Improved Deadline Miss Models for Real-Time Systems using Typical Worst-Case Analysis

Wenbo Xu, Zain A. H. Hammadeh, Alexander Kröller, Rolf Ernst
Technische Universität Braunschweig, Germany

Sophie Quinton
Inria Grenoble – Rhône-Alpes, France

***Abstract*—We focus on the problem of computing tight deadline miss models for real-time systems, which bound the number of potential deadline misses in a given sequence of activations of a task. In practical applications, such guarantees are often sufficient because many systems are in fact not hard real-time. Our major contribution is a general formulation of that problem in the context of systems where some tasks occasionally experience sporadic overload. Based on this new formulation, we present an algorithm that can take into account fine-grained effects of overload at the input of different tasks when computing deadline miss bounds. Finally, we show in experiments with synthetic as well as industrial data that our algorithm produces bounds that are much tighter than in previous work, in sufficiently short time.**

***Keywords*—real time systems, performance analysis, deadline misses, weakly-hard guarantees**

I. Introduction

When analyzing real-time systems, formal worst-case analysis is an established technology, e.g. in automotive design [1]. However, worst-case bounds can be overly pessimistic, especially for mass production solutions with weak or weakly-hard real-time constraints: in many cases (e.g. related to human perception), it is admissible to miss some deadlines[1] without jeopardizing the system's behavior. On the other hand, probabilistic approaches make strong assumptions on independence and stationarity which are rarely met in practice. In this context, Typical Worst-Case Analysis (TWCA) [2] constitutes a new approach with promising opportunities, based on considering typical bounds for a first analysis and then handling exceptional behaviors separately. The outcome of TWCA is a set of weakly-hard bounds — i.e., bounds which can be violated at most a given number of times in a sequence — on response times of tasks and more recently on deadline misses as well [3].

This approach is particularly interesting because it answers a need in current industrial practice. It is customary, e.g., in CAN bus design, to analyze a system under typical load conditions. For example, the cyclic bus load for CAN buses corresponds to a system in which all spontaneous messages are ignored. TWCA bridges a gap, as there was previously no formal way to quantify the significance of the computed bounds, see e.g. [4]. It has also been shown in [5] that the weakly-hard guarantees provided by TWCA can be used e.g. for efficient verification of closed-loop properties of control software.

In this paper, we improve on the work presented in [3] that leverages TWCA to compute deadline miss models for tasks running on a single resource according to the static-priority preemptive (SPP) or non-preemptive (SPNP) scheduling policy. This was an answer to the problem of providing bounds on how many deadlines a task might miss (in the worst case) out of k consecutive activations. Our major contribution is a general formulation of that problem in the context of systems where some tasks occasionally experience sporadic overload. Based on this new formulation, we present an improved algorithm that can take into account fine-grained effects of overload at the input of different tasks when computing deadline miss bounds. In addition, we improve, in the SPP case, the criterion proposed in [3] for establishing how much overload can be tolerated in a time window while still guaranteeing absence of deadline misses: our new criterion is a necessary and sufficient condition (as opposed to the sufficient condition of [3]) and therefore yields better results. Our experiments with synthetic as well as industrial data show that our algorithm produces bounds that are much tighter than in previous work, in sufficiently short time.

The paper is organized as follows: Section II discusses related work. Section III introduces our system model and formulates the problem that we address. Then, Section IV presents the basics of TWCA before Sections V, VI and VII describe the core contribution of our paper. Finally, Section VIII shows our experimental results while Section IX proposes some conclusions.

II. Related Work

Typical Worst-Case Analysis was introduced in [2], and later extended to better handle sporadic bursts [6]. It is based on the standard response-time analysis that is sometimes called *busy window analysis* — see [7] for the SPP case, [8] for the SPNP case, and e.g. [2] and [9] for the corresponding extensions to arbitrary activation patterns.

[1] In this work we are interested in bounding the number of potential deadline misses assuming a deadline-agnostic scheduler, meaning that the scheduler always starts and lets all tasks run to completion independent of any deadline misses.

1068-3070/15 $31.00 © 2015 IEEE
DOI 10.1109/ECRTS.2015.29

TWCA is tightly related to the concept of *weakly-hard* systems [10], which guarantees that out of k consecutive executions of a task, not more than m deadline misses may occur. Unfortunately, the approach of [10] and the related articles rely on an analysis of the system hyper-period and are therefore restricted in practice to periodic tasks.

Surprisingly, there is little work on the analysis of weakly-hard systems. Another recent work [11] addresses issues similar to [2] in the context of Real-Time Calculus [12]. In that paper, *rare events* represent the possible deviation from the *nominal timing model* (corresponding to the typical worst-case of our paper). The presented analysis computes the *settling-time*, i.e. the longest time window after the rare event until the system returns to normal. The main difference to TWCA is that [11] assumes that only one task may experience rare events and excludes that a second rare event may occur before the first one has settled. We are particularly interested in systems in which this is not the case.

III. System Model and Problem Statement

We consider single-resource real-time systems using a static priority scheduler, either with preemption (Static-Priority Preemptive, SPP) or without (Static-Priority Non-Preemptive, SPNP). We assume given a set of n tasks $\tau_1, \ldots, \tau_n$ competing for the resource, which can be for example a processor or a bus. For simplicity of notation, we assume that the tasks are already given in order of their static priority, i.e., τ_j has higher priority than τ_i for every $j < i$. Denote by $\mathcal{T} := \{1, \ldots, n\}$ the set of tasks (respectively, their indices). Every task τ_j, $j \in \mathcal{T}$, is further defined by its worst-case execution time c_j, its relative deadline D_j and its activation model as we discuss below. The response time of τ_j is the maximum time between an activation of τ_j and its completion, and an activation of τ_j is said to *miss its deadline* if it has not been fully processed D_j time after it has occurred. We assume a deadline-agnostic scheduler that will always start and let all tasks run to completion even if some task misses its deadline.

A. Task activation models

To model the possible activations of τ_j, we assume given arrival curves as in e.g. Real-Time Calculus, i.e., functions $\eta_j^-, \eta_j^+ : \mathbb{N} \to \mathbb{N}$ such that for any time window ΔT, $\eta_j^+(\Delta T)$ defines the maximum number of activations of τ_j that might occur, and $\eta_j^-(\Delta T)$ the minimum. This is particularly useful to model precisely activation patterns of sporadic tasks, less conservatively than using a periodic activation pattern based on the minimum interarrival time.

In the spirit of Typical Worst-Case Analysis (TWCA), we further assume that the activation model is more detailed and that the system designer has classified task activations as either *typical* or *overload*. For example, τ_j could be a periodic task (typical activations) that sometimes is triggered through some external event (overload activations). As a result, we assume a model $(\eta_j^{+typ}, \eta_j^{-typ})$ for the typical activations, and $(\eta_j^{+over}, \eta_j^{-over})$ for overload. To better distinguish the models, we refer to (η_j^+, η_j^-) as the *worst-case model*, $(\eta_j^{+typ}, \eta_j^{-typ})$ as the *typical model* and $(\eta_j^{+over}, \eta_j^{-over})$ as the *overload model*.

Note that in practice, these activation models can be derived from application models and/or execution traces. Unlike probabilistic approaches, TWCA does not depend on the accuracy of event distributions in such a trace, but only requires that the worst case event arrival bounds occur at least once for each individual task, irrespective in which phase they might occur. This is the case in typical industrial test scenarios where the applications functions are tested one-by-one. In addition, we have been able to identify tight models for both typical and overload activations in the context of a CAN bus case study [4].

We will need the following additional notations in the paper. The inverse of the activation model (η_j^+, η_j^-) is denoted by so-called minimum distance functions $\delta_j^-, \delta_j^+ : \mathbb{N} \to \mathbb{N}$ such that for any sequence of k consecutive activations of τ_j, $\delta_j^-(k)$ (respectively $\delta_j^+(k)$) defines the minimum (respectively maximum) time that might pass between the first and the last activation. The functions δ_j^{-typ}, δ_j^{+typ}, δ_j^{-over}, and δ_j^{+over} are defined analogously.

B. Problem Definition

Our goal is to compute a *deadline miss model* (DMM) for task τ_i, that is, a function $dmm_i : \mathbb{N} \to \mathbb{N}$, with the property that out of any k consecutive activations of τ_i, at most $dmm_i(k)$ might miss their deadline D_i. This bound does not depend on the choice of typical and sporadic models, rather it is a correct worst-case bound on deadline misses. Obviously we are interested in small values for $dmm_i(k)$, the optimal DMM is the function defining exactly how many deadline misses might occur in the worst case.

IV. Introduction to Typical Worst-Case Analysis

As the rest of this paper assumes familiarity with TWCA, we present in this section the state of the art in this domain. First, let us recall basics of standard response-time analysis.

A. Busy-Window Analysis

A *level-i busy window* (originally called busy period in [13]) is a maximal time interval during which the resource still has activations of tasks of equal or higher priority than τ_i pending. The longest such window, called *worst-case level-i busy window* and denoted BW_i, is built by assuming the occurrence of a so-called *critical instant*, where τ_i and higher-priority tasks are all activated at the same time, inducing maximum interference with τ_i, as shown in Figure 1. In SPNP, the task with the largest execution time among lower-priority tasks is activated just before the critical instant and therefore induces the maximum blocking time

b_i. It is also assumed that all tasks are activated as early as possible after the critical instant, and that they always use their maximum execution time. The maximum level-i busy window stops at the first instant when no activation of τ_i or any higher priority task remains incomplete. To compute BW_i, we determine the earliest possible time $w_i(q)$ at which the q-th activation of τ_i can be started:

$$w_i(q) = \min\{w \geqslant 0 : \quad w = b_i + (q-1)c_i + \sum_{j<i} \eta_j^+(w)c_j\} \ , \tag{1}$$

where b_i is the maximum blocking time from lower-priority tasks, i.e., 0 for SPP and $\max_{j>i} c_j$ for SPNP. This can be used to determine the maximum number of activations of τ_i in a level-i busy window as

$$K_i := \min\{q \geqslant 1 : \quad w_i(q+1) \leqslant \delta_i^-(q+1)\}.$$

K_i is the smallest number such that the resource would be able to start processing the (K_i+1)-th activation before this activation can occur according to δ_i, which implies an idle time. The worst-case level-i busy window can then be determined as $BW_i = w_i(K_i+1)$ for SPP and SPNP.

For a task τ_i and $k \geqslant 1$, the *multiple event busy time*, denoted $B_i(q)$, represents the maximum time it may take to process q activations of τ_i within a level-i busy window starting with the first of these q activations. For $1 \leqslant q \leqslant K_i$:

$$B_i(q) = w_i(q+1) \text{ (SPP)}, \tag{2}$$

respectively

$$B_i(q) = w_i(q) + c_i \text{ (SPNP)}. \tag{3}$$

The response time of τ_i is bounded by

$$WCRT_i = \max_{1 \leqslant q \leqslant K_i} \{B_i(q) - \delta_i^-(q)\} \tag{4}$$

We refer the reader to [7] for detailed explanations about the SPP case, [8] for the SPNP case, and e.g. [2] and [9] for the corresponding extensions to arbitrary activation patterns.

B. Principle of TWCA

Consider as an example Figure 1 representing a CAN bus with three messages (tasks) τ_1, τ_2 and τ_3, scheduled according to SPNP. Task τ_1 is a "mixed" task, i.e., activated periodically with additional overload activations. In the model that we use for describing mixed tasks overload activations do not delay subsequent periodic activations, so the typical activations follow a fully periodic pattern. τ_2 is sporadic and τ_3 periodic. The response time of τ_3 is 9 ms in the worst-case scenario while it is at most 4 ms in absence of overload. In this paper we are interested in cases where such a task may miss its deadline due to overload activations, while being schedulable in absence of overload.

Let $\bar{c} \subseteq \mathcal{T}$ be a set of tasks. By $RT_i^{\bar{c}}$ we denote the worst possible response time of τ_i, assuming only the tasks in $\bar{c}$ may experience sporadic overload activations, whereas all other tasks follow their typical model. In this context, $\bar{c}$ is called a *combination*. $WCRT_i := RT_i^{\mathcal{T}}$ is the usual worst-case response time of τ_i, and $TWCRT_i := RT_i^{\varnothing}$ is called the *typical worst-case response time*.

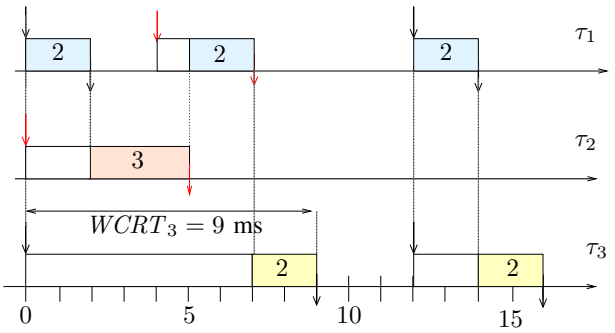


Figure 1: Busy window analysis for a CAN bus with 3 messages (tasks). The "↓" symbols represent activations and completions. White boxes are blocking times, colored boxes are execution times. The worst-case activation models are: for τ_1, $\delta_1^-(2) = 4$ ms and $\delta_1^-(3) = 12$ ms; for τ_2, $\delta_2^-(2) = 50$ ms; and for τ_3, $\delta_3^-(2) = 12$ ms. The worst-case execution times are: $c_1 = 2$ ms, $c_2 = 3$ ms and $c_3 = 2$ ms. This yields a worst-case response time for τ_3 of 9 ms. Besides, here the second activation of τ_1 and the only activation of τ_2 are classified as overload activations; the typical models for τ_3 and the periodic activations of τ_1 have a period equal to 12 while τ_2 is not activated in the typical model. As a result, if no overload occurs the response time of τ_3 cannot be larger than 4 ms.

In the following, we assume that $TWCRT_i \leqslant D_i \ \forall i$, i.e., no deadline misses occur as long as the system experiences no overload, but $WCRT_i > D_i$ for some task τ_i. Our goal is to quantify how often such deadline misses may happen, in the form of a DMM. For simplicity we will suppose that τ_i itself does not experience sporadic overload.

C. Deadline Miss Model Computation

We now show how deadline miss models are constructed in the state of the art [3]. Let us consider a sequence of k activations of a given task τ_i and compute a DMM value for k. Note that the sequence may span multiple busy windows. For all tasks $j < i$, we first need to bound the maximum time window $\Delta T_k^{j \to i}$, during which activations of τ_j might have an impact on the considered k activations of τ_i. Note that activations in one level-i busy window cannot influence the response time of τ_i in another level-i busy window. As a result, only activations of τ_j occurring at most BW_i time before the first activation of τ_i in the k-sequence and before the last activation finishes (SPP) respectively starts (SPNP) can have an impact. This time interval is thus bounded by:

$$\Delta T_k^{j \to i} = BW_i + \delta_i^+(k) + WCRT_i \text{ (SPP)}, \tag{5}$$

respectively

$$\Delta T_k^{j \to i} = BW_i + \delta_i^+(k) + QD_i \text{ (SPNP)}. \tag{6}$$

where the queuing delay

$$QD_i := \max_{1 \leqslant q \leqslant K_i} \{w_i(q) - \delta_i^-(q)\} \tag{7}$$

is the maximum delay between activation and start of a task under SPNP scheduling.

Assume that task τ_i is schedulable in the typical model, i.e., $TWCRT_i \leqslant D_i$, but unschedulable in the worst case. This implies that, out of the K_i activations of τ_i in the worst-case level-i busy window, some are missing their deadlines. These activations are denoted by $\mathcal{N}_i \subseteq \{1, \ldots, K_i\}$, and bounded with $N_i := |\mathcal{N}_i|$, which can be determined by the busy window analysis. Hence, $1 \leqslant N_i \leqslant K_i$.

We say that a combination of tasks $\bar{c} \subseteq \mathcal{T}$ is *schedulable* (implicitly meaning with respect to τ_i) if $RT_i^{\bar{c}} \leqslant D_i$, i.e., an activation of τ_i will not miss its deadline if only tasks in $\bar{c}$ experience overload activations in the same busy window. We denote $\mathcal{C}$ the set of all *schedulable combinations*, while $\tilde{\mathcal{C}}$ as the set of all unschedulable combinations.

Consider a schedulable combination $\bar{c} \in \mathcal{C}$. If, in addition to tasks in $\bar{c}$, a task $j \notin \bar{c}$ with $j < i$ experiences an overload activation, this potentially (but not necessarily, as $\bar{c} \cup \{j\}$ may be schedulable as well) causes deadline misses (at most N_i). The number of potential deadline misses in the sequence of k activations of τ_i can then bounded by accumulating the number of overload activations in $\Delta T_k^{j \to i}$ for every τ_j that is not in $\bar{c}$:

$$dmm_i^{\bar{c}}(k) = N_i \times \sum_{j<i, j \notin \bar{c}} \eta_j^{+over}(\Delta T_k^{j \to i}) \tag{8}$$

A better DMM can then trivially be determined by:

$$dmm_i(k) = \min_{\bar{c}} \{ dmm_i^{\bar{c}}(k) \mid \bar{c} \in \mathcal{C} \} \tag{9}$$

which can be formulated as an integer linear programming (ILP) with the objective function:

$$\textbf{min} \quad N_i \times \sum_{j<i} (1 - b_j) \times \Omega_j \tag{10}$$

where Ω_j denotes $\eta_j^{+over}(\Delta T_k^{j \to i})$, and b_j is a binary variable indicating whether $j \in \bar{c}$ or not.

Now, we are still missing a fast criterion for establishing schedulability of a combination (optimization space). The explanation of the work in [3] is based on Figure 2, in which τ_3 misses its deadline in the worst-case busy window.

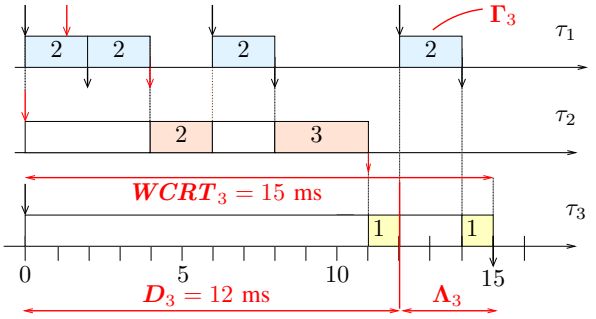


Figure 2: Illustration of Λ_3 and Γ_3 in the ILP problem for task τ_3. The second activation of τ_1 and the activation of τ_2 are overload activations. It is assumed that the activations for the typical worst-case correspond exactly to the typical activations in the worst-case busy window.

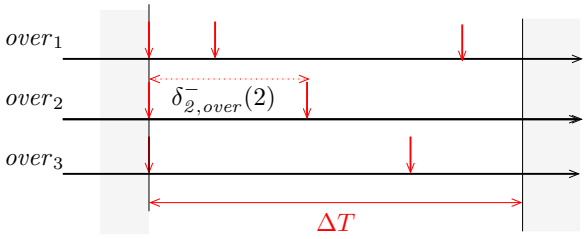


Figure 3: Overload models for tasks with higher priority than τ

Define:

$$\Lambda_i = WCRT_i - D_i$$

Λ_i is the amount by which we need to reduce the size of the busy window to avoid deadline misses (see Figure 2). In fact, some of this workload will disappear by itself if we can enforce our deadline, namely the workload resulting from activations taking place *after* D_i. More specifically, this kind of workload for the SPP case is defined as:

$$\Gamma_i = \sum_{j<i} [\eta_j^+(WCRT_i) - \eta_j^+(D_i)] \times c_j$$

It is then sufficient to remove a workload equal to $\Lambda_i - \Gamma_i$ to ensure that the deadline will be met.

The workload induced by the overload of a task τ_j before the deadline and contributing to the worst-case response time of τ_i is $\eta_j^{+over}(D_i) \times c_j$. We can therefore define our constraint as:

$$\textbf{such that} \sum_{j<i} wl_{over}^j \times (1 - b_j) \geqslant \Lambda_i - \Gamma_i \tag{11}$$

where wl_{over}^j denotes $\eta_j^{+over}(D_i) \times c_j$.

Equations (10) and (11) represent the state of the art of computing deadline miss models.

D. Limitations of the state of the art

Actually, by reasoning on several combinations at the same time one can come up with a better solution than the one of [3]. Let us show this here on an example.

Assume that we want to compute the number of potential deadline misses for a sequence of executions of a low priority task τ_i corresponding to the given time window ΔT. Figure 3 displays the overload models for the only three tasks with priority higher than τ_i which have some sporadic overload. Assume that the set of schedulable combinations is $\mathcal{C} = \{\{1\}, \{2\}, \{3\}, \emptyset\}$, namely only one task having overload is tolerable. The approach of [3] would come up with an DMM of $\min_{\bar{c}}\{dmm_i^{\bar{c}}(k)\} = 4$ with $\bar{c} = \{1\}$.

Note, however, that in this example a deadline miss may happen only if at least two tasks experience overload in the same level-i busy window, according to the schedulable combinations. If we pair overload activations of different tasks, the deadline misses can happen here at most 3 times. In other words, if we consider several combinations in ΔT, we can improve, possibly drastically, on the results of [3].

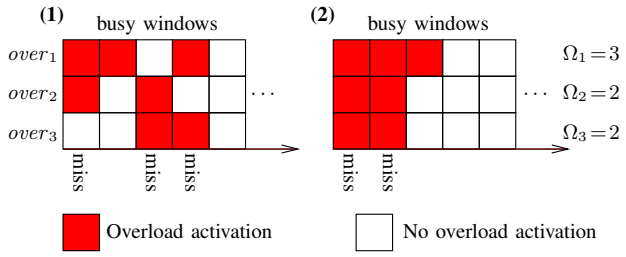


Figure 4: Assume as in Section IV-D that any two tasks in $\{\tau_1, \tau_2, \tau_3\}$ experiencing overload in the same busy window lead to a deadline miss of low-priority τ_i. Matrices (1) and (2) are two possible packings of overload activations into busy windows, each line being constrained by the Ω_j of its corresponding task τ_j. (1) is the scenario that maximizes the number of misses and as a result represents a safe deadline miss bound.

V. Improved Deadline Miss Model Computation

Our objective now is to come up with an efficient solution that takes the above mentioned dependencies into account. For this, we need to reformulate our problem.

Let us fix k and focus on the computation of $dmm_i(k)$. The number of overload activations of each higher priority task is limited by its activation model $(\eta_j^{+over}, \eta_j^{-over})$. For $j < i$, let $\Omega_j := \eta_j^{+over}(\Delta T_k^{j \to i})$, as in Equation (10). Remember that, in busy windows where no overload activation occurs, τ_i does not miss any deadline, so let us consider only busy windows in which there is at least one overload activation — there are at most $\sum_{j<i} \Omega_j$ such busy windows.

Revisiting the problem in Figure 3, and assuming we have all unschedulable combinations at hand, we want to pack as many of them as possible into the 7 possible busy windows, as illustrated in Figure 4. Therefore the problem becomes a multi-dimensional knapsack problem. The items correspond to unschedulable combinations and capacities are bounded by Ω_j for line j corresponding to the overload of τ_j. Based on this intuition, we now formulate our first main result.

Theorem 1. *The ILP represented by* (12) *— at the top of next page — provides a DMM for* τ_i.

Proof. As shown in Figure 4, in order to make some activation of τ_i miss its deadline, other tasks forming an unschedulable combination need to experience overload activations in the same level-i busy window. In the worst case, up to N_i out of the K_i activations during that busy window miss their deadlines. By setting $x_{\bar{c}}$ to the number of times the combination $\bar{c} \in \tilde{\mathcal{C}}$ is used to impact a busy window in this way, $N_i \sum_{\bar{c} \in \tilde{\mathcal{C}}} x_{\bar{c}}$ is an upper bound to the number of deadline misses within the relevant time window. As there are at most Ω_j overload activations of τ_j within that window, we can limit how often a combination $\bar{c}$ may be used. □

It has to be pointed out that overload activations may occur so frequently that every busy window could be impacted. In this situation, Theorem 1 still gives a correct answer, but might exceed the trivial value, namely k — as at most k deadlines may be missed out of a sequence of k activations. In practice, if a result bigger than the trivial one is obtained, the latter one will be chosen.

Besides, note that $dmm_i(k)$ suffers from counting N_i for every impacted busy window. This could be improved by computing an actual count of deadline misses for a given $\bar{c} \in \tilde{\mathcal{C}}$. This would complicate a crucial algorithm component (Equation (25) below), so we leave exploring this possibility for future work.

The following sections are dedicated to the issue of computing efficiently $dmm_i(k)$ as defined in Theorem 1.

VI. Improved Criterion for SPP Schedulability

First, let us focus on efficiently establishing schedulability of a combination. We improve here over the sufficient condition proposed in [3] and repeated in Equation (11) by presenting a *necessary and sufficient* condition for this in the SPP case. For the more complicated SPNP case, we rely on the result of [3].

As in [3], we will assume that the activations in the maximum busy window for the typical worst-case correspond exactly to the typical activations in the maximum busy window for the worst case. This does not necessarily hold in general but it does for the activation patterns that we are interested in, namely sporadic and mixed tasks (even for analyses taking *offsets* into account [14]). As a result, this is not a restriction in practice.

Let us define, for every $\ell \in \mathcal{N}_i$ and $j < i$:

$$\rho_i^\ell(t) := \sum_{j<i} \eta_j^+(t) c_j + \ell c_i - t \qquad \forall t \leqslant B_i(\ell) \quad (13)$$

$$wl_{over}^j(t) := \eta_j^{+over}(t) c_j \qquad \forall t \leqslant BW_i \quad (14)$$

Intuitively, $\rho_i^\ell(t)$ refers to the workload in the worst-case level-i busy window that remains to be processed at time t before the ℓ-th activation of τ_i finishes. In contrast, $wl_{over}^j(t)$ is the accumulated amount of overload from τ_j until time t. Note that these values do not consider the activations arriving at or after t. If a task τ_j is assumed to be in the typical case, i.e. $j \notin \bar{c}$, then all the overload due to τ_j is removed and therefore the workload that remains to be processed at time t before the ℓ-th activation of τ_i finishes in the level-i busy window corresponding to $\bar{c}$ is $\rho_i^\ell(t) - wl_{over}^j(t)$. Hence the following lemma.

Lemma 1. *For a combination* $\bar{c}$*, the* ℓ*-th activation of* τ_i *has been fully processed at time* t *if and only if there is at least a* t' *between its activation time (excluded) and* t *(included) such that:*

$$\rho_i^\ell(t') - \sum_{j<i,\ j \notin \bar{c}} wl_{over}^j(t') \leqslant 0. \quad (15)$$

Proof. As explained above, this condition means that the workload remaining to be processed at instant t' is smaller than the overload removed from the typical-only tasks. □

$$dmm_i(k) := \max \left\{ N_i \sum_{\bar{c} \in \tilde{\mathcal{C}}} x_{\bar{c}} \ : \sum_{\bar{c}:\, j \in \bar{c} \in \tilde{\mathcal{C}}} x_{\bar{c}} \leqslant \Omega_j \ \forall j < i,\ x_{\bar{c}} \in \mathbb{N} \ \forall \bar{c} \in \tilde{\mathcal{C}} \right\} . \tag{12}$$

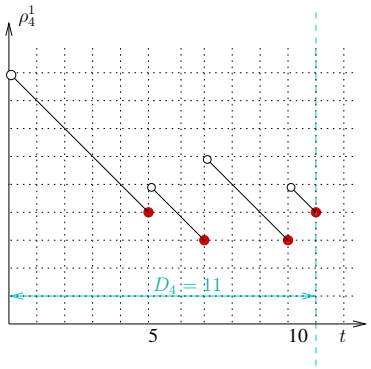


Figure 5: The $\rho_4^1(t)$ function corresponding to Figure 6. To make τ_4 finish before its deadline, we have to remove more workload than what remains to process at some time point. $\rho(t)$ is a piecewise function: Each discontinuity corresponds to one or several higher priority activations. Within each piece, it decreases monotonically while $wl_{over}^j(t)$ remains the same. It is sufficient to analyze only the end point of each piece, i.e. the local minima.

Based on this result, we can now define our criterion for schedulability.

Lemma 2. *Let $\bar{c} \subseteq \mathcal{T}$, $i \in \bar{c}$. Then $\bar{c}$ is schedulable, if and only if the following inequality set holds:*

$$\forall \ell \in \mathcal{N}_i \ \exists t \in \mathcal{P}_i^\ell : \sum_{j<i,\ j \notin \bar{c}} wl_{over}^j(t) \geqslant \rho_i^\ell(t) \tag{16}$$

where

$$\mathcal{P}_i^\ell := \{\delta_i^-(\ell) < t \leqslant \delta_i^-(\ell) + D_i \mid \textit{a higher priority task is activated at } t \textit{ in the worst case } \} \cup \{\delta_i^-(\ell) + D_i\} \tag{17}$$

Proof. $\mathcal{P}_i^\ell$ corresponds to all the activation time points (from higher priority tasks) between the ℓ-th activation of τ_i and its deadline, as well as the deadline point. Figure 5 is an example of a worst-case activation graph and its $\rho(t)$ function, where $\mathcal{P}_i^\ell$ is marked using colored dots.

1) If $\bar{c}$ is schedulable, then (16) holds: if $\bar{c}$ is schedulable, according to Lemma 1, for every activation ℓ of τ_i, there must be a time point t before or equal to τ_i's deadline, so that (15) holds. Note also that $\rho_i^\ell(t)$ is a piecewise function, and is monotonically decreasing within each piece (Figure 5), while $wl_{over}^j(t)$ remains unchanged within this time interval because there is no activation at all. So if (15) holds for t then it also holds for the right most-point of the piece where t stays, which is in $\mathcal{P}_i^\ell$.

2) If (16) holds, then $\bar{c}$ is schedulable: (16) implies that for every activation ℓ of τ_i, there is at least a time point for which (15) holds. According to Lemma 1, every activation can finish before their deadlines, which means $\bar{c}$ is schedulable. □

Consequently, we can deduce that the set

$$\tilde{\mathcal{C}} := \{\bar{c} \subseteq \mathcal{T} : \exists \ell \in \mathcal{N}_i \ \forall t \in \mathcal{P}_i^\ell : \sum_{j<i,\ j \notin \bar{c}} wl_{over}^j(t) < \rho_i^\ell(t)\} \tag{18}$$

consists of all unschedulable combinations.

Example. Let us now illustrate the improvement of our new criterion over that presented in [3] on the example in Figure 6. If we follow [3] and use the criterion in (11), $\bar{c} = \{1, 3\}$, where the overload at the input of τ_2 is ignored, is regarded as not schedulable because $\Lambda_4 - \Gamma_4 = 4$ while the left side of (11) is 3. In fact $RT_4^{\bar{c}} = 10$ ms and $\bar{c}$ is schedulable. It turns out that, with our necessary and sufficient criterion, we can determine that $\bar{c}$ is schedulable because $wl_{over}^2(10) \geqslant \rho_4^1(10)$, since $wl_{over}^2(10) = \rho_4^1(10) = 3$.

VII. An efficient ILP solution

As mentioned, finding $dmm_i(k)$ means solving a multi-dimensional knapsack problem over the item set $\tilde{\mathcal{C}}$. Unfortunately, $\tilde{\mathcal{C}} \subseteq 2^{\mathcal{T}}$ may have exponential size, which means (12) is not only NP-hard [15], but also might have an exponential size input. Hence directly computing $dmm_i(k)$ is impractical. Instead, we consider the following LP relaxation

$$dmm_i'(k) := \max \ N_i \sum_{\bar{c} \in \tilde{\mathcal{C}}} x_{\bar{c}} \tag{19}$$

$$\text{s.t.} \sum_{\bar{c}:\, j \in \bar{c} \in \tilde{\mathcal{C}}} x_{\bar{c}} \leqslant \Omega_j \quad \forall j < i \tag{20}$$

$$x_{\bar{c}} \geqslant 0 \qquad \forall \bar{c} \in \tilde{\mathcal{C}} \ . \tag{21}$$

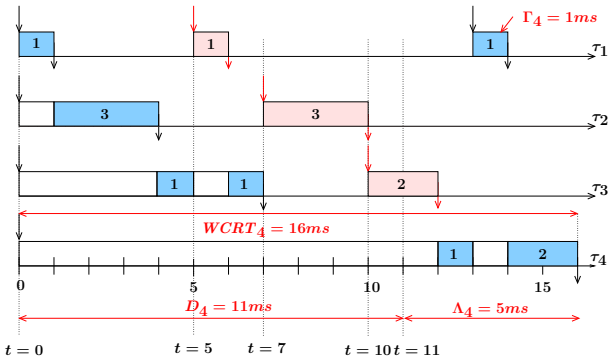


Figure 6: An example under SPP where our new criterion guarantees schedulability while [3] fails. We use different colors for representing executions due to typical load and overload.

Trivially $dmm'_i(k) \geqslant dmm_i(k)$, so this defines a DMM as well. In the following, we sketch an algorithm based on column generation that solves $dmm'_i(k)$, producing upper bounds in the process. The dual linear program of (19)–(21) reads as follows:

$$\min \sum_{j<i} \Omega_j y_j \tag{22}$$

$$\text{s.t.} \sum_{j\in\bar{c},\, j<i} y_j \geqslant N_i \quad \forall \bar{c} \in \tilde{\mathcal{C}} \tag{23}$$

$$y_j \geqslant 0 \qquad \forall j \in \mathcal{T} \ . \tag{24}$$

We analyze solutions where $\tilde{\mathcal{C}}$ was reduced to a smaller sample $Z \subseteq \tilde{\mathcal{C}}$. Initially, this could be $Z := \{\mathcal{T}\}$ ($\mathcal{T}$ is guaranteed to be unschedulable). Let x^* be an optimal primal solution and y^* be an optimal dual solution for this reduced LP, with objective value $z^* := N_i \sum_{\bar{c}} x^*_{\bar{c}}$. In the interest of generating columns, we seek to find a violated constraint of the complete dual LP by computing

$$\nu := \min_{\bar{c}\in\tilde{\mathcal{C}}} \sum_{j\in\bar{c},\, j<i} y^*_j \ . \tag{25}$$

The following lemma shows how to identify whether z^* is a DMM (a single value for k):

Lemma 3. *If $\nu \geqslant N_i$, then x^* is optimal for* (19)–(21)*, and $z^* \geqslant dmm_i(k)$.*

Proof. If the condition holds, then y^* is feasible for (22)–(24). By LP duality, it follows that

$$z^* = dmm'_i(k) \geqslant dmm_i(k)$$

□

This lemma allows us to deduce a DMM when column generation finishes, i.e., a globally optimal solution has been found. As this might take a long time due to the exponential size of $\tilde{\mathcal{C}}$, we seek to construct a DMM from suboptimal solutions as well. We define

$$dmm''_i(k) := \frac{N_i}{\nu} z^* \ . \tag{26}$$

Lemma 4. *If $0 < \nu \leqslant N_i$, then the obtained function dmm''_i is a DMM.*

Proof. From the construction of ν (25), it follows that

$$\sum_{j\in\bar{c},j<i} \frac{N_i}{\nu} y^*_j \geqslant N_i$$

for every $\bar{c} \in \tilde{\mathcal{C}}$. Consequently, $\frac{N_i}{\nu} y^*$ is a feasible solution for (22)–(24), with objective value $\frac{N_i}{\nu} z^*$. Using weak LP duality, we get $dmm''_i \geqslant dmm'_i \geqslant dmm_i$, so dmm''_i is indeed a DMM. □

Algorithms. Before we can describe the overall algorithm, we discuss a subroutine to compute ν. Following the characterization of $\tilde{\mathcal{C}}$ in Section VI, we can rewrite (25), given y^*, as follows:

$$\nu = \min \sum_{j<i} y^*_j q_j \tag{27}$$

$$\text{s.t. } \exists \ell \in \mathcal{N}_i \ \forall t \in \mathcal{P}^\ell_i : \sum_{j<i} wl^j_{over}(t) q_j \leqslant \rho^\ell_i(t) - \varepsilon \tag{28}$$

$$q_j \in \{0, 1\} \tag{29}$$

using a sufficiently small $\varepsilon > 0$.

The overall algorithm to compute the DMM $dmm''_i(k)$ is shown in Algorithm 1.

Algorithm 1: Get the approximate upper bound of LP (19)–(21)

```
Z := {T}, dmm := k
repeat
    Solve LP (19)–(21) (with Z instead of C̃), obtain
    primal solution x*, dual solution y*, objective
    value z*
    Compute ν according to (25), let c̄ be the
    combination attaining the minimum
    if ν ⩾ N_i then
        dmm := z*
    else
        Z := Z ∪ {c̄}
        if ν > 0 and (N_i/ν) z* < dmm then dmm := (N_i/ν) z*
    output upper bound dmm
until z* = dmm
```

Theorem 2. *Algorithm 1 produces a decreasing sequence of correct deadline miss bounds.*

Proof. The algorithm solves a sequence of (19)–(21), using dual separation to increase the combination pool Z, until no additional columns can be generated. The initial deadline miss bound k is obviously correct. Lemmas 3 and 4 provide that all values assigned to dmm later are valid. □

This theorem ensures us a suboptimal but correct DMM by terminating the algorithm at any iteration. That is meaningful when dealing with a too complicated system and only requiring an approximated result.

VIII. Experiments

To evaluate the improvement brought by our new approach, we first tested it on the examples proposed in [3], over which we directly claim to improve. In addition, we built a set of synthetic examples to obtain a more general impression on the quality of our approach for a large variety of systems. Finally, we have also applied our approach to a case study presented in [4]. All these experiments are described in the rest of this section.

k	10	100	1000
$dmm_{15}(k)$ ([3])	6	12	21
$dmm_{15}(k)$	3	7	13

Table I: DMMs obtained using [3] and this paper

A. Comparison with [3]

The synthetic example used in [3] consists of 15 tasks, out of which 5 have a periodic activation pattern, 5 are sporadic and 5 are mixed (i.e., periodic with sporadic additional activations) — more information is available in [3].

The usual worst-case response time analysis of tasks shows that the lowest priority task τ_{15} is the only one that may miss its deadline, as $WCRT_{15} = 149$ ms exceeds deadline $D_{15} = 100$ ms. How often this may happen is not given by worst-case analysis, hence the use of TWCA.

Table I shows the deadline miss bounds computed using the ILP approach of [3] for various values of k and those obtained using the approach of this paper. As one can see, even for this simple example the improvement is quite important, as the deadline miss bounds are about 1.8 times smaller with the new approach.

B. Synthetic test cases

In this section, we present a set of synthetic test cases that we have developed to test more extensively our approach on a variety of systems. Particularly, we are interested in comparing results for systems with different characteristics such as utilization, system size, etc.

Model Generation. We need to define a set of tasks — some periodic, others sporadic, or mixed — with a priority, a worst-case execution time, a period (for periodic and mixed tasks), and a minimum distance function (for sporadic and mixed tasks). We only study systems whose busy window analysis converges, as the others cannot be used for TWCA (or even for standard worst-case analysis).

The standard approach is to first define the system utilization and then assign a share of it to each task [16]. However, we want to model sporadic (and mixed) tasks using minimum distance functions rather the usual periodic model. To handle this, we first decide on the utilization to be shared among sporadic tasks. We then apply UUnifast [16] separately to the sporadic and periodic tasks, which are allocated what is left of the global utilization. Note that there is no standard approach to generating minimum distance functions for sporadic tasks, especially as they must be super-additive, i.e., $\delta^-(a+b) \geqslant \delta^-(a) + \delta^-(b)$ for all a, b [17]. We therefore propose the following approach based on trace generation. First, we use the fact that, by analogy with periodic tasks:

$$U_j = \lim_{w \longrightarrow \infty} \eta_j^+(w) c_j / w$$

In practice, we define $\delta_j^-(M) = Mc_j/U_j$, with a sufficiently high M ($M = 100$ suffices for all of our experiments). Then, we randomly generate $M-2$ activation times between 0 and $\delta_j^-(M)$. A trace analysis provides the observed minimum distance functions.

For better analyzability of the results, we choose a limited set of possible values for e.g. utilization, number of sporadic tasks, sporadic overload, among which actual values will be chosen. Deadlines are arbitrary. Additionally, we have chosen to consider only systems such that there is at most one overload activation per task in the worst-case level-i busy window for the task τ_i under analysis. It is in practice a realistic assumption in many contexts and we want to ignore in our evaluation the pessimism induced by this factor.

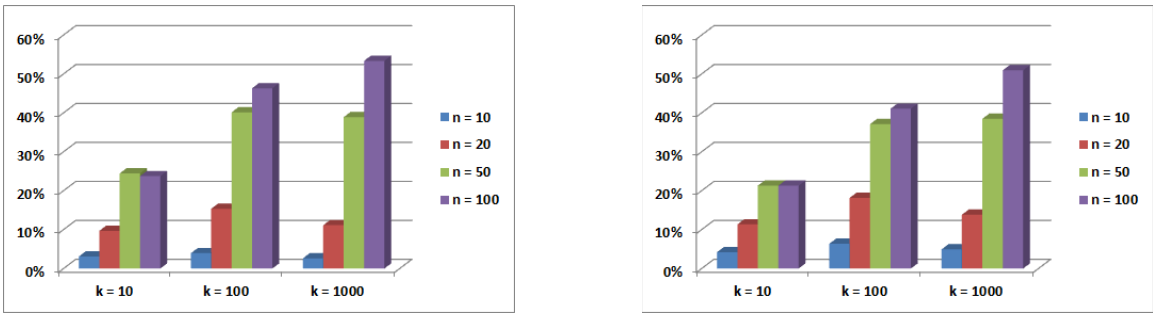
Results. We generated 750 systems as above for every number of tasks $n \in \{10, 20, 50, 100\}$ respectively, with a system utilization of 50% or 80% — so 3000 systems in total. For each system we calculated $dmm_i(k)$ for $k \in \{10, 100, 1000\}$. For the first 1000 systems, periods are randomly chosen among $\{500, 1000, 2000, 5000, 10000\}$ as this fits our experience with industrial data. To estimate whether this introduces a bias, another set of 1000 systems has been generated with harmonic periods and a third set with random periods. Results are similar for the three sets (with slighly more regular and better results for arbitrary periods). We focus here on the first, more realistic set.

Figures 7 and 8 show the relative improvement obtained by using our new analysis over the previous approach [3] for different system configurations. Figure 7 shows the average improvement, defined as $1 - \frac{dmm_i(k)}{dmm_i(k)[3]}$ and reads as follows: for $n = 50$ and $k = 100$ in Figure 7b (i.e for a utilization of 80%) the new bounds are on average 37% smaller than the original bounds of [3]. Theoretically our new approach does not give worse results than [3], so we report in Figure 8 the percentage of cases where there is actually an improvement ($\frac{dmm_i(k)}{dmm_i(k)\ [3]} < 1$).

Note that there is a trend indicating that the larger the system is, the more the new approach outperforms [3] on average. In addition, our approach tends to perform better for larger k, most likely because we have more busy windows to pack unschedulable combinations. On the other hand, there is no indication that system utilization has any influence on the comparative performance of our approach. Many non-trivial parameters impact DMMs, e.g. the number of activations in a busy window which may miss a deadline. This makes it very difficult to identify why the above-mentioned trends are not always verified. All in all, it is clear that our new algorithm offers a significant improvement over the state of the art.

C. Industrial case study

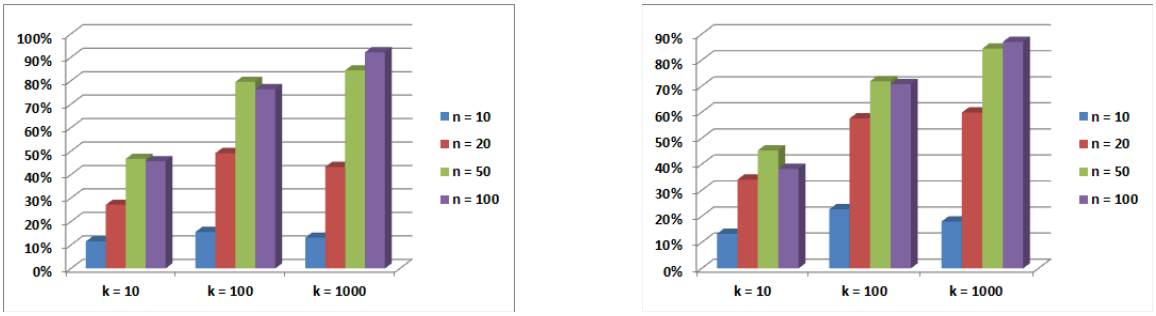
In this section we show how the new ILP solution performs compared to the one of [3] on the real-life example presented in [4].



(a) Utilization = 50%

(b) Utilization = 80%

Figure 7: Average relative improvement for a system utilization of 50% (left) and 80% (right).



(a) Utilization = 50%

(b) Utilization = 80%

Figure 8: Percentage of improved cases for a system utilization of 50% (left) and 80% (right).

System Parameters. The analyzed system is a CAN bus: it has an SPNP scheduling policy, with offsets for periodic tasks. See [4] for details on how to extract relevant parameters (including information about the frequency of sporadic overload activations for TWCA) from available data, namely network vehicle trace data used in conjunction with a configuration file (called communication matrix). There are 212 tasks appearing in the trace, out of which 76 are periodic, 17 are sporadic, 119 are mixed. In addition, [4] performs a response-time analysis of different models of the CAN bus, from a model derived only from trace data to the base-load model that is entirely based on the communication matrix, with some model in-between combining information from the trace and the communication matrix. The models that are relevant for this paper are the following.

- The so-called *combined* model is a worst-case model obtained using together information from the trace and the communication matrix.
- The *typical* model corresponds to a combination that has been chosen empirically based on the number of overload activations. [4] shows a guarantee obtained for each task based on 10,000 consecutive executions.
- Finally, the *base-load* model is what we call in this paper the typical worst-case model.

We now discuss the improvement obtained by the new algorithm. CAN tasks have implicit deadlines (i.e., deadlines are equal to periods); in order to incur deadline misses, we consider a slower bus than the original one. Three periodic tasks may now miss their deadline, of which one may miss its deadline even in the typical worst-case. We consider the remaining two tasks, to which TWCA is applicable. Table II shows the corresponding results. Note that the quality of the results is due to the fact that the deadline is quite close to the WCRT, which means that many overload activations must occur at the same time for a deadline miss to happen. The difference in the obtained DMM is due to the different level of priority of the two tasks. In addition, the high level of symmetry in the system is an advantage for our new approach, as no single combination is really better than the others. Note also that, for the task shown on at the top of Table II, the results of [3] are hardly useful while the new ones do provide hard guarantees. It is worth mentioning here that these results can be improved further by improving our criterion for schedulability also for SPNP.

IX. Conclusion

Deadline miss models, which bound the number of potential deadline misses in a given sequence of activations of a task, have been shown to be a good model for practical

Algorithm \ k	10	50	100	500	1000
$dmm(k)$ [3]	10	50	100	125	142
$dmm(k)$	6	9	10	16	20

Algorithm \ k	10	50	100	500	1000
$dmm(k)$ [3]	5	5	5	5	5
$dmm(k)$	1	1	1	1	1

Table II: Compared DMM obtained for the two relevant tasks in the case study using the approach of [3] and this paper.

applications. The reason for this is that many timing requirements in actual systems are in fact not hard and the system can survive a limited number of deadline misses.

This paper improves over previous work to compute such deadline miss models for SPP and SPNP scheduled systems. While previous work distinguished typical worst-case and worst-case pattern per task and then identified the optimum combination of cases, this paper proposed a more fine-grained implicit consideration of all possible overload situations. This is a more accurate model of the system behavior, but it significantly extends the design space. Utilizing the specific properties of the design space, we were able to come up with a solution that is computationally feasible even for larger systems. The experiments show that the new model allows for much better results than previous approaches.

X. Acknowledgements

This work was partially supported by the Deutsche Forschungsgemeinschaft (DFG) under contract numbers KR3133/2-1 and QU368/1-1 (Controlling Concurrent Change), and partially funded by Thales.

References

[1] ISO, "Road vehicles – functional safety – part 1: Vocabulary," International Organization for Standardization, Geneva, Switzerland, ISO 26262-1:2011, 2011.

[2] S. Quinton, M. Hanke, and R. Ernst, "Formal analysis of sporadic overload in real-time systems," in *DATE*. IEEE, 2012, pp. 515–520.

[3] Z. A. H. Hammadeh, S. Quinton, and R. Ernst, "Extending typical worst-case analysis using response-time dependencies to bound deadline misses," in *Proceedings of the International Conference on Embedded Software*, New Delhi, India, October 2014, to appear.

[4] S. Quinton, T. T. Bone, J. Hennig, M. Neukirchner, M. Negrean, and R. Ernst, "Typical worst case response-time analysis and its use in automotive network design," in *Proc. of DAC*. ACM, 2014, pp. 1–6.

[5] G. Frehse, A. Hamann, S. Quinton, and M. Woehrle, "Formal analysis of timing effects on closed-loop properties of control software," in *Proceedings of RTSS'14*. IEEE, 2014, pp. 53–62.

[6] S. Quinton, M. Negrean, and R. Ernst, "Formal analysis of sporadic bursts in real-time systems," in *DATE*, 2013, pp. 767–772.

[7] K. Tindell, A. Burns, and A. J. Wellings, "An extendible approach for analyzing fixed priority hard real-time tasks," *Real-Time Systems*, vol. 6, no. 2, pp. 133–151, 1994.

[8] R. Davis, A. Burns, R. Bril, and J. Lukkien, "Controller area network (CAN) schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, vol. 35, no. 3, pp. 239–272, 2007.

[9] M. Negrean and R. Ernst, "Response-time analysis for non-preemptive scheduling in multi-core systems with shared resources," in *7th IEEE International Symposium on Industrial Embedded Systems, SIES 2012, Karlsruhe, Germany, June 20-22, 2012*, 2012, pp. 191–200. [Online]. Available: http://dx.doi.org/10.1109/SIES.2012.6356585

[10] G. Bernat, A. Burns, and A. Llamosí, "Weakly hard real-time systems," *IEEE Trans. Computers*, vol. 50, no. 4, pp. 308–321, 2001.

[11] P. Kumar and L. Thiele, "Quantifying the effect of rare timing events with settling-time and overshoot." in *Proceedings of RTSS'33*, 2012, pp. 149–160.

[12] S. Chakraborty, S. Künzli, and L. Thiele, "A general framework for analysing system properties in platform-based embedded system designs," in *Proc. of DATE'03*. IEEE Computer Society, 2003, pp. 190–195.

[13] J. P. Lehoczky, "Fixed priority scheduling of periodic task sets with arbitrary deadlines," in *IEEE Real-Time Systems Symposium*, 1990, pp. 201–213.

[14] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Extending schedulability analysis of controller area network (can) for mixed (periodic/sporadic) messages." in *Proceedings of ETFA'16*, 2011, pp. 1–10.

[15] M. J. Magazine and M.-S. Chern, "A note on approximation schemes for multidimensional knapsack problems," *Mathematics of Operations Research*, vol. 9, no. 2, pp. 244–247, 1984. [Online]. Available: http://dx.doi.org/10.1287/moor.9.2.244

[16] E. Bini and G. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Systems*, vol. 30, no. 1-2, pp. 129–154, 2005. [Online]. Available: http://dx.doi.org/10.1007/s11241-005-0507-9

[17] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis — the SymTA/S approach," in *IEE Proceedings Computers and Digital Techniques*, 2005.