Technical Report

Errata: Timing Analysis of Fixed Priority Self-Suspending Sporadic Tasks

Geoffrey Nelissen
José Fonseca
Gurulingesh Raravi
Vincent Nélis

CISTER-TR-170205

2017
Errata: Timing Analysis of Fixed Priority Self-Suspending Sporadic Tasks

Geoffrey Nelissen, José Fonseca, Gurulingesh Raravi, Vincent Nélis

*CISTER Research Centre
Polytechnic Institute of Porto (ISEP-IPP)
Rua Dr. António Bernardino de Almeida, 431
4200-072 Porto
Portugal
Tel.: +351.22.8340509, Fax: +351.22.8321159
E-mail: grpn@isep.ipp.pt, jcnfo@isep.ipp.pt, guhri@isep.ipp.pt, nelis@isep.ipp.pt
http://www.cister.isep.ipp.pt

Abstract

In the paper "Timing Analysis of Fixed Priority Self-Suspending Sporadic Tasks" published in ECRTS 2015, a MILP formulation is provided to compute an upper-bound on the worst-case response time (WCRT) of one self-suspending task running concurrently with a set of higher priority non-self-suspending tasks. Section VI of that paper extends the MILP formulation to the case where the higher priority tasks are also self-suspending. This generalisation is incorrect. We present the problem and its solution in this technical report.
I. INCORRECT STATEMENT

In [1], a MILP formulation is provided to compute an upper bound on the worst-case response time (WCRT) of one self-suspending task running concurrently with a set of higher priority non-self-suspending tasks. Section VI of [1] extends the MILP formulation to the case where the higher priority tasks are also self-suspending. It is stated that:

Claim 1 (in [1]). “[...] each higher priority self-suspending task \( \tau_k \) can safely be replaced by a non-self-suspending task \( \tau'_k \) in the response time analysis. The new parameter \( J_k \) is the jitter and is given by \( J_k = WCRT_k - C_k \). The worst-case execution time \( C_k \) of the equivalent task \( \tau'_k \) is defined as the sum of the worst-case execution times of all \( \tau_k \)'s execution regions, that is, \( C_k = \sum_{j=1}^{m} C_{k,j} \)."

This claim is supported by Theorem 2 repeated below.

Theorem 2 (in [1]). The interference caused by \( \tau_k \in \text{lp}(\tau_i) \) on a self-suspending task \( \tau_i \) is upper bounded by the interference caused by the transformed task \( \tau'_k \) defined as \( \langle C_k, D_k, T_k, J_k \rangle \) in the response time analysis. The new parameter \( J_k \) is the jitter and is given by \( J_k = WCRT_k - C_k \). The worst-case execution time \( C_k \) of the equivalent task \( \tau'_k \) is defined as the sum of the worst-case execution times of all \( \tau_k \)'s execution regions, that is, \( C_k = \sum_{j=1}^{m} C_{k,j} \)."

Although Theorem 2 is correct, Claim 1 is not. It is demonstrated with a counter-example below.

Counter-Example 1. Assume the task set composed of three tasks \( \tau_1 = ((1), 4, 4, 0) \), \( \tau_2 = ((1, 9, 1), 29, 29, 0) \) and \( \tau_3 = ((3, 5, 3), 100, 100, 0) \). \( \tau_1 \) has the highest priority and \( \tau_3 \) the lowest. We are interested in computing the WCRT of \( \tau_3 \).

Since \( \tau_1 \) does not self-suspend we get \( \tau'_1 = \tau_1 \) and using the definition provided in Claim 1, we get \( \tau'_2 = \langle (2), 29, 29, J_2 \rangle \) where \( J_2 = WCRT_2 - C_2 = WCRT_2 - 2 \). Since the minimum inter-arrival time of \( \tau_1 \) is smaller than the suspension time of \( \tau_2 \), task \( \tau_1 \) generates the worst-case interference when it is released synchronously with each execution region of \( \tau_2 \) (see Figure 1(b)). In which case, we get WCRT_2 = 13 and thus \( J_2 = 13 - 2 = 11 \).

Figure 1(a) depicts one of the release patterns that generates the WCRT of \( \tau_3 \) when executed concurrently with the modified tasks \( \tau'_1 \) and \( \tau'_2 \). In that execution scenario, the WCRT of \( \tau_3 \) is 16. Indeed, due to its large inter-arrival time, task \( \tau'_2 \) can interfere at most once with \( \tau_3 \) since, even considering its release jitter, the earliest possible release for its second job is at time \( T_2 - J_2 = 18 \) (see Figure 1(a)).

Figure 1(b) shows the WCRT of \( \tau_3 \) when it executes concurrently with the actual tasks \( \tau_1 \) and \( \tau_2 \). As it can be seen, the WCRT of \( \tau_3 \) is in fact 17, thus contradicting the claim that \( \tau_2 \) can “safely be replaced by” \( \tau'_2 \) in the WCRT analysis of \( \tau_3 \).

Note that Counter-Example 1 does not invalidate Theorem 2. Tasks \( \tau_2 \) and \( \tau'_2 \) cause the same amount of interference to \( \tau_3 \). In fact, Theorem 2 is correct. However, Theorem 2 does not prove Claim 1. Theorem 2 defines an upper bound on the worst-case interference generated by one self-suspending task (i.e., either neglecting the impact of the other tasks or assuming that the WCRT is already known). Claim 1 however claims an upper bound on the interference generated by a set of self-suspending tasks.

The main issue with Theorem 2 is that it does not tell us how the interference of a task such as \( \tau_2 \) is distributed between the execution regions of a lower priority task (in this case \( \tau_3 \)). However, as shown in Counter-Example 1, the interference distribution is of prime importance to compute a valid upper bound on the WCRT of \( \tau_3 \) since it directly impacts the number of jobs of other tasks (\( \tau_1 \) in this case) that can interfere with \( \tau_3 \).

II. SOLUTION

The error in Claim 1 is to model the whole self-suspending task \( \tau_k \) as a single non-self-suspending task \( \tau'_k \). In fact, each
execution region $\tau_{k,j}$ of $\tau_k$ should be modelled by a different non-self-suspending task $\tau_{k,j}'$ with jitter $J_{k,j}'$. Such solution was already proposed in [2]. In [2], the jitter $J_{k,j}$ is given by the difference between the WCRT and the best-case response time (BCRT) of the partial self-suspending task composed of the $(j-1)$ first execution and suspension regions of $\tau_k$.

**Lemma 4.** Let $\tau_{k,j}$ be the $j$th execution region of $\tau_k$, and let $\tau_k^j$ be a self-suspending task composed of the $(j-1)$ first execution and suspension regions of $\tau_k$, that is, $\tau_k^j = \langle C_{k,1}, S_{k,1}, \ldots, C_{k,j-1}, S_{k,j-1}, D_k, T_k \rangle$. The release jitter of $\tau_{k,j}$ is upper bounded by $\tau_{k,j}^j \text{ def } = \text{WCRT}_k^j - \text{BCRT}_k^j$, where WCRT$_k^j$ and BCRT$_k^j$ are the worst-case and best-case response time of $\tau_k^j$, respectively.

**Proof.** The minimum inter-arrival time of the execution region $\tau_{k,j}$ of $\tau_k$ is inherited from the minimum inter-arrival time of $\tau_k$. However, the execution region $\tau_{k,j}$ can start to execute only when the $(j-1)$th suspension region of $\tau_k$ completes, that is, when the partial self-suspending task $\tau_k^j$ completes its execution. Since the response time of $\tau_k^j$ may vary between different jobs released by $\tau_k$, the release of $\tau_{k,j}$ experiences a jitter. This jitter is upper bounded by the difference between the longest and the shortest response time of $\tau_k^j$, i.e., it is upper bounded by the difference between WCRT$_k^j$ and BCRT$_k^j$.

Let $\text{hp}(\tau_{ss})$ be a set of self-suspending tasks with higher priorities than $\tau_{ss}$. And let $\text{hp}(\tau_{ss})'$ be a set of non-self-suspending tasks where for each task $\tau_k \in \text{hp}(\tau_{ss})$, the set $\text{hp}(\tau_{ss})'$ contains $m_k$ non-self-suspending tasks $\tau_{k,j}' = \langle C_{k,j}, D_k, T_k, J_{k,j} \rangle$ with $1 \leq j \leq m_k$, where $J_{k,j}$ is defined as in Lemma 4 and each task $\tau_{k,j}' (1 \leq j \leq m_k)$ has the same priority as $\tau_k$. We prove below that replacing $\text{hp}(\tau_{ss})$ with $\text{hp}(\tau_{ss})'$ in the WCRT analysis of $\tau_{ss}$ provides a response time upper bound which is at least as large as the WCRT when using $\text{hp}(\tau_{ss})$. Therefore, replacing $\text{hp}(\tau_{ss})$ with $\text{hp}(\tau_{ss})'$ is safe.

We first define what is a legal release pattern for a task set.

**Definition 1** (Legal release pattern for a task set $\tau$). A release pattern $\mathcal{R}$ defines all the instants at which each execution region of the tasks in $\tau$ releases jobs. A release pattern $\mathcal{R}$ is legal if all the constraints defined by the tasks in $\tau$ (i.e., minimum inter-arrival time, precedence constraints and release jitter) are respected in $\mathcal{R}$.

Now, we prove that the release pattern of the task set $\text{hp}(\tau_{ss})$ that generates the WCRT of $\tau_{ss}$ can be transformed in a legal release pattern for the tasks in $\text{hp}(\tau_{ss})'$.

**Lemma 5.** Let $\mathcal{R}$ be any legal release pattern of the execution regions of the tasks in $\text{hp}(\tau_{ss})$ such that the tasks in $\text{hp}(\tau_{ss})$ generate the worst-case interference on $\tau_{ss}$. Let $\mathcal{R}'$ be a release pattern for the tasks in $\text{hp}(\tau_{ss})'$ such that whenever an execution region $\tau_{k,j} \in \text{hp}(\tau_{ss})$ releases a job in $\mathcal{R}$, the corresponding task $\tau_{k,j}'$ releases a job at the same instant in $\mathcal{R}'$. The release pattern $\mathcal{R}'$ is a legal release pattern for the tasks in $\text{hp}(\tau_{ss})'$.

**Proof.** We have to prove that the minimum inter-arrival times, release jitters and precedence constraints defined for the task in $\text{hp}(\tau_{ss})'$ are all respected in $\mathcal{R}'$.

1) The minimum inter-arrival time of $\tau_{k,j}$ is $T_k$ and its release jitter is smaller than or equal to $J_{k,j}$ (from Lemma 4). Let $\tau_{k,j}'$ be the $\ell$th instance (job) released by $\tau_{k,j}$. Since $\mathcal{R}$ is legal, the time between any two jobs $\tau_{k,j}'$ and $\tau_{k,j}'^{+p}$ released by $\tau_{k,j}$ is at least $(p \times T_k) - J_{k,j}$. Therefore, the time between any two jobs $\tau_{k,j}'$ and $\tau_{k,j}'^{+p}$ released by $\tau_{k,j}'$ is at least $(p \times T_k) - J_{k,j}$ in the release pattern $\mathcal{R}'$. Since by definition, the minimum inter-arrival time and the release jitter constraints on $\tau_{k,j}'$.

2) Since the tasks in $\text{hp}(\tau_{ss})'$ do not have any precedence constraints, the release pattern $\mathcal{R}'$ trivially respects those constraints.

By 1. and 2., the release pattern $\mathcal{R}'$ is legal for $\text{hp}(\tau_{ss})'$.

We finally prove that replacing $\text{hp}(\tau_{ss})$ by $\text{hp}(\tau_{ss})'$ in the WCRT analysis of $\tau_{ss}$ is safe.

**Theorem 3.** The worst-case interference generated by the tasks in $\text{hp}(\tau_{ss})'$ is lower bounded by the worst-case interference generated by the tasks in $\text{hp}(\tau_{ss})$.

**Proof.** The proof is based on the following facts:

F1. If a job of $\tau_{k,j}$ or $\tau_{k,j}'$ interferes with the execution region $\tau_{ss,p}$ of $\tau_{ss}$ than it does not interfere with any other execution region of $\tau_{ss}$. This statement is true because (i) both $\tau_{k,j}$ and $\tau_{k,j}'$ have a higher priority than $\tau_{ss}$, and (ii) they do not self-suspend. Therefore, when they start to interfere with one execution region of $\tau_{ss}$, that execution region cannot resume its execution before $\tau_{k,j}$ or $\tau_{k,j}'$ complete their own execution.

F2. When they execute for their WCET, one job of $\tau_{k,j}$ generates as much interference as one job of $\tau_{k,j}'$. It is simply due to the fact that $\tau_{k,j}$ and $\tau_{k,j}'$ have the same WCET.

Let $\mathcal{R}$ be any legal release pattern of the execution regions of the tasks in $\text{hp}(\tau_{ss})$ such that the tasks in $\text{hp}(\tau_{ss})$ generates the worst-case interference on $\tau_{ss}$. And let $\mathcal{R}'$ be the corresponding release pattern for the tasks in $\text{hp}(\tau_{ss})'$ such that whenever an execution region $\tau_{k,j}$ of a task $\tau_k \in \text{hp}(\tau_{ss})$ releases a job in $\mathcal{R}$, the corresponding task $\tau_{k,j}'$ releases a job at the same instant in $\mathcal{R}'$. By Lemma 5, $\mathcal{R}'$ is a legal release pattern for the tasks in $\text{hp}(\tau_{ss})'$. Since by Fact F2., each job released by each task $\tau_{k,j}'$ generates as much interference than each job released by the corresponding execution region $\tau_{k,j}$, and because by Fact F1., this interference is generated in the same execution region of $\tau_{ss}$, the total interference generated by the set of tasks in $\text{hp}(\tau_{ss})'$ under the release pattern $\mathcal{R}'$ is equal to the worst-case interference generated by the corresponding self-suspending tasks in $\text{hp}(\tau_{ss})$ under $\mathcal{R}$.
Therefore, because we proved that there exists at least one legal release pattern of the tasks in \( hp(\tau_{ss})' \) generating as much interference as the worst-case interference generated by \( hp(\tau_{ss}) \), the worst-case interference generated by the tasks in \( hp(\tau_{ss})' \) is lower bounded by the worst-case interference generated by the tasks in \( hp(\tau_{ss}) \).

**Theorem 4.** The WCRT of \( \tau_{ss} \) running concurrently with \( hp(\tau_{ss})' \) is no smaller than its WCRT when it runs concurrently with \( hp(\tau_{ss}) \).

**Proof.** Theorem 3 proves that \( hp(\tau_{ss})' \) generates at least as much interference on \( \tau_{ss} \) than \( hp(\tau_{ss}) \). Therefore, the WCRT of \( \tau_{ss} \) when it runs concurrently with \( hp(\tau_{ss})' \) is no smaller than its WCRT when it runs concurrently with \( hp(\tau_{ss}) \). ■

### A. Upper Bounding \( J_{k,j} \)

The solution presented above requires an upper bound on the jitter \( J_{k,j} \) experienced by each execution region \( \tau_{k,j} \). In this section, we provide three different upper bounds (stated in Lemmas 6, 7 and 8) on the jitter \( J_{k,j} \).

**Lemma 6.** The release jitter \( J_{k,j} \) of \( \tau_{k,j} \) is upper bounded by \( WCRT_k - \sum_{p=j}^{m_k} C_{k,p} - \sum_{p=j}^{m_k-1} S_{k,p} \).

**Proof.** Let \( a_k \) and \( f_k \) be the release time and the completion time of any job of \( \tau_k \), and let \( a_{k,j} \) be the release time of the execution region \( \tau_{k,j} \) in that job. Instant \( a_{k,j} \) also corresponds to the completion time of the partial self-suspending task \( \tau_{k,j}^l \). We prove that \( a_{k,j} \) is no later than \( a_k + WCRT_k - \sum_{p=j}^{m_k} C_{k,p} - \sum_{p=j}^{m_k-1} S_{k,p} \).

The proof is by contradiction. Let us assume that the completion of \( \tau_{k,j} \), and hence the release of \( \tau_{k,j} \), happens after \( a_k + WCRT_k - \sum_{p=j}^{m_k} C_{k,p} - \sum_{p=j}^{m_k-1} S_{k,p} \) that is,

\[
ak_{k,j} > a_k + WCRT_k - \sum_{p=j}^{m_k} C_{k,p} - \sum_{p=j}^{m_k-1} S_{k,p} \quad (1)
\]

If every execution region executes for its worst-case execution time and every suspension region suspends for its worst-case suspension time, then \( \tau_k \) must still execute for \( \sum_{p=j}^{m_k} C_{k,p} \) time units and suspend for \( \sum_{p=j}^{m_k-1} S_{k,p} \) time units after \( a_k \). Therefore, even without interference from higher priority tasks, task \( \tau_k \) completes its execution at time

\[
f_k \geq a_{k,j} + \sum_{p=j}^{m_k} C_{k,p} + \sum_{p=j}^{m_k-1} S_{k,p}
\]

Replacing \( a_{k,j} \) with Eq. (1), we get

\[
f_k > a_k + WCRT_k - \sum_{p=j}^{m_k} C_{k,p} - \sum_{p=j}^{m_k-1} S_{k,p} + \sum_{p=j}^{m_k} C_{k,p} + \sum_{p=j}^{m_k-1} S_{k,p}
\]

Simplifying and passing \( a_k \) from the right hand side to the left-hand side, we obtain

\[
f_k - a_k > WCRT_k
\]

which is a clear contradiction with the fact that \( WCRT_k \) is an upper bound on the response time of \( \tau_k \). It results that for any job of \( \tau_k \), the partial self-suspending task \( \tau_{k,j}^l \) completes at time \( a_{k,j} \leq a_k + WCRT_k - \sum_{p=j}^{m_k} C_{k,p} - \sum_{p=j}^{m_k-1} S_{k,p} \). The worst-case response time \( WCRT_k \) is therefore upper bounded by \( WCRT_k - \sum_{p=j}^{m_k} C_{k,p} - \sum_{p=j}^{m_k-1} S_{k,p} \).

Since the best-case response time \( BCRT_k \) of \( \tau_{k,j} \) is trivially lower bounded by 0, the jitter \( J_{k,j} \), which by definition is equal to \( WCRT_k - BCRT_k \), is upper bounded by \( WCRT_k - \sum_{p=j}^{m_k} C_{k,p} - \sum_{p=j}^{m_k-1} S_{k,p} \). ■

**Lemma 7.** The release jitter \( J_{k,j} \) of \( \tau_{k,j} \) is upper bounded by \( \sum_{p=1}^{j-1}(UB_{k,p} + S_{k,p}) \) where \( UB_{k,p} \) is an upper bound on the WCRT of each execution region \( \tau_{k,p} \) given by the smallest positive \( t \) such that

\[
t = C_{k,p} + \sum_{\tau \in hp(\tau_{k,p})} \left[ t + \frac{J_{k,j}}{T_k} \right] C_{\ell}
\]

**Proof.** It was proven in [3] that the WCRT of a self-suspending task \( \tau_{j,i} \) is upper bounded by \( \sum_{j=1}^{p-1}(UB_{k,j} + S_{k,p}) \).

Since \( J_{k,j} \) is defined as \( WCRT_k - BCRT_k \), and because \( BCRT_k \) is lower bounded by 0, we get that \( J_{k,j} \leq \sum_{p=1}^{j-1}(UB_{k,p} + S_{k,p}) \). ■

**Lemma 8.** The release jitter \( J_{k,j} \) of \( \tau_{k,j} \) is upper bounded by \( UB_{k,j} + S_{k,j-1} \) where \( UB_{k,j} \) is given by the smallest positive \( t \) such that

\[
t = \sum_{p=1}^{j-1} C_{k,p} + \sum_{p=1}^{j-2} S_{k,p} + \sum_{\tau \in hp(\tau_{k,p})} \left[ t + \frac{J_{k,j}}{T_k} \right] C_{\ell}
\]

**Proof.** It was proven in [3] that the WCRT of a self-suspending task \( \tau_{j,i} \) is upper bounded by \( UB_{k,j} + S_{k,j-1} \). Because the last suspension region \( S_{k,j-1} \) of \( \tau_{j,i} \) cannot be preempted, the WCRT of \( \tau_{j,i} \) is given by \( UB_{k,j} + S_{k,j-1} \). Since \( J_{k,j} \) is defined as \( WCRT_k - BCRT_k \), and because \( BCRT_k \) is lower bounded by 0, we get that \( J_{k,j} \) is upper bounded by \( UB_{k,j} + S_{k,j-1} \). ■

### III. Discussion

Using Theorem 4, each higher priority self-suspending task can be transformed in a set of non-self-suspending tasks with jitter. One can therefore use the MILP formulation proposed in [1], which computes an upper bound on the WCRT a self-suspending task \( \tau_{ss} \) running concurrently with a set of non-self-suspending tasks with jitter.

For the convenience of the reader, we reproduce below the MILP formulation.
Maximize: \( \sum_{j=1}^{m_{ss}} R_{ss,j} \) \hspace{1cm} (2)

Subject to:
\[ \sum_{j=1}^{m_{ss}} R_{ss,j} + \sum_{j=1}^{m_{ss}-1} S_{ss,j} \leq UB_{ss} \] \hspace{1cm} (3)

\[ \forall \tau_{ss,j} \in \tau_{ss} : R_{ss,j} = C_{ss,j} + \sum_{\tau_p \in hp(\tau_{ss})} NI_{p,j} \times C_p \] \hspace{1cm} (4)

\[ R_{ss,j} \leq UB_{ss,j} \] \hspace{1cm} (5)

\[ \forall \tau_k \in hp(\tau_{ss})', \forall \tau_{ss,j} \in \tau_{ss} : \]
\[ O_{k,j} \geq -J_k \] \hspace{1cm} (6)

\[ O_{k,j+1} \geq O_{k,j} + NI_{k,j} \times T_k - (R_{ss,j} + S_{ss,j}) - J_k \] \hspace{1cm} (7)

\[ NI_{k,j} \geq 0 \] \hspace{1cm} (8)

\[ NI_{k,j} \leq \left \lfloor \frac{R_{ss,j} - O_{k,j}}{T_k} \right \rfloor \] \hspace{1cm} (9)

where
\[ rel_{k,j} \overset{\text{def}}{=} O_{k,j} + (NI_{k,j} - 1) \times T_k \]
\[ dp_{p,j} \overset{\text{def}}{=} O_{p,j} + NI_{p,j} \times T_p \]

and where \( UB_{ss} \) is an upper bound on the WCRT of \( \tau_{ss} \) given by the smallest positive \( t \) such that
\[ t = \sum_{j=1}^{m_{ss}} C_{ss,j} + \sum_{j=1}^{m_{ss}-1} S_{ss,j} + \sum_{\tau_p \in hp(\tau_{ss})} \left \lfloor \frac{t + J_p}{T_p} \right \rfloor C_p \]

and \( UB_{ss,j} \) is an upper bound on the WCRT of each execution region \( \tau_{ss,j} \) given by the smallest positive \( t \) such that
\[ t = C_{ss,j} + \sum_{\tau_p \in hp(\tau_{ss})} \left \lfloor \frac{t + J_p}{T_p} \right \rfloor C_p \]

Finally, an upper bound on the WCRT of \( \tau_{ss} \) is given by
\[ \sum_{j=1}^{m_{ss}} R_{ss,j} + \sum_{j=1}^{m_{ss}-1} S_{ss,j} \]

where \( \sum_{j=1}^{m_{ss}} R_{ss,j} \) is the solution to the MILP formulation.

A. Impact on Other Results in [1]

At the exception of Claim 1, none of the other results presented in [1], including the experimental section, are impacted by the error reported in this errata.

B. Impact on Related Work

To the best of the authors’ knowledge, three papers [4]–[6] building on top of [1] were published recently. As far as the authors can tell, the results in those papers were not affected by the error reported in this technical report.

REFERENCES


