A Generic Coq Proof of Typical Worst-Case Analysis

Pascal Fradet, Maxime Lesourd, Jean-François Monin, Sophie Quinton

A Generic Coq Proof of Typical Worst-Case Analysis

Pascal Fradet[1], Maxime Lesourd[1,2], Jean-François Monin[2], Sophie Quinton[1]
[1]*Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, F-38000 Grenoble France*
[2]*Univ. Grenoble Alpes, CNRS, Grenoble INP, VERIMAG, F-38000 Grenoble France*

***Abstract*—This paper presents a generic proof of Typical Worst-Case Analysis (TWCA), an analysis technique for weakly-hard real-time uniprocessor systems. TWCA was originally introduced for systems with fixed priority preemptive (FPP) schedulers and has since been extended to fixed-priority nonpreemptive (FPNP) and earliest-deadline-first (EDF) schedulers. Our generic analysis is based on an abstract model that characterizes the exact properties needed to make TWCA applicable to any system model. Our results are formalized and checked using the Coq proof assistant along with the Prosa schedulability analysis library. Our experience with formalizing real-time systems analyses shows that this is not only a way to increase confidence in our claimed results: The discipline required to obtain machine checked proofs helps understanding the exact assumptions required by a given analysis, its key intermediate steps and how this analysis can be generalized.**

***Index Terms*—formal proofs, weakly-hard real-time systems, Coq, deadline miss models.**

I. Introduction

Context and motivation

The analysis of weakly-hard real-time systems is currently the subject of renewed attention in the research community [5], [6], [30] as well as in industry [15], [18], [26], [33].

Originally called (m, k)-firm real-time systems [17], weakly-hard real-time systems [8] are systems that have strict functional requirements but are designed in such a way that they can tolerate a few deadlines misses as long as their number can be bounded (no more than m out of k deadlines missed). Typical instances of weakly-hard real-time systems are found in control applications: The functional closed-loop properties of a control system are usually resilient to a few deadline misses [15], [23]. The industrial relevance of weakly-hard guarantees is higher today than ever in the automotive domain with the advent of electrical powertrains. Such systems require advanced control approaches and have higher dynamics, making the number and the impact of deadline misses less predictable than before [33].

There are surprisingly few results on the analysis of weakly-hard real-time systems and most of them only apply to a restricted class of systems running periodic, independent tasks [8], [30]. Typical Worst-Case Analysis (TWCA) is one exception which can handle tasks with complex activation models [27], task dependencies [18], finite ready queues [6] and even recently multiprocessor architectures [5]. Still, the approach needs improving. It must be generalized to other scheduling policies and more complex system models, and pessimism in the computed bounds must be reduced. Also, most of the above mentioned extensions have been proven correct in isolation and combining them is far from trivial.

In parallel, the recent finding of serious bugs in a series of papers deemed important [25] has made it clear that there is a need for more formal proofs in real-time systems research. Building a theory as complex as that underlying TWCA, we want to guarantee the absence of such mistakes and make it easier to build on top of the existing results. Computer-assisted proofs have many advantages:

- Since they are mechanically checked, one does not have to read them in order to trust a result. One can thus focus on the specification of the result (which can still be erroneous) and the key steps required to prove it.
- Formal proofs require making all hypotheses explicit, which helps understanding precisely the role of each of them. This makes it easier to generalize results as we discuss in this paper.
- Finally, a proof environment such as the Coq proof assistant [1] encourages the reuse of proof components, and thus the elaboration of libraries of proofs sharing lemmas. This is very appealing to the researcher who wants to build a complex theory in which many results share similar features.

For all these reasons, we have decided to provide a formal framework for TWCA.

Contribution

In this paper, we propose the first formal proof of an analysis for computing weakly-hard guarantees. The analysis that we prove is a generalization of Typical Worst-Case Analysis [32] (TWCA) that we propose to make it easier to apply TWCA to new system models. Our generic proof is based on an abstract model that characterizes the exact properties needed to make TWCA directly applicable to any system model Intuitively, what we need is a way to partition an execution trace into time intervals over which a local deadline miss analysis can be performed in isolation, based on an abstraction of the trace in each interval. In our paper, we formalize this intuition and show how to apply our generic analysis to common scheduling policies, including fixed priority preemptive (FPP), fixed-priority nonpreemptive (FPNP) and earliest-deadline-first

This work has been partially supported by the LabEx PERSYVAL-Lab (grant ANR-11-LABX-0025-01) through the CASERM project and the French national research organization ANR (grants ANR-15-CE25-0008 and ANR-17-CE25-0016) through the VOCAL and RT-PROOFS projects.

(EDF). The chosen trace abstraction is different from that in the state-of-the-art TWCA, so the derived analyses that we present are new. In addition, we show in the appendix how to derive a result that is closer to the original analysis [32].

Our results are formalized and checked using the Coq proof assistant [1] based on the Prosa schedulability analysis library [10]. The complete specification and the proofs can be found online [4].

The main contributions of this paper are:

- a generic analysis inspired by TWCA which can be instantiated to a TWCA for any system model with a few well-defined properties;
- an instantiation of this analysis to FPP, FPNP and EDF, as well as a derivation of a result close to [32];
- a formal proof in Coq of the analysis and its instantiation to FPP.

Furthermore, our work opens up new research directions for TWCA by providing a formal framework for the trade-off that must be found between time efficiency and precision of the analysis. By providing a generic proof of TWCA, our result will make it easier to extend TWCA to more complex models in the future.

In addition, our experience with formalizing real-time systems analyses shows that this is not only a way to increase confidence in our claimed results: The discipline required to obtain machine checked proofs helps understanding the exact assumptions required by a given analysis, its key intermediate steps and how this analysis can be generalized.

Structure of the paper

This paper is organized as follows. Section II provides an overview of Typical Worst-Case Analysis as developed in [32]. Section III then revisits the main proof of that paper by proposing a more formal, generic version of it that is instantiated to several scheduling policies in Section IV. Section V discusses our experience with developing the analysis described here using a proof assistant. Finally, Section VII discusses the questions raised by our work and indicates research directions. Additionally, we provide in the appendix a proof of the TWCA from [32] based on our analysis.

II. State of the art TWCA

In this section, we provide an overview of the state of the art regarding Typical Worst-Case Analysis (TWCA), from which we drew inspiration to build our formal proof. Specifically, we present the main results of [32], with small changes to notations and definitions (that do not impact their semantics unless we explicitly mention it) in order to reflect more directly their implementation in Prosa. The definitions provided in this section are generalized in Section III and the appendix shows how the results presented here can be derived by instantiating our generic analysis.

Note that we suppose throughout this paper a representation of time based on natural numbers. This is reasonable so far since we consider for the moment single processor systems, which operate according to a unique, discrete clock. Unless otherwise stated, all parameters have positive integer values.

A. System model and runtime behavior

The system model under consideration in [32] is a single processing resource scheduling a task set $TS := \{\tau_1, \tau_2, \ldots, \tau_n\}$ according to a Fixed Priority Preemptive (FPP) policy.

Definition 1 (Task)**.** A task $\tau \in TS$ is defined by its

- worst-case execution time W_τ,
- *activation pattern* $(\eta^+_\tau, \eta^-_\tau)$ — see below,
- *priority* π_τ,
- arbitrary, relative *deadline* D_τ.

Tasks are assumed to have distinct priorities and $hp(\tau)$, resp. $hpe(\tau)$, denotes the set of tasks with higher, resp. equal or higher, priority than τ.

Activation patterns are modeled using *arrival curves* [21], which can capture the activation of sporadic tasks less pessimistically than models based on a minimum interarrival time.

Definition 2 (Activation pattern)**.** The *activation pattern* of a task $\tau \in TS$ is defined by two functions $\eta^+_\tau : \mathbb{N} \to \mathbb{N}$, and $\eta^-_\tau : \mathbb{N} \to \mathbb{N}$, that upper-bound and lower bound, the number of activations of τ in any time interval: $\eta^+_\tau(\Delta)$ upper bounds and $\eta^-_\tau(\Delta)$ lower bounds the number of activations of τ that might occur within any time interval of length Δ.

Pseudo-inverse functions of η^+_τ and η^-_τ, denoted respectively $\delta^-_\tau : \mathbb{N} \to \mathbb{N}$ and $\delta^+_\tau : \mathbb{N} \to \mathbb{N}^\infty$ are also useful: for any sequence of k consecutive activations of τ, $\delta^-_\tau(k)$ lower bounds and $\delta^+_\tau(k)$ upper bounds[1] the time that might pass between the first and the last activation in the sequence.

At runtime, tasks are activated according to their activation pattern. Whenever a task is activated, a corresponding task instance, called *job*, is created. A job j of task τ is characterized by its activation time act_j, deadline $d_j := act_j + D_\tau$ and execution time $cost_j \leq W_\tau$. In order to complete, a job must receive $cost_j$ time units of *service* from the scheduler. A job is said to be *pending* at an instant t if it has been activated and not yet completed at t. According to the FPP scheduling policy, at each instant t, the scheduler *schedules* (*i.e.,* provides service to) the job pending at t that has the maximum priority among jobs pending at t. Pending jobs of the same task are scheduled in FIFO order. A job misses its deadline if its *response time* (that is, the duration between its activation and its completion time) is larger than its deadline. Jobs that miss their deadline still run to completion.

Finally, the system model assumed by TWCA distinguishes between so-called typical and overload activations. This is where we choose to deviate from that model and consider instead *typical* and *overload* tasks, in line with more recent papers [18], [20]. The rationale behind the distinction between typical and overload tasks is that TWCA was introduced as a technique to analyze systems that may be temporarily

[1]Note that, for sporadic tasks, $\forall k \geq 2, \delta^+_\tau(k) = \infty$.

overloaded due to the activation of some rarely activated sporadic tasks. In such systems, sporadic overload can lead to deadline misses, but after some time the system goes back to a typical state in which no deadlines are missed. TWCA computes how often overload situations may occur and how many deadline misses they may induce. To do so, TWCA heavily relies on classical worst-case response-time analysis, which we recall now.

B. Worst-case response-time analysis

A system is *schedulable* if no job of any task can miss its deadline. A standard approach to establish schedulability of systems scheduled with FPP is to compute, for each task, an upper bound on the response time of its jobs for any runtime behavior compatible with the system model. Such an analysis, often called worst-case response-time analysis, is based on the concept of busy period, or *busy window* [31][2].

Definition 3 (Level-τ busy window)**.** At runtime, an instant t is said to be *quiet* for τ if all jobs of tasks in $hpe(\tau)$ activated strictly before t have completed by t. Otherwise it is said to be *busy* for τ. A *level-τ busy window* is an interval $[t_1, t_2[$ such that $t_1 < t_2$, t_1 and t_2 are quiet times for τ and any $t \in]t_1, t_2[$ is a busy time for τ.

Level-τ busy windows have the nice property that the response time of jobs of τ in a given level-τ busy window only depends on jobs activated in the same busy window. The analysis of a task τ can then focus on the worst-case behavior *w.r.t.* τ within a single busy window, following the steps that we now recall.

- For $q \geq 1$, the time it may take for the q-th job of τ to complete within any level-τ busy window that contains at least q activations of τ is upper bounded by

$$B_\tau(q) := \min\{\Delta \geq 0 \mid \Delta = F(\Delta)\} \tag{1}$$

with $$F(\Delta) := q \times W_\tau + \sum_{\tau' \in hp(\tau)} \eta^+_{\tau'}(\Delta) \times W_{\tau'}$$

- For $q \geq 1$, the response time of the q-th job of τ within any level-τ busy window that contains at least q activations of τ is upper bounded by

$$RT_\tau(q) := B_\tau(q) - \delta^-_\tau(q) \tag{2}$$

- The number of activations of τ in any level-τ busy window is upper bounded by

$$K_\tau := \min\{q \geq 1 \mid B_\tau(q) < \delta^-_\tau(q+1)\} \tag{3}$$

K_τ is such that the resource would be able to start processing the $(K_\tau + 1)$-th activation of τ before this activation can occur according to δ^-_τ, which implies a quiet time for τ.

- The length of any level-τ busy window is bounded by

$$BW_\tau := B_\tau(K_\tau) \tag{4}$$

- The response time of τ is bounded by

$$R_\tau := \max_{1 \leq q \leq K_\tau} \{RT_\tau(q)\} \tag{5}$$

We refer the reader to [31] for detailed explanations about the FPP worst-case response-time analysis.

C. Typical Worst-Case Analysis

The objective of TWCA is to compute a *deadline miss model* (DMM) for each typical task τ.

Definition 4. A *deadline miss model* (DMM) for task τ is a function $dmm_\tau : \mathbb{N} \to \mathbb{N}$, with the property that out of any sequence of k consecutive jobs (called k-sequence) of τ, at most $dmm_\tau(k)$ might miss their deadline D_τ.

A trivial DMM is the identity function. The objective of TWCA is to provide another, hopefully better DMM by studying the impact of activations of overload tasks on jobs in a k-sequence. TWCA assumes that τ is guaranteed not to miss any deadline in absence of any overload activation. In other words, the system without the overload tasks must be schedulable. The reasoning behind TWCA is as follows.

- The number of deadline misses for τ in any level-τ busy window is bounded by

$$N_\tau := |\{1 \leq q \leq K_\tau \mid RT_\tau(q) > D_\tau\}| \tag{6}$$

where $|E|$ denotes the cardinality of a set E. Note that one activation of an overload task cannot result in more than N_τ deadline misses of τ as it can only impact activations of τ which are in the same level-τ busy window.

- For a given k-sequence $\mathcal{J}_k := < j_1, \ldots, j_k >$, activations of overload tasks that occur before the level-τ busy window in which j_1 is activated, or after the completion of j_k, cannot impact the response time of any job in $\mathcal{J}_k$. As a result, for any k-sequence $\mathcal{J}_k$ of τ, the length of the time interval during which the activation of an overload task can impact the response time of jobs in $\mathcal{J}_k$ is upper bounded[3] by

$$\Delta_\tau(k) := BW_\tau + \delta^+_\tau(k) + BW_\tau \tag{7}$$

that is, an upper bound on the length of an interval containing k activations flanked by two busy windows. This bound is slightly more pessimistic than in the state of the art where the second BW_τ is replaced by R_τ. For readability, we choose to use this bound as it is easier to generalize in Section III and removing this source of pessimism would require additional hypotheses in the generic analysis.

- Let $\mathcal{O}_\tau$ denote the set of overload tasks in $hp(\tau)$. The number of activations of a task $\tau' \in \mathcal{O}_\tau$ in a time interval of size $\Delta_\tau(k)$ is upper bounded by

$$\Omega_{\tau' \to \tau}(k) := \eta^+_{\tau'}(\Delta_\tau(k)) \tag{8}$$

[2]Our definition of level-τ busy window is taken from Prosa and it is discussed in Section IV.

[3]Note that $\Delta_\tau = \infty$ for sporadic tasks and therefore TWCA cannot provide guarantees for such tasks — but it does apply, *e.g.,* to periodic tasks in systems that also execute sporadic tasks.

• At this point, it is already possible to bound the number of deadline misses in a k-sequence of τ by

$$\sum_{\tau' \in \mathcal{O}_\tau} N_\tau \times \Omega_{\tau' \to \tau}(k)$$

This bound directly results from the fact that, for each $\tau' \in \mathcal{O}_\tau$, at most $\Omega_{\tau' \to \tau}$ jobs of τ' can impact a given k-sequence, each of them causing at most N_τ deadline misses. This bound is however easily improved, as the activation of a single overload task in a level-τ busy window is usually not sufficient to cause a deadline miss. TWCA as presented in [32] therefore improves on [27] by introducing a concept of *(un)schedulable combinations*.

Definition 5. A *schedulable combination* $\overline{C}$ with respect to τ is a subset of the overload tasks such that τ is schedulable when only typical tasks and tasks in $\overline{C}$ are activated. Otherwise, $\overline{C}$ is said to be an *unschedulable combination w.r.t.* τ. The set of unschedulable combinations *w.r.t.* τ is denoted $\mathcal{U}_\tau$.

An efficient criterion to determine whether a combination is unschedulable is given in [32].

• A key observation is that deadline misses of τ can only happen in level-τ busy windows in which there is at least one activation of each overload task in $\overline{C}$ for some unschedulable combination $\overline{C} \in \mathcal{U}_\tau$. Since the number of activations of an overload task τ' that impact a k-sequence can be upper bounded by $\Omega_{\tau' \to \tau}(k)$, one can look for the assignment of these activations to level-τ busy windows which results in the maximum number of unschedulable combinations. Based on this observation, the main result of [32] is the proof that the following function is a DMM, as well as an efficient Integer Linear Programming (ILP) solution to compute it.

$$dmm_\tau(k) := \\ N_\tau \times \max_X \left\{ \sum_{\overline{C} \in \mathcal{U}_\tau} X_{\overline{C}} \mid \forall \tau' \in \mathcal{O}_\tau, \sum_{\overline{C} \in \mathcal{U}_\tau^{\tau'}} X_{\overline{C}} \le \Omega_{\tau' \to \tau}(k) \right\} \tag{9}$$

where

– X denotes a so-called *abstract scenario* that assigns overload activations to busy windows; $X_{\overline{C}}$ denotes the number of times combination $\overline{C}$ appears in scenario X.

– $\mathcal{U}_\tau^{\tau'}$ denotes the set of combinations in $\mathcal{U}_\tau$ that contain τ'. Only scenarios compatible with the upper bound $\Omega_{\tau' \to \tau}(k)$ for each task $\tau' \in \mathcal{O}_\tau$ are considered.

Note that, all unschedulable combinations are assumed to potentially result in N_τ deadline misses. Additionally, even if a combination might need several activations of its tasks to be classified as unschedulable, Equation 9 counts as if each task of the combination were activated only once. These two safe approximations upper bound deadline misses.

Example 1. Let us now illustrate the DMM computation on a concrete example. We consider a system with 4 tasks: τ_1, τ_2, τ_3 are overload tasks while τ_4 is a typical task. In order to compute $dmm_{\tau_4}(5)$ we assume that $\mathcal{U}_{\tau_4}$ contains any combination where at least two overload tasks are present and that $\Omega_{\tau_1 \to \tau_4}(5) = \Omega_{\tau_1 \to \tau_4}(5) = \Omega_{\tau_1 \to \tau_4}(5) = 2$.

τ_1	✓	✓	
τ_2	✓		✓
τ_3		✓	✓

τ_1	✓	✓
τ_2	✓	✓
τ_3	✓	✓

Fig. 1. Packing overload activations into unschedulable combinations for τ_4.

Figure 1 shows two scenarios where columns represent unschedulable combinations and thus the number of columns is the number of level-τ_4 busy windows which may miss up to N_{τ_4} deadlines. In the first scenario $X_{\{\tau_1;\tau_2\}} = X_{\{\tau_1;\tau_3\}} = X_{\{\tau_2;\tau_3\}} = 1$ leads to $3 \times N_{\tau_4}$ deadline misses. In the second example $X_{\{\tau_1;\tau_2;\tau_3\}} = 2$ leads to $2 \times N_{\tau_4}$ deadline misses. The maximization problem in the computation of $dmm_\tau(k)$ considers the set of all these scenarios under the constraint that the number of checkmarks on row τ_i is not greater than $\Omega_{\tau_i \to \tau_4}$. It is worth noting that if a busy window exhibiting the combination $\{\tau_1; \tau_2\}$ required two activations of τ_1 to be unschedulable, the first scenario would be unrealistic. A refined analysis could filter out these impossible scenarios to improve its precision.

More than the distinction between overload and typical tasks, the essence of TWCA is to take advantage of the fact that different local analyses can be performed in different level-τ busy windows. Note that, in fact, the local analyses that we perform are based on a deadline miss analysis rather than a worst-case response time analysis.

While the definitions in [32] are based on the assumption that the scheduling policy is FPP, we show in the next section that it is possible to abstract these definitions in order to analyze any system model with some well-identified properties.

III. A GENERIC PROOF OF TWCA

Our goal is to build a generic framework in order to extend TWCA to a wide range of scheduling policies and task models. To do so, let us first review the state-of-the-art analysis to identify where specific assumptions about the model are made.

First, properties of the activation pattern presented in Definition 1 are used for the worst-case response-time analysis, which forms the basis of TWCA since it is used to compute N_τ. In addition, the activation pattern of tasks is also used to compute $\Omega_{\tau' \to \tau}$, which constrains the set of possible scenarios in the DMM computation. Note that $\Delta_\tau(k)$ also depends on the activation pattern of tasks, but it is only used to compute $\Omega_{\tau' \to \tau}$. We build our analysis over an abstract activation pattern that can be instantiated to various models, *e.g.* arrival curves.

Second, and most importantly, the definition of busy window, which is a building block of the analysis, is specific to FPP. A similar concept exists for other scheduling policies, *e.g.* Fixed Priority Non-Preemptive (FPNP) [26] and Earliest Deadline First (EDF) [19], but so far a separate proof is required to apply TWCA to these policies. What we show in the following is that a more general concept of *analyzable window* with a few key properties is sufficient to apply TWCA.

A. Principle of the generic TWCA

In this section, we provide an abstract definition of analyzable window that generalizes the concept of busy window, and a specification of the local deadline miss analysis that is needed for TWCA. These two concepts can then be used as interfaces between a specific system model and the generic analysis that we propose, as we illustrate in Section IV. The definitions we give here are as close as possible to the Coq formalization of our work. Though the following theorems have been proved in Coq, we provide high-level pen-and-paper proofs of our results.

Intuitively, the concept of analyzable window captures a notion of isolation between different time intervals. Indeed, the only property of busy windows that is used in the proof of TWCA is that an activation of an overload task can only impact the response time of jobs of τ that are in the same analyzable window. In fact, this property can be weakened even further: In our generic analysis, we only require analyzable windows to be disjoint and such that the local analysis performed on each analyzable window does not depend on the activation of tasks in other analyzable windows.

We follow the intuition of the state of the art analysis and bound the number of deadline misses for a k-sequence of a task τ by packing activations of tasks over an abstract scenario. This local scenario describes the number of occurrences of combinations in the analyzable windows affecting our k-sequence. The scenarios we consider are constrained by the task model. The local deadline miss analysis bounds the number of deadline misses in an analyzable window based on its combination.

The local analysis relies on an abstraction, called *combination*, of the actual activation sequence over an analyzable window to bound deadline misses. We require that, given such a combination, the local analysis provides a bound on the actual number of deadline misses for any analyzable window exhibiting this combination. This is mentioned as a possible refinement of N_τ, in [32] which gives a global bound on the number of deadline misses in a busy windows with overload.

Using these generic requirements, we can perform TWCA by partitioning our interval of interest into analyzable windows and looking at the resulting combinations. This allows us to reason on *abstract scenarios* which associate to each interval its combination. Such scenarios satisfy some constraints induced by the task model. Using the local analysis we can bound the number of deadline misses based on the abstract scenario for any interval I containing k activations of τ. Thus we just have to maximize the number of deadline misses over the possible abstract scenarios.

B. System model and runtime behavior

Let us now define the abstract system model that we use for our generic analysis. In contrast to the model introduced in Section II and used in the state of the art TWCA, we avoid definitions which are specific to a scheduling policy or a task model in favor of definitions based on traces. This is possible because, in order to perform our generic analysis, we need very little knowledge about the system except for the properties defined in the next section, which are defined on traces.

To relate the abstract concepts introduced in this section with the specific ones presented in Section II, we use Figure 2 as a running example based on the FPP policy throughout this section.

We start with an abstract notion of task, which only specifies a deadline (needed for TWCA).

Definition 6 (Task)**.** A *task* τ is characterized by a relative deadline D_τ.

From now on we consider a finite set of tasks TS. As before, instances of a task at runtime are called *jobs*.

Definition 7 (Jobs)**.** A *job* j is characterized by its task τ_j, activation time act_j and execution time $cost_j$. The deadline d_j of a job j of task τ is $act_j + D_\tau$.

Note that jobs are simply defined as requests for service to which we assign a deadline. In particular, they are not constrained as usual by properties of their task such as worst-case execution time or activation pattern. Such constraints are implicitly encoded in the trace based definition of the runtime behavior of the system.

Definition 8 (Trace)**.** An execution trace σ of a task set is a pair $(\mathcal{A}, \mathcal{S})$ where $\mathcal{A}$ and $\mathcal{S}$ are functions called respectively *activation sequence* and *schedule*.

In a trace $\sigma = (\mathcal{A}, \mathcal{S})$, $\mathcal{A}(t)$ is the set of jobs *activated* at time t and $\mathcal{S}(t)$ is the (unique if it exists) job *scheduled* at time t. If there is no job scheduled at time t we write $\mathcal{S}(t) = \perp$.

In a trace σ, the *service* $serv^\sigma(j,t)$ received by a job j at time t is 1 if $\mathcal{S}(t) = j$ and 0 otherwise. The *cumulative service* for a job j over a time interval I is

$$serv^\sigma(j, I) := \sum_{t \in I} serv^\sigma(j,t)$$

In practice, the set of execution traces of a system is constrained by its task model and scheduling policy. From now on, we consider a set $\mathcal{T}$ of execution traces of TS.

Based on the previous definition, we can formally define a deadline miss.

Definition 9 (Deadline miss)**.** A job j *misses its deadline* if $serv^\sigma(j, [act(j), d_j[) \leq cost(j)$.

Figure 2 illustrates these definitions on a system made of three tasks τ_1, τ_2, τ_3 scheduled according to the FPP policy with $\pi_{\tau_1} < \pi_{\tau_2} < \pi_{\tau_3}$ and all deadlines equal to 3. Notice that the third job of τ_1 misses its deadline.

C. Prerequisites for the analysis

Our generic analysis is built on top of the abstract model presented in the previous section and is therefore independent from a specific task model or scheduling policy. We now specify the exact requirements that must be fulfilled for our analysis to apply. As already explained, what we need is a notion of analyzable interval and a corresponding local deadline miss analysis.

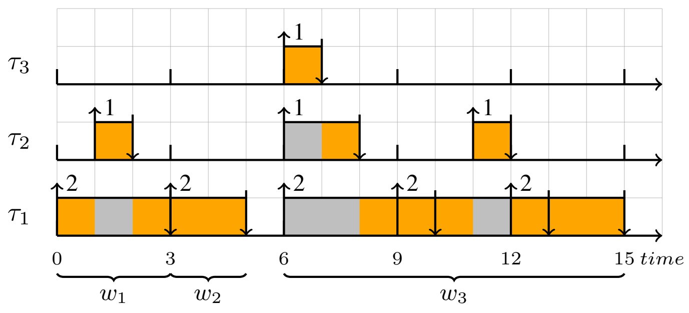

Fig. 2. Prefix of a trace for the system model of Section II. Upwards arrows represent activations with the cost of the corresponding job above, downwards arrows mark the completion of a job. Orange rectangles indicate the currently scheduled task and gray rectangles blocked tasks (tasks that have pending jobs and which are not scheduled).

Following the idea that busy windows in Section II provide sufficient isolation to perform independent local analyses on each of them, we require here a way to partition execution traces into *analyzable windows*. Note that the fact that these windows are indeed analyzable will require an additional property about local deadline miss analysis.

Hypothesis 1 (Analyzable windows)**.** Given a task τ and a trace $\sigma \in \mathcal{T}$, we assume given a partition AW_τ^σ of the timeline into half open time intervals called τ-analyzable windows.

We write $AW_\tau^\sigma(t)$ for the τ-analyzable window around time t and, by abuse of notation, $AW_\tau^\sigma(j)$ for the τ-analyzable window around the activation of a job j of τ. In the example, the visible τ_1-analyzable windows are denoted by w_1, w_2, w_3 and we have $AW_{\tau_1}^\sigma(9) = [6, 15[$.

The analysis of Section II performs a local deadline miss analysis for each busy window of a scenario based on an abstraction of the actual trace called combination which captures the presence of overload tasks in a busy window. In order to address the precision issue illustrated in Example 1, we choose a more precise abstraction based on the multiset of tasks activated in an interval.

Definition 10 (Combination)**.** A *combination* C is a multiset of tasks in TS. $C(\tau)$ denotes the multiplicity of task τ in C. In a given trace $\sigma \in \mathcal{T}$, for a time interval I, $c^\sigma(I)$ is the combination which associates to each task its number of activations in I. We say that σ *maps to* $c^\sigma(I)$ in I.

In our running example of Figure 2, the combination for busy window w_1 is $[0, 1, 1]$.

Based on the decomposition of a trace into analyzable windows and the means to abstract the trace within an analyzable window by a combination, we can now give the specification of the local deadline miss analysis.

Hypothesis 2 (Local analysis)**.** We assume given an analysis which, given a combination C, provides a bound $N_\tau(C)$ such that: for any $\sigma \in \mathcal{T}$, for any $w \in AW_\tau^\sigma$ where $c^\sigma(w) = C$, the number of deadline misses in w is bounded by $N_\tau(C)$.

Note that, the local deadline miss analysis of Section II can be summarized as follows:

- If a combination C is schedulable, then there are no deadline misses in a busy window that maps to it so we can take $N_\tau(C) := 0$.
- Otherwise there are at most N_τ deadline misses so $N_\tau(C) := N_\tau$ is a correct upper bound.

We see in Section IV how to adapt the response time analysis for FPP to get a local analysis using our notion of combination rather than the one of the state of the art.

Finally, in order to formulate our optimization problem, it is convenient to consider a finite number of possible combinations — which is the case whenever we can bound the length of τ-analyzable windows.[4].

Hypothesis 3 (Possible combinations)**.** We assume given a finite over approximation $\mathcal{C}$ of the set of combinations which can appear in τ-analyzable windows.

D. Bounding deadline misses

Now that we have specified the prerequisites of the analysis, let us show how we can compute a DMM if our three hypotheses are fulfilled. The analysis that we present here is formally verified in Coq.

Let us focus on a task τ for which we want to compute a DMM. We find it more convenient to reason about time intervals of bounded length rather than k-sequences of activations of τ. Therefore, we first focus on the problem of finding, for a given duration dt, an upper bound denoted $gdmmd_\tau(dt)$[5] on the number of deadline misses for jobs of τ activated within any time interval I of length at most dt. Once we have a $gdmmd_\tau(dt)$, we can easily derive an upper bound $gdmm_\tau(k)$ for a k-sequence of τ by taking $gdmm_\tau(k) := gdmmd_\tau(\delta_\tau^+(k))$.

In the spirit of the original analysis, let us reduce the computation of a DMM to an optimization problem. Since $gdmmd_\tau(dt)$ is an upper bound on the number $dm_\tau^\sigma(I)$ of deadline misses for τ over any time interval I of length $|I| \leq dt$ in any trace $\sigma \in \mathcal{T}$, we first focus on bounding $dm_\tau^\sigma(I)$ for a given σ and I.

Consider $\sigma \in \mathcal{T}$ and a time interval I over σ. We know from σ exactly what happens in I, so we can count the number of deadline misses in I. Our objective however is to upper bound that number using an abstraction, based on combinations, of the actual activation sequence in I. The first step is to partition I into τ-analyzable windows. Note that the first window may start before the beginning of I and the last one may end after the end of I.

[4]This is also required in practice to compute the DMM as the number of variables in the optimization problem is equal to the number of possible combinations.

[5]The notations $gdmm$ is used to avoid ambiguity with dmm used in the previous section. $gdmmd$ is used for the DMM based on interval lengths.

	w_1	w_2	w_3
τ_3	0	0	1
τ_2	1	0	2
τ_1	1	1	3
Misses	0	0	2

Fig. 3. The abstract scenario corresponding to Fig. 2.

Definition 11 (Window cover)**.** For a given σ and I, $\mathcal{A}_\tau^\sigma(I)$, called a *window cover* of I, is the set of τ-analyzable windows in AW_τ^σ that intersect I:

$$\mathcal{A}_\tau^\sigma(I) := \{AW_\tau^\sigma(t) \mid t \in I\} \tag{10}$$

In our running example, the window cover for the time interval $I = [2, 6[$ is $\{w_1, w_2, w_3\}$ since I intersects the three busy windows.

The trace in each analyzable window w covering I is abstracted by its corresponding combination $c^\sigma(w)$, which indicates the number of activations of each task in w. We can now abstract the concrete trace into an abstract scenario where we are only concerned about the combinations appearing in an interval.

Definition 12 (Abstract scenario)**.** An abstract scenario is a function that maps combinations to a number of occurrences.

Let $S_C^\sigma(I)$ denote the number of τ-analyzable windows in $\mathcal{A}_\tau^\sigma(I)$ which map to combination C in I, that is

$$S_C^\sigma(I) := |\{w \in \mathcal{A}_\tau^\sigma(I) \mid c^\sigma(w) = C\}| \tag{11}$$

The *abstract scenario* for I is the collection

$$S^\sigma(I) := \langle S_C^\sigma(I) \mid C \in \mathcal{C}\rangle \tag{12}$$

mapping each combination to its number of occurrences in the window cover $\mathcal{A}_\tau^\sigma(I)$.

Figure 3 illustrates on the running example our abstraction based on combinations of the trace prefix of Figure 2: Each column represents the combination in one analyzable window in the cover. The last row shows the maximum number of deadline misses for τ_1 in a window mapping onto such a combination as obtained by the local deadline miss analysis.

Note that, our abstract scenario leads to a loss of precision that is twofold:

- An analyzable window is abstracted by a combination, and therefore the exact activation time and cost of jobs is abstracted away.
- The relative positions of analyzable windows within an abstract scenario is not preserved.

Let $dm_\tau^\sigma(I)$ denote the number of deadline misses for activations of τ in interval I. We can now use the local analysis to safely bound this quantity using only the abstract scenario, by grouping windows according to their combination.

Theorem 1 (Deadline miss bound)**.**

$$dm_\tau^\sigma(I) \leq \sum_{C\in\mathcal{C}} N_\tau(C) \times S_C^\sigma(I) \tag{13}$$

Proof. We prove the inequality in 3 steps:

- The activations of τ can be grouped according to the analyzable window in which they occur. This leads us to consider all activations in the window cover of I. This step relies on the fact that analyzable windows form a partition. We deduce that

$$dm_\tau^\sigma(I) \leq \sum_{w\in\mathcal{A}_\tau^\sigma(I)} dm_\tau^\sigma(w)$$

- The local analysis bounds the number of deadline misses for τ in a window w according to the combination $c^\sigma(w)$ exhibited by w.

$$\sum_{w\in\mathcal{A}_\tau^\sigma(I)} dm_\tau^\sigma(w) \leq \sum_{w\in\mathcal{A}_\tau^\sigma(I)} N_\tau(c^\sigma(w))$$

- For any analyzable window w we have that $c^\sigma(w) \in \mathcal{C}$, so we can rewrite the previous sum by counting the number of occurrences of each combination.

$$\sum_{w\in\mathcal{A}_\tau^\sigma(I)} N_\tau(c^\sigma(w)) = \sum_{C\in\mathcal{C}} N_\tau(C) \times S_C^\sigma(I)$$

Hence the result. □

We can now relate deadline misses in I to deadline misses in its corresponding abstract scenario. Let us now suppose that $|I| \leq dt$ and bound the number of deadline misses in I by the maximum number of deadline misses in all possible abstract scenarios over time intervals of size at most dt.

Here, we have to consider the possibility that we cannot bound the number of activations in any interval of length dt. In that situation it is clear that we cannot bound the number of deadline misses in any interval of length dt.

1) DMMs for arrival curves: Let us consider for a moment that we have a bound $\eta_{\tau'}^+(dt)$ on the number of activations of any task τ' in any time interval of size dt and a bound B_τ on the length of a τ-analyzable window. From this information we can derive a constraint $\Omega_{\tau'\to\tau}(dt) := \eta_{\tau'}^+(dt + 2B_\tau)$ on the number of activations of τ in any abstract scenario. This bound is closely related to the constraint on rows in Section II. Under these assumptions we provide the following DMM:

Theorem 2 ($gdmmd_\tau(dt)$ for arrival curves)**.**
$dm_\tau^\sigma(I) \leq gdmmd_\tau(dt)$ with
$gdmmd_\tau(dt) :=$

$$\max_{\langle X_C \mid C\in\mathcal{C}\rangle} \left\{ \sum_{C\in\mathcal{C}} N_\tau(C) X_C \,|\, \forall\tau', \sum_{C\in\mathcal{C}} X_C C(\tau') \leq \Omega_{\tau'\to\tau}(dt) \right\} \tag{14}$$

Proof. Using Theorem 1, it suffices to show that

$$\sum_{C\in\mathcal{C}} N_\tau(C) \times S_C^\sigma(I) \leq gdmmd_\tau(dt)$$

This follows from the fact that the abstract scenario $S^\sigma(I)$ satisfies the constraint given by $\Omega_{\tau'\to\tau}(dt)$, that is, for any task τ' :

$$\sum_{C\in\mathcal{C}} S_C^\sigma(I) \times C(\tau') \leq \Omega_{\tau'\to\tau}(dt)$$

Note that the constraint on scenarios imposed by $\Omega_{\tau' \to \tau}(dt)$ implies that we only consider a finite number of scenarios. $\square$

This result allows us to reduce the problem of finding a deadline miss model to an Integer Linear Programming problem.

2) DMMs in the general case: Going back to the general case, we see that the only required property of the constraint in the optimization problem is that it must hold on any concrete trace. We can thus prove a general result for any sound constraint on abstract scenarios, that is, for any set $P(dt)$ of abstract scenarios covering all possible $S^{\sigma'}(I')$, where I' is a time interval of length at most dt and $\sigma' \in \mathcal{T}$. This generic result is used in the formal proof in order to stay independent from the chosen activation model and to prove Theorem 2.

Hypothesis 4 (Constraint on abstract scenarios)**.** Suppose given a set $P(dt)$ of abstract scenarios such that, for any $\sigma' \in \mathcal{T}$ and any time interval I' such that $|I'| \leq dt$, we have $S^{\sigma'}(I') \in P(dt)$.

Following the same reasoning as in the case of arrival curves, we get the following bound:

Theorem 3.

$$dm_\tau^\sigma(I) \leq \sup_{X \in P(dt)} \left\{ \sum_{C \in \mathcal{C}} N_\tau(C) X_C \right\} \tag{15}$$

Proof. The proof is the same as for Theorem 2. Instead of constraining abstract scenarios with $\Omega_{\tau' \to \tau}(dt)$, we use an abstract constraint $P(dt)$. $\square$

We now finally obtain our generic DMM for any k-sequence of task τ with $\delta_\tau^+(k) < \infty$:

Theorem 4 ($gdmm_\tau(k)$)**.** *Suppose that k consecutive activations of τ are separated by at most $\delta_\tau^+(k)$ time units, then*

$$gdmm_\tau(k) := \sup_{X \in P(\delta_\tau^+(k))} \left\{ \sum_{C \in \mathcal{C}} N_\tau(C) X_C \right\} \tag{16}$$

is a bound on the number of deadline misses for k consecutive activations of τ.

IV. Instantiating the proof to well-known scheduling policies

In this section, we show how to apply our generic analysis to several scheduling policies. All the following instantiations are done using arrival curves as an activation model in order to focus on the genericity of the analysis *w.r.t.* the scheduling policy (in contrast to the task model). As such we only rely on Theorem 2 in this section. We give a detailed account of the instantiation to the FPP policy, for which we have a proof in Coq, and instantiations to the FPNP and EDF policies. We argue that the interface defined by the hypotheses of the generic analysis does not introduce additional complexity compared to building a TWCA analysis from scratch for a policy by reusing the theory developed in Section II. We follow the same method to instantiate the requirements of the analysis which we recall here :

(H1) A notion of τ-analyzable window
(H2) A local analysis to compute the $N_\tau(C)$
(H3) A set of possible combinations $\mathcal{C}$

A. TWCA for FPP and FPNP

Since the analysis is built as a generalization of the TWCA for FPP with arrival curves, it is natural to look at its instantiation to the original model. We now see how to use Theorem 2 for the system model with FPP and arrival curves.

(H1) We naturally use the concept of level-τ busy window defined in Section II as analyzable window since the latter is meant to be a generalization of the former.

(H2) The state-of-the-art analysis includes a way to bound the size of level-τ busy windows by some BW_τ.

(H3) We can deduce that there are at most $\eta_{\tau'}^+$ activations of τ' in a τ-analyzable window. This gives us the following set of possible combinations:

$$\mathcal{C} := \{C \mid \forall \tau', C_{\tau'} \leq \eta_{\tau'}^+(B_\tau)\}$$

where the combination C is a multiset of tasks and $C_{\tau'}$ denotes the number of occurrences of τ' in C.

The local analysis is based on the analysis of Section II-B and Equation 6, which provides a bound on the number of deadline misses in a busy window. For a task τ_i, a combination C and $q \geq 1$, we need to refine the computation of $B_\tau(q)$ in order to take into account the information provided by the combination. The following $B_\tau(C, q)$ is only valid in a level-i busy window exhibiting the combination C, which is exactly the requirement on the local analysis:

$$\begin{aligned} &B_\tau(C,q) := \min\{\Delta \geq 0 \mid \Delta = F(\Delta)\} \text{ with} \\ &F(x) := q \times W_\tau + \sum_{\tau' \in hp(\tau)} \min(\eta_{\tau'}^+(x), C_{\tau'}) \times W_{\tau'} \end{aligned} \tag{17}$$

The local deadline miss bound $N_\tau(C)$ is then obtained as in Section II. We can now use Theorem 2 to get a deadline miss bound:

Note that the instantiation in our Coq development is slightly different from what we have described. The reason is that our proof reuses the FPP response time analysis of the Prosa library. The latter is (at the moment) more pessimistic than needed as it only computes the maximal length $BW_\tau(C)$ of a busy window exhibiting a combination C. The deadline miss bound is then taken to be $N_\tau(C) := \eta_\tau^+(BW_\tau(C))$. Still, we were able to adapt the analysis to take into account the information provided by the combinations without rewriting it from scratch.

The same method can be applied to Fixed Priority Non Preemptive (FPNP) scheduling. We use the same definition of busy window and combinations as for FPP and only need to add a blocking term to the computation of $B_\tau(C, q)$ as in [11].

B. TWCA for EDF

We outline here the required modifications to the response time analysis in [29] for the EDF scheduling policy to turn it into a local deadline miss analysis. In other words, we show how to systematically derive the result of [19], which introduces TWCA for EDF. We work under the same assumptions as the response time analysis and consider periodic tasks. In order to use Theorem 2, we use the period T_τ as a minimum interarrival time with $\eta^+_\tau(dt) := \lceil \frac{dt}{T_\tau} \rceil$.

(H1) There exists a notion of busy window for EDF, which is a maximal interval $[t_1, t_2[$ such that the resource is in use for any $t_1 < t < t_2$ and idle at t_1 and t_2.

(H2) Spuri describes an analysis [29] to compute a bound L on the length of such a busy window which can be used to define Ω_τ. The response time analysis then computes the worst case response time by considering a set of possible alignments between τ and the other tasks. The idea is that each scenario corresponds to an alignment between the first activation of τ and a critical instant. For each of these scenarios, the maximum response time over consecutive activations of τ is computed and the bound on the response time is the maximum over all scenarios. This analysis only needs a small adjustment to yield the worst case response time of the q-th activation of τ in a busy window, which yields a local deadline miss analysis.

(H3) The last requirement for the analysis is a set of possible combinations. We can reuse the set of combinations of the previous section by using η^+_τ and L as a bound on the size of busy windows. In order to take into account the local information provided by combinations when bounding the length of busy windows and processing times in a busy window, we can proceed as for FPP by refining the bounds to the number of activations for a task τ' with $C_{\tau'}$. It is also possible to further reduce the set of combinations in the periodic case as we discuss next.

C. TWCA for tasks with dependencies

The analysis developed for task models based on arrival curves can be used for any model for which the activation pattern can be approximated by an arrival curve η^+. In addition, more fine-grained activation patterns can be taken into account by restricting the set of possible combinations. For instance, if we consider two periodic tasks τ and τ' with periods T_τ and T'_τ such that $T_\tau = n \times T_{\tau'}$, we only have to consider combinations C such that $n(C_\tau - 1) \leq C_{\tau'} \leq n(C_\tau + 1)$. Another example where restricting the set of possible combinations would be meaningful is that of tasks with dependencies. Such constraints can be directly expressed in our generic optimization problem and open up interesting possibilities of extension for TWCA.

V. Discussion

In this section, we discuss our experience developing proofs using the Coq proof assistant and explore the questions raised by our approach regarding the future of TWCA.

A. Proof development in Coq

Our proof [4] is built on top of the Prosa [10] library, which aims at formalizing models of real time systems and proving analyses using the Coq proof assistant. We plan to integrate our development to the main branch in the near future. This will allow us to get new instantiations of our analysis "for free" as the library grows by sharing the basic definitions of traces with the rest of the library.

The proof of the initial TWCA analysis took around two man-months, the generic analysis three man-weeks and the instantiation around two man-weeks. The Prosa library allowed us to reuse the formalization of traces and the specification of system models for the instantiation. The use of the SSReflect library [2] made proofs of arithmetic properties much easier.

The most obvious advantage of formal proofs is that they provide strong guarantees: one only needs to read the specification and the main correctness theorem; there is no need to look at the proof itself which is machine-checked. Furthermore, using a proof assistant brings other benefits. Since the assumptions are explicit and tracked by Coq, we can rely on the proof checker to guide generalization. In our case, the proof started as a proof of the original TWCA for FPP with arrival curves [27]. The proof assistant allowed us to see which properties were required to prove the analysis. After removing the direct dependencies on FPP, mostly syntactic occurrences of priorities, the relevant properties were isolated. This led us to the requirements of Section III on busy windows which allowed us to develop the generic analysis. The generic proof reuses parts of the original proof in a safe way by replacing references to properties of the system model by properties which are consequences of the hypotheses in Section III. The use of a proof assistant was critical in identifying the relevant assumptions for the generic analysis.

B. Perspectives

We have chosen to use a more precise notion of combinations than in [32] to abstract analyzable intervals and show in the appendix how to relate the two notions. We do not use a decomposition into typical and overload tasks in the generic analysis but see it, instead, as an optimization for scalability of the analysis. The fact that the analysis does not rely on a definition of typical component also means that we can define the typical component depending on the problem at hand. For example, we could consider a threshold in the number of a task activations to delimit the overload. These modifications to the analysis do not require to start a new proof from scratch. Instead, we show how to derive them from the generic analysis.

One consequence of this approach for generalization is that we do not have to commit to a specific definition of analyzable window. This allows us to use existing definitions available in the literature, as in Section IV, and may allow our analysis to be compatible with more generic concepts of isolation such as the scheduling horizon in [12]. We only specify the required properties of these intervals relevant to the analysis. It is possible to use our analysis with a notion of analyzable window that only partially isolates jobs, provided that the local analysis takes into account interference from outside the window.

Note that the approach we use here to bound the number of deadline misses can be easily adapted to count other events such as properties on task chains or generalized deadline misses [6].

VI. Related work

Research related to our work falls into two categories: analysis of weakly-hard real-time systems and formal proofs of analysis techniques for real-time systems.

A. Analysis of weakly-hard real-time systems

There are surprisingly few results on the analysis of weakly-hard real-time systems.

The first paper on the subject [17], as well as [9], focus on enforcing (m, k)-firm constraints (no more than m out of k deadline misses) by resorting to dedicated scheduling mechanisms. Our approach is therefore more in the line of [8], [30] which focus on the analysis of such systems. Those papers, however, only apply to a restricted class of systems running periodic, independent tasks [8].

[28] addresses a similar issue in the context of Real-Time Calculus. Rare events represent the possible deviation from the nominal timing model and the proposed analysis computes the settling-time *i.e.,* the longest time window after the rare event until the system returns to normal. However, the system model is quite restrictive here too: only one task may experience rare events and no second rare event may occur before the first one has settled. This line of work has not been pursued further.

Typical Worst-Case Analysis (TWCA) [27] is another approach which can handle not only tasks with complex activation models [27], but also task dependencies [18], finite ready queues [6] and, even recently, multiprocessor architectures [5]. Work in this area progresses in two directions: generalization to other scheduling policies and more realistic system models on the one hand, and reduction of the pessimism in the computed bounds on the other hand (see, for example, [32]).

Note that there is also a growing literature focusing on the control side of the problem [15], [23].

None of the above mentioned papers provide a formal framework for their proofs, which makes it difficult to build on top of the existing results.

B. Formal proofs of analysis techniques for real-time systems

The advent of large-scale mechanized proofs constitutes one of the major success stories of the past decades. Projects such as SEL4 [3], CompCert [22] and DeepSpec [7] show that proof assistants (in particular Coq [1]) have now achieved a maturity level such that they can be used for industrial applications.

There are for the moment relatively few formal proofs of real-time systems analysis techniques. Early attempts include [13], [14] and [16] based on the PVS proof assistant, and [24] based Isabelle/HOL aimed at certifying the results of Network Calculus computations [21].

Part of the research community interested in having formal proofs for real-time systems analysis (including some authors of [24] and the authors of the present paper) has now converged to a unique framework called Prosa [10]. Our work is part of this global effort.

VII. Conclusion

We proposed a generic weakly hard schedulability analysis based on the intuitions of Typical Worst Case Analysis. Instead of focusing on a specific system model, our analysis only relies on a notion of analyzable window, a generalization of the pervasive busy windows, and a local deadline miss analysis based on abstractions, called combinations, of such windows. These concepts can be instantiated to various task models: independent tasks described with arrival curves and following the FPP or EDF scheduling policies. We also used our development to prove the state of the art TWCA analysis for FPP, showing along the way how to refine our generic analysis. These refinements include introducing a notion of typical case and taking into account the fact that, under the FPP policy, lower priority tasks do not interfere.

Our generic analysis and its instantiation to FPP have been completely formalized [4] using the Coq proof assistant [1]. These proofs should be integrated into the Prosa library soon.

For further research, we plan to use our generic analysis to define and certify TWCA for other scheduling policies (*e.g.,* mixed FP(N)P policies) or intra-task dependencies task models (*e.g.,* (generalized) multiframe models). We would also like to investigate if our analysis can be adapted for the multiprocessor TWCA [5]. It would also be interesting to certify the ILP solutions in order to get a working certified analyzer using Coq extraction. We should compare TWCA analyses with other approaches (*e.g.,* [9] and [30]) in terms of precision and efficiency. Other abstractions (combinations) should be investigated. Finally it should be possible to make the generic analysis parametric *w.r.t.* combinations under some restrictions.

References

[1] The Coq proof assistant, project web site. https://coq.inria.fr/.

[2] Mathematical components library, project web site. http://math-comp.github.io/math-comp/.

[3] The sel4 microkernel, project web site. https://sel4.systems/.

[4] Coq proofs of typical worst-case analyses. available at https://team.inria.fr/spades/twcaproofs/, 2017-2018.

[5] L. Ahrendts, S. Quinton, T. Boroske, and R. Ernst. Verifying weakly-hard real-time properties of traffic streams in switched networks. In *30th Euromicro Conference on Real-Time Systems (ECRTS)*, 2018.

[6] L. Ahrendts, S. Quinton, and R. Ernst. Finite ready queues as a mean for overload reduction in weakly-hard real-time systems. In *RTNS*, Grenoble, France, 2017.

[7] A. W. Appel, L. Beringer, A. Chlipala, B. C. Pierce, Z. Shao, S. Weirich, and S. Zdancewic. Position paper: the science of deep specification. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 375(2104), 2017.

[8] G. Bernat, A. Burns, and A. Llamosí. Weakly hard real-time systems. *IEEE Trans. Computers*, 50(4):308–321, 2001.

[9] I. Broster, G. Bernat, and A. Burns. Weakly hard real-time constraints on controller area network. In *Proceedings 14th Euromicro Conference on Real-Time Systems. Euromicro RTS 2002*, pages 134–141, 2002.

[10] F. Cerqueira, F. Stutz, and B. B. Brandenburg. PROSA: A case for readable mechanized schedulability analysis. In *28th Euromicro Conference on Real-Time Systems, ECRTS 2016, Toulouse, France, July 5-8, 2016*, pages 273–284, 2016.

[11] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien. Controller area network (can) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):239–272, Apr 2007.

[12] J. F. Diemer. *Predictable architecture and performance analysis for general-purpose networks-on-chip*. Verlag Dr. Hut, 2016.

[13] B. Dutertre. Formal analysis of the priority ceiling protocol. In *21st IEEE Real-Time Systems Symposium (RTSS)*, pages 151–160, 2000.

[14] B. Dutertre and V. Stavridou. Formal analysis for real-time scheduling. In *19th Digital Avionics Systems Conference (DASC)*, 2000.

[15] G. Frehse, A. Hamann, S. Quinton, and M. Woehrle. Formal analysis of timing effects on closed-loop properties of control software. In *Proceedings of the 35th IEEE Real-Time Systems Symposium (RTSS)*, pages 53–62, 2014.

[16] V. Ha, M. Rangarajan, D. Cofer, H. Rues, and B. Dutertre. Feature-based decomposition of inductive proofs applied to real-time avionics software: An experience report. In *26th International Conference on Software Engineering (ICSE)*, pages 304–313, 2004.

[17] M. Hamdaoui and P. Ramanathan. Hamdaoui, m.: A dynamic priority assignment technique for streams with (m,k)-firm deadlines. ieee trans. comput. 44, 1443-1451. 44:1443 – 1451, 01 1996.

[18] Z. A. H. Hammadeh, R. Ernst, S. Quinton, R. Henia, and L. Rioux. Bounding deadline misses in weakly-hard real-time systems with task dependencies. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 584–589, 2017.

[19] Z. A. H. Hammadeh, S. Quinton, and R. Ernst. Weakly-hard real-time guarantees for earliest deadline first scheduling of independent tasks, Under submission.

[20] Z. A. H. Hammadeh, S. Quinton, M. Panunzio, R. Henia, L. Rioux, and R. Ernst. Budgeting under-specified tasks for weakly-hard real-time systems. In *29th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 17:1–17:22, 2017.

[21] J.-Y. Le Boudec and P. Thiran, editors. *Network Calculus*, pages 3–81. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[22] X. Leroy. Formal verification of a realistic compiler. *Communications of the ACM*, 52(7):107–115, 2009.

[23] S. Linsenmayer and F. Allgöwer. Stabilization of networked control systems with weakly hard real-time dropout description. In *56th IEEE Annual Conference on Decision and Control, CDC 2017, Melbourne, Australia, December 12-15, 2017*, pages 4765–4770, 2017.

[24] E. Mabille, M. Boyer, L. Fejoz, and S. Merz. Towards certifying network calculus. In *Interactive Theorem Proving - 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings*, pages 484–489, 2013.

[25] G. Nelissen, J. Fonseca, G. Raravi, and V. Nelis. Timing analysis of fixed priority self-suspending sporadic tasks. In *Proc. of the 27th Euromicro Conference on Real-Time Systems (ECRTS'15)*, 2015.

[26] 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 *The 51st Annual Design Automation Conference 2014, DAC '14, San Francisco, CA, USA, June 1-5, 2014*, pages 44:1–44:6, 2014.

[27] S. Quinton, M. Hanke, and R. Ernst. Formal analysis of sporadic overload in real-time systems. In *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 515–520, 2012.

[28] M. Shirazi, M. Kargahi, and L. Thiele. Resilient scheduling of energy-variable weakly-hard real-time systems. In *Proceedings of the 25th International Conference on Real-Time Networks and Systems, RTNS 2017, Grenoble, France, October 04 - 06, 2017*, pages 297–306, 2017.

[29] M. Spuri. Analysis of Deadline Scheduled Real-Time Systems. Research Report RR-2772, INRIA, 1996. Projet REFLECS.

[30] Y. Sun and M. Di Natale. Weakly hard schedulability analysis for fixed priority scheduling of periodic real-time tasks. 16:1–19, 09 2017.

[31] K. W. Tindell, A. Burns, and A. J. Wellings. An extendible approach for analyzing fixed priority hard real-time tasks. *Real-Time Systems*, 6(2):133–151, Mar 1994.

[32] W. Xu, Z. A. H. Hammadeh, A. Kröller, R. Ernst, and S. Quinton. Improved deadline miss models for real-time systems using typical worst-case analysis. In *27th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 247–256, 2015.

[33] D. Ziegenbein. Industrial requirements and solutions for a suitable interface between function development and real-time systems integration. https://team.inria.fr/spades/files/2017/10/2017-10-15_ESWEEK_BeyondTheDeadline_ZiegenbeinHamannMayer-John.pdf.

APPENDIX

SOTA FROM THE GENERIC RESULT

We present how to use Theorem 2 to derive the state of the art TWCA presented in Section II. This shows that the formal derivation of the state of the art bound from our generic bound is much easier that writing a proof from scratch for TWCA.

Working under the FPP policy with arrival curves as in Section II, we operate under the following assumptions and properties:

- We use the same definition of τ-level busy window.
- We assume that the size of a level-τ busy window is bounded by BW_τ.
- We can compute an over approximation $\mathcal{U}$ of unschedulable combinations. Note that these combinations only record the presence of tasks in $\mathcal{O}_\tau$.
- We assume that there are no deadline misses in the absence of overload tasks in a τ-level busy window.
- We can bound by some $\tilde{N}_\tau$[6] the number of deadline misses of τ in any τ-level busy window.

As in Section III, we start by bounding $dm^\sigma_\tau(I)$, the number of deadline misses for task τ in any interval I of size less than or equal to dt.

Our first goal is thus to prove that $dm^\sigma_\tau(I) \leq dmm_\tau(dt)$ where $dmm_\tau(dt)$ is the DMM from Section II in which we replace $\Omega_{\tau'\to\tau}(k)$ with $\Omega_{\tau'\to\tau}(dt) := \eta^+_{\tau'}(dt + 2BW_\tau)$.

$$dmm_\tau(dt) := \tilde{N}_\tau \times \max_X \left\{ \sum_{\overline{C}\in\mathcal{U}} X_{\overline{c}} \mid \forall\tau', \sum_{\{\overline{c}\in\mathcal{U} \mid \tau'\in\overline{c}\}} X_{\overline{c}} \leq \Omega_{\tau'\to\tau}(dt) \right\} \tag{18}$$

In order to apply Theorem 2, we need to fulfill the hypotheses (H1) through (H3) of the analysis. Since the scenarios used in [32] use sets as combinations and those of Section III use multisets, we need a mapping to relate them. We do this by forgetting the extra information carried by a combination C compared to its counterpart. It is done using the function $f(C) := \{\tau' \in \mathcal{O}_\tau \mid C(\tau') > 0\}$ which forgets the multiplicity of activations and only keeps higher priority overload tasks. Let us now go through the instantiation of each hypothesis.

(H1) We use the notion of level-τ busy window from Section II as our τ-analyzable windows.

(H2) The local analysis checks if a combination is schedulable using $\mathcal{U}$ and relies on $\tilde{N}_\tau$ to bound the number of deadline misses if the combination is classified as unschedulable:

$$N_\tau(C) := \begin{cases} \tilde{N}_\tau & \text{if } f(C) \in \mathcal{U} \\ 0 & \text{otherwise} \end{cases}$$

(H3) We reuse the set of possible combinations from the improved TWCA presented above.

[6]This is the same as N_τ from Section II, we rename it here as $N_\tau(C)$ is used for the local analysis.

Theorem 2 ensures that $dm^\sigma_\tau(I) \leq gdmmd_\tau(dt)$ with

$$gdmmd_\tau(dt) := \max_{\langle X_C \mid C\in\mathcal{C}\rangle} \left\{ \sum_{C\in\mathcal{C}} N_\tau(C) X_C \mid \forall\tau', \sum_{C\in\mathcal{C}} X_C C(\tau') \leq \Omega_{\tau'\to\tau}(dt) \right\} \tag{19}$$

Based on this result, we can prove that $dmm_\tau(dt)$ is a DMM by showing that each candidate scenario satisfying the constraint in $gdmmd_\tau(dt)$ is dominated by a scenario in $dmm_\tau(dt)$. Consider a scenario $\langle X_C \mid C \in \mathcal{C}\rangle$ such that $\forall\tau', \sum_{C\in\mathcal{C}} X_C C_{\tau'} \leq \Omega_{\tau'\to\tau}(dt)$. The corresponding scenario X' in the state of the art DMM is defined as

$$X'_{\overline{C}} := \sum_{\substack{C\in\mathcal{C} \\ f(C)=\overline{C}}} X_C$$

The maximum number of deadline misses is the same in the two scenarios :

$$\sum_{C\in\mathcal{C}} N_\tau(C) X_C = \tilde{N}_\tau \sum_{\substack{C\in\mathcal{C} \\ f(C)\in\mathcal{U}}} X_C = \tilde{N}_\tau \sum_{\overline{C}\in\mathcal{U}} X'_{\overline{C}}$$

Thus, the multiset scenario X is dominated by its counterpart. We can deduce that X' satisfies the constraints of the DMM from the constraints on X :

$$\begin{aligned} \sum_{\substack{\overline{C}\in\mathcal{U} \\ \tau'\in\overline{C}}} X'_{\overline{C}} &= \sum_{\substack{\overline{C}\in\mathcal{U} \\ \tau'\in\overline{C}}} \sum_{\substack{C\in\mathcal{C} \\ f(C)=\overline{C}}} X_C \\ &= \sum_{\substack{C\in\mathcal{C} \\ f(C)\in\mathcal{U} \\ C(\tau')>0}} X_C \\ &\leq \sum_{\substack{C\in\mathcal{C} \\ C(\tau')>0}} X_C \\ &\leq \Omega_{\tau'\to\tau}(dt) \end{aligned}$$

Then, we can conclude that $gdmmd_\tau(dt) \leq dmm_\tau(dt)$. Applying the same reasoning used in Section III to convert a DMM over bounded intervals to a DMM for k-sequences and equation (19), we can conclude that

$$dm^\sigma_\tau(I) \leq dmm_\tau(k)$$

We have seen how to derive the state of the art TWCA from our generic proof. It also shows that the use of a more precise notion of combinations still allows us to pick a trade-off between precision and scalability without rebuilding the whole theory. This is especially useful with machine checked proofs where such a derivation can save much work compared to a proof from scratch.