

# Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption revisited

**Citation for published version (APA):**

Bril, R. J., Lukkien, J. J., & Verhaegh, W. F. J. (2006). *Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption revisited*. (Computer science reports; Vol. 0634). Technische Universiteit Eindhoven.

**Document status and date:**

Published: 01/01/2006

**Document Version:**

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

**Please check the document version of this publication:**

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

**General rights**

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

[www.tue.nl/taverne](http://www.tue.nl/taverne)

**Take down policy**

If you believe that this document breaches copyright please contact us at:

[openaccess@tue.nl](mailto:openaccess@tue.nl)

providing details and we will investigate your claim.

# Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption revisited

Reinder J. Bril and Johan J. Lukkien

*Technische Universiteit Eindhoven (TU/e),  
Den Dolech 2, 5600 AZ Eindhoven,  
The Netherlands  
{r.j.bril, j.j.lukkien}@tue.nl*

Wim F.J. Verhaegh

*Philips Research Laboratories,  
Prof. Holstlaan 4, 5656 AA Eindhoven,  
The Netherlands  
wim.verhaegh@philips.com*

## Abstract

*Fixed-priority scheduling with deferred preemption (FPDS) has been proposed in the literature as a viable alternative to fixed-priority pre-emptive scheduling (FPPS), that obviates the need for non-trivial resource access protocols and reduces the cost of arbitrary preemptions.*

*This paper shows that existing worst-case response time analysis of hard real-time tasks under FPDS, arbitrary phasing and relative deadlines at most equal to periods is pessimistic and/or optimistic. The same problem also arises for fixed-priority non-pre-emptive scheduling (FPNS), being a special case of FPDS. This paper provides revised analysis, resolving the problems with the existing approaches. The analysis is based on known concepts of critical instant and busy period for FPPS. To accommodate for our scheduling model for FPDS, we need to slightly modify existing definitions of these concepts. The analysis assumes a continuous scheduling model, which is based on a partitioning of the timeline in a set of non-empty, right semi-open intervals. It is shown that the critical instant, longest busy period, and worst-case response time for a task are suprema rather than maxima for all tasks, except for the lowest priority task, i.e. that instant, period, and response time cannot be assumed. Moreover, it is shown that the analysis is not uniform for all tasks, i.e. the analysis for the lowest priority task differs from the analysis of the other tasks. To build on earlier work, the worst-case response time analysis for FPDS is expressed in terms of known worst-case analysis results for FPPS. The paper includes pessimistic variants of the analysis, which are uniform for all tasks.*

## 1 Introduction

### 1.1 Motivation

Based on the seminal paper of Liu and Layland [27], many results have been achieved in the area of analysis for fixed-priority preemptive scheduling (FPPS). Arbitrary preemption of real-time tasks has a number of drawbacks, though. In systems requiring mutual access to shared resources, arbitrary preemptions induce the need for non-trivial resource access protocols, such as the priority ceiling protocol [32]. In systems using cache memory, e.g. to bridge the speed gap between processors and main memory, arbitrary preemptions induce additional cache flushes and reloads. As a consequence, system performance and predictability are degraded, complicating system design, analysis and testing [13, 18, 24, 29, 33]. Although fixed-priority non-preemptive scheduling (FPNS) may resolve these problems, it generally leads to reduced schedulability compared to FPPS. Therefore, alternative scheduling schemes have been proposed between the extremes of arbitrary preemption and no preemption. These schemes are also known as deferred preemption or co-operative scheduling [11], and are denoted by fixed-priority scheduling with deferred preemption (FPDS) in the remainder of this paper.

Worst-case response time analysis of periodic real-time tasks under FPDS, arbitrary phasing, and relative deadlines within periods has been addressed in a number of papers [10, 11, 13, 24]. The existing analysis is not exact, however. In [10], it has already been shown that the analysis presented in [11, 13, 24] is pessimistic. More recently, it has been shown in [6, 7] that the analysis presented in [10, 11, 13] is optimistic. Unlike the implicit assumptions in those latter papers, the worst-case response time of a task under FPDS and arbitrary phasing is not necessarily assumed for the first job of that task

upon its critical instant. Hence, the existing analysis may provide guarantees for tasks that in fact miss their deadlines in the worst-case. In [8, 9], it has been shown that the latter problem also arises for FPNS, being a special case of FPDS, and its application for the schedulability analysis of controller area network (CAN) [35, 36, 37]. Revised analysis for CAN resolving the problem with the original approach in an evolutionary fashion can be found in [15].

## 1.2 Contributions

This paper resolves the problems with the existing approaches by presenting novel worst-case response time analysis for hard real-time tasks under FPDS, arbitrary phasing and arbitrary relative deadlines. The analysis assumes a *continuous* scheduling model rather than a *discrete* scheduling model [4], e.g. all task parameters are taken from the real numbers. The motivation for this assumption stems from the observation that a discrete view on time is in many situations insufficient; see for example [2, 20, 23]. The scheduling model is based on a partitioning of the timeline in a set of non-empty, right semi-open intervals [14, 20]. The analysis is based on the concepts of *critical instant* [27] and *busy period* [25]. To accommodate for our scheduling model for FPDS, we need to slightly modify the existing definitions of these concepts. To prevent confusion with the existing definition of busy period, we use the term *active period* for our definition in this document.

In this document, we discuss conditions for termination of an active period, and present a sufficient condition with a formal proof. Moreover, we show that the critical instant, longest active period, and worst-case response time for a task are suprema rather than maxima for all tasks, except for the lowest priority task, i.e. that instant, period, and response time cannot be assumed. Our worst-case response time analysis is not uniform for all tasks. In particular, the analysis for the lowest priority task differs from the analysis for the other tasks. To build on earlier results, worst-case response times under FPDS are expressed in terms of worst-case response times and worst-case occupied times [5] under FPPS. We also present pessimistic variants of the analysis, which are uniform for all tasks, and show that the revised analysis for CAN presented in [15] conforms to a pessimistic variant.

## 1.3 Related work

Next to continuous scheduling models, one can find discrete scheduling models in the literature, e.g. in [16, 19], and models in which domains are not explicitly specified [14, 22, 28]. Because the equations for response time analysis depend on the model, we prefer to be explicit about the domains in our model. As mentioned above, our scheduling model is based on a partitioning of the timeline in a set of non-empty, right semi-open intervals. Alternatively, the scheduling model in [28] is based on left semi-open intervals.

In this paper, we assume that each job (or activation) of a task consists of a sequence of non-preemptable subjobs, where each subjob has a known worst-case computation time, and present novel worst-case response time analysis to determine schedulability of tasks under FPDS. Similarly, George et al assume in [16] that the worst-case computation time of each non-preemptive job is known, and present worst-case response time analysis of tasks under FPNS. Conversely, Baruah [3] determines the largest non-preemptive ‘chunks’ into which jobs of a task can be broken up to still ensure feasibility under earliest deadline first (EDF).

For worst-case response time analysis of tasks under FPPS, arbitrary phasing, and relative deadlines at most equal to periods, it suffices to determine the response time of the first job of a task upon its critical instant. For tasks with relative deadlines larger than their respective periods, Lehoczky [25] introduced the concept of a busy period, and showed that all jobs of a task in a busy period need to be considered to determine its worst-case response time. Hence, when the relative deadline of a task is larger than its period, the worst-case response time of that task is not necessarily assumed for the first job of a task when released at a critical instant. Similarly, González Harbour et al [17] showed that if relative deadlines are at most equal to periods, but priorities vary during execution, then again multiple jobs must be considered to determine the worst-case response time. Initial work on pre-emption thresholds [38] failed to identify this issue. The resulting flaw was later corrected by Regehr [31]. Worst-case response time analysis of tasks under EDF and relative deadlines at most equal to periods described by Spuri [34] is also based on the concept of busy period.

## 1.4 Structure

This paper has the following structure. First, in Section 2, we present real-time scheduling models for FPPS and FPDS. Next, worst-case analysis for FPPS is briefly recapitulated in Section 3. Section 4 presents various examples refuting the existing worst-case response time analysis for FPDS. The notion of active period is the topic of Section 5. We present a formal definition of active period and theorems with a recursive equation for the length of an active period and an iterative procedure to determine its value. Worst-case analysis for FPDS is addressed in Section 6. We present a theorem for critical instant and theorems to determine the worst-case response time of a task under FPDS and arbitrary phasing. Section 7 illustrates the worst-case response time analysis by applying it to some examples presented in Section 4. Section 8 compares the notion

of level- $i$  active period with similar definitions in the literature, and presents pessimistic variants of the worst-case response time analysis. The paper is concluded in Section 9.

## 2 Real-time scheduling models

This section starts with a presentation of a basic real-time scheduling model for FPPS. Next, that basic model is refined for FPDS. The section is concluded with remarks.

### 2.1 Basic model for FPPS

We assume a single processor and a set  $\mathcal{T}$  of  $n$  periodically released, independent tasks  $\tau_1, \tau_2, \dots, \tau_n$  with unique, fixed priorities. At any moment in time, the processor is used to execute the highest priority task that has work pending. So, when a task  $\tau_i$  is being executed, and a release occurs for a higher priority task  $\tau_j$ , then the execution of  $\tau_i$  is preempted, and will resume when the execution of  $\tau_j$  has ended, as well as all other releases of tasks with a higher priority than  $\tau_i$  that have taken place in the meantime.

A *schedule* is an assignment of the tasks to the processor. A schedule can be defined as an integer step function  $\sigma : \mathbb{R} \rightarrow \{0, 1, \dots, n\}$ . Informally,  $\sigma(t) = i$  with  $i > 0$  means that task  $\tau_i$  is being executed at time  $t$ , while  $\sigma(t) = 0$  means that the processor is idle. More formally,  $\sigma$  partitions the timeline in a set of non-empty, right semi-open intervals  $\{[t_j, t_{j+1})\}_{j \in \mathbb{Z}}$ , such that  $\sigma(t)$  is right-continuous and piece-wise continuous in each of those intervals, and discontinuous at the ends. At times  $t_j$ , the processor performs a *context switch*. Figure 1 shows an example of the execution of a set  $\mathcal{T}$  of three periodic tasks and the corresponding value of the schedule  $\sigma(t)$ .

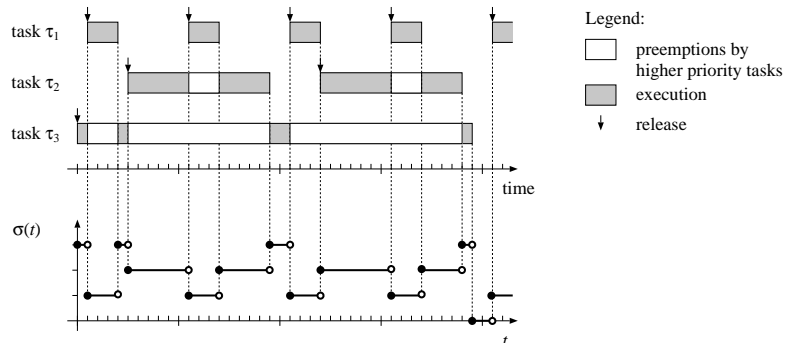


Figure 1. An example of the execution of a set  $\mathcal{T}$  of three independent periodic tasks  $\tau_1$ ,  $\tau_2$ , and  $\tau_3$ , where task  $\tau_1$  has highest priority, and task  $\tau_3$  has lowest priority, and the corresponding value of  $\sigma(t)$ .

Each task  $\tau_i$  is characterized by a (*release*) *period*  $T_i \in \mathbb{R}^+$ , a (*computation time*)  $C_i \in \mathbb{R}^+$ , a (*relative*) *deadline*  $D_i \in \mathbb{R}^+$ , where  $C_i \leq \min(D_i, T_i)$ , and a (*phasing*)  $\varphi_i \in \mathbb{R}^+ \cup \{0\}$ . An (*activation* (or *release*)) *time* is a time at which a task  $\tau_i$  becomes ready for execution. A release of a task is also termed a *job*. The first job of task  $\tau_i$  is released at time  $\varphi_i$  and is referred to as *job zero*. The release of job  $k$  of  $\tau_i$  therefore takes place at time  $a_{ik} = \varphi_i + kT_i$ ,  $k \in \mathbb{N}$ . The (*absolute*) *deadline* of job  $k$  of  $\tau_i$  takes place at  $d_{ik} = a_{ik} + D_i$ . The (*begin* (or *start*)) *time*  $b_{ik}$  and (*finalization* (or *completion*)) *time*  $f_{ik}$  of job  $k$  of  $\tau_i$  is the time at which  $\tau_i$  actually starts and ends the execution of that job, respectively. The set of phasings  $\varphi_i$  is termed the phasing  $\varphi$  of the task set  $\mathcal{T}$ .

The (*active* (or *response*)) *interval* of job  $k$  of  $\tau_i$  is defined as the time span between the activation time of that job and its finalization time, i.e.  $[a_{ik}, f_{ik})$ . The (*response time*)  $r_{ik}$  of job  $k$  of  $\tau_i$  is defined as the length of its active interval, i.e.  $r_{ik} = f_{ik} - a_{ik}$ . Figure 2 illustrates the above basic notions for an example job of task  $\tau_i$ .

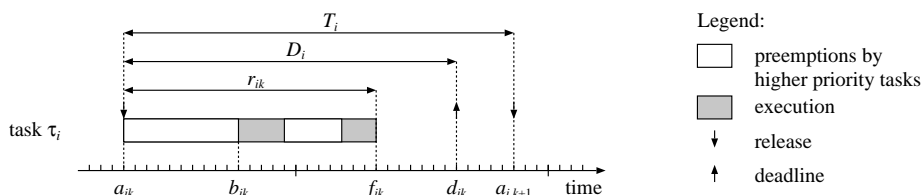


Figure 2. Basic model for task  $\tau_i$ .

The *worst-case response time*  $WR_i$  of a task  $\tau_i$  is the largest response time of any of its jobs, i.e.

$$WR_i = \sup_{\phi, k} r_{ik}. \quad (1)$$

In many cases, we are not interested in the worst-case response time of a task for a particular computation time, but in the value as a function of the computation time. We will therefore use a functional notation when needed, e.g.  $WR_i(C_i)$ . A *critical instant* of a task is defined to be an (hypothetical) instant that leads to the worst-case response time for that task. 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.

We assume that we do not have control over the phasing  $\phi$ , for instance since the tasks are released by external events, so we assume that any arbitrary phasing may occur. This assumption is common in real-time scheduling literature [21, 22, 27]. We also assume other standard basic assumptions [27], i.e. tasks are ready to run at the start of each period and do not suspend themselves, tasks will be preempted instantaneously when a higher priority task becomes ready to run, a job of task  $\tau_i$  does not start before its previous job is completed, and the overhead of context switching and task scheduling is ignored. Finally, we assume that the deadlines are hard, i.e. each job of a task must be completed before its deadline. Hence, a set  $\mathcal{T}$  of  $n$  periodic tasks can be scheduled if and only if

$$WR_i \leq D_i \quad (2)$$

for all  $i = 1, \dots, n$ . For notational convenience, we assume that the tasks are given in order of decreasing priority, i.e. task  $\tau_1$  has highest priority and task  $\tau_n$  has lowest priority.

The (*processor*) *utilization factor*  $U$  is the fraction of the processor time spent on the execution of the task set [27]. The fraction of processor time spent on executing task  $\tau_i$  is  $C_i/T_i$ , and is termed the *utilization factor*  $U_i^\tau$  of task  $\tau_i$ , i.e.

$$U_i^\tau = \frac{C_i}{T_i}. \quad (3)$$

The *cumulative utilization factor*  $U_i$  for tasks  $\tau_1$  till  $\tau_i$  is the fraction of processor time spent on executing these tasks, and is given by

$$U_i = \sum_{j \leq i} U_j^\tau. \quad (4)$$

Therefore,  $U$  is equal to the cumulative utilization factor  $U_n$  for  $n$  tasks.

$$U = U_n = \sum_{j \leq n} U_j^\tau = \sum_{j \leq n} \frac{C_j}{T_j}. \quad (5)$$

In [27], the following necessary condition is determined for the schedulability of a set  $\mathcal{T}$  of  $n$  periodic tasks under any scheduling algorithm.

$$U \leq 1. \quad (6)$$

Unless explicitly stated otherwise, we assume in this document that task sets satisfy this condition.

## 2.2 Refined model for FPDS

For FPDS, we need to refine our basic model of Section 2.1. Each job of task  $\tau_i$  is now assumed to consist of  $m_i$  subjobs. The  $k^{\text{th}}$  subjob of  $\tau_i$  is characterized by a computation time  $C_{ik} \in \mathbb{R}^+$ , where  $C_i = \sum_{k=1}^{m_i} C_{ik}$ . We assume that subjobs are non-preemptable. Hence, tasks can only be preempted at subjob boundaries, i.e. at so-called *preemption points*. For convenience, we will use the term  $F_i$  to denote the computation time  $C_{i,m_i}$  of the final subjob of  $\tau_i$ . Note that when  $m_i = 1$  for all  $i$ , we have FPNS as special case.

## 2.3 Concluding remarks

In this document, we will use the superscript P to denote FPPS, e.g.  $WR_i^P$  denotes the worst-case response time of task  $\tau_i$  under FPPS and arbitrary phasing. Similarly, we will use the superscripts D and N to denote FPDS and FPNS, respectively.

In our basic model for FPPS, we introduced notions for points in time with a subscript identifying a task and optionally a job of that task, e.g.  $a_{ik}$  is the release time of job  $k$  of task  $\tau_i$ . In this document, we will need similar notions that are expressed relative to a particular moment in time, e.g. the relative release time of the first job of a task at or after time  $t_s$ . We will therefore also use relative versions of the notions, where relative can refer to the identification of the job and/or to the

particular moment in time, depending on the notion. As an example, let  $\phi_i(t)$  denote the earliest (absolute) activation of a job of task  $\tau_i$  at or after time  $t$ , i.e.

$$\phi_i(t) = \phi_i + \left( \left\lceil \frac{t - \phi_i}{T_i} \right\rceil \right)^+ \cdot T_i.$$

Given  $\phi_i(t)$ , the *relative phasing*  $\varphi_i(t)$  is given by  $\varphi_i(t) = \phi_i(t) - t$ . The release of job  $k$  of task  $\tau_i$  relative to  $t$  takes place at the *relative activation time*  $a_{ik}(t) = \varphi_i(t) + kT_i$ ,  $k \in \mathbb{N}$ . For  $a_{ik}(t)$ , both the identification of the job and the time are therefore relative to  $t$ . Similarly, the notions *relative begin time*  $b_{ik}(t)$  and *relative finalization time*  $f_{ik}(t)$  denote a time relative to  $t$  and concern the job  $k$  of task  $\tau_i$  relative to  $t$ . For the *relative response time*  $r_{ik}(t)$ , only the identification of the job is relative to  $t$ . We will use abbreviated representations for the relative notions using a prime (') when the particular moment in time is clear from the context. As an example, in a context concerning a particular moment  $t_s$ , the relative activation time  $a'_{ik}$  denotes  $a_{ik}(t_s)$ .

### 3 Recapitulation of worst-case analysis for FPPS

For the analysis under FPPS, we only consider cases where the deadlines of tasks are less than or equal to the respective periods. For illustration purposes, we will use a set  $\mathcal{T}_1$  of two independent periodic tasks  $\tau_1$  and  $\tau_2$  with characteristics as given in Table 1.

	$T_i = D_i$	$C_i$
$\tau_1$	5	2
$\tau_2$	7	3

Table 1. Task characteristics of  $\mathcal{T}_1$ .

Figure 3 shows an example of the execution of the tasks  $\tau_1$  and  $\tau_2$  under FPPS. Note that even an infinitesimal increase of the computation time of either task  $\tau_1$  or  $\tau_2$  will immediately cause the job of task  $\tau_2$  released at time 0 to miss its deadline at time 7.

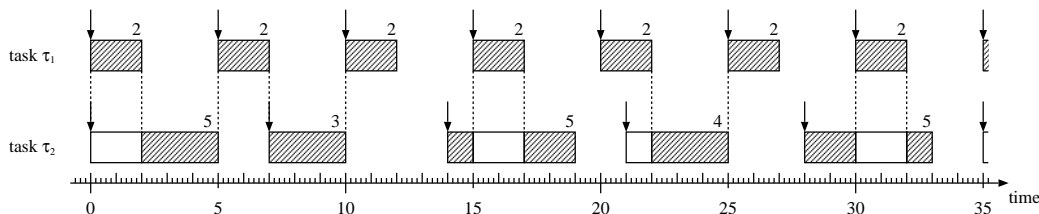


Figure 3. Timeline for  $\mathcal{T}_1$  under FPPS with a simultaneous release of both tasks at time zero. The numbers to the top right corner of the boxes denote the response times of the respective releases.

#### 3.1 Worst-case response times

This section presents theorems for the notion of critical instant and to determine worst-case response times of tasks. Although these theorems are taken from [5], most of these results were already known; see for example [1, 21, 27]. Auxiliary lemmas on which the proofs of these theorems and theorems in subsequent sections are based are included in Appendix A.

**Theorem 1 (Theorem 4.1 in [5]).** *In order to have a maximal response time for an execution  $k$  of task  $\tau_i$ , i.e. to have  $f_{ik} - a_{ik} = WR_i$ , we may assume without loss of generality that the phasing  $\varphi$  is such that  $\varphi_j = a_{ik}$  for all  $j < i$ . In other words, the phasing of the tasks' release times is such that the release of the considered execution of  $\tau_i$  coincides with the simultaneous release for all higher priority tasks. This latter point in time is called a critical instant for task  $\tau_i$ .  $\square$*

Given this theorem, we conclude that time 0 in Figure 3 is a critical instant for both task  $\tau_1$  and  $\tau_2$ . From this figure, we therefore derive that the worst-case response times of tasks  $\tau_1$  and  $\tau_2$  are 2 and 5, respectively. The next theorems can be used to determine the worst-case response times analytically.

**Theorem 2 (Theorem 4.2 in [5]).** *The worst-case response time  $WR_i$  of a task  $\tau_i$  is given by the smallest  $x \in \mathbb{R}^+$  that satisfies the following equation, provided that  $x$  is at most  $T_i$ .*

$$x = C_i + \sum_{j < i} \left\lceil \frac{x}{T_j} \right\rceil C_j \quad (7)$$

□

**Theorem 3 (Theorem 4.3 in [5]).** *The worst-case response time  $WR_i$  of task  $\tau_i$  can be found by the following iterative procedure.*

$$WR_i^{(0)} = C_i \quad (8)$$

$$WR_i^{(l+1)} = C_i + \sum_{j<i} \left\lceil \frac{WR_i^{(l)}}{T_j} \right\rceil C_j, \quad l = 0, 1, \dots \quad (9)$$

*The procedure is stopped when the same value is found for two successive iterations of  $l$ , or when the deadline  $D_i$  is exceeded.*

□

### 3.2 Worst-case occupied times

In Figure 3, task  $\tau_2$  is preempted at time 15 due to a release of task  $\tau_1$ , and resumes its execution at time 17. The span of time from a task  $\tau$ 's release till the moment in time that  $\tau$  can start or resume its execution after completion of a computation time  $C$  is termed *occupied time*. The *worst-case occupied time (WO)* of a task  $\tau$  is the longest possible span of time from a release of  $\tau$  till the moment in time that  $\tau$  can start or resume its execution after completion of a computation  $C$ . In [5], it has been shown that the worst-case occupied time can be described in terms of the worst-case response time as follows.

$$WO_i(C_i) = \lim_{x \downarrow C_i} WR_i(x). \quad (10)$$

Considering Figure 3, we derive that worst-case occupied times  $WO_2(0)$  and  $WO_2(C_2)$  of task  $\tau_2$  are equal to 2 and 7, respectively. The next theorems can be used to determine the worst-case occupied times analytically.

**Theorem 4 (Theorem 4.4 in [5]).** *When the smallest positive solution of (7) for a computation time  $C'_i$  is at most  $D_i$ , the worst-case occupied time  $WO_i$  of a task  $\tau_i$  with a computation time  $C_i \in [0, C'_i]$  is given by the smallest non-negative  $x \in \mathbb{R}$  that satisfies*

$$x = C_i + \sum_{j<i} \left( \left\lfloor \frac{x}{T_j} \right\rfloor + 1 \right) C_j. \quad (11)$$

□

**Theorem 5 (Theorem 4.5 in [5]).** *The worst-case occupied time  $WO_i$  of task  $\tau_i$  can be found by the following iterative procedure.*

$$WO_i^{(0)} = \begin{cases} \sum_{j<i} C_j & \text{for } C_i = 0 \\ WR_i & \text{for } C_i > 0 \end{cases} \quad (12)$$

$$WO_i^{(l+1)} = C_i + \sum_{j<i} \left( \left\lfloor \frac{WO_i^{(l)}}{T_j} \right\rfloor + 1 \right) C_j, \quad l = 0, 1, \dots \quad (13)$$

*The procedure is stopped when the same value is found for two successive iterations of  $l$ .*

□

### 3.3 Concluding remarks

The proof of Theorem 4 derives Equation (11) by starting from Equation (10) and subsequently using Lemma 16.

Similarly to Equation 10, we can express  $WR_i$  in terms of  $WO_i$ , i.e.

$$WR_i(C_i) = \lim_{x \uparrow C_i} WO_i(x). \quad (14)$$

The next two equations express that  $WR_i(C_i)$  and  $WO_i(C_i)$  are left-continuous and right-continuous, respectively.

$$WR_i(C_i) = \lim_{x \uparrow C_i} WR_i(x) \quad (15)$$

$$WO_i(C_i) = \lim_{x \downarrow C_i} WO_i(x) \quad (16)$$

Lemmas related to these latter three equations can be found in Appendix A.

## 4 Existing response time analysis for FPDS refuted

In this section, we first recapitulate existing response time analysis under FPDS. Next, we show that the existing analysis is pessimistic. We subsequently give examples refuting the analysis, i.e. examples that show that the existing analysis is optimistic.

### 4.1 Recapitulation of existing worst-case response time analysis for FPDS

In this section, we recapitulate existing worst-case response time analysis for FPDS with arbitrary phasing and deadlines within periods as described in [11, 13]. We include a recapitulation of the analysis for FPNS as presented in [37]. The main reason for including the latter is that it looks different from the analysis for FPDS and is a basis for the analysis of controller area network (CAN).

#### 4.1.1 Existing analysis for FPDS

The non-preemptive nature of subjobs may cause blocking of a task by at most one lower priority task under FPDS. Moreover, a task can be blocked by at most one subjob of a lower priority task. The maximum blocking  $B_i^D$  of task  $\tau_i$  by a lower priority task is therefore equal to the longest computation time of any subjob of a task with a priority lower than task  $\tau_i$ . This blocking time is given by

$$B_i^D = \max_{j>i} \max_{1 \leq k \leq m_j} C_{j,k}. \quad (17)$$

Strictly spoken,  $B_i^D$  is a supremum for all but the lowest priority task, i.e. that value cannot be assumed.

The worst-case response time  $\widetilde{WR}_i^D$  of a task  $\tau_i$  under FPDS, arbitrary phasing, and deadlines less than or equal to periods, as presented in [11] and [13], is given by

$$\widetilde{WR}_i^D = WR_i^P(B_i^D + C_i - (F_i - \Delta)) + (F_i - \Delta), \quad (18)$$

where  $WR_i^P$  denotes the worst-case response time of  $\tau_i$  under FPNS. According to [13],  $\Delta$  is an arbitrary small positive value needed to ensure that the final subjob has actually started. Hence, when task  $\tau_i$  has consumed  $C_i - (F_i - \Delta)$ , the final subjob has (just) started.

#### 4.1.2 Existing analysis for FPNS

In this section, we first recapitulate the update of [21] given in [37] to take account of tasks being non-preemptive. Next, we show that the update is actually a specialization of (18).

The non-preemptive nature of tasks may cause blocking of a task by at most one lower priority task. The maximum blocking  $B_i^N$  of task  $\tau_i$  by a lower priority task is equal to the longest computation time of a task with a priority lower than task  $\tau_i$ , i.e.

$$B_i^N = \max_{j>i} C_j. \quad (19)$$

Similarly to  $B_i^D$ ,  $B_i^N$  is a supremum for all but the lowest priority task, i.e. that value cannot be assumed.

The worst-case response time  $\widetilde{WR}_i^N$  is given by

$$\widetilde{WR}_i^N = w_i + C_i, \quad (20)$$

where  $w_i$  is the smallest  $x \in \mathbb{R}^+$  that satisfies

$$x = B_i^N + \sum_{j<i} \left\lceil \frac{x + \tau_{res}}{T_j} \right\rceil C_j. \quad (21)$$

In this latter equation,  $\tau_{res}$  is the resolution with which time is measured. To calculate  $w_i$ , an iterative procedure based on recurrence relationships can be used. An appropriate initial value of this procedure is  $w_i^{(0)} = B_i^N + \sum_{j<i} C_j$ .

We now show that these results for FPNS are a specialization of (18). To this end, we substitute  $w_i = w'_i - \tau_{res}$ ,  $x = x' - \tau_{res}$ , and  $\tau_{res} = \Delta$  in equations (20) and (21). Hence, the worst-case response time  $\widetilde{WR}_i^N$  is given by

$$\widetilde{WR}_i^N = w'_i + (C_i - \Delta),$$



where  $w'_i$  is the smallest  $x' \in \mathbb{R}^+$  that satisfies

$$x' = B_i^N + \Delta + \sum_{j < i} \left\lceil \frac{x'}{T_j} \right\rceil C_j.$$

Reusing the results for FPPS, we therefore get

$$\widetilde{WR}_i^N = WR_i^P(B_i^N + \Delta) + (C_i - \Delta). \quad (22)$$

Because we have  $F_i = C_i$  and  $B_i^D = B_i^N$  for FPNS, Equation (22) for FPDS is a specialization of Equation (18) for FPNS.

## 4.2 Existing analysis is pessimistic

Consider the set  $\mathcal{T}_2$  consisting of three tasks with characteristics as described in Table 2.

	$T_i$	$D_i$	$C_i$
$\tau_1$	5	4	2
$\tau_2$	7	7	1 + 2
$\tau_3$	30	30	2 + 2

Table 2. Task characteristics of  $\mathcal{T}_2$ .

Based on (18) and using  $\Delta = 0.2$  we derive

$$\begin{aligned} \widetilde{WR}_2^D &= WR_2^P(B_2^D + C_2 - (F_2 - \Delta)) + (F_2 - \Delta) \\ &= WR_2^P(2 + 3 - 1.8) + 1.8 \\ &= WR_2^P(3.2) + 1.8 = 7.2 + 1.8 = 9. \end{aligned}$$

However, the existing analysis does not take into account that  $\tau_i$  can only be blocked by a subjob of a lower priority task if that subjob starts an amount of time  $\Delta$  before the simultaneous release of  $\tau_i$  and all tasks with a higher priority than  $\tau_i$ . This aspect can be taken into account in the analysis by replacing  $B_i^D$  in (18) by  $(B_i^D - \Delta)^+$ . Here, the notation  $w^+$  stands for  $\max(w, 0)$ , which is used to indicate that the blocking time can not become negative for the lowest priority task. The worst-case response time of  $\tau_2$  now becomes 6.8, as illustrated in Figure 4. As a result, the existing analysis is pessimistic.

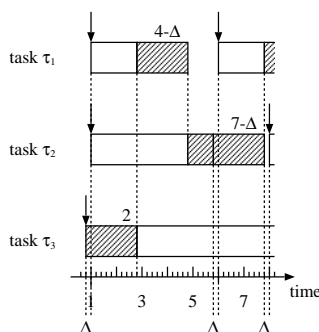


Figure 4. Timeline for  $\mathcal{T}_2$  under FPDS with a release of tasks  $\tau_1$  and  $\tau_2$  at time  $t = 1$  and a release of task  $\tau_3$  at time  $t = 1 - \Delta$ .

We observe that the characteristics of the tasks of  $\mathcal{T}_2$  are integral multiples of a value  $\delta = 1$ , and  $\Delta \leq \delta$ . As a consequence, reducing  $\Delta$  to an arbitrary small positive value does not change the value for  $\widetilde{WR}_2^D$ . When  $\Delta$  is larger than  $\delta$ , reducing  $\Delta$  can change the value of the derived worst-case response time. As an example, consider set  $\mathcal{T}_3$  consisting of three tasks with characteristics as described in Table 3. For this example, the task characteristics are integral multiples of  $\delta = 0.5$ . We will now determine the worst-case response time  $\widetilde{WR}_2^D$  of  $\tau_2$  for two values of  $\Delta$ , one value larger than  $\delta$  and another value smaller than  $\delta$ .

Let  $\Delta = 0.6 > \delta$ , i.e. the computation times  $C_1$ ,  $C_2$ , and  $C_3$  are integral multiples of  $\Delta$ , but the period  $T_1$  is not. For the worst-case response time  $\widetilde{WR}_2^D$  we find

$$\begin{aligned} \widetilde{WR}_2^D &= WR_2^P(B_2^D + C_2 - (F_2 - \Delta)) + (F_2 - \Delta) \\ &= WR_2^P(3 + 3 - (3 - 0.6)) + (3 - 0.6) \\ &= WR_2^P(3.6) + 2.4 = 9.6 + 2.4 = 12. \end{aligned}$$

	$T_i = D_i$	$C_i$
$\tau_1$	6.5	3
$\tau_2$	9	3
$\tau_3$	30	3

Table 3. Task characteristics of  $\mathcal{T}_3$ .

For this value of  $\Delta$ ,  $\widetilde{WR}_2^D$  is larger than  $\tau_2$ 's deadline.

Let  $\Delta = 0.4$ , i.e.  $\Delta < \delta$ . For the worst-case response time  $\widetilde{WR}_2^D$  of  $\tau_2$  we find

$$\begin{aligned} \widetilde{WR}_2^D &= WR_2^P(B_2^D + C_2 - (F_2 - \Delta)) + (F_2 - \Delta) \\ &= WR_2^P(3 + 3 - (3 - 0.4)) + (3 - 0.4) \\ &= WR_2^P(3.4) + 2.6 = 6.4 + 2.6 = 9. \end{aligned}$$

For this value of  $\Delta$ ,  $\widetilde{WR}_2^D \leq D_2$ , and reducing the value of  $\Delta$  will not change the value found for  $\widetilde{WR}_2^D$ .

If we take into account that the blocking subjob of a lower priority task has to start an amount of time  $\Delta$  before the simultaneous release of  $\tau_i$  and all higher priority tasks, we find values 8.4 and 8.6 for  $\Delta = 0.6$  and  $\Delta = 0.4$ , respectively. We will return to this example in Section 8.3.2.

### 4.3 Existing analysis is optimistic

We will give three examples illustrating that the existing analysis is optimistic. For all three examples, deadlines are equal to periods, i.e.  $D_i = T_i$ . The first section shows an obvious example, i.e. an example with a utilization factor  $U > 1$ . The second section shows an example with  $U < 1$ . The third section shows an example with  $U = 1$ .

For all three examples, the task set consists of just two tasks. For such task sets, the worst-case response time analysis under FPDS presented in [11, 12, 13] and in [10] is very similar. In particular, the worst-case response time  $\widetilde{WR}_2^D$  of task  $\tau_2$  is determined by looking at the response time of the first job of task  $\tau_2$  upon a simultaneous release with task  $\tau_1$ . However, the worst-case response time of task  $\tau_2$  is not assumed for the first job for all three examples.

#### 4.3.1 An example with $U > 1$

An example refuting the worst-case response time analysis is given in Table 4. Note that the utilization factor  $U$  of this set of tasks  $\mathcal{T}_4$  is given by  $U = \frac{2}{5} + \frac{4.5}{7} > 1$ . Hence, the task set is not schedulable.

	$T_i = D_i$	$C_i$
$\tau_1$	5	2
$\tau_2$	7	1.5 + 3

Table 4. Task characteristics of  $\mathcal{T}_4$ .

Based on (18) and using  $\Delta = 0.1$ , we derive

$$\begin{aligned} \widetilde{WR}_2^D &= WR_2^P(B_2 + C_2 - (F_2 - \Delta)) + (F_2 - \Delta) \\ &= WR_2^P(0 + 4.5 - 2.9) + 2.9 \\ &= WR_2^P(1.6) + 2.9 = 3.6 + 2.9 = 6.5. \end{aligned}$$

This value corresponds with the response time of the 1<sup>st</sup> job of task  $\tau_2$  upon a simultaneous release with task  $\tau_1$ , as illustrated in Figure 5. However, the same figure also illustrates that the second job of  $\tau_2$  misses its deadline. Stated in other words, the existing worst-case response time analysis is optimistic.

#### 4.3.2 An example with $U < 1$

Another example refuting the worst-case response time analysis is given in Table 5. Note that the utilization factor  $U$  of this set of tasks  $\mathcal{T}_5$  is given by  $U = \frac{2}{5} + \frac{4.1}{7} < 1$ . Hence, the task set could be schedulable. Applying (18) yields  $\widetilde{WR}_2^D = 6.1$ , which corresponds with the response time of the first job of task  $\tau_2$  upon a simultaneous release with task  $\tau_1$ ; see Figure 6. However, the same figure also illustrates that the second job of task  $\tau_2$  misses its deadline.

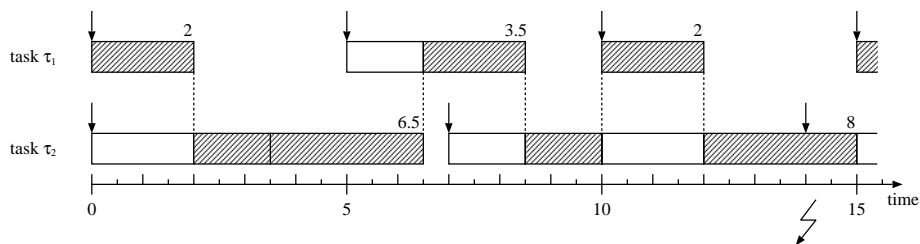


Figure 5. Timeline for  $\mathcal{T}_4$  under FPDS with a simultaneous release of both tasks at time zero.

	$T_i = D_i$	$C_i$
$\tau_1$	5	2
$\tau_2$	7	2 + 2.1

Table 5. Task characteristics of  $\mathcal{T}_5$ .

### 4.3.3 An example with $U = 1$

Consider task set  $\mathcal{T}_6$  given in Table 6. The utilization factor  $U$  of this set of tasks is given by  $U = \frac{2}{5} + \frac{4.2}{7} = 1$ . The task set is not schedulable by FPDS, as we showed in Section 3 that the task set is only schedulable when  $C_2$  is at most 3. Figure 7 shows a timeline with the executions of these two tasks under FPDS with a simultaneous release at time zero in an interval of length 35, i.e. equal to the *hyperperiod* of the tasks. Applying (18) yields  $\widetilde{WR}_2^D = 6.2$ , which corresponds with the response time of the first job of task  $\tau_2$  in Figure 7. However, the response time of the 5<sup>th</sup> job of task  $\tau_2$  is equal to 7, illustrating once again that the existing analysis is too optimistic. Nevertheless, the task set is schedulable under FPDS for this phasing.

Now, consider task set  $\mathcal{T}_7$  given in Table 7, which is similar to task set  $\mathcal{T}_6$  given in Table 6, except for the fact that rather than having a second subjob for task  $\tau_2$  it has a task  $\tau_3$ . Figure 8 shows a timeline with the executions of these three tasks under FPNS with a simultaneous release at time zero in an interval of length 35, i.e. equal to the *hyperperiod* of the tasks. Applying (18) yields  $\widetilde{WR}_3^D = 6.2$ , which corresponds to the response time of the first job of task  $\tau_3$  in Figure 8. However, the response time of the 5<sup>th</sup> job of task  $\tau_3$  is equal to 7, illustrating once again that the existing analysis is too optimistic. Nevertheless, the task set is schedulable under FPNS for this phasing.

## 4.4 Concluding remark

We have shown that we cannot restrict ourselves to the response time of the first job of a task when determining the worst-case response time of that task under FPDS. The reason for this is that the final subjob of a task  $\tau_i$  can defer the execution of higher priority tasks, which can potentially give rise to higher interference for subsequent jobs of task  $\tau_i$ . We observe that González Harbour et al [17] identified the same influence of jobs of a task for relative deadlines at most equal to periods in the context of fixed priority scheduling of periodic tasks with varying execution priority.

Considering Figure 7, we see that every job of task  $\tau_2$  in the interval  $[0, 26.8)$  defers the execution of a job of task  $\tau_1$ . Moreover, that deferred job of task  $\tau_1$  subsequently gives rise to additional interference for the next job of task  $\tau_2$ . This situation ends when the job of  $\tau_2$  is started at time  $t = 28$ , i.e. the 5<sup>th</sup> job of  $\tau_2$  does not defer the execution of a job of  $\tau_1$ . Viewed in a different way, we may state that the active intervals of the jobs of tasks  $\tau_1$  and  $\tau_2$  overlap in the interval  $[0, 35)$ . Note that this overlapping starts at time  $t = 0$  and ends at time  $t = 35$ , and we therefore term this interval  $[0, 35)$  a *level-2 active period*. Informally, a *level- $i$  active period* is a *smallest* interval that only contains entire active intervals of jobs of task

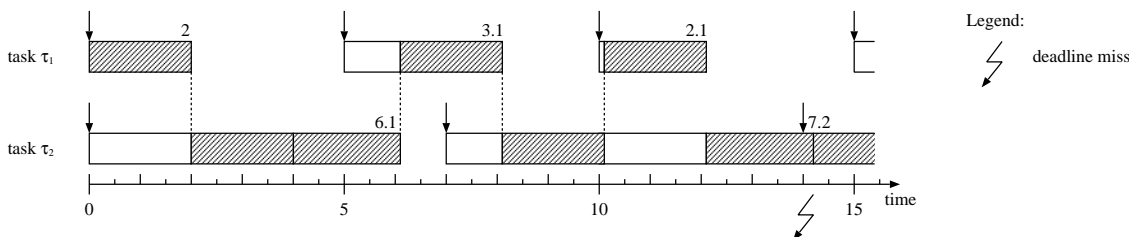


Figure 6. Timeline for  $\mathcal{T}_5$  under FPDS with a simultaneous release of all tasks at time zero.

	$T_i = D_i$	$C_i$
$\tau_1$	5	2
$\tau_2$	7	1.2 + 3

Table 6. Task characteristics of  $\mathcal{T}_6$ .

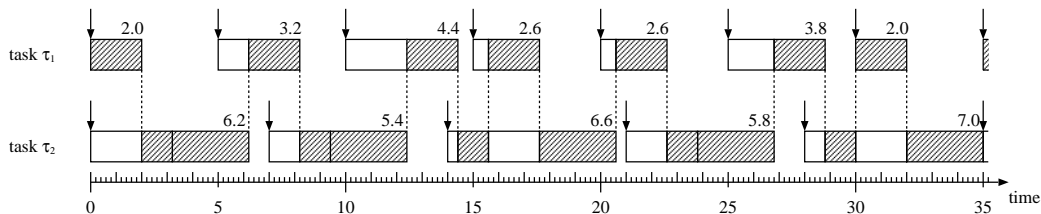


Figure 7. Timeline for  $\mathcal{T}_6$  under FPDS with a simultaneous release of all tasks at time zero.

$\tau_i$  and jobs of tasks with a higher priority than task  $\tau_i$ . Hence, the active interval of every job of a task  $\tau_i$  is contained in a level- $i$  active period. To determine the worst-case response time of a task  $\tau_i$ , we therefore only have to consider level- $i$  active periods. However, as illustrated by the examples shown in this section and mentioned above, we cannot restrict ourselves to the response time of the first job of a task  $\tau_i$  when determining the worst-case response time of that task under FPDS. Instead, we have to consider the response times of *all* jobs in a level- $i$  active period. In a subsequent section, we will show that it suffices to consider only the response times of jobs in a level- $i$  active period that starts at a so-called  $\epsilon$ -critical instant.

## 5 Active period

This section presents a formal definition of a *level- $i$  active period* and theorems to determine the length of a level- $i$  active period. As mentioned above, a level- $i$  active period may contain multiple jobs of  $\tau_i$ . We therefore also define the notion of a *level- $(i, k)$  active period*, and present a theorem to determine the length of such a period. Informally, level- $(i, k)$  active period is a *smallest* interval that contains  $k$  successive active intervals of jobs of task  $\tau_i$  and all jobs of tasks with a higher priority than task  $\tau_i$ .

We start with the definition of the notion level- $i$  active period in Section 5.1. Next, we provide examples of level- $i$  active periods in Section 5.2. The length of a level- $i$  active period is the topic of Section 5.3. We refine the notion of level- $i$  active period to level- $(i, k)$  active period in Section 5.4, and conclude with a theorem to determine its length in Section 5.5.

### 5.1 Level- $i$ active period

The notion of level- $i$  active period is defined in terms of the notion of *pending load*, which on its turn is defined in terms of the notion of *active job*.

#### 5.1.1 Active job and pending load

**Definition 1.** A job  $k$  of a task  $\tau_i$  is active at time  $t$  if and only if  $t \in [a_{ik}, f_{ik})$ , where  $a_{ik}$  and  $f_{ik}$  are the activation (or release) time and the finalization (or completion) time of that job, respectively.  $\square$

The *active interval* of job  $k$  of task  $\tau_i$  is defined as the time span between the activation time of that job and its completion, i.e.  $[a_{ik}, f_{ik})$ . We now define the notion of pending load in terms of active job, and derive properties for the pending load.

**Definition 2.** The pending load  $P_i^\tau(t)$  is the amount of processing at time  $t$  that still needs to be performed for the active jobs of tasks  $\tau_i$  that are released before time  $t$ , i.e.

$$P_i^\tau(t) = \left( \left\lceil \frac{t - \phi_i}{T_i} \right\rceil \right)^+ \cdot C_i - \int_0^t \sigma_i^\tau(t') dt', \quad (23)$$

	$T_i$	$C_i$
$\tau_1$	5	2
$\tau_2$	7	1.2
$\tau_3$	7	3

Table 7. Task characteristics of  $\mathcal{T}_7$ .

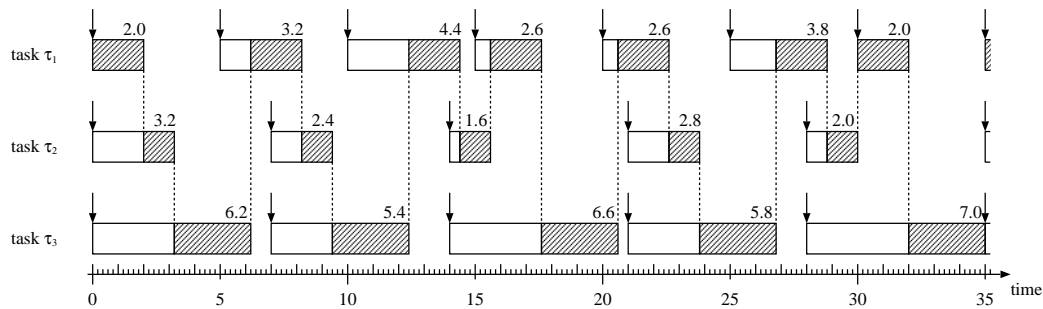


Figure 8. Timeline for  $\mathcal{T}_7$  under FPNS with a simultaneous release of all tasks at time zero. The numbers to the top right corner of the boxes denote the response times of the respective releases.

where

$$\sigma_i^\tau(t) = \begin{cases} 1 & \text{if task } \tau_i \text{ is being executed at time } t, \text{ i.e. } \sigma(t) = i \\ 0 & \text{otherwise.} \end{cases}$$

□

Note that the term  $\left(\left\lceil \frac{t - \Phi_i}{T_i} \right\rceil\right)^+$  in (23) is equal to the number of releases of  $\tau_i$  in  $[0, t)$ . By multiplying this term with  $C_i$ , we therefore get the amount of processing that needs to be performed due to releases of task  $\tau_i$  in that interval. The term  $\int_0^t \sigma_i^\tau(t') dt'$  is equal to the amount of processing that has been performed for  $\tau_i$ . The righthand side of (23) is therefore equal to the amount of processing at time  $t$  due to releases of jobs of task  $\tau_i$  before  $t$  that still needs to be performed.

We subsequently define the notions of (cumulative) pending load  $P_i(t)$  and (processor) pending load  $P(t)$ .

**Definition 3.** The (cumulative) pending load  $P_i(t)$  is the amount of processing at time  $t$  that still needs to be performed for the active jobs of tasks  $\tau_j$  with  $j \leq i$  that are released before time  $t$ , i.e.

$$P_i(t) = \sum_{j \leq i} P_j^\tau(t) = \sum_{j \leq i} \left( \left\lceil \frac{t - \Phi_j}{T_j} \right\rceil \right)^+ \cdot C_j - \int_0^t \sigma_j^\tau(t') dt', \quad (24)$$

where

$$\sigma_i(t) = \sum_{j \leq i} \sigma_j^\tau(t) = \begin{cases} 1 & \text{if } \sigma(t) \in \{1, \dots, i\} \\ 0 & \text{otherwise.} \end{cases}$$

□

**Definition 4.** The (processor) pending load  $P(t)$  is the amount of processing at time  $t$  that still needs to be performed for the active jobs of all tasks that are released before time  $t$ , i.e.

$$P(t) = P_n(t). \quad (25)$$

□

**Corollary 1.** The order in which the tasks  $\tau_j$  with  $j \leq i$  are executed is immaterial for the cumulative pending load  $P_i$ . □

For  $i < n$ , the cumulative pending load  $P_i$  also depends on blocking due to a lower priority task. As an example, let  $P_i(t_s) = 0$ , than  $P_i(t) = C_s$  for all  $t \in (t_s, t'_s)$  under FPDS if the following three conditions hold:

- a task  $\tau_s$  with  $s \leq i$  is released at time  $t_a$ ,
- no other releases of  $\tau_j$  for  $j \leq i$  takes place in  $[t_s, t'_s)$ , and
- a subjob of a lower priority task is executing at time  $t_s$  and blocks task  $\tau_s$  during  $[t_s, t'_s)$  due to the non-preemptive nature of the subjob.

Because blocking due to a lower priority task does not play a role for the (processor) pending load,  $P(t)$  only depends on the activations of tasks.

**Corollary 2.** The (processor) pending load  $P(t)$  is independent of the scheduling algorithm. □

### 5.1.2 Definition of a level- $i$ active period

We now define the notion of level- $i$  active period in terms of the pending load  $P_i(t)$ .

**Definition 5.** A level- $i$  active period is an interval  $[t_s, t_e)$  with the following three properties.

1.  $P_i(t_s) = 0$ ;
2.  $P_i(t_e) = 0$ ;
3.  $P_i(t) > 0$  for  $t \in (t_s, t_e)$ .

□

**Lemma 1.** If a level- $i$  active period starts at time  $t_s$  and ends at time  $t_e$ , then the following properties hold:

- (i) Tasks  $\tau_j$  with  $j \leq i$  are continuously executing in  $[t_s, t_e)$ , except for an (optionally empty) initial interval  $[t_s, t_s + B)$  with  $0 \leq B < B_i^D$  during which the tasks are blocked by a lower priority task.
- (ii) The length  $L_i(t_s)$  of that level- $i$  active period is at least  $B + C_s$ , where task  $\tau_s$  is released at time  $t_s$ .
- (iii) The order in which the tasks  $\tau_j$  with  $j \leq i$  are executed is immaterial.

*Proof.* (i) This property follows immediately from the non-preemptive nature of subjobs and the assumptions for fixed-priority scheduling.

(ii) By definition,  $P_i(t_s) = 0$ . Because the tasks  $\tau_j$  with  $j \leq i$  are blocked in the (optionally empty) initial interval  $[t_s, t_s + B)$ , and the level- $i$  active period contains at least the active interval of task  $\tau_s$ , the length  $L_i(t_s)$  of that level- $i$  active period is at least  $B + C_s$ .

(iii) This property follows immediately from the definition of a level- $i$  active period and Corollary 1. □

From this definition of the level- $i$  active period in terms of the pending load  $P_i(t)$ , we draw the following conclusion.

**Corollary 3.** The level- $n$  active period is independent of the scheduling algorithm. □

Note that a level- $i$  active period may, but need not, contain activations of task  $\tau_i$ . In the sequel, we will call a level- $i$  active period that contains an activation of task  $\tau_i$  a *true* level- $i$  active period. Unless explicitly stated otherwise, we use the phrase ‘level- $i$  active period’ to denote a true level- $i$  active period in the remainder of this document.

## 5.2 Examples

We will now consider two examples, one for FPPS based on the timeline in Figure 3 for  $\mathcal{T}_1$  and one for FPDS based on the timeline in Figure 7 for  $\mathcal{T}_6$ .

Consider Figure 9, with a timeline for  $\mathcal{T}_1$  under FPPS, pending loads  $P_1(t)$ ,  $P_2^r(t)$ , and  $P_2(t)$ , and level- $i$  active periods. Note that  $P_1(t)$  is equal to  $P_1^r(t)$  by definition. From the graph for  $P_1(t)$ , we find that the interval  $[0, 35)$  contains seven level-1 active periods, corresponding with the seven activations of task  $\tau_1$ , i.e.  $[0, 5)$ ,  $[5, 7)$ ,  $[10, 12)$ ,  $[15, 17)$ ,  $[20, 22)$ ,  $[25, 27)$ , and  $[30, 32)$ . The horizontal line fragments in the graph for  $P_2^r(t)$  are caused by the fact that  $\tau_2$  is preempted by a job of task  $\tau_1$ . From the graph for the pending load  $P_2(t)$ , we find that the interval  $[0, 35)$  contains eight level-2 active periods, i.e.  $[0, 5)$ ,  $[5, 7)$ ,  $[7, 10)$ ,  $[10, 12)$ ,  $[14, 19)$ ,  $[20, 25)$ ,  $[25, 27)$ , and  $[28, 33)$ . As mentioned before, the level-2 active period only depends on the activations of  $\tau_1$  and  $\tau_2$ , and is independent of the scheduling algorithm.

Consider Figure 10, with a timeline for  $\mathcal{T}_6$  under FPDS, pending loads  $P_1(t)$ ,  $P_2^r(t)$ , and  $P_2(t)$ , and level- $i$  active periods. From the graph for  $P_1(t)$ , we find that the interval  $[0, 35)$  contains seven level-1 active periods, corresponding with the seven activations of task  $\tau_1$ , i.e.  $[0, 2)$ ,  $[5, 8.2)$ ,  $[10, 14.4)$ ,  $[15, 17.6)$ ,  $[20, 22.6)$ ,  $[25, 28.8)$ , and  $[30, 32)$ . The horizontal line fragments in the graph for  $P_1(t)$  are caused by the fact that  $\tau_1$  is blocked by a subjob of task  $\tau_2$ . From the graph for the pending load  $P_2(t)$ , we find that the interval  $[0, 35)$  contains a single level-2 active period, i.e.  $[0, 35)$ .

## 5.3 Length of a level- $i$ active period

This section presents three theorems for the length of a level- $i$  active period. A first theorem presents a recursive equation for the length of a level- $i$  active period. A next theorem states that under the following assumption a level- $i$  active period that starts will also end.

**Assumption 1.** Either  $U < 1$  or  $U \leq 1$  and the least common multiple (lcm) of the periods of the tasks of  $\mathcal{T}$  exists. □

Hence, the assumption is a sufficient condition to guarantee that a level- $i$  active period will end when its starts. Because we assume  $\phi_i \geq 0$  for all  $i \leq n$ ,  $P_i(0) = 0$  for all  $i \leq n$ . We therefore conclude that, when Assumption 1 holds, the timeline consists

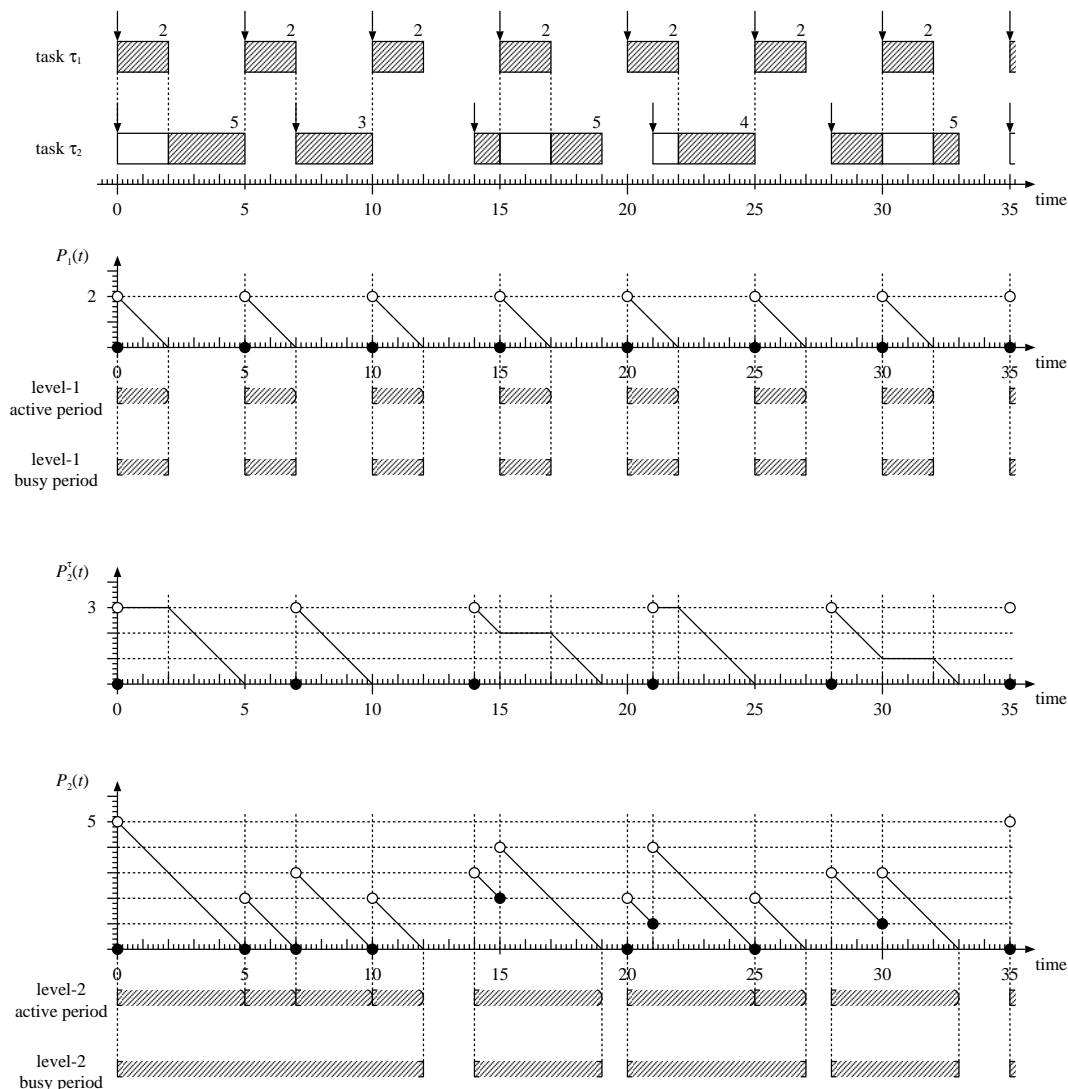


Figure 9. Timeline for  $\tau_1$  under FPPS, pending loads  $P_1(t)$ ,  $P_2^1(t)$ , and  $P_2(t)$ , and level- $i$  active periods and level- $i$  busy periods.

of a sequence of level- $i$  active periods, optionally preceded by and separated by idle-periods. A final theorem provides an iterative procedure to determine the length of a level- $i$  active period.

Appendix B shows an example illustrating that the level- $n$  active period need not end when Assumption 1 does not hold.

### 5.3.1 A recursive equation

**Theorem 6.** The length  $L_i(t_s)$  of a level- $i$ - active period that starts at time  $t_s$  is found for the smallest  $x \in \mathbb{R}^+$  that satisfies the following equation

$$x = B + \sum_{j \leq i} \left( \left\lceil \frac{x - \phi_j(t_s)}{T_j} \right\rceil \right)^+ \cdot C_j, \tag{26}$$

where  $B$  denotes the amount of time that a task with a lower priority than task  $\tau_i$  is executing non-preemptively as from time  $t_s$ , with  $0 \leq B < B_i^D$  for  $i < n$  and  $B = 0$  for  $i = n$ .

*Proof.* Because the level- $i$  active period starts at time  $t_s$ ,  $P_i(t_s) = 0$  by definition. Now assume the level- $i$  active period under consideration ends at time  $t_e$ . Hence, time  $t_e$  is the *smallest*  $t$  larger than  $t_s$  for which  $P(t) = 0$ , and the length  $L_i(t_s)$  of the

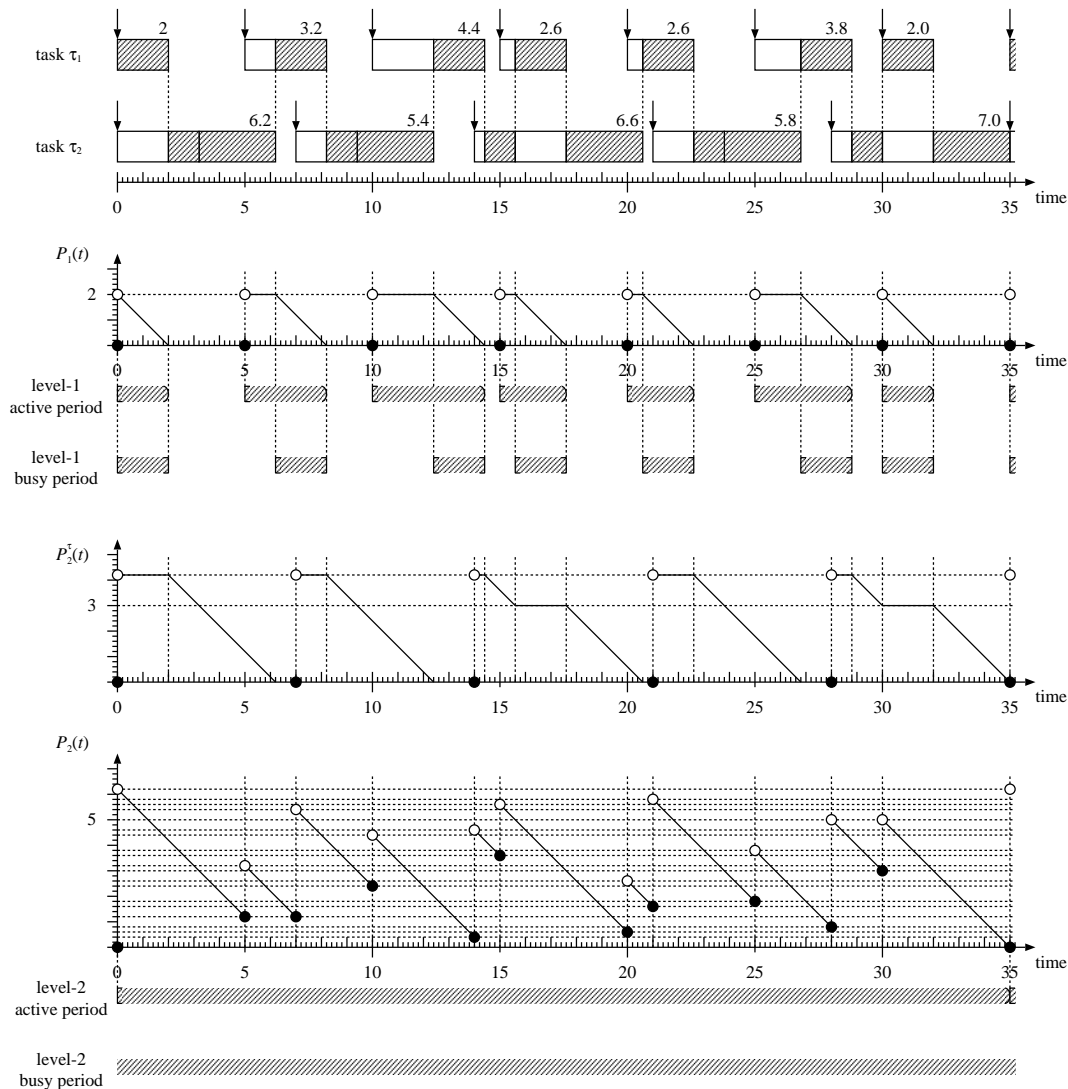


Figure 10. Timeline for  $\mathcal{T}_6$  under FPDS, pending loads  $P_1(t)$ ,  $P_2^T(t)$ , and  $P_2(t)$ , and level- $i$  active periods and level- $i$  busy periods.

active period becomes  $t_e - t_s$ . We now derive (26) from  $P(t_e) = 0$ .

$$\begin{aligned}
 P_i(t_e) &= \{(24)\} \sum_{j \leq i} \left( \left\lceil \frac{t_e - \phi_j}{T_j} \right\rceil \right)^+ \cdot C_j - \int_0^{t_e} \sigma_i(t) dt \\
 &= P_i(t_s) + \sum_{j \leq i} \left( \left\lceil \frac{t_e - (t_s + \phi_j(t_s))}{T_j} \right\rceil \right)^+ \cdot C_j - \int_{t_s}^{t_e} \sigma_i(t) dt \\
 &= \{P_i(t_s) = 0\} \sum_{j \leq i} \left( \left\lceil \frac{t_e - (t_s + \phi_j(t_s))}{T_j} \right\rceil \right)^+ \cdot C_j - \int_{t_s}^{t_e} \sigma_i(t) dt \\
 &= \sum_{j \leq i} \left( \left\lceil \frac{t_e - (t_s + \phi_j(t_s))}{T_j} \right\rceil \right)^+ \cdot C_j - \int_{t_s}^{t_e} \sigma_i(t) dt = 0
 \end{aligned}$$

From Lemma 1, we derive that the lower priority task is executing in  $[t_s, t_s + B)$ , and only tasks  $\tau_j$  with  $j \leq i$  are executing in



$[t_s + B, t_e)$ . Hence, we can conclude that

$$\int_{t_s}^{t_e} \sigma_i(t) dt = t_e - (t_s + B).$$

Substituting this result in the former equation, we get

$$t_e - (t_s + B) = \sum_{j \leq i} \left( \left\lceil \frac{t_e - (t_s + \varphi_j(t_s))}{T_j} \right\rceil \right)^+ \cdot C_j,$$

and by subsequently substituting  $t_e = x + t_s$ , we get (26). Because time  $t_e$  is the smallest  $t$  (larger than  $t_s$ ) for which  $P_i(t) = 0$ ,  $x = t_e - t_s$  is the smallest value in  $\mathbb{R}^+$  that satisfies (26), which proves the theorem.  $\square$

### 5.3.2 End of a level- $i$ active period

We now present a theorem which states that there exist positive solutions for the recursive equation (26) if Assumption 1 holds. To that end, we will use Lemma 4.3 from [5] (see Lemma 15 in Appendix A), and first prove two lemmas.

**Lemma 2.** *There exists a positive solution for the recursive equation (26) for the length of the level- $i$  active period if  $U_i < 1$ .*

*Proof.* We will prove that the condition  $U_i < 1$  is sufficient by means of Lemma 4.3 of [5]. Let  $f$  be defined as

$$f(x) = B + \sum_{j \leq i} \left( \left\lceil \frac{x - \varphi_j(t_s)}{T_j} \right\rceil \right)^+ \cdot C_j.$$

We choose  $a = \min_{l \leq i} \frac{C_l}{2}$ , hence

$$f(a) = f\left(\min_{l \leq i} \frac{C_l}{2}\right) = B + \sum_{j \leq i} \left( \left\lceil \frac{\min_{l \leq i} \frac{C_l}{2} - \varphi_j(t_s)}{T_j} \right\rceil \right)^+ \cdot C_j.$$

By definition, there is at least one task that is released at the start of the level- $i$  active period. Let task  $\tau_k$  with  $k \leq i$  be released at time  $t = t_s$ , i.e.  $\varphi_k(t_s) = 0$ . We now get

$$f(a) \geq B + \left\lceil \frac{\min_{l \leq i} \frac{C_l}{2}}{T_k} \right\rceil C_k = B + C_k > \min_{l \leq i} \frac{C_l}{2} = a,$$

hence  $f(a) > a$ . In order to choose an appropriate  $b$ , we make the following derivation.

$$f(x) \leq B + \sum_{j \leq i} \left\lceil \frac{x}{T_j} \right\rceil C_j < B + \sum_{j \leq i} \left( \frac{x}{T_j} + 1 \right) C_j = B + xU_i + \sum_{j \leq i} C_j.$$

As  $U_i < 1$ , the relation

$$x \geq B + xU_i + \sum_{j \leq i} C_j$$

holds for

$$x \geq \frac{B + \sum_{j \leq i} C_j}{1 - U_i}.$$

We now choose

$$b = \frac{B + \sum_{j \leq i} C_j}{1 - U_i},$$

and therefore get  $b > f(b)$ . Now the conditions for Lemma 15 hold, i.e. the function  $f(x)$  is defined and strictly non-decreasing in an interval  $[a, b]$  with  $f(a) > a$  and  $f(b) < b$ . Hence, there exists an

$$x \in \left( \min_{l \leq i} \frac{C_l}{2}, \frac{B + \sum_{j \leq i} C_j}{1 - U_i} \right)$$

such that  $x = f(x)$ .  $\square$

**Lemma 3.** *There exists a positive solution for the recursive equation (26) for the length of the level- $n$  active period if  $U = 1$  and the least common multiple of the periods of  $\mathcal{T}$  exists.*

*Proof.* We first observe that  $B = 0$  for the level- $n$  active period, i.e. the lowest priority task is never blocked. Next, we distinguish two complementary cases, a first case with  $\varphi_i(t_s) = 0$  for all  $i$  and a second case where this does not hold. We prove the lemma by considering both cases separately.

For the first case, we prove that for  $B = 0$  and  $\varphi_i(t_s) = 0$  for all  $i$  the value  $x = \text{lcm}(T_1, \dots, T_n)$  is a solution of (26). For these values of  $B$  and  $\varphi_i(t_s)$ , equation (26) simplifies to

$$x = \sum_{j \leq n} \left\lceil \frac{x}{T_j} \right\rceil C_j.$$

Because  $\left\lceil \frac{\text{lcm}(T_1, \dots, T_n)}{T_j} \right\rceil C_j = \text{lcm}(T_1, \dots, T_n) \frac{C_j}{T_j}$  and  $\sum_{j \leq n} \frac{C_j}{T_j} = U = 1$ , we immediately see that  $\text{lcm}(T_1, \dots, T_n)$  is a (positive) solution.

For the second case, we prove that the condition  $U = 1$  and lcm of the periods of  $\mathcal{T}$  exists is sufficient by means of Lemma 15. Let  $f$  be defined as

$$f(x) = \sum_{j \leq n} \left( \left\lceil \frac{x - \varphi_j(t_s)}{T_j} \right\rceil \right)^+ \cdot C_j.$$

We choose  $a = \min_{j \leq n} C_j/2$ . Similar to the proof of Lemma 2, we find  $f(a) > a$ . In order to choose an appropriate  $b$ , we make the following derivation.

$$f(x) \leq \sum_{j \leq n} \left\lceil \frac{x}{T_j} \right\rceil C_j$$

We now consider two disjunct cases for  $x = \text{lcm}(T_1, \dots, T_n)$ . If  $f(x) < \sum_{j \leq n} \left\lceil \frac{x}{T_j} \right\rceil C_j$ , we choose  $b = \text{lcm}(T_1, \dots, T_n)$ , and therefore get  $b > f(b)$ . Now the conditions for Lemma 15 hold, i.e. the function  $f(x)$  is defined and strictly non-decreasing in an interval  $[a, b]$  with  $f(a) > a$  and  $f(b) < b$ . Hence, there exists an  $x \in (\min_{j \leq n} \frac{C_j}{2}, \text{lcm}(T_1, \dots, T_n))$  such that  $x = f(x)$ . If  $f(x) = \sum_{j \leq i} \left\lceil \frac{x}{T_j} \right\rceil C_j$ , we found a (positive) solution and we are also done.  $\square$

Appendix B.1 presents an example consisting of two tasks with  $U = 1$  and the least common multiple of the periods does not exist, where the level- $n$  active period does not end.

**Theorem 7.** *When Assumption 1 holds, a level- $i$  active period that is started at time  $t_s$  is guaranteed to end.*

*Proof.* The theorem follows immediately from Lemmas 2 and 3.  $\square$

### 5.3.3 An iterative procedure

The next theorem provides an iterative procedure to determine the length of a level- $i$  active period.

**Theorem 8.** *Let the level- $i$  active period start with a release of task  $\tau_s$  at time  $t_s$ , and let the period start with an (optionally empty) initial interval of length  $B$  during which the tasks  $\tau_j$  with  $j \leq i$  are blocked by a subjob of a task with a lower priority. If Assumption 1 holds, the length  $L_i(t_s)$  of that level- $i$  active period can be found by the following iterative procedure.*

$$L_i^{(0)}(t_s) = B + C_s \tag{27}$$

$$L_i^{(l+1)}(t_s) = B + \sum_{j \leq i} \left( \left\lceil \frac{L_i^{(l)}(t_s) - \varphi_j(t_s)}{T_j} \right\rceil \right)^+ \cdot C_j, \quad l = 0, 1, \dots \tag{28}$$

*Proof.* From Lemma 2 and Lemma 3, we know that there exists a positive solution of Equation (26) when Assumption 1 holds. To prove the lemma, we first prove that the sequence is non-decreasing. Next, we prove that the procedure stops when the length  $L_i(t_s)$  is reached, i.e. for the smallest solution of Equation (26). To that end, we show that all values in the sequence  $L_i^{(l)}(t_s)$  are lower bounds on  $L_i(t_s)$ . To show that the procedure terminates, we show that the sequence can only take a finite number of values to reach that solution.

We prove that the sequence is non-decreasing, by induction. To this end, we start by noting that  $L_i^{(0)}(t_s) = B + C_s > 0$ , and

$$\begin{aligned} L_i^{(1)}(t_s) &= B + \sum_{j < i} \left( \left\lceil \frac{L_i^{(0)}(t_s) - \Phi_j(t_s)}{T_j} \right\rceil \right)^+ \cdot C_j \\ &\geq \{\Phi_s(t_s) = 0\} B + C_s = L_i^{(0)}(t_s) \end{aligned}$$

Next, if  $L_i^{(l+1)}(t_s) \geq L_i^{(l)}(t_s)$ , then we can conclude from Equation (28) that also  $L_i^{(l+2)}(t_s) \geq L_i^{(l+1)}(t_s)$ , as filling in a higher value in the right-hand side of Equation (28) gives a higher or equal result.

We next prove  $L_i^{(l)}(t_s) \leq L_i(t_s)$ , for all  $l = 0, 1, \dots$ , by induction. From Lemma 1 item (ii) we know  $L_i^{(0)}(t_s) = B + C_s \leq L_i(t_s)$ . Next, if  $L_i^{(l)}(t_s)$  is a lower bound on  $L_i(t_s)$ , then

$$\sum_{j < i} \left( \left\lceil \frac{L_i^{(l)}(t_s) - \Phi_j(t_s)}{T_j} \right\rceil \right)^+ \cdot C_j$$

is a lower bound on the amount of processing that needs to be performed due to releases of task  $\tau_i$  and its higher priority tasks in the interval of length  $L_i^{(l)}(t_s)$ , and hence  $L_i^{(l+1)}(t_s)$  is also a lower bound on  $L_i(t_s)$ .

Finally, we prove that the sequence can only take on a finite number of values. To this end, we note that  $L_i^{(l)}(t_s)$  is bounded from below by  $B + C_s$  and from above by the solution. □

#### 5.4 Level- $(i, k)$ active period

Similar to a level- $i$  active period, a level- $(i, k)$  active period is defined in terms of the notion pending load. For the definition of a level- $(i, k)$  active period, we first need to refine the notion of pending load.

##### 5.4.1 A refinement of pending load

Let a true level- $i$  active period start at time  $t = t_s$ . As described above, the length of that period is given by the smallest  $x > 0$  satisfying (26). Let the length of that period be  $L_i$ . The number of jobs  $l_i$  of task  $\tau_i$  in that period is now given by

$$l_i = \left\lceil \frac{L_i - \Phi_i(t_s)}{T_i} \right\rceil. \quad (29)$$

We now refine our notion of pending load  $P_i(t)$  by considering only the first  $k + 1 \leq l_i$  jobs of  $\tau_i$  in the active period, where  $k \in \mathbb{N}$ .

**Definition 6.** *The pending load  $P_{ik}(t)$  in a level- $i$  active period that started at time  $t_s < t$  is the amount of processing at time  $t$  that still needs to be performed for at most the first  $k + 1 \leq l_i$  jobs of  $\tau_i$  and the jobs of tasks  $\tau_j$  with  $j < i$  that are released in  $[t_s, t)$ , i.e.*

$$P_{ik}(t) = \min \left( \left( \left\lceil \frac{t - (t_s + \Phi_i(t_s))}{T_i} \right\rceil \right)^+, k + 1 \right) \cdot C_i + \sum_{j < i} \left( \left\lceil \frac{t - (t_s + \Phi_j(t_s))}{T_j} \right\rceil \right)^+ \cdot C_j - \int_{t_s}^t \sigma_i(t') dt', \quad (30)$$

where  $\sigma_i(t)$  as defined in Definition 3. □

##### 5.4.2 Definition of a level- $(i, k)$ active period

Similarly, we refine our notion of level- $i$  active period to level- $(i, k)$  active period.

**Definition 7.** *A level- $(i, k)$  active period is an interval  $[t_s, t_e)$  with the following three properties.*

1.  $P_{ik}(t_s) = 0$ ;
  2.  $P_{ik}(t_e) = 0$ ;
  3.  $P_{ik}(t) > 0$  for  $t \in (t_s, t_e)$ .
-

### 5.5 Length of a level- $(i, k)$ active period

**Theorem 9.** *Let the number of jobs of task  $\tau_i$  in a level- $i$  active period be given by  $l_i$ . The length  $L_{ik}$  of a level- $(i, k)$  active period with  $0 \leq k < l_i$  that starts at time  $t = t_s$  is the smallest  $x \in \mathbb{R}^+$  satisfies the following equation*

$$x = B + (k + 1)C_i + \sum_{j < i} \left( \left\lceil \frac{x - \Phi_j(t_s)}{T_j} \right\rceil \right)^+ \cdot C_j, \quad (31)$$

where  $B$  denotes the amount of time that a task with a lower priority than task  $\tau_i$  is executing non-preemptively as from time  $t = t_s$ .

*Proof.* The proof is similar to the proof of Theorem 6. □

## 6 Worst-case analysis for FPDS

This section provides theorems for the notions of critical instant and worst-case response times for tasks under FPDS and arbitrary phasing, and theorems to determine the worst-case response times analytically. We assume in this section that Assumption 1 holds. Moreover, we consider an arbitrary true level- $i$  active period with a start at time  $t_s$ . As described in Section 2.3, we will use abbreviated representations for the relative notions using a prime ( $'$ ) to denote the value of these notions relative to time  $t_s$ , e.g. we use  $a'_{ik}$  to denote  $a_{ik}(t_s)$ .

### 6.1 A critical instant

Similar to Equation (1), the worst-case response time  $WR_i^D$  of a task  $\tau_i$  under FPDS is the largest response time under arbitrary phasing, i.e.

$$WR_i^D = \sup_{\phi, k} r_{ik}.$$

We can refine this equation by taking blocking of tasks and the notion of level- $i$  active period into account. Assuming the start of an arbitrary true level- $i$  active period at time  $t_s$ , the worst-case response time  $WR_i^D$  of task  $\tau_i$  can therefore be described as

$$WR_i^D = \sup_{B, \phi'_1, \dots, \phi'_i} \max_{0 \leq k < l'_i(B, \phi'_1, \dots, \phi'_i)} r'_{ik}(B, \phi'_1, \dots, \phi'_i), \quad (32)$$

where  $B$  denotes the amount of time that a task with a lower priority than task  $\tau_i$  is executing non-preemptively as from the start of the level- $i$  active period, and  $l'_i$  is the number of jobs of task  $\tau_i$  in that level- $i$  active period.

We will now first present a lemma to determine the response time of job  $k$  of task  $\tau_i$  in a true level- $i$  active period. We subsequently present a theorem which states that given an infinitesimal time  $\varepsilon > 0$ , the maximum response time of task  $\tau_i$  is assumed in a level- $i$  active period which starts at an  $\varepsilon$ -critical instant. A next theorem refines Equation (32).

**Lemma 4.** *The response time  $r'_{ik}$  of job  $k$  of task  $\tau_i$  in a level- $i$  active period that starts at time  $t = t_s$  with  $0 \leq k < l'_i$  and  $l'_i$  the number of jobs of task  $\tau_i$  in that level- $i$  active period is given by*

$$r'_{ik}(B, \phi'_1, \dots, \phi'_i) = b'_{ik, m_i}(B, \phi'_1, \dots, \phi'_{i-1}) + F_i - (kT_i + \phi'_i), \quad (33)$$

where  $B$  denotes the amount of time that a task with a lower priority than task  $\tau_i$  is executing non-preemptively as from time  $t = t_s$ , and  $b'_{ik, m_i}$  is the relative begin time of the final subjob of job  $k$ , given by the smallest non-negative  $x \in \mathbb{R}$  satisfying

$$x = B + (k + 1)C_i - F_i + \sum_{j < i} \left( \left\lceil \frac{x - \phi'_j}{T_j} \right\rceil + 1 \right)^+ \cdot C_j. \quad (34)$$

*Proof.* We first look at the relative begin time  $b'_{ik, m_i}$  of the final subjob of that job  $k$ , and subsequently describe  $r'_{ik}$  in terms of the relative begin time, the relative activation time  $a'_{ik}$  and the computation time  $F_i$  of that final subjob.

The final subjob of job  $k$  of task  $\tau_i$  in the level- $i$  active period that starts at time  $t_s$  can begin at time  $t_s + b'_{ik, m_i}$  when

- the blocking subjob of the lower priority task has executed  $B$ ;
- all higher priority tasks that are released in  $[t_s, t_s + b'_{ik, m_i}]$  have a completion in that interval;

- all earlier jobs of task  $\tau_i$  and all earlier subjobs of job  $k$  that are released in  $[t_s, t_s + b'_{ik,m_i}]$  have a completion in that interval.

Note that the *order* in which the subjobs in the interval  $[t_s, t_s + b'_{ik,m_i}]$  are executed is irrelevant for the begin time of the final subjob of job  $k$  of task  $\tau_i$ . Stated in other words, the final subjob of job  $k$  of task  $\tau_i$  can start for the smallest  $t > t_s + \max(B, a'_{ik})$  for which  $P_{ik}(t) = F_i$ . The relative begin time  $b'_{ik,m_i}(B, \phi'_1, \dots, \phi'_{i-1})$  is therefore the smallest non-negative  $x \in \mathbb{R}$  satisfying the following equation:

$$x = B + (k+1)C_i - F_i + \sum_{j < i} \left( \left\lfloor \frac{x - \phi'_j}{T_j} \right\rfloor + 1 \right) \cdot C_j.$$

The relative completion time  $f'_{ik}$  of job  $k$  of  $\tau_i$  is now given by the relative begin time  $b'_{ik,m_i}$  plus the computation time  $F_i$ , i.e.  $f'_{ik} = b'_{ik,m_i} + F_i$ . The response time  $r'_{ik}$  of the job  $k$  is given by the relative completion time  $f'_{ik}$  minus the relative activation time  $a'_{ik}$ , i.e.

$$r'_{ik}(B, \phi'_1, \dots, \phi'_i) = b'_{ik,m_i}(B, \phi'_1, \dots, \phi'_{i-1}) + F_i - (kT_i + \phi'_i).$$

□

**Theorem 10.** *Given an infinitesimal time  $\varepsilon > 0$ , the maximum response time of task  $\tau_i$  under FPDS and arbitrary phasing is assumed when the level- $i$  active period is started at an  $\varepsilon$ -critical instant, i.e. when  $\tau_i$  has a simultaneous release with all higher priority tasks and a subjob of the lower priority tasks with computation time  $B_i^D$  starts a time  $\varepsilon$  before that simultaneous release.*

*Proof.* Let  $R'_i(B, \phi'_1, \dots, \phi'_i)$  denote  $\max_{0 \leq k < l'_i(B, \phi'_1, \dots, \phi'_i)} r'_{ik}(B, \phi'_1, \dots, \phi'_i)$ . We will prove that  $R'_i(B, \phi'_1, \dots, \phi'_i)$  assumes a maximum for  $\phi'_j = 0$  with  $j \leq i$  and  $B = (B_i^D - \varepsilon)^+$ . Hence, the maximum is assumed when  $\tau_i$  has a simultaneous release with all higher priority tasks, and a subjob of a lower priority task with computation time  $B_i^D$  starts an infinitesimal time  $\varepsilon > 0$  before that simultaneous release, which proves the theorem.

Based on Theorem 7, i.e. termination of a level- $i$  active period under Assumption 1, we conclude that

- only a finite number of jobs need to be considered to determine the worst-case response time of task  $\tau_i$ ;
- every job of  $\tau_i$  in a level- $i$  active period has a finite response time.

We will now look at the value of the length  $L'_i$  of the level- $i$  active period, the number  $l'_i$  of jobs of task  $\tau_i$  in the level- $i$  active period, and the response time  $r'_{ik}$  as a function of the relative phasing  $\phi'_j$  with  $j \leq i$  and the blocking time  $B$ . Consider Equation (26) for the length  $L'_i$  of a level- $i$  active period. The term  $\left\lfloor \frac{x - \phi'_j}{T_j} \right\rfloor$  in that equation is a strictly non-increasing function of  $\phi'_j$  with  $j \leq i$ . Because  $\phi'_j \geq 0$ , a maximum of that term is assumed for  $\phi'_j = 0$ . Moreover, the righthand side of Equation (26) is a strictly increasing function of  $B$ , and the length  $L'_i$  is therefore also a strictly increasing function of  $B$ . The largest value of  $L'_i$  is found for the largest value of  $B$  under consideration, i.e. for  $B = (B_i^D - \varepsilon)^+$ . As a consequence,  $L'_i$  assumes a maximum for  $\phi'_j = 0$  for all  $j \leq i$  and  $B = (B_i^D - \varepsilon)^+$ .

Given the behavior of  $L'_i$  and Equation (29), we conclude that the number of jobs  $l'_i$  of task  $\tau_i$  in the level- $i$  active period is a strictly non-increasing function of  $\phi'_j$  with  $j \leq i$  and a strictly non-decreasing function of  $B$ . As a consequence,  $l'_i$  assumes a maximum for  $\phi'_j = 0$  for all  $j \leq i$  and  $B = (B_i^D - \varepsilon)^+$ .

From Equation (33), we conclude that  $r'_{ik}(B, \phi'_1, \dots, \phi'_i)$  is a strictly decreasing function of  $\phi'_i$ . Because  $\phi'_i \geq 0$ , a maximum is assumed for  $\phi'_i = 0$ . Now consider Equation (34) for the relative begin time  $b'_{ik,m_i}$ . The term  $\left\lfloor \frac{x - \phi'_j}{T_j} \right\rfloor$  in that equation is a strictly non-increasing function of  $\phi'_j$ . Similarly to  $\phi'_i$ ,  $\phi'_j \geq 0$ , a maximum of that term is therefore assumed for  $\phi'_j = 0$ . Hence,  $b'_{ik,m_i}(B, 0, \dots, 0)$  dominates  $b'_{ik,m_i}(B, \phi'_1, \dots, \phi'_{i-1})$  for all values of  $B$  and all values of  $\phi'_j$  with  $j < i$ . Moreover, the righthand side of Equation (34) is a strictly increasing function of  $B$ , and  $b'_{ik,m_i}(B, 0, \dots, 0)$  is therefore also a strictly increasing function of  $B$ . The largest value of  $b'_{ik,m_i}(B, 0, \dots, 0)$  is found for the largest value of  $B$  under consideration, i.e. for  $B = (B_i^D - \varepsilon)^+$ . As a consequence,  $r'_{ik}(B, \phi'_1, \dots, \phi'_i)$  also assumes a maximum for  $\phi'_j = 0$  for all  $j \leq i$  and  $B = (B_i^D - \varepsilon)^+$ .

From the values of  $L'_i$ ,  $l'_i$  and  $r'_{ik}$  as a function of the relative phasing  $\phi'_j$  with  $j \leq i$  and the blocking time  $B$ , we conclude that  $R'_i(B, \phi'_1, \dots, \phi'_i)$  is a strictly non-increasing function of  $\phi'_1, \dots, \phi'_{i-1}$ , a strictly decreasing function of  $\phi'_i$ , and a strictly increasing function of  $B$ . As a result,  $R'_i(B, \phi'_1, \dots, \phi'_i)$  assumes a maximum for  $\phi'_j = 0$  with  $j \leq i$  and  $B = (B_i^D - \varepsilon)^+$ , which proves the theorem. □

**Theorem 11.** *The worst-case response time  $WR_i^D$  of task  $\tau_i$  under FPDS and arbitrary phasing is given by*

$$WR_i^D = \lim_{\varepsilon \downarrow 0} \max_{0 \leq k < l'_i((B_i^D - \varepsilon)^+, 0, \dots, 0)} r'_{ik} \left( (B_i^D - \varepsilon)^+, 0, \dots, 0 \right). \quad (35)$$

*Proof.* Once again, let  $R_i^D(B, \varphi'_1, \dots, \varphi'_i)$  denote  $\max_{0 \leq k < l'_i((B_i^D - \varepsilon)^+, 0, \dots, 0)} r'_{ik} \left( (B_i^D - \varepsilon)^+, 0, \dots, 0 \right)$ . From the proof of Theorem 10, we derive that  $R_i^D(B, 0, \dots, 0)$  dominates  $R_i^D(B, \varphi'_1, \dots, \varphi'_i)$  for all values of  $B$  and all values of  $\varphi'_j$  with  $j \leq i$ , i.e.

$$\begin{aligned} WR_i^D &= \sup_{B, \varphi'_1, \dots, \varphi'_i} R_i^D(B, \varphi'_1, \dots, \varphi'_i) \\ &= \sup_B R_i^D(B, 0, \dots, 0) \end{aligned}$$

Moreover,  $R_i^D(B, \varphi'_1, \dots, \varphi'_i)$  is a strictly increasing, i.e. monotonic, function of  $B$ . Hence,

$$\begin{aligned} WR_i^D &= \sup_B R_i^D(B, 0, \dots, 0) \\ &= \lim_{\varepsilon \downarrow 0} R_i^D \left( (B_i^D - \varepsilon)^+, 0, \dots, 0 \right), \end{aligned}$$

which proves the theorem. □

From the previous two theorems, we draw the following conclusions.

**Corollary 4.** *The worst-case response time  $WR_i^D$  is a supremum (and not a maximum) for all but the lowest priority task, i.e. that value cannot be assumed.* □

**Corollary 5.** *A critical instant is a supremum for all but the lowest priority task, i.e. that instant cannot be assumed.* □

## 6.2 Worst-case response times

The next theorem describes  $WR_i^D$  in terms of the worst-case response time  $WR_i^P$  and worst-case occupied time  $WO_i^P$  under FPPS.

First, we prove the following three lemmas for the worst-case length  $WL_i^D$  of a level- $i$  active period, the maximum number  $wl_i^D$  jobs of task  $\tau_i$  in a level- $i$  active period, and the worst-case response time  $WR_{ik}^D$  of job  $k$  of task  $\tau_i$ .

**Lemma 5.** *The worst-case length  $WL_i^D$  of a level- $i$  active period with  $i \leq n$  under FPDS is given by the smallest  $x \in \mathbb{R}^+$  that satisfies the following equation*

$$x = B_i^D + \sum_{j \leq i} \left\lceil \frac{x}{T_j} \right\rceil C_j. \quad (36)$$

*Proof.* The term  $\left\lceil \frac{x - \varphi'_j}{T_j} \right\rceil$  in Equation (26) is a strictly non-increasing function of  $\varphi'_j$  with  $j \leq i$ . Because  $\varphi'_j \geq 0$ , a maximum of that term is assumed for  $\varphi'_j = 0$ . Now let  $L'_i(B)$  denote the length of a level- $i$  active period with  $i \leq n$  for a simultaneous release of task  $\tau_i$  with all tasks with a higher priority. Hence,  $L'_i(B)$  is the smallest  $x \in \mathbb{R}^+$  satisfying equation (26) with  $\varphi'_j = 0$ , i.e. the smallest  $x \in \mathbb{R}^+$  satisfying

$$x = B + \sum_{j \leq i} \left\lceil \frac{x}{T_j} \right\rceil C_j. \quad (37)$$

We will now consider the cases  $i < n$  and  $i = n$  separately.

$\{i = n\}$  The lowest priority task is never blocked, therefore  $B_n^D = 0$ , and we immediately get (36) by substituting  $B = 0$  in equation (37) for  $i = n$ .

$\{i < n\}$  The righthand side of equation (37) is a strictly increasing function of  $B$ , and  $L'_i(B)$  is therefore also a strictly increasing function of  $B$ . The largest value for  $L'_i(B)$  is found for the largest value of  $B < B_i^D$ . Hence,  $WL_i^D$  is given by

$$WL_i^D = \lim_{B \uparrow B_i^D} L'_i(B). \quad (38)$$

Given Lemma 17, we can make the following derivation starting from this equation.

$$\begin{aligned}
WL_i^D &= \{(37)\} \lim_{B \uparrow B_i^D} \left( B + \sum_{j \leq i} \left\lceil \frac{L'_i(B)}{T_j} \right\rceil C_j \right) \\
&= B_i^D + \sum_{j \leq i} \lim_{B \uparrow B_i^D} \left\lceil \frac{L'_i(B)}{T_j} \right\rceil C_j \\
&= \{\text{Lemma 17}\} B_i^D + \sum_{j \leq i} \left\lceil \lim_{B \uparrow B_i^D} \frac{L'_i(B)}{T_j} \right\rceil C_j \\
&= \{(38)\} B_i^D + \sum_{j \leq i} \left\lceil \frac{WL_i^D}{T_j} \right\rceil C_j
\end{aligned}$$

Hence, the worst-case length  $WL_i^D$  is the smallest  $x \in \mathbb{R}^+$  satisfying (36), which proves the lemma.  $\square$

Because  $B_i^D$  is a supremum (and not a maximum) for all but the lowest priority task, we draw the following conclusion from the previous lemma.

**Corollary 6.** *The worst-case length  $WL_i^D$  is a supremum (and not a maximum) for all but the lowest priority task, i.e. that value cannot be assumed.*  $\square$

**Lemma 6.** *The maximum number  $wl_i^D$  of jobs of task  $\tau_i$  in a level- $i$  active period with  $i \leq n$  under FPDS is given by*

$$wl_i^D = \left\lceil \frac{WL_i^D}{T_i} \right\rceil. \quad (39)$$

*Proof.* We first derive Equation 39 and subsequently prove that  $WL_i^D$  is a maximum.

As described in the proof of Theorem 10,  $l'_i(B)$  is a strictly non-decreasing function of the blocking time  $B$ . Because  $B_i^D$  is a supremum that cannot be assumed, the largest value for  $l'_i(B)$  is therefore found for the largest value of  $B < B_i^D$ . Hence,  $wl_i^D$  is given by

$$wl_i^D = \lim_{B \uparrow B_i^D} l'_i(B). \quad (40)$$

Because  $\frac{L'_i(B)}{T_i}$  is a strictly increasing function of  $B$ , we can use Lemma 17 in the following derivation

$$\begin{aligned}
\lim_{B \uparrow B_i^D} l'_i(B) &= \lim_{B \uparrow B_i^D} \left\lceil \frac{L'_i(B)}{T_i} \right\rceil \\
&= \{\text{Lemma 17}\} \left\lceil \lim_{B \uparrow B_i^D} \frac{L'_i(B)}{T_i} \right\rceil \\
&= \{(38)\} \left\lceil \frac{WL_i^D}{T_i} \right\rceil.
\end{aligned}$$

Equation (39) immediately follows from Equation (40) and this latter equation.

The proof that  $wl_i^D$  is a maximum consists of two steps. We first prove that  $l'_i(B)$  is left-continuous in  $B_i^D$ , i.e.

$$l'_i(B_i^D) = \lim_{B \uparrow B_i^D} l'_i(B), \quad (41)$$

and subsequently prove that  $l'_i(B)$  is constant in an interval  $(B_i^D - \gamma, B_i^D]$  for a sufficiently small  $\gamma \in \mathbb{R}^+$ , i.e.

$$\forall_{B_i^D - \gamma < B \leq B_i^D} l'_i(B) = wl_i^D.$$

To prove that  $l'_i(B)$  is left-continuous in  $B_i^D$ , we show that  $L'_i(B_i^D)$  is defined and equal to  $WL_i^D$ , and subsequently show that  $l'_i(B_i^D) = wl_i^D$ . From Theorem 7, we know that  $L'_i(B)$  exists when Assumption 1 holds. Moreover, considering Theorem 6 and

Lemma 5, we conclude that  $WL_i^D$  and  $L'_i(B_i^D)$  are solutions of the same equation, i.e.  $L'_i(B_i^D) = WL_i^D$ . As a result, we get

$$l'_i(B_i^D) = \left\lceil \frac{L'_i(B_i^D)}{T_i} \right\rceil = \left\lceil \frac{WL_i^D}{T_i} \right\rceil = wl_i^D.$$

To prove that  $l'_i(B)$  is constant in an interval  $(B_i^D - \gamma, B_i^D]$  for a sufficiently small  $\gamma \in \mathbb{R}^+$ , we use the definition of limit:

$$\lim_{x \uparrow X} f(x) = Y \Leftrightarrow \forall \varepsilon > 0 \exists \delta > 0 \forall x \in (X - \delta, X) |f(x) - Y| < \varepsilon.$$

Because  $l'_i(B)$  is strictly non-decreasing and defined in  $B_i^D$ , we have

$$\forall_{0 \leq B \leq B_i^D} l'_i(B) \leq wl_i^D.$$

Let  $\varepsilon \in (0, 1]$ . Now there exists a  $\delta \in (0, B_i^D)$  such that  $0 \leq wl_i^D - l'_i(B) < \varepsilon \leq 1$  for all  $B \in (B_i^D - \delta, B_i^D]$ , hence  $wl_i^D \geq l'_i(B) > wl_i^D - 1$ . Because  $wl_i^D, l'_i(B) \in \mathbb{N}$ , this completes the proof.  $\square$

Note that unlike  $WL_i^D$ , the value for  $wl_i^D$  can be assumed. Based on Lemma 6, we conclude that  $l'_i((B_i^D - \gamma)^+) = wl_i^D$  for a sufficiently small  $\gamma \in \mathbb{R}^+$ , and we can therefore exchange the order of the operators in Equation 35, i.e.

$$WR_i^D = \max_{0 \leq k < wl_i^D} \lim_{\varepsilon \downarrow 0} r'_{ik} \left( (B_i^D - \varepsilon)^+ \right). \quad (42)$$

In the next lemma, we use  $WR_{ik}^D$  as a shorthand, i.e.

$$WR_{ik}^D = \lim_{\varepsilon \downarrow 0} r'_{ik} \left( (B_i^D - \varepsilon)^+ \right), \quad (43)$$

**Lemma 7.** *The worst-case response time  $WR_{ik}^D$  of job  $k$  with  $0 \leq k < wl_i^D$  of a task  $\tau_i$  under FPDS and arbitrary phasing is given by*

$$WR_{ik}^D = \begin{cases} WR_i^P(B_i^D + (k+1)C_i - F_i) + F_i - kT_i & \text{for } i < n \\ WO_n^P((k+1)C_n - F_n) + F_n - kT_n & \text{for } i = n \end{cases}, \quad (44)$$

where  $WR_i^P(B_i^D + (k+1)C_i - F_i)$  and  $WO_n^P(B_i^D + (k+1)C_i - F_i)$  are the worst-case response time and the worst-case occupied time under FPPS of a task  $\tau'_i$  with a computation time  $C'_i = B_i^D + (k+1)C_i - F_i$ , a period  $T'_i = kT_i + D_i - F_i$  and a deadline  $D'_i = T'_i$ .

*Proof.* Starting from Equation (43), we derive

$$\begin{aligned} WR_{ik}^D &= \lim_{\varepsilon \downarrow 0} r'_{ik} \left( (B_i^D - \varepsilon)^+ \right) \\ &= \{(33)\} \lim_{\varepsilon \downarrow 0} \left( b'_{ik, m_i} \left( (B_i^D - \varepsilon)^+ \right) + F_i - kT_i \right) \\ &= \lim_{\varepsilon \downarrow 0} b'_{ik, m_i} \left( (B_i^D - \varepsilon)^+ \right) + F_i - kT_i, \end{aligned}$$

where  $b'_{ik, m_i} \left( (B_i^D - \varepsilon)^+ \right)$  denotes the relative begin time of the final subjob of job  $k$  of task  $\tau_i$  with  $0 \leq k < wl_i$  and  $\phi'_j = 0$  for  $j \leq i$  as given in Equation (34). Hence,  $b'_{ik, m_i} \left( (B_i^D - \varepsilon)^+ \right)$  is the smallest  $x \in \mathbb{R}^+$  satisfying

$$x = \left( (B_i^D - \varepsilon)^+ \right) + (k+1)C_i - F_i + \sum_{j < i} \left( \left\lfloor \frac{x}{T_j} \right\rfloor + 1 \right) C_j.$$

Now let task set  $\mathcal{T}'$  be identical to  $\mathcal{T}$  except for the characteristics of task  $\tau_i$ , i.e.  $\tau'_i$  has characteristics  $C'_i = (B_i^D - \varepsilon)^+ + (k+1)C_i - F_i$ ,  $T'_i = kT_i + D_i - F_i$ , and  $D'_i = T'_i$ . Hence, task  $\tau'_i$  of  $\mathcal{T}'$  misses its deadline under FPPS and arbitrary phasing if and



only if task  $\tau_i$  of  $\mathcal{T}$  misses its deadline under FPDS, and arbitrary phasing and an amount of blocking  $(B_i^D - \varepsilon)^+$ . Based on Theorem 4, we can now write

$$b'_{ik,m_i} \left( (B_i^D - \varepsilon)^+ \right) = WO_i^P \left( (B_i^D - \varepsilon)^+ + (k+1)C_i - F_i \right).$$

For  $i = n$ , we substitute  $B_n^D = 0$ . We now immediately arrive at Equation (44), which proves the lemma for  $i = n$ .

For  $i < n$ , we derive

$$\begin{aligned} WR_{ik}^D &= \lim_{\varepsilon \downarrow 0} WO_i^P \left( (B_i^D - \varepsilon)^+ + (k+1)C_i - F_i \right) + F_i - kT_i \\ &= \{(14)\} WR_i^P (B_i^D + (k+1)C_i - F_i) + F_i - kT_i, \end{aligned}$$

which proves the lemma for  $i < n$ . □

Note that  $WR_{ik}^D$  is a supremum (and not a maximum) for all but the lowest priority task, i.e. that value cannot be assumed.

**Theorem 12.** *The worst-case response time  $WR_i^D$  of a task  $\tau_i$  under FPDS and arbitrary phasing is given by*

$$WR_i^D = \max_{0 \leq k < wl_i^D} WR_{ik}^D. \quad (45)$$

*Proof.* The theorem follows immediately from Equations (42) and (43), and requires Lemma 7. □

### 6.3 An iterative procedure

The next theorem provides an iterative procedure to determine the worst-case response time  $WR_i^D$  for task  $\tau_i$  under FPDS and arbitrary phasing. The procedure is stopped when the worst-case response time  $WR_{ik}^D$  of job  $k$  for task  $\tau_i$  exceeds the deadline  $D_i$  or when the level- $i$  active period is over. This latter condition is based on a property of  $WL_i^D$ .

**Lemma 8.** *The worst-case length  $WL_{ik}^D$  of a level- $(i,k)$  active period under FPDS is the smallest positive  $x \in \mathbb{R}^+$  satisfying the following equation*

$$x = B_i^D + (k+1)C_i + \sum_{j < i} \left\lceil \frac{x}{T_j} \right\rceil C_j. \quad (46)$$

*Proof.* The proof is similar to the proof of Lemma 5. □

Note that because  $B_i^D$  is a supremum (and not a maximum) for all tasks, except the lowest priority task,  $WL_{ik}^D$  is also supremum (and not a maximum) for all tasks, except the lowest priority task, i.e. that value cannot be assumed.

**Lemma 9.** *The worst-case length  $WL_{ik}^D$  of a level- $(i,k)$  active period under FPDS is given by*

$$WL_{ik}^D = WR_i^P (B_i^D + (k+1)C_i). \quad (47)$$

where  $WR_i^P (B_i^D + (k+1)C_i)$  is the worst-case response time under FPPS and arbitrary phasing of a task  $\tau'_i$  with a computation time  $C'_i = B_i^D + (k+1)C_i$ , a period  $T'_i = (k+1)T_i + D_i$  and a deadline  $D'_i = T'_i$ .

*Proof.* The lemma follows from the similarity between Equations (7) and (46). The period and deadline of task  $\tau'_i$  have been chosen to be equal to the deadline of job  $k+1$  of task  $\tau_i$ . Hence, when the iterative procedure to determine  $WR_i^P (B_i^D + (k+1)C_i)$  stops because the deadline  $D'_i$  is exceeded, the deadline  $d_{i,k+1}$  will be exceeded as well. □

**Lemma 10.** *Let  $k' \in \mathbb{N}$  be the smallest value for which  $WR_i^P (B_i^D + (k'+1)C_i) \leq (k'+1)T_i$ . The worst-case length  $WL_i^D$  of a level- $i$  active period is now given by  $WR_i^P (B_i^D + (k'+1)C_i)$ .*

*Proof.* To prove the lemma, we will prove the following equivalent relation by means of a contradiction argument

$$\forall_{0 \leq k < wl_i^D} (WL_{ik}^D \leq (k+1)T_i \Rightarrow k = wl_i^D - 1).$$

We only consider  $k < wl_i^D - 1$ , because the proof for  $k = wl_i^D - 1$  is similar.

Let  $WL_{i,k}^D \leq (k+1)T_i$  for  $0 \leq k < wl_i^D - 1$ . Using Lemma 9, we derive  $WR_i^P(B_i^D + (k+1)C_i) \leq (k+1)T_i$ . Hence, task  $\tau'_i$  has a completion at or before  $(k+1)T_i$ , and all higher priority tasks that are released in the interval  $[0, WR_i^P(B_i^D + (k+1)C_i))$  have a completion in that interval. Because task  $\tau'_i$  represents the executions of both the blocking lower priority task as well as task  $\tau_i$ , all executions of the corresponding jobs also have a completion in that interval. Hence, the level- $i$  active period that started with an  $\varepsilon$ -critical instant ends at time  $WR_i^P(B_i^D + (k+1)C_i)$ . However, we now have that the length of the level- $i$  active period equals  $WL_{i,k}^D$ , a value that is strictly smaller than  $WL_i^D$ , which is a contradiction. Therefore, our assumption that  $WL_{i,k}^D \leq (k+1)T_i$  for  $0 \leq k < wl_i^D - 1$  is wrong, which proves the lemma.  $\square$

From this lemmas, we draw the following conclusion.

**Corollary 7.** *The level- $i$  active period is over for the smallest  $k' \in \mathbb{N}$  for which  $WR_i^P(B_i^D + (k'+1)C_i) \leq (k'+1)T_i$ .*  $\square$

**Theorem 13.** *The worst-case response time  $WR_i^D$  of a task  $\tau_i$  can be found by the following iterative procedure under Assumption 1, using (44).*

$$WR_i^{(0)} = WR_{i,0}^D \quad (48)$$

$$WR_i^{(l+1)} = \max(WR_i^{(l)}, WR_{i,l+1}^D) \quad l = 0, 1, \dots \quad (49)$$

*The procedure is stopped when the worst-case response time  $WR_{ik}^D$  of job  $k$  of task  $\tau_i$  exceeds the deadline  $D_i$  or when the level- $i$  active period is over, i.e.  $WR_i^P(B_i^D + (k+1)C_i) \leq (k+1)T_i$ .*

*Proof.* Corollary 7 states that  $WR_i^P(B_i^D + (k+1)C_i) \leq (k+1)T_i$  is a proper termination condition to determine whether or not the level- $i$  active period is over before the release of job  $k+1$ . Because of Theorem 7, the level- $i$  active periods ends under Assumption 1, and we therefore have to consider at most a finite number  $wl_i^D$  of jobs of task  $\tau_i$ . As a result, the iterative procedure ends. We observe that the iterative procedure also stops when the deadline  $D_i$  is exceeded, by the worst-case response time  $WR_{ik}^D$  of job  $k$  of  $\tau_i$  i.e. when the task set is not schedulable.  $\square$

**Corollary 8.** *When Assumption 1 holds, we can derive the schedulability of a set of tasks  $\mathcal{T}$  under FPDS and arbitrary phasing by checking the schedulability criterion  $WR_i^D \leq D_i$  using Theorem 13.*  $\square$

**Corollary 9.** *To check the schedulability criterion  $WR_i^D \leq D_i$  we do not need to determine the length  $WL_i^D$  of the worst-case level- $i$  active period under FPDS first. Instead, we can simply check whether or not the level- $i$  active period is over after every iteration.*  $\square$

Finally note that

- $WR_{i,k}^D$  can be used as initial value to calculate  $WR_i^P(B_i^D + (k+1)C_i)$  to determine whether or not the level- $i$  active period is over before the release of job  $k+1$ ;
- $WR_i^P(B_i^D + (k+1)C_i)$  can be used as initial value to calculate  $WR_i^P(B_i^D + (k+2)C_i - F_i)$  to determine  $WR_{i,k+1}^D$ .

## 7 Examples

In this section, we will illustrate the worst-case response time analysis presented in Section 6 to determine the schedulability of tasks and task sets under FPDS and arbitrary phasing of some examples of Section 4 using the iterative procedure presented in Theorem 13.

### 7.1 Schedulability of task $\tau_2$ of $\mathcal{T}_2$

The schedulability of task  $\tau_2$  of task set  $\mathcal{T}_2$  is the topic of this section. The characteristics of the tasks of  $\mathcal{T}_2$  can be found in Table 2 on page 8 in Section 4.2.

To determine the worst-case response time  $WR_2^D$  for task  $\tau_2$ , we first derive  $B_2^D = 2$  using Equation (17). Next, we determine  $WR_2^{(0)}$  using Lemma 7, i.e.

$$WR_2^{(0)} = WR_{2,0}^D = WR_2^P(B_2^D + C_2 - F_2) + F_2 = WR_2^P(3) + 2 = 5 + 2 = 7.$$

Because  $WR_{2,0}^D \leq D_2 = 7$  and  $WR_2^P(B_2^D + C_2) = WR_2^P(5) = 9 > T_2 = 7$ , i.e. the level-2 active period is not over yet, we proceed with the 2<sup>nd</sup> job.

For the 2<sup>nd</sup> job, we find

$$WR_{2,1}^D = WR_2^P(B_2^D + 2C_2 - F_2) + F_2 - T_2 = WR_2^P(6) - 5 = 10 - 5 = 5,$$

and therefore  $WR_2^{(1)} = \max(WR_2^{(0)}, WR_{2,1}^D) = \max(7, 5) = 7$ . Now  $WR_{2,1}^D = 5 \leq D_2$  and  $WR_2^P(B_2^D + 2C_2) = WR_2^P(8) = 14 \leq 2T_2 = 14$ . Hence, we know that the level-2 active period is over, all jobs of task  $\tau_2$  meet their deadlines in that period, and the worst-case response time  $WR_2^D = 7$ .

## 7.2 Schedulability of task $\tau_2$ of $\mathcal{T}_5$

We will determine the schedulability of task  $\tau_2$  of task set  $\mathcal{T}_5$  in this section. The characteristics of the tasks of  $\mathcal{T}_5$  can be found in Table 5 on page 10 in Section 4.3.2.

We first determine  $WR_2^{(0)}$  using Lemma 7, i.e.

$$WR_2^{(0)} = WR_{2,0}^D = WO_2^P(B_2^D + C_2 - F_2) + F_2 = WO_2^P(2) + 2.1 = 4 + 2.1 = 6.1.$$

Because  $WR_{2,0}^D \leq D_2 = 7$  and  $WR_2^P(B_2^D + C_2) = WR_2^P(4.1) = 8.1 > T_2 = 7$ , we proceed with the 2<sup>nd</sup> job.

For the 2<sup>nd</sup> job, we find

$$WR_{2,1}^D = WO_2^P(B_2^D + 2C_2 - F_2) + F_2 - T_2 = WO_2^P(6.1) - 4.9 = 12.1 - 4.9 = 7.2.$$

Because  $WR_{2,1}^D > D_2 = 7$ , we conclude that task  $\tau_2$  is not schedulable.

## 7.3 Schedulability of the task set $\mathcal{T}_6$

In this section, we will determine the schedulability of the task set  $\mathcal{T}_6$ . The characteristics of the tasks of  $\mathcal{T}_6$  can be found in Table 6 on page 11 in Section 4.3.3.

To determine the worst-case response time  $WR_1^D$  for task  $\tau_1$ , we first derive  $B_1^D = 3$  using Equation (17). Next, we determine  $WR_2^{(0)}$  using Lemma 7, i.e.

$$WR_1^{(0)} = WR_{1,0}^D = WR_1^P(B_1^D + C_1 - F_1) + F_1 = 3 + 2 = 5.$$

Now  $WR_{1,0}^D = D_1$  and  $WR_1^P(B_1^D + C_1) = 5 = T_1$ . Hence, we know that the level-1 active period is over, all jobs of task  $\tau_1$  meet their deadlines, and the worst-case response time  $WR_1^D = 5$ .

Next, we determine the worst-case response time  $WR_2^D$  for task  $\tau_2$ . To this end, we first determine  $WR_2^{(0)}$  using Lemma 7, i.e.

$$WR_2^{(0)} = WR_{2,0}^D = WO_2^P(B_2^D + C_2 - F_2) + F_2 = WO_2^P(1.2) + 3 = 3.2 + 3 = 6.2.$$

Because  $WR_{2,0}^D < D_2 = 7$  and  $WR_2^P(B_2^D + C_2) = 8.2 > T_2 = 7$ , we proceed with the 2<sup>nd</sup> job.

For the 2<sup>nd</sup> job, we find

$$WR_{2,1}^D = WO_2^P(B_2^D + 2C_2 - F_2) + F_2 - T_2 = WO_2^P(5.4) - 4 = 9.4 - 4 = 5.4,$$

and therefore  $WR_2^{(1)} = \max(WR_2^{(0)}, WR_{2,1}^D) = \max(6.2, 5.4) = 6.2$ . Because  $WR_{2,1}^D < D_2$  and  $WR_2^P(B_2^D + 2C_2) = 14.4 > 2T_2 = 14$ , we proceed with the 3<sup>rd</sup> job.

For the 3<sup>rd</sup> job, we find

$$WR_{2,2}^D = WO_2^P(B_2^D + 3C_2 - F_2) + F_2 - 2T_2 = WO_2^P(9.6) - 11 = 17.6 - 11 = 6.6,$$

and therefore  $WR_2^{(2)} = \max(WR_2^{(1)}, WR_{2,2}^D) = \max(6.2, 6.6) = 6.6$ . Because  $WR_{2,2}^D < D_2$  and  $WR_2^P(B_2^D + 3C_2) = 22.6 > 3T_2 = 21$ , we proceed with the 4<sup>th</sup> job.

For the 4<sup>th</sup> job, we find

$$WR_{2,3}^D = WO_2^P(B_2^D + 4C_2 - F_2) + F_2 - 3T_2 = WO_2^P(13.8) - 18 = 23.8 - 18 = 5.8.$$

and therefore  $WR_2^{(3)} = \max(WR_2^{(2)}, WR_{2,3}^D) = \max(6.6, 5.8) = 6.6$ . Because  $WR_{2,3}^D < D_2$  and  $WR_2^P(B_2^D + 4C_2) = 28.8 > 4T_2 = 28$ , we proceed with the 5<sup>th</sup> job.

For the 5<sup>th</sup> job, we find

$$WR_{2,4}^D = WO_2^P(B_2^D + 5C_2 - F_2) + F_2 - 4T_2 = WO_2^P(18) - 25 = 32 - 25 = 7,$$

and therefore  $WR_2^{(4)} = \max(WR_2^{(3)}, WR_{2,4}^D) = \max(6.6, 7) = 7$ . Now  $WR_{2,4}^D = D_2$  and  $WR_2^P(B_2^D + 5C_2) = 35 = 5T_2$ . Hence we know that the level-2 active period is over, all jobs of task  $\tau_2$  meet their deadlines in that period, and the worst-case response time  $WR_2^D = 7$ .

Because  $WR_i^D \leq D_i$  for all  $i \leq n$ , we conclude that  $\mathcal{T}_6$  is schedulable under FPDS and arbitrary phasing when deadlines are equal to periods.

## 8 Discussion

This section presents a theorem for the worst-case response time of the highest priority task, compares the notion of level- $i$  active period with similar notions in the literature, and presents pessimistic variants for the worst-case response time analysis of tasks under FPDS and arbitrary phasing.

### 8.1 Worst-case response time of highest priority task

The next theorem states that the worst-case response time of the highest priority task  $\tau_1$  can be found by only considering the first job of  $\tau_1$  in a level-1 active period started at an  $\varepsilon$ -critical instant.

First, we prove the following lemma.

**Lemma 11.** *The first job of task  $\tau_1$  in a level-1 active period has the largest response time of all jobs of  $\tau_1$  in that period.*

*Proof.* The highest priority task  $\tau_1$  experiences blocking of at most one subjob of a lower priority task. If the first job of  $\tau_1$  in a level-1 active period is blocked by an amount  $B$ , its response time  $r'_{1,0}(B)$  becomes

$$r'_{1,0}(B) = B + C_1.$$

Now, assume the level-1 active period contains  $l_1 > 1$  jobs of task  $\tau_1$ . The response time  $r'_{1,k}(B)$  of job  $k$ , with  $0 \leq k < l_1$ , becomes

$$\begin{aligned} r'_{1,k}(B) &= B + (k+1)C_1 - kT_1 \\ &= B + C_1 + k(C_1 - T_1) \\ &= B + C_1 + k(U_1 - 1)T_1 \end{aligned}$$

When task  $\tau_1$  is blocked by a lower priority task,  $U_1 < 1$ . Hence, we find

$$r'_{1,k}(B) < B + C_1 = r'_{1,0}(B),$$

which proves the lemma. □

**Theorem 14.** *The worst-case response time  $WR_1^D$  of the highest priority task  $\tau_1$  under FPDS is equal to*

$$WR_1^D = B_1^D + C_1. \tag{50}$$

*Proof.* From equation  $r'_{1,0}(B) = B + C_1$ , we conclude that  $r'_{1,0}(B)$  is a strictly increasing function of  $B$ . Hence, we derive

$$WR_1^D = \sup_B r'_{1,0}(B) = \lim_{B \uparrow B_1^D} (B + C_1) = B_1^D + C_1,$$

which proves the theorem. □

### 8.2 A comparison with existing notions

We will now compare our notion of level- $i$  active period with similar notions in the literature.

### 8.2.1 Level- $i$ busy period in [25]

The notion of *level- $i$  busy period* originates from [25], where it has been introduced as an expedient to determine the worst-case response times of tasks for deadlines larger than periods under FPPS and arbitrary phasing. The level- $i$  busy period is defined as follows.

**Definition 8.** A level- $i$  busy period is a time interval  $[a, b]$  within which jobs of priority  $i$  or higher are processed throughout  $[a, b]$  but no jobs of level  $i$  or higher are processed in  $(a - \varepsilon, a)$  or  $(b, b + \varepsilon)$  for sufficiently small  $\varepsilon > 0$ .  $\square$

Figure 9 also shows the level-1 busy periods and level-2 busy periods for  $\mathcal{T}_1$ . The level-1 busy periods in this figure only differ from the level-1 active periods by the inclusion of the end-points of the intervals by the former. The difference between level-2 busy periods and level-2 active periods is more significant, however. Whereas the interval  $[0, 12]$  is constituted by four level-2 active periods, i.e.  $[0, 5)$ ,  $[5, 7)$ ,  $[7, 10)$ , and  $[10, 12)$ , the interval is contained in a single level-2 busy period  $[0, 12]$ . Stated in other words, the level-2 busy period unifies four adjacent level-2 active periods. Similarly, the interval  $[20, 27]$  is constituted by two level-2 active periods, i.e.  $[20, 25)$  and  $[25, 27)$ , and the interval is contained in a single level-2 busy period  $[20, 27]$ .

Figure 10 shows the level-1 busy periods and level-2 busy periods for  $\mathcal{T}_1$ . From this figure, we see that the level-2 busy period never ends for  $U = 1$ , as also becomes immediately clear from Definition 8. Conversely, the level-2 active period that started at time  $t = 0$  ends at time  $t = 35$ ; see also Assumption 1 and Theorem 7. We observe that the definition of level- $i$  busy period is included in [22] (on page D-4, referring to [25]), and the notion is used in Technique 5 “Calculating Response Time with Arbitrary Deadlines and Blocking.” As shown above, the busy period never ends for  $U = 1$ . Notably, in [22] on page 4–35 it is only mentioned that *we must be sure that the [...] utilization [...] is not greater than one*. In Step 6 “Decide if the busy period is over” the notion is used to determine whether or not the iterative procedure can be stopped. Notably, that decision is not based on the end of the busy period, but on the end of the level- $i$  active period, i.e. when the (worst-case) response time  $WR_{ik}^P$  of job  $k$  of task  $\tau_i$  is less than or equal to  $T_i$ ; see also Theorem 13.

There is another striking difference between the level- $i$  active period and the level- $i$  busy period. A level- $i$  active period may start when a task with a lower priority is still being processed, as illustrated by the level-1 active period that starts at time  $t = 5$  in Figure 10. The corresponding level-1 busy period does not start at time  $t = 5$ , but at time  $t = 6.2$  instead.

The fundamental difference between both notions can be traced back to their definitions; a busy period is based on a schedule, i.e. the definition refers to processing of jobs, whereas an active period is based on (pending) load or active jobs.

### 8.2.2 $\tau_i$ -busy period in [17]

In [17], the notion of busy period is slightly modified to accommodate the fact that a task  $\tau_i$  may be composed of distinct subtasks, each of which may have its own timing requirements and fixed priority. In the following definition,  $\rho_i$  denotes the minimum priority of the subtasks of task  $\tau_i$ .

**Definition 9.** A  $\tau_i$ -idle instant is any time  $t$  such that all work of priority  $\rho_i$  or higher started before  $t$  and all  $\tau_i$  jobs also started before  $t$  have completed at or before  $t$ .  $\square$

**Definition 10.** A  $\tau_i$ -busy period is an interval of time  $[A, B]$  such that both  $A$  and  $B$  are  $\tau_i$ -idle instants and there is no time  $t \in (A, B)$  such that  $t$  is a  $\tau_i$ -idle instant.  $\square$

This notion of  $\tau_i$ -busy period is similar to our level- $i$  active period, with as main difference that a  $\tau_i$ -busy period is a closed interval rather than a right semi-open interval. Although this difference may be viewed as philosophical, we prefer the usage of a right semi-open interval, which we will motivate by means of Figure 10. Given Definition 9 and 10, time  $t = 35$  belongs to two  $\tau_2$ -busy periods, i.e.  $[0, 35]$  and  $[35, 70]$ . Moreover, time  $t = 35$  is also a  $\tau_2$ -idle instant. Hence,  $\tau_i$ -busy periods can overlap, and when they overlap, the overlap is termed a  $\tau_i$ -idle instant. This is considered to be counter-intuitive.

### 8.2.3 Level- $i$ busy period in [16]

After a brief recapitulation of the notion of level- $i$  busy period of [25] for FPPS, an informal description of a level- $i$  busy period for FPNS under discrete scheduling [4] is given in Appendix A.2 of [16]. Note that for discrete scheduling, all task parameters are integers, i.e.  $T_i, C_i, D_i \in \mathbb{Z}^+$  and  $\varphi_i \in \mathbb{Z}^+ \cup \{0\}$ , and preemptions are restricted to integer time points.

Unfortunately, there is an inconsistency in [16]. In Appendix A.2, the following definition is given.

**Definition 11.** A level- $i$  busy period is a processor busy period in which only instances of tasks with a priority greater than or equal to that of  $\tau_i$  execute.  $\square$

Accordingly, the interval of time that a lower priority task blocks task  $\tau_i$  and its higher priority tasks is not included in the level- $i$  busy period in both the text of the proof of Lemma 6 in Section 4.3.1 and Figure 6, which is used to illustrate that proof. Conversely, that interval *is* included in the equation to determine the length of the level- $i$  busy period for the non-preemptive case, as described in Appendix A.2 in [16].

Note that [16] does not reproduce the definition of [25] (see Definition 8 above), but presents a new definition. Surprisingly, the differences between these definitions are not discussed. As an example, a (synchronous processor) busy period in [16] is described as a right semi-open interval on page 6, whereas the level- $i$  busy period in [25] is a closed interval.

The notion of level- $i$  busy period for FPNS in [16] is similar to our notion of level- $i$  active period under the assumption that the equation to determine the length of a level- $i$  busy period for the non-preemptive case properly reflects the intention of the authors.

#### 8.2.4 Level- $\pi_i$ busy interval in [28]

In [28], an analysis method is described to determine the schedulability of tasks under FPPS whose relative deadlines are larger than their respective periods, using the term *level- $\pi_i$  busy interval*. A level- $\pi_i$  busy interval is defined as a left semi-open interval  $(t_0, t]$ , i.e. the partitioning of the timeline in [28] differs from ours. Given the description in [28], our definition of level- $i$  active period can be viewed as a slightly modified definition of level- $\pi_i$  busy interval to accommodate our scheduling model for FPDS.

### 8.3 Pessimistic variants

Given Equation (45) in Theorem 12, we observe that the worst-case response time analysis is not uniform for all tasks. The analysis can be made uniform at the cost of potentially introducing pessimism. This section presents two lemmas with pessimistic variants for the worst-case response time analysis, one based on worst-case occupied times and one based on worst-case response times. For both variants, the iterative procedure presented in Theorem 13 can be used, i.e. only the equations for  $WR_{ik}^D$  change, not the iterative procedure.

#### 8.3.1 A uniform analysis based on $WO^P$

**Lemma 12.** *A pessimistic worst-case response time  $\widehat{WR}_{ik}^D$  of job  $k$  with  $0 \leq k < wl_i^D$  of a task  $\tau_i$  under FPDS and arbitrary phasing is given by*

$$\widehat{WR}_{ik}^D = WO_i^P(B_i^D + (k+1)C_i - F_i) + F_i - kT_i, \quad (51)$$

where  $WO_i^P(B_i^D + (k+1)C_i - F_i)$  is the worst-case occupied time under FPPS of a task  $\tau_i'$  with a computation time  $C_i' = B_i^D + (k+1)C_i - F_i$ , a period  $T_i' = kT_i + D_i - F_i$ , and a deadline  $D_i' = T_i'$ .

*Proof.* By definition,  $WR_i^P(C) \leq WO_i^P(C)$ , hence  $WR_{ik}^D \leq \widehat{WR}_{ik}^D$ . Because  $WR_1^P(C) = WO_1^P(C)$ ,  $\widehat{WR}_{ik}^D$  is potentially pessimistic for  $1 < i < n$ .  $\square$

The pessimism is illustrated by the set  $\mathcal{T}_2$  consisting of three tasks with characteristics as described in Table 2 on page 8 in Section 4.2. For the worst-case response time  $\widehat{WR}_{2,0}^D$  of the 1<sup>st</sup> job of task  $\tau_2$  we find

$$\begin{aligned} \widehat{WR}_{2,0}^D &= WO_2^P(B_2^D + C_2 - F_2) + F_2 \\ &= WO_2^P(2 + 3 - 2) + 2 \\ &= WO_2^P(3) + 2 = 7 + 2 = 9. \end{aligned}$$

Because  $\widehat{WR}_{2,0}^D > D_2$ ,  $\mathcal{T}_2$  is considered unschedulable under FPDS based on Theorem 12. Conversely, application of Theorem 12 yields a value  $WR_2^D = 7 \leq D_2$ .

We observe that  $\widehat{WR}_{2,0}^D$  is equal to  $\widetilde{WR}_2^D$  as determined in Section 4.2 by means of the existing analysis as presented in [11] and [13]. This equality is not a coincidence, for the following two reasons. Firstly, remember that because the characteristics of the tasks of  $\mathcal{T}_2$  are integral multiples of a value  $\delta = 1$  and  $\Delta = 0.2 \leq \delta$ , the value for  $\widehat{WR}_2^D$  does not change when  $\Delta$  is reduced to an arbitrary small positive value, i.e.

$$\widehat{WR}_2^D = \lim_{\Delta \downarrow 0} (WR_2^P(B_2^D + C_2 - (F_2 - \Delta)) + (F_2 - \Delta)).$$

Secondly, we can make the following derivation using Equation (10)

$$\begin{aligned} \lim_{\Delta \downarrow 0} (WR_2^P(B_2^D + C_2 - (F_2 - \Delta)) + (F_2 - \Delta)) &= \lim_{\Delta \downarrow 0} (WR_2^P(B_2^D + C_2 - (F_2 - \Delta))) + F_2 \\ &= \{(10)\} WO_2^P(B_2^D + C_2 - F_2) + F_2 \\ &= \widehat{WR}_{2,0}^D. \end{aligned}$$

These two results show that  $\widehat{WR}_{2,0}^D = \widetilde{WR}_2^D$  for  $\mathcal{T}_2$ .

### 8.3.2 A uniform analysis based on $WR^P$

We will give another pessimistic approach that is uniform for all tasks, which assumes a value  $\Delta$  and is based on  $WR^P$ .

**Lemma 13.** *A pessimistic worst-case response time  $\widehat{WR}_{ik}^D$  of job  $k$  with  $0 \leq k < wl_i^D$  of a task  $\tau_i$  under FPDS and arbitrary phasing is given by*

$$\widehat{WR}_{ik}^D = WR_i^P(B_i^D + (k+1)C_i - (F_i - \Delta)) + (F_i - \Delta) - kT_i \quad (52)$$

where

(i)  $WR_i^P(B_i^D + (k+1)C_i - (F_i - \Delta))$  is the worst-case response time under FPPS of a task  $\tau_i^!$  with a computation time  $C_i^! = B_i^D + (k+1)C_i - (F_i - \Delta)$ , a period  $T_i^! = kT_i + D_i - (F_i - \Delta)$ , and a deadline  $D_i^! = T_i^!$ ;

(ii)  $\Delta$  is an arbitrary small positive value.

*Proof.* Because  $WR_1^P(C) = WO_1^P(C) = C$ ,  $\widehat{WR}_{1,0}^D = \widetilde{WR}_{1,0}^D = WR_1^D$ . Hence, this approach is not pessimistic for  $i = 1$ . We will now prove that  $WR_i^P(C + \Delta) - \Delta \geq WO_i^P(C)$  for  $1 < i \leq n$ . The potentially additional pessimism introduced by Equation (52) now immediately follows from Lemma 12, i.e.  $\widehat{WR}_{ik}^D \geq \widetilde{WR}_{ik}^D$ .

By definition, task  $\tau_i$  can start executing an additional amount of computation time  $\Delta$  after having executed an amount  $C$  at time  $WO_i^P(C)$ . Because execution of that additional computation time  $\Delta$  takes at least an amount of time  $\Delta$ , we immediately get  $WR_i^P(C + \Delta) \geq WO_i^P(C) + \Delta$ , which proves the theorem.  $\square$

Based on Equation (52) and Equation (18), we first conclude that  $\widehat{WR}_{i,0}^D = \widetilde{WR}_i^D$ . For task  $\tau_2$  of  $\mathcal{T}_2$  we therefore also find a pessimistic value, i.e.  $\widehat{WR}_{2,0}^D = 9$ .

The additional pessimism is illustrated by the set  $\mathcal{T}_3$  consisting of three tasks with characteristics as described in Table 3 on page 9 in Section 4.2. We now reconsider that example. As explained in Section 4.2, the task characteristics are integral multiples of  $\delta = 0.5$ . For  $\Delta = 0.6 > \delta$ , we find  $\widehat{WR}_{2,0}^D = \widetilde{WR}_2^D = 12$ , which is larger than  $\tau_2$ 's deadline. Conversely, the worst-case response time  $\widehat{WR}_2^D$  of task  $\tau_2$  determined by means of Theorem 13 using Lemma 12 yields  $\widehat{WR}_2^D = WR_2^D = 9 \leq D_2$ . For  $\Delta = 0.4 < \delta$ , we find  $\widehat{WR}_{2,0}^D = \widetilde{WR}_2^D = 9$ . For this value of  $\Delta$ ,  $\widehat{WR}_{2,0}^D = \widetilde{WR}_2^D = WR_2^D = 9 \leq D_2$ , and reducing the value of  $\Delta$  will not change the value found for  $\widehat{WR}_{2,0}^D$ .

**Lemma 14.** *When the greatest common divisor ( $\gcd^{\mathbb{R}^+}$ ) of the periods and computation times of the tasks exists, and is equal to  $\delta$ ,  $\Delta < \delta$  is a sufficient condition to guarantee that Lemma 13 introduces no additional pessimism compared to Lemma 12.*

*Proof.* To prove the lemma, it suffices to prove

$$\Delta < \delta \Rightarrow WR_i^P(B_i^D + (k+1)C_i - (F_i - \Delta)) - \Delta = WO_i^P(B_i^D + (k+1)C_i - F_i).$$

From Theorem 2, we derive that  $WR_i^P(B_i^D + (k+1)C_i - (F_i - \Delta))$  is given by the smallest  $x \in \mathbb{R}^+$  that satisfies the following equation, provided that  $x$  is at most  $kT_i + D_i - (F_i - \Delta)$ ,

$$x = B_i^D + (k+1)C_i - (F_i - \Delta) + \sum_{j < i} \left\lceil \frac{x}{T_j} \right\rceil C_j.$$

By substituting  $x = x' + \Delta$ , we get

$$x' = B_i^D + (k+1)C_i - F_i + \sum_{j < i} \left\lceil \frac{x' + \Delta}{T_j} \right\rceil C_j.$$

When the greatest common divisor ( $\gcd^{\mathbb{R}^+}$ ) of the periods and computation times of the tasks exists and is equal to  $\delta$ , all task parameters are integral multiples of  $\delta$  (by definition), and  $x'$  will also be an integral multiple of  $\delta$ . Let  $x' = n_{x'} \cdot \delta$  and  $T_j = n_{T_j} \cdot \delta$  for an arbitrary  $j < i$ , where  $n_{x'}, n_{T_j} \in \mathbb{N}^+$ . Now we get

$$\left\lceil \frac{x' + \Delta}{T_j} \right\rceil = \left\lceil \frac{n_{x'} + \frac{\Delta}{\delta}}{n_{T_j}} \right\rceil.$$

Moreover,

$$0 < \frac{\Delta}{\delta} < 1 \Rightarrow \left\lceil \frac{n_{x'} + \frac{\Delta}{\delta}}{n_{T_j}} \right\rceil = \left\lfloor \frac{n_{x'}}{n_{T_j}} \right\rfloor + 1.$$

Hence, when  $\gcd^{\mathbb{R}^+}$  exists and is equal to  $\delta > \Delta$ , the smallest  $x' \in \mathbb{R}^+$  satisfying the recursive equation given above is a solution for both  $WR_i^P(B_i^D + (k+1)C_i - (F_i - \Delta)) - \Delta$  and  $WO_i^P(B_i^D + (k+1)C_i - F_i)$ , which proves the lemma.  $\square$

We finally observe that the analysis presented in Lemma 13 is similar to the revised schedulability analysis for CAN presented in [15]. The latter analysis is an evolutionary improvement of the analysis given by Tindell in [36, 35, 37]. A fixed value for  $\Delta$  is used in [15], corresponding to the transmission time for a single bit on CAN.

## 9 Conclusions

In this paper, we revisited existing worst-case response time analysis of hard real-time tasks under FPDS, arbitrary phasing and relative deadlines at most equal to periods. We showed by means of a number of examples that existing analysis is pessimistic and/or optimistic, both for FPDS as well as for FPNS, being a special case of FPDS. From these examples, we concluded that the worst-case response time of a task is not necessarily assumed for the first job of a task when released at a critical instant. The reason for this is that the final subjob of a task can defer the execution of higher priority tasks, which can potentially give rise to higher interference for subsequent jobs of that task. We observed that González Harbour et al [17] identified the same influence of jobs of a task for relative deadlines at most equal to periods in the context of fixed priority scheduling of periodic tasks with varying execution priority.

We provided revised worst-case response time analysis, resolving the problems with existing approaches. The analysis is based on known concepts of critical instant and busy period for FPPS, for which we gave slightly modified definitions to accommodate for our scheduling model for FPDS. To prevent confusion with existing definitions of busy period, we used the term active period for our definition in this document. We discussed conditions for the termination of an active period, and presented a sufficient condition with a formal proof.

We showed that the critical instant, longest active period, and worst-case response time for a task are suprema rather than maxima for all tasks, except for the lowest priority task, i.e. that instant, period, and response time cannot be assumed. We expressed worst-case response times under FPDS in terms of worst-case response times and worst-case occupied time under FPPS, and presented an iterative procedure to determine worst-case response times under FPDS.

We briefly compared the notion of level- $i$  active period with similar notions in the literature. We concluded that the notions of  $\tau_i$ -busy period in [17], level- $i$  busy period in [16], and level- $\pi_i$  busy interval in [28] are similar to our notion of level- $i$  active period. There are striking differences with the notion of busy period in [25], however. In particular, the level- $n$  busy period never ends for a utilization factor  $U = 1$ . Moreover, we observed that although [22] refers to the notion of busy period from [25] in their description of a method to determine worst-case response times of tasks under FPDS, arbitrary phasing and deadlines larger than periods, their termination condition is actually based on the notion of active period rather than busy period. We also presented uniform, but pessimistic variants of our worst-case response time analysis, and showed that the evolutionary improvement of the analysis for CAN as presented in [15] corresponds to one of these variants.

## Acknowledgements

We thank Alan Burns and Robert I. Davis from the University of York for discussions, and the IST-004527 funded ARTIST 2 Network of Excellence on Embedded Systems Design for making those discussions possible.



## References

- [1] N.C. Audsley, A. Burns, M.F. Richardson, and A.J. Wellings. Hard real-time scheduling: The deadline monotonic approach. In *Proc. 8<sup>th</sup> IEEE Workshop on Real-Time Operating Systems and Software (RTOS)*, pages 133–137, May 1991.
- [2] J.C.M. Baeten and C.A. Middelburg. *Process Algebra with Timing*. Springer, 2002.
- [3] S. Baruah. The limited-preemption uniprocessor scheduling of sporadic systems. In *Proc. 17<sup>th</sup> Euromicro Conference on Real-Time Systems (ECRTS)*, pages 137–144, July 2005.
- [4] S.K. Baruah, L.E. Rosier, and R.R. Howell. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems*, 2(4):301–324, November 1990.
- [5] 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>.
- [6] R.J. Bril. Existing worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption is too optimistic. Technical Report CS 06-05, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven (TU/e), The Netherlands, February 2006.
- [7] R.J. Bril. Existing worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption refuted. In *Proc. Work-in-Progress (WiP) session of the 18<sup>th</sup> Euromicro Conference on Real-Time Systems (ECRTS)*, pages 1–5, July 2006.
- [8] R.J. Bril, J.J. Lukkien, R.I. Davis, and A. Burns. Message response time analysis for ideal controller area network (CAN) refuted. Technical Report CS 06-19, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven (TU/e), The Netherlands, May 2006.
- [9] R.J. Bril, J.J. Lukkien, R.I. Davis, and A. Burns. Message response time analysis for ideal controller area network (CAN) refuted. In *(to appear) Proc. 5<sup>th</sup> International Workshop on Real Time Networks (RTN)*, 2006.
- [10] R.J. Bril, W.F.J. Verhaegh, and J.J. Lukkien. Exact worst-case response times of real-time tasks under fixed-priority scheduling with deferred preemption. In *Proc. Work-in-Progress (WiP) session of the 16<sup>th</sup> Euromicro Conference on Real-Time Systems (ECRTS), Technical Report from the University of Nebraska-Lincoln, Department of Computer Science and Engineering (TR-UNL-CSE-2004-0010)*, pages 57–60, June 2004.
- [11] A. Burns. Preemptive priority based scheduling: An appropriate engineering approach. In S. Son, editor, *Advances in Real-Time Systems*, pages 225–248. Prentice-Hall, 1994.
- [12] A. Burns. Defining new non-preemptive dispatching and locking policies for Ada. In *Proc. 6<sup>th</sup> Ada-Europe International Conference, Lecture Notes in Computer Science (LNCS) 2043*, pages 328–336, May 2001.
- [13] A. Burns and A.J. Wellings. Restricted tasking models. In *Proc. 8<sup>th</sup> International Real-Time Ada Workshop*, pages 27–32, 1997.
- [14] G.C. Buttazzo. *Hard real-time computing systems - predictable scheduling algorithms and applications (2<sup>nd</sup> edition)*. Springer, 2005.
- [15] R.I. Davis, A. Burns, R.J. Bril, and J.J. Lukkien. Controller area network (CAN) schedulability analysis: Refuted, revisited and revised. Technical Report YCS-2006-408, University of York, Department of Computer Science, September 2006. <http://www.cs.york.ac.uk/ftplib/reports/YCS-2006-408.pdf>.
- [16] L. George, N. Rivierre, and M. Spuri. Preemptive and non-preemptive real-time uni-processor scheduling. Technical Report 2966, Institut National de Recherche et Informatique et en Automatique (INRIA), France, September 1996.
- [17] M. González Harbour, M.H. Klein, and J.P. Lehoczky. Fixed-priority scheduling with varying execution priority. In *Proc. 12<sup>th</sup> IEEE Real-Time Systems Symposium (RTSS)*, pages 116–128, December 1991.
- [18] R. Gopalakrishnan and G.M. Parulkar. Bringing real-time scheduling theory and practice closer for multimedia computing. In *Proc. ACM Sigmetrics Conference on Measurement & Modeling of Computer Systems*, pages 1–12, May 1996.
- [19] J.-F. Hermant, L. Leboucher, and N. Rivierre. Real-time fixed and dynamic priority driven scheduling algorithms: theory and practice. Technical Report 3081, Institut National de Recherche et Informatique et en Automatique (INRIA), France, December 1996.
- [20] J. Hooman. *Specification and Compositional Verification of Real-Time Systems*. PhD thesis, Technische Universiteit Eindhoven (TU/e), The Netherlands, May 1991.

- [21] M. Joseph and P. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395, 1986.
- [22] 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.
- [23] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, November 1990.
- [24] S. Lee, C.-G. Lee, M. Lee, S.L. Min, and C.-S. Kim. Limited preemptible scheduling to embrace cache memory in real-time systems. In *Proc. ACM Sigplan Workshop on Languages, Compilers and Tools for Embedded Systems (LCTES), Lecture Notes in Computer Science (LNCS) 1474*, pages 51–64, June 1998.
- [25] J.P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Proc. 11<sup>th</sup> IEEE Real-Time Systems Symposium (RTSS)*, pages 201–209, December 1990.
- [26] 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.
- [27] 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.
- [28] J.W.S. Liu. *Real-Time Systems*. Prentice Hall, 2000.
- [29] A.K. Mok and W.-C. Poon. Non-preemptive robustness under reduced system load. In *Proc. 26<sup>th</sup> IEEE Real-Time Systems Symposium (RTSS)*, pages 200–209, December 2005.
- [30] J.C. Palencia and M. González Harbour. Offset-based response time analysis of distributed systems scheduled under EDF. In *Proc. 15<sup>th</sup> Euromicro Conference on Real-Time Systems (ECRTS'03)*, pages 3–12, July 2003.
- [31] J. Regehr. Scheduling tasks with mixed preemption relations for robustness to timing faults. In *Proc. 23<sup>rd</sup> IEEE Real-Time Systems Symposium (RTSS)*, pages 315–326, December 2002.
- [32] L. Sha, R. Rajkumar, and J.P. Lehoczky. Priority inheritance protocols: an approach to real-time synchronisation. *IEEE Transactions on Computers*, 39(9):1175–1185, September 1990.
- [33] J. Simonson and J.H. Patel. Use of preferred preemption points in cache-based real-time systems. In *Proc. IEEE International Computer Performance and Dependability Symposium (IPDS)*, pages 316–325, April 1995.
- [34] M. Spuri. Analysis of deadline scheduled real-time systems. Technical Report 2772, Institut National de Recherche et Informatique et en Automatique (INRIA), France, January 1996.
- [35] K. Tindell and A. Burns. Guaranteeing message latencies on Controller Area Network (CAN). In *Proc. 1<sup>st</sup> International CAN Conference*, pages 1–11, September 1994.
- [36] K. Tindell, A. Burns, and A.J. Wellings. Calculating controller area network (CAN) message response times. *Control Engineering Practice*, 3(8):1163–1169, August 1995.
- [37] K. Tindell, H. Hansson, and A.J. Wellings. Analysing real-time communications: Controller area network (CAN). In *Proc. 15<sup>th</sup> IEEE Real-Time Systems Symposium (RTSS)*, pages 259–263, December 1994.
- [38] Y. Wand and M. Saksena. Scheduling fixed-priority tasks with preemption threshold. In *Proc. 6<sup>th</sup> International Conference on Real-Time Computing Systems and Applications (RTCISA)*, pages 328–335, December 1999.

## A Auxiliary definitions and lemmas

This appendix presents auxiliary definitions for greatest common divisor and least common multiple for both positive rational numbers and positive real numbers. Moreover, it presents auxiliary lemmas for a strictly increasing function  $f(x)$ .

**Definition 12.** *The greatest common divisor for positive rational numbers ( $\text{gcd}^{\mathbb{Q}^+}$ ) is defined as*

$$\text{gcd}^{\mathbb{Q}^+}\left(\frac{p}{q}, \frac{r}{s}\right) = \frac{\text{gcd}(p \cdot s, q \cdot r)}{q \cdot s} \text{ where } p, q, r, s \in \mathbb{N}^+ \quad (53)$$

$$\text{gcd}^{\mathbb{Q}^+}(r_1, \dots, r_l) = \text{gcd}^{\mathbb{Q}^+}(r_1, \dots, r_{l-2}, \text{gcd}^{\mathbb{Q}^+}(r_{l-1}, r_l)) \text{ for } l \in \mathbb{N} \text{ and } l > 2, \text{ and } r_1, \dots, r_l \in \mathbb{Q}^+ \quad (54)$$

□

**Definition 13.** The least common multiple for positive rational numbers ( $\text{lcm}^{\mathbb{Q}^+}$ ) is defined as

$$\text{lcm}^{\mathbb{Q}^+}(r_1, \dots, r_l) = \frac{\prod_{1 \leq i \leq l} r_i}{\text{gcd}^{\mathbb{Q}^+}(r_1, \dots, r_l)}, \quad (55)$$

where  $l \in \mathbb{N}$  and  $l \geq 2$ , and  $r_1, \dots, r_l \in \mathbb{Q}^+$ . □

Unlike  $\text{gcd}^{\mathbb{Q}^+}$  and  $\text{lcm}^{\mathbb{Q}^+}$ , the greatest common divisor for positive real numbers  $\text{gcd}^{\mathbb{R}^+}$  and the least common multiple for positive real numbers  $\text{lcm}^{\mathbb{R}^+}$  need not exist. We therefore give the following alternative definitions.

**Definition 14.** The least common multiple for positive real numbers ( $\text{lcm}^{\mathbb{R}^+}$ ) is defined as

$$\text{lcm}^{\mathbb{R}^+}(r_1, \dots, r_l) = \min\{r \in \mathbb{R}^+ \mid r = n_1 \cdot r_1 = \dots = n_l \cdot r_l \text{ with } n_1, \dots, n_l \in \mathbb{N}^+\}, \quad (56)$$

where  $l \in \mathbb{N}$  and  $l \geq 2$ , and  $r_1, \dots, r_l \in \mathbb{R}^+$ . □

**Definition 15.** The greatest common divisor for positive real numbers ( $\text{gcd}^{\mathbb{R}^+}$ ) is defined as

$$\text{gcd}^{\mathbb{R}^+}(r_1, \dots, r_l) = \max\{r \in \mathbb{R}^+ \mid n_1 \cdot r = r_1, \dots, n_l \cdot r = r_l \text{ with } n_1, \dots, n_l \in \mathbb{N}^+\}, \quad (57)$$

where  $l \in \mathbb{N}$  and  $l \geq 2$ , and  $r_1, \dots, r_l \in \mathbb{R}^+$ . □

**Lemma 15 (Lemma 4.3 of [5]).** Let  $f(x)$  be defined and strictly non-decreasing in an interval  $[a, b]$  with  $f(a) > a$  and  $f(b) < b$ . Then there exists a value  $c \in (a, b)$  such that  $f(c) = c$ .

*Proof.* See [5]. □

**Lemma 16 (Lemma 4.5 in [5]).** When  $\lim_{x \downarrow X} f(x)$  is defined, and  $f(x)$  is strictly increasing in an interval  $(X, X + \gamma)$  for sufficiently small  $\gamma \in \mathbb{R}^+$ , then the following equation holds.

$$\lim_{x \downarrow X} \lceil f(x) \rceil = \left\lceil \lim_{x \downarrow X} f(x) \right\rceil + 1 \quad (58)$$

*Proof.* See [5]. □

**Lemma 17.** When  $\lim_{x \uparrow X} f(x)$  is defined, and  $f(x)$  is strictly increasing in an interval  $(X - \gamma, X)$  for a sufficiently small  $\gamma \in \mathbb{R}^+$ , then the following equation holds.

$$\lim_{x \uparrow X} \lceil f(x) \rceil = \left\lceil \lim_{x \uparrow X} f(x) \right\rceil \quad (59)$$

*Proof.* The proof uses the definition of limit:

$$\lim_{x \uparrow X} f(x) = Y \Leftrightarrow \forall \varepsilon > 0 \exists \delta > 0 \forall x \in (X - \delta, X) |f(x) - Y| < \varepsilon.$$

We first prove the relation

$$\forall_{X - \gamma < x < X} f(x) < Y,$$

and subsequently prove the lemma.

The proof of the relation is based on a contradiction argument. Because  $\lim_{x \uparrow X} f(x)$  is defined, we may write  $\lim_{x \uparrow X} f(x) = Y$ . Assume  $f(x_1) \geq Y$  for an  $x_1 \in (X - \gamma, X)$ . Choose an  $x_2 \in (x_1, X)$ . Because  $f(x)$  is strictly increasing in  $(X - \gamma, X)$ ,  $f(x_2) > f(x_1) \geq Y$ . Now choose  $\varepsilon = f(x_2) - Y$ , then

$$\forall_{x \in (x_2, X)} f(x) > f(x_2) > Y$$

and hence

$$|f(x) - Y| > |f(x_2) - Y| = \varepsilon,$$

which contradicts the fact that  $\lim_{x \uparrow X} f(x) = Y$ .

For the proof of the lemma, we consider two main cases:  $Y \in \mathbb{Z}$  and  $Y \notin \mathbb{Z}$ . Let  $Y \in \mathbb{Z}$ . According to the relation proven above,  $0 < Y - f(x)$  for all  $x \in (X - \gamma, X)$ . Let  $\varepsilon \in (0, 1]$ . Now there exists a  $\delta_1 \in (0, \gamma)$  such that  $0 < Y - f(x) < \varepsilon \leq 1$  for all  $x \in (X - \delta_1, X)$ , hence  $Y > f(x) > Y - 1$ , i.e.  $\lceil f(x) \rceil = Y = \lceil Y \rceil$ . So,

$$\lim_{x \uparrow X} \lceil f(x) \rceil = \lim_{x \uparrow X} \lceil Y \rceil = \lceil Y \rceil = \left\lceil \lim_{x \uparrow X} f(x) \right\rceil.$$

Next, let  $Y \notin \mathbb{Z}$ . Let  $\varepsilon \in (0, Y - \lfloor Y \rfloor]$ . Now there exists a  $\delta_2 \in (0, \gamma)$  such that for all  $x \in (X - \delta_2, X)$

$$0 < Y - f(x) < \varepsilon \leq Y - \lfloor Y \rfloor,$$

hence

$$Y > f(x) > Y - \varepsilon \geq \lfloor Y \rfloor,$$

i.e.

$$\lceil f(x) \rceil = \lceil Y \rceil.$$

For this second main case we therefore also find

$$\lim_{x \uparrow X} \lceil f(x) \rceil = \lim_{x \uparrow X} \lceil Y \rceil = \lceil Y \rceil = \left\lceil \lim_{x \uparrow X} f(x) \right\rceil,$$

which proves the lemma. □

The proofs of the following two lemmas are similar to the proofs of the previous two lemmas.

**Lemma 18.** *When  $\lim_{x \uparrow X} f(x)$  is defined, and  $f(x)$  is strictly increasing in an interval  $(X - \gamma, X)$  for a sufficiently small  $\gamma \in \mathbb{R}^+$ , then the following equation holds.*

$$\lim_{x \uparrow X} \lfloor f(x) \rfloor = \left\lfloor \lim_{x \uparrow X} f(x) \right\rfloor - 1 \tag{60}$$

□

**Lemma 19.** *When  $\lim_{x \downarrow X} f(x)$  is defined, and  $f(x)$  is strictly increasing in an interval  $(X, X + \gamma)$  for sufficiently small  $\gamma \in \mathbb{R}^+$ , then the following equation holds.*

$$\lim_{x \downarrow X} \lfloor f(x) \rfloor = \left\lfloor \lim_{x \downarrow X} f(x) \right\rfloor \tag{61}$$

□

## B On termination of a level- $n$ active period

In this appendix, we give two examples of task sets with a utilization equal to 1 where the level- $n$  active period does not end upon a simultaneous release of the tasks. For the first example, the least common multiple of the periods does not exist. Hence, the example shows that when Assumption 1 does not hold, the level- $n$  active period need not end. The second example requires an extension of the scheduling model presented in Section 2 with activation (or release) jitter. For this extended model, it illustrates that even when the least common multiple of the periods exists, the level- $n$  active period does not necessarily end for a processor utilization  $U = 1$ .

### B.1 Least common multiple of the periods does not exist

Consider the task set  $\Gamma_8$  with task characteristics as given in Table 8. The utilization  $U$  of  $\Gamma_8$  is equal to  $\frac{1}{2} + \frac{\pi/2}{\pi} = 1$ . Because

	$T_i$	$C_i$	$\phi_i$
$\tau_1$	2	1	0
$\tau_2$	$\pi$	$\frac{\pi}{2}$	0

Table 8. Task characteristics of  $\Gamma_8$ .

the ratio of the periods of the tasks is irrational, the least common multiple of the periods does not exist. We will now show that the following relation holds for the finalization time  $f_{2,k}^P$  of job  $k$  of task  $\tau_2$  under FPPS and a simultaneous release of  $\tau_1$  and  $\tau_2$  at time  $t = 0$

$$(k+1)\pi < f_{2,k}^P < (k+1)\pi + 1 \quad \text{for } k \geq 0. \tag{62}$$

Based on Corollary 7, we therefore conclude that the level-2 active period does not end. Now let  $D_1 = T_1 = 2$  and  $D_2 = \pi + 1$ . Given Equation (62), we derive

$$T_2 = \pi < R_{2,k}^P < \pi + 1 = D_2 \quad \text{for } k \geq 0,$$

and therefore know that the task set is schedulable under FPPS. However, if we try to determine whether or not the task set is schedulable under FPPS by means of the iterative procedure as described in, for example, [22], we find that the procedure does not terminate. This is because the termination condition of the procedure never holds, i.e. the response time of every job of task  $\tau_2$  is smaller than the deadline  $D_2$ , and larger than the period  $T_2$ .

We will now prove Equation (62). Task  $\tau_1$  is executing in the intervals  $[lT_1, lT_1 + C_1) = [ln, ln + 1)$  for  $l \in \mathbb{N}$ , and the finalization time  $f_{2,k}^P$  of job  $k$  of task  $\tau_2$  is therefore in a complementary interval  $[lT_1 + C_1, (l+1)T_1) = [2l+1, 2l+2)$ . Let job  $k$  of  $\tau_2$  complete in the interval  $[2m+1, 2m+2)$  for some  $m \in \mathbb{N}$ , i.e.

$$2m+1 < f_{2,k}^P \leq 2m+2.$$

Because the utilization is 1 and we assume the tasks to be non-idling, there is no idle time in the interval  $[0, f_{2,k}^P)$ . Therefore, the interval  $[0, f_{2,k}^P)$  contains exactly  $m+1$  executions of task  $\tau_1$  and  $k+1$  executions of task  $\tau_2$ , i.e.

$$f_{2,k}^P = (m+1)C_1 + (k+1)C_2 = (m+1) + (k+1)\frac{\pi}{2}.$$

Substituting this latter equation in the former relation yields

$$2m+1 < (m+1) + (k+1)\frac{\pi}{2} \leq 2m+2 \Leftrightarrow m < (k+1)\frac{\pi}{2} \leq m+1.$$

Because  $k, m \in \mathbb{N}$ , we get

$$m+1 > (k+1)\frac{\pi}{2}$$

and therefore

$$f_{2,k}^P = (m+1) + (k+1)\frac{\pi}{2} > (k+1)\pi.$$

Moreover, because  $m < (k+1)\frac{\pi}{2}$ , we derive

$$f_{2,k}^P = (m+1) + (k+1)\frac{\pi}{2} < (k+1)\pi + 1.$$

Together, these latter two relations for  $f_{2,k}^P$  prove Equation (62).

## B.2 Activation jitter

With *activation* (or *release*) *jitter*, the releases of a task  $\tau_i$  do not take place strictly periodically, with period  $T_i$ , but we assume they take place somewhere in an interval of length  $AJ_i$  that is repeated with period  $T_i$ . More specifically, the activations  $a_{ik}$  satisfy

$$\varphi_i^{lwb} + kT_i \leq a_{ik} \leq \varphi_i^{upb} + kT_i,$$

for some  $\varphi_i^{lwb}, \varphi_i^{upb} \in \mathbb{R}^+ \cup \{0\}$  with  $\varphi_i^{upb} - \varphi_i^{lwb} = AJ_i$ . Consider task set  $\Gamma_9$  with task characteristics as given in Table 9. The

	$T_i$	$C_i$	$AJ_i$
$\tau_1$	4	2	1
$\tau_2$	4	2	0

Table 9. Task characteristics of  $\Gamma_9$ .

least common multiple of the periods  $T_1$  and  $T_2$  is given by  $\text{lcm}(T_1, T_2) = 4$ . Figure 11 shows the activations for task  $\tau_1$  and  $\tau_2$ , i.e.  $a_{1,0} = AJ_1 = 1$ ,  $a_{1,l} = (l+1)T_1$  for  $l \in \mathbb{N}$ , and  $a_{2,0} = 1$ , and the processor pending load  $P(t)$ . These activations correspond to a critical instant for task  $\tau_2$  for FPPS and FPDS. For this example, the pending load is periodic, i.e.  $P(t+4) = P(t)$  for  $t > 1$ . Because  $P(t) > 0$  for  $t > 1$ , the level-2 active period never ends. As a consequence, the worst-case response time of  $\tau_2$  cannot be determined by means of an iterative procedure in which the response times of all activations in the level-2 active period are considered, irrespective of the scheduling algorithm. Hence, the common approach to determine the worst-case response time for  $\tau_2$  under FPPS, FPDS, and EDF [34] does not work.

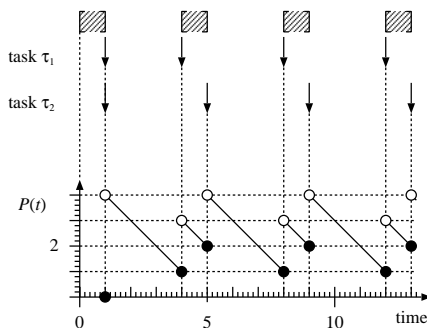


Figure 11. Activations for  $\Gamma_9$  and processor pending load  $P(t)$ .

Without proof, we merely state that the worst-case length  $WL_n$  of the level- $n$  active period under arbitrary phasing and activation jitter is given by the smallest  $x \in \mathbb{R}^+$  satisfying the following equation

$$x = \sum_{j \leq n} \left\lceil \frac{x + AJ_j}{T_j} \right\rceil C_j,$$

where  $AJ_j$  is the activation jitter of task  $\tau_j$ . As mentioned in [30], there exists a positive solution for this recursive equation when  $U < 1$ . The proof of this latter claim is similar to the proof of Lemma 2 on page 16.

Figure 12 shows timelines for  $\Gamma_9$  under FPPS, FPDS, and EDF. The figure illustrates that  $\Gamma_9$  is schedulable under FPDS

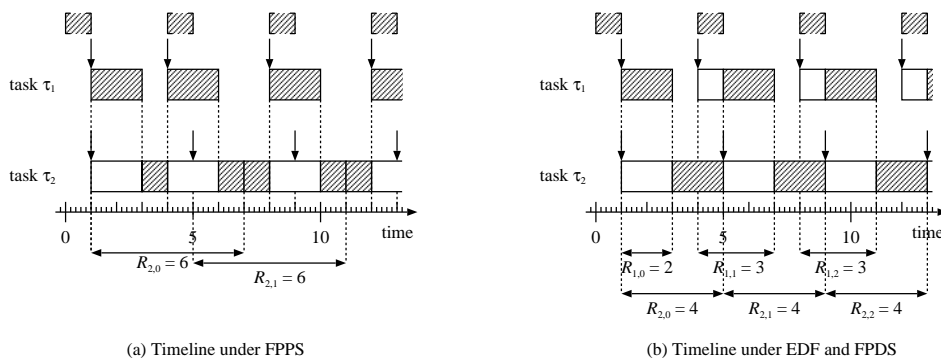


Figure 12. Timelines for  $\Gamma_9$  under FPPS, FPDS, and EDF with release jitter and a simultaneous release of both tasks at time  $t = 1$ .

and EDF for the given activations. Moreover, the schedule is periodic, i.e.  $\sigma(t + 4) = \sigma(t)$  for  $t \geq 1$ .  $\Gamma_9$  is also schedulable under FPPS when the deadline  $D_2 \geq 6$  for task  $\tau_2$ . Under FPPS, the schedule is also periodic, i.e.  $\sigma(t + 4) = \sigma(t)$  for  $t \geq 3$ . Because the schedule is periodic, the worst-case response time of task  $\tau_2$  can be determined by considering the response times of all jobs of  $\tau_2$  in a ‘sufficiently long’ interval, e.g. similar to the approach described in [26].