, 1.0\}$ using a periodic server σ^{P}.

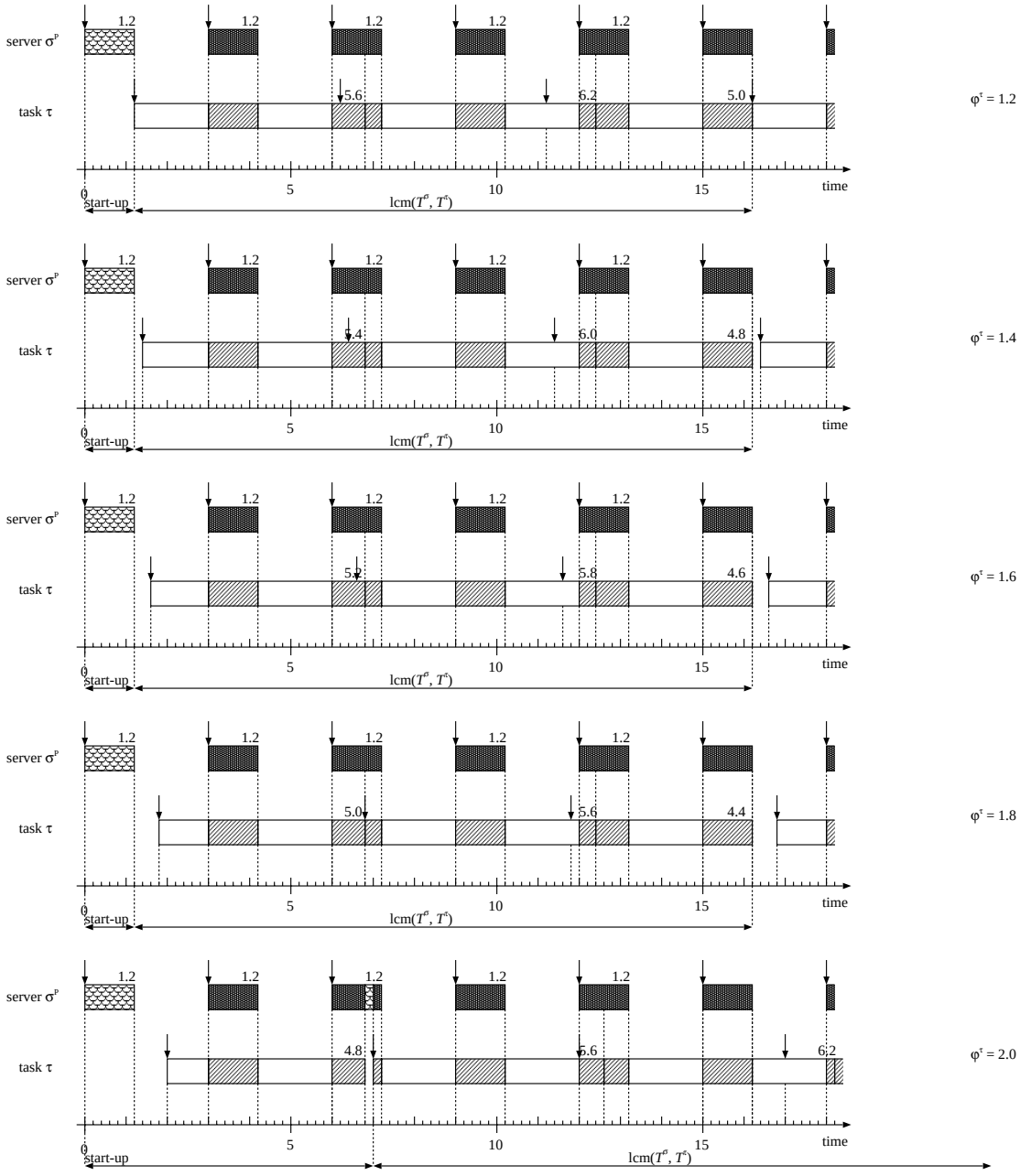

Figure 17. Timelines for task τ with a first release at $\varphi^{\tau} \in \{1.2, 1.4, 1.6, 1.8, 2.0\}$ using a periodic server σ^{P}.

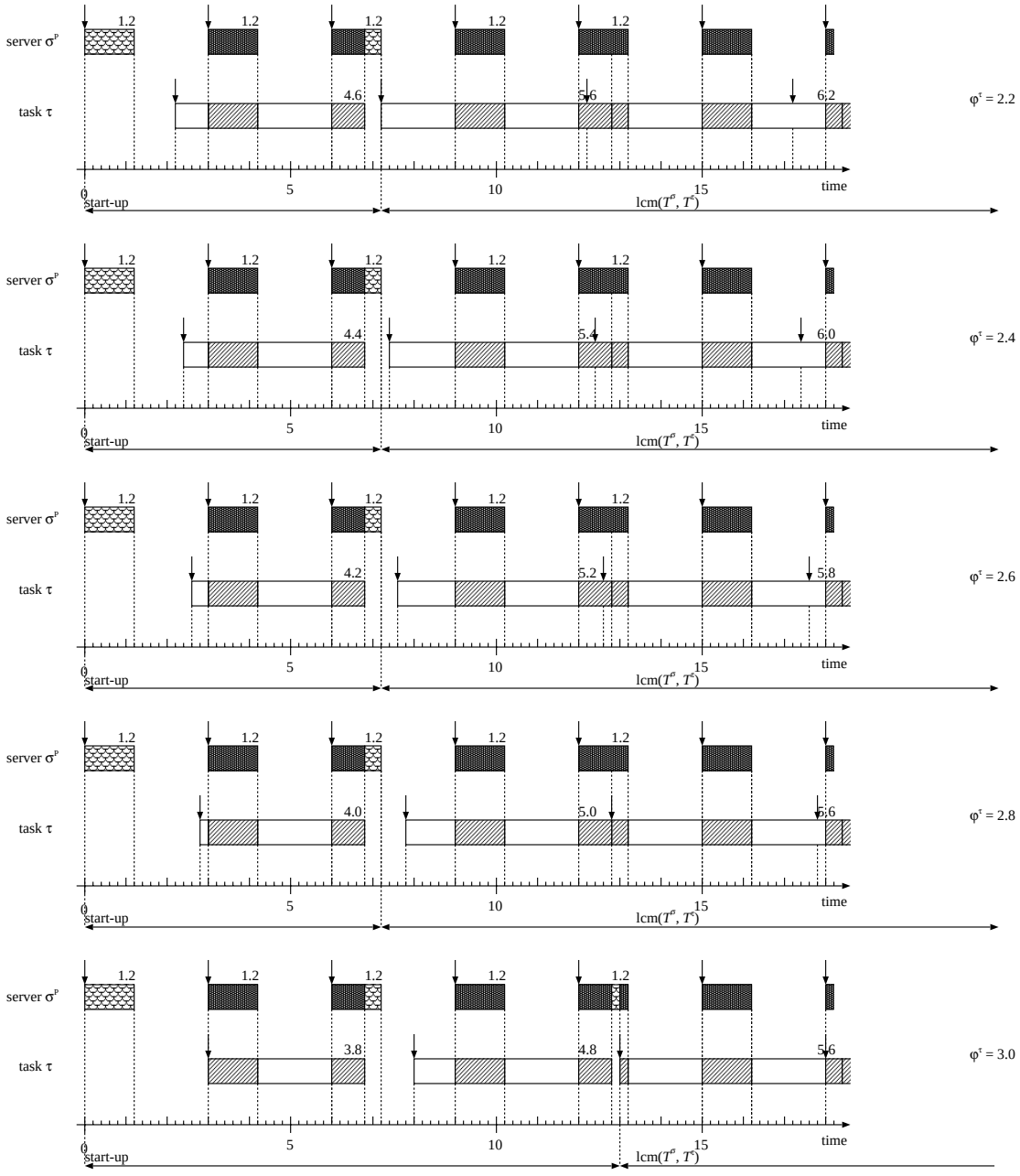

Figure 18. Timelines for task τ with a first release at $\varphi^{\tau} \in \{2.2, 2.4, 2.6, 2.8, 3.0\}$ using a periodic server σ^{P}.