

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

Reinder J. Bril · Johan J. Lukkien ·
Wim F.J. Verhaegh

Published online: 28 April 2009

© The Author(s) 2009. This article is published with open access at Springerlink.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 a 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. Hence, that instant, period, and response time cannot be assumed for any task, except for the lowest priority task. 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

Excerpts of this document have been published as Bril et al. (2007).

R.J. Bril (✉) · J.J. Lukkien

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

J.J. Lukkien

e-mail: j.j.lukkien@tue.nl

W.F.J. Verhaegh

Philips Research Laboratories, High Tech Campus 11, 5656 AE, Eindhoven, The Netherlands
e-mail: wim.verhaegh@philips.com

analysis of the other tasks. These anomalies for the lowest priority task are an immediate consequence of the fact that only the lowest priority task cannot be blocked. 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, illustrates the revised analysis for an advanced model for FPDS, where tasks are structured as flow graphs of subjobs rather than sequences, and shows that our analysis is sustainable.

Keywords Level-*i* active period · Level-*i* busy period · Worst-case response time · Worst-case occupied time · Periodic tasks · Fixed-priority scheduling · Deferred preemption · Real-time systems

1 Introduction

1.1 Motivation

Based on the seminal paper of Liu and Layland (1973), 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 (Sha et al. 1990). 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 (Burns and Wellings 1997; Gopalakrishnan and Parulkar 1996; Lee et al. 1998; Mok and Poon 2005; Simonson and Patel 1995). 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 (Burns 1994), 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 (Bril et al. 2004; Burns 1994; Burns and Wellings 1997; Lee et al. 1998). The existing analysis is not exact, however. In Bril et al. (2004), it has already been shown that the analysis presented in Burns (1994), Burns and Wellings (1997), Lee et al. (1998) is pessimistic. More recently, it has been shown in Bril (2006) that the analysis presented in Bril et al. (2004), Burns (1994), Burns and Wellings (1997) 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 Bril et al. (2006), 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 networks (CAN) (Tindell and Burns 1994; Tindell et al. 1994, 1995). Revised

analysis for CAN resolving the problem with the original approach in an evolutionary fashion can be found in Davis et al. (2007).

1.2 Contributions

This paper resolves the problems with the existing approaches by presenting a 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 (Baruah et al. 1990b), 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 Baeten and Middelburg (2002), Hooman (1991), Koymans (1990). The scheduling model is based on a partitioning of the timeline in a set of non-empty, right semi-open intervals (Buttazzo 2005; Hooman 1991). The analysis is based on the concepts of *critical instant* (Liu and Layland 1973) and *busy period* (Lehoczky 1990). 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* in this document, for which we give a formal definition.

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. Hence, that instant, period, and response time cannot be assumed for any task, except for the lowest priority task. 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. These anomalies for the lowest priority task are an immediate consequence of the fact that, unlike the other tasks, the lowest priority task cannot be blocked. 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 (Bril 2004) under FPPS. We also present pessimistic variants of the analysis, which are indeed uniform for all tasks, and show that the revised analysis for CAN presented in Davis et al. (2007) conforms to a pessimistic variant. We illustrate our analysis for an advanced model for FPDS, where tasks are structured as flow graphs of subjobs rather than sequences, and we show that our analysis for FPDS is sustainable (Baruah and Burns 2006) and therefore also applicable for sporadic task systems (Mok 1983; Baruah et al. 1990a).

1.3 Related work

Next to continuous scheduling models, one can find discrete scheduling models in the literature, e.g. in George et al. (1996), Hermant et al. (1996), and models in which domains are not explicitly specified (Buttazzo 2005; Klein et al. 1993; Liu 2000). 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 Liu (2000) 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 we present a novel worst-case response time analysis to determine schedulability of tasks under FPDS. Similarly, George et al. (1996) assume 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 (2005) 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 (1990) 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. (1991) 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 (Wang and Saksena 1999) failed to identify this issue. The resulting flaw was later corrected by Regehr (2002). Worst-case response time analysis of tasks under EDF and relative deadlines at most equal to periods described by Spuri (1996) is also based on the concept of busy period.

1.4 Structure

This paper has the following structure. First, in Sect. 2, we present basic real-time scheduling models for FPPS and FPDS, in which we assume fixed values for computation times and periods. These assumptions ease the presentation in subsequent sections, and will be lifted in Sect. 8. Next, worst-case analysis for FPPS is briefly recapitulated in Sect. 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 Sect. 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 Sect. 6. We present a theorem for critical instant and theorems to determine the worst-case response time of a task under FPDS, arbitrary phasing, and arbitrary relative deadlines. Section 7 illustrates the worst-case response time analysis by applying it to some examples presented in Sect. 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 for FPDS. Moreover, this section illustrates the revised analysis for an advanced model for FPDS and briefly discusses the sustainability of the analysis, i.e. we show that our schedulability analysis remains valid if we replace fixed computation times by worst-case computation times and fixed periods by minimal inter-arrival times (or worst-case periods) of tasks in our real-time scheduling models. The paper is concluded in Sect. 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 τ_i is being executed, and a release occurs for a higher priority task τ_j , then the execution of τ_i is preempted, and will resume when the execution of τ_j has ended, as well as all other releases of tasks with a higher priority than τ_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 $\sigma : \mathbb{R} \rightarrow \{0, 1, \dots, n\}$, where $\sigma(t) = i$ with $i > 0$ means that task τ_i is being executed at time t , while $\sigma(t) = 0$ means that the processor is idle. More specifically, we define $\sigma(t)$ as a right-continuous and piece-wise constant function, i.e. σ partitions the timeline in a set of non-empty, right semi-open intervals $\{[t_j, t_{j+1})\}_{j \in \mathbb{Z}}$. 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 $\sigma(t)$.

Each task τ_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* $\varphi_i \in \mathbb{R}^+ \cup \{0\}$. In this paper, we assume arbitrary deadlines, i.e. the deadline of a task may exceed its period. An *activation* (or *release*) *time* is a time at which a task τ_i becomes ready for execution. A release of a task is also termed a *job*. The first job of task τ_i is released at time φ_i and is referred to as *job zero*. The release of job k of τ_i therefore takes place at time $a_{ik} = \varphi_i + kT_i, k \in \mathbb{N}$. The (*absolute*) *deadline* of job k of τ_i takes place at $d_{ik} = a_{ik} + D_i$. The *begin time* b_{ik} and *finalization* (or

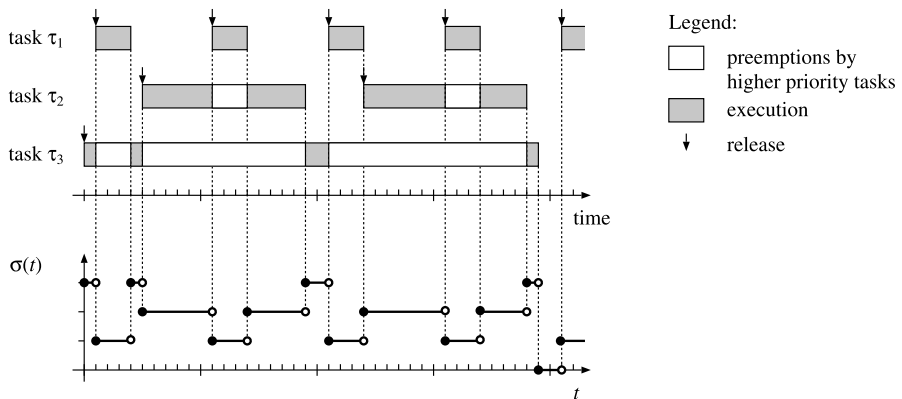


Fig. 1 An example of the execution of a set \mathcal{T} of three independent periodic tasks τ_1, τ_2 , and τ_3 , where task τ_1 has highest priority, and task τ_3 has lowest priority, and the corresponding value of $\sigma(t)$

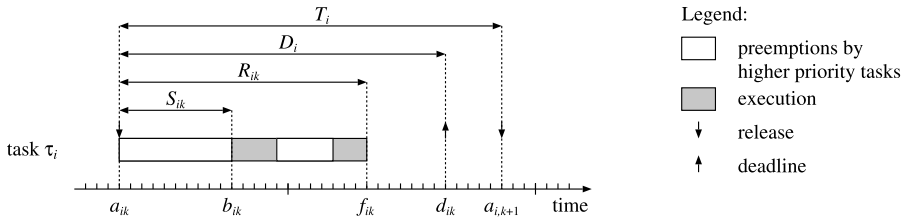


Fig. 2 Basic model for task τ_i

completion) time f_{ik} of job k of τ_i is the time at which τ_i actually starts and ends the execution of that job, respectively. The set of phasings φ_i is termed the phasing φ of the task set \mathcal{T} .

The active (or response) interval of job k of τ_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 τ_i is defined as the length of its active interval, i.e. $R_{ik} = f_{ik} - a_{ik}$. Similarly, the begin (or start) interval is defined as $[a_{ik}, b_{ik})$, and the start time S_{ik} as $S_{ik} = b_{ik} - a_{ik}$. Figure 2 illustrates the above basic notions for an example job of task τ_i .

The worst-case response time WR_i and the worst-case start time of a task τ_i are the largest response time and the largest start time of any of its jobs, respectively, i.e.

$$WR_i = \sup_{\varphi, k} R_{ik}(\varphi), \tag{1}$$

$$WS_i = \sup_{\varphi, k} S_{ik}(\varphi). \tag{2}$$

Note that the response time R_{ik} and the start time S_{ik} have been parameterized in these equations to denote their dependency on the phasing φ .

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 $C \in \mathbb{R}^+$. We will therefore use a functional notation when needed, e.g. $WR_i(C)$. 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 φ , 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 (Joseph and Pandya 1986; Klein et al. 1993; Liu and Layland 1973). We also adopt other standard basic assumptions (Liu and Layland 1973), 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 τ_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 at or before its deadline. Hence, a set \mathcal{T} of n periodic tasks

can be scheduled if and only if

$$WR_i \leq D_i \quad (3)$$

for all $i = 1, \dots, n$. For notational convenience, we assume that the tasks are given in order of decreasing priority, i.e. task τ_1 has highest priority and task τ_n has lowest priority.

The (*processor*) *utilization factor* U is the fraction of the processor time spent on the execution of the task set (Liu and Layland 1973). The fraction of processor time spent on executing task τ_i is C_i/T_i , and is termed the *utilization factor* U_i^τ of task τ_i , i.e.

$$U_i^\tau = \frac{C_i}{T_i}. \quad (4)$$

The *cumulative utilization factor* U_i for tasks τ_1 till τ_i is the fraction of processor time spent on executing these tasks, and is given by

$$U_i = \sum_{j \leq i} U_j^\tau. \quad (5)$$

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

$$U = U_n = \sum_{j \leq n} U_j^\tau = \sum_{j \leq n} \frac{C_j}{T_j}. \quad (6)$$

In Liu and Layland (1973), the following necessary condition is determined for the schedulability of a set \mathcal{T} of n periodic tasks under any scheduling algorithm:

$$U \leq 1. \quad (7)$$

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 Sect. 2.1. Each job of task τ_i is now assumed to consist of a sequence of m_i subjobs. The k th subjob of τ_i is characterized by a computation time $C_{ik} \in \mathbb{R}^+$, where $C_i = \sum_{k=1}^{m_i} C_{ik}$. For convenience, we will use the term F_i to denote the computation time C_{i,m_i} of the final subjob of τ_i , i.e.

$$F_i = C_{i,m_i}. \quad (8)$$

We assume that subjobs are non-preemptable. Hence, a task can only be preempted at subjob boundaries, i.e. at so-called *preemption points*. A task can therefore *defer* the preemption and execution of a higher priority task. We will use the term *blocking* of a task τ_i to denote the time that the execution of τ_i is deferred by lower priority tasks. 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 τ_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 absolute release time of job k of task τ_i . 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 a 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 τ_i at or after time t , i.e.

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

Here, the notation x^+ stands for $\max(x, 0)$, which is used to indicate that there are no releases of τ_i before time φ_i . Because $\varphi_i \geq 0$, the term $(\lceil \frac{t - \varphi_i}{T_i} \rceil)^+$ is equal to the number of releases of τ_i in $[0, t)$. 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 τ_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 τ_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 τ_1 and τ_2 with characteristics as given in Table 1.

Figure 3 shows an example of the execution of the tasks τ_1 and τ_2 under FPPS. Note that even an infinitesimal increase of the computation time of either task τ_1 or τ_2 will immediately cause the job of task τ_2 released at time 0 to miss its deadline at time 7.

Table 1 Task characteristics of \mathcal{T}_1

	$T_i = D_i$	C_i
τ_1	5	2
τ_2	7	3

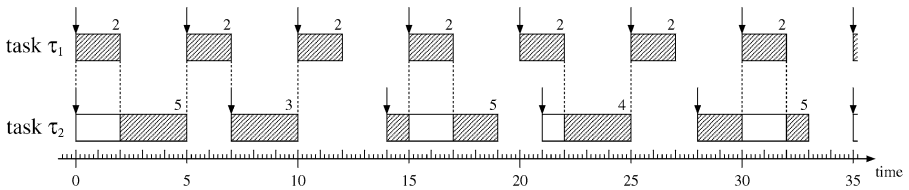


Fig. 3 Timeline for T_1 under FPPS with a simultaneous release of both tasks at time zero. The numbers to the top right corner of the boxes denote the response times of the respective releases

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 Bril (2004), most of these results were already known; see for example Audsley et al. (1991), Joseph and Pandya (1986), Liu and Layland (1973). Auxiliary lemmas on which the proofs of these theorems and theorems in subsequent sections are based are included in the Appendix.

Theorem 1 (Bril 2004, Theorem 4.1) *In order to have a maximal response time for an execution k of task τ_i , i.e. to have $f_{ik} - a_{ik} = WR_i$, we may assume without loss of generality that the phasing φ 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 τ_i coincides with the simultaneous release for all higher priority tasks. This latter point in time is called a critical instant for task τ_i .*

Given this theorem, we conclude that time 0 in Fig. 3 is a critical instant for both task τ_1 and τ_2 . From this figure, we therefore derive that the worst-case response times of tasks τ_1 and τ_2 are 2 and 5, respectively. The next theorems can be used to determine the worst-case response times analytically.

Theorem 2 (Bril 2004, Theorem 4.2)¹ *The worst-case response time WR_i of a task τ_i is given by the smallest $x \in \mathbb{R}^+$ that satisfies the following equation, provided that x is at most T_i :*

$$x = C_i + \sum_{j < i} \left\lceil \frac{x}{T_j} \right\rceil C_j. \tag{9}$$

Theorem 3 (Bril 2004, Theorem 4.3)¹ *The worst-case response time WR_i of task τ_i can be found by the following iterative procedure:*

$$WR_i^{(0)} = C_i, \tag{10}$$

¹The theorems in Bril (2004, Chap. 4) are based on worst-case computation times of tasks. Because our basic model is based on fixed computation times, we replaced WC_i and WC_j in the original equation by C_i and C_j , respectively. We consider worst-case computation times in Sect. 8.5.

$$WR_i^{(l+1)} = C_i + \sum_{j < i} \left\lceil \frac{WR_i^{(l)}}{T_j} \right\rceil C_j, \quad l = 0, 1, \dots \quad (11)$$

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 Fig. 3, task τ_2 is preempted at time 15 due to a release of task τ_1 , and resumes its execution at time 17. The span of time from a task τ 's release till the moment in time that τ can start its execution or resume its execution after completion of a computation time C is termed *occupied time*. The *worst-case occupied time (WO)* of a task τ is the largest occupied time of any of its jobs. In Bril (2004), it has been shown that the worst-case occupied time can be described in terms of the worst-case response time by taking a limit from above,² i.e.

$$WO_i(C) = \lim_{x \downarrow C} WR_i(x). \quad (12)$$

Considering Fig. 3, we derive that worst-case occupied times $WO_2(0)$ and $WO_2(3)$ of task τ_2 are equal to 2 and 7, respectively. Note that the worst-case occupied time $WO_2(0)$ of task τ_2 is the longest possible span of time from the release of τ_2 till the moment in time that τ_2 can start its execution, i.e. $WO_2(0)$ of τ_2 is the *worst-case start time* WS_2 of τ_2 . Hence, unlike worst-case response times, worst-case occupied times are also defined for a computation time equal to zero. Further note that $WR_2(C_2) = 5 < WO_2(C_2) = 7$. The next theorems can be used to determine the worst-case occupied times analytically.

Theorem 4 (Bril 2004, Theorem 4.4)¹ *When the smallest positive solution of (9) for a computation time C_i' is at most D_i , the worst-case occupied time WO_i of a task τ_i for a computation time $C_i \in [0, C_i']$ is given by the smallest non-negative $x \in \mathbb{R}$ that satisfies*

$$x = C_i + \sum_{j < i} \left(\left\lfloor \frac{x}{T_j} \right\rfloor + 1 \right) C_j. \quad (13)$$

Theorem 5 (Bril 2004, Theorem 4.5)¹ *The worst-case occupied time WO_i of task τ_i can be found by the following iterative procedure.*

$$WO_i^{(0)} = \begin{cases} \sum_{j < i} C_j & \text{for } C_i = 0, \\ WR_i & \text{for } C_i > 0; \end{cases} \quad (14)$$

²A limit taken from above is sometimes also denoted as $\lim_{x \rightarrow C+}$; see Weisstein (2003). When a function $f(x)$ is defined for a value $x = x_0$ and the limit from above (or right-hand limit) of $f(x)$ in x_0 is equal to $f(x_0)$, f is termed right (-hand) continuous at $x = x_0$.

$$WO_i^{(l+1)} = C_i + \sum_{j < i} \left(\left\lfloor \frac{WO_i^{(l)}}{T_j} \right\rfloor + 1 \right) C_j, \quad l = 0, 1, \dots \quad (15)$$

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 (13) by starting from (12) and subsequently using Lemma 16.

Similarly to (12), we can express WR_i in terms of WO_i by taking a limit from below,³ i.e.

$$WR_i(C) = \lim_{x \uparrow C} WO_i(x). \quad (16)$$

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

$$WR_i(C) = \lim_{x \uparrow C} WR_i(x), \quad (17)$$

$$WO_i(C) = \lim_{x \downarrow C} WO_i(x). \quad (18)$$

Lemmas related to these latter three equations can be found in the [Appendix](#).

4 Existing response time analysis for FPDS refuted

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 Burns (1994), Burns and Wellings (1997). We include a recapitulation of the analysis for FPNS as presented in Tindell et al. (1994). 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).

³A limit taken from below is sometimes also denoted as $\lim_{x \rightarrow C^-}$; see Weisstein (2003). When a function $f(x)$ is defined for a value $x = x_0$ and the limit from below (or left-hand limit) of $f(x)$ in x_0 is equal to $f(x_0)$, f is termed left (-hand) continuous at $x = x_0$.

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* $B_i^D \in \mathbb{R}^+ \cup \{0\}$ of task τ_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 τ_i . This blocking time is given by

$$B_i^D = \begin{cases} \max_{j>i} \max_{1 \leq k \leq m_j} C_{j,k} & \text{for } i < n, \\ 0 & \text{for } i = n. \end{cases} \tag{19}$$

Strictly spoken, this blocking time is a supremum (and not a maximum) for all tasks, except for the lowest priority task, i.e. that value cannot be assumed for $i < n$.

The worst-case response time \widetilde{WR}_i^D of a task τ_i under FPDS, arbitrary phasing, and deadlines less than or equal to periods, as presented in Burns (1994) and Burns and Wellings (1997), is given by

$$\widetilde{WR}_i^D = WR_i^P(B_i^D + C_i - (F_i - \Delta)) + (F_i - \Delta), \tag{20}$$

where WR_i^P denotes the worst-case response time of τ_i under FPPS. According to Burns and Wellings (1997), Δ is an arbitrary small positive value needed to ensure that the final subjob has actually started. Hence, when task τ_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 Joseph and Pandya (1986) given in Tindell et al. (1994) to take account of tasks being non-preemptive. Next, we show that the update is very similar to the analysis for FPDS as given by (20).

The non-preemptive nature of tasks may cause blocking of a task by at most one lower priority task. The *maximum blocking* $B_i^N \in \mathbb{R}^+ \cup \{0\}$ of task τ_i by a lower priority task is equal to the longest computation time of a task with a priority lower than task τ_i , i.e.

$$B_i^N = \begin{cases} \max_{j>i} C_j & \text{for } i < n, \\ 0 & \text{for } i = n. \end{cases} \tag{21}$$

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

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

$$\widetilde{WR}_i^N = w_i + C_i, \tag{22}$$

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

$$x = B_i^N + \sum_{j<i} \left\lceil \frac{x + \tau_{res}}{T_j} \right\rceil C_j. \tag{23}$$

In this latter equation, τ_{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 similar to the existing analysis for FPDS. To this end, we substitute $w_i = w'_i - \tau_{res}$, $x = x' - \tau_{res}$, and $\tau_{res} = \Delta$ in (22) and (23). 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

$$\widetilde{WR}_i^N = WR_i^P(B_i^N + \Delta) + (C_i - \Delta). \tag{24}$$

Because we have $F_i = C_i$ and $B_i^D = B_i^N$ for FPNS, (24) for FPNS is similar to (20) for FPDS. There is an aspect requiring further attention, however. In particular, (20) is based on an arbitrary small positive value Δ whereas the analysis for FPNS is based on the resolution τ_{res} with which time is measured. We will return to this issue in Sect. 8.3.

4.2 Existing analysis is pessimistic

Consider the set \mathcal{T}_2 consisting of three tasks with characteristics as described in Table 2. Based on (20) we derive

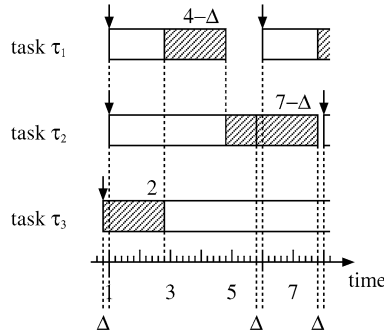
$$\begin{aligned} \widetilde{WR}_2^D &= WR_2^P(B_2^D + C_2 - (F_2 - \Delta)) + (F_2 - \Delta) \\ &= WR_2^P(2 + 3 - (2 - \Delta)) + (2 - \Delta) \\ &= WR_2^P(3 + \Delta) + (2 - \Delta) = 7 + \Delta + (2 - \Delta) = 9. \end{aligned}$$

However, the existing analysis does not take into account that τ_i can only be blocked by a subjob of a lower priority task if that subjob starts *before* the simultaneous release of τ_i and all tasks with a higher priority than τ_i . This aspect can be taken into account in the analysis by replacing B_i^D in (20) by $(B_i^D - \Delta)^+$. The notation x^+ stands for $\max\{x, 0\}$, which is used to indicate that the blocking time can not become negative

Table 2 Task characteristics of \mathcal{T}_2

	T_i	D_i	C_i
τ_1	5	4	2
τ_2	7	7	1 + 2
τ_3	30	30	2 + 2

Fig. 4 Timeline for τ_2 under FPDS with a release of tasks τ_1 and τ_2 at time $t = 1$ and a release of task τ_3 at time $t = 1 - \Delta$



for the lowest priority task. The worst-case response time of τ_2 now becomes $7 - \Delta$, as illustrated in Fig. 4. For $\Delta \downarrow 0$, we therefore find a supremum (and not a maximum) equal to 7 for the worst-case response time of τ_2 . As a result, the existing analysis is pessimistic.

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 Burns (1994, 2001), Burns and Wellings (1997) and in Bril et al. (2004) is very similar. In particular, the worst-case response time \widetilde{WR}_2^D of task τ_2 is determined by looking at the response time of the first job of task τ_2 upon a simultaneous release with task τ_1 . However, the worst-case response time of task τ_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 3. Note that the utilization factor U of this set of tasks \mathcal{T}_3 is given by $U = \frac{2}{3} + \frac{4.5}{7} > 1$. Hence, the task set is not schedulable. Based on (20), we derive

$$\begin{aligned} \widetilde{WR}_2^D &= WR_2^P(B_2 + C_2 - (F_2 - \Delta)) + (F_2 - \Delta) \\ &= WR_2^P(0 + 4.5 - (3 - \Delta)) + (3 - \Delta) \\ &= WR_2^P(1.5 + \Delta) + (3 - \Delta) = 3.5 + \Delta + (3 - \Delta) = 6.5. \end{aligned}$$

This value corresponds with the response time of the first job of task τ_2 upon a simultaneous release with task τ_1 , as illustrated in Fig. 5. However, the same figure also illustrates that the second job of τ_2 misses its deadline. Stated in other words, the existing worst-case response time analysis is optimistic.

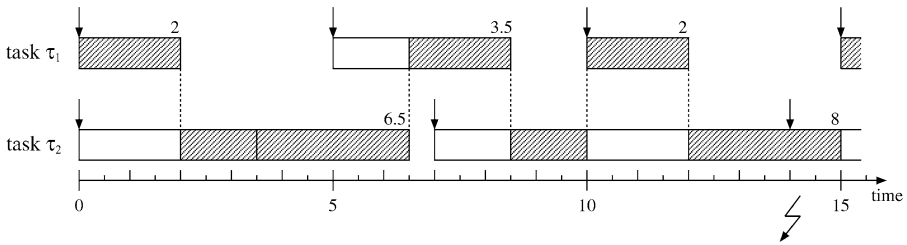


Fig. 5 Timeline for \mathcal{T}_3 under FPDS with a simultaneous release of both tasks at time zero

Table 3 Task characteristics of \mathcal{T}_3

	$T_i = D_i$	C_i
τ_1	5	2
τ_2	7	1.5 + 3

Table 4 Task characteristics of \mathcal{T}_4

	$T_i = D_i$	C_i
τ_1	5	2
τ_2	7	2 + 2.1

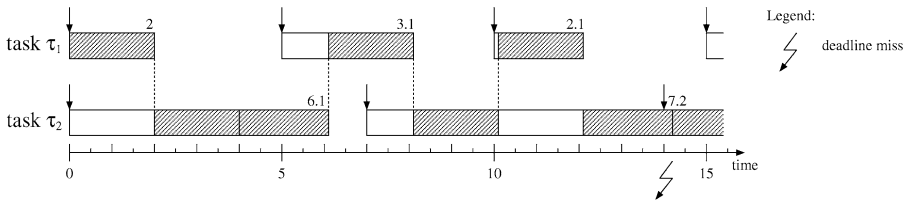


Fig. 6 Timeline for \mathcal{T}_4 under FPDS with a simultaneous release of all tasks at time zero

4.3.2 An example with $U < 1$

Another 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 $U = \frac{2}{5} + \frac{4.1}{7} < 1$. Hence, the task set could be schedulable. Applying (20) yields $\widehat{WR}_2^D = 6.1$, which corresponds with the response time of the first job of task τ_2 upon a simultaneous release with task τ_1 ; see Fig. 6. However, the same figure also illustrates that the second job of task τ_2 misses its deadline.

4.3.3 An example with $U = 1$

Consider task set \mathcal{T}_5 given in Table 5. The utilization factor U of this set of tasks is given by $U = \frac{2}{5} + \frac{4.5}{7} = 1$. The task set is not schedulable by FPPS, as we showed in Sect. 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

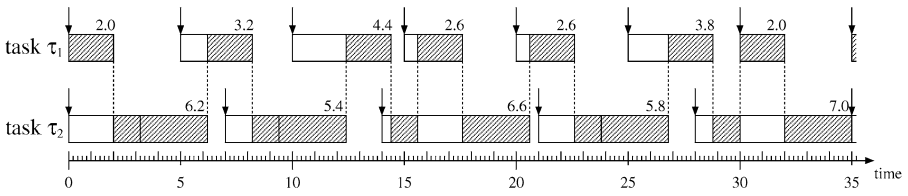


Fig. 7 Timeline for \mathcal{T}_5 under FPDS with a simultaneous release of all tasks at time zero

Table 5 Task characteristics of \mathcal{T}_5

	$T_i = D_i$	C_i
τ_1	5	2
τ_2	7	1.2 + 3

Table 6 Task characteristics of \mathcal{T}_6

	T_i	C_i
τ_1	5	2
τ_2	7	1.2
τ_3	7	3

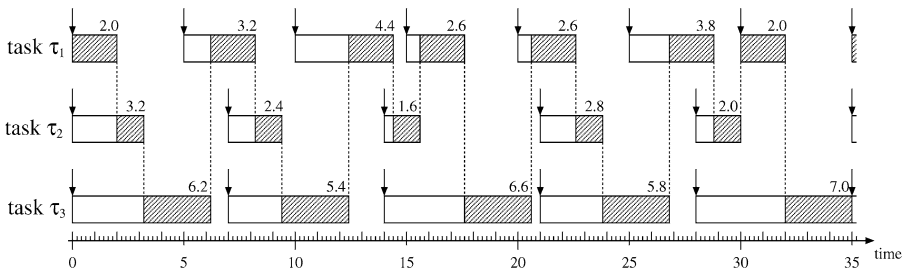


Fig. 8 Timeline for \mathcal{T}_6 under FPNS with a simultaneous release of all tasks at time zero. The numbers to the top right corner of the boxes denote the response times of the respective releases

release at time zero in an interval of length 35, i.e. equal to the *hyperperiod* of the tasks. Applying (20) yields $\widetilde{WR}_2^D = 6.2$, which corresponds with the response time of the first job of task τ_2 in Fig. 7. However, the response time of the 5th job of task τ_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}_6 given in Table 6, which is similar to task set \mathcal{T}_5 given in Table 5, except for the fact that rather than having a second subjob for task τ_2 it has a task τ_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 (20) yields $\widetilde{WR}_3^D = 6.2$, which corresponds to the response time of the first job of task τ_3 in Fig. 8. However, the response time of

the 5th job of task τ_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 τ_i can defer the execution of higher priority tasks, which can potentially give rise to higher interference for subsequent jobs of task τ_i . This problem can therefore arise for all tasks, except for the highest priority task. González Harbour et al. (1991) identified the same influence of jobs of a task for relative deadlines at most equal to periods in the context of FPPS of periodic tasks with varying execution priority.

Considering Fig. 7, we see that every job of task τ_2 in the interval $[0, 26.8)$ defers the execution of a job of task τ_1 . Moreover, that deferred job of task τ_1 subsequently gives rise to additional interference for the next job of task τ_2 . This situation ends when the job of τ_2 is started at time $t = 28$, i.e. the 5th job of τ_2 does not defer the execution of a job of τ_1 . Viewed in a different way, we may state that the active intervals of the jobs of tasks τ_1 and τ_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 τ_i and jobs of tasks with a higher priority than task τ_i . Hence, the active interval of every job of a task τ_i is contained in a level- i active period. To determine the worst-case response time of a task τ_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 τ_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 ε -critical instant.

5 Active period

This section presents a formal definition of a *level- i active period*, which is based on the notion of *pending load*, and theorems to determine the length of a level- i active period. As mentioned before, a level- i active period may contain multiple jobs of τ_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, a level- (i, k) active period is a *smallest* interval that only contains $(k + 1)$ entire active intervals of jobs⁴ of task τ_i and entire active intervals of jobs of tasks with a higher priority than task τ_i .

⁴The jobs of task τ_i in a level- (i, k) active period are numbered from 0 (zero) to k , giving rise to $(k + 1)$ jobs.

These notions and theorems form the basis for the worst-case analysis for FPDS in the next section.

We start with the definition of the notion level- i active period in Sect. 5.1. Next, we provide examples of level- i active periods in Sect. 5.2. The length of a level- i active period is the topic of Sect. 5.3. We refine the notion of level- i active period to level- (i, k) active period in Sect. 5.4, and conclude with a theorem to determine its length in Sect. 5.4.3.

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 τ_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 time* and the *finalization time* of that job, respectively.

As described in Sect. 2.1, the *active interval* of job k of task τ_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 task τ_i that are released before time t , i.e.

$$P_i^\tau(t) = \left(\left\lceil \frac{t - \varphi_i}{T_i} \right\rceil \right)^+ \cdot C_i - \int_0^t \sigma_i^\tau(t') dt', \tag{25}$$

where

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

The term $(\lceil \frac{t - \varphi_i}{T_i} \rceil)^+ \cdot C_i$ in (25) is equal to the amount of processing that needs to be performed due to releases of task τ_i in $[0, t)$. The term $\int_0^t \sigma_i^\tau(t') dt'$ is equal to the amount of processing that has been performed for τ_i . The right-hand side of (25) is therefore equal to the amount of processing at time t due to releases of jobs of task τ_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 τ_j with $j \leq i$ that are released before time t , i.e.

$$P_i(t) = \sum_{j \leq i} P_j^\tau(t) = \sum_{j \leq i} \left(\left\lceil \frac{t - \varphi_j}{T_j} \right\rceil \right)^+ \cdot C_j - \int_0^t \sigma_i(t') dt', \tag{26}$$

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.

$$P(t) = P_n(t). \quad (27)$$

Corollary 1 The order in which the tasks τ_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 depends on blocking due to a lower priority task. As an example, let $P_i(t_s) = 0$, then $P_i(t) = C_k$ for all $t \in (t_s, t'_s)$ under FPDS if the following three conditions hold:

- A task τ_k with $s \leq i$ is released at time t_s .
- No other releases of τ_j for $j \leq i$ take place in $[t_s, t'_s)$.
- A subjob of a lower priority task is executing at time t_s and blocks task τ_k 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, provided that the algorithm is non-idling.

Note that the notions of pending load, cumulative pending load, and processor pending load are independent of relative deadlines of tasks.

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 all $t \in (t_s, t_e)$.

Let the *blocking time* $B_i(t_s)$ of a level- i active period that starts at time t_s be defined as the length of the (optionally empty) initial interval during which the tasks τ_j with $j \leq i$ are blocked by a subjob of a task with a lower priority. Note that $B_n(t_s) = 0$ and $0 \leq B_i(t_s) < B_i^D$ for $i < n$.

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 τ_j with $j \leq i$ are continuously executing in $[t_s, t_e)$, except for an (optionally empty) initial interval $[t_s, t_s + B_i(t_s))$ 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 $B_i(t_s) + C_k$, where a task τ_k with $k \leq i$ is released at time t_s .*
- (iii) *The order in which the tasks τ_j with $j \leq i$ are executed is immaterial for the length $L_i(t_s)$.*

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

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

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

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, provided that the algorithm is non-idling.*

Note that the notion of level- i active period is independent of relative deadlines of tasks. Further note that a level- i active period may, but need not, contain activations of task τ_i . In the sequel, we will call a level- i active period that contains an activation of task τ_i a *proper* level- i active period. Similarly, we call a level- i active period that does not contain an activation of τ_i an *improper* level- i active period. Unless explicitly stated otherwise, we use the phrase ‘level- i active period’ to denote a proper 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 Fig. 3 for \mathcal{T}_1 and one for FPDS based on the timeline in Fig. 7 for \mathcal{T}_5 .

Consider Fig. 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 τ_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 τ_2 is preempted by a job of task τ_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 $[28, 33)$. From these eight level-2 active periods, $[0, 5)$, $[7, 10)$,

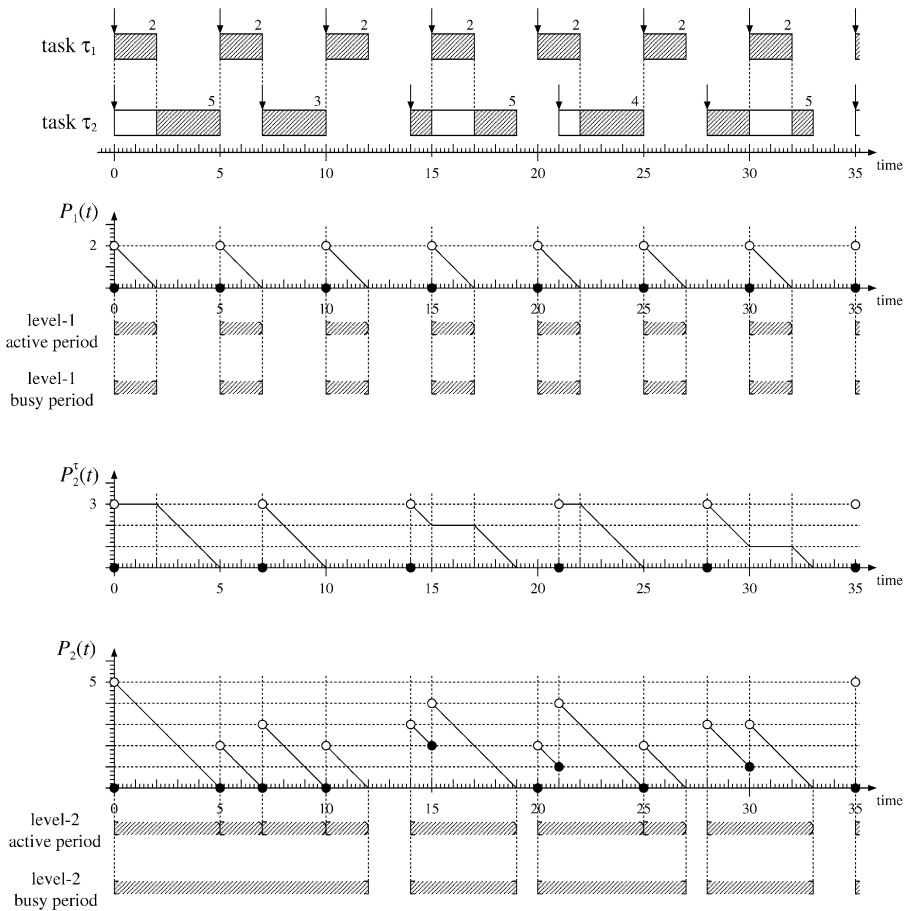


Fig. 9 Timeline for \mathcal{T}_1 under FPPS, pending loads $P_1(t)$, $P_2^\tau(t)$, and $P_2(t)$, and level- i active periods and level- i busy periods. From the eight level-2 active periods in the interval $[0, 35)$, five are proper, i.e. $[0, 5)$, $[7, 10)$, $[14, 19)$, $[20, 25)$, and $[28, 33)$ contain activations of task τ_2 . The other three are improper, i.e. $[5, 7)$, $[10, 12)$, and $[25, 27)$

$[14, 19)$, $[20, 25)$, and $[28, 33)$ are proper, i.e. contain activations of task τ_2 , and $[5, 7)$, $[10, 12)$, and $[25, 27)$ are improper. As mentioned before, the level-2 active period only depends on the activations of τ_1 and τ_2 , and is independent of the scheduling algorithm.

Consider Fig. 10, with a timeline for \mathcal{T}_5 under FPDS, pending loads $P_1(t)$, $P_2^\tau(t)$, 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 τ_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 τ_1 is blocked by a subjob of task τ_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. $[0, 35)$.

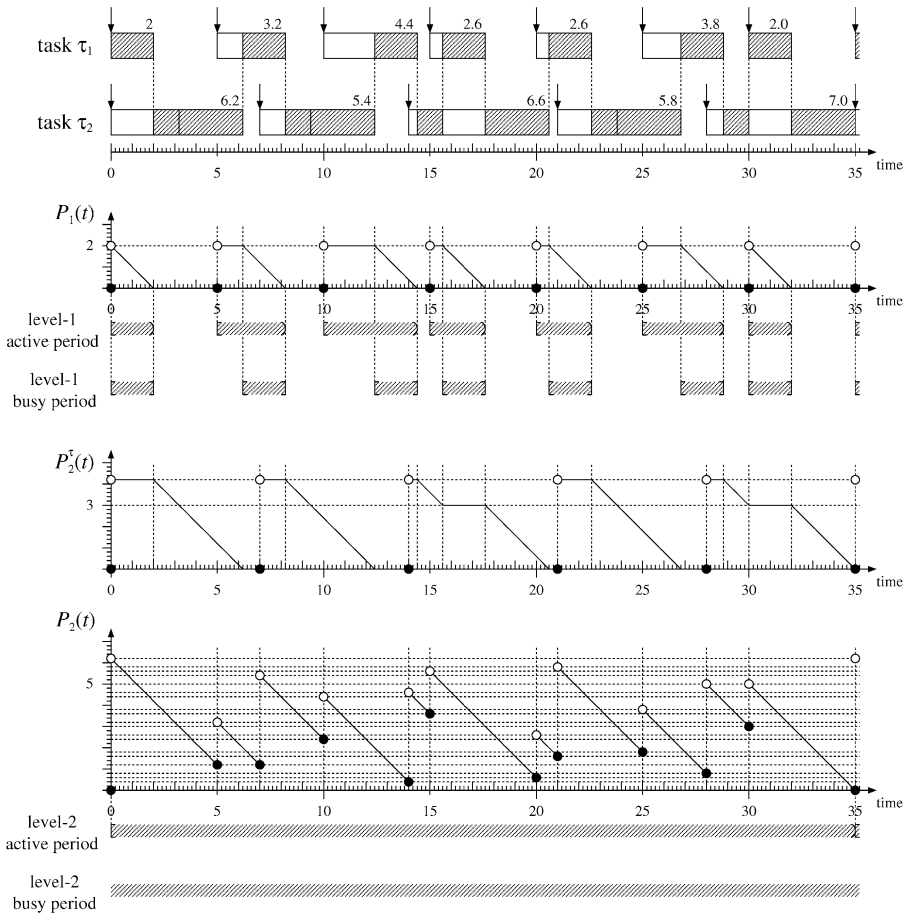


Fig. 10 Timeline for \mathcal{T}_5 under FPDS, pending loads $P_1(t)$, $P_2^i(t)$, and $P_2(t)$, and level- i active periods and level- i busy periods

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 it starts. Because we assume $\varphi_i \geq 0$ 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 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.

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*

$$x = B_i(t_s) + \sum_{j \leq i} \left(\left\lceil \frac{x - \varphi_j(t_s)}{T_j} \right\rceil \right)^+ \cdot C_j. \tag{28}$$

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 $P_i(t) = 0$, and the length $L_i(t_s)$ of the active period becomes $t_e - t_s$. We now derive (28) from $P_i(t_e) = 0$:

$$\begin{aligned} P_i(t_e) &= \{(26)\} \sum_{j \leq i} \left(\left\lceil \frac{t_e - \varphi_j}{T_j} \right\rceil \right)^+ \cdot C_j - \int_0^{t_e} \sigma_i(t) dt \\ &= P_i(t_s) + \sum_{j \leq i} \left(\left\lceil \frac{t_e - (t_s + \varphi_j(t_s))}{T_j} \right\rceil \right)^+ \cdot C_j - \int_{t_s}^{t_e} \sigma_i(t) dt \\ &= \{P_i(t_s) = 0\} \sum_{j \leq i} \left(\left\lceil \frac{t_e - (t_s + \varphi_j(t_s))}{T_j} \right\rceil \right)^+ \cdot C_j - \int_{t_s}^{t_e} \sigma_i(t) dt \\ &= 0. \end{aligned}$$

From Lemma 1, we derive that the lower priority task is executing in $[t_s, t_s + B_i(t_s))$, and only tasks τ_j with $j \leq i$ are executing in $[t_s + B_i(t_s), t_e)$. Hence, we conclude that

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

Substituting this result in the former equation, we get

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

and by subsequently substituting $t_e = x + t_s$, we get (28). 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 (28), 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 (28) if Assumption 1 holds. To that end, we will use Lemma 4.3 from Bril (2004) (see Lemma 15 in the Appendix), and first prove two lemmas.

Lemma 2 *There exists a positive solution for the recursive equation (28) 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 Brill (2004). Let f be defined as

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

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

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

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

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

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

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

As $U_i < 1$, the relation

$$x \geq B_i(t_s) + xU_i + \sum_{j \leq i} C_j$$

holds for

$$x \geq \frac{B_i(t_s) + \sum_{j \leq i} C_j}{1 - U_i}.$$

We now choose

$$b = \frac{B_i(t_s) + \sum_{j \leq i} C_j}{1 - U_i},$$

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

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

such that $x = f(x)$. □

Lemma 3 *There exists a positive solution for the recursive equation (28) 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 $B_n(t_s) = 0$ 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 $B_n(t_s) = 0$ and $\varphi_i(t_s) = 0$ for all i the value $x = \text{lcm}(T_1, \dots, T_n)$ is a solution of (28). For these values of $B_n(t_s)$ and $\varphi_i(t_s)$, (28) simplifies to

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

Because $\lceil \frac{\text{lcm}(T_1, \dots, T_n)}{T_j} \rceil C_j = \text{lcm}(T_1, \dots, T_n) \frac{C_j}{T_j}$ 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 the least common multiple 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 $f(x) < \sum_{j \leq n} \lceil \frac{x}{T_j} \rceil C_j$, 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 $f(x) = \sum_{j \leq i} \lceil \frac{x}{T_j} \rceil C_j$, we found a (positive) solution and we are also done. □

Theorem 7 *If 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. □

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 a task τ_k at time t_s . If Assumption 1 holds, the length $L_i(t_s)$ of that level- i active period can be found by the following iterative procedure:*

$$L_i^{(0)}(t_s) = B_i(t_s) + C_k, \quad (29)$$

$$L_i^{(l+1)}(t_s) = B_i(t_s) + \sum_{j \leq i} \left(\left\lceil \frac{L_i^{(l)}(t_s) - \varphi_j(t_s)}{T_j} \right\rceil \right)^+ \cdot C_j, \quad l = 0, 1, \dots \quad (30)$$

Proof From Lemmas 2 and 3, we know that there exists a positive solution of (28) when Assumption 1 holds. To prove the theorem, 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 (28). 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 $L_i^{(0)}(t_s) = B_i(t_s) + C_k > 0$, and

$$\begin{aligned} L_i^{(1)}(t_s) &= B_i(t_s) + \sum_{j \leq i} \left(\left\lceil \frac{L_i^{(0)}(t_s) - \varphi_j(t_s)}{T_j} \right\rceil \right)^+ \cdot C_j \\ &\geq \{\varphi_s(t_s) = 0\} B_i(t_s) + C_k = L_i^{(0)}(t_s). \end{aligned}$$

Next, if $L_i^{(l+1)}(t_s) \geq L_i^{(l)}(t_s)$, then we can conclude from (30) 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 (30) 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 $L_i^{(0)}(t_s) = B_i(t_s) + C_k \leq L_i(t_s)$. Next, if $L_i^{(l)}(t_s)$ is a lower bound on $L_i(t_s)$, then

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

is a lower bound on the amount of processing that needs to be performed due to releases of task τ_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 $B_i(t_s) + C_k$ and from above by the solution. This means that each factor $(\lceil \frac{L_i^{(l)}(t_s) - \varphi_j(t_s)}{T_j} \rceil)^+$ in (30) can only take on a finite number of values. Combining this for all higher priority tasks τ_j , we can conclude that the right-hand side of (30) can only take on a finite number of values. \square

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. We assume in this section that Assumption 1 holds.

5.4.1 A refinement of pending load

Let a level-*i* active period start at time t_s . As described above, the length of that period is given by the smallest $x > 0$ satisfying (28). Let the length of that period be $L_i(t_s)$. The number of jobs $l_i(t_s)$ of task τ_i in that period is now given by

$$l_i(t_s) = \left\lceil \frac{L_i(t_s) - \varphi_i(t_s)}{T_i} \right\rceil. \tag{31}$$

We now refine our notion of pending load $P_i(t)$ by considering only the first $k + 1 \leq l_i(t_s)$ jobs of τ_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 $t_s < t$ and ends at time $t_e \geq t$ is the amount of processing at time t that still needs to be performed for at most the first $k + 1 \leq l_i(t_s)$ jobs of τ_i and the jobs of tasks τ_j with $j < i$ that are released in $[t_s, t)$, i.e.

$$P_{ik}(t) = \min \left(\left(\left\lceil \frac{t - (t_s + \varphi_i(t_s))}{T_i} \right\rceil \right)^+, k + 1 \right) \cdot C_i + \sum_{j < i} \left(\left\lceil \frac{t - (t_s + \varphi_j(t_s))}{T_j} \right\rceil \right)^+ \cdot C_j - \int_{t_s}^t \sigma_i(t') dt', \tag{32}$$

with $\sigma_i(t)$ as defined in Definition 3. At the start of a level-*i* active period and outside level-*i* active periods, $P_{ik}(t)$ is equal to zero.

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 $t \in (t_s, t_e)$.

5.4.3 Length of a level-(i, k) active period

Theorem 9 Let the number of jobs of task τ_i in a level-*i* active period that starts at time t_s be given by $l_i(t_s)$. The length $L_{ik}(t_s)$ of that level-(*i, k*) active period with

$0 \leq k < l_i(t_s)$ is the smallest $x \in \mathbb{R}^+$ satisfies the following equation

$$x = B_i(t_s) + (k + 1)C_i + \sum_{j < i} \left(\left\lceil \frac{x - \varphi_j(t_s)}{T_j} \right\rceil \right)^+ \cdot C_j. \tag{33}$$

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, arbitrary phasing, and arbitrary relative deadlines, and theorems to determine the worst-case response times analytically. We assume in this section that Assumption 1 holds. Moreover, we consider an arbitrary level- i active period with a start at time t_s . As described in Sect. 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 (1), the worst-case response time WR_i^D of a task τ_i under FPDS is the largest response time under arbitrary phasing, i.e.

$$WR_i^D = \sup_{\varphi, k} R_{ik}(\varphi).$$

We can refine this equation by taking blocking of tasks and the notion of level- i active period into account. In particular, we observe that all active intervals of jobs of task τ_i are contained in level- i active periods. Assuming the start of an arbitrary level- i active period at time t_s , the worst-case response time WR_i^D of task τ_i can therefore be described as

$$WR_i^D = \sup_{B'_i, \varphi'_1, \dots, \varphi'_i} \max_{0 \leq k < l'_i(B'_i, \varphi'_1, \dots, \varphi'_i)} R'_{ik}(B'_i, \varphi'_1, \dots, \varphi'_i), \tag{34}$$

where l'_i is the number of jobs of task τ_i in that level- i active period.

We will now first present a lemma to determine the response time of job k of task τ_i in a level- i active period. We subsequently present a theorem which states that given an infinitesimal time $\varepsilon > 0$, the maximum response time of task τ_i is assumed in a level- i active period which starts at an ε -critical instant. A next theorem refines (34).

Lemma 4 *The response time R'_{ik} of job k of task τ_i in a level- i active period that starts at time t_s with $0 \leq k < l'_i$ and l'_i the number of jobs of task τ_i in that level- i active period is given by*

$$R'_{ik}(B'_i, \varphi'_1, \dots, \varphi'_i) = b'_{ik, m_i}(B'_i, \varphi'_1, \dots, \varphi'_{i-1}) + F_i - (kT_i + \varphi'_i), \tag{35}$$

where 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

$$x = B'_i + (k + 1)C_i - F_i + \sum_{j < i} \left(\left\lfloor \frac{x - \phi'_j}{T_j} \right\rfloor + 1 \right)^+ \cdot C_j. \tag{36}$$

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 τ_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'_i .
- 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 τ_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 τ_i . Stated in other words, the final subjob of job k of task τ_i can start for the smallest $t_s + x \geq t_s + \max(B'_i, a'_{ik})$ for which $\lim_{t \downarrow t_s+x} P_{ik}(t) = F_i$. We now derive

$$\begin{aligned} \lim_{t \downarrow t_s+x} P_{ik}(t) &= \{(32)\} \lim_{t \downarrow t_s+x} \left(\min \left(\left(\left\lfloor \frac{t - (t_s + \phi'_i)}{T_i} \right\rfloor \right)^+, k + 1 \right) \cdot C_i \right. \\ &\quad \left. + \sum_{j < i} \left(\left\lfloor \frac{t - (t_s + \phi'_j)}{T_j} \right\rfloor \right)^+ \cdot C_j - \int_{t_s}^t \sigma_i(t') dt' \right) \\ &= \min \left(\left(\lim_{t \downarrow t_s+x} \left\lfloor \frac{t - (t_s + \phi'_i)}{T_i} \right\rfloor \right)^+, k + 1 \right) \cdot C_i \\ &\quad + \sum_{j < i} \left(\lim_{t \downarrow t_s+x} \left\lfloor \frac{t - (t_s + \phi'_j)}{T_j} \right\rfloor \right)^+ \cdot C_j - \int_{t_s}^{t_s+x} \sigma_i(t') dt' \\ &= \{\text{Lemma 16}\} \min \left(\left(\left\lfloor \frac{x - \phi'_i}{T_i} \right\rfloor + 1 \right)^+, k + 1 \right) \cdot C_i \\ &\quad + \sum_{j < i} \left(\left\lfloor \frac{x - \phi'_j}{T_j} \right\rfloor + 1 \right)^+ \cdot C_j - \int_{t_s}^{t_s+x} \sigma_i(t') dt' \\ &= \{x \geq \max(B'_i, \phi'_i + k \cdot T_i)\} (k + 1) \cdot C_i \\ &\quad + \sum_{j < i} \left(\left\lfloor \frac{x - \phi'_j}{T_j} \right\rfloor + 1 \right)^+ \cdot C_j - (x - B'_i) \\ &= F_i. \end{aligned}$$

The relative begin time $b'_{ik,m_i}(B'_i, \varphi'_1, \dots, \varphi'_{i-1})$ is therefore the smallest non-negative $x \in \mathbb{R}$ satisfying the following equation:

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

The relative completion time f'_{ik} of job k of τ_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.

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

Theorem 10 *Given an infinitesimal time $\varepsilon > 0$, the maximum response time of task τ_i under FPDS and arbitrary phasing is assumed when the level- i active period is started at an ε -critical instant, i.e. when τ_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 ε before that simultaneous release.*

Proof Let $R'_i(B'_i, \varphi'_1, \dots, \varphi'_i)$ denote $\max_{0 \leq k < l'_i(B'_i, \varphi'_1, \dots, \varphi'_i)} R'_{ik}(B'_i, \varphi'_1, \dots, \varphi'_i)$. We will prove that $R'_i(B'_i, \varphi'_1, \dots, \varphi'_i)$ assumes a maximum for $\varphi'_j = 0$ with $j \leq i$ and $B'_i = (B_i^D - \varepsilon)^+$. Hence, the maximum is assumed when τ_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 τ_i .
- Every job of τ_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 τ_i in the level- i active period, and the response time R'_{ik} as a function of the relative phasing φ'_j with $j \leq i$ and the blocking time B'_i . Consider (28) for the length L'_i of a level- i active period. The term $\lceil \frac{x - \varphi'_j}{T_j} \rceil$ in that equation is a strictly non-increasing function of φ'_j with $j \leq i$. Because $\varphi'_j \geq 0$, a maximum of that term is assumed for $\varphi'_j = 0$. Moreover, the right-hand side of (28) is a strictly increasing function of B'_i , and the length L'_i is therefore also a strictly increasing function of B'_i . The largest value of L'_i is found for the largest value of B'_i under consideration, i.e. for $B'_i = (B_i^D - \varepsilon)^+$. As a consequence, L'_i assumes a maximum for $\varphi'_j = 0$ for all $j \leq i$ and $B'_i = (B_i^D - \varepsilon)^+$.

Given the behavior of L'_i and (31), we conclude that the number of jobs l'_i of task τ_i in the level- i active period is a strictly non-increasing function of φ'_j with $j \leq i$ and a strictly non-decreasing function of B'_i . As a consequence, l'_i assumes a maximum for $\varphi'_j = 0$ for all $j \leq i$ and $B'_i = (B_i^D - \varepsilon)^+$.

From (35), we conclude that $R'_{ik}(B'_i, \varphi'_1, \dots, \varphi'_i)$ is a strictly decreasing function of φ'_i . Because $\varphi'_i \geq 0$, a maximum is assumed for $\varphi'_i = 0$. Now consider (36) for the relative begin time b'_{ik,m_i} . The term $\lfloor \frac{x-\varphi_j}{T_j} \rfloor$ in that equation is a strictly non-increasing function of φ'_j . Similarly to $\varphi'_j \geq 0$, a maximum of that term is therefore assumed for $\varphi'_j = 0$. Hence, $b'_{ik,m_i}(B'_i, 0, \dots, 0)$ dominates $b'_{ik,m_i}(B'_i, \varphi'_1, \dots, \varphi'_{i-1})$ for all values of B'_i and all values of φ'_j with $j < i$. Moreover, the right-hand side of (36) is a strictly increasing function of B'_i , and $b'_{ik,m_i}(B'_i, 0, \dots, 0)$ is therefore also a strictly increasing function of B'_i . The largest value of $b'_{ik,m_i}(B'_i, 0, \dots, 0)$ is found for the largest value of B'_i under consideration, i.e. for $B'_i = (B_i^D - \varepsilon)^+$. As a consequence, $R'_{ik}(B'_i, \varphi'_1, \dots, \varphi'_i)$ also assumes a maximum for $\varphi'_j = 0$ for all $j \leq i$ and $B'_i = (B_i^D - \varepsilon)^+$.

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

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

$$WR_i^D = \lim_{\varepsilon \downarrow 0} \max_{0 \leq k < l'_i((B_i^D - \varepsilon)^+, 0, \dots, 0)} R'_{ik}((B_i^D - \varepsilon)^+, 0, \dots, 0). \tag{37}$$

Proof Once again, let $R'_i(B'_i, \varphi'_1, \dots, \varphi'_i)$ denote

$$\max_{0 \leq k < l'_i((B_i^D - \varepsilon)^+, 0, \dots, 0)} R'_{ik}((B_i^D - \varepsilon)^+, 0, \dots, 0).$$

From the proof of Theorem 10, we derive that $R'_i(B'_i, 0, \dots, 0)$ dominates $R'_i(B'_i, \varphi'_1, \dots, \varphi'_i)$ for all values of B'_i and all values of φ'_j with $j \leq i$, i.e.

$$WR_i^D = \sup_{B'_i, \varphi'_1, \dots, \varphi'_i} R'_i(B'_i, \varphi'_1, \dots, \varphi'_i) = \sup_{B'_i} R'_i(B'_i, 0, \dots, 0).$$

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

$$WR_i^D = \sup_{B'_i} R'_i(B'_i, 0, \dots, 0) = \lim_{\varepsilon \downarrow 0} R'_i((B_i^D - \varepsilon)^+, 0, \dots, 0),$$

which proves the theorem. \square

In the sequel, we will omit trailing zeros in the parameter list, e.g. we write $R'_{ik}((B_i^D - \varepsilon)^+)$ when $\varphi'_j = 0$ for $j \leq i$.

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 tasks, except for the lowest priority task, i.e. that value cannot be assumed for $i < n$.*

Corollary 5 *A critical instant is a supremum for all tasks, except for the lowest priority task, i.e. that instant cannot be assumed for $i < n$.*

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 present definitions and prove 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 τ_i in a level- i active period, and the worst-case response time WR_{ik}^D of job k of task τ_i .

Definition 8 The worst-case length WL_i^D of level- i active period under FPDS is the largest length of that period under arbitrary phasing, i.e.

$$WL_i^D = \sup_{B'_i, \varphi'_1, \dots, \varphi'_i} L'_i(B'_i, \varphi'_1, \dots, \varphi'_i). \tag{38}$$

Definition 9 The worst-case number wl_i^D of jobs of task τ_i in a level- i active period under FPDS is the largest number in that period under arbitrary phasing, i.e.

$$wl_i^D = \sup_{B'_i, \varphi'_1, \dots, \varphi'_i} l'_i(B'_i, \varphi'_1, \dots, \varphi'_i). \tag{39}$$

Definition 10 The worst-case response time WR_{ik}^D of job k of task τ_i , with $1 \leq k < wl_i^D$, under FPDS is the largest response time of job k of τ_i under arbitrary phasing, i.e.

$$WR_{ik}^D = \sup_{B'_i, \varphi'_1, \dots, \varphi'_i} R'_{ik}(B'_i, \varphi'_1, \dots, \varphi'_i). \tag{40}$$

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*

$$x = B_i^D + \sum_{j \leq i} \left\lceil \frac{x}{T_j} \right\rceil C_j. \tag{41}$$

Proof The term $\lceil \frac{x - \varphi'_j}{T_j} \rceil$ in (28) is a strictly non-increasing function of φ'_j with $j \leq i$. Because $\varphi'_j \geq 0$, a maximum of that term is assumed for $\varphi'_j = 0$. Now let $L'_i(B'_i)$ denote the length of a level- i active period with $i \leq n$ for a simultaneous release of task τ_i with all tasks with a higher priority. Hence, $L'_i(B'_i)$ is the smallest $x \in \mathbb{R}^+$

satisfying (28) with $\phi'_j = 0$, i.e. the smallest $x \in \mathbb{R}^+$ satisfying

$$x = B'_i + \sum_{j \leq i} \left\lceil \frac{x}{T_j} \right\rceil C_j. \tag{42}$$

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 (41) by substituting $B'_i = 0$ in (42) for $i = n$.

$\{i < n\}$ The right-hand side of (42) is a strictly increasing function of B'_i , and $L'_i(B'_i)$ is therefore also a strictly increasing function of B'_i . The largest value for $L'_i(B'_i)$ is found for the largest value of $B'_i < B_i^D$. Hence, WL_i^D is given by

$$WL_i^D = \lim_{B'_i \uparrow B_i^D} L'_i(B'_i). \tag{43}$$

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

$$\begin{aligned} WL_i^D &= \{(42)\} \lim_{B'_i \uparrow B_i^D} \left(B'_i + \sum_{j \leq i} \left\lceil \frac{L'_i(B'_i)}{T_j} \right\rceil C_j \right) \\ &= B_i^D + \sum_{j \leq i} \lim_{B'_i \uparrow B_i^D} \left\lceil \frac{L'_i(B'_i)}{T_j} \right\rceil C_j \\ &= \{\text{Lemma 17}\} B_i^D + \sum_{j \leq i} \left\lceil \lim_{B'_i \uparrow B_i^D} \frac{L'_i(B'_i)}{T_j} \right\rceil C_j \\ &= \{(43)\} B_i^D + \sum_{j \leq i} \left\lceil \frac{WL_i^D}{T_j} \right\rceil C_j. \end{aligned}$$

Hence, the worst-case length WL_i^D is the smallest $x \in \mathbb{R}^+$ satisfying (41), which proves the lemma. □

Because B_i^D is a supremum (and not a maximum) for all tasks, except for 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 tasks, except for the lowest priority task, i.e. that value cannot be assumed for $i < n$.*

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

$$wl_i^D = \left\lceil \frac{WL_i^D}{T_i} \right\rceil. \tag{44}$$

Proof We first derive (44) and subsequently prove that wl_i^D is a maximum.

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

$$wl_i^D = \lim_{B'_i \uparrow B_i^D} l'_i(B'_i). \tag{45}$$

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

$$\begin{aligned} \lim_{B'_i \uparrow B_i^D} l'_i(B'_i) &= \lim_{B'_i \uparrow B_i^D} \left\lceil \frac{L'_i(B'_i)}{T_i} \right\rceil \\ &= \{\text{Lemma 17}\} \left\lceil \lim_{B'_i \uparrow B_i^D} \frac{L'_i(B'_i)}{T_i} \right\rceil \\ &= \{(43)\} \left\lceil \frac{WL_i^D}{T_i} \right\rceil. \end{aligned}$$

Equation (44) immediately follows from (45) and this latter equation.

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

$$l'_i(B_i^D) = \lim_{B'_i \uparrow B_i^D} l'_i(B'_i), \tag{46}$$

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

$$\forall B'_i \quad \forall B'_i \in (B_i^D - \gamma, B_i^D] \quad l'_i(B'_i) = wl_i^D.$$

To prove that $l'_i(B'_i)$ 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 $L'_i(B'_i)$ exists if 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 $l'_i(B'_i)$ 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 a limit:

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

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

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

Let $\varepsilon \in (0, 1]$. Now there exists a $\delta \in (0, B_i^D)$ such that $0 \leq wl_i^D - l'_i(B'_i) < \varepsilon \leq 1$ for all $B'_i \in (B_i^D - \delta, B_i^D]$, hence $wl_i^D \geq l'_i(B'_i) > wl_i^D - 1$. Because $wl_i^D, l'_i(B'_i) \in \mathbb{N}$, 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 (37), i.e.

$$WR_i^D = \max_{0 \leq k < wl_i^D} \lim_{\varepsilon \downarrow 0} R'_{ik}((B_i^D - \varepsilon)^+). \tag{47}$$

Hence, WR_{ik}^D is given by

$$WR_{ik}^D = \lim_{\varepsilon \downarrow 0} R'_{ik}((B_i^D - \varepsilon)^+). \tag{48}$$

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

$$WR_{ik}^D = \begin{cases} WR_i^P(B_i^D + (k + 1)C_i - F_i) + F_i - kT_i & \text{for } i < n, \\ WO_n^P((k + 1)C_n - F_n) + F_n - kT_n & \text{for } i = n, \end{cases} \tag{49}$$

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 τ'_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 (48), we derive

$$\begin{aligned} WR_{ik}^D &= \lim_{\varepsilon \downarrow 0} R'_{ik}((B_i^D - \varepsilon)^+) \\ &= \{(35)\} \lim_{\varepsilon \downarrow 0} (b'_{ik,m_i}((B_i^D - \varepsilon)^+) + F_i - kT_i) \\ &= \lim_{\varepsilon \downarrow 0} b'_{ik,m_i}((B_i^D - \varepsilon)^+) + F_i - kT_i, \end{aligned}$$

where $b'_{ik,m_i}((B_i^D - \varepsilon)^+)$ denotes the relative begin time of the final subjob of job k of task τ_i with $0 \leq k < wl_i$ and $\varphi'_j = 0$ for $j \leq i$ as given in (36). Hence, $b'_{ik,m_i}((B_i^D - \varepsilon)^+)$ is the smallest $x \in \mathbb{R}^+$ satisfying

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

Now let task set \mathcal{T}' be identical to \mathcal{T} except for the characteristics of task τ_i , i.e. τ'_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 τ'_i of \mathcal{T}' misses its deadline under FPPS and arbitrary phasing if and only if job k of task τ_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 $B_n^D = 0$, and immediately arrive at (49), which proves the lemma for $i = n$.

For $i < n$, we derive

$$\begin{aligned} WR_{ik}^D &= \lim_{\varepsilon \downarrow 0} WO_i^P \left((B_i^D - \varepsilon)^+ + (k + 1)C_i - F_i \right) + F_i - kT_i \\ &= \{(16)\} WR_i^P \left(B_i^D + (k + 1)C_i - F_i \right) + F_i - kT_i, \end{aligned}$$

which proves the lemma for $i < n$. □

Note that because the lowest priority task is the only task that cannot be blocked, i.e. $B_n^D = 0$, the worst-case response time analysis for FPDS is not uniform for all tasks. Moreover, note that WR_{ik}^D is a supremum (and not a maximum) for all tasks, except for the lowest priority task, i.e. that value cannot be assumed for $i < n$.

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

$$WR_i^D = \max_{0 \leq k < wl_i^D} WR_{ik}^D. \tag{50}$$

Proof The theorem follows immediately from (47) and (48), 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 τ_i under FPDS and arbitrary phasing. The procedure is stopped when the worst-case response time WR_{ik}^D of job k for task τ_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*

$$x = B_i^D + (k + 1)C_i + \sum_{j < i} \left\lceil \frac{x}{T_j} \right\rceil C_j. \tag{51}$$

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 for $i < n$.

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

$$WL_{ik}^D = WR_i^P(B_i^D + (k + 1)C_i), \tag{52}$$

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 τ'_i with a computation time $C'_i = B_i^D + (k + 1)C_i$, a period $T'_i = kT_i + D_i$ and a deadline $D'_i = T'_i$.

Proof The lemma follows from the similarity between (9) and (51). The period and deadline of task τ'_i have been chosen to be equal to the deadline of job $k + 1$ of task τ_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 τ'_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 τ'_i represents the executions of both the blocking lower priority task as well as task τ_i , all executions of the corresponding jobs also have a completion in that interval. Hence, the level- i active period that started with an ε -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. □

Theorem 13 *The worst-case response time WR_i^D of a task τ_i can be found by the following iterative procedure under Assumption 1, using (49):*

$$WR_i^{(0)} = WR_{i,0}^D, \tag{53}$$

$$WR_i^{(l+1)} = \max(WR_i^{(l)}, WR_{i,l+1}^D), \quad l = 0, 1, \dots \tag{54}$$

The procedure is stopped when the worst-case response time WR_{ik}^D of job k of task τ_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 Lemma 10 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 w_i^D of jobs of task τ_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 τ_i i.e. when the task set is not schedulable. \square

Corollary 7 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.

Corollary 8 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.

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 Sect. 6 to determine the schedulability of tasks and task sets under FPDS and arbitrary phasing of some examples of Sect. 4 using the iterative procedure presented in Theorem 13.

7.1 Schedulability of task τ_2 of \mathcal{T}_2

The schedulability of task τ_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 in Sect. 4.2.

To determine the worst-case response time WR_2^D for task τ_2 , we first derive $B_2^D = 2$ using (19). 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 2nd job.

For the 2nd 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 τ_2 meet their deadlines in that period, and the worst-case response time $WR_2^D = 7$.

7.2 Schedulability of task τ_2 of \mathcal{T}_4

We will determine the schedulability of task τ_2 of task set \mathcal{T}_4 in this section. The characteristics of the tasks of \mathcal{T}_4 can be found in Table 4 in Sect. 4.3.2.

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

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

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 2nd job.

For the 2nd job, we find

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

Because $WR_{2,1}^D > D_2 = 7$, we conclude that task τ_2 is not schedulable.

7.3 Schedulability of the task set \mathcal{T}_5

In this section, we will determine the schedulability of the task set \mathcal{T}_5 . The characteristics of the tasks of \mathcal{T}_5 can be found in Table 5 in Sect. 4.3.3.

To determine the worst-case response time WR_1^D for task τ_1 , we first derive $B_1^D = 3$ using (19). 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 τ_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 τ_2 . To this end, we first determine $WR_2^{(0)}$ using Lemma 7, i.e.

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

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 2nd job.

For the 2nd job, we find

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

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 3rd job.

For the 3rd job, we find

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

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 4th job.

For the 4th job, we find

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

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 5th job.

For the 5th job, we find

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

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 τ_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}_5 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, presents pessimistic variants for the worst-case response time analysis of tasks, illustrates the revised analysis for an advanced model for FPDS, and shows that our analysis is sustainable.

8.1 Worst-case response time of highest priority task

In Sect. 4.4, we concluded that the optimism in the existing analysis does not occur for the highest priority task. The next theorem provides a formal basis for that conclusion, by stating that the worst-case response time of the highest priority task τ_1 can be found by only considering the first job of τ_1 in a level-1 active period started at an ε -critical instant. Intuitively, a later job k of τ_1 with $k > 1$ in a level- i active period can only have a longer response time than the first job when it is deferred by its previous job for a longer time than the time the first job is blocked by a task with a lower priority than τ_1 . This would require that the utilization U_1^τ of τ_1 is larger than 1, however.

First, we prove the following lemma.

Lemma 11 *The first job of task τ_1 in a level-1 active period has the largest response time of all jobs of τ_1 in that period.*

Proof The highest priority task τ_1 experiences blocking of at most one subjob of a lower priority task. If the first job of τ_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 τ_1 . The response time $R'_{1,k}(B)$ of job k , with $0 \leq k < l_1$, becomes

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

Because we assume $U \leq 1$, we must have $U_1^\tau < 1$ when task τ_1 is blocked by a lower priority task. 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 τ_1 under FPDS is equal to*

$$WR_1^D = B_1^D + C_1. \tag{55}$$

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 Lehoczky (1990)

The notion of *level- i busy period* originates from Lehoczky (1990), 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 11 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 $\varepsilon > 0$.

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 11. 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 Klein et al. (1993) (on page D-4, referring to Lehoczky (1990)), 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 Klein et al. (1993, pp. 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 τ_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 Fig. 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 processing of jobs, whereas an active period is based on (pending) load or active jobs.

8.2.2 τ_i -Busy period in González Harbour et al. (1991)

In González Harbour et al. (1991), the notion of busy period is slightly modified to accommodate the fact that a task τ_i may be composed of distinct subtasks, each of which may have its own timing requirements and fixed priority. In the following definition, ρ_i denotes the minimum priority of the subtasks of task τ_i .

Definition 12 A τ_i -idle instant is any time t such that all work of priority ρ_i or higher started before t and all τ_i jobs also started before t have completed at or before t .

Definition 13 A τ_i -busy period is an interval of time $[A, B]$ such that both A and B are τ_i -idle instants and there is no time $t \in (A, B)$ such that t is a τ_i -idle instant.

This notion of τ_i -busy period is similar to our level- i active period, with as main difference that a τ_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 Fig. 10. Given Definition 12 and 13, time $t = 35$ belongs to two τ_2 -busy periods, i.e. $[0, 35]$ and $[35, 70]$. Moreover, time $t = 35$ is also a τ_2 -idle instant. Hence, τ_i -busy periods can overlap, and when they overlap, the overlap is termed a τ_i -idle instant. This is considered to be counter-intuitive.

8.2.3 Level- i busy period in George et al. (1996)

After a brief recapitulation of the notion of level- i busy period of Lehoczky (1990) for FPPS, an informal description of a level- i busy period for FPNS under discrete scheduling (Baruah et al. 1990b) is given in George et al. (1996, Appendix A.2). 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 George et al. (1996). In Appendix A.2, the following definition is given.

Definition 14 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 τ_i execute.

Accordingly, the interval of time that a lower priority task blocks task τ_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 Sect. 4.3.1 and Fig. 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 George et al. (1996, Appendix A.2).

Note that George et al. (1996) does not reproduce the definition of Lehoczky (1990) (see Definition 11 above), but presents a new definition. Surprisingly, the differences between these definitions are not discussed. As an example, a (synchronous processor) busy period in George et al. (1996) is described as a right semi-open interval on page 6, whereas the level- i busy period in Lehoczky (1990) is a closed interval.

The notion of level- i busy period for FPNS in George et al. (1996) 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- π_i busy interval in Liu (2000)

In Liu (2000), 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- π_i busy interval*. A level- π_i busy interval is defined as a *left* semi-open interval $(t_0, t]$, i.e. the partitioning of the timeline in Liu (2000) differs from ours. Given the description in Liu (2000), our definition of level- i active period can be viewed as a slightly modified definition of level- π_i busy interval to accommodate our scheduling model for FPDS.

8.3 Pessimistic variants

Given (49) in Lemma 7, 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. We conclude this section with a retrospect on the analysis for FPDS.

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 τ_i under FPDS and arbitrary phasing is given by*

$$\widehat{WR}_{ik}^D = WO_i^P(B_i^D + (k + 1)C_i - F_i) + F_i - kT_i, \tag{56}$$

where $WO_i^P(B_i^D + (k + 1)C_i - F_i)$ is the worst-case occupied time under FPPS of a task τ'_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 $1 < i < n$. □

The pessimism is illustrated by the set \mathcal{T}_2 consisting of three tasks with characteristics as described in Table 2 in Sect. 4.2. For the worst-case response time $\widehat{WR}_{2,0}^D$ of the first job of task τ_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 Lemma 12. Conversely, application of Lemma 7 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 Sect. 4.2 by means of the existing analysis as presented in Burns (1994) and Burns and Wellings (1997). 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 Δ 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 (12)

$$\begin{aligned} &\lim_{\Delta \downarrow 0} (WR_2^P(B_2^D + C_2 - (F_2 - \Delta)) + (F_2 - \Delta)) \\ &= \lim_{\Delta \downarrow 0} (WR_2^P(B_2^D + C_2 - (F_2 - \Delta))) + F_2 \\ &= WO_2^P(B_2^D + C_2 - F_2) + F_2 = \widehat{WR}_{2,0}^D \quad \{(12)\}. \end{aligned}$$

These two results show that $\widehat{WR}_{2,0}^D = \widehat{WR}_2^D$ 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 small positive value Δ 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 τ_i under FPDS and arbitrary phasing is given by*

$$\widehat{WR}_{ik}^D = WR_i^P(B_i^D + (k + 1)C_i - (F_i - \Delta)) + (F_i - \Delta) - kT_i \quad (57)$$

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 τ'_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) Δ is a sufficiently small positive number.

Proof Because $WR_1^P(C) = WO_1^P(C) = C$, $\widehat{WR}_{1,0}^D = \widehat{WR}_{1,0}^D = WR_1^D$. 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 potential additional pessimism introduced by (57) now immediately follows from Lemma 12, i.e. $\widehat{WR}_{ik}^D \geq \widehat{WR}_{ik}^D$.

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

Based on (12), we first conclude that both lemmas are similar for an arbitrary small positive value of Δ , i.e. $\lim_{\Delta \downarrow 0} \widehat{WR}_{ik}^D = \widehat{WR}_{ik}^D$. The additional pessimism potentially introduced by Lemma 13 is illustrated by the set \mathcal{T}_7 consisting of three tasks with

Table 7 Task characteristics of \mathcal{T}_7

	$T_i = D_i$	C_i
τ_1	6.5	3
τ_2	9	3
τ_3	30	3

characteristics as described in Table 7. For this example, the task characteristics are integral multiples of $\delta = 0.5$. For $\Delta = 0.6 > \delta$, we find $\widehat{WR}_{2,0}^D = 12$, which is larger than τ_2 's deadline. Conversely, the worst-case response time \widehat{WR}_2^D of task τ_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 $\widehat{WR}_{2,0}^D = 9$. For this value of Δ , $\widehat{WR}_{2,0}^D = \widehat{WR}_2^D = WR_2^D = 9 \leq D_2$, and reducing the value of Δ will not change the value found for $\widehat{WR}_{2,0}^D$.

The next lemma provides a sufficient condition to guarantee that Lemma 13 introduces no additional pessimism compared to Lemma 12.

Lemma 14 *If the greatest common divisor ($\gcd^{\mathbb{R}^+}$) of the periods and computation times of the tasks exists, and is equal to δ , then $\Delta < \delta$ 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

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

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 δ , all task parameters are integral multiples of δ (by definition), and x' will also be an integral multiple of δ . 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

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

Based on Lemma 20, we conclude

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

Hence, if the $\text{gcd}^{\mathbb{R}^+}$ 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 Davis et al. (2007). The latter analysis is an evolutionary improvement of the analysis given by Tindell et al. (1994, 1995), and Tindell and Burns (1994). A fixed value for Δ is used in Davis et al. (2007), corresponding to the transmission time for a single bit on CAN.

8.3.3 A retrospect

Using our notation, the worst-case response time of a task τ_i under FPDS, arbitrary phasing, and deadlines less than or equal to periods, as described in Liu (2000) can be given by $WR_i^P(B_i^D + C_i)$. As observed in Burns et al. (1993), this analysis is pessimistic, because a task τ_i cannot be preempted while executing its last subjob, i.e. F_i . The original improvement of the worst-case response time of a task τ_i under FPDS as presented in Burns et al. (1993) was not based on B_i^D as given in (19), but on the *maximum length of deferred preemption*. We interpret this latter phrase as a blocking time \widehat{B} given by⁵

$$\widehat{B} = \max_{2 \leq j \leq n} \max_{1 \leq k \leq m_j} C_{j,k}. \tag{58}$$

Though pessimistic, this original improvement is correct, i.e. not optimistic. The problem with the analysis in Burns (1994), Burns and Wellings (1997) is caused by the fact that the non-preemptive behavior of the final subjob of task τ_i itself is not taken into account for $i > 1$, as illustrated by Fig. 7 in Sect. 4.3.3 for task τ_2 . As described in Davis et al. (2007) in the context of schedulability analysis for CAN, this problem can therefore be resolved at the cost of potentially introducing additional pessimism by using \widehat{B}_i^D , which is given by

$$\widehat{B}_i^D = \max(B_i^D, F_i). \tag{59}$$

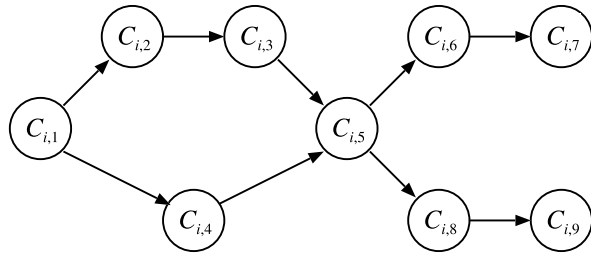
Conversely, the problem with the analysis in Burns (1994), Burns and Wellings (1997) does not occur when $\widehat{B}_i^D = B_i^D$, i.e. when $B_i^D \geq F_i$.

8.4 An advanced model for FPDS

The model for FPDS described in Sect. 2.2 assumes that each job of a task τ_i consists of a sequence of m_i non-preemptable subjobs. In this section, we will illustrate by means of an example how our analytical results can be applied in a context where

⁵From Lemma 11, we conclude that we may ignore the highest priority task in (58).

Fig. 11 An example of a DAG of subjobs, representing the flow graph of task τ_i



a task τ_i consists of a (rooted and connected) directed acyclic graph (DAG) of m_i non-preemptable subjobs.

Consider Fig. 11, with a DAG of subjobs representing the flow graph of task τ_i . The nodes of this graph represent the subjobs and the edges represent the successor relationships of subjobs. The graph has a single root node, with a computation time of $C_{i,1}$, and two leaf nodes, with computation times $C_{i,7}$ and $C_{i,9}$, respectively. During the execution of a job, a single path from the root node to a leaf is traversed. Hence, a job will either execute the subjobs with computation times $C_{i,2}$ and $C_{i,3}$ or the subjob with computation time $C_{i,4}$. Similarly, a job will either execute $C_{i,6}$ and $C_{i,7}$ or $C_{i,8}$ and $C_{i,9}$. The structure of task τ_i plays a role during the analysis of the task itself, and for a lower priority task. The analysis of tasks with a higher priority than τ_i is similar to the case where a job consists of a sequence of subjobs. For the analysis of a task with a lower priority than τ_i , we need to determine the longest computation time C_i of τ_i for all possible paths through the graph. For our example, this is equal to

$$C_i = C_{i,1} + \max(C_{i,2} + C_{i,3}, C_{i,4}) + C_{i,5} + \max(C_{i,6} + C_{i,7}, C_{i,8} + C_{i,9}).$$

For the analysis of task τ_i itself, every leaf node of the DAG gives rise to a case that needs to be examined individually. For our example, we therefore get two cases, a first case for the leaf node $C_{i,7}$, i.e.

$$\begin{aligned} C'_i &= C_{i,1} + \max(C_{i,2} + C_{i,3}, C_{i,4}) + C_{i,5} + C_{i,6} + C_{i,7}, \\ F'_i &= C_{i,7}, \end{aligned}$$

and a second case for the leaf node $C_{i,9}$, i.e.

$$\begin{aligned} C''_i &= C_{i,1} + \max(C_{i,2} + C_{i,3}, C_{i,4}) + C_{i,5} + C_{i,8} + C_{i,9}, \\ F''_i &= C_{i,9}. \end{aligned}$$

The worst-case response time WR_i^D of task τ_i is the maximum of the worst-case response times of these two cases. Note that if $C'_i - F'_i \geq C''_i - F''_i$ and $F'_i \geq F''_i$, then it suffices to consider the first case only. Similarly, if $C''_i - F''_i \geq C'_i - F'_i$ and $F''_i \geq F'_i$, then it suffices to consider only the second case. As an alternative, we can also take a pessimistic approach, and determine WR_i^D based on

$$\begin{aligned} \widehat{C}_i &= \max(C'_i - F'_i, C''_i - F''_i) + \max(F'_i, F''_i), \\ \widehat{F}_i &= \max(F'_i, F''_i). \end{aligned}$$

Table 8 Task characteristics of \mathcal{T}_8

	$T_i = D_i$	C_i
τ_1	16	2
τ_2	24	15
τ_3	36	3

We will now illustrate the analysis for τ_i with a numerical example. Consider the set \mathcal{T}_8 in Table 8. Assume a structure of each job of τ_2 as illustrated in Fig. 11, and let the computation times of the subjobs of task τ_2 be given by $C_{2,1} = 1$, $C_{2,2} = 3$, $C_{2,3} = 4$, $C_{2,4} = 6$, $C_{2,5} = 1$, $C_{2,6} = 3$, $C_{2,7} = 2$, $C_{2,8} = 1$, $C_{2,9} = 5$. We now find $C'_2 = 1 + \max(3 + 4, 6) + 1 + 3 + 2 = 14$, $F'_2 = 2$, $C''_2 = 1 + \max(3 + 4, 6) + 1 + 1 + 5 = 15$, and $F''_2 = 5$. Because $C'_2 - F'_2 = 12 > C''_2 - F''_2 = 10$ and $F'_2 = 2 < F''_2 = 5$, we have to determine the worst-case response times for both cases. Using the analysis presented in Sect. 6, we find 21 for the first case and 20 for the second case. The worst-case response time of τ_2 is therefore assumed for the first case, i.e. $WR_2^D = 21$. For the pessimistic approach, we find $\widehat{C}_2 = \max(12, 10) + \max(2, 5) = 17$, $\widehat{F}_2 = 5$, and derive a worst-case response time for task τ_2 equal to 24.

8.5 A note on sustainability

This section briefly discusses the impact of assuming worst-case rather than fixed computation times and minimum inter-arrival times (or worst-case periods) rather than fixed periods. Stated differently, we consider the sustainability (Baruah and Burns 2006) of our schedulability analysis for FPDS.

As described in Baruah and Burns (2006), a schedulability test is *sustainable* if any task system deemed schedulable by the test remains so if it behaves ‘better’ than mandated by its system specifications, i.e. sustainability requires that schedulability be preserved in situations in which it should be ‘easier’ to ensure schedulability. The concept of sustainability is defined as

Definition 15 (From Baruah and Burns 2006) A schedulability test for a scheduling policy is *sustainable* if any system deemed schedulable by the schedulability test remains schedulable when the parameters of one or more individual job[s] are changed in any, some, or all of the following ways: (i) decreased execution requirements; (ii) later arrival times; (iii) smaller jitter; and (iv) larger relative deadlines.

That paper also proves that response time analysis for FPPS is sustainable.

For FPDS, we have to adapt Definition 15 to our model, i.e. Definition 15 is based on a model that only considers jobs of tasks and does not explicitly consider subjobs of tasks.

Definition 16 A schedulability test for our real-time scheduling model for FPDS is *sustainable* if any system deemed schedulable by the test remains schedulable when the parameters are changed in any, some, or all of the following ways: (i) decreased

execution requirements of subjobs; (ii) later arrival times of jobs; (iii) larger relative deadlines of jobs.

With such an adaptation, the schedulability analysis for our model for FPDS is sustainable, as expressed by the following theorem, for which we merely provide a sketch of a proof.

Theorem 15 *Based on Definition 16, the schedulability analysis for our model of FPDS as expressed by the schedulability test (3) Theorem 12 and Lemmas 7, 12, and 13 is sustainable.*

Sketch of proof Sustainability of our schedulability analysis for FPDS immediately follows from (3) and the fact that:

- the maximum number w_i^D of jobs of task τ_i in a level- i active period, and
- the (pessimistic) worst-case response times WR_{ik}^D in (49), \widehat{WR}_{ik}^D in (56), and $\widehat{\widehat{WR}}_{ik}^D$ in (57) are strictly non-increasing for decreasing computation times of subjobs and increasing periods of tasks. □

Based on Theorem 15, we conclude that we can replace computation times by worst-case computation times and periods by minimum inter-arrival times (or worst-case periods) in our real-time scheduling models in Sect. 2.

To illustrate the significance of our adaptation let’s consider an example showing that Definition 15 is not sufficient for our model. In particular, we show that a schedulable task system under FPDS becomes unschedulable when the computation time C of a task remains the same, but its distribution to the task its subjobs changes. As an example, the task characteristics of \mathcal{T}_9 in Table 9 only differ of those of \mathcal{T}_5 in Table 5 in Sect. 4.3.3 in the distribution of the computation time $C_2 = 4.2$ of τ_2 to its subjobs. Unlike \mathcal{T}_5 , \mathcal{T}_9 is not schedulable, i.e. the second job of task τ_2 in Fig. 12 misses its deadline upon a simultaneous release of both tasks.

Table 9 Task characteristics of \mathcal{T}_9

	$T_i = D_i$	C_i
τ_1	5	2
τ_2	7	2 + 2.2

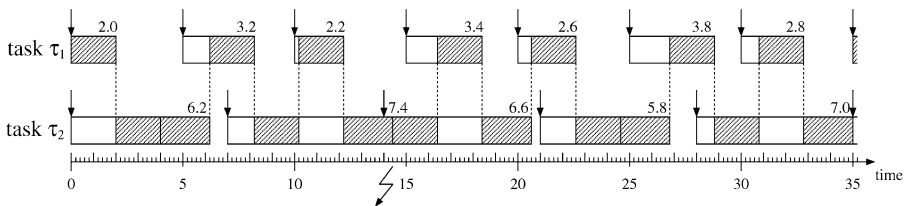


Fig. 12 Timeline for \mathcal{T}_9 under FPDS with a simultaneous release of all tasks at time zero and a deadline miss of task τ_2 at time $t = 14$

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. This problem can therefore arise for all tasks, except for the highest priority task. We observed that González Harbour et al. (1991) identified the same influence of jobs of a task for relative deadlines at most equal to periods in the context of FPPS 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 gave a formal definition of active period, discussed conditions for its termination, 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. Hence, that instant, period, and response time cannot be assumed for any task, except for the lowest priority task. These anomalies for the lowest priority task are caused by the fact that only the lowest priority task cannot be blocked. 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 τ_i -busy period in González Harbour et al. (1991), level- i busy period in George et al. (1996), and level- π_i busy interval in Liu (2000) are similar to our notion of level- i active period. There are striking differences with the notion of busy period in Lehoczky (1990), however. In particular, the level- n busy period never ends for a utilization factor $U = 1$. Moreover, we observed that although Klein et al. (1993) refers to the notion of busy period from Lehoczky (1990) in their description of a method to determine worst-case response times of tasks under FPPS, 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 Davis et al. (2007) corresponds to one of these variants. We illustrated our analysis for an advanced model for FPDS, where tasks are structured as flow graphs of subjobs rather than sequences. Finally, we showed that our analysis for FPDS is sustainable and therefore also applicable for sporadic task systems.

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. We also thank Mike J. Holenderski and the anonymous referees of the Real Time Systems Journal for their comments on earlier versions of this paper.

Open Access This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

Appendix: 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 $f(x)$ and an auxiliary lemma for the ceiling function and the floor function.

Definition 17 The *least common multiple for positive rational numbers* ($\text{lcm}^{\mathbb{Q}^+}$) is defined as

$$\text{lcm}^{\mathbb{Q}^+}(r_1, \dots, r_l) = \min\{r \in \mathbb{Q}^+ \mid r = n_1 \cdot r_1 = \dots = n_l \cdot r_l \text{ with } n_1, \dots, n_l \in \mathbb{N}^+\}, \quad (60)$$

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

Definition 18 The *greatest common divisor for positive rational numbers* ($\text{gcd}^{\mathbb{Q}^+}$) is defined as

$$\text{gcd}^{\mathbb{Q}^+}(r_1, \dots, r_l) = \max\{r \in \mathbb{Q}^+ \mid n_1 \cdot r = r_1, \dots, n_l \cdot r = r_l \text{ with } n_1, \dots, n_l \in \mathbb{N}^+\}, \quad (61)$$

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

Definition 19 The *least common multiple for positive real numbers* ($\text{lcm}^{\mathbb{R}^+}$) is defined as

$$\text{lcm}^{\mathbb{R}^+}(r_1, \dots, r_l) = \min\{r \in \mathbb{R}^+ \mid r = n_1 \cdot r_1 = \dots = n_l \cdot r_l \text{ with } n_1, \dots, n_l \in \mathbb{N}^+\}, \quad (62)$$

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

Definition 20 The *greatest common divisor for positive real numbers* ($\text{gcd}^{\mathbb{R}^+}$) is defined as

$$\text{gcd}^{\mathbb{R}^+}(r_1, \dots, r_l) = \max\{r \in \mathbb{R}^+ \mid n_1 \cdot r = r_1, \dots, n_l \cdot r = r_l \text{ with } n_1, \dots, n_l \in \mathbb{N}^+\}, \quad (63)$$

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

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.

Lemma 15 (Bril 2004, Lemma 4.3) *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 Bril (2004). □

Lemma 16 (Bril (2004, Lemma 4.5)) *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:*

$$\lim_{x \downarrow X} \lceil f(x) \rceil = \left\lfloor \lim_{x \downarrow X} f(x) \right\rfloor + 1. \tag{64}$$

Proof See Bril (2004). □

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:*

$$\lim_{x \uparrow X} \lceil f(x) \rceil = \left\lceil \lim_{x \uparrow X} f(x) \right\rceil. \tag{65}$$

Proof The proof uses the definition of limit:

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

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 proved 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:*

$$\lim_{x \uparrow X} \lfloor f(x) \rfloor = \left\lfloor \lim_{x \uparrow X} f(x) \right\rfloor - 1. \tag{66}$$

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:*

$$\lim_{x \downarrow X} \lfloor f(x) \rfloor = \left\lfloor \lim_{x \downarrow X} f(x) \right\rfloor. \tag{67}$$

Lemma 20 *For $n \in \mathbb{Z}$, $m \in \mathbb{Z} \setminus \{0\}$, and $\varepsilon \in \mathbb{R}$, the following relation holds:*

$$0 < \varepsilon < 1 \Rightarrow \left\lceil \frac{n + \varepsilon}{m} \right\rceil = \left\lfloor \frac{n}{m} \right\rfloor + 1. \tag{68}$$

Proof The proof is based on properties of ceiling and floor functions. Let

$$\left\lceil \frac{n + \varepsilon}{m} \right\rceil = l,$$

with $l \in \mathbb{Z}$, hence

$$m(l - 1) < n + \varepsilon \leq ml.$$

For $0 < \varepsilon < 1$, we get $n + \varepsilon \notin \mathbb{Z}$ and $n < n + \varepsilon < n + 1$. Moreover, because $m(l - 1), ml \in \mathbb{Z}$, we derive

$$0 < \varepsilon < 1 \Rightarrow (m(l - 1) < n + \varepsilon \leq ml \Leftrightarrow m(l - 1) \leq n < ml).$$

With $m \neq 0$, we derive from $m(l - 1) \leq n < ml$ that

$$\left\lfloor \frac{n}{m} \right\rfloor = l - 1,$$

which proves the lemma. □

References

- Audsley NC, Burns A, Richardson MF, Wellings AJ (1991) Hard real-time scheduling: The deadline monotonic approach. In: Proc of the 8th IEEE workshop on real-time operating systems and software (RTOS), May 1991, pp 133–137
- Baeten JCM, Middelburg CA (2002) Process algebra with timing. Springer, Berlin
- Baruah S (2005) The limited-preemption uniprocessor scheduling of sporadic systems. In: Proc of the 17th Euromicro conference on real-time systems (ECRTS), July 2005, pp 137–144
- Baruah S, Burns A (2006) Sustainable schedulability analysis. In: Proc of the 27th IEEE real-time systems symposium (RTSS), December 2006, pp 159–168
- Baruah SK, Mok AK, Rosier LE (1990a) Preemptively scheduling hard-real-time sporadic tasks on one processor. In: Proc of the 11th IEEE real-time systems symposium (RTSS), December 1990, pp 182–190
- Baruah SK, Rosier LE, Howell RR (1990b) Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Syst* 2(4):301–324
- Bril RJ (2004) Real-time scheduling for media processing using conditionally guaranteed budgets. PhD thesis, Technische Universiteit Eindhoven (TU/e), The Netherlands. <http://alexandria.tue.nl/extra2/200412419.pdf>
- Bril RJ (2006) 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
- Bril RJ, Verhaegh WFJ, Lukkien JJ (2004) Exact worst-case response times of real-time tasks under fixed-priority scheduling with deferred preemption. In: Proc of the work-in-progress (WiP) session of the 16th 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), June 2004, pp 57–60
- Bril RJ, Lukkien JJ, Davis RI, Burns A (2006) Message response time analysis for ideal controller area network (CAN) refuted. In: Proc of the 5th international workshop on real time networks (RTN), July 2006, pp 13–18
- Bril RJ, Lukkien JJ, Verhaegh WFJ (2007) Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption revisited. In: Proc of the 19th Euromicro conference on real-time systems (ECRTS), July 2007, pp 269–279
- Burns A (1994) Preemptive priority based scheduling: An appropriate engineering approach. In: Son S (ed) *Advances in real-time systems*. Prentice-Hall, Englewood Cliffs, pp 225–248
- Burns A (2001) Defining new non-preemptive dispatching and locking policies for Ada. In: Proc of the 6th Ada-Europe international conference, May 2001. Lecture notes in computer science, vol 2043. Springer, Berlin, pp 328–336
- Burns A, Wellings AJ (1997) Restricted tasking models. In: Proc of the 8th international real-time Ada workshop, pp 27–32
- Burns A, Nicholson M, Tindell K, Zhang N (1993) Allocating and scheduling hard real-time tasks on a point-to-point distributed system. In: Proc of the 1st workshop on parallel and distributed real-time systems, April 1993, pp 11–20
- Buttazzo GC (2005) *Hard real-time computing systems—predictable scheduling algorithms and applications*, 2nd edn. Springer, Berlin
- Davis RI, Burns A, Bril RJ, Lukkien JJ (2007) Controller area network (CAN) schedulability analysis: refuted, revisited and revised. *Real-Time Syst* 35(3):239–272
- George L, Rivierre N, Spuri M (1996) Preemptive and non-preemptive real-time uni-processor scheduling. Technical report 2966, Institut National de Recherche en Informatique et en Automatique (INRIA), France, September 1996

- González Harbour M, Klein MH, Lehoczky JP (1991) Fixed-priority scheduling with varying execution priority. In: Proc of the 12th IEEE real-time systems symposium (RTSS), December 1991, pp 116–128
- Gopalakrishnan R, Parulkar GM (1996) Bringing real-time scheduling theory and practice closer for multimedia computing. In: Proc of the ACM Sigmetrics conference on measurement & modeling of computer systems, May 1996, pp 1–12
- Hermant J-F, Leboucher L, Rivierre N (1996) Real-time fixed and dynamic priority driven scheduling algorithms: theory and practice. Technical report 3081, Institut National de Recherche en Informatique et en Automatique (INRIA), France, December 1996
- Hooman J (1991) Specification and compositional verification of real-time systems. PhD thesis, Technische Universiteit Eindhoven (TU/e), The Netherlands
- Joseph M, Pandya P (1986) Finding response times in a real-time system. *Comput J* 29(5):390–395
- Klein MH, Ralya T, Pollak B, Obenza R, González Harbour M (1993) A practitioner's handbook for real-time analysis: guide to rate monotonic analysis for real-time systems. Kluwer Academic, Dordrecht
- Koymans R (1990) Specifying real-time properties with metric temporal logic. *Real-Time Syst* 2(4):255–299
- Lee S, Lee C-G, Lee M, Min SL, Kim C-S (1998) Limited preemptible scheduling to embrace cache memory in real-time systems. In: Proc of the ACM Sigplan workshop on languages, compilers and tools for embedded systems (LCTES), June 1998. Lecture notes in computer science, vol 1474. Springer, Berlin, pp 51–64
- Lehoczky JP (1990) Fixed priority scheduling of periodic task sets with arbitrary deadlines. In: Proc of the 11th IEEE real-time systems symposium (RTSS), December 1990, pp 201–209
- Liu JWS (2000) Real-time systems. Prentice-Hall, Englewood Cliffs
- Liu CL, Layland JW (1973) Scheduling algorithms for multiprogramming in a real-time environment. *J ACM* 20(1):46–61
- Mok AK-L (1983) Fundamental design problems of distributed systems for the hard-real-time environment. PhD thesis, Massachusetts Institute of Technology. <http://www.lcs.mit.edu/publications/pubs/pdf/MIT-LCS-TR-297.pdf>
- Mok AK, Poon W-C (2005) Non-preemptive robustness under reduced system load. In: Proc of the 26th IEEE real-time systems symposium (RTSS), December 2005, pp 200–209
- Regehr J (2002) Scheduling tasks with mixed preemption relations for robustness to timing faults. In: Proc of the 23rd IEEE real-time systems symposium (RTSS), December 2002, pp 315–326
- Sha L, Rajkumar R, Lehoczky JP (1990) Priority inheritance protocols: an approach to real-time synchronisation. *IEEE Trans Comput* 39(9):1175–1185
- Simonson J, Patel JH (1995) Use of preferred preemption points in cache-based real-time systems. In: Proc of the IEEE international computer performance and dependability symposium (IPDS), April 1995, pp 316–325
- Spuri M (1996) Analysis of deadline scheduled real-time systems. Technical report 2772, Institut National de Recherche en Informatique et en Automatique (INRIA), France, January 1996
- Tindell K, Burns A (1994) Guaranteeing message latencies on controller area network (CAN). In: Proc of the 1st international CAN conference, September 1994, pp 1–11
- Tindell K, Hansson H, Wellings AJ (1994) Analysing real-time communications: Controller area network (CAN). In: Proc of the 15th IEEE real-time systems symposium (RTSS), December 1994, pp 259–263
- Tindell K, Burns A, Wellings AJ (1995) Calculating controller area network (CAN) message response times. *Control Eng Pract* 3(8):1163–1169
- Wang Y, Saksena M (1999) Scheduling fixed-priority tasks with preemption threshold. In: Proc of the 6th international conference on real-time computing systems and applications (RTCSA), December 1999, pp 328–335
- Weisstein EW (2003) CRC Concise Encyclopedia of Mathematics. Chapman & Hall/CRC, London



Reinder J. Bril received a B.Sc. and an M.Sc. (both with honors) from the University of Twente, and a Ph.D. from the Technische Universiteit Eindhoven, the Netherlands. He started his professional career in January 1984 at the Delft University of Technology. From May 1985 till August 2004, he has been with Philips, and worked in both Philips Research as well as Philips' Business Units. He worked on various topics, including fault tolerance, formal specifications, software architecture analysis, and dynamic resource management, and in different application domains, e.g. high-volume electronics consumer products and (low volume) professional systems. In September 2004, he made a transfer back to the academic world, i.e. to the System Architecture and Networking (SAN) group of the Mathematics and Computer Science department of the Technische Universiteit Eindhoven. His main research interests are currently in the area of reservation-based resource management for networked embedded systems with real-time constraints.



Johan J. Lukkien is head of the System Architecture and Networking Research group at Eindhoven University of Technology since 2002. He received M.Sc. and Ph.D. from Groningen University in the Netherlands. In 1991 he joined Eindhoven University after a two years leave at the California Institute of Technology. His research interests include the design and performance analysis of parallel and distributed systems. Until 2000 he was involved in large-scale simulations in physics and chemistry. Since 2000, his research focus has shifted to the application domain of networked resource-constrained embedded systems. Contributions of the SAN group are in the area of component-based middleware for resource-constrained devices, distributed coordination, Quality of Service in networked systems and schedulability analysis in real-time systems.



Wim F.J. Verhaegh received the mathematical engineering degree with honors in 1990 from the Technische Universiteit Eindhoven, the Netherlands. Since then, he is with the Philips Research Laboratories in Eindhoven, the Netherlands. From 1990 until 1998, he has been a member of the department Digital VLSI, where he has been working on high-level synthesis of DSP systems for video applications, with the emphasis on scheduling problems and techniques. Based on this work, he received a Ph.D. degree in 1995 from the Technische Universiteit Eindhoven. Since 1998, he is working on various optimization aspects of multimedia systems, networks, and applications. On the one hand, this concerns application-level resource management and scheduling, for optimization of quality of service of multimedia systems. On the other hand, this concerns adaptive algorithms and machine learning algorithms for user interaction issues, such as content filtering and automatic playlist generation.