Extending Typical Worst-Case Analysis Using Response-Time Dependencies to Bound Deadline Misses*

Zain A. H. Hammadeh
Institute of Computer and Network Engineering (IDA)
TU Braunschweig, Germany
hammadeh@ida.ing.tu-bs.de

Sophie Quinton
Inria Grenoble - Rhône-Alpes
Montbonnot, France
sophie.quinton@inria.fr

Rolf Ernst
Institute of Computer and Network Engineering (IDA)
TU Braunschweig, Germany
ernst@ida.ing.tu-bs.de

ABSTRACT

Weakly-hard time constraints have been proposed for applications where occasional deadline misses are permitted. Recently, a new approach called Typical Worst-Case Analysis (TWCA) has been introduced which exploits similar constraints to bound response times of systems with sporadic overload. In this paper, we extend that approach for static priority preemptive and non-preemptive scheduling to determine the maximum number of deadline misses for a given deadline. The approach is based on an optimization problem which trades off higher priority interference versus miss count. We formally derive a lattice structure for the possible combinations that lays the ground for an integer linear programming (ILP) formulation. The ILP solution is evaluated showing effectiveness of the approach and far better results than previous TWCA.

1. INTRODUCTION

There exist efficient methods for computing upper bounds on the worst-case timing scenario for a given real-time system [7, 4]. However, results provided by such methods do not reflect the frequency of worst-case occurrences even though, in many practical cases, embedded system applications accept occasional deadline violations as long as their number can be bounded. Typical Worst-Case Analysis [12] (TWCA) was introduced to bound the frequency of occurrence of undesired behaviors based on the concept of weakly-hard constraints [2]. The idea is to identify system behaviors considered as typical and to consider remaining scenarios as the result of some sporadic overload. The outcome of TWCA is, for every task τ_i:

- a typical worst-case response time bound $TWCRT_i$, along with
- for every $k \in \mathbb{N}^+$, a bound on the number of executions of τ_i which may have a response time larger than $TWCRT_i$ in a window of k consecutive executions.

* This work was partially funded by Thales.

ESWEEK'14, October 12–17 2014, New Delhi, India

ACM 978-1-4503-3052-7/14/10 ...$15.00.
http://dx.doi.org/10.1145/2656045.2656059

In the current state of theory, as published in [12], TWCA can only provide the number of deviations from the typical worst case for a given configuration of tasks and their activation. That knowledge is of limited practical value: what a designer would really like to know is the number of deadline misses in a given time window. This figure would indicate the number of late arrivals of sensor samples in a control loop or of overwritten messages in an automotive CAN communication system. So, the theory has to be extended to handle deadlines. The number of deadline misses, however, does not only depend on the properties of a single task, but also on the properties of all interfering tasks.

Consider a system of tasks with fixed priority scheduling. For every task τ_i, assume that we have a worst-case activation pattern W_i and a typical-case activation pattern T_i — it is out the scope of this paper to explain how to come up with such T_i; note that it can in general be derived from the system function. The typical-case activation patterns can be used to compute for each task τ_i an error function err_i such that $err_i(k)$ describes the number of deviations from the typical worst-case behavior out of k consecutive activations. Now, given a deadline D_i for τ_i, what we want to get is a tight bound for the deadline misses err_i^D.

In this paper, we show that the determination of err_i^D can be formulated as an optimization problem. For every interfering, i.e., higher or equal priority task, we can either select the W_i pattern or the T_i pattern. The T_i pattern induces less interference, but adds to the number of errors. Then, the optimization problem is to select a combination of T_i and W_i for all higher priority tasks such that $err_i(k)$ is minimal while the deadline is met. We use an ILP approach to solve this problem. With this approach, TWCA can now be used to better solve many practical problems, where limited deadline misses are allowed. We illustrate the efficiency of this technique on a well-chosen set of synthetic examples for static priority preemptive as well as non-preemptive scheduling. We show that the results are dramatically better than earlier work [12] which is deadline agnostic. In particular, we study the impact of the ratio between deadline, worst-case and typical worst-case response time. We also discuss timing performance of the compared approaches.

The paper is organized as follows: Section 2 discusses related work. Section 3 motivates our work on an example. Section 4 recalls the main principles of TWCA. Section 5 presents a first improvement on the error computation taking deadlines into account. Section 6 then shows how we extend the state of the art using combinations of worst-case and typical-case models while Section 7 shows our ILP solution to make the problem manageable in practice. Finally, Section 8 presents experimental results while Section 9 concludes.

2. RELATED WORK

This work is an improvement of TWCA as introduced in [12]. The latter was extended by [13] to better handle sporadic bursts. TWCA is based on the concept of *weakly-hard constraints* [2], which guarantee that out of k consecutive executions of a task, not more than m deadline misses may occur. Although the results are in a similar form, the approach of [2] and the related articles rely on an analysis of the system hyper-period (i.e. the least common multiplier of all task periods) and is therefore restricted in practice to periodic tasks.

A recent work [8] addresses issues similar to [12] in the context of Real-Time Calculus [4]. In that paper, *rare events* represent the possible deviation from the *nominal timing model* (corresponding to the typical worst-case of [12]). The presented analysis computes the *settling time*, i.e. the longest time window after the rare event until the system returns to normal. In addition, the *overshoot* during the settling time quantifies how many deadline misses may then occur. This can easily be translated into a weakly-hard guarantee. The main difference between this line of work and TWCA is that [8] only considers one source of rare event at a time (only one task may experience rare events) and excludes that a second rare event may occur before the first one has settled.

Finally, another somewhat related research topic is that of probabilistic schedulability analysis [10, 3]. Probabilistic approaches work with distributions on execution times, inter-arrival times etc. in order to come up with distributions on response times, which can be turned into probabilistic guarantees on deadline misses. Such approaches are mostly useful when the various parameters are random and independent. We focus on systems in which this is not the case and the worst-case dependencies must be taken into account.

3. MOTIVATION

Let us explain on an example the principle of TWCA and the extension that we propose. Consider a processor with 3 tasks τ_1, τ_2 and τ_3 scheduled according to the static priority preemptive policy, and with the following priority order: $\tau_1 > \tau_2 > \tau_3$. Assume that τ_1 is a "mixed" task, i.e., it is activated periodically with some additional sporadic activations; τ_2 is a sporadic task and τ_3 is a periodic task.

Let us focus on the response time of τ_3. The standard worst-case response time analysis would build the largest level-3 *busy window* (originally called busy period [9]), that is, the maximum time interval the processor might be busy executing τ_3 or a higher-priority task, as illustrated in Figure 1 (colored boxes represent execution times, white boxes are blocking times). This yields a response time bound for τ_3 of $WCRT_3 = 15$ ms.

Now, if the sporadic activations of τ_1 and τ_2 are rare, then the above bound may in practice be reached only very seldom (if ever). Hence the idea of TWCA [12], where sporadic activations (in bold and red in Figure 1) are considered as *sporadic overload* and ignored for the response time analysis. In our example, this would lead to a "typical bound" for τ_3 of $TWCRT_3 = 4$.

There remains to formally quantify how often the typical bound may be optimistic. This is achieved using an *error model* similar to the overshoot model of [8], which is a function $err_3 : \mathbb{N}^+ \longrightarrow \mathbb{N}$ such that, for all k consecutive activations of τ_3, at most $err_3(k)$ response times may be larger than $TWCRT_3$. The formula defined in [12] to compute the error model, which we recall in Section 4, is based on the fact that the impact of an overload (sporadic) activation cannot last longer than the worst-case busy window. Roughly, the error model is obtained by computing how many overload activations of τ_1 or τ_2 may occur in a given time window and how many executions of τ_3 each of them can influence.

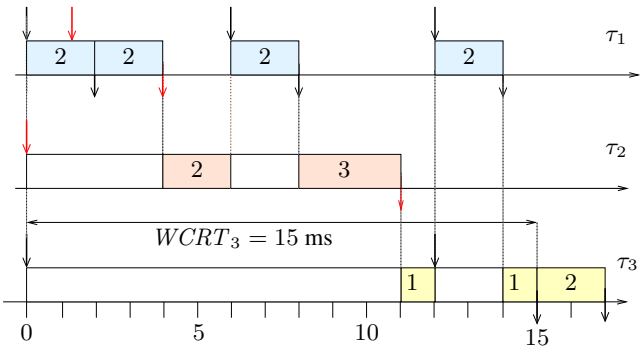

Figure 1: Worst-case busy window analysis

In this paper, we are interested in weakly-hard guarantees regarding deadline misses instead of $TWCRT$. Assume that in our example the deadline of τ_3 is equal to its period $D_3 = 12$ ms. This means that we only want to know how often the response time of τ_3 may be larger than 12 ms. Because the notion of typical worst-case is relative, two other options are possible: ignore only the sporadic activations of τ_1 in the busy window analysis, or only those of τ_2. This would yield respectively typical bounds on the response time of τ_3 equal to 11 ms and 6 ms. These bounds are larger than $TWCRT_3$ but still below the deadline. However, because our two new options regard fewer activations as overload, they will result in better error models that the original one. As a result, by taking the deadline into account and combining worst-case and typical worst-case models of tasks, we are able to significantly improve our error model. In Section 6, we explain in detail how we come up with the combination that results in the best error model.

4. TWCA: STATE OF THE ART

In this section, we give the definitions related to TWCA that will be needed in the rest of the paper.

We consider systems made of a set of tasks (i.e. software components) mapped onto a single hardware component (e.g. a processor, a memory or a bus) called resource, on which they execute in the order decided by a given scheduling policy. In this paper, we consider the static-priority preemptive (SPP) [14] and the static-priority non-preemptive (SPNP) [5] policies. These define a strict order between tasks by assigning them a priority such that higher-priority tasks execute first and may also, in the preemptive case, interrupt the execution of lower-priority tasks. In the sequel, we use for tasks the notation τ_i with $i \in \{1, \ldots, n\}$ (n being the total number of tasks).

4.1 Activation traces and event models

The execution of a task is triggered by an input event called *activation*. An activation trace describes the set of instants (described as natural numbers) at which a given task is activated. Formally, an activation trace act is an increasing (possibly infinite) sequence of instants where $act(n) \in \mathbb{N}$ denotes the time of the n-th activation in the trace. As is often the case [7, 4], sets of traces are

abstractly represented as *event models* using either distance functions (δ^-, δ^+) or arrival curves (η^-, η^+). A trace act *satisfies* an event model if:

- $\delta^-(n)$ bounds the minimum size of a time interval containing n activations in act — respectively the maximum size for $\delta^+(n)$;

- $\eta^+(\Delta t)$ bounds the maximum number of activations that may occur in a time interval of size Δt in act — respectively the minimum number for $\eta^-(\Delta t)$.

4.2 Worst-case response time analysis

We assume that each task τ_i is assigned a relative *deadline* D_i defining a constraint on the *response time* of τ_i, i.e., how much time (at most) may elapse until an activation of τ_i has been fully processed. Therefore, to find out if a task may miss a deadline, we can analyze its worst-case response time. Note that if a deadline is missed, the corresponding job (that is, processing of an activation) is not stopped but proceeds until completion, possibly delaying other jobs. The worst-case response time analysis that we use is based on the notion of busy window [9].

Definition 1. *A level-i busy window is a maximal time interval during which the resource still has jobs of tasks of priority higher or equal to τ_i pending.*

The maximum level-i busy window for a task τ_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 — and therefore induce maximum interference with τ_i, while 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[1]. It is also assumed that all tasks are activated as early after the critical instant as possible (according to their activation model) and that they always use their maximum execution time. The maximum level-i busy window stops at the first instant when no job of τ_i or any higher priority task remains incomplete.

To improve readability, we provide here only definitions, for SPNP then for SPP. The proofs are available in [5] (see [1] for a generalization to event models) while proofs for SPP can be found in [14] and e.g. [12]. In fact, TWCA applies to any scheduler whose response time bounds can be found using a busy window analysis.

We first iteratively compute the maximum time $w_i(q)$ that a task may have to wait from the beginning of a busy window until its q-th activation starts executing.

Theorem 1. *For $q \in \mathbb{N}^+$:*

$$w_i(q) = b_i + (q-1) \times WCET_i + \sum_{j \in hp(i)} \eta_j^+(w_i(q)) \times WCET_j$$

where

$$b_i = \max_{j \in lp(i)} \{WCET_j\}$$

and

- *$WCET_i$ is the worst-case execution time of τ_i*

- *$hp(i)$, respectively $lp(i)$, is the set of tasks with higher, respectively lower, priority than τ_i*

[1]Note that this is relevant only for the non-preemptive case.

- *$\eta_j^+(\Delta t)$ is the maximum number of activations of task τ_j in a time window of size Δt.*

The size of a level-i busy window is bounded by $BW_i = w_i(K_i)$ where $K_i = \min\{q \geq 1 \mid w_i(q+1) \leq \delta_i^-(q+1)\}$. That is, 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 possibly occur according to δ_i^-, which implies an idle time.

The maximum level-i busy window BW_i is a key element in our analysis. Informally, it indicates the maximum duration that the impact of an activation can last: after a duration of BW_i, the resource will necessarily have been idle at some point, and therefore whatever happened before the idle time does not matter anymore. Note that K_i represents the maximum number of activations of τ_i which can be in the same busy window.

For a task τ_i and $k \geq 1$, 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 $1 \leq q \leq K_i$:

$$B_i(q) = w_i(q) + WCET_i$$

The worst-case response time and worst-case delay are then obtained by comparing, for each activation in the maximum level-i busy window, their finishing time (respectively start of execution time) to their activation time.

Theorem 2. *The response time of τ_i is bounded by*

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

In addition, the queuing delay*, i.e. the maximum time that a task may have to wait between the moment it gets activated and the moment it starts executing, is bounded by*

$$QD_i = \max_{1 \leq q \leq K_i} \{w_i(q) - \delta_i^-(q)\} \tag{2}$$

Note that for each q as above, $B_i(q) - \delta_i^-(q)$ bounds the response time of the q-th activation in a busy window.

For SPP, the following changes must be applied to the formulas:

- The blocking factor b_i disappears in the definition of $w_i(q)$.

- The definition of $B_i(q)$ must be changed to take into account blocking time due to preemptions after the start of the execution of τ_i: with SPP, $B_i(q) = w_i(q+1)$.

4.3 Typical worst-case response time analysis

TWCA [12] relies on the decomposition of some activation models into a typical part and a non-typical element called *overload model*. A busy window analysis is then performed using the typical activation models (that is, ignoring the overload activations) while the overload model is used to determine the significance of the obtained response time bound for each task τ_i, denoted $TWCRT_i$.

Formally, the relation between worst-case, typical worst-case and overload model is expressed as follows.

Definition 2. *An event model (η^-, η^+) can be decomposed into a*

typical model $(\eta^-_{typ}, \eta^+_{typ})$ *and an* overload model $(\eta^-_{over}, \eta^+_{over})$ *if the three event models are such that any trace satisfying* (η^-, η^+) *can be decomposed into a trace satisfying* $(\eta^-_{typ}, \eta^+_{typ})$ *and another one satisfying* $(\eta^-_{over}, \eta^+_{over})$.

In practice we are only interested in η^+_{typ} (respectively η^+_{over}), and not in η^-_{typ} (respectively η^-_{over}). So we use in the sequel the term typical (respectively overload) model to refer only to the maximum arrival curve, which we simply denote η^i_{typ} (respectively η^i_{over}) for a task τ_i.

There is no restriction to what can be a typical model. Note, however, that a typical model that would be less tight than the worst case, meaning that it accepts more traces, is useless. We therefore ignore this case. There is, in addition, a trade-off to be found between typical and overload model in the sense that choosing a tighter typical model (so that fewer activations are considered as typical) is likely to lead to a less tight (i.e. worse) overload model.

We now define the key concept for quantifying the quality of a typical model, namely the *error model*.

Definition 3. *Considering a task* τ_i, *an* error model *for* $TWCRT_i$ *is a function* $err_i : \mathbb{N}^+ \longrightarrow \mathbb{N}$ *such that* $err_i(k)$ *bounds the number of executions of* τ_i *which may have a response time larger than* $TWCRT_i$ *in a window of* k *consecutive executions of* τ_i.

We call such a window a *k-sequence*. It is shown in [12] and [13] that, given worst-case, typical worst-case and overload models, err_i can be obtained as follows. For clarity, we adopt a presentation similar to that of [13]. As overload may appear at the input of multiple tasks, $err_i(k)$ is conservatively defined as the sum of the error models induced by single tasks. That is,

$$err_i(k) = \sum_{j \in hpe(i)} err_i^j(k) \tag{3}$$

where $err_i^j(k)$ denotes the error induced by overload at the input of τ_i itself or at the input of a task τ_j with higher priority than τ_i[2]. Note that taking the sum of the task-specific errors is conservative because each $err_i^j(k)$ is computed (as explained below) assuming that whenever an overload activation occurs at the input of τ_j, the system may experience its worst-case behavior: that is, $err_i^j(k)$ takes into account possible overload at the input of other tasks.

The computation of $err_i^j(k)$ can be summarized as follows.

1. We first compute the impact of an overload activation at the input of τ_j, that is, how many response times of τ_i may be larger than $TWCRT_i$ because of this overload activation.

As already mentioned, an activation cannot influence the response time of activations which are in other busy windows. An overload activation can therefore impact at most K_i response times of τ_i, i.e., the maximum number of activations of τ_i which may be in the same busy window.

2. We then compute the time interval during which an overload activation at the input of τ_j may have an impact on the response times in the considered k-sequence of τ_i.

[2]Remember that for the sake of simplicity we assume in this paper a strict priority order between tasks.

This time interval is bounded by:

$$\Delta T_k^{j \to i} = \underbrace{BW_i}_{(b)} + \underbrace{\delta_i^+(k)}_{(a)} + \underbrace{D^+_{i,j}}_{(c)} \tag{4}$$

where

$$D^+_{i,j} = \begin{cases} 0 & \text{if } i = j \\ QD_i & \text{if } i \neq j \text{ and the scheduling is SPNP} \\ WCRT_i & \text{if } i \neq j \text{ and the scheduling is SPP} \end{cases}$$

Each component in the above equation is explained as follows.

(a) An overload activation that occurs during the k-sequence of activations of τ_i (see Figure 2) may have an impact on the response times in the k-sequence. The corresponding time interval is of length at most $\delta_i^+(k)$.

(b) An overload activation occurring more than BW_i before the first activation of the k-sequence (see left part of Figure 2) cannot be in the same busy window as this first activation and therefore has no impact on its response time or that of subsequent activations.

(c) If $i = j$ then an overload activation of τ_j after the last activation in the k-sequence has no impact on the response times in the k-sequence, because activations of a task are handled in a FIFO order.

If $i \neq j$, what happens after the last activation in the k-sequence starts executing if the scheduling policy is SPNP (respectively finishes executing if it is SPP) has no impact on the response times of the k-sequence (see right part of Figure 2). The maximum interval of impact after the k-sequence is then the maximum queuing delay QD_i of τ_i (respectively maximum response time $WCRT_i$).

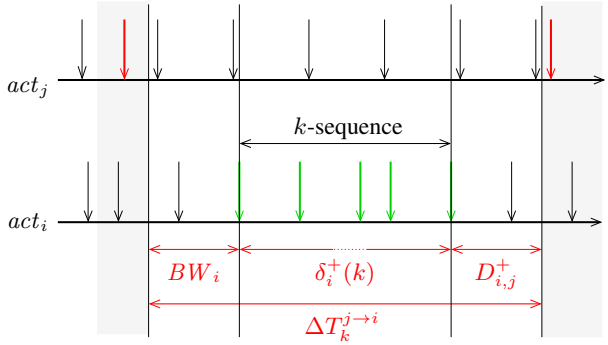

Figure 2: Impact of overload at the input of τ_j on τ_i

3. Finally we compute how many overload activations at most may occur at the input of τ_j during this time interval $\Delta T_k^{j \to i}$ and take into account the impact of each overload activation on the response times of the k-sequence. Hence the following.

$$err_i^j(k) = K_i \times \eta^j_{over}(\Delta T_k^{j \to i}) \tag{5}$$

By putting together Equations 3, 4 and 5, we can provide a the following guarantee.

Theorem 3. *For any sequence of* k *jobs of* τ_i, *at most* $err_i(k)$ *response times may be larger than* $TWCRT_i$.

Proof. This follows directly from the previous definitions. □

5. IMPROVED ERROR COMPUTATION

We have presented in Section 4 the state of the art in TWCA. Let us now adapt the concept of error model taking deadlines into account.

Definition 4. *Considering a task τ_i, an* error model *for D_i is a function $err_i^D : \mathbb{N}^+ \longrightarrow \mathbb{N}$ such that $err_i^D(k)$ bounds the number of executions of τ_i which may have a response time larger than D_i in a window of k consecutive executions of τ_i.*

One can obtain an error model for D_i based on the state of the art by taking err_i as in Equations 3, 4 and 5 under the condition that $TWCRT_i \leq D_i$.

However, if $TWCRT_i \leq D_i$, then one can come up with an improved error model for D_i. Indeed, it is considered in [12] that an overload event may result in as many deadline misses as there are activations in the maximum busy window. This is correct but may be unnecessarily pessimistic. Indeed, the busy window analysis provides more guarantees than the worst-case response time of a given task. Namely, it provides, for a given task τ_i, a guarantee on the worst-case response time of the first activation of τ_i in a busy window, of the second activation of τ_i in a busy window, etc. up to the K_i-th activation. This is useful because it may be the case that even in the worst case — that is, in the maximum busy window — only e.g. the first activation of τ_i may miss a deadline. Hence the following result.

Theorem 4. *The number of activations of τ_i which may experience a deadline miss due to a given overload event at the input of a task τ_j is bounded by*

$$N_i = \#\{q \in \mathbb{N}^+ \mid 1 \leq q \leq K_i \wedge D_i < B_i(q) - \delta_i^-(q)\} \quad (6)$$

Proof. As already mentioned, a given overload event at the input of task τ_j can only impact one busy window of τ_i. Within that busy window there are by definition at most K_i activations of τ_i. Furthermore, we know that for $q \in \{1, \ldots, K_i\}$, the response time of the q-th instance of τ_i in a busy window (if it exists) is bounded by $B_i(q) - \delta_i^-(q)$. So at most N_i activations may experience a deadline miss as a result of an overload activation of τ_j. □

Based on this, we can now replace K_i by N_i in Equation 5. As one can see, the impact of this change is to divide the error by a factor that is potentially quite large.

6. USING COMBINATIONS FOR TWCA

We now explore the possibility to choose different typical worst-case models for a task to obtain error models which are good error models for D_i. In practice, there are often obvious candidates for the typical model. For example, it seems natural for a so-called mixed task to consider its periodic activations as typical and its sporadic additional activations as overload. However, if the sporadic component is too big, this will lead to mediocre error models. In that case, it is better to stick to the original worst-case model and consider all activations as typical. In the following, we only consider for each task two options:

- a given typical worst-case and overload model as before;
- or we ignore this typical worst-case model and consider instead the worst-case model, therefore having an empty overload model.

The challenge is to handle a set of possible combinations that can be extremely large in practice.

6.1 Modeling combinations

On top of the concepts already introduced for TWCA, we now need to formalize these different *combinations*. Our objective is to consider, for a given task τ_i, all the relevant combinations of worst-case and typical worst-case activation models to compute the optimal error model of τ_i which bounds how often τ_i may miss its deadline.

Definition 5. *In a system with n tasks, a* combination *is a tuple $\bar{c} = (c_1, c_2, \ldots, c_n)$ where for all $i \in \{1, \ldots, n\}$, $c_i \in \{\mathrm{W}, \mathrm{T}\}$.*

A combination defines which activation model — the worst-case pattern W_i or the predefined typical worst-case model T_i — should be considered for every task τ_i in the busy window analysis computing the typical worst-case response time. To illustrate this, in our motivating example of Section 3, $(\mathrm{T}, \mathrm{W}, \mathrm{W})$ is the combination where the typical behavior of τ_1 is T_1, i.e. periodic, that of τ_2 is W_2, i.e. sporadic and that of τ_3 is W_3, which is periodic. This means that for TWCA based on $(\mathrm{T}, \mathrm{W}, \mathrm{W})$ only the sporadic activations of τ_1 are ignored. From now on, to distinguish between the response time bounds and error models obtained from different combinations, we introduce the following notations.

Notation 1. *We denote $RT_i^{\bar{c}}$ the worst-case response time bound for τ_i when the busy window analysis is performed on the system where activations are defined according to combination $\bar{c}$, and similarly for $\eta_{typ}^{i,\bar{c}}$, $\eta_{over}^{i,\bar{c}}$ and $err_i^{\bar{c}}$.*

Our objective is to find for a given task τ_i and $k \in \mathbb{N}^+$ the combination $\bar{c}$ such that $RT_i^{\bar{c}} \leq D_i$ — the typical worst-case guarantees that the deadline is met — and $err_i^{\bar{c}}(k)$ is minimal among combinations for which the deadline is met. Based on this, we will define a much better error model err_i^D for the deadline than before.

6.2 Properties of combinations

According to our definition, the number of possible combinations for a system with n tasks is 2^n. It is however not necessary to consider all of them, as we explain now.

Definition 6. *We say that $(c'_1, c'_2, \ldots, c'_n) \geq (c_1, c_2, \ldots, c_n)$ if and only if*

$$\forall i \in \{1, \ldots, n\} : c'_i = \mathrm{T} \implies c_i = \mathrm{T}$$

For example, in a system consisting of four tasks, we have that $(\mathrm{W}, \mathrm{T}, \mathrm{W}, \mathrm{T}) \geq (\mathrm{W}, \mathrm{T}, \mathrm{T}, \mathrm{T})$ and $(\mathrm{W}, \mathrm{T}, \mathrm{W}, \mathrm{T}) \geq (\mathrm{T}, \mathrm{T}, \mathrm{W}, \mathrm{T})$ but $(\mathrm{W}, \mathrm{T}, \mathrm{T}, \mathrm{T})$ and $(\mathrm{T}, \mathrm{T}, \mathrm{W}, \mathrm{T})$ are incomparable.

Property 5. *The set of possible combinations $\mathcal{C}$ equipped with $\geq$ defines a bounded lattice where the top element is $(\mathrm{W}, \ldots, \mathrm{W})$ and the bottom element is $(\mathrm{T}, \ldots, \mathrm{T})$.*

Proof.

- $(\mathcal{C}, \geq)$ is a partially ordered set: it is reflexive, anti-symmetric, and transitive.
- For all combinations $\bar{c}$ and $\bar{c}'$, the least upper bound $\bar{c}_\sqcap$ and greatest lower bound $\bar{c}_\sqcup$ of $\{\bar{c}, \bar{c}'\}$ are obtained by respectively choosing $\mathrm{T} \sqcap \mathrm{W} = \mathrm{W}$ and $\mathrm{T} \sqcup \mathrm{W} = \mathrm{T}$.

• $(\mathcal{C}, \geq)$ has a greatest element $(\mathrm{W}, \ldots, \mathrm{W})$ and a least element $(\mathrm{T}, \ldots, \mathrm{T})$. □

The lattice structure of the set of combinations is interesting because the order on combinations entails an order on the corresponding response times and error models.

Definition 7. *We say that a combination $\bar{c}$ is a* feasible *combination if and only if $RT_i^{\bar{c}} \leq D_i$ for this combination, otherwise the combination is said to be* infeasible.

Theorem 6. *Consider a task τ_i and two combinations $\bar{c}_1, \bar{c}_2$. If $\bar{c}_1 \geq \bar{c}_2$ then $RT_i^{\bar{c}_1} \geq RT_i^{\bar{c}_2}$ and $err_i^{\bar{c}_1} \leq err_i^{\bar{c}_2}$.*

Proof. Let $\bar{c}_1$ and $\bar{c}_2$ be combinations such that $\bar{c}_1 \geq \bar{c}_2$. Because $\bar{c}_1 \geq \bar{c}_2$, we have for every $j \in hp(i)$:

- either $\eta_{typ}^{j,\bar{c}_1} = \eta_{typ}^{j,\bar{c}_2} = \eta_{typ}^{j}$ or $\eta_{typ}^{j,\bar{c}_1} = \eta_j^+$
- either $\eta_{over}^{j,\bar{c}_1} = \eta_{over}^{j,\bar{c}_2} = \eta_{over}^{j}$ or $\eta_{over}^{j,\bar{c}_1} = 0$

As a result, we have for any time interval (ΔT):

- $\eta_{typ}^{j,\bar{c}_1}(\Delta T) \geq \eta_{typ}^{j,\bar{c}_2}(\Delta T)$
- $\eta_{over}^{j,\bar{c}_1}(\Delta T) \leq \eta_{over}^{j,\bar{c}_2}(\Delta T)$

1. $RT_i^{\bar{c}_1} \geq RT_i^{\bar{c}_2}$: The only element altered in Theorems 1 and 2 when the typical and the overload models are modified is η_j^+ which is replaced by η_{typ}^j, for all $j \in hp(i)$. Hence $RT_i^{\bar{c}_1} \geq RT_i^{\bar{c}_2}$.

2. $err_i^{\bar{c}_1} \leq err_i^{\bar{c}_2}$: Similarly, Equations 3 and 5 make it clear that increasing η_{over}^j makes the error larger. Hence $err_i^{\bar{c}_1} \leq err_i^{\bar{c}_2}$. □

Note that $RT_i^{(\mathrm{W},\ldots,\mathrm{W})} = WCRT_i$, while $err_i^{(\mathrm{W},\ldots,\mathrm{W})}(k) = 0$. On the other hand, $(\mathrm{T}, ..., \mathrm{T})$ has the minimum response time and maximum error among all combinations.

This theorem implies that if $RT_i^{\bar{c}} \leq D_i$ for a combination $\bar{c}$ (i.e., if $\bar{c}$ is a feasible combination), then one does not need to examine the combinations which are smaller than $\bar{c}$ to determine the best possible err_i^D, as they can only have a worse error model. In other words, one only has to check *maximal combinations* for τ_i as we define them now.

Definition 8. *For a task τ_i, a* maximal combination *$\bar{c}$ is a feasible combination such that there is no feasible combination $\bar{c}'$ with $\bar{c} < \bar{c}'$. We denote $maxComb_i$ the set of maximal combinations for τ_i.*

Figure 3 illustrates a possible set of maximal combinations for the lattice of combinations of a system with four tasks, along with the corresponding set of feasible and infeasible combinations.

Once the set of maximal combinations for τ_i is known, we can define err_i^D as follows.

Theorem 7. *err_i^D as defined below is an error model for D_i.*

$$\forall k : err_i^D(k) = \min\{err_i^{\bar{c}}(k) \mid \bar{c} \in maxComb_i\} \qquad (7)$$

Proof. Let $k \in \mathbb{N}^+$. We must prove that $err_{D_i}(k)$ bounds the number of executions of τ_i which may have a response time larger than D_i in a window of k consecutive executions of τ_i. By definition of err_i^D, there exists a combination $\bar{c} \in maxComb_i$ such that $err_i^D(k) = err_{D_i}^{\bar{c}}(k)$. As $\bar{c}$ is in $maxComb_i$, it is a feasible combination and therefore: out of k consecutive executions of τ_i, at most $err_i^{\bar{c}}(k)$ may have a response time larger than $RT_i^{\bar{c}} \leq D_i$. Hence the result. □

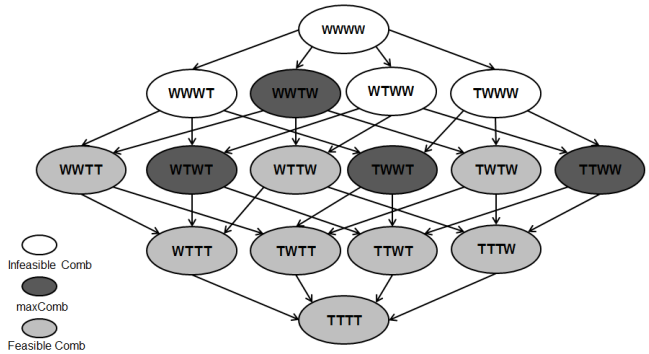

Figure 3: Lattice of combinations for a system with four tasks

6.3 Methodology for TWCA

Let us present now the overall proposed methodology for TWCA.

1. We first compute the worst-case response time as usual. If none of the tasks may miss a deadline, i.e., if $WCRT_i \leq D_i$ for all i, then TWCA is not necessary as the system is proven to be schedulable, even in the worst case.

2. For all tasks which may miss a deadline according to the worst-case model, we then compute the typical worst-case response time as in [12]. We can only provide weakly-hard guarantees for tasks which never miss a deadline according to the typical model.

3. Let us now consider a task τ_i that is schedulable in the typical worst case, but not in the worst case. We compute all maximal combinations such that no deadline can be missed. Note that in fact only 2^{n_i} combinations must be checked, where n_i is the number of tasks with higher priority than τ_i, and possibly τ_i itself, for which a typical worst-case model different from the worst case has been defined. This excludes in particular periodic tasks.

4. Based on the error models corresponding to the computed maximal combinations, we derive the error model err_i^D describing the frequency of deadline misses for task τ_i as in Equation 7.

Computing the set of maximal combinations is essential for this last step, which can be performed on a restricted number of combinations. Unfortunately, even $maxComb_i$ may be prohibitively large. This approach is therefore not an option for most systems.

We thus need a method which can compute $err_i^D(k)$ as defined above without having to compute $maxComb_i$. That is, we focus on the following problem: for a given $i \in N^+$ and $k \in N^+$, what is the best weakly-hard guarantee that we can provide about a sequence of k consecutive activations of task τ_i? To answer this question, we need to find a combination $\bar{c}$ which:

- is such that τ_i meets all its deadlines in the typical case
- and minimizes $err_i^D(k)$.

We solve this problem by formalizing it as an ILP problem.

7. AN ILP SOLUTION FOR TWCA WITH COMBINATIONS

An ILP problem consists of an objective function and a set of constraints, all of which are represented using linear combinations of integers. The vector of variables to be determined is in our case a vector of Booleans $(b_1, \ldots, b_{i-1})$ such that b_j holds exactly when τ_j is in the typical case.

Let us focus first on the objective function. We want to minimize $err_i^{\bar{c}}(k)$ and we have to represent it as a linear combination of b_j. If one expands the definition of $err_i^{\bar{c}}(k)$ as in Equations 3 and 5, we get:

$$err_i^{\bar{c}}(k) = \sum_{j \in hpe(i)} err_i^{j,\bar{c}}(k) = \sum_{j \in hpe(i)} (N_i \times \eta_{over}^{j,\bar{c}}(\Delta T_k^{j \to i}))$$

with $\Delta T_k^{j \to i}$ and N_i defined as in Equations 4 and 6.

The key point here is that for any ΔT, we have

$$\eta_{over}^{j,\bar{c}}(\Delta T) = b_j \times \eta_{over}^{j}(\Delta T)$$

Hence the following objective function.

Definition 9. *If we denote $\Omega_j^k = \eta_{over}^j(\Delta T_k^{j \to i})$, we define our objective function as*[3]*:*

$$\textbf{min} \sum_{j \in hpe(i)} b_j \times \Omega_j^k \tag{8}$$

We now focus on the constraint to be preserved, namely absence of deadline miss of τ_i in the typical case. 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 [11]). As a result, this is not a restriction in practice.

For simplicity, we also assume that τ_i does not itself experience sporadic overload. We will discuss at the end of this section how to lift this assumption. Let us now explain how we formalize our constraint in the specific case where there is only one activation of τ_i in the worst-case busy window.

Case 1: only one activation of τ_i in the maximum busy window

Define:

$$\Lambda_i = WCRT_i - D_i \tag{9}$$

Λ_i is the amount by which we need to reduce the size of the busy window to avoid deadline misses (see Figure 4). 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, let us define for the SPP case:

$$\Gamma_i = \sum_{j \in hp(i)} [\eta_j^+(WCRT_i) - \eta_j^+(D_i)] \times WCET_j \tag{10}$$

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

[3]Note that N_i has disappeared here, because it is a global constant that can be factored out in the optimization problem.

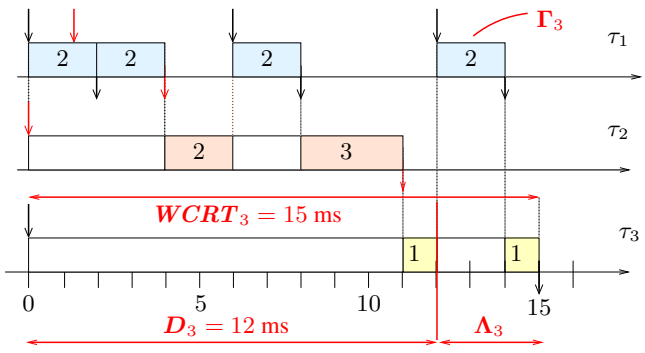

Figure 4: Λ_3 and Γ_3 in the ILP problem for task τ_3

Proof. Activations in the maximum busy window which occur after the deadline D_i but before $WCRT_i$ contribute fully to the worst-case response time of τ_i but not to any typical worst-case response time that meets the deadline. □

The workload induced by the overload of a task τ_j and contributing to the worst-case response time of τ_i but which is not part of Γ is $\eta_{over}^j(D_i) \times WCET_j$.

Definition 10. *We can therefore define our constraint as:*

$$\textbf{such that} \sum_{j \in hp(i)} wl_{over}^j \times b_j \geq \Lambda_i - \Gamma_i \tag{11}$$

where wl_{over}^j denotes $\eta_{over}^j(D_i) \times WCET_j$.

In the SPNP case, one has to exclude from Γ the activations which occur after τ_i has started executing as they are blocked by τ_i and therefore do not contribute to $WCRT_i$. However, activations which occur between $D_i - WCET_i$ and D_i cannot influence a typical worst-case response time meeting the deadline. Hence the following alternative definition for SPNP.

$$\Gamma_i = \sum_{j \in hp(i)} [\eta_j^+(QD_i) - \eta_j^+(D_i - WCET_i)] \times WCET_j \tag{12}$$

Similarly wl_{over}^j must be defined as $\eta_{over}^j(D_i - WCET_i) \times WCET_j$ in the SPNP case.

Case 2: several activations of τ_i in the maximum busy window

We can now generalize this to the case where there are K_i activations of τ_i in the worst-case busy window (out of which at most N_i may miss their deadline). In that context, each activation in the busy window which misses its deadline generates one constraint.

Consider such an activation, say the l-th in the worst-case busy window, for $1 \leq l \leq K_i$. The response time of activation l is equal to $B_i(l) - \delta_i^-(l)$ so that the amount Λ_i^l by which we must reduce it to avoid a deadline miss of activation l is $B_i(l) - \delta_i^-(l) - D_i$.

Γ_i^l is now defined in the general case for SPP as

$$\sum_{j \in hp(i)} [\eta_j^+(B_i(l)) - \eta_j^+(D_i + \delta_i^-(l))] \times WCET_j$$

and

$$wl_{over}^{j,l} = \eta_{over}^j(D_i + \delta_i^-(l)) \times WCET_j$$

Note that to determine how many activations occur between the

deadline of activation l and its response-time bound, we need to consider the worst-case activation patterns η^+ starting from the beginning of the busy window, as this is how activation times are determined in the maximum busy window.

The constraint associated to l is then the same as before but based on Λ_i^l, Γ_i^l and $wl_{over}^{j,l}$. The ILP constraint is the conjunction of such constraints, one for each activation missing its deadline in the worst-case busy window.

Similarly, for the general SPNP case we define:

$$\Gamma_i^l = \sum_{j \in hp(i)} [\eta_j^+(w_i(l)) - \eta_j^+(D_i + \delta_i^-(l) - WCET_i)] \times WCET_j$$

and

$$wl_{over}^{j,l} = \eta_{over}^j(D_i + \delta_i^-(l) - WCET_i) \times WCET_j$$

If we now consider that τ_i itself may suffer from sporadic overload, observe that τ_i does not contribute to Γ_i^l because activations of τ_i which occur after the l-th activation do not contribute to its response time (since activations are treated in a FIFO order). Besides, τ_i contributes to the l-th response time only before the l-th activation, that is:

$$wl_{over}^{i,l} = \eta_{over}^i(\delta_i^-(l)) \times WCET_i$$

Finally, let us underline here that the condition we are coming up with is a sufficient condition but it is not necessary. Indeed, it may happen that by removing less than Λ_i workload one still reduces sufficiently the response time, because some activations then fall out of the busy window, as is the case for those dealt with in Γ_i. This means that our optimization problem may not come up with a maximum combination. We study this in our experiments.

8. EXPERIMENTS

In this section, we report on various experiments that we have performed to evaluate the efficiency of our approach for TWCA with multiple combinations in different set-ups. We use an extension of the pyCPA tool [6] to perform our worst-case busy window analysis, and GNU octave to solve the ILP problem[4].

8.1 Comparison with the state of the art

First, we have built an example in order to compare the results presented in this paper to those of [12]. This example is a system with an SPP scheduling policy and the characteristics presented in Table 1, chosen to represent a realistic setting.

We consider 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). We use implicit deadlines, i.e. we define the deadline of each task as being equal to its period. Note here that TWCA is not applicable to sporadic tasks: our error model computation is based on the task maximum distance function δ^+, which is not defined for sporadic tasks. As a result, we do not assign deadlines to sporadic tasks and do not compute their typical worst-case response time.

The usual worst-case response time analysis of tasks in Table 1 shows that τ_{15} is the only task that may miss its deadline, since

[4]We use the `glpk` command.

task ID	period	$WCET$	$WCRT$	$TWCRT$
τ_1	20	2	2	2
τ_2	20	5	7	7
τ_3	sporadic	2	9	–
τ_4	40	4	13	11
τ_5	sporadic	6	19	–
τ_6	40	3	29	14
τ_7	sporadic	3	32	–
τ_8	40 (mixed)	1	34	15
τ_9	40 (mixed)	1	38	17
τ_{10}	sporadic	2.5	57	–
τ_{11}	sporadic	1.5	59	–
τ_{12}	100 (mixed)	4	74	28
τ_{13}	100 (mixed)	3	80	31
τ_{14}	150 (mixed)	2	115	33
τ_{15}	100	10	149	60

Table 1: System parameters and task response times

$WCRT_{15} = 149$ ms, which exceeds the deadline $D_{15} = 100$ ms. How often this may happen is not given by worst-case analysis. TWCA as in [12] returns as typical bound $TWCRT_{15} = 60$ ms, that is, largely below D_{15}. Table 2 shows the error model that is then computed for several values of k. Note that all values obtained by [12] could be divided by 2 simply by using the improvement proposed in Section 5.

Table 2 also shows the error model that is computed according to the so-called naive approach of Section 6 which computes all maximum combinations, as well as with the ILP problem of Section 7. The results are significantly better than for [12] due to the fact that our approach comes up with a typical-case response time bound that is much closer to the deadline. Note that the (possibly not optimal) ILP solution here is the same as the naive solution.

The running time of the ILP problem was 2.7 seconds, which is significantly shorter than the time required by our naive approach that took 182.9 seconds. We have performed this experiment on an Intel Core 2 Duo CPU P9700 @ 2.80GHz × 2 with 4GB memory.

Let us detail here the results obtained using the ILP approach of Section 7, for $k = 50$. There is only activation of τ_{15} in the maximum busy window that misses its deadline, so only one constraint. Remember that we only consider in our ILP problem tasks τ_j which have some sporadic overload, i.e. sporadic or mixed tasks.

k	50	100	150	200	250
$err_{15}(k)$ ([12])	80	98	112	118	124
$err_{15}^D(k)$ (naive)	11	12	15	16	18
$err_{15}^D(k)$ (ILP)	11	12	15	16	18

Table 2: Error models obtained using [12] and this paper

We have:

- $\Delta T_{50}^{j \to 15} = 5227$ ms for all $j \neq i$
- $\Omega^k = [4, 4, 4, 4, 3, 4, 4, 5, 4, 4]$
- $wl_{over} = [2, 6, 3, 1, 2, 2.5, 1.5, 4, 3, 2]$
- $\Lambda_{15} = 49$
- $\Gamma_{15} = 38$

The ILP solver returns $\bar{c} = (\text{WTWWTWWWTW})$ so that we have:

$$err_{15}^{D}(50) = err_{15}^{\bar{c}}(50) = 11$$

Note that $\sum_{j \in hp(15)} wl_{over}^{j} = 25 \leq \Lambda_{15}$. This means that our ILP approach would not find a solution if we were not using Γ_{15} to reduce the workload to be removed.

8.2 Influence of the deadlines on the results

We have repeated our experiment while choosing various deadlines for τ_{15} that are smaller than the period $P_{15} = 100$ ms, namely for $D_{15} \in \{90, 80, 70\}$ ms. Our objective is to show how the deadline influences the error models obtained with ILP. Figure 5 shows the obtained results for different values of k.

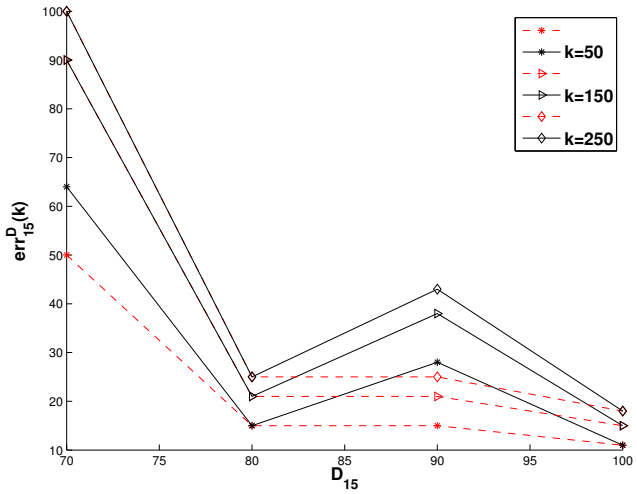

Figure 5: Relation between D_{15} and $err_{15}^{D}(k)$

Note that we experience here some pessimism for $D_{15} = 90$ ms due to the fact that our ILP problem does not compute the best possible error model for D_{15} (as could be obtained with the naive approach). Indeed, reducing the deadline cannot decrease the error: a shorter deadline is in fact likely to entail more deadline misses. In addition, for $D_{15} = 70$ ms we have $err_{15}^{D}(50) = 64$ although it is clear from the definition that $err_{15}^{D}(k) \leq k$. So, it would be straightforward to derive a more realistic monotonous error function for each deadline, as shown in dashed red lines in Figure 5.

The origin of the pessimism observed for $D_{15} = 90$ ms can be explained based on Table 3 which shows the values of Λ_{15} and Γ_{15} for the various deadlines: between $D_{15} = 90$ ms and $D_{15} = 80$ ms, Λ_{15} increases by 10 but Γ_{15} increases by 17, because some activations now occur after the deadline.

Note that for $D_{15} = 70$ ms we have two values for Λ_{15} and Γ_{15}, because there are two activations of τ_{15} in the longest busy window that have a response time larger than the deadline — which means there are two constraints in the ILP problem.

D_{15}	100	90	80	70
Λ_{15}	[49]	[59]	[69]	[79,8]
Γ_{15}	[38]	[38]	[55]	[55,0]

Table 3: Λ_{15} and Γ_{15} for different values of D_{15}

Further experiments (see Table 4) show that the ILP approach is reasonably pessimistic compared to the naive method — and still performs much better than [12]: remember that the naive approach is not applicable to systems with more than 10 tasks having a typical worst-case model. Note also that using the improvement of Section 5 on the error for [12] would yield an error of 49 instead of 98 for $D_{15} \geq 80$.

D_{15}	100	95	90	85	80	75	70
$err_{15}(100)$ ([12])	98	98	98	98	98	98	98
$err_{15}^{D}(100)$ (naive)	12	17	17	17	17	54	76
$err_{15}^{D}(100)$ (ILP)	12	19	32	49	17	54	80

Table 4: err_{15} obtained for different deadlines

Finally, Figure 6 illustrates the relation between k and $err_{15}^{D}(k)$ for different values of D_{15}. As one can see, the error slowly increases with k. Note that the curve for $D_{15} = 90$ ms is above that for $D_{15} = 80$ ms due to the optimality problem mentioned above.

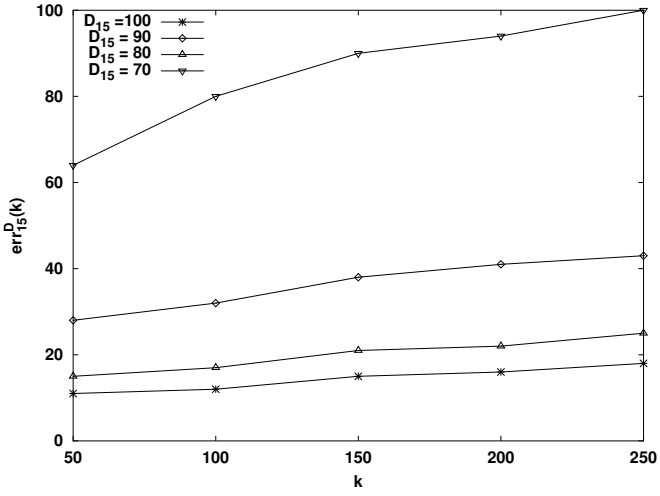

Figure 6: Relation between k and $err_{15}^{D}(k)$

8.3 Test with the SPNP scheduling policy

We now repeat our first experiment after changing our scheduling policy from SPP to SPNP with the basic setup (i.e. $D_{15} = 100$ ms) and then for $D_{15} = 90$ ms

The usual worst-case response time analysis computes a bound of $WCRT_{15} = 125$ ms, so again task τ_{15} may in the worst case miss a deadline $D_{15} = 100$ ms. TWCA as in [12] returns as typical bound $TWCRT_{15} = 43$ ms. Tables 5 and 6 show the error model obtained using ILP and the naive approach for D_{15} equal to respectively 100 ms and 90 ms.

k	50	100	150	200	250
$err_{15}^{D}(k)$ (naive)	4	4	6	6	6
$err_{15}^{D}(k)$ (ILP)	11	12	15	16	18

Table 5: Error models obtained for $D_{15} = 100$ under SPNP

k	50	100	150	200	250
$err_{15}^{D}(k)$ (naive)	4	4	6	6	6
$err_{15}^{D}(k)$ (ILP)	4	4	6	6	6

Table 6: Error models obtained for $D_{15} = 90$ under SPNP

Again, we experience a non-optimal result when $D_{15} = 100$ ms. It is explained by the values of Λ_i and Γ_i computed for the ILP solution, which are shown in Table 7.

Note that, unsurprinsingly, the error is smaller for SPNP than for SPP, because the worst-case response time is smaller.

D_{15}	100	90	80	70
Λ_{15}	[25]	[35]	[45]	[55]
Γ_{15}	[14]	[14]	[31]	[38]

Table 7: Λ_{15} and Γ_{15} for different D_{15} under SPNP

9. CONCLUSION

In this paper, we have shown how to extend Typical Worst-Case Analysis (TWCA) [12] to determine the maximum number of deadline misses in a sequence of executions of a given task. The solution uses an optimization algorithm which resolves dependencies between response time bounds obtained from busy-window analyses based on different definitions of typical worst case. Our approach addresses systems with SPNP and SPP scheduling but it could be used for other scheduling algorithms if these can be analyzed using a busy window approach.

The ILP solution introduced here, which is based on an underlying lattice structure, has proven suitable to respond to different deadline requirements and combinations of worst case and typical worst-case. Where it can be compared, it achieves much better results than the current state of TWCA. This new flexibility and efficiency is an important step towards using TWCA for the practical design of networks and embedded systems.

10. REFERENCES

[1] Philip Axer, Maurice Sebastian, and Rolf Ernst. Probabilistic response time bound for CAN messages with arbitrary deadlines. In *Proceedings of DATE'12*. IEEE Computer Society, 2012.

[2] Guillem Bernat, Alan Burns, and Albert Llamosí. Weakly hard real-time systems. *IEEE Trans. Computers*, 50(4):308–321, 2001.

[3] Francisco J. Cazorla, Eduardo Quiñones, Tullio Vardanega, Liliana Cucu, Benoit Triquet, Guillem Bernat, Emery D. Berger, Jaume Abella, Franck Wartel, Michael Houston, Luca Santinelli, Leonidas Kosmidis, Code Lo, and Dorin Maxim. PROARTIS: Probabilistically analyzable real-time systems. *ACM Trans. Embedded Comput. Syst.*, 12(2s):94, 2013.

[4] Samarjit Chakraborty, Simon Künzli, and Lothar Thiele. A general framework for analysing system properties in platform-based embedded system designs. In *Proceedings of DATE'03*, pages 190–195. IEEE Computer Society, 2003.

[5] Robert I. Davis, Alan Burns, Reinder J. Bril, and Johan J. Lukkien. Controller area network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):239–272, 2007.

[6] Jonas Diemer, Philip Axer, and Rolf Ernst. Compositional performance analysis in python with pyCPA. In *Proceedings of WATERS'12*, 2012.

[7] Rafik Henia, Arne Hamann, Marek Jersak, Razvan Racu, Kai Richter, and Rolf Ernst. System level performance analysis — the SymTA/S approach. In *IEE Proceedings Computers and Digital Techniques*, 2005.

[8] Pratyush Kumar and Lothar Thiele. Quantifying the effect of rare timing events with settling-time and overshoot. In *Proceedings of RTSS'12*, pages 149–160, 2012.

[9] John P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *IEEE Real-Time Systems Symposium*, pages 201–213, 1990.

[10] José María López, José Luis Díaz, Joaquín Entrialgo, and Daniel F. García. Stochastic analysis of real-time systems under preemptive priority-driven scheduling. *Real-Time Systems*, 40(2):180–207, 2008.

[11] Saad Mubeen, Jukka Mäki-Turja, and Mikael Sjödin. Extending schedulability analysis of controller area network (CAN) for mixed (periodic/sporadic) messages. In *Proceedings of ETFA'11*, pages 1–10, 2011.

[12] Sophie Quinton, Matthias Hanke, and Rolf Ernst. Formal analysis of sporadic overload in real-time systems. In *Proceedings of DATE'12*, pages 515–520. IEEE Computer Society, 2012.

[13] Sophie Quinton, Mircea Negrean, and Rolf Ernst. Formal analysis of sporadic bursts in real-time systems. In *Proceedings of DATE'13*, pages 767–772. IEEE Computer Society, 2013.

[14] Ken Tindell, Alan Burns, and Andy J. Wellings. An extendible approach for analyzing fixed priority hard real-time tasks. *Real-Time Systems*, 6(2):133–151, 1994.