Weakly-Hard Real-Time Guarantees for Weighted Round-Robin Scheduling of Real-Time Messages

Zain A. H. Hammadeh
TU Braunschweig, Germany
Email: hammadeh@ida.ing.tu-bs.de

Rolf Ernst
TU Braunschweig, Germany
Email: ernst@ida.ing.tu-bs.de

***Abstract*—Communication resources often exist in distributed real-time systems, therefore, providing guarantees on a pre-defined end-to-end deadline requires a timing analysis of the communication resource. Worst-case response time analysis techniques for guaranteeing the system's schedulability are not expressive enough for weakly-hard real-time systems. In weakly-hard real-time systems, the timing analysis ought to ensure that the distribution of the system deadline misses/mets is precisely bounded. In this paper, we compute weakly-hard real-time guarantees in the form of a deadline miss model using Typical Worst-Case Analysis for real-time messages with weighted round-robin scheduling. We evaluate the proposed analysis's scalability and the tightness of the computed deadline miss models. We illustrate also the applicability of our analysis in an industrial case study.**

I. Introduction and Related Work

Distributed real-time systems often consist of computing resources connected via communication resources e.g. buses. The increasing of the distributed real-time system's complexity makes timing analyses notoriously difficult. *Response time analysis* [22] is a well established technology to verifying timing requirements, e.g. in automotive domain, where it considers the worst-case scenario during which a real-time task experiences its *worst-case response time*. However, bounds that are reported by the response time analysis might be overly pessimistic and they are not expressive enough for systems in which missing deadlines is admissible as long as the system's behavior is not jeopardized. Such systems are called *weakly-hard real-time* systems.

A weakly-hard real-time (WHRT) system [3] is a system where, within a time interval Δ, the distribution of its missed/met deadlines is precisely bounded. The notation (m, k) [8] is used to indicate these distributions. (m, k) means that there are no more than m deadline misses out of any consecutive k deadlines. A wide range of real-time systems fill in the category of WHRT systems where an individual deadline miss will not cause a failure [2]. A clear example of WHRT systems is *image processing* systems where the admission of deadline misses is related to human perception. Another well-known example is *control systems* where research works showed that the required control performance is satisfied in spite of missing up to m deadlines in a sequence of k deadlines [15][7].

In a communication resource, a deadline might be a pre-defined time-out or a share of the end-to-end deadline. In both cases, a deadline miss may cause a deadline miss for the successor task in the next resource and/or an end-to-end deadline miss. If non of those three deadline misses leads to an error, the deadline miss is then admissible. When a task tolerates m deadline misses (not necessary consecutive) in a sequence of k deadlines, this (m, k) represents its WHRT constraint (*(m, k)-constraint*). This work focuses on verifying these (m, k)-constraints, but not on extracting them, for messages with weighted round-robin(WRR).

Whereas the classical round-robin is used for *soft* real-time systems or for *best efforts* QoS, WRR is used for hard real-time systems such as Asynchronous Transfer Mode (ATM) local area network [14] and Ethernet switches [20]. In this paper, we compute WHRT guarantees[1] for real-time messages which share a temporary overloaded resource where WRR is considered. Messages may miss their deadlines due to transient overload resulting from sporadic messages, e.g. generated by interrupt service routines. For that purpose, we adopt the *Typical Worst-Case Analysis* (TWCA) [9], which is a response time based analysis to compute WHRT guarantees for temporarily overloaded systems. TWCA is applicable for systems with Static Priority Preemptive (SPP) and Non-Preemptive (SPNP) scheduling polices which covers computing and communication resources like CAN bus [12] and Switched Ethernet [1].

TWCA is, however, not the only proposed analysis to address the problem of computing WHRT guarantees. Besides the original work of Bernanrt et al. [3], several works have been published considering different system models. In [3], WHRT guarantees are computed for periodic tasks with static priorities. [19] extended [3] to offset-free periodic tasks. [19] is proper for systems where only small k matters where it does not scale beyond 20 tasks and $k > 10$ because of the complexity of their proposed MILP. In [5], the probability of missing m deadlines in a row was computed for a task in faulty execution conditions due to soft errors. The system model in [5] considers independent sporadic tasks in a uniprocessor system under a static priority scheduling policy. The most recent work concerning WHRT systems is [1] where the first analysis to compute end-to-end WHRT guarantees for distributed real-time systems was presented.

We motivate this work to be a cornerstone for a distributed WHRT scheduling analysis where a local analysis is required whether such a WHRT scheduling analysis follows the *Compo-*

[1]These guarantees state that there will be no more than m deadline misses out of k.

sitional Performance Analysis (CPA) [10][21] or the *Holistic Analysis* [23][18].

The rest of this paper is organized as follows: Our system model and the problem statement are introduced in Section II. Section III recalls the worst-case analysis of WRR which we use in Section V for the core contribution of our work. Then, Section IV explains the principles of TWCA. The core contribution of our paper is shown in Section V. Finally, Section VI shows an evaluation of the proposed analysis and our experimental results while Section VII proposes some conclusions.

II. System Model

We consider a single communication resource[2] shared between a fixed set of messages $\mathcal{M}$, where WRR scheduling policy is used to arbitrate between them. Each message is generated and queued by a sender task. We call the time instant at which a new instance of the message μ_i is ready for transmission as *instantiation time*, while the time instant at which the transmission is done is called the *completion time*. WRR scheduling algorithm assigns the communication resource to the messages in a cyclic fashion where each message μ_i has a predefined time slot θ_i. The maximum resource's cycle time, denoted by WRR_{turn}, can be determined:

$$WRR_{turn} = \sum_{i \in \mathcal{M}} \theta_i \tag{1}$$

When it is μ_i's turn and there is no pending instance, the communication resource will not be assigned to μ_i and the scheduling algorithm checks μ_{i+1}. Otherwise, μ_i occupies the communication resource with no preemption for at most θ_i units of time. If μ_i did not complete transmission at the end of its time slot, it will be preempted by the scheduling algorithm, otherwise, μ_i is allowed to transmit the next pending instances under the FIFO manner if there is any. In other words, WRR is a work conserving scheduling policy. The longest time that a message μ_i needs to be transmitted is called the worst-case transmission time C_i.

The generating of a message μ_i follows an event model. In this work, we define the event models using *arrival curves*.

Definition 1. Arrival curves *are functions $\eta_i^+(\Delta)$, $\eta_i^-(\Delta)$ such that for any time window Δ, $\eta_i^+(\Delta)$ defines the maximum number of instances of μ_i that might be generated within Δ, and η_i^- the minimum. The pseudo-inverse of arrival curves are $\delta_i^-(k)$ and $\delta_i^+(k)$ such that $\delta_i^-(k)$ (respectively $\delta_i^+(k)$) defines the minimum (respectively maximum) time interval during which k consecutive instances of μ_i have been generated.*

In this work, we are interested in temporarily overloaded real-time systems (utilization $\leqslant 1$) where the transient overload is due to sporadic messages (overload messages) e.g. generated by interrupt service routines or recovery tasks and/or due to additional sporadic instances (overloaded messages). Concerning the overload/overloaded messages, we consider

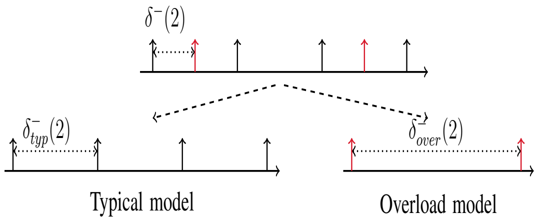

Fig. 1. Typical and overload event models. The red upward arrow refers to a sporadic instance.

another two event models: $(\delta^-_{over}, \infty)$ describes the overload (sporadic) instances and another one $(\delta^-_{typ}, \delta^+_{typ})$ to describe the non-overload instances. We call the former *Overload model* and the latter *Typical model*. The three event models are related such that any trace satisfying (δ^-, δ^+) can be decomposed into a trace satisfying $(\delta^-_{typ}, \delta^+_{typ})$ and another one satisfying $(\delta^-_{over}, \infty)$. Note that an overload (sporadic) message has only an overload model and no typical model, while a non-overloaded message has only a typical model and no overload model. Figure 1 shows an example of typical and overload event models of an overloaded message. Following this consideration, the message set $\mathcal{M}$ can be seen from the overlord's perspective as two message sets: a typical message set $\mathcal{T}$ where all messages are represented only by their typical model and an overload message set $\mathcal{O}$ where all messages are represented only by their overload model. The typical model might be predefined because of the expected behavior of generating the message e.g. periodic or periodic with maximum jitter, in this case, the overload model can be computed using the analysis proposed in [1] where the overload model satisfies $\eta_j^+(\Delta) \leqslant \eta^+_{j,typ}(\Delta) + \eta^+_{j,over}(\Delta)$. In the more general case, when the typical model is not predefined, we define the typical model of a message μ_j w.r.t. message μ_i (where the transmission of μ_j instances may interfere with the transmission of μ_i instances) as the *maximum* event model such that μ_i is schedulable (miss no deadline) and $\eta^+_{j,typ}(\Delta) \leqslant \eta_j^+(\Delta)$. However, we suppose in this work that the typical model and the overload model are given for each message.

Each message has a relative deadline D_i, which is allowed to be arbitrary. Such a deadline represents, in practice of distributed real-time system, a predefined time-out or a share of the end-to-end deadline [17]. We assume that each message can continue transmitting to completion even if a deadline miss occurs.

A deadline miss takes place for an instance l of μ_i when its response time R_i^l exceeds the given relative deadline D_i, where the response time is the time elapsed between the instantiation time and the completion time. The worst-case response time R_i^+ of a message μ_i is then the longest response time of it.

Problem Statement:

For each message μ_i in a message set $\mathcal{M}$ and when WRR is considered, we aim to bound the number of message instances that may miss their deadline within a sequence of k instances

[2]The proposed analysis in this paper applies to communication resources as well as to computing resources

of μ_i in the form of a *Deadline Miss Model (DMM)*.

Definition 2. *A* DMM *for a message μ_i is a function $dmm_i : \mathbb{N} \to \mathbb{N}$, with the property that out of any sequence of k instances of μ_i, at most $dmm_i(k)$ might miss their deadline D_i.*

III. Worst-Case Response Time Analysis

WRR is considered, classically, as a conservative extension of TDMA [16]. In this work we use the busy-window analysis [22] to bound the worst-case response time.

Let act_i^l denotes the instantiation time of the l-th μ_i's instance, and let end_i^l denotes its completion time. The response time of the l-th μ_i's instance is then:

$$R_i^l = end_i^l - act_i^l$$

We are looking for the maximum R_i^l over all μ_i instances.

Definition 3. *A μ_i's busy window is a time interval $[t_1, t_2[$ such that $\forall t \in [t_1, t_2[$ the resource is not idle at t and for each instance of μ_i which has $act_i^l \in [t_1, t_2[$ then $end_i^l \in [t_1, t_2[$.*

Note that when the utilization $\leqslant 1$, then a μ_i's busy window is bounded and it includes a finite number of μ_i's instances.

Lemma 1. *The worst-case response time of μ_i is found in a μ_i's busy window $[t_1, t_2[$ where 1) all messages are simultaneously instantiated at t_1 and right after the μ_i's time slot expired, 2) all messages request the worst-case transmission time and 3) instances are queued at the maximum rate.*

Proof. Under WRR, the resource is assigned to messages in a cyclic fashion, thus, the μ_i instances within $[t_1, t_2[$, which satisfies 1), will have the last slot within WRR_{turn}. That with 2) and 3) can only increase the interference with the transmission of μ_i instances within $[t_1, t_2[$, which implies the existence of a local maxima within $[t_1, t_2[$. $\square$

A μ_i's busy window $[t_1, t_2[$ that satisfies 1), 2) and 3) in the above lemma is the maximum μ_i's busy window [13] and it includes the maximum number of μ_i instances within one μ_i's busy window. Let BW_i^+ denotes the maximum μ_i's busy window and let Q_i denotes the number of μ_i instances within BW_i^+, see Figure2.

To bound BW_i^+ we define the maximum q-instance busy window function $B_i^+(q)$ which returns an upper bound on the time interval between the instantiation time of the first instance and the completion time of the q-th instance:

$$B_i^+(q) := q \,.\, C_i + \sum_{j \in \mathcal{M} \setminus \{i\}} \min\left(\left\lceil \frac{q \,.\, C_i}{\theta_i} \right\rceil \times \theta_j \;, \right.$$
$$\left. \eta_j^+(B_i^+(q)) \times C_j \right) \quad (2)$$

Therefore,

$$Q_i := \min\{q \geqslant 1 : B_i^+(q) \leqslant \delta_i^-(q+1)\} \quad (3)$$

$$BW_i^+ := B_i^+(Q_i) \quad (4)$$

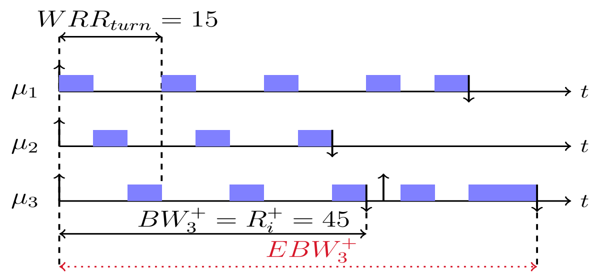

Fig. 2. The μ_3's maximum busy window, the worst-case response time and the extended busy window. Where $C_1 = 25, C_2 = 15, C_3 = 15$ and $\theta_1 = \theta_2 = \theta_3 = 5$. The upward arrow represents the instantiation time while the downward arrow represents the completion time.

The worst-case response-time R_i^+ is then upper bounded by

$$R_i^+ = \max_{1 \leqslant q \leqslant Q_i} \{B_i^+(q) - \delta_i^-(q)\} \quad (5)$$

All needed concepts from the worst-case response time analysis are presented. Next, we present the state of the art of TWCA.

IV. Principle of Typical Worst-Case Analysis

Typical Worst-Case Analysis (TWCA) [9] is a response time based analysis to compute WHRT guarantees for temporarily overloaded systems where tasks may miss their deadlines due to transient overload. For each task τ_i has a typical model $(\delta_{i,typ}^-, \delta_{i,typ}^+)$, TWCA computes a DMM that is a function $dmm_i(k)$ returns the maximum number of deadline misses out of any sequence of k deadlines. We call any consecutive k deadlines of τ_i as a *k-sequence*.

Let us drive this section with an example of 3 tasks scheduled to a resource with SPP in order to clarify how to compute a DMM, see Figure 3, where τ_1 and τ_2 are sporadic tasks thus they have no typical models. Worst-case response time analysis [22] is a foundation for TWCA, and the concept of busy widow is essential for computing DMMs using TWCA. For SPP a τ_i's busy window is a time interval during which the processor is busy w.r.t. τ_i. That is, what occurs before or after a τ_i's busy window will never impact any instance of τ_i within it. TWCA exploits this property of the τ_i's busy window to bound the DMM by:

1) bounding the number of deadline misses within one τ_i's busy window, let N_i denote this bound.
2) bounding the number of τ_i busy windows, which include at least one instance of the k-sequence, that may be impacted (contain at least one deadline miss), let BW_i^{miss} denote this bound.

Then, $dmm_i(k)$ can be bounded safely as

$$dmm_i(k) = N_i \,.\, BW_i^{miss} \quad (6)$$

Formally, N_i is defined as follows:

Definition 4. *N_i is an upper bound on the number of deadlines that τ_i may miss within a busy window.*

Bounding BW_i^{miss} is not straightforward like N_i and it is tricky. In TWCA, we assume that any τ_i's busy window during which no overload instance executes, then τ_i misses no deadline within this busy window, see Figure 3 (d). On the other hand, one or more tasks have to experience overload instances in order to cause a deadline miss within a τ_i's busy window. Figure 3 (a), (b) and (c) illustrate the three possible scenarios in which a τ_3's busy window will be impacted because of experiencing the three possible combinations of the sporadic tasks. The one can observe that τ_3 misses its deadline when a τ_3's busy window experiences the combination $\{\tau_1, \tau_2\}$ or $\{\tau_2\}$. Hence, the first step in bounding BW_i^{miss} is to define the combinations of sporadic tasks which may cause a deadline miss.

Definition 5. *A combination $\bar{c}$ is a subset of the overload task set $\mathcal{O}$.*

Then a combination could be a **schedulable** combination with respect to τ_i (causing 0 deadline misses) or **unschedulable** combination. We need, therefore, a criterion of schedulability CS to classify the combinations. We can then compute the set of unschedulable combinations $\tilde{\mathcal{C}}$.

After defining a metric to address the impacted busy window, we need an upper bound on the number of unschedulable combinations which may interfere with the considered k-sequence. To achieve that we need first to compute the maximum number of overload instances that may impact the execution of any instance of the k-sequence.

Definition 6. $\Omega_k^{j \to i}$ *is the maximum number of overload instances of τ_j which may interfere with any τ_i's busy window containing instances of the k-sequence.*

We use these overload instances to compose unschedulable combinations until we consume the $\Omega_k^{j \to i}$ instances $\forall j \in \mathcal{O}$. Let $x_{\bar{c}}$ represents the number of composed instances of the unschedulable combination $\bar{c}$, then BW_i^{miss} can be bounded as follows:

$$BW_i^{miss} = \max \Big\{ \sum_{\bar{c} \in \tilde{\mathcal{C}}} x_{\bar{c}} \Big\} \tag{7}$$

where

$$\sum_{\bar{c}: j \in \bar{c} \in \tilde{\mathcal{C}}} x_{\bar{c}} \leqslant \Omega_k^{j \to i} \ \forall j \in \mathcal{O} \tag{8}$$

By substituting 7 and 8 in 6 we get the integer linear programming (ILP) formulation in 9 which can be used to compute a DMM for any task $\tau_i \in \mathcal{Z}$.

Example. Let us extend our example to consider 4 tasks where τ_1, τ_2, τ_3 are sporadic overload tasks with $\Omega_k^{1 \to 4} = 2$, $\Omega_k^{2 \to 4} = 2$ and $\Omega_k^{3 \to 4} = 2$. With 3 overload tasks we can combine 8 combinations, assume that the set of unschedulable combinations is $\tilde{\mathcal{C}} = \{\bar{c}_1 = \{\tau_1, \tau_2, \tau_3\}, \bar{c}_2 = \{\tau_1, \tau_2\}, \bar{c}_3 = \{\tau_1, \tau_3\}, \bar{c}_4 = \{\tau_2, \tau_3\}\}$ and the other 4 combinations are schedulable. Using the given $\Omega_k^{s \to 4}$ values we can compose unschedulable combinations in 5 different ways:

1) $x_{\bar{c}_1} = 2 \Rightarrow \sum_{\bar{c} \in \tilde{\mathcal{C}}} x_{\bar{c}} = 2$
2) $x_{\bar{c}_1} = 1, x_{\bar{c}_2} = 1 \Rightarrow \sum_{\bar{c} \in \tilde{\mathcal{C}}} x_{\bar{c}} = 2$
3) $x_{\bar{c}_1} = 1, x_{\bar{c}_3} = 1 \Rightarrow \sum_{\bar{c} \in \tilde{\mathcal{C}}} x_{\bar{c}} = 2$
4) $x_{\bar{c}_2} = 1, x_{\bar{c}_3} = 1 \Rightarrow \sum_{\bar{c} \in \tilde{\mathcal{C}}} x_{\bar{c}} = 2$
5) $x_{\bar{c}_2} = 1, x_{\bar{c}_3} = 1, x_{\bar{c}_4} = 1 \Rightarrow \sum_{\bar{c} \in \tilde{\mathcal{C}}} x_{\bar{c}} = 3$

Then $dmm_4(k) = 3 \times N_4$ is the maximum number of deadline misses that task τ_4 may miss out of any k consecutive deadlines.

V. DMM for Messages with WRR Scheduling

In this section we show how to adopt TWCA to compute DMMs for real-time messages sharing a communication resource under WRR scheduling. The key concept in computing a DMM using TWCA is the busy widow concept, that is, what occurs before or after a μ_i's busy window will never impact any instance of μ_i within it.

To prove the applicability of TWCA to compute DMMs for messages under WRR, we need to define the concept of busy window and we have to show that such a busy window satisfies the required property. In Definition 3, we already defined the μ_i's busy window for WRR. Unfortunately, the μ_i's busy window does not satisfy the preferred property as Figure 2 shows. In Figure 2, μ_1 interferes with two μ_3 instances which belong to two different μ_3 busy windows.

To overcome this limitation, we extend the definition of μ_i's busy window conservatively such that $\forall j \in \mathcal{M}$ if $act_j^l \in [t_1, t_2[$ then $end_j^l \in [t_1, t_2[$, see the red dotted line in Figure 2. Let EBW_i^+ denotes the maximum extended busy window. We can bound EBW_i^+ by the following function:

$$EB_i^+(q) := q \ . \ C_i + \sum_{j \in \mathcal{M} \setminus \{i\}} \eta_j^+(EB_i^+(q)) \times C_j \tag{10}$$

Then:

$$EQ_i := \min\{q \geqslant 1 : EB_i^+(q) \leqslant \delta_i^-(q+1)\} \tag{11}$$

$$EBW_i^+ := EB_i^+(Q_i) \tag{12}$$

Note that the extended busy window is needed only for TWCA and the correctness of the worst-case response time analysis is not impacted and the function $B_i^+(q)$ is still required for computing the worst-case response time.

Now, the key concept of TWCA is defined. However, to compute a DMM we need to:

1) bound N_i, the number of deadlines that μ_i may miss within an extended busy window.
2) define a schedulability criterion in order to determine the unschedulable combinations. See Definition 5.
3) bound $\Omega_k^{j \to i}$, the maximum number of overload instances of μ_j which may interfere with any μ_i busy window containing instances of the k-sequence, to constrain $x_{\bar{c}}$ that is the number of $\bar{c}$ instances.

Compute N_i. The computation of N_i is straightforward where it is found within EBW_i^+

$$N_i = \#\{q | 1 \leqslant q \leqslant EQ_i \wedge R_i^q > D_i\} \tag{13}$$

Schedulability criterion. The goal is to define a criterion to classifying the combinations into schedulable and unschedulable. Computing the set of unschedulable combinations using the worst-case response time analysis is not efficient because

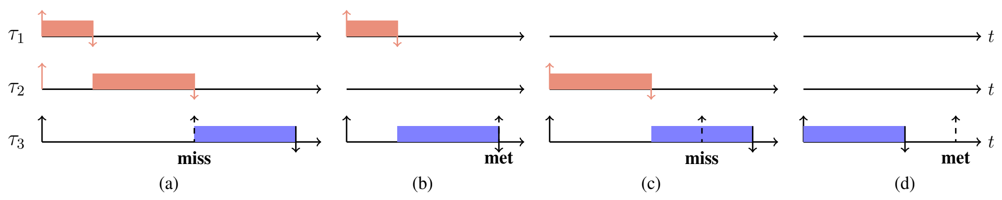

Fig. 3. The impact of overload task combinations on τ_3 busy windows where SPP is considered: $\bar{c}_1 = \{\tau_1, \tau_2\}$ in (a), $\bar{c}_2 = \{\tau_1\}$ in (b), $\bar{c}_3 = \{\tau_2\}$ in (c) and $\bar{c}_4 = \{\emptyset\}$ in (d). The dashed arrow represents D_3 and the red color refers to the overload tasks (τ_1, τ_2). Note that $N_3 = 1$.

$$dmm_i(k) \overset{def}{=} N_i \ . \ \max\Big\{ \sum_{\bar{c} \in \tilde{\mathcal{C}}} x_{\bar{c}} \ : \sum_{\bar{c}: j \in \bar{c} \in \tilde{\mathcal{C}}} x_{\bar{c}} \leqslant \Omega_k^{j \to i} \ \forall j \in \mathcal{O}, \, x_{\bar{c}} \in \mathbb{N} \ \forall \bar{c} \in \tilde{\mathcal{C}} \Big\} \tag{9}$$

it is a fixed-point problem (see Equation 2) and therefore it is of a high complexity. Note that if $R_i^+ \leqslant D_i$ then all instances meet their deadline, thus, it is sufficient and necessary to study the schedulability within the maximum busy window BW_i^+.

A μ_i's instance (l) misses its deadline if and only if its response time is larger than the relative deadline, thus, the response time has to be reduced by Λ_i^l to meet the deadline:

$$\Lambda_i^l := R_i^l - D_i \tag{14}$$

where $R_i^l = B_i^+(l) - \delta_i^-(l)$ is the response time of the l-th instance in BW_i^+, see Figure 4.

Λ_i^l comprises two parts of interfering workload: that appears before D_i and the workload taking place after D_i. The latter, denoted by Γ_i^l, will not interfere any more with the μ_i's instance if it meets its deadline. Hence, what we need to consider in defining a schedulability criterion is the interfering workload which appears before the deadline $(\Lambda_i^l - \Gamma_i^l)$. It is sufficient then to reduce the interfering workload by $\Lambda_i^l - \Gamma_i^l$ to ensure that the deadline will be met.

If $\mu_j \notin \bar{c}$, then its sporadic overload $wl_{j,over}^l$ will not contribute to the response time of μ_i when $\bar{c}$ is considered, where

$$wl_{j,over}^l := \min\Bigg(\Big\lceil \frac{l \ . \ C_i}{\theta_i} \Big\rceil \times \theta_j \ , \ \eta_{j,over}^+(\delta_i^-(l) + D_i) \times C_j \Bigg) \tag{15}$$

If μ_i itself is overloaded (it has an overload event model):

$$wl_{i,over}^l := \eta_{i,over}^+(\delta_i^-(l)) \times C_i \tag{16}$$

When $\bar{c}$ is considered and

$$\sum_{j \notin \bar{c}} wl_{j,over}^l \geqslant \Lambda_i^l - \Gamma_i^l$$

then we ensure that the deadline will be met and $\bar{c}$ is a schedulable combination.

To bound Γ_i^l, we bound the interfering workload up to the completion time of l-th instance $(B_i^+(l))$ and up to the relative deadline $(\delta_i^-(l) + D_i)$:

$$\alpha_j = \min\Bigg(\Big\lceil \frac{l \ . \ C_i}{\theta_i} \Big\rceil \times \theta_j \ , \ \eta_j^+(B_i^+(l)) \times C_j \Bigg) \tag{17}$$

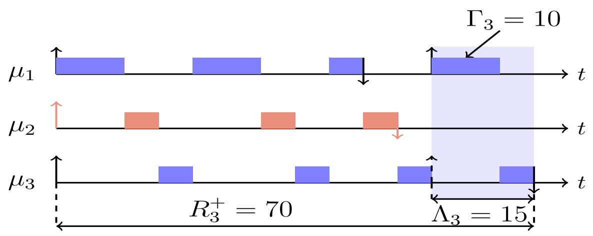

Fig. 4. Λ_3 and Γ_3 where μ_2 is an overload message. The dashed upward arrow refers to D_3.

$$\beta_j = \min\Bigg(\Big\lceil \frac{l \ . \ C_i}{\theta_i} \Big\rceil \times \theta_j \ , \ \Big\lceil \frac{\eta_j^+(\delta_i^-(l) + D_i) \times C_j}{\theta_j} \Big\rceil \times \theta_j \Bigg) \tag{18}$$

Then:

$$\Gamma_i^l := \sum_{j \in \mathcal{M} \setminus \{i\}} \max\Big(\alpha_j - \beta_j, 0\Big) \tag{19}$$

Note that Equation 17 is directly derived from Equation 2, while Equation 18 is modified to cover the case when $\frac{C_j}{\theta_j} \notin \mathbb{N}$. Figure 4 illustrate such a case where $\alpha_1 = \min\{40, 50\} = 40$ which means it bounds the workload before D_i by $\lceil \frac{C_j}{\theta_j} \rceil \times \theta_j$ and therefore it has to be bounded the same way in β_j.

Theorem 1. *A combination $\bar{c}$ is schedulable if (sufficient condition):*

$$\forall l \in [1, Q_i], R_i^l > D_i : \sum_{j \notin \bar{c}} wl_{j,over}^l \geqslant \Lambda_i^l - \Gamma_i^l \tag{20}$$

Proof. When all μ_i instances within BW_i^+ meet the deadline then μ_i is guaranteed to miss no deadline.

However, the bound $\Lambda_i^l - \Gamma_i^l$ is over-approximated such that for a combination $\bar{c}'$ where $\sum_{j \notin \bar{c}'} wl_{j,over}^l < \Lambda_i^l - \Gamma_i^l$, it is still possible that l meets its deadline. That is, our condition is sufficient but not necessary. ☐

Consequently, the following set contains all unschedulable combinations:

$$\tilde{\mathcal{C}} := \{\bar{c} | \forall l \in [1, Q_i], R_i^l > D_i : \sum_{j \notin \bar{c}} wl_{j,over}^l < \Lambda_i^l - \Gamma_i^l\} \quad (21)$$

Compute $\Omega_k^{j \to i}$. What still missing is the maximum number of overload instances that may impact the transmission of any instance of the k-sequence. That is $\Omega_k^{j \to i}$.

Lemma 2.

$$\Omega_k^{j \to i} := \eta_{j,over}^+(EBW_i^+ + \delta_i^+(k) + R_i^+) \quad (22)$$

Proof. $\delta_i^+(k)$: the maximum distance between k instances.

EBW_i^+: if an overload instance shares the same extended busy window with the first instance of the k-sequence, then it may impact its transmission.

R_i^+: any overload instance that is instantiated within the response time of the last instance of the k-sequence, may impact its transmission.

Any overload instance that is instantiated within $EBW_i^+ + \delta_i^+(k) + R_i^+$ may impact the transmission of one or more of the k-sequence instances.

$\eta_{j,over}^+$: returns the maximum number of overload instances that may be instantiated within $EBW_i^+ + \delta_i^+(k) + R_i^+$. □

Compute DMM. Up to now, all intermediate steps have been achieved and therefore the DMM of μ_i can be computed.

Theorem 2. *We can compute a DMM for any message $\mu_i \in \mathcal{M}$, where WRR is considered, using the ILP in Equation 9.*

Proof. On one hand, within one extended busy window there will be no more than N_i deadline misses.

On the other hand, there are no more than $\sum_{j \in \mathcal{O}} \Omega_k^{j \to i}$ overload instances that may impact the transmission of the k-sequence instances. Consequently, the number of unschedulable combination instances is bounded, that means, the maximum number of impacted extended busy windows (each includes at least one deadline miss) can be bounded.

The ILP in Equation 9 provides, therefore, an upper bound on the number of deadlines that might be missed out of any k consecutive deadlines. □

VI. Experiments and Discussion

A. Experiments

In this work, we derived a case study from the WATERS 2015 industrial challenge[3]: Aerial video tracking system. The considered case study is illustrated in Figure 5 where we are interested in the communication resource. That is, we have a single resource with 4 messages scheduled according to WRR policy. The timing characteristics are shown in Table I. The worst-case response time analysis showed that no message misses its deadline, see the most right column in Table I. The worst-case analysis was performed using pyCPA [6].

We considered the given model as the typical model of the system and we added then a synthetic sporadic overload for each message as follows: We have a typical utilization $U_{typ} \approx 0.59$, we chose an overload utilization as following

$$U_{over} = rand(0.01\%, 4\%) * U_{typ}.$$

We then applied UUnifast [4] to assign a share for each message. In the next step we generated a random trace of Z instances where we have in practice

$$\delta_{j,over}^-(Z) = (Z-1)C_j / U_{j,over}$$

with a sufficiently large Z ($Z = 100$ is sufficient for our experiments). That means, there will be an instance at 0 and the last one at $\delta_{j,over}^-(Z)$ and $Z-2$ in between. pyCPA was used to provide the observed minimum distance functions.

For the worst-case model we computed

$$\eta_j^+(\Delta) = \eta_{j,typ}^+(\Delta) + \eta_{j,over}^+(\Delta)$$

therefore, $\delta_j^-(2) = 0$.

Through the experiments we address mainly the following points:

- The tightness of results
- The impact of sporadic overload's share
- The influence of scheduling policy WRR

Results: We repeated the experiment 8000 times where in each time we generated a synthetic sporadic overload. We divided the U_{over}'s range into 8 sub-ranges with 1000 iterations per sub-range. The sub-ranges are shown in Table III. We computed then $dmm_i(k)$ for the 4 messages in the system where $k = 10, 100, 1000$.

[3]http://waters2015.inria.fr/challenge/

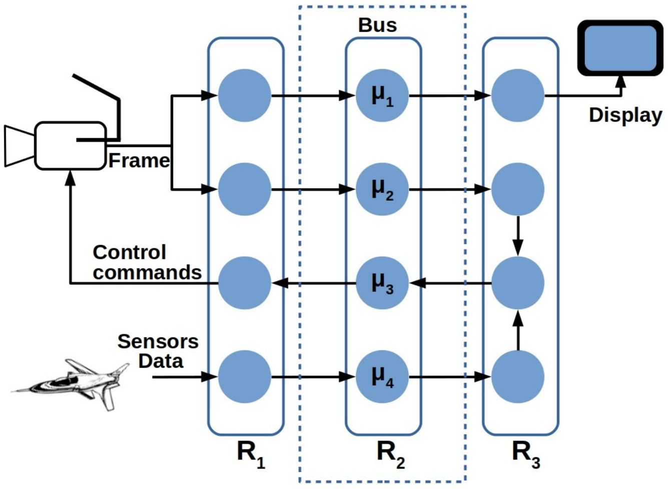

Fig. 5. Aerial video tracking system

Message	P	J	C	θ	D	R^+
μ_1	40	2	6	2	38	26
μ_2	40	2	6	3	38	20
μ_3	40	20	4	4	20	12
μ_4	100	20	6	3	80	20

TABLE I
TIMING CHARACTERISTICS OF MESSAGES IN R_2

k	10			100			1000		
Message	μ_1	μ_2	μ_3	μ_1	μ_2	μ_3	μ_1	μ_2	μ_3
Min	2	1	1	4	2	1	6	4	3
Q1	4	3	2	9	6	4	26	17.75	12
Q2	6	4	4	12	8	8	36	27	21
Q3	6	6	6	15	12	12	50	39	38
Max	10	10	10	36	24	24	123	100	93

TABLE II
MINIMUM, QUARTILES AND MAXIMUM OF $dmm_i(k)$ FOR $k = 10, 100, 1000$ AND $U_{over} = 0.006 \times U_{typ}$

The ILP in Equation 9 represents a multidimensional knapsack problem [24]. Finding an approximate algorithm for it is NP-hard [11] and implementing it directly is impractical. Instead, we considered the LP relaxation in [24] where we solved the optimization problem using cplex.

The first observation is that μ_4 misses no deadline as long as $U_{over}/U_{typ} \leqslant 2\%$, while it does in 9 test cases out of 1000 for $2\% < U_{over}/U_{typ} \leqslant 3\%$ and in 14 test cases out of 1000 for $3\% < U_{over}/U_{typ} \leqslant 4\%$. Therefore, we exclude it from the results.

Tightness of results. Table II shows the minimum, the quartiles and the maximum of the set of $dmm_i(k)$ values over the 1000 iterations for $k = 10, 100, 1000$ and $U_{over} = 0.006 \times U_{typ}$. The one can conclude that the results are more pessimistic for smaller k, for instance, the third quartile $Q3$ of μ_1 illustrates that for 750 iterations $dmm_1(10) \leqslant 6$, $dmm_1(100) \leqslant 15$, and $dmm_1(1000) \leqslant 50$. The main sources of pessimism in $dmm_i(k)$ will be discussed in the next subsection. However, the improvement in the tightness for larger k is due to the decreasing of the sporadic overload's density ($\Omega_k^{j\to i}/k$). That is attributed to the sub-additivity property of sporadic event models $\eta^+_{j,over}$.

Scaling with U_{over}/U_{typ}. To clarify the impact of the ratio U_{over}/U_{typ} on $dmm_i(k)$, we reported the minimum, the quartiles and the maximum of the set of $dmm_i(100)$ values over the 8000 iterations (1000 per sub-range) in Table III. k was chosen to be 100 to reduce the impact of pessimism and to focus on the studied parameter (U_{over}/U_{typ}). It is obvious that the ratio has a direct impact on the DMMs of messages where $dmm_i(100)$ increases with increasing U_{over}/U_{typ}. A transient overload equivalent to $U_{over}/U_{typ} \geqslant 3\%$ is prohibitively large and it causes all instances to miss their deadlines.

Scheduling policy's influence. The scheduling policy (WRR) may have an influence on the number of deadline misses. To test that, we plotted in Figure 6 the average of $dmm_i(100)$ for the first 5 sub-ranges of U_{over}/U_{typ}, that is, the most left blue bar represents $average(dmm_1(100))$ over 1000 iterations for $U_{over}/U_{typ} = 0.2\%$. Figure 6 shows that μ_1 misses more deadlines than μ_2 and μ_3 and μ_2 in turn misses more deadlines than μ_3. To explain that, we need to look at Table I where we observe that μ_1 has $C_1/\theta_1 = 3$, μ_2 has $C_2/\theta_2 = 2$ and μ_3 has $C_3/\theta_3 = 1$. This ratio implies that each instance of μ_i may get at most an interference of $(C_i/\theta_i) \cdot (WRR_{turn} - \theta_i)$, in other words, a message with large C_i/θ_i will suffer from any sporadic overload instance more than messages with small C_i/θ_i. The impact of the ratio U_{over}/U_{typ} is also observable in Figure 6.

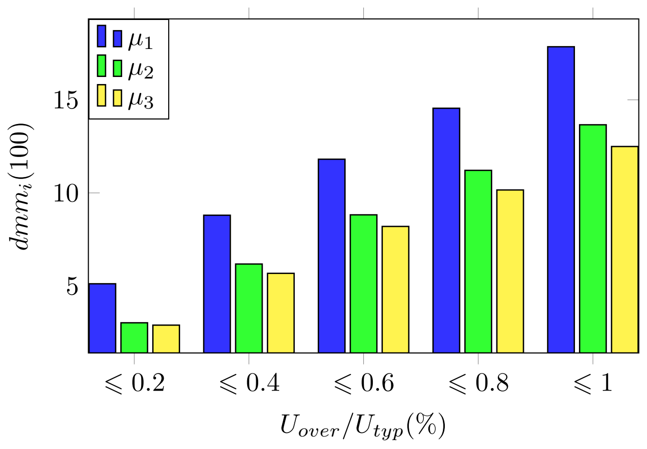

Fig. 6. The scheduling policy's impact on DMMs

Using worst-case analysis we will not be able to check the schedulability of any these 8000 test cases because it goes no farther than $m = 0$. TWCA, however, goes much farther to verify (m, k)-constraint. For instance if μ_1 can tolerate $m = 6$ out of $k = 100$, TWCA can cover 1360 test cases in these experiments while worst-case analysis covers none.

B. Discussion

Computing DMMs using the proposed analysis in this paper has two main sources of pessimism:

1) N_i, because not every unschedulable combination will cause the same number of deadline misses where N_i is only an upper bound,

2) TWCA assumes that every busy window within the time window of the k-sequence experiences the maximum interference from all messages $\mu_j \in \mathcal{T}$, which leads to over-approximate the maximum number of impacted busy windows.

Nevertheless, TWCA provides a low complexity which leads to a high scalability w.r.t. k and the number of overload/overloaded messages in the system. It is a trade-off between the DMM's tightness and the computation complexity.

To justify our claims, let us first focus on k. From Equation 9 we have $\Omega_k^{j\to i}$ as a function of k, see Equation 22. Because of the sub-additivity property that η^+ has (that is $\forall \Delta_1, \Delta_2 > 0 : \eta^+(\Delta_1 + \Delta_2) \leqslant \eta^+(\Delta_1) + \eta^+(\Delta_2)$), $\Omega_k^{j\to i}$ increases *logarithmically* as k increases. No other parameters are impacted by k.

Let n_s denotes the number of overload/overloaded messages in the system ($n_s = \#\{\mathcal{O}\}$). Clearly, the number of unschedulable combinations increases *exponentially* as n_s increases: $\#\{\tilde{\mathcal{C}}\} \leqslant 2^{n_s} - 1$. To overcome this, the LP relaxation in [24], which was considered in our experiments, depends on column-generation based algorithm such that there is no need to generate all unschedulable combinations beforehand.

$U_{over}/U_{typ}(\%)$	$\leqslant 0.2$			$\leqslant 0.4$			$\leqslant 0.6$			$\leqslant 0.8$		
Message	μ_1	μ_2	μ_3	μ_1	μ_2	μ_3	μ_1	μ_2	μ_3	μ_1	μ_2	μ_3
Min	2	1	1	4	2	2	4	2	1	4	2	2
Q1	4	2	2	6	4	3	9	6	4	10	7	6
Q2	5	3	2	9	6	4	12	8	8	15	11	10
Q3	6	4	3	11	8	9	15	12	12	18	15	15
Max	14	12	12	24	20	16	36	24	24	40	32	28
$U_{over}/U_{typ}(\%)$	$\leqslant 1$			$\leqslant 2$			$\leqslant 3$			$\leqslant 4$		
Min	6	3	2	4	3	2	6	5	2	6	6	5
Q1	12	9	7	16	12	12	28	20	18	35	24	24
Q2	15	12	12	24	18	15	36	27	24	50	36	30
Q3	21	18	18	32	24	21	50	39	33	72	55.25	42
Max	54	72	32	85	84	70	100	100	100	100	100	100

TABLE III
MINIMUM, QUARTILES AND MAXIMUM OF $dmm_i(100)$

VII. Conclusion

Computing more expressive guarantees for WHRT systems than those provided by worst-case response time analysis is the reason behind several papers have been published in the last decade. In this work, we computed WHRT guarantees in the form of a DMM using TWCA for real-time messages with WRR scheduling.

The proposed analysis's scalability and the tightness of the computed DMMs have been discussed. A realistic case study was derived from the WATERS'15 industrial challenge in order to illustrate the applicability of the proposed analysis.

This work showed that TWCA is useful to compute WHRT guarantees for temporary overloaded system under WRR scheduling policy and not only for static priority scheduling policies (SPP and SPNP). That paves the way to proposing a heterogeneous distributed WHRT scheduling analysis.

References

[1] L. Ahrendts, S. Quinton, T. Boroske, and R. Ernst. Verifying weakly-hard real-time properties of traffic streams in switched networks. In *The 30th Euromicro Conference on Real-Time Systems (ECRTS18)*, Barcelona, Spain, July 2018.

[2] A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, Jan 2004.

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

[4] E. Bini and G. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Systems*, 30(1-2):129–154, 2005.

[5] K. H. Chen and J. J. Chen. Probabilistic schedulability tests for uniprocessor fixed-priority scheduling under soft errors. In *2017 12th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 1–8, June 2017.

[6] J. Diemer, P. Axer, and R. Ernst. Compositional performance analysis in python with pycpa. In *3rd International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, jul 2012.

[7] 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 IEEE 35th IEEE Real-Time Systems Symposium, RTSS 2014, Rome, Italy, December 2-5, 2014*, pages 53–62, 2014.

[8] M. Hamdaoui and P. Ramanathan. A dynamic priority assignement technique for streams with (m, k)-firm deadlines. *IEEE Trans. Computers*, 44(12):1443–1451, 1995.

[9] Z. A. H. Hammadeh, S. Quinton, and R. Ernst. Extending typical worst-case analysis using response-time dependencies to bound deadline misses. In *Proceedings of the 14th International Conference on Embedded Software*, EMSOFT '14, pages 10:1–10:10. ACM, 2014.

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

[11] M. J. Magazine and M.-S. Chern. A note on approximation schemes for multidimensional knapsack problems. *Mathematics of Operations Research*, 9(2):244–247, 1984.

[12] 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 *Proceedings of DAC*, pages 1–6. ACM, 2014.

[13] R. Racu, L. Li, R. Henia, A. Hamann, and R. Ernst. Improved response time analysis of tasks scheduled under preemptive round-robin. In *Proceedings of the International Conference on Hardware-Software Codesign and System Synthesis*, pages 179–184, Salzburg, Austria, oct 2007.

[14] A. Raha, N. Malcolm, and W. Zhao. Hard real-time communications with weighted round robin service in atm local area networks. In *Engineering of Complex Computer Systems, 1995. Held jointly with 5th CSESAW, 3rd IEEE RTAW and 20th IFAC/IFIP WRTP, Proceedings., First IEEE International Conference on*, pages 96–103, Nov 1995.

[15] P. Ramanathan. Overload management in real-time control applications using (m, k)-firm guarantee. *IEEE Transactions on Parallel and Distributed Systems*, 10(6):549–559, Jun 1999.

[16] K. Richter. *Compositional Scheduling Analysis Using Standard Event Models*. PhD thesis, TU Braunschweig, 2005.

[17] M. Saksena and S. Hong. An engineering approach to decomposing end-to-end delays on a distributed real-time system. In *Proceedings of the 4th International Workshop on Parallel and Distributed Real-Time Systems*, pages 244–251, Apr 1996.

[18] M. Spuri. Holistic analysis for deadline scheduled real-time distributed systems. Technical report, INRIA, France, 1996.

[19] Y. Sun and M. D. Natale. Weakly hard schedulability analysis for fixed priority scheduling of periodic real-time tasks. *International Conference on Embedded Software (EMSOFT), ACM Transactions on Embedded Computing Systems ESWEEK Special Issue*, Oct 2017.

[20] D. Thiele, J. Diemer, P. Axer, R. Ernst, and J. Seyler. Improved formal worst-case timing analysis of weighted round robin scheduling for ethernet. In *2013 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 1–10, Sept 2013.

[21] L. Thiele, S. Chakraborty, M. Gries, A. Maxiaguine, and J. Greutert. Embedded software in network processors - models and algorithms. In *Proceedings of the First International Workshop on Embedded Software*, EMSOFT '01, pages 416–434, London, UK, UK, 2001. Springer-Verlag.

[22] K. 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, 1994.

[23] K. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocess. Microprogram.*, 40(2-3):117–134, Apr. 1994.

[24] 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 *2015 27th Euromicro Conference on Real-Time Systems*, pages 247–256, July 2015.