Guarantees for Runnable Entities with Heterogeneous Real-Time Requirements

Leonie Ahrendts, Zain A. H. Hammadeh and Rolf Ernst
Institute of Computer and Network Engineering
TU Braunschweig, 38106 Braunschweig, Germany
{ahrendts,hammadeh,ernst}@ida.ing.tu-bs.de

***Abstract*—Classical real-time (RT) analysis proves temporal properties of tasks. In industrial practice, however, tasks are often composed of runnable entities with heterogeneous RT requirements. If RT guarantees are only available at task granularity, the strictest RT requirement of a runnable entity determines the RT requirement of the entire task. However, by giving RT guarantees for each runnable entity, this over-provisioning can be avoided. We provide an analysis which is fine-grained enough to provide hard and weakly-hard response time guarantees for runnable entities and show the improvement in an industrial case study.**

I. Introduction

In a number of application domains a task represents a mere collection container for a set of functions that are scheduled as a unit but do not contribute to the same mission and do not share the same real-time (RT) requirements. The clustering of functions in tasks is often necessary since an RT operating system (OS) supports only a limited number of tasks, which is typically exceeded by the number of functions to be scheduled in a complex software system.

An insightful example of this situation is the AUTomotive Open System ARchitecture (AUTOSAR) [1] which is an industrial standard for automotive software (SW) systems. Applications are defined as individual *SW components* and each SW component is composed of a number of possibly communicating *runnable entities* (REs), cf. Figure 1. An RE itself is an independently executable procedure which is activated by events. For the purpose of scheduling, REs from different SW components are mapped to a common task which represents the smallest schedulable SW entity. The RE-to-task mapping generally aims at optimizing performance and reducing resource requirements while ensuring data consistency and timing predictability. For instance, blocking time caused by access to a shared resource or communication can be reduced by appropriate re-ordering of REs. Remarkably, the RE-to-task mapping implies that REs from different applications and with heterogeneous RT requirements belong to the same task. But even if the REs in a task are selected from a single SW component, it is still probable that they differ in their RT requirements. According to the state-of-the-art, a task is classified as imposing hard RT contraints if at least one hard RT RE is included in the task. However, if it were possible to give distinct timing guarantees for each RE in a task, over-provisioning with respect to the RT requirements could be avoided. In this paper, we provide such hard and weakly-hard guarantees for REs and we do so by extending the classical response time analysis (RTA) as well as the recently introduced typical worst-case analysis (TWCA) [2]–[4] to the finer granularity of REs. We assume that REs have either hard, weakly-hard or best-effort RT requirements. An RE which is never allowed to miss a deadline imposes a *hard RT constraint*. An RE which may miss at most m deadlines out of any sequence of k consecutive executions ($0 < m < k$) imposes a *weakly-hard RT constraint*. Weakly-hard constraints are especially useful in the context of signal-processing or control systems where the algorithms may tolerate window-constrained deadline misses. An RE which imposes no constraint on the maximum number of tolerated deadline misses is considered to be a *best-effort* RE.

This work has been partially funded by the German Research Foundation (DFG) as part of the project "TypicalCPA".

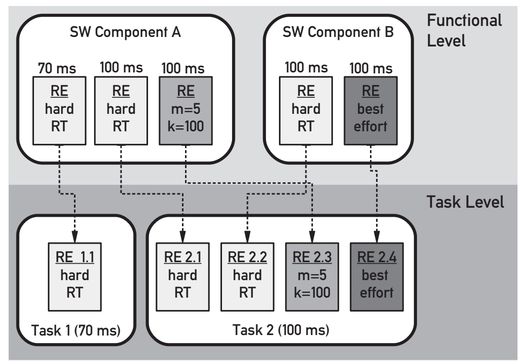

Fig. 1. Example Runnable-Entity-to-Task Mapping

II. Related Work

The two related concepts of (m, k)-firm deadlines [5] and weakly-hard constraints [6] have been introduced as an extension of the traditional classification of RT tasks into soft, firm and hard. The work of [2]–[4] introduced TWCA which is closely related to the concept of weakly-hard tasks. It is assumed that in the so-called typical case, the considered system is schedulable and only in the worst case, which is marked by added sporadic overload to a (sub)set of tasks, deadline misses can be observed. TWCA aims at giving task-related, weakly-hard bounds for deadline misses in the worst case. TWCA, however, does not have any notion about REs inside tasks so far and can therefore not analyze and exploit fine-grained RT requirements.

REs have been foremost considered in the context of AU-

TOSAR. However, the main focus was placed on identifying a mapping of REs to container tasks and communication mechanisms which preserve the communication semantics of the functional model: [7] proposes the proof of absence of interference, disabling of preemption, communication buffers and semaphores as possibilities to ensure data consistency and time determinism on a single-core resource. [8] proposes similar mechanisms for the preservation of communication semantics, however, with respect to the multi-core problem. In [9] a mixed-integer linear programming problem is formulated which provides an optimal execution order of REs inside a container task with respect to minimal memory usage while respecting a set of functional constraints. The work in [10] presents algorithms to best partition and sequence REs in container tasks while uniformizing load over time.

In contrast to that work, we relate to a given RE-to-task-mapping and compute hard and weakly-hard guarantees for individual REs although we are also able to formulate guidelines how to order REs inside container tasks in order to improve RT characteristics.

III. Problem Statement

We focus on an static priority preemptive (SPP)-scheduled resource to which a set of independent container tasks with arbitrary activation patterns and arbitrary deadlines is mapped. The container tasks are composed of REs with heterogeneous RT requirements and each RE $\rho_{i,p}$ inherits the deadline, the priority and the activation pattern from the task τ_i to which it is mapped as it is common industrial practice. The index p indicates that the RE $\rho_{i,p}$ is in the pth position with regard to the composition of task τ_i. REs may communicate via shared variables; alternative communication mechanism will be covered in future work. Further, we suppose that tasks may experience sporadic overload such that deadlines are occasionally missed by REs.

Our objective is to compute a correct and tight upper bound (m, k) with respect to each RE $\rho_{i,p}$. This upper bound indicates the maximum number of deadline misses m that the RE $\rho_{i,p}$ may experience in any sequence of k consecutive instances where k is given. On the basis of this upper bound, we want to verify whether each RE in the system satisfies its given RT constraint.

IV. System Model

We consider a RT computing system with one processing resource. There is a fixed number of *SW components* to be executed on the processing resource. Each SW component consists of a *set of REs* which are independently executable procedures activated by events. Each RE has individual RT requirements ranging from hard through weakly-hard to soft. REs are mapped to *container tasks* (hereafter: tasks) which are scheduled by the OS according to the SPP policy. Once a task is selected for execution by the OS, the REs are executed in the order in which they have been mapped to the task. In this paper, we suppose that the mapping of REs to tasks is given. The mapping may be the result of a heuristic algorithm optimizing a given performance quality. However, we impose the common mapping constraint that a task and its REs must share the same activation pattern. Note that an identical activation pattern does not imply an identical RT requirement.

A task τ_i as an element of the set of independent tasks $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_{|\mathcal{T}|}\}$ is characterized by its worst-case execution time C_i, its priority index Π_i (the smaller the index the higher the priority) and its arbitrary relative deadline D_i. The task is activated by events and the activation trace is equivalently described by the distance functions $(\delta_i^-(n), \delta_i^+(n))$ and the arrival curves $(\eta_i^-(\Delta), \eta_i^+(\Delta))$. The distance function $\delta_i^-(n)$ $[\delta_i^+(n)]$ indicates the minimum [maximum] time interval that contains n activation events. On the other hand, the arrival curve $\eta_i^+(\Delta)$ $[\eta_i^-(\Delta)]$ indicates the maximum [minimum] number of activation events that may occur in a time interval Δ. Each task τ_i is composed of n_i REs $\rho_{i,p}$ with $1 \leqslant p \leqslant n_i$. Due to the great industrial relevance of this configuration, we suppose that an RE $\rho_{i,p}$ inherits the deadline and the priority from the task τ_i to which it is mapped and with which it shares the same activation pattern. The worst-case execution time of an RE $\rho_{i,p}$ shall be denoted as $C_{i,p}$ and the sum of the worst-case execution times $C_{i,p}$ of all REs $\rho_{i,p}$ with $1 \leqslant p \leqslant n_i$ is clearly the worst-case execution time C_i of task τ_i : $C_i = \sum_{p=1}^{n_i} C_{i,p}$. Furthermore, an RE $\rho_{i,p}$ is either associated with a hard RT constraint, a weakly-hard RT constraint, or a best-effort constraint. A constraint of the form (m, k) with $k \in \mathbb{N}$ and $m = 0$ is considered as a hard RT constraint. A constraint of the form (m, k) with $0 < m < k$ for a given k is considered as a weakly-hard RT constraint. A best-effort constraint imposes no maximum on the tolerated number of deadline misses, i.e., for instance in a high-load situation, all deadlines may be missed.

To be able to compute a safe upper bound for (m, k) with respect to each RE $\rho_{i,p}$, we focus on a specific *workload scenario*. Assume that the task set $\mathcal{T}$ is *schedulable* under SPP in a typical worst case but that it is *not schedulable* in the worst case in which additional sporadic overload is present: We state, following [2], that a *worst-case arrival curve* $\eta_i^+(\Delta)$ of a task τ_i can be decomposed into two components: a *typical worst-case arrival curve* $\eta_{typ}^i(\Delta)$ and an *overload arrival curve* $\eta_{over}^i(\Delta)$. We say the *worst case* occurs if all tasks τ_i in $\mathcal{T}$ are activated with their worst-case arrival curves $\eta_i^+(\Delta)$ and consume their worst-case execution time C_i. The *typical worst case* occurs if all tasks τ_i in $\mathcal{T}$ are activated with their typical worst-case arrival curve $\eta_{typ}^i(\Delta)$ and consume their worst-case execution time C_i. Note that the decomposition of $\eta_i^+(\Delta)$ is arbitrary but with the restriction that in the typical worst case the system is schedulable.

V. Worst-Case Analysis for REs

The purpose of this Section is to provide means to decide whether for an RE a hard RT guarantee ($\forall k \in \mathbb{N} : \ m = 0$) can be given. Therefore, it is first necessary to be able to give worst-case response times for REs. The computation, while being an extension of the well-known *level-i busy window (BW)* approach [11], gives instructive insights about the RT behavior of REs inside tasks.

A. Worst-Case Response Time of REs

To compute the worst-case response time $WCRT_{i,p}$ for an RE $\rho_{i,p}$, the level-i BW approach, which has originally been proposed for tasks, is adapted. Literature concerned with RE-based systems, e.g. [7], has so far proposed worst-case response time computations for REs with the restriction of periodic activations and deadlines smaller than or equal to the period. We generalize this computation to our generic system model with *arbitrary activation patterns and deadlines*.

A level-i BW is the maximum time interval in which instances of tasks with priorities equal to or higher than Π_i ($hpe(i) = \{\tau_j \in \mathcal{T} | \Pi_j \leqslant \Pi_i\}$) are processed. At the beginning and at the end of the time interval the processing resource is idle with respect to instances of tasks with priority Π_i or higher. The *critical instant* for a level-i BW occurs when all tasks with higher or equal priority, scheduled under SPP, are activated at the same instant. The longest response time $WCRT_i$ for an instance of a task τ_i can be observed during the *longest level-i BW*. The longest level-i BW is initiated at the critical instant if just before the critical instant a lower priority task ($lp(i) = \{\tau_j \in \mathcal{T} | \Pi_j > \Pi_i\}$) enters a critical section and induces the maximum blocking time CS_i which is determined by the chosen shared resource access protocol. We define $B_i(q)$ $[B_{i,p}(q)]$ as the maximum time between the beginning of the longest level-i BW and the finishing time of the qth instance of task τ_i [the finishing time of the p-th RE $\rho_{i,p}$ in the qth instance of task τ_i].

Theorem 1. *The worst-case response time $WCRT_{i,p}$ of the p-th RE $\rho_{i,p}$ in task τ_i is*

$$WCRT_{i,p} = \max_{1 \leqslant q \leqslant K_i} \left\{ B_{i,p}(q) - \delta_i^-(q) \right\} \tag{1}$$

- $B_{i,p}(q)$ *is computed using the fixed point equation*

$$B_{i,p}^{(n)}(q) = CS_i + (q-1) \cdot C_i + \sum\nolimits_{\nu=1}^{p} c_{i\nu} + \sum\nolimits_{l \in hpe(i)} \eta_l^+ (B_{i,p}^{(n-1)}(q)) \cdot C_l \tag{2}$$

  with the initial value

$$B_{i,p}^{(0)}(q) = B_i(q-1) + \sum\nolimits_{\nu=1}^{p} c_{i\nu}. \tag{3}$$

- $\delta_i^-(q)$ *is the earliest moment of activation of the qth instance of task τ_i in the maximum level-i BW.*
- K_i *is the number of task instances in the maximum level-i BW:*

$$K_i = \min \left\{ q \geqslant 1 | B_i(q) < \delta_i^-(q+1) \right\} \tag{4}$$

Proof: The maximum level-i BW is the valid window in which not only $WCRT_i$ but also $WCRT_{i,p}$ can be found. This is due to the fact that all REs are activated synchronously with their container task so that with the above defined critical instant the maximum interference is provoked among tasks as well as among REs. Thus, $B_{i,p}(q)$ is composed of CS_i, the processing demand of task τ_i up to the pth RE of instance q, and the maximum processing demand of all $hpe(i)$-tasks until $B_{i,p}(q)$ has passed. The initial value considers that $B_{i,p}^{(0)}(q)$ cannot be smaller than the maximum time necessary to finish $q-1$ task instances and the execution time of all REs up to the p-th RE $\rho_{i,p}$ in the qth instance of task τ_i. Since the load development over time inside the maximum level-i BW is not known, a search over all response times in $B_{i,p}$ is necessary to spot the $WCRT_{i,p}$ of a RE $\rho_{i,p}$.

Corollary 1. *The instance q_i^* of task τ_i in the maximum level-i BW at which the worst-case response time $WCRT_i$ is observed may differ from the instance $q_{i,p}^*$ of task τ_i at which the worst-case response time $WCRT_{i,p}$ of the RE $\rho_{i,p}$ is found.*

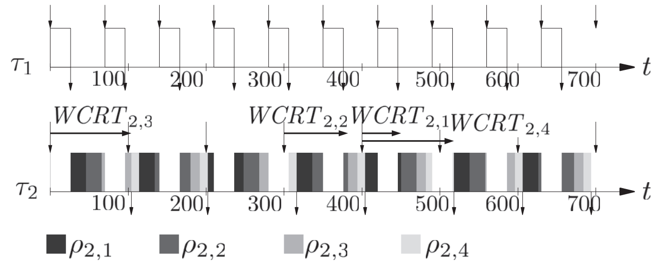

Fig. 2. Worst-Case Busy Window for Example 1

Proof: The distribution of interference within the instance of a task τ_i does not have an impact on its response time $RT_i(q)$, however, it does have an impact on the response times $RT_{i,p}(q)$ of the instances of its REs.

For illustration of Corollary 1 consider the example system below which is based on [11] with added REs:

Example 1. *We assume an SPP-scheduled system with two periodic tasks τ_1, τ_2 and denote the periods as T_1 resp. T_2. The higher priority task τ_1 consists of a single RE, the lower priority task τ_2 is composed of four REs:*

$$\begin{aligned} \tau_1 : &\quad C_1 = C_{1,1} = 26,\ T_1 = 70,\ \Pi_1 = 1, \\ \tau_2 : &\quad C_2 = 62,\ T_2 = 100,\ \Pi_2 = 2, \\ &\quad C_{2,1} = C_{2,2} = 20,\ C_{2,3} = 12,\ C_{2,4} = 10. \end{aligned}$$

Table I shows the response times $RT_{2,p}(q)$ for each RE $\rho_{2,p}$ in the worst-case BW. The timing diagram is depicted in Fig. 2. The task instances for which the $WCRT_{2,p}$ can be observed in the BW differ from the 1st to the 5th task instance.

	RT_{2p}	$WCRT_{2,p}$
ρ_{21}	$[46, 34, 48, 36, \mathbf{50}, 38, 26]$	50 ($q_{2,1}^* = 5$)
ρ_{22}	$[66, 80, 68, \mathbf{82}, 70, 58, 72]$	82 ($q_{2,2}^* = 4$)
ρ_{23}	$[\mathbf{104}, 92, 80, 94, 82, 96, 84]$	104 ($q_{2,3}^* = 1$)
ρ_{24}	$[114, 102, 116, 104, \mathbf{118}, 106, 94]$	118 ($q_{2,4}^* = 5$)

TABLE I. Response Time Results for Task τ_2

B. Schedulability of REs

A RE $\rho_{i,p}$ is schedulable under hard RT constraints if $WCRT_{i,p} \leqslant D_i$. In the considered worst case, a subset of the REs $\rho_{i,p}$ of task τ_i will be schedulable and the complementary subset of REs $\rho_{i,p}$ of task τ_i will not be schedulable.

Corollary 2. *If the RE ρ_{i,P_i} does not miss its deadline in the worst case, then all REs $\rho_{i,p}$ with $p < P_i$ do also not miss their deadline in the worst case. If the RE ρ_{i,P_i+1} misses its deadline in the worst case, then all REs $\rho_{i,p}$ with $p > P_i$ do also miss their deadline in the worst case.*

$$\forall i, p \leqslant P_i : WCRT_{i,p} \leqslant D_i \tag{5}$$
$$\forall i, p > P_i : WCRT_{i,p} > D_i \tag{6}$$

Proof: According to the system model, all REs $\rho_{i,p}$ in a task τ_i have an identical activation pattern and an identical deadline. It is then evident that $WCRT_{i,p_k} > WCRT_{ip_l}$ if $p_k > p_l$ and $WCRT_{i,p_k} < WCRT_{ip_l}$ if $p_k < p_l$. Thus $WCRT_{i,P_i} - D_i < 0$, $p < P_i \Rightarrow WCRT_{i,p} - D_i < 0$ and $WCRT_{i,P_i+1} - D_i > 0$, $p > P_i \Rightarrow WCRT_{i,p} - D_i > 0$.

From a design perspective, Corollary 2 implies that a task τ_i may include REs with hard and weakly-hard RT requirements. For a given RE-to-task mapping hard RT requirements are satisfiable up to ρ_{i,P_i}.

Example 2. *For illustration, we reconsider the above example and assume the deadline to be $D_2 = 0.95 \cdot T_2 = 95$. On the basis of Table I, we can derive: $P_2 = 2$.*

VI. Typical Worst-Case Analysis for REs

In the last section we have conducted a worst-case response time analysis for REs. On the basis of the results it could be decided whether a hard RT guarantee for an RE can be given or not. For every non-hard RT RE $\rho_{i,p}$, the below introduced *typical worst-case analysis for REs* (*TWCA-RE*) allows to specify (1) the response time of $\rho_{i,p}$ in the typical worst case $TWCRT_{i,p}$ (2) the number of deadline misses m that $\rho_{i,p}$ may experience in the worst case in any sequence of k consecutive instances where k is given. TWCA-RE extends TWCA presented in [2]–[4] which can so far only provide for a *task* τ_i (1) a $TWCRT_i$ and (2) an (m,k)-guarantee in the form of a deadline miss model $dmm_i(k)$ which bounds m from above for a given deadline D_i: $m \leqslant dmm_i(k)$. In contrast, TWCA-RE additionally provides an (m,k)-guarantee for every non-hard RT RE $\rho_{i,p}$ in the form of an upper bound $dmm_{i,p}(k)$ which can be defined by analogy with [3], [4]: A deadline miss model for RE $\rho_{i,p}$ of a task τ_i is a function $dmm_{i,p}(k) : \mathbb{N}^+ \rightarrow \mathbb{N}$ such that $dmm_{i,p}(k)$ bounds the number of executions of $\rho_{i,p}$ which may have a response time larger than D_i in a window of k consecutive executions of $\rho_{i,p}$.

A. Typical Worst-Case Reponse Time for an RE

The $TWCRT_{i,p}$ for an RE $\rho_{i,p}$ can be computed in analogy to the $WCRT_{i,p}$ according to Section V if instead of worst case the typical worst case is assumed. The $TWCRT_{i,p}$ is needed to verify that in the typical worst-case every RE $\rho_{i,p}$ meets its deadline and thus serves to confirm the correctness of the typical workload scenario.

B. Bounding Deadline Misses for an RE

In the following, we adapt the basic idea of TWCA [2]–[4], i.e. how to construct a deadline miss model, to the granular level of REs. We then improve the tightness of the computed "naive" $dmm_{i,p}(k)$ by exploiting the approach presented in [3] and tailoring it to the RE problem. Let $dmm^j_{i,p}(k)$ denote the maximum number of deadline misses that the RE $\rho_{i,p}$ may experience due to overload activations of a higher or equal priority task $\tau_j : j \in$ hpe(i). The maximum number of deadline misses $dmm^j_{i,p}(k)$ with respect to $\rho_{i,p}$ caused by the overload events of task τ_j is produced if task τ_i and all higher and equal priority tasks $\tau_{hpe(i)}$ require their worst-case processing demand. The upper bound $dmm_{i,p}(k)$ can safely be computed by summing up $dmm^j_{i,p}(k)$ for every task with higher or equal priority as task τ_i:

$$dmm_{i,p}(k) = \sum\nolimits_{j \in hpe(i)} dmm^j_{i,p}(k). \tag{7}$$

The computation of the bound $dmm^j_{i,p}(k)$ is based on the idea that there is a finite time interval $\Delta T_k^{j\rightarrow i,p}$ during which an overload activation of task τ_j can possibly have an impact on the response times of the considered k consecutive instances of the RE $\rho_{i,p}$. The maximum distance between the start of a BW and the termination of an RE $\rho_{i,p}$ is $B_{i,p}$ according to Theorem 1. Thus, an overload activation occurring more than $B_{i,p}$ before the first activation of the k-sequence of $\rho_{i,p}$ cannot be in the same BW as this first activation and therefore has no impact on its response time or that of subsequent activations. An overload activation that occurs during the k-sequence of activations may have an impact on the response times in the k-sequence. The longest duration of a k-sequence is $\delta_i^+(k)$. If $i = j$ then an overload activation after the last activation in the k-sequence has no impact on the response times in the k-sequence, because activations of a task are handled in a FIFO order. If $i \neq j$, the maximum interval of impact after the k-sequence is the maximum response time $WCRT_{i,p}$.

$$\Delta T_k^{j\rightarrow i,p} = \begin{cases} B_{i,p} + \delta_i^+(k) & \text{if } i = j \\ B_{i,p} + \delta_i^+(k) + WCRT_{i,p} & \text{if } i \neq j \end{cases} \tag{8}$$

From the worst-case analysis in Section V, the worst-case BW $B_{i,p}$ is known and the associated maximum number $N_{i,p}$ of deadline misses with respect to the RE $\rho_{i,p}$ can be derived

$$N_{i,p} = \#\left\{q \in \mathbb{N}^+ | 1 \leqslant q \leqslant K_i \wedge D_i < B_{i,p}(q) - \delta_i^-(q)\right\}. \tag{9}$$

The maximum number of overload activations of τ_j during the time interval $\Delta T_k^{j\rightarrow i,p}$ is given by $\eta^j_{over}(\Delta T_k^{j\rightarrow i,p})$ and every overload activation will at most cause $N_{i,p}$ deadline misses because it only has an impact during one BW

$$dmm^j_{i,p}(k) = N_{i,p} \cdot \eta^j_{over}(\Delta T_k^{j\rightarrow i,p}). \tag{10}$$

Note that for the last RE $\rho_{i,p}|_{p=n_i}$ in the task τ_i, Eq. 7, 8, 9, 10 reduce to the form indicated in [3] for the entire task τ_i. Thus $dmm_{i,p}(k)|_{p=n_i} = dmm_i(k)$, $dmm^j_{i,p}(k)|_{p=n_i} = dmm^j_i(k)$, $N_{i,p}|_{p=n_i} = N_i$ and $\Delta T_k^{j\rightarrow i,p}|_{p=n_i} = \Delta T_k^{j\rightarrow i}$.

Theorem 2. *The deadline miss model $dmm^j_{i,p}(k)$ of the non-hard RE $\rho_{i,p}$ with $p > P_i$ cannot be larger than the deadline miss model $dmm^j_i(k)$ of the task τ_i:*

$$\forall i, p > P_i : dmm^j_{i,p}(k) \leqslant dmm^j_i(k) \tag{11}$$

Proof: The Theorem follows from Eq. 10 combined with the fact that for $p_k > p_l \Rightarrow N_{ip_k} \geqslant N_{ip_l} \wedge \Delta T_k^{j\rightarrow i,p_k} \geqslant \Delta T_k^{j\rightarrow i,p_l}$ and $dmm^j_{i,p}(k)|_{p=n_i} = dmm^j_i(k)$.

Corollary 3. *The improvement of the bound $dmm^j_{i,p}(k)$ with respect to $dmm^j_i(k)$ for a given $p > P_i$ is dependent on the system configuration and the considered value of k:*

$$\begin{aligned} \Delta dmm^j_{i,p}(k) = dmm^j_i(k) - dmm^j_{i,p}(k) = \\ N_i \cdot \eta^j_{over}(\Delta T_k^{j\rightarrow i}) - N_{i,p} \cdot \eta^j_{over}(\Delta T_k^{j\rightarrow i,p}) \end{aligned} \tag{12}$$

For the improvement over all interfering overloaded tasks τ_j, we have

$$dmm_i(k) - dmm_{i,p}(k) = \sum_{j \in hpe(i)} \Delta dmm^j_{i,p}(k). \quad (13)$$

It is thus possible to guarantee a stricter (m,k)-bound for a non-hard RE $\rho_{i,p}$ with $P_i < p < n_i$ than for the task τ_i.

Large differences in value of $N_{i,p}$ and N_i can be observed for long BWs B_i (K_i large) because then the RE $\rho_{i,p}$ may miss its deadline in significantly less instances of task τ_i than the last RE ρ_{i,n_i} ($N_{i,n_i} = N_i$). Compare for illustration the number of deadlines misses observed for the RE $\rho_{2,3}$ (i.e. $N_{2,3} = 2$) and for the RE $\rho_{2,4}$ (i.e. $N_{2,4} = 6$) in Example 1 & 2. Differences in value of $\eta^j_{over}(\Delta T^{j\to i}_k)$ and $\eta^j_{over}(\Delta T^{j\to i,p}_k)$ occur when η^j_{over} is sensitive to small changes of arguments which happens for (1) high overload and (2) small absolute argument values which implies a small k.

Eq. 7-10 constitute the naive computation of $dmm_{i,p}(k)$. However, consider that, depending on the decomposition of the worst-case activation patterns of the task set $\mathcal{T}$ into typical-case patterns and overload patterns, the results for the deadline miss model $dmm_{i,p}(k)$ differ. Obviously, there will be a decomposition for the task set $\mathcal{T}$ which leads to a minimal $dmm_{i,p}(k)$. Since there are countably infinite possibilities to define the decomposition, we follow [3] and consider a restricted set of possible decompositions, which are denominated as combinations, in order to find a good (small) $dmm_{i,p}(k)$ for an RE $\rho_{i,p}$. A *combination* is defined as a tuple $\bar{c} = (c_1, c_2, \dots c_{|\mathcal{T}|})$ and indicates which activation pattern, $\eta^+_i(\Delta)$ ($c_i = \mathbf{W}$) or $\eta_{typ,i}(\Delta)$ ($c_i = \mathbf{T}$), should be considered as "typical" for every task in the task set $\mathcal{T}$ when computing $dmm_{i,p}(k)$. If $\eta^+_i(\Delta)$ is considered as typical then the overload pattern $\eta^i_{over}(\Delta)$ equals zero. If $\eta_{typ,i}(\Delta)$ is considered as typical then the overload pattern $\eta^i_{over}(\Delta)$ is different from zero.
In order to find a good deadline miss model $dmm_{i,p}(k)$ for an RE $\rho_{i,p}$, an *optimization problem* must be solved: The objective is to find for a given RE $\rho_{i,p}$ and a given k the combination $\bar{c}$ so that the combination-dependent $dmm^{\bar{c}}_{i,p}(k)$ is minimal. The objective function is derived in the following using (1) the definition of $dmm_{i,p}(k)$ (Eq. 7) and $dmm^j_{i,p}(k)$ (Eq. 10), and (2) the Boolean vector $\mathbf{b} = (b_1, b_2, \dots, b_{|\mathcal{T}|})$ which parametrizes the combination $\bar{c}$.

$$\begin{aligned} &\min dmm^{\bar{c}}_{i,p}(k) \qquad (14)\\ &= \min \sum_{j \in hpe(i)} N_{i,p} \cdot dmm^{j,\bar{c}}_{i,p}(k) \\ &= \min \sum_{j \in hpe(i)} N_{i,p} \cdot \eta^{j,\bar{c}}_{over}(\Delta T^{j\to i,p}_k) \\ &= N_{i,p} \cdot \min \sum_{j \in hpe(i)} b_j \cdot \eta^j_{over}(\Delta T^{j\to i,p}_k) \\ &= N_{i,p} \cdot \min \sum_{j \in hpe(i)} b_j \cdot \Omega^k_{j,i,p} \end{aligned}$$

The set of constraints, imposing that the RE $\rho_{i,p}$ must not miss its deadline in the typical worst-case, can be derived from the set of constraints given for the task problem in [3] by replacing the task-related variables with their RE-related counterparts:

$$0 \leqslant l \leqslant K_i : \quad \sum_{j \in hpe(i)} wl^{j,l}_{over} \cdot b_j \geqslant \Lambda^l_{i,p} - \Gamma^l_{i,p} \quad (15)$$

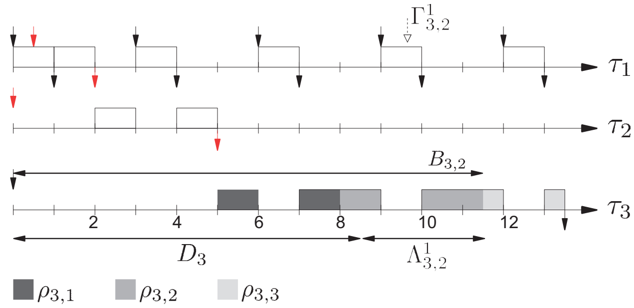

Fig. 3. Illustration of the Set of Constraints for RE $\rho_{3,2}$: The figure shows the worst-case level-3 BW ($K_3 = 1$) of a set of tasks $\{\tau_1, \tau_2, \tau_3\}$. The black arrows indicate the typical activation [completion] patterns whereas the red arrows indicate the overload activation [completion] patterns. The expression $\Lambda^1_{3,2} = 3$ indicates the amount of workload that needs to be removed in order to avoid the deadline miss of the RE $\rho_{3,2}$. In fact, some of this workload disappears by itself if the deadline is enforced, namely the workload resulting from activations taking place after D_3 and before $B_{3,2}$ i.e. $\Gamma^1_{3,2} = 1$. The constraint $\sum_{j \in hpe(3)} wl^{j,1}_{over} \cdot b_j \geqslant \Lambda^1_{3,2} - \Gamma^1_{3,2}$ therefore formulates the sufficient condition that if the overload-induced workload is larger or equal than the amount of extra workload that is necessary to provoke a deadline miss of RE $\rho_{3,2}$, then in the typical worst-case the RE $\rho_{3,2}$ does not miss its deadline.

with l indicating the considered task instance $\tau_i(l)$ in the worst-case BW and

$$wl^{j,l}_{over} = \begin{cases} \eta^j_{over}(D_i + \delta^-_i(l)) \cdot C_j & \text{for } j \in hp(i) \\ \eta^i_{over}(\delta^-_i(l)) \cdot C_i & \text{for } j = i \end{cases} \quad (16)$$

$$\Lambda^l_{i,p} = \quad B_{i,p}(l) - \delta^-_i(l) - D_i \quad (17)$$

$$\Gamma^l_{i,p} = \begin{cases} \sum\limits_{j \in hp(i)} \left[\eta^+_j(B_{i,p}(l)) - \eta^+_j(D_i + \delta^-_i(l))\right] \cdot C_j & \text{for } j \in hp(i) \\ 0 \quad \text{for } j = i. \end{cases} \quad (18)$$

For further explanation of the set of constraints specified in Eq. 15-18 refer to Figure 3.

The optimization may impact the improvement of the bound $dmm_{i,p}(k)$ with respect to $dmm_i(k)$: Since it tightens the bounds $dmm_{i,p}(k)$ and $dmm_i(k)$, the difference $dmm_i(k) - dmm_{i,p}(k)$ decreases, remains constant or increases compared to Corollary 3 depending on the overestimation of each bound in the naive approach. By nature of the system, $dmm_i(k)$ remains a safe upper bound for $dmm_{i,p}(k)$.

VII. Methodology

In this Section, it is briefly explained how to best apply the presented analysis approaches (RTA, RTA-RE, TWCA, TWCA-RE) for a given system configuration:
(**Step 1**) Use the classical RTA to verify if the system is schedulable: Yes $\to$ Feasible System. No $\to$ Go to (Step 2). (**Step 2**) Use RTA-RE for every task τ_i which has been found unschedulable in (Step 1). Check if the system requirements are satisfied with respect to the REs $\rho_{i,p}$ which require hard RT guarantees: Yes $\to$ Go to (Step 3). No $\to$ Infeasible System. (**Step 3**) Use TWCA to compute the $dmm_i(k)$ for every task τ_i which has been found unschedulable in (Step 1). Check if the system requirements are satisfied with respect to the REs $\rho_{i,p}$ which require weakly-hard RT guarantees: Yes $\to$ Feasible System. No $\to$ Go to (Step 4). (**Step 4**) Use TWCA-RE to

compute the $dmm_{i,p}(k)$ for every RE $\rho_{i,p}$ whose weakly-hard RT requirements have not been fulfilled in (Step 3) and check whether the tighter individual bound can satisfy the requirement. Yes $\rightarrow$ Feasible System. No $\rightarrow$ Infeasible System.

VIII. Evaluation

For the evaluation of our analysis we consider an authentic industrial example from the automotive domain, kindly provided by Bosch. The software system consists of a set of 21 tasks $\mathcal{T} = \{\tau_1, \ldots, \tau_{21}\}$ and is scheduled under an SPP-algorithm. Each task is characterized by the tuple $(C_i, \Pi_i, T_i, D_i, n_i)$ where T_i is the period. Task τ_1 has the highest priority while τ_{21} has the lowest priority. Every task is composed of a given number n_i of REs, n_i varies between 2 and 732. The system is given in its typical configuration i.e. no deadline misses can be observed in the worst case. This is a common set-up for real systems where tasks may include hard RT REs. We want to show that TWCA-RE can be used to analyze the schedulability of the given system under sporadic overload by identifying hard RT REs, which no other RT verification method can do in the overloaded case, and by giving weakly-hard bounds for non-hard RT REs, which often improve compared to TWCA results. Since a typical system model is given, we complement the given experimental set-up by synthetic sporadic overload i.e. a subset of tasks $\{\tau_{11}, \tau_{15}, \tau_{16}\}$ is additionally activated by rare events which may be classified as non-typical. The sporadic overload is chosen such that in the worst case task τ_{17} ($C_{17} = 6.302ms, \Pi_{17} = 17, T_{17} = 20ms, D_i = 18ms,\ n_i = 732$) misses its deadline. Task τ_{17} is the task in the task set $\mathcal{T}$ with the largest number of REs and thus of particular interest to evaluate the presented analysis. Note that due to the given typical model no upper weakly-hard bounds for tasks and REs are available as requirements.

In **Scenario A**, we add a lightweight synthetic sporadic overload, meaning that tasks $\{\tau_{11}, \tau_{15}, \tau_{16}\}$ only rarely experience non-typical activations ($WCRT_{17} = 25.618ms$, $TWCRT_{17} = 14.414ms$). The results are shown in Table II. In contrast to TWCA, TWCA-RE can provide *hard* RT guarantees for the vast majority of REs of task τ_{17}, namely for $\{\rho_{17,p} | p \in \{0 \ldots 612\}\}$. For every non-hard RT RE $\{\rho_{17,p} | p \in \{613 \ldots 731\}\}$ TWCA-RE provides the same $dmm_{17,p}(k)$ for $k = 9, 50, 100$ as TWCA would do for the entire task ($dmm_{17}(k)$). This is due to the fact that the worst-case BW includes only one instance and, consequently, no difference in N_{ip} and N_i is possible. Furthermore, the lightweight overload arrival curves η^{11}_{over}, η^{15}_{over}, η^{16}_{over} are not sensitive to small changes of arguments even under small k.

In **Scenario B**, the synthetic sporadic overload is increased for the tasks $\{\tau_{11}, \tau_{15}, \tau_{16}\}$ ($WCRT_{17} = 29.521ms$, $TWCRT_{17} = 14.414ms$). The results are shown in Table II. The additional load causes that (1) the worst-case BW includes now 2 task instances instead of 1 and (2) the overload arrival curves η^{11}_{over}, η^{15}_{over}, η^{16}_{over} are now sensitive to small changes of arguments at small k. Thus, TWCA-RE can not only give hard RT guarantees for about half of the REs of task τ_{17}, namely here $\{\rho_{17,p} | p \in \{0 \ldots 351\}\}$, but also give significantly better weakly-hard RT guarantees for the individual REs than TWCA is capable to do.

We have performed the experiments on an Intel Core i5-3230M CPU @ 2.60GHz $\times$ 4 with 8 GB memory with a run time of $19.3s$ for Scenario A and $38.2s$ for Scenario B.

Scenario A		
$dmm_{17,p}(9)$	$dmm_{17,p}(50)$	$dmm_{17,p}(100)$
<u>2</u> ($613 \leqslant p \leqslant 731$)	**<u>2</u>** ($613 \leqslant p \leqslant 731$)	**<u>2</u>** ($613 \leqslant p \leqslant 731$)
Scenario B		
$dmm_{17,p}(9)$	$dmm_{17,p}(50)$	$dmm_{17,p}(100)$
3 ($352 \leqslant p \leqslant 391$) **4** ($392 \leqslant p \leqslant 473$) **<u>8</u>** ($474 \leqslant p \leqslant 731$)	**8** ($352 \leqslant p \leqslant 473$) **<u>16</u>** ($474 \leqslant p \leqslant 731$)	**10** ($352 \leqslant p \leqslant 473$) **<u>20</u>** ($474 \leqslant p \leqslant 731$)

TABLE II. DMMs for the REs of Task τ_{17}
The underlined values correspond to $dmm_{17}(k)$ for the entire task.

IX. Conclusion

In this paper, we extended RTA and TWCA such that RT guarantees for REs can be computed for a given RE-to-task-mapping which preserves all constraints of the functional model. Though existing theories are adopted, we are able to contribute analysis results and insights into system behavior which lead to new and interesting design aspects with respect to RE-based systems as well as to improved system utilization.

Acknowledgment

The authors would like to thank A. Hamann and D. Ziegenbein from Robert Bosch GmbH for providing the data set for our case study.

References

[1] *AUTOSAR Standard, Release 4.2*, http://www.autosar.org, AUTOSAR Consortium.

[2] S. Quinton, M. Negrean, and R. Ernst, “Formal analysis of sporadic bursts in real-time systems,” in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2013, pp. 767–772.

[3] Z. Hammadeh, S. Quinton, and R. Ernst, “Extending typical worst-case analysis using response-time dependencies to bound deadline misses,” in *Embedded Software (EMSOFT), 2014 International Conference on*, New York, 2014, pp. 1–10.

[4] W. Xu *et al.*, “Improved deadline miss models for real-time systems using typical worst-Case analysis,” in *27th Euromicro Conference on Real-Time Systems (ECRTS)*, Lund, 2015.

[5] M. Hamdaoui and P. Ramanathan, “A dynamic priority assignment technique for streams with (m, k)-firm deadlines,” *Computers, IEEE Transactions on*, vol. 44, no. 12, pp. 1443–1451, 1995.

[6] G. Bernat, A. Burns, and A. Liamosi, “Weakly Hard Real-Time Systems,” *Computers, IEEE Transactions on*, vol. 50, no. 4, pp. 308–321, 2001.

[7] A. Ferrari *et al.*, “Time and memory tradeoffs in the implementation of AUTOSAR components,” in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2009, pp. 864–869.

[8] H. Zeng and M. Di Natale, “Mechanisms for guaranteeing data consistency and flow preservation in AUTOSAR software on multi-core platforms,” in *6th IEEE International Symposium on Industrial Embedded Systems (SIES)*, 2011, pp. 140–149.

[9] H. Zeng and M. Di Natale, “Efficient implementation of AUTOSAR components with minimal memory usage,” in *7th IEEE International Symposium on Industrial Embedded Systems (SIES)*, 2012, pp. 130–137.

[10] A. Monot *et al.*, “Multisource software on multicore Automotive ECUs - combining runnable sequencing with task scheduling,” *IEEE Transactions on Industrial Electronics*, vol. 59, no. 10, pp. 3934–3942, 2012.

[11] J. P. Lehoczky, “Fixed priority scheduling of periodic task sets with arbitrary deadlines,” in *Proceedings of the 11th Real-Time Systems Symposium*, 1990, pp. 201–209.