Towards best-case response times of real-time tasks under fixed-priority scheduling with preemption thresholds

H.J. Rivera Verduzco
Technische Universiteit Eindhoven (TU/e),
Den Dolech 2, 5600 AZ Eindhoven,
The Netherlands
h.j.rivera.verduzco@student.tue.nl

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

***Abstract*—Fixed-priority scheduling with preemption thresholds (FPTS) is currently implemented in many real-time operating systems, such as in ThreadX and the AUTOSAR/OSEK standard. Furthermore, FPTS can be seen as a generalization for fixed-priority preemptive scheduling (FPPS) and for fixed-priority non-preemptive scheduling (FPNS). In this paper, we show that the best-case response time analysis for FPTS is most likely not an straight forward extension of the current best-case analysis for FPPS. In addition, we show, as an intermediate step towards the exact best-case response time analysis for FPTS, that the execution time of a task scheduled under FPNS is a tight lower bound for the response time of such a task.**

1. Introduction

Fixed-priority scheduling with preemption thresholds (FPTS) was first introduced in the ThreadX kernel [1] to limit the number of preemptions that a task may experience with the aim to improve schedulability and reduce preemption overheads. It is currently also supported by the AUTOSAR/OSEK standard [2]. Although an exact worst-case response time analysis for FPTS has already been proposed in the literature [3], little attention has been paid to its best-case counterpart.

FPTS is a generalization for fixed-priority preemptive scheduling (FPPS) and fixed-priority non-preemptive scheduling (FPNS). Therefore, it could be desirable to consider first the best-case response time analysis for FPPS and FPNS before investigating the corresponding analysis for FPTS. In particular, the best-case response time analysis for FPPS has been addressed in [4] for constraint deadlines and in [5] for arbitrary deadlines. On the other hand, the best-case response time for FPNS has not been formally studied yet.

FPNS is widely used in message-based protocols under a network environment. For example, the Controller Area Network (CAN) uses a priority based arbitration and non-preemptive scheduling for message transmission. Special attention has been paid to the worst-case analysis of FPNS [7] to guarantee real-time requirements. However, when considering jitter induced by a network, it becomes relevant to study best-case response times next to worst-case response times of messages scheduled under FPNS.

In [6], a lower bound for fixed-priority scheduling with deferred preemptions (FPDS) is presented. Since FPNS is a special case of FPDS, this analysis also applies for FPNS. In particular, this analysis gives as a result that the execution time of a task is a lower bound for the best-case response time of such a task under FPNS. However, we would like to investigate whether this lower bound is tight for all cases.

The rest of this paper is organized as follows. Section 2 introduces the basic real-time scheduling model for FPTS that we will analyze as well as some related notions. The best-case response time for FPNS is presented in Section 3. Furthermore, Section 4 introduces an example that refutes an intuitive notion of optimal instant for FPTS. Finally, we conclude this paper in Section 5.

2. Real-time scheduling model

2.1. Basic model for FPTS

We assume a single processor and a set $\mathcal{T}$ of n independent periodic tasks $\tau_1, \tau_2, ..., \tau_n$ with unique and fixed priorities π_1, $\pi_2 ... , \pi_n$. We also assume that tasks are given in order of decreasing priority, i.e. τ_1 has the highest priority whereas τ_n has the lowest priority. A higher priority is represented by a higher value.

Each task τ_i generates an infinite sequence of jobs ι_{ik} with $k \in \mathbb{Z}$. In addition, each task is characterized by a *period* $T_i \in \mathbb{R}^+$, a *worst-case computation time* $WC_i \in \mathbb{R}^+$, a *best-case computation time* $BC_i \in \mathbb{R}^+$, where $BC_i \leq WC_i$, a *phasing* $\phi_i \in \mathbb{R}$, a (relative) *worst-case deadline* $WD_i \in \mathbb{R}^+$, and a (relative) *best-case deadline* $BD_i \in \mathbb{R}^+ \cup \{0\}$, where $BD_i \leq WD_i$. The set of phasings ϕ_i is termed the phasing ϕ of the task-set $\mathcal{T}$. We assume arbitrary deadlines; hence, deadlines BD_i and WD_i may be smaller than, equal to, or larger than period T_i. The deadlines BD_i and WD_i are relative to the activations of the jobs. In addition, we assume that tasks do not suspend themselves, jobs do not start before the completion of previous jobs of the same task, and the overhead of context switching and task scheduling is ignored.

For fixed-priority preemptive scheduling with preemption thresholds (FPTS), each task τ_i has an additional property called *preemption threshold* denoted by θ_i, where $\pi_1 \geq \theta_i \geq \pi_i$. A task τ_i can be preempted by a higher priority task τ_h if and only if $\pi_h > \theta_i$. FPPS is a special case of FPTS when $\forall_{1\leq i\leq n}\theta_i = \pi_i$, and FPNS is a special case when $\forall_{1\leq i\leq n}\theta_i = \pi_1$.

2.2. Derived notions

Given the *activation time* $a_{i,k}$ of the k^{th} job of a task τ_i and its (absolute) *finalization time* $f_{i,k}$, the *response time* $R_{i,k}$ of such a job is defined as the time elapsed between its finalization and activation, i.e. $R_{i,k} = f_{i,k} - a_{i,k}$. The *absolute start time* $s_{i,k}$ is the time at which job k of τ_i starts its execution. Furthermore, The *relative start time* $S_{i,k}$ is the start time relative to the activation of a job, i.e. $S_{i,k} = s_{i,k} - a_{i,k}$. Finally, we define the *hold time* $H_{i,k}$ as the time elapsed between the start of a job and its completion, this is expressed as $H_{i,k} = f_{i,k} - s_{i,k}$. In general, under this model, it always holds that $R_{i,k} = S_{i,k} + H_{i,k}$. Figure 1 shows the basic task model with these notions.

The *worst-case response time* WR_i and the *best-case response time* BR_i of a task τ_i are defined as the longest and the shortest response times of its jobs respectively, i.e.

$$WR_i \stackrel{\text{def}}{=} \sup_{\phi,k} R_{i,k}(\phi) \tag{1}$$

$$BR_i \stackrel{\text{def}}{=} \inf_{\phi,k} R_{i,k}(\phi), \tag{2}$$

where $R_{i,k}(\phi)$ denotes a dependency of response time $R_{i,k}$ on phasing ϕ.

We say that a set $\mathcal{T}$ of n periodic tasks is schedulable if and only if

$$\forall_{1\leq i\leq n}(BD_i \leq BR_i \wedge WR_i \leq WD_i). \tag{3}$$

Finally, we define the best-case utilization $U^{\mathcal{T}}$ as the best-case fraction of the processor time spent on the execution of $\mathcal{T}$, i.e.

$$U^{\mathcal{T}} \stackrel{\text{def}}{=} \sum_{1\leq i\leq n} \frac{BC_i}{T_i}. \tag{4}$$

3. Best-case response time for FPNS

Since FPNS is a special case of FPTS when thresholds of tasks are set to the highest priority, we first investigate the best-case response time under non-preemptive scheduling as an intermediate step towards the best-case analysis for FPTS.

Lemma 1. *Let τ_i be a non-preemptive task, i.e. $\theta_i = \pi_1$. The best-case response time of τ_i is always equal to its best-case computation time, i.e. $BR_i = BC_i$.*

Proof. Given a task-set $\mathcal{T}$ and a non-preemptive task $\tau_i \in \mathcal{T}$, assume that all jobs of τ_i have a computation time equal to BC_i. Since τ_i is non-preemptive, the hold time of all its jobs is simply equal to BC_i. Recall that the response time of a job k of τ_i is given by $R_{i,k} = S_{i,k} + H_{i,k}$. Therefore, in order to prove the lemma, it is sufficient to show that it is always possible to schedule task τ_i in such a way that the relative start time of a job k of τ_i is zero, i.e. $S_{i,k} = 0$. We divide this proof in two cases:

{*Case* $U^{\mathcal{T}} < 1$}. Given an arbitrary schedule for $\mathcal{T}$, let $[t_s, t_e)$ be an idle interval in such a schedule. Note that it is always possible to find an idle interval because $U^{\mathcal{T}} < 1$. Furthermore, let $\iota_{i,k}$ be the first job of τ_i activated at or after time t_e, i.e. $t_e \leq a_{i,k}$ and there are no jobs of τ_i activated within $[t_e, a_{i,k})$. We now show that, by pushing the activation of $\iota_{i,k}$ to occur in the interval $[t_s, t_e)$, the relative start time $S_{i,k}$ of job $\iota_{i,k}$ becomes zero.

First observe that there is no other job of τ_i between the idle interval $[t_s, t_e)$ and $\iota_{i,k}$; hence, after pushing the activations of τ_i till $a_{i,k}$ occurs in $[t_s, t_e)$, job $\iota_{i,k}$ does not experience blocking by a previous job. Furthermore, since $[t_s, t_e)$ was originally idle, job $\iota_{i,k}$ can start upon activation, leading to $S_{i,k} = 0$.

{*Case* $U^{\mathcal{T}} = 1$ *and the lcm of the periods exists*}. In order to prove this case, first recall that the hyperperiod is defined as the length of the shortest time interval in which the schedule repeats itself after an initial start-up. For a set of periodic tasks, Leung and Merrill proved in [4] that the hyperperiod H can be calculated as the least common multiple of the periods, i.e. $H = lcm(T_1, ..., T_n)$ for n periodic tasks.

Let $n_i = H/T_i$ be the number of jobs of τ_i in every hyperperiod and assume that no job of τ_i starts upon activation. Furthermore, let $\iota_{i,1}$ and ι_{i,n_i} be the first and last job of τ_i respectively in an hyperperiod. Hence, for each interval $[a_{i,k'}, s_{i,k'})$ with $1 \leq k' \leq n_i$, there are only higher priority tasks of τ_i and/or at most one blocking lower priority task executing within such an interval. As an example of this situation, consider the timeline depicted in Figure 2 for task-set $\mathcal{T}_1$ described in Table 1. As can be seen, no job of τ_2 can start upon activation.

TABLE 1. TASK CHARACTERISTICS OF $\mathcal{T}_1$.

task	T_i	WD_i	C_i	π_i	θ_i
τ_1	4	4	2	2	2
τ_2	6	6	3	1	2

The hyperperiod is $H = 12$ and $U^{\mathcal{T}_1} = 1$.

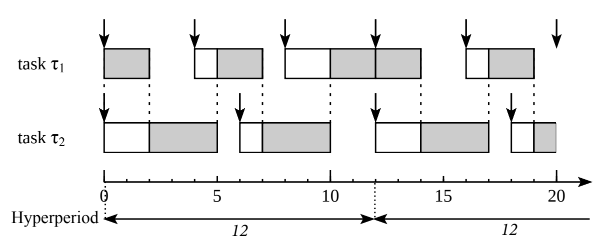

Figure 2. A timeline for $\mathcal{T}_1$. The schedule repeats itself in every interval with length $H = 12$.

Now let $\iota_{i,k}$ be the job of τ_i with the shortest relative start time among all jobs of τ_i in an hyperperiod. Since

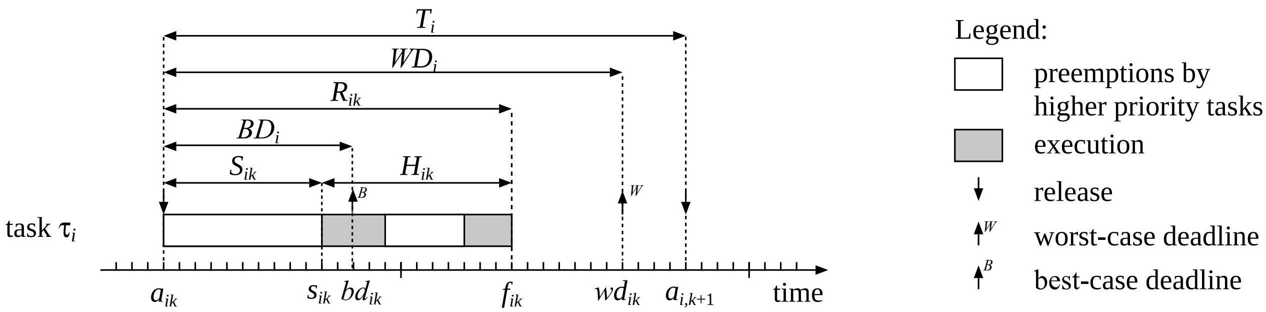

Figure 1. Basic model for a periodic task τ_i.

no job of τ_i starts upon activation and $\iota_{i,k}$ is the job with the shortest start time, we can push the activation of all jobs of τ_i to a later moment in time, without modifying the schedule, till job $\iota_{i,k}$ starts upon activation. After pushing the activations of τ_i, it holds that $S_{i,k} = 0$ therefore concluding the proof. Figure 3 shows an example where the schedule shown in Figure 2 is preserved after pushing the activations of τ_2 till its second job starts upon activation. □

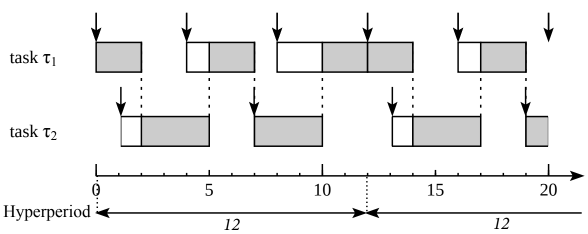

Figure 3. A timeline for $\mathcal{T}_1$. After pushing the activations of τ_2, the second job can immediately start.

4. An introductory example

For FPPS, a task τ_i assumes its best-case response when one of its jobs experiences an optimal instant [5]. The optimal instant occurs when the completion of a job of τ_i coincides with the simultaneous activation of all its higher priority tasks. Based on this, it could be tentative to think that an optimal instant for FPTS may occur when the completion of a job of a task τ_i coincides with the simultaneous activation of all higher priority tasks that can preempt τ_i. These are the tasks in $\{\tau_h | \pi_h > \theta_i\}$. In this section, we introduce an example that refutes this intuition of optimal instant for FPTS.

Consider the set $\mathcal{T}_2$ of three tasks with characteristics as described in Table 2. The best-case response time values in this table were found by means of extensive simulations. This is, by considering all values of $\phi_1 = \{0, ..., 34\}$ and $\phi_2 = \{0, ..., 49\}$, while fixing the phasing of τ_3 to $\phi_3 = 0$.

Figure 4 shows a timeline for task-set $\mathcal{T}_2$ with the configuration that we aim to refute for the last job of task τ_3. As can be seen, the completion of this job coincides with an activation of the highest priority task τ_1 at time $t = 0$.

TABLE 2. TASK CHARACTERISTICS OF $\mathcal{T}_2$.

task	T_i	WD_i	$WC_i = BC_i$	π_i	θ_i	WR_i	BR_i
τ_1	35	20	10	3	3	10	10
τ_2	50	65	20	2	2	62	20
τ_3	70	70	22	1	2	66	32

The *least common multiple* of the periods is 350 and $U^{\mathcal{T}_2} = 1$.

Note that task τ_1 can preempt task τ_3 because $\pi_1 > \theta_3$. On the other hand, task τ_2 cannot preempt task τ_3 because $\pi_2 = \theta_3$. For the latter reason, the activation of task τ_2 is set to a sufficiently small amount of time after the start time of the last job of task τ_3, therefore, allowing more time before $t = 0$ for the execution of task τ_3.

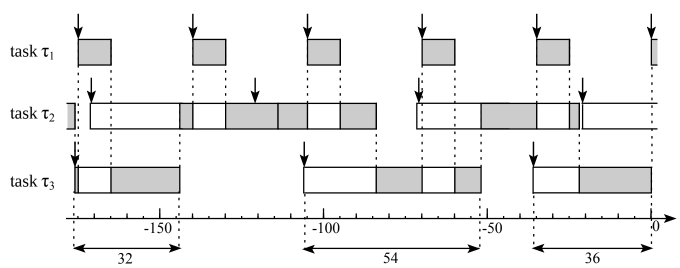

Figure 4. A timeline for $\mathcal{T}_2$ where the activation of the highest priority task τ_1 coincides with the completion of a job of task τ_3 at time $t = 0$.

As can be seen from Figure 4, the last job of task τ_3 that is experiencing the optimal instant has a response time of $R_{3,1} = 36$. However, the best-case response time of task τ_3 is $BR_3 = 32$, and it is assumed by the job activated at time $t = -176$. Note also that the response time of the job experiencing the optimal instant cannot be reduced by pushing the activations of task τ_3 because the job activated at time $t = -176$ already can start upon activation. Hence, pushing the activations of task τ_3 to a later moment in time would modify the schedule.

Based on the previous example, we formulate the following fact.

Fact 1. *In FPTS, a job k of task τ_i does not necessarily assume the best-case response time when the activation of all higher priority tasks that can preempt task τ_i coincides with the completion of job k.*

5. Conclusion

In this paper, we showed that the best-case response time of a non-preemptive task is equal to its execution time when scheduled under FPTS or FPNS. Furthermore, we also showed by means of an example that the notion of optimal instant for FPTS is possibly not an straight forward extension of the existent notion for FPPS. An exact best-case response time analysis for FPTS is currently under study.

Acknowledgments

The authors would like to thank...

References

[1] Express Logic Inc. ThreadX Technical Features, v5.1. [Online]. Available: http://rtos.com/products/threadx, June 2010.

[2] S. Bunzel. *AUTOSAR - the standardized software architecture*. Informatik-Spektrum, vol. 34, no. 1, pp. 7983, 2011.

[3] U. Keskin, R. Bril, and J. Lukkien. *Exact response-time analysis for fixed-priority preemption-threshold scheduling.* In IEEE Conf. on Emerging Technologies and Factory Automation (ETFA), Sept. 2010, pp. 14.

[4] J. Leung and R. Merrill. *A note on the preemptive scheduling of periodic, real-time tasks*. Information Processing Letters, 18:115118, 1980.

[5] R.J. Bril, J.J. Lukkien, and R.H. Mak. *Best-case response times and jitter analysis of real-time tasks with arbitrary deadlines.* In Proc. 21st International Conference on Real-Time Networks and Systems (RTNS), ACM, pp. 193-202, October 2013.

[6] R.L. Bril, W.F.J. Verhaegh. *Towards best-case response times of real-time tasks under fixed-priority scheduling with deferred preemption.* In Proc. Work-in-Progress (WiP) session of the 17th Euromicro Conf. on Real-Time Systems (ECRTS), pages 1720, July 2005.

[7] R.I. Davis, A. Burns, R.J. Bril, J.J. Lukkien. *Controller area network (CAN) schedulability analysis: refuted, revisited and revised.* Real-Time Syst 35(3):239272, 2007.