Existing worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption refuted

Reinder J. Bril

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

Abstract

This paper revisits worst-case response time analysis of real-time tasks under fixed priority scheduling with deferred preemption (FPDS), arbitrary phasing, and deadlines within periods. We show that existing worst-case response time analysis, as presented in [3, 4, 5], is too optimistic. In particular, the worst-case response time of a task is not necessarily assumed for the first job of that task when released at an ε-critical instant.

1. Introduction

Based on the seminal paper of Liu and Layland [11], 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 particular 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, which complicates system design, analysis and testing [5, 6, 9, 12]. 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 [4], 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 deadlines with periods has been addressed in a number of papers [3, 4, 5, 9]. In this paper, we will show that the existing analysis is not exact. Whereas it has been shown in [3] that the analysis presented in [4, 5, 9] is pessimistic, we will show by means of an example consisting of just two tasks that the analysis presented in [3, 4, 5] is optimistic. We explore the example by presenting the worst-case response times of both tasks as a function of the relative phasing between the tasks. The exploration reveals that, although the example refutes the existing analysis, it does not refute the conjecture in [3] about an ε-critical instant.

This paper is organized as follows. Section 2 briefly describes a real-time scheduling model for FPDS. Response time analysis for FPDS is recapitulated in Section 3. In Section 4, we present an example that refutes existing worst-case response time analysis under FPDS. We subsequently present the results of the exploration. The paper is concluded in Section 5.

2. Real-time scheduling models

This section describes a basic scheduling model for FPPS and a refined model for FPDS. Most of the definitions and assumptions of these models originate from [11].

2.1. Basic model for FPPS

We assume a single processor and a set $\mathcal{T}$ of n periodically released, independent tasks $\tau_1, \tau_2, \ldots, \tau_n$. At any moment in time, the processor is used to execute the highest priority task that has work pending.

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* $\varphi_i \in \mathbb{R}$. 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 job of task τ_i with release time φ_i serves as a reference activation, and is referred to as job zero. The release of job k of τ_i therefore takes place at time $a_{ik} = \varphi_i + kT_i$, $k \in \mathbb{Z}$. The deadline of job k of τ_i takes place at $d_{ik} = a_{ik} + D_i$. The set of phasings φ_i is termed the phasing φ of the task set $\mathcal{T}$.

The *response interval* of job k of τ_i is defined as the time span between the activation time of that job and its completion time c_{ik}, i.e. $[a_{ik}, c_{ik})$. The *response time* r_{ik} of job k of τ_i is defined as the length of its response interval, i.e. $r_{ik} = c_{ik} - a_{ik}$. The *worst-case response time* WR_i of a task τ_i is the largest response time of any of its jobs, i.e.

$$WR_i = \sup_{\varphi,k} r_{ik}. \tag{1}$$

A *critical instant* of a task is defined as an (hypothetical) instant that leads to the worst-case response time for that task.

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 [7, 8, 11]. We also assume other standard basic assumptions [11], i.e. tasks are ready to run at the start of each period and do no suspend themselves, tasks will be preempted instantaneously when a higher priority task becomes ready to run, a job of a task does not start before its previous job is completed, and the overhead of context switching and task scheduling is ignored. Finally, we assume that the deadlines are hard, i.e. each job of a task must be completed before its deadline. Hence, a set $\mathcal{T}$ on n periodic tasks can be scheduled if and only if

$$WR_i \leq D_i \tag{2}$$

for all $i = 1, \ldots, n$.

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.

2.2. Refined model for FPDS

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

3. Recapitulation of response time analysis

In this section, we recapitulate worst-case response time analysis for both FPPS and FPDS. Because we will express response times under FPDS in terms of response times under FPPS, we will use subscripts D and P to denote FPDS and FPPS, respectively. Moreover, we will use a functional notation for response times when needed, e.g. $WR_i(C_i)$.

3.1. Worst-case analysis for FPPS

To determine worst-case response times under arbitrary phasing, it suffices to consider only critical instants. For FPPS, critical instants are given by time points at which all tasks have a simultaneous release [11].

From this notion of critical instants, Joseph and Pandya [7] have derived that for deadlines within periods (i.e. $D_i \leq T_i$) the worst-case response time WR_i^{P} of a task τ_i is given by the smallest $x \in \mathbb{R}^+$ that satisfies

$$x = C_i + \sum_{j<i} \left\lceil \frac{x}{T_j} \right\rceil C_j. \tag{3}$$

To calculate worst-case response times, we can use an iterative procedure based on recurrence relationships [1]. The procedure starts with a lower bound.

$$\begin{aligned} wr_i^{(0)} &= C_i \\ wr_i^{(k+1)} &= C_i + \sum_{j<i} \left\lceil \frac{wr_i^{(k)}}{T_j} \right\rceil C_j \end{aligned}$$

The procedure is stopped when the same value is found for two successive iterations of k or when the deadline D_i is exceeded. In the former case, it yields the smallest solution of the recursive equation, i.e. the worst-case response time of τ_i. In the latter case the task is not schedulable. Termination of the procedure is ensured by the fact that the sequence $wr_i^{(k)}$ is bounded (from below by C_i, and from above by D_i) and non-decreasing, and that different values for successive iterations differ at least $\min_{j<i} C_j$.

The interested reader is referred to [8, 10, 13] for techniques to derive worst-case response times for arbitrary deadlines. The main difference with deadlines within periods is that for arbitrary deadlines the worst-case response time of a task is not necessarily assumed for the first job that is released at the critical instant.

3.2. Worst-case analysis for FPDS

In this section, we recapitulate response time analysis for FPDS and arbitrary phasing for deadlines within periods as described in [3, 4, 5].

The non-preemptive nature of subjobs may cause blocking of a task by at most one lower priority task. The maximum blocking B_i of task τ_i by a lower priority task is equal to the longest computation time of any subjob of a task with a priority lower than task τ_i, i.e.

$$B_i = \max_{j>i} \max_{1 \leq k \leq m(j)} C_{j,k}. \tag{4}$$

The worst-case response time $\widetilde{WR}_i^{\mathrm{D}}$ under FPDS and arbitrary phasing presented in [4] and [5] is given by

$$\widetilde{WR}_i^{\mathrm{D}}(\Delta) = WR_i^{\mathrm{P}}(B_i + C_i - (F_i - \Delta)) + (F_i - \Delta). \tag{5}$$

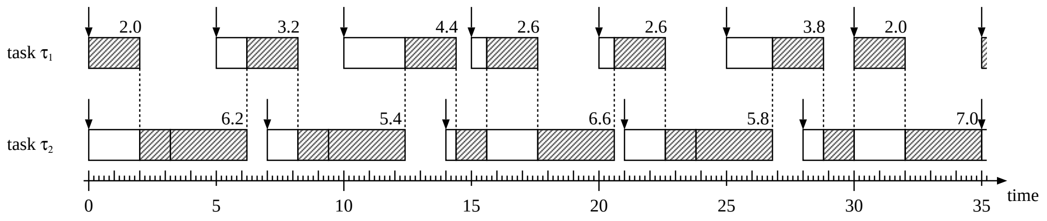

Figure 1. Timeline for $\mathcal{T}_1$ under FPDS with a simultaneous release at time zero. The numbers at the top right corner of the boxes denote the response times of the respective releases.

According to [5], Δ is an arbitrary small positive value needed to ensure that the final subjob has actually started, i.e. $0 < \Delta \ll F_i$. Hence, when task τ_i has consumed $C_i - (F_i - \Delta)$, the final subjob has (just) started.

As described in [3], the analysis in [4, 5] does not take into account that τ_i can only be blocked by a subjob of a lower priority task if that subjob starts an amount of time Δ *before* the simultaneous release of τ_i and all tasks with a higher priority than τ_i. That paper therefore revisits critical instants, and postulates the following conjecture.

Conjecture 1 *An ε-critical instant of a task τ_i under FPDS and arbitrary phasing occurs when that task is released simultaneously with all tasks with a higher priority than τ_i, and the subjob with the longest computation time of all lower priority tasks starts an infinitesimal time $\varepsilon > 0$ before that simultaneous release.*

From this conjecture, it is concluded that a critical instant for FPDS is a supremum for all but the lowest priority task, i.e. that instant can not be assumed. The results in [4, 5] are identical to the results in [3] for the lowest priority task, and the results become similar for the other tasks by replacing B_i in (5) by $(B_i - \Delta)^+$, i.e.

$$WR_i^{\mathrm{D}}(\Delta) = WR_i^{\mathrm{P}}((B_i - \Delta)^+ + C_i - (F_i - \Delta)) + (F_i - \Delta). \quad (6)$$

Here, the notation w^+ stands for $max(w, 0)$, which is used to indicate that the blocking time can not become negative for the lowest priority task. According to [3], the worst-case response time is actually a supremum for all but the lowest priority task, i.e.

$$WR_i^{\mathrm{D}} = \lim_{\Delta \downarrow 0} WR_i^{\mathrm{D}}(\Delta). \quad (7)$$

4. A counterexample

The task characteristics of our counterexample are given in Table 1. The table includes the results of the exploration. Note that the (*processor*) *utilization factor U* of the task set $\mathcal{T}_1$ is given by $U = \frac{2}{5} + \frac{4.2}{7} = 1$.

4.1. Existing analysis is too optimistic

We will now show that the worst-case response time of task τ_2 as determined by (6) is too optimistic.

task	T	C	D	WR^{D}
τ_1	5	2	5	5
τ_2	7	1.2 + 3	6.8	7

Table 1. Task characteristics of $\mathcal{T}_1$ and worst-case response times under FPDS.

Based on (6) and using $\Delta = 0.1$, we derive

$$\begin{aligned} WR_2^{\mathrm{D}}(\Delta) &= WR_2^{\mathrm{P}}((B_2 - \Delta)^+ + C_2 - (F_2 - \Delta)) + (F_2 - \Delta) \\ &= WR_2^{\mathrm{P}}(0 + 4.2 - (3.0 - 0.1)) + (3.0 - 0.1) \\ &= WR_2^{\mathrm{P}}(1.3) + 2.9 = 6.2, \end{aligned}$$

which is smaller than the deadline $D_2 = 6.8$ of task τ_2.

Figure 1 shows a timeline with the executions of the two tasks of $\mathcal{T}_1$ in an interval of length 35, i.e. equal to the *hyperperiod H* of the tasks, which is equal to the least common multiple (lcm) of the periods. The schedule in $[0, 35)$ is repeated in the intervals $[hH, (h+1)H)$ with $h \in \mathbb{Z}$, i.e. the schedule is periodic with period H. As illustrated in Figure 1, the derived value for $WR_2^{\mathrm{D}}(\Delta)$ corresponds with the response time of the 1*st* job of task τ_2 upon a simultaneous release with task τ_1, i.e. when task τ_2 is released at an ε-critical instant. However, the response time of the 5*th* job of task τ_2 is equal to 7 in that figure, which is larger than the deadline $D_2 = 6.8$ of τ_2. Task τ_2 is therefore not schedulable, which illustrates that the existing analysis is too optimistic.

4.2. Exploration

Above, we have shown that even when deadlines are within periods, we cannot restrict ourselves to the response time of a single 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.

We will now explore the example in more detail, by considering the worst-case response times for both tasks under FPDS for specific phasings. To this end, we vary the relative phasing φ_{R} of task τ_2 with respect to τ_1, i.e. $\varphi_{\mathrm{R}} = \varphi_2 - \varphi_1$. Because the greatest common divisor of T_1 and T_2 is equal to 1, we can restrict φ_{R} to values in the interval $[0, 1)$. In this section, we will vary the phasing φ_2 of τ_2 and keep the phasing φ_1 of task τ_1 equal to zero, i.e. $\varphi_R = \varphi_2$.

The worst-case response times of both task τ_1 and task τ_2 under FPDS are shown as a function of the phasing in Figure 2. WR_2^D is equal to 7.0 and assumed for a relative

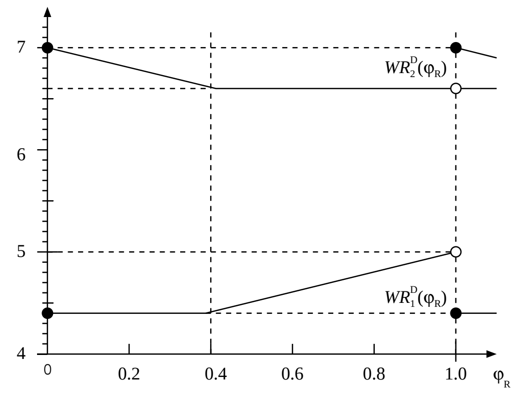

Figure 2. Worst-case response times under FPDS as a function of the relative phasing φ_R.

phasing $\varphi_R = 0$, i.e. when task τ_2 is released at an ε-critical instant. Note that WR_1^D, given by

$$WR_1^D = \sup_{\varphi_R} WR_1^D(\varphi_R) = \lim_{\varphi_R \uparrow 1} WR_1^D(\varphi_R) = 5.0,$$

is a supremum and not a maximum, i.e. that value can not be assumed. We therefore conclude that although the example refutes the worst-case response time analysis, it does not refute Conjecture 1 concerning an ε-critical instant.

5. Conclusion

In this document, we revisited worst-case response time analysis of real-time tasks under FPDS and arbitrary phasing. We showed by means of an example consisting of just two tasks that existing worst-case response time analysis for deadlines within periods as presented in [3, 4, 5] is too optimistic. Notably, the example does not refute Conjecture 1 from [3] concerning the notion of ε-critical instant. The example merely reveals that the worst-case response time of a task scheduled under FPDS is not necessarily assumed for the first job of that task when released at an ε-critical instant. This is a similar result as presented in [10] for critical instants of tasks under FPPS with arbitrary phasing and deadlines greater than periods.

Worst-case response time analysis under FPDS and arbitrary phasing is a topic of future work. We are currently investigating the possibility to determine the worst-case response time of a task τ_i based on the response times of jobs of τ_i in a so-called *level-i active period* that starts at an ε-critical instant [2]. Initial results suggest that the technique is similar to existing techniques for FPPS with arbitrary phasing and arbitrary deadlines [8, 10, 13].

Acknowledgement

First of all, I thank my son Wander for his inspiration; the problem with the existing analysis occurred to me while watching him play in a sandpit. Next, I thank Wim F.J. Verhaegh from Philips Research and Johan J. Lukkien from the TU/e for discussions. Finally, I thank Alan Burns from the University of York and Gerhard Fohler from the University of Kaiserslautern for suggestions and remarks.

References

[1] N. Audsley, A. Burns, M. Richardson, and A. Wellings. Hard real-time scheduling: The deadline monotonic approach. In *Proc.* 8th *IEEE Workshop on Real-Time Operating Systems and Software (RTOSS)*, pp. 133–137, May 1991.

[2] R. Bril. Existing worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption is too optimistic. CS-Report 06-05, Technische Universiteit Eindhoven (TU/e), February 2006.

[3] R. Bril, W. Verhaegh, and J. Lukkien. Exact worst-case response times of real-time tasks under fixed-priority scheduling with deferred preemption. In *Proc. Work-in-Progress (WiP) session of the* 16th *Euromicro Conf. on Real-Time Systems (ECRTS)*, pp. 57–60, June 2004.

[4] A. Burns. Preemptive priority based scheduling: An appropriate engineering approach. In S. Son, editor, *Advances in Real-Time Systems*, pp. 225–248. Prentice-Hall, 1994.

[5] A. Burns and A. Wellings. Restricted tasking models. In *Proc.* 8th *Int. Real-Time Ada Workshop*, pp. 27–32, 1997.

[6] R. Gopalakrishnan and G. Parulkar. Bringing real-time scheduling theory and practice closer for multimedia computing. In *Proc. ACM Sigmetrics Conf. on Measurement & Modeling of Computer Systems*, pp. 1–12, May 1996.

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

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

[9] S. Lee, C.-G. Lee, M. Lee, S. Min, and C.-S. Kim. Limited preemptible scheduling to embrace cache memory in real-time systems. In *Proc. ACM Sigplan Workshop on Languages, Compilers and Tools for Embedded Systems (LCTES), LNCS 1474*, pp. 51–64, June 1998.

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

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

[12] J. Simonson and J. Patel. Use of preferred preemption points in cache-based real-time systems. In *Proc. IEEE Int. Computer Performance and Dependability Symposium (IPDS)*, pp. 316–325, April 1995.

[13] K. Tindell. An extendible approach for analysing fixed priority hard real-time tasks. Report YCS 189, Department of Computer Science, University of York, December 1992.