Finite Ready Queues As a Mean for Overload Reduction in Weakly-Hard Real-Time Systems

Sophie Quinton, Leonie Ahrendts, Rolf Ernst

Finite Ready Queues As a Mean for Overload Reduction in Weakly-Hard Real-Time Systems

Leonie Ahrendts
Institute of Computer and Network Engineering, TU Braunschweig
Braunschweig, Germany
ahrendts@ida.ing.tu-bs.de

Sophie Quinton
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG
F-38000 Grenoble, France
sophie.quinton@inria.fr

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

ABSTRACT

Finite ready queues, implemented by buffers, are a system reality in embedded real-time computing systems and networks. The dimensioning of queues is subject to constraints in industrial practice, and often the queue capacity is sufficient for typical system behavior but is not sufficient in peak overload conditions. This may lead to overflow and consequently to the discarding of jobs. In this paper, we explore whether finite queue capacity can also be used as a mean of design in order to reduce workload peaks and thus shorten a transient overload phase. We present an analysis method which is to the best of our knowledge the first one able to give (a) worst-case response times guarantees as well as (b) weakly-hard guarantees for tasks which are executed on a computing system with finite queues. Experimental results show that finite queue capacity may only a have weak overload limiting effect. This unexpected outcome can be explained by the system behavior in the worst-case corner cases. The analysis shows nevertheless that a trade-off between weakly-hard guarantees and queue sizes is possible.

CCS CONCEPTS

• **Computer systems organization → Embedded and cyber-physical systems**; **Real-time systems**;

KEYWORDS

Real-Time Systems, Weakly-Hard Real-Time Guarantees, Limited Queue Capacity

ACM Reference format:
Leonie Ahrendts, Sophie Quinton, and Rolf Ernst. 2017. Finite Ready Queues As a Mean for Overload Reduction in Weakly-Hard Real-Time Systems. In *Proceedings of RTNS '17, Grenoble, France, October 4–6, 2017,* 10 pages.
DOI: 10.1145/3139258.3139259

1 INTRODUCTION

Real-time computing systems are said to be overloaded if not all tasks can be completed within their deadlines. Overload is often due to an unusually high number of additional sporadic jobs which appear as a reaction to a scarce environmental condition. Interestingly, it has been observed that many embedded applications, previously classified as having hard real-time requirements, actually tolerate occasional deadline misses. For instance, the performance of an imaging application is still acceptable if once in a while a frame is lost or repeated [2]. Likewise the quality of a control application is still satisfactory if once in a while a sensor data sample is lost and the control algorithm is performed on an old sensor data sample [10]. Results from control engineering [4] could even show Lyapunov stability of the feedback control of an unstable plant that runs in open loop from time to time due to failures. The modern concept of weakly-hard real-time (WHRT) computing systems [2] takes this reality into account. The timing guarantees for a WHRT computing system are formulated for every task in the task set in the following manner: A task is guaranteed not to miss more than m deadlines in any sequence of k consecutive executions even in the worst case. Note that this precise statement of the worst-case distribution pattern of deadline misses distinguishes weakly-hard from soft real-time computing systems.

The objective of our work is to explore the timing impact of finite queues. Our hypothesis is that finite queues can serve as a mean to effectively limit overload peaks. According to our model, if a ready queue of a task is full, any new incoming job is rejected while accepted jobs are always run to completion. The disburdening of the system by rejecting jobs in overload conditions leads to an improved schedulability of accepted jobs. Ideally, the rejection mechanism frees the system from jobs, which would otherwise anyhow miss their deadline due to long waiting times in the queue. If finite queue capacity succeeds in the efficient limitation of overload peaks, then the number of jobs which become schedulable under reduced service demand is larger than the number of rejected jobs. Consequently, WHRT guarantees can be improved, and that at significantly smaller queue sizes than required for overflow-free behavior.

Beyond this very desirable property, the virtue of such a workload limiting mechanism would be that it can be realized at a minor implementation impact: Discarding of workload is implicitly realized by queue overflow, no monitoring or task abortion is required. The original scheduling algorithm can still be employed, and finite queues are already system reality: In a context where processing resources are micro-controllers or similar, finite ready queues restrict the number of jobs that can be activated at a time. For instance the size of a ready queue may be specified by the real-time operating system (RTOS) standard, as in the case of the automotive OSEK RTOS standard which allows for a task of the basic performance class only one activated job at a time. In a network context, finite queues result from limited storage capacity for packets in switches.

The analysis method presented in this paper is to the best of our knowledge the first one which is able to give (a) worst-case

RTNS '17, Grenoble, France

DOI: 10.1145/3139258.3139259

response times guarantees as well as (b) WHRT guarantee for each task which is executed on a computing system with finite queues. By WHRT guarantee for a task, we understand in this work an upper bound m on the number of *generalized* deadline misses in any consecutive sequence of k executions of this task. We define a generalized deadline miss as either a computational result produced by a task after its given deadline, or a missing computational result which should be produced by the task but is not due to the rejection of a job. Note here the difference to current research, where only conventional deadline misses have been quantified by WHRT guarantees due to the assumption of infinite queues.

Our approach is based on typical worst-case analysis (TWCA) [21], a verification method for WHRT systems with a single processing resource and static priority scheduling. TWCA can principally be included in a compositional performance analysis framework to cover multi-resource processing platforms. TWCA assumes that in a typical workload scenario, where all tasks are activated according to their typical activation pattern (e.g. periodic pattern), no deadline misses occur. Only in the rare non-typical workload scenarios, where a subset of tasks has additional overload (e.g. sporadic) activations, are some deadlines missed and TWCA provides WHRT guarantees. In case of finite queues, previous approaches to compute the required TWCA input parameters are not sufficient. In particular, we show in this paper how to derive the accepted workload over time by using techniques from operations research.

At this point we would like to provide an outlook on the experimental results. Surprisingly, the obtained worst-case response times and WHRT guarantees did not systematically profit from finite queue sizes. As will be explained in detail in the evaluation section, this is primarily caused by the characteristics of the worst-case system behavior under which the upper timing bounds are derived. The experiments give a novel insight into system behavior under finite queues, and show that – if not a systematic improvement of worst-case WHRT guarantees – nevertheless a trade-off between obtainable WHRT guarantees and queue sizes is possible.

In the following, we first give an overview of related work followed by a detailed description of the system model. Then we successively determine the required TWCA input variables starting with workload arrival curves and then moving to busy period computations including the derivation of worst-case response times. Subsequently we apply TWCA and evaluate the behavior of systems with finite queue capacity in overload conditions.

2 RELATED WORK

The handling of overload conditions in real-time systems has been a research topic since the 1990s. Both scheduling algorithms and scheduling analysis have focused on this. Baruah et al. [1] and Buttazzo et al. [5] introduced value-based performance metrics for evaluating the behavior of scheduling algorithms in overload conditions. Their system model assumes that each job contributes at its completion a "value" to the system. Late completion generally reduces significantly the value of a job. The objective of value-based scheduling is to maximize the cumulative value contributed by n jobs. Value constraints assure that it is prohibitively expensive to complete hard real-time jobs after their deadline. Thus, hard real-time jobs are typically scheduled at the expense of firm and soft real-time tasks using appropriate priority assignments as well as on-line or off-line acceptance schemes [5, 6, 12]. A related line of research (e.g. [9]) derives bounds on the tardiness of tasks, where tardiness is defined as the difference between completion time and deadline of a task. In the context of soft real-time tasks bounded tardiness is acceptable as long as in the long term each task may utilize the processor as specified.

Based on network calculus, a few works analyze systems with losses. The computed guarantees are upper bounds on the packet loss rates in the presence of traffic clippers, which are traffic shapers with zero buffering capacity [8]. The authors of [7] derive an optimal traffic regulator with constraints on maximum tolerable delay and maximum buffer size, which generates output traffic conforming to a subadditive traffic envelope and minimizes the number of discarded packets.

The concept of WHRT systems [2] allows to apply a different scheduling approach in overload conditions. WHRT jobs tolerate by definition a bounded number of deadline misses, where the tolerance is specified by (m, k)-constraints. Consequently scheduling algorithms have been proposed which attempt to guarantee or guarantee by design (m, k)-constraints of weakly-hard tasks and apply best effort for soft real-time tasks e.g. [11] [3]. TWCA [21] verifies (m, k)-constraints for static priority scheduling on single processing resources.

In this paper, we concentrate on the analysis of WHRT systems using a fixed priority scheduling with the special property of finite queues. The timing analysis for such a setting is more complex than in previous works because (1) the (m, k)-guarantees are not inherent in the design of the scheduling algorithm and (2) the limited queue capacity represents a nonlinearity in system behavior which has not been treated so far. However, the system model is realistic and results are relevant for the behavior of industrial real-time systems in overload conditions. We compute both maximum response times and frequency of deadline misses for each task in the system.

3 SYSTEM MODEL

A system consists of one processing resource which schedules a task set $\mathcal{T}$ according to the static priority preemptive (SPP) policy. The task set $\mathcal{T} = \left\{\tau_j | 1 \leq j \leq N\right\}$ consists of N independent tasks. Each task τ_j is characterized by its static priority π_j, its minimum and maximum workload-based arrival curve $\left[\alpha_i^{-,\infty}(\Delta t), \alpha_i^{+,\infty}(\Delta t)\right]$ cf. Sec. 3.1, the lower and upper bound of its execution time $\left[C_j^-, C_j^+\right]$, and its relative deadline d_i. The priority π_j of each task $\tau_j \in \mathcal{T}$ corresponds to its index j and is therefore unique. Smaller priority indices mean higher priority. The task set $\mathcal{T}_i = \left\{\tau_j | 1 \leq j \leq i\right\}$ includes the i tasks with the i highest priorities in the task set $\mathcal{T}$. The system is said to have *finite queue capacity* if for every task $\tau_j \in \mathcal{T}$ the maximum accepted amount of pending workload is limited to $b_j = s_j \cdot C_j^+$ due to the finite number of slots s_j in the queue. The system is said to have *infinite queue capacity* if for every task $\tau_j \in \mathcal{T}$ the accepted amount of pending workload is not limited. The workload of the qth job $\tau_i(q)$ is said to be *accepted* by the system if a free queue slot is available. Otherwise, the job $\tau_j(q)$ is discarded and the workload of $\tau_j(q)$ is said to have been *rejected* by the system.

3.1 Workload-based arrival curves

We use workload-based arrival curves to describe the workload imposed by a task on the system. In contrast to previous work, our definition of the workload-based arrival curves takes the queue capacity of the system for pending workload into account and only considers accepted workload as incurred workload.

Definition 3.1. The workload-based arrival curve $\alpha_i[s,t)$ of a task τ_i with $s \leq t$ and $s,t \in \mathbb{R}$ describes the accepted workload of τ_i between s and t taking into account the queue capacity of the system and is bounded by

$$\alpha_i^-(\Delta t) \leq \alpha_i[s,t) \leq \alpha_i^+(\Delta t)$$

where $\alpha_i^-(\Delta t)$ and $\alpha_i^+(\Delta t)$ provide a lower and an upper bound on $\alpha_i[s,t)$ for any s and t such that $t - s = \Delta t$.

The concept of quantifying the accepted workload of a task τ_i can be adapted to a set of tasks $\mathcal{T}_i$.

Definition 3.2. A cumulative workload-based arrival curve $A_i[s,t)$ models the accepted workload incurred by the task set $\mathcal{T}_i$ in the time interval $[s,t)$ with $s \leq t$ and $s,t \in \mathbb{R}$ taking into account the queue capacity of the system. The cumulative workload-based arrival curve $A_i[s,t)$ is bounded by

$$A_i^-(\Delta t) \leq A_i[s,t) \leq A_i^+(\Delta t).$$

In the following, whenever the queue capacity is assumed to be infinite, we add the superscript ∞ to the corresponding (cumulative) workload-based arrival curves.

It is important to realize that for the case of *finite* queue capacity, the variables $\alpha_i[s,t)$ and $A_i[s,t) = \sum_{j=1}^{i} \alpha_i[s,t)$ as well as their lower and upper bounds are not known. There is some computational effort required to derive them from the information provided by the system model cf. Section 4. This is a contrast to $\alpha_i^\infty(\Delta t)$ and $A_i^\infty(\Delta t) = \sum_{j=1}^{i} \alpha_i^\infty(\Delta t)$, which are usually either given or can be easily derived.

3.2 Service curve, remaining service, and processed, pending workload

While in the previous subsection the incurred workload of tasks has been described, this subsection introduces concepts which characterize the processing behavior of the system.

Definition 3.3. The *service curve* $\beta[s,t) := t - s$, or equivalently $\beta(\Delta t) := \Delta t$, indicates the available processing time in any time interval Δt of length $t - s$ with $s \leq t$ and $s,t \in \mathbb{R}$.

Definition 3.4. The *remaining service for task* τ_i, denoted as $\gamma_i[s,t)$, is the amount of time in $[s,t)$ with $s \leq t$ and $s,t \in \mathbb{R}$ during which the processor is not used by higher priority tasks such that its service can be assigned to task τ_i.

Definition 3.5. The *processed workload of task* τ_i, denoted as $\text{proc}_i[s,t)$, corresponds to the amount of time that task τ_i occupies the processor in the time interval $[s,t)$ with $s \leq t$ and $s,t \in \mathbb{R}$.

Definition 3.6. The *pending workload of task* τ_i denoted as $\text{pend}_i(t)$ is the amount of accepted workload that has been requested to be processed by a task τ_i but has not yet been processed at instant t taking into account the queue capacity of the system.

The definition of the processed [pending] workload of a task τ_i can be adapted to a set of tasks $\mathcal{T}_i$ analogous to Def. 3.2. The cumulative processed [pending] workload of the task set $\mathcal{T}_i$ is then denoted as $\text{Proc}_i[s,t)$ $[\text{Pend}_i(t)]$.

Again we add the superscript ∞ for variables corresponding to the infinite queue capacity assumption. Lower, resp. upper, bounds are additionally indexed with the superscript $-$, resp. $+$.

4 COMPUTATION OF WORKLOAD BOUNDS UNDER FINITE QUEUES

4.1 Motivational Example

In the well-known case of infinite ready queues, when workload is always accepted, the maximum cumulative workload-based arrival curve $A_i^{\infty,+}(\Delta t)$ is computed by the superposition of the maximum workload-based arrival curves $\alpha_j^{\infty,+}(\Delta t)$ of each task τ_j in the task set $\mathcal{T}_i$

$$A_i^{\infty,+}(\Delta t) = \sum_{j=1}^{i} \alpha_j^{\infty,+}(\Delta t). \tag{1}$$

In the case of finite ready queues, the accepted workload of each task $\tau_j \in \mathcal{T}_i$ depends not only on the processing request of task τ_j itself but also on the demand of tasks with higher priority. All three factors determine the queue filling of the queue of task τ_j. Therefore $A_i^+(\Delta t)$ cannot be computed by simple superposition of independent $\alpha_j^{+,\infty}(\Delta t)$ as in the case of infinite queue capacity. Let us illustrate this important property with an example.

Example 4.1. We assume an SPP-scheduled resource. The task set $\mathcal{T} = \{\tau_1, \tau_2, \tau_3, \tau_4\}$ is schedulable if no sporadic overload activations are present. The queues are so dimensioned that if no overload is present, all task instances will be accepted.

priorities	$\pi_1 = 1,\ \pi_2 = 2,\ \pi_3 = 3,\ \pi_4 = 4$
periods	$T_1 = T_2 = 4,\ T_3 = 8,\ T_4 = 16$
deadlines	$d_1 = 4,\ d_2 = 8,\ d_3 = 8,\ d_4 = 16$
WCETs	$C_1^+ = 1.5, C_2^+ = 1, C_3^+ = 2, C_4^+ = 0.5$
queues	$b_1 = 2C_1^+,\ b_2 = C_2^+,\ b_3 = C_3^+,\ b_4 = C_4^+$

Figure 1a illustrates the classical worst case with infinite ready queues. The overload is represented by one additional sporadic overload activation of task τ_1 (marked in red). The resource demand of each task τ_j behaves according to $\alpha_j^{\infty,+}(\Delta t) = C_j^+ \cdot \lceil \Delta t / T_j \rceil$. As a consequence of overload, task instance $\tau_3(1)$ misses its deadline. Figure 1b shows the same system behavior in the case of finite ready queues. Contrary to the previous situation, only a part of the resource demand is accepted w.r.t. some tasks. Task instance $\tau_3(2)$ is rejected because the queue is still blocked by $\tau_3(1)$. In Figure 1c, the situation is slightly different. The difference to the scenario in Figure 1b is that task instance $\tau_2(2)$ only demands half of its WCET. Consequently task τ_3 becomes schedulable. Figure 1d compares the cumulative accepted workload of the task set $\mathcal{T}$ over the time interval $[0,16)$. As can be seen, at a given point in time either scenario 2 or scenario 3 or both scenarios produce the larger cumulative accepted workload demand $A_4[0,t)$. For the time interval $[0,8)$, scenario 2 produces the absolute worst case since $A_4[0,t) = A_4^{\infty,+}(\Delta t)$. Figure 1c shows that the workload curves of the two scenarios overtake each other and that at different points

in time a different scenario represents the worst case w.r.t. the cumulative workload-based arrival curves.

The above example illustrates the fact that it is in general *not* possible to derive, for a given task set, a single scenario which produces the maximum cumulative load-based arrival curve $A_i(\Delta t)$ for all Δt such that we could define $A_i^+(\Delta t) = A_i(\Delta t)$. We must therefore find an envelope over all possible cumulative workload-based arrival curves so that $A_i^+(\Delta t)$ is an upper bound. Of course, $A_i^{+,\infty}(\Delta t)$ is a possible solution yet it is not tight and we want to improve over it.

4.2 Computing $A_i^+(\Delta t)$ for a given interval

In this section we present an inductive calculation rule to bound from above the cumulative workload-based arrival curve $A_i[s,t)$ for an arbitrary time interval $[s,t)$ of length $\Delta t = t - s$ where $A_{i-1}[s,t)$ is assumed to be known. First remember that by definition

$$A_i[s,t) = A_{i-1}[s,t) + \alpha_i[s,t). \tag{2}$$

The so far unknown accepted resource demand $\alpha_i[s,t)$ can be derived as follows. The balance equation

$$\text{pend}_i(t) = \text{pend}_i(s) + \alpha_i[s,t) - \text{proc}_i[s,t). \tag{3}$$

expresses that the workload of task τ_i that is pending at t is the sum of the workload pending at s and the workload incoming during the time interval $[s,t)$ minus the workload processed within $[s,t)$. Eq. 3 can now be used to expand the calculation rule given in Eq. 2.

$$A_i[s,t) = A_{i-1}[s,t) + \text{proc}_i[s,t) + \text{pend}_i(t) - \text{pend}_i(s) \tag{4}$$

Problem 1. *Our objective is to maximize*

$$A_{i-1}[s,t) + \text{proc}_i[s,t) + \text{pend}_i(t) - \text{pend}_i(s) \tag{5}$$

The maximal value of $A_i[s,t)$ thus obtained can then be used to define $A_i^+(\Delta t)$ where $\Delta t = t - s$, i.e. the maximum accepted cumulative workload demand in any time interval of length Δt.

The constraints defined in Eq. 6, 7, 8, 10, and 14 restrict the solution.

In the following, we detail the above mentioned constraints.

C1 Incoming workload. The cumulative workload-based arrival curve $A_{i-1}[s,t)$ in $[s,t)$ is bounded by

$$A_{i-1}^-(t-s) \le A_{i-1}[s,t) \le A_{i-1}^+(t-s). \tag{6}$$

Also, we know that $\alpha_i[s,t) \le \alpha_i^{\infty,+}(t-s)$, which corresponds after expansion of $\alpha_i[s,t)$ to

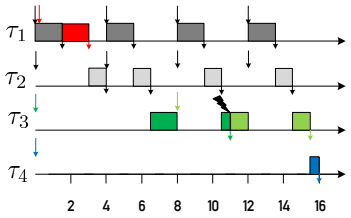
$$\text{proc}_i[s,t) + \text{pend}_i(t) - \text{pend}_i(s) \le \alpha_i^{\infty,+}(t-s). \tag{7}$$

Note that with the given constraints on incoming workload only, Problem 1 could be solved yet this would result in the trivial solution $A_{i-1}[s,t) = A_{i-1}^+(t-s)$.

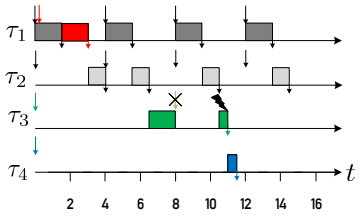
C2 Pending workload. The pending workload is for any instant restricted by the finite queue capacity of task τ_i. For the starting value $\text{pend}_i(s)$, we have

$$0 \le \text{pend}_i(s) \le b_i. \tag{8}$$

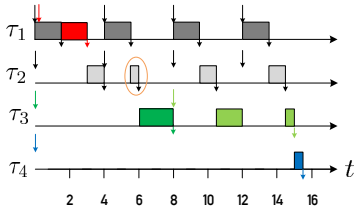
For every subsequent instant $s < t^* \le t$, the pending workload $\text{pend}_i(t^*)$ follows directly from the initial queue filling $\text{pend}_i(s)$ and the defined system behavior. Suppose that at $t^* - \epsilon$ an amount $\tilde{s}_i$ of the queue slots s_i is occupied, and immediately afterwards at t^* a number n of task instances $\tau(q_1), \tau(q_2), \ldots, \tau(q_n)$ with execution demand $C_i(q)$ are activated. The system will admit as many of the



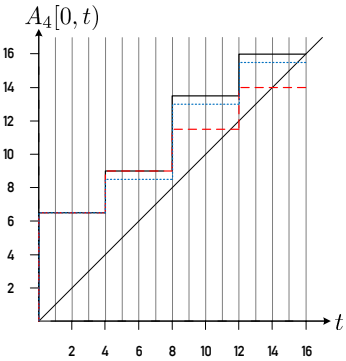
(a) Scenario 1: all task instances are activated simultaneously and demand their WCET; infinite queues



(b) Scenario 2: all task instances are activated simultaneously and demand their WCET; finite queues



(c) Scenario 3: all task instances are activated simultaneously and demand their WCET except $\tau_2(2)$; finite queues



(d) Comparison of the cumulative workload-based arrival curves $A_4(t)$ of Sc. 1 (solid black line), Sc. 2 (dashed red line), and Sc. 3 (dotted blue line). The bisector indicates the service curve.

Figure 1: Challenges in determining a maximum cumulative arrival curve. *A crossed-out activation arrow illustrates the discarding of a task instance, the lightning symbol indicates a deadline miss.*

n ordered task instances as free queue space is available

$$\text{pend}_i(t^*) = \begin{cases} \text{pend}_i(t^* - \epsilon) & \text{if } n = 0 \vee \tilde{s}_i = s_i \\ \text{pend}_i(t^* - \epsilon) + \sum_{q=q_1}^{q_1+M-1} C_i(q) & \text{else.} \end{cases} \quad (9)$$

where $M = \max\{m \mid m \leq n \wedge m \leq s_i - \tilde{s}_i\}$, i.e. the largest number of task instances that fit into the queue.

Note that the constraint for pending workload in Eq. 9 is nonlinear and has to be verified for every t^* with $s < t^* \leq t$. Therefore, we relax the constraint in Eq. 9 into

$$0 \leq \text{pend}_i(t) \leq b_i. \quad (10)$$

such that with Eq. 8 and 10 the limited queue size is enforced at the beginning and end of the time interval $[s, t)$. Applying the the queue constraint only twice instead at each activation of task τ_i can only reduce the amount of rejected workload. Furthermore, a continuous kind of workload clipping is modeled which ignores the slot organization of the queue. Continuous workload clipping always rejects less or equal workload than the discarding of discrete task instances because also "parts" of task instances can be accepted. Thus the constraint relaxation delivers a safe upper bound on the accepted workload.

C3 Processed workload. The processed workload $\text{proc}_i[s, t)$ depends on the remaining service $\gamma_i[s, t)$ and the processing request of task τ_i, where the latter is given by the pending workload $\text{pend}_i(s)$ at s and the incoming accepted processing request $\alpha_i[s, t) \leq \alpha_i^\infty$. It is the minimum of the remaining service and the accepted processing request since not more than the remaining service but also not more than the accepted workload request can be processed in $[s, t)$. In order to simplify the constraint, we bound the processed workload $\text{proc}_i[s, t)$ from above by the remaining service $\gamma_i[s, t)$:

$$\text{proc}_i[s, t) = \min\left\{\gamma_i[s, t), \text{pend}_i(s) + \alpha_i[s, t)\right\} \leq \gamma_i[s, t). \quad (11)$$

The remaining service $\gamma_i[s, t)$ has been derived [19] as

$$\gamma_i[s, t) = \sup_{s \leq t^* \leq t} \{0, \beta[s, t^*) - A_{i-1}[s, t^*) - \text{Pend}_{i-1}(s)\} \quad (12)$$

where $A_{i-1}[s, t^*) + \text{Pend}_{i-1}(s)$ is the accepted workload of higher priority tasks. We bound the remaining service from above by

$$\gamma_i[s, t) \leq \gamma_i^+[s, t) = \sup_{s \leq t^* \leq t} \{0, \beta[s, t^*) - A_{i-1}[s, t^*)\}. \quad (13)$$

for simplification. Thus, a simplified constraint which upper-bounds the processed workload $\text{proc}_i[s, t)$ is

$$0 \leq \text{proc}_i[s, t) \leq \gamma_i^+[s, t). \quad (14)$$

Problem 2. *Problem 1 does not yet correspond to a standard ILP-problem due to the involved constraint for processed workload in Eq. 14. The upper bound of* $\text{proc}_i[s, t)$ *depends on the curve* A_{i-1} *and not only the value* $A_{i-1}[s, t)$*. We strive in the following to transform Eq. 14 into an ILP constraint, so that the optimization problem with the objective function given in Eq. 4 is then constrained by (Eq. 6, 7, 8, 14, 10) and (Eq. 30, 31, 32, 33, 34, 35, 36).*

The constraint in Eq. 14 which we repeat here

$$0 \leq \text{proc}_i[s, t) \leq \gamma_i^+[s, t) = \sup_{s \leq t^* \leq t} \{0, \beta[s, t^*) - A_{i-1}[s, t^*)\} \quad (15)$$

prevents the optimization from being treated as a linear problem. The reason is that $\gamma_i^+[s, t)$ is a *nonlinear* function and this in the *curve* A_{i-1} (not in the optimization variable $A_{i-1}[s, t)$!). In order to simplify the constraint (14), it is our intention to

(Step 1): bound $\gamma_i^+[s, t)$ by a function Γ_i in the variable $A_{i-1}[s, t)$ — not the curve A_{i-1} — such that

$$\gamma_i^+[s, t) \leq \Gamma_i(A_{i-1}[s, t)). \quad (16)$$

(Step 2): over-approximate the nonlinear decreasing function $\Gamma_i(A_i[s, t))$ by a step function. The step function can be encoded by the weighted sum of 0-1-integer variables. This transforms the optimization problem in a linear integer programming (ILP) problem which can be handled by standard methods.

(Step 1) To provide a solution for (Step 1), we need to find the cumulative workload curve $\tilde{A}_{i-1}$ which maximizes the supremum expression in Eq. 15 and goes through the given value $A_{i-1}[s, t)$. The function $\Gamma_i(A_{i-1}[s, t))$, indicating an upper bound of the remaining service, is therefore

$$\Gamma_i(A_{i-1}[s, t)) = \sup_{s \leq t^* \leq t} \{0, \beta[s, t^*) - \tilde{A}_{i-1}[s, t^*)\} \quad (17)$$

where the argument $A_{i-1}[s, t)$ of Γ_i refers to the constraint $\tilde{A}_{i-1}[s, t) = A_{i-1}[s, t)$. In Lemma 4.2, we first present a definition of $\tilde{A}_{i-1}$ and show that this curve has some required properties. In the subsequent Theorem 4.3, we show that $\tilde{A}_{i-1}$ actually maximizes the supremum expression in Eq. 15 for a given value $A_{i-1}[s, t)$.

Lemma 4.2. *The cumulative load-based arrival function* $\tilde{A}_{i-1}[s, t^*)$ *in variable* t^* *with* $0 \leq s \leq t^* \leq t$ *is defined as*

$$\tilde{A}_{i-1}[s, t^*) = \quad (18)$$
$$\begin{cases} 0 & \text{for } s \leq t^* \leq s + (t - s) - t_{min} \\ A_{i-1}[s, t) - A_{i-1}^+(t - t^*) & \text{for } s + (t - s) - t_{min} \leq t^* \leq t \end{cases}$$

where

$$A_{i-1}[s, t) = const. \quad (19)$$
$$A_{i-1}^-(t - s) \leq A_{i-1}[s, t) \leq A_{i-1}^+(t - s) \quad (20)$$
$$t_{min} = \min\left\{\sigma \mid A_{i-1}^+(\sigma) = A_{i-1}[s, t)\right\}. \quad (21)$$

We want to show that $\tilde{A}_{i-1}[s, t^*)$ *is bounded by*

$$0 \leq \tilde{A}_{i-1}[s, t^*) \leq A_{i-1}^+(t^* - s), \quad (22)$$

and fulfills the constraint

$$\tilde{A}_{i-1}[s, t) = A_{i-1}[s, t). \quad (23)$$

Proof. We first verify the lower and upper bound given in Eq. 22. For the interval $s \leq t^* \leq s + (t - s) - t_{min}$, Eq. 22 is satisfied because $\tilde{A}_{i-1}[s, t^*) = 0$.

For the interval $s + (t - s) - t_{min} \leq t^* \leq t$, the upper bound is guaranteed since $A_{i-1}[s, t) - A_{i-1}^+(t - t^*) \leq A_{i-1}^+(t - s) - A_{i-1}^+(t - t^*) \leq A_{i-1}^+(t^* - s)$. The lower bound is also guaranteed since $A_{i-1}[s, t) - A_{i-1}^+(t - t^*) \geq A_{i-1}[s, t) - A_{i-1}^+(t_{min}) = 0$. That the constraint 23 is met by the function $\tilde{A}_{i-1}$ can be shown by substituting t^* by t in Eq. 18 and considering the fact that $A_{i-1}^+(0) := 0$. □

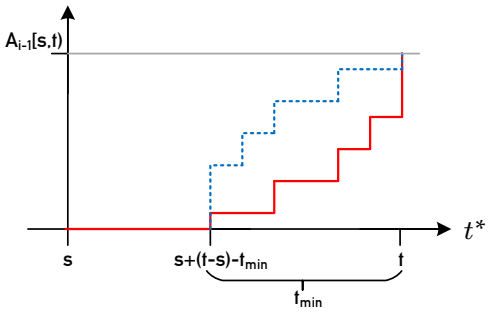


Figure 2: Illustration of theorem proof. *The blue dashed curve shows* $A^{+}_{i-1}(\Delta t)$ *for* $0 \leq \Delta t \leq t_{min}$ *in* $[s + (t-s) - t_{min}, t)$ *and the red curve represents* $\tilde{A}_{i-1}$ *in* $[s, t)$.

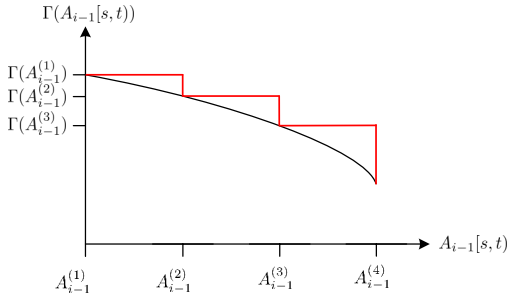


Figure 3: Over-approximation of $\Gamma_i(A_i[s,t))$ by a step function

Theorem 4.3. *There is no other cumulative load-based arrival function* $\check{A}_{i-1}$ *than* $\tilde{A}_{i-1}$ *as given in Lemma 4.2 which fulfills the constraints*

$$0 \leq \check{A}_{i-1}[s, t^*) \leq A^{+}_{i-1}(t^* - s) \tag{24}$$

$$\check{A}_{i-1}[s, t^*) = A_{i-1}[s, t) \tag{25}$$

with $A^{-}_{i-1}(t-s) \leq A_{i-1}[s,t) \leq A^{+}_{i-1}(t-s)$ *AND maximizes the expression*

$$\sup_{s \leq t^* \leq t} \{0, \beta[s, t^*) - \check{A}_{i-1}[s, t^*)\}. \tag{26}$$

Proof. The function $\tilde{A}_{i-1}$ fulfills the given constraints (24, 25), this has been shown in Lemma 4.2. The function $\tilde{A}_{i-1}$ also maximizes the supremum expression $\sup_{s \leq t^* \leq t}\{0, \beta[s,t^*) - A_{i-1}[s,t^*)\}$, if there is no other left-continuous step function $\check{A}_{i-1}$ fulfilling the constraints (24, 25) which is at any instant $s \leq t^* \leq t$ smaller than $\tilde{A}_{i-1}$. In the interval $s \leq t^* \leq s + (t-s) - t_{min}$, we have by definition $\tilde{A}_{i-1}[s,t^*) = 0$ so that $\tilde{A}_{i-1} \leq \check{A}_{i-1}$ is true. The time interval $t_{min} = \min\left\{\sigma | A^{+}_{i-1}(\sigma) = x\right\}$ is the shortest time interval required for any curve $\check{A}_{i-1}$ to grow up to the value x. This means that any curve $\check{A}_{i-1}$ fulfilling (24, 25) has to be non-zero starting at the latest from $s + (t-s) - t_{min}$.

In the time interval $s + (t-s) - t_{min} \leq t^* \leq t$, the definition of $\tilde{A}_{i-1}[s,t^*)$ ensures that load is introduced as late possible. This due to the fact that $A^{+}_{i-1}(\Delta t)$ is by definition a subadditive load function. By horizontally and vertically mirroring $A^{+}_{i-1}(\Delta t)$ for $0 \leq \Delta t \leq t_{min}$, a load curve with the steepest possible increase in growth rate over time is created in this interval cf. Figure 2. This mirrored function defines the value domain in the interval $s+(t-s)-t_{min} \leq t^* \leq t$ of $\tilde{A}_{i-1}$, since $\tilde{A}_{i-1}[s,t^*) = x - A^{+}_{i-1}(t-t^*)$ for $s + (t-s) - t_{min} \leq t^* \leq t$. This means no other curve $\check{A}_{i-1}$ can increase later than $\tilde{A}_{i-1}$, which proves that $\forall t^* : \tilde{A}_{i-1}[s,t^*) \leq \check{A}_{i-1}[s,t^*)$. □

(Step 2) The upper bound of the remaining service $\Gamma_i(A_{i-1}[s,t))$ is a nonlinear but monotonically decreasing function. To be able to formulate the optimization problem as an integer linear programming (ILP) problem, $\Gamma_i(A_{i-1}[s,t))$ is approximated from above by a step function with M steps as shown in Figure 3. Let $A^{(m)}_{i-1}$ with $1 \leq m \leq M+1$ define a uniformly discretized set of possible values for $A_{i-1}[s,t)$

$$A^{(m)}_{i-1} = A^{-}_{i-1}(t-s) + \frac{m-1}{M} \cdot \left(A^{+}_{i-1}(t-s) - A^{-}_{i-1}(t-s)\right). \tag{27}$$

If $A^{(m)}_{i-1} < A_{i-1}[s,t) \leq A^{(m+1)}_{i-1}$ then the step size is $\Gamma_i(A^{(m)}_{i-1})$. This approximation can be formalized by the following constraints

$$\xi_m \cdot A_{i-1}[s,t) - A^{(m)}_{i-1} \cdot \xi_m \geq 0 \tag{28}$$

$$-\xi_m \cdot A_{i-1}[s,t) + A^{(m+1)}_{i-1} \cdot \xi_m \geq 0 \tag{29}$$

$$\sum_{m=1}^{M} \xi_m = 1 \tag{30}$$

$$\gamma^{+}_i[s,t) \leq \sum_{m=1}^{M} \xi_m \Gamma_i(A^{(m)}_{i-1}). \tag{31}$$

where (28, 29) determine the interval in which $A_{i-1}[s,t)$ falls. Eq. (30) marks this interval by setting the corresponding 0-1-integer variable to 1, and Eq. 31 selects the step size.

In Eqs. (28, 29) the product of the continuous variable $A_{i-1}[s,t)$ and the integer variable ξ_m appears. For linearization, the product can be replaced by an auxiliary continuous variable Ξ_m so that

$$\Xi_m - A^{(m)}_{i-1} \cdot \xi_m \geq 0 \tag{32}$$

$$-\Xi_m + A^{(m+1)}_{i-1} \cdot \xi_m \geq 0 \tag{33}$$

if the following additional constraints are introduced [20]

$$\Xi_m - A^{(m+1)}_{i-1} \cdot \xi_m \leq 0 \tag{34}$$

$$-A_{i-1}[s,t) + \Xi_m \leq 0 \tag{35}$$

$$A_{i-1}[s,t) - \Xi_m + A^{(m+1)}_{i-1} \cdot \xi_m \leq \sum_{\nu=1}^{M} A^{(\nu+1)}_{i-1} \cdot \xi_\nu. \tag{36}$$

We have thus found that the complete optimization problem can be described by the objective function given in Eq. 4, and the constraints (Eq. 6, 7, 8, 14, 10) as well as (Eq. 30, 31, 32, 33, 34, 35, 36).

4.3 Computing $A^{+}_{i}(\Delta t)$ Over a Sequence of Intervals

In order to construct the curve $A^{+}_{i}(\Delta t)$ over time, the computation of the optimization problem has to be performed repeatedly for the time intervals Δt_ν which are integer multiples of the discretization time step $\Delta\tau$

$$\nu \in \mathbb{N} : \ \Delta t_\nu = \nu \cdot \Delta\tau. \tag{37}$$

The time step is preferably chosen according to the system tick or, respectively, the smallest time constant of the system. If $A_i^+(\Delta t_\nu)$ is computed iteratively over increasing time intervals, then we know that workload which has definitely been rejected in a previous time interval cannot reappear in a subsequent time interval:

$$A_i^+(\Delta t_\nu) \leq A_i^{+,\infty}(\Delta t_\nu) - R(\Delta t_{\nu-1}) \tag{38}$$

where $R(\Delta t_{\nu-1})$ is the amount of workload which is definitely rejected in the time interval $\Delta t_{\nu-1}$

$$R(\Delta t_{\nu-1}) = \max_\nu \left\{ A_i^{+,\infty}(\Delta t_{\nu-1}) - A_i^+(\Delta t_{\nu-1}) \right\}. \tag{39}$$

The constraint in Eq. 38 couples the individual optimization problems for each time interval Δt_ν.

4.3.1 Reduction of Computational Complexity. The maximum time interval Δt_ν that needs to be computed corresponds to the largest representative time window needed for response time computation. According to [14], this is the largest level-i busy window BW_i of the investigated task set.

Definition 4.4. A level-i busy window BW_i, resp. BW_i^∞, is the maximal time interval during which the processing resource is busy with tasks having an equal or higher priority than task τ_i in the presence of finite, resp. infinite, queue capacity.

The maximum level-i busy window is known to correspond the earliest intersection between the maximum cumulative load-based arrival curve and the service curve because this is the first time when the accepted workload and the available service equal each other and the resource becomes idle again. This principle can be expressed by the following fixed-point equation

$$A_i^+(BW_i) = \beta(BW_i) \text{ resp.} \tag{40}$$

$$A_i^{\infty,+}(BW_i^\infty) = \beta(BW_i^\infty). \tag{41}$$

If we assume that a reasonable time step is in the order of $\Delta\tau = 10\ \mu s$ and the largest level-i busy window BW_i is in the order of $1 \ldots 1000\ ms$, then $100 \ldots 100\,000$ small ILP problems have to be solved. This is prohibitively costly. A powerful mean to reduce the number of ILP problems to be solved, is a heuristic which indicates if workload rejection is unlikely. We assume for the computation of $A_i^+(\Delta t_\nu)$ that the queue capacity is sufficient if the maximum utilization U_i incurred by $\mathcal{T}_i$ in the considered time interval, i.e., $U_i(\Delta t_\nu) = A_i^{+,\infty}(\Delta t_\nu)/\Delta t_\nu$ is below a certain utilization threshold $U_{i,th}$. The utilization threshold is heuristically set to $U_{i,th} = 1.2$. This means if $A_i^+(\Delta t_{\nu-1})$ is known and $U_i(\Delta t_\nu) < U_{i,th}$, then we immediately write

$$A_i^+(\Delta t_\nu) = A_{i-1}^+(\Delta t_\nu) + (\alpha_i^{+,\infty}(\Delta t_\nu) - \alpha_i^{+,\infty}(\Delta t_{\nu-1})). \tag{42}$$

We also assume if no new event occurs in the current time interval Δt_ν compared to the previous one $\Delta t_{\nu-1}$, no further workload rejection occurs. The heuristic delivers always a safe upper bound for $A_i^+(\Delta t_\nu)$ because the amount of accepted workload can only be overestimated by assuming sufficient queue capacity. In our experiments, we had to solve approximately 15-150 optimization problems per system, depending on the workload, with a run time of several seconds.

5 WORST-CASE RESPONSE TIME ANALYSIS

The worst-case response time $WCRT_i^\infty$ for task τ_i in case of infinite queue capacity can be precisely computed since a single worst-case scenario, captured in BW_i^∞, is known in detail. The derivation of $WCRT^\infty$, i.e. the worst case response time of a task τ_i under the SPP scheduling policy and in the presence of sufficient queue capacity, is based on [17]. Due to the fact that there is not a single worst-case scenario for all times w.r.t. to the response time in case of *finite* ready queues and queue overflow as described in Example 4.1, it is not possible to determine the $WCRT_i$ of a task τ_i on the basis of a concrete busy window as in the case of infinite ready queues. The only available information on the "worst-case" busy window, which has been constructed from the overestimated A_i^+, is its maximum length BW_i see 41. Therefore, we bound $WCRT_i$ as specified in Theorem 5.1.

Theorem 5.1.

$$WCRT_i = \min\{BW_i,\ WCRT_i^\infty,\ b_i + A_{i-1}^+(WCRT_i)\} \tag{43}$$

Proof. No worst-case response time $WCRT_i$ can be larger than the longest busy window BW_i since by definition a busy window BW_i starts with an activation of task τ_i only closes if no pending workload of task τ_i is left. Therefore BW_i is a safe upper bound for $WCRT_i$.

Alternatively, the worst-case response time of a task τ_i can be computed if the following is known: (1) the maximum execution demand C_i^+ of a task instance $\tau_i(q)$, and (2) the maximum interference that task instance $\tau_i(q)$ sees until its termination. The interference is caused by the execution of tasks with higher priority but also by task instances of task τ_i which haven been activated before $\tau_i(q)$ and are not processed yet. The maximum pending workload of task τ_i is the queue size b_i, while the maximum higher priority interference is $A_{i-1}^+(WCRT_i)$. Therefore, a safe upper bound on the $WCRT_i$ in case of finite queue capacity is $WCRT_i = b_i + A_{i-1}^+(WCRT_i)$. Note that this is an over-estimation because it is assumed that a full queue coincides with the maximum higher priority interference. Finally it is known that the worst-case response time $WCRT_i^\infty$ will always be equal or larger than $WCRT_i$ because the latter is derived without any discarding of pending workload. Since it is not known for a given data set which bound approach delivers the tightest result, the infimum of all three safe upper bounds is introduced. □

6 TYPICAL WORST-CASE ANALYSIS

TWCA as defined in [21] derives formal weakly-hard guarantees for every task τ_i in an SP(N)P-scheduled task set assuring that no more than m deadlines in any sequence of k consecutive executions of a task τ_i are missed. In this section, we extend the verification capability of TWCA to weakly-hard guarantees of *generalized* deadline misses considering systems with limited queue capacity.

TWCA assumes that the system under analysis has a typical configuration which is schedulable. Deadline misses are only caused when additionally so called overload activations occur. For instance, a given system may be schedulable if all tasks are activated periodically but additional sporadic activations may induce overload. The method states that there is a maximal but limited time window $\Delta T_k^{j\to i}$ during which overload activations of a task τ_j can have

an impact on k consecutive instances of task τ_i where with $j \leq i$. This time window is composed in case of an SPP-scheduled system according to the following equation

$$\Delta T^{j \to i} = \begin{cases} BW_i + \delta_i^+(k) & i = j \\ BW_i + \delta_i^+(k) + WCRT_i & i \neq j. \end{cases} \tag{44}$$

The term $\delta_i^+(k)$ describes the longest possible time window in which exactly k activations of task τ_i can be observed. The term BW_i is the longest interval before $\delta_i^+(k)$ during which overload can impact the first instance of task τ_i in the k sequence. The term $WCRT_i$ is the longest time interval after $\delta_i^+(k)$ during which overload can impact the last instance of task τ_i in the k sequence due to preemption.

With the maximum overload activation event arrival curve $\eta^j_{over}(\Delta t)$ of task τ_j, the maximum number of overload activations in the interval $\Delta T_k^{j \to i}$ can be determined as $\eta^j_{over}(\Delta T_k^{j \to i})$. Each overload activation can at most cause N_i deadline misses of task τ_i. So an upper bound for the maximum number of deadline misses in any sequence of k consecutive executions of a task τ_i (also called deadline miss model DMM) is[1]

$$dmm_i(k) = N_i \cdot \sum_{j=1}^{i} \eta^j_{over}(\Delta T_k^{j \to i}). \tag{45}$$

For the case of finite queue capacity, this basic TWCA principle remains valid if (1) the input parameters are re-computed under this new restriction as done above, and (2) the schedulability condition is adapted to the detection of *generalized* deadline misses as following. In the case of finite queue capacity and generalized deadline misses, a task τ_i is schedulable if neither a deadline miss or a queue overflow is possible. As (3)rd requirement, we ask for a queue dimensioning that is sufficient if a task τ_i behaves according to its timing restriction. I.e. if the worst-case response time of τ_i is smaller than the required deadline, no queue overflow should take place.

The finite queue capacity impacts the DMM $dmm_i(k)$ compared to the case of infinite queues as follows: For the impact interval of an overload activation, we have $\Delta T_k^{j \to i} \leq \Delta T_k^{j \to i, \infty}$ since $BW_i \leq BW_i^\infty$ and $WCRT_i \leq WCRT_i^\infty$. The effect decreases with increasing k. With regard to N_i and N_i^∞, there is no generally true order relation. N_i^∞ corresponds to the number of jobs of task τ_i which miss their deadlines in the longest level-i busy window [21]. For finite queue capacity, we have indeed $BW_i \leq BW_i^\infty$ but a safe bound on N_i equals the number of *all* jobs of task τ_i in BW_i. Consider e.g. task τ_3 Fig. 1b, where one overload activation causes 2 of 2 jobs in the level-3 busy window to experience a generalized deadline miss (a conventional deadline miss and a queue overflow). However, in the worst case with infinite queues depicted in Fig. 1a only 1 deadline is missed in BW_i^∞. The reason is that a queue overflow may reject a job, even if it would theoretically meet its deadline if accepted. In conclusion, whether or not $dmm_i(k) \leq dmm_i^\infty(k)$ depends essentially on BW_i / BW_i^∞.

[1] A more refined TWCA approach has been proposed by [21] for infinite queue capacity. The concept cannot be immediately applied to finite queue capacity. We therefore use the basic approach for finite queues, and [21] for infinite queues in our experiments.

7 EVALUATION

The performed experiments serve to test whether transient overload peaks can be reduced by finite queues due to the effect of overflow. If so, the maximum busy windows and the worst-case response times will be shortened and the weakly-hard guarantees will be improved under finite queues compared to the case of infinite queues.

7.1 Experimental Setup

We have tested our overload analysis on embedded software systems which have characteristics as described in "Real World Automotive Benchmarks For Free" [13]. The benchmark is suitable for our purpose since it defines the timing properties of an automotive engine control application, which is typically marked by both a high static utilization and transient workload peaks caused by interrupts. Such industrial systems can often not be proven to have hard real-time properties in the worst case, but often WHRT properties are sufficient due to functional robustness [18]. The benchmark addresses both single core and multicore architectures, and we selected the single core variant according to our system model. Basically, the single core variant can neglect communication overhead and scales workload.

The engine control application conforms to the AUTOSAR standard, and is structured into 10 periodically activated tasks with periods ranging from $1ms$ to $1000ms$ and implicit deadlines. Each task is a container for a set of runnable entities, i.e. a set of software functions which are executed in a row at each task activation. We scaled the total number of runnables R in the system from $[1000, 1500]$ to $[250, 375]$ due to the use of a single core platform, and each task contains the defined share of the total number of runnables. WCETs of runnables, which are then accumulated to the task WCET, are determined according to the rules defined in the benchmark. Note that variability in the generated systems results from the total number of runnables and the runnable WCETs, the latter being extracted from a Weibull distribution. Tasks are scheduled according to the SPP scheduling algorithm, and priorities are assigned proportionally to the activation rate. Each task is of the basic conformance class such that it can only be preempted by tasks of higher priority, several concurrent activations of a task are not allowed. This means that each task has a ready queue with one slot. Task properties are summarized in Table 1.

The benchmark also specifies a high interrupt workload at top priorities, however, details on interrupt characteristics are scarce. It is known that interrupts sporadically activate some runnables, and that the overall interrupt-related utilization may reach up to 30%. Arbitrary event arrival curves are proposed to model the arrival pattern of activation events, yet none are indicated. We chose to add 8 interrupts service routines (ISRs), each consisting of two runnables with a runnable WCET of $10us$. The ISRs have a sporadically bursty event arrival pattern [16], which is defined by a set of 3 parameters. The outer period T_{out} describes the minimum distance between any two bursts of interrupt events. A single burst contains at most b events, and the inner period T_{in} describes the minimum distance between any two burst events. Those 8 ISRs are schedulable. The transient overload in the system is caused by an additional ISR with high worst-case workload containing 10 runnables. The parameters of the overload ISR, namely (1) the runnable WCET and (2) the

characteristics of the sporadically bursty event arrival pattern, are added as controlled impact factors in the experiments.

As a tool for system generation we created pySMFF, which is a self-written version of System Models for Free (SMFF) [15] in Python. The use of Python instead of Java eases scripting and integrates well with our analysis framework in Python. We generated 15 different systems according to the benchmark, which vary in the total number of runnables R and the runnable WCETs of the periodic tasks.

Table 1: Benchmark Data [13] and Added ISRs

Task	Prio.	# Runn.	Activation Pattern
$\tau_1 - \tau_8$: ISR	$1-8$	2	$b = 5,\ T_{in} = 20us\ T_{out} = 25ms$
τ_9: Overload ISR	9	10	$b = 5,\ T_{out} = 500ms$; Note: T_{in}, $runnWCET$ controlled
τ_{10}: 1 ms task	10	$0.03R$	periodic
τ_{11}: 2 ms task	11	$0.02R$	periodic
τ_{12}: 5 ms task	12	$0.02R$	periodic
τ_{13}: 6.6 ms task [2]	13	$0.15R$	periodic
τ_{14}: 10 ms task	14	$0.25R$	periodic
τ_{15}: 20 ms task	15	$0.25R$	periodic
τ_{16}: 50 ms task	16	$0.03R$	periodic
τ_{17}: 100 ms task	17	$0.20R$	periodic
τ_{18}: 200 ms task	18	$0.01R$	periodic
τ_{19}: 1000 ms task	19	$0.04R$	periodic

7.2 Experiments

In the following, we investigate the behavior of the described benchmark system with implicit deadlines for periodic tasks and one ready queue slot.

7.2.1 Rejected Workload. Finite queues are appear as a promising mean to disburden the system in overload conditions, since only a limited number of pending jobs per task is accepted at a time. The amount of rejected workload due to queue overflow is an insightful variable showing to which extent the system is guaranteed to be freed from workload when overload occurs. Therefore, we compute the difference $A_i^{\infty,+}(\Delta t) - A_i^{+}(\Delta t)$ which indicates the amount of discarded workload in Δt w.r.t. the task set $\mathcal{T}_i$ when maximizing the accepted workload $A_i(\Delta t)$. Maximizing the accepted workload corresponds to the worst case w.r.t. busy window, the response times, and the (m,k)-guarantees. In the considered benchmark system, it is the 1ms-task τ_{10} which is most affected by overload because the period of the task is comparable to the duration of the interrupt burst. Thus it is likely that a second job is activated before the first job has terminated, and consequently the overflow mechanism becomes active. Figure 4a shows $A_{10}^{\infty,+}(\Delta t) - A_{10}^{+}(\Delta t)$ for $\Delta t = 10ms$. Beyond this time interval, rejection is unlikely because the maximum utilization $A_{10}^{\infty,+}(\Delta t)/\Delta t$ approaches the overload-free behavior. We have observed, that tasks $\tau_{>10}$ experience no guaranteed rejection of workload in the worst case. The amount of guaranteed rejected workload $A_{10}^{\infty,+}(\Delta t) - A_{10}^{+}(\Delta t)$ is expected to depend on the parameters characterizing the overload peak, which are therefore varied in the experiments. Intuitively, the higher an overload peak is, the more workload has to be discarded. Increasing the runnable WCET of the overload ISR actually leads to a monotonic increase of discarded workload, this is illustrated in Fig. 4a. At the same time, we found that the amount of discarded workload is very sensitive to the activation pattern of overload events, cf. T_{in}.

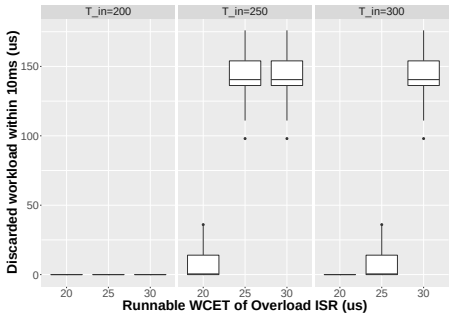
Considering Fig. 4a, the principal observation is, however, that the amount of rejected workload is surprisingly small in the worst case independent of the chosen overload parameters. W hy is that? The worst case w.r.t. busy window, the response times, and the (m,k)-guarantees occurs – as mentioned above – if as much workload as possible is accepted in a given interval of size Δt. The optimization algorithm therefore selects the solution which rejects as little workload as possible. Often corner cases can be constructed, where queues are nearly full but rejection can just be avoided.

Reconsider the encouraging Scenario 2 illustrated in Fig. 1b, where workload rejection in case of full queues is effective and significantly reduces, e.g., the busy window of task τ_4 compared to the classical worst case with infinite queue capacity in Fig. 1a. However, Scenario 3 in Fig. 1c differs from the classical worst case in Fig. 1a only in the slightly shortened execution time of the 2nd job of task τ_2. Rejection of jobs can be completely avoided in Scenario 3 despite the finite queue capacity. Consequently, the workload-based arrival curves in Fig. 1d of the classical worst case with infinite queues and the Scenario 3 hardly differ. This means that finite queues *can* lead to significant workload rejection in some scheduling scenarios but *do not have to* in all high-load cases.

We generalize now this observation. The solution to the problem of maximizing the accepted workload $A_i^{+}(\Delta t)$ is to find a scheduling scenario in Δt which respects the constraints (i.e. allowed execution times, activation patterns), includes as much workload as possible and avoids queue overflow if possible. The less restrictive the constraints are, the easier it is to find a scheduling scenarios which just avoid rejection. That is, a large ratio of (a) C_i^{+} and C_i^{-} and (b) $A_i^{\infty,+}(\Delta t)$ and $A_i^{\infty,-}(\Delta t)$ as well as (c) large queues favor rejection-free worst case corner cases.

7.2.2 Response Times and Deadline Miss Model. The behavior of a system, which is guaranteed to reject some workload while maximizing $A_i(\Delta t)$, is shown in Table 4b. The table contains queue sizes, BWs, WCRTs and DMMs for all late tasks, and compares those values with the results for an identical system but with sufficiently sized queues. The busy window BW_i is the basic information to compute the maximum response time $WCRT_i$ (Eq. 43) and the DMM $dmm_i(k)$ (Eq. 45) for a given k. It could be reduced in all cases, but only for the 10ms task to a significant extent. Consequently, we obtain $WCRT_i \leq WCRT_i^{\infty}$. The 10ms task can even meet its deadline under finite queue capacity due to the guaranteed workload rejection by higher-priority tasks. (Note that there are other analyzed systems, where the 10ms task does not profit this much from rejection.) Let us now discuss the obtained DMMs for the finite queue capacity which do not all improve. For instance, the 1ms task and the 2ms tasks have even a *worse* DMM than in the case of infinite queues. This effect is related to the factor N_i in Eq. 45. N_i counts the maximum number of both late and discarded jobs in BW_i. As mentioned in Section 6, a safe and realistic upper bound on N_i corresponds to the number of all jobs of τ_i in BW_i.

[2] Operating point of angle-sychronous task



(a) Discarded workload $A_{10}^{\infty,+}(\Delta t) - A_{10}^{+}(\Delta t)$ over varying parameters of the overload ISR.

task	# queue slots	busy windows	worst-case response times	deadline miss models for $k \in \{20, 50, 100\}$	
	s_i, s_i^∞	BW_i, BW_i^∞	$WCRT_i, WCRT_i^\infty$	$dmm_i(k)$	$dmm_i^\infty(k)$
1ms task	1, ≥ 2	2650, 2798	2466, 2466	15, 15, 15	10, 10, 10
2ms task	1, ≥ 2	2780, 2934	2710, 2866	10, 10, 10	5, 5, 5
10ms task	1, ≥ 2	9910, 15813	9904, 10558	0, 0, 0	5, 10, 15

(b) Results for two generated systems which differ in the queue capacity (finite vs. infinite). The overload ISR is characterized by $runnWCET = 30us$, $b = 5$, $T_{in} = 250$, $T_{out} = 500ms$.

Figure 4: Experimental results

This can be larger than N_i^∞, which only counts the number of conventional deadline misses BW_i^∞. Therefore DMMs only profit from finite queues, if BW_i is significantly smaller than BW_i^∞ such that fewer jobs are included in BW_i than in BW_i^∞. From a scheduling perspective, this means that the discarding of jobs is compensated by the improved schedulability of other jobs of the same task. In our experiments, BW_i is still relatively large compared to BW_i^∞, since little workload is guaranteed to be rejected. In conclusion, WHRT guarantees could not be improved by finite queues except for the 10ms task. Yet it is interesting to see that when increased DMMs are acceptable, the queue size could be divided by 2. This allows a trade-off between available queue sizes, and achieved timing guarantees. Such a trade-off is interesting, if it is important to use standardized queue sizes and/or if memory is expensive e.g. networks-on-chip.

8 CONCLUSION

In this work, we analyzed the worst-case timing behavior of weakly-hard real-time (WHRT) systems with finite ready queues. The analysis is, to the best of our knowledge, the first one to provide worst-case response times as well as formal bounds on deadline misses and overflow losses for such systems. In particular, we investigated whether finite queues sizes can limit overload peaks due to the effect of overflow such that the overall system schedulability is improved. We summarize now the lessons learned. Finite queues can principally limit the amount of accepted workload. However, often corner cases can be constructed where overflow can just be avoided or is small, so that *in the worst case* the workload-limiting of finite queues is weak. As a consequence, better worst-case timing guarantees than in the infinite queue case are rare as demonstrated by our use case of an engine control system. Still the analysis allows to formally explore the interesting trade-off between weakly-hard guarantees and costly queue sizes. In the future, we aim to identify and evaluate conditions additionally to the limited queue size, which make job dropping also in the worst case efficient for WHRT systems.

ACKNOWLEDGMENTS

This work has received funding from the German Research Foundation (DFG) under the contract number TWCA ER168/30-1.

REFERENCES

[1] S. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, and F. Wang. 1991. On the competitiveness of on-line real-time task scheduling. In *Real-Time Systems Symposium, 1991. Proceedings., Twelfth.* 106–115.
[2] G. Bernat, A. Burns, and A. Liamosi. 2001. Weakly Hard Real-Time Systems. *IEEE Trans. Comput.* 50, 4 (2001), 308–321.
[3] Guillem Bernat and Ricardo Cayssials. 2001. Guaranteed on-line weakly-hard real-time systems. In *RTSS*. IEEE, 25–35.
[4] Rainer Blind and Frank Allgöwer. 2015. Towards Networked Control Systems with guaranteed stability: Using weakly hard real-time constraints to model the loss process. In *Decision and Control, 2015 IEEE 54th Annual Conference on*. 7510–7515.
[5] G. Buttazzo, M. Spuri, and F. Sensini. 1995. Value vs. deadline scheduling in overload conditions. In *RTSS, 1995. Proceedings., 16th IEEE*. 90–99.
[6] Giorgio C Buttazzo and John A Stankovic. 1995. Adding robustness in dynamic preemptive scheduling. In *Responsive Computer Systems: Steps Toward Fault-Tolerant Real-Time Systems*. Springer, 67–88.
[7] Cheng-Shang Chang and R. L. Cruz. 1999. A time varying filtering theory for constrained traffic regulation and dynamic service guarantees. In *INFOCOM '99, Proceedings. IEEE*, Vol. 1. 63–70 vol.1.
[8] RL Cruz and M Taneja. 1998. An analysis of traffic clipping. In *Princeton University*. Citeseer.
[9] UmaMaheswari C Devi and James H Anderson. 2008. Tardiness bounds under global EDF scheduling on a multiprocessor. *Real-Time Systems* 38, 2 (2008), 133–189.
[10] Goran Frehse, Arne Hamann, Sophie Quinton, and Matthias Wöhrle. 2014. Formal Analysis of Timing Effects on Closed-loop Properties of Control Software. In *RTSS*. Rome, Italy.
[11] Moncef Hamdaoui and Parameswaran Ramanathan. 1995. A dynamic priority assignment technique for streams with (m, k)-firm deadlines. *IEEE transactions on Computers* 44, 12 (1995), 1443–1451.
[12] Gilad Koren and Dennis Shasha. 1995. Skip-over: Algorithms and complexity for overloaded systems that allow skips. In *RTSS, 1995*. IEEE, 110–117.
[13] Simon Kramer, Dirk Ziegenbein, and Arne Hamann. 2015. Real world automotive benchmarks for free. In *WATERS*.
[14] John P Lehoczky. 1990. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *RTSS, 1990. Proceedings., 11th*. IEEE, 201–209.
[15] Moritz Neukirchner, Steffen Stein, and Rolf Ernst. 2011. SMFF: System Models for Free. In *WATERS*. Porto, Portugal.
[16] K. Richter. 2005. *Compositional Scheduling Analysis Using Standard Event Models: The SymTA-S Approach.* Ph.D. Dissertation. TU Braunschweig.
[17] Ken W Tindell, Alan Burns, and Andy J. Wellings. 1994. An extendible approach for analyzing fixed priority hard real-time tasks. *Real-Time Systems* 6, 2 (1994), 133–151.
[18] Sebastian Tobuschat, Rolf Ernst, Arne Hamann, and Dirk Ziegenbein. 2016. System-level timing feasibility test for cyber-physical automotive systems. In *SIES*. IEEE, 1–10.
[19] Ernesto Wandeler. 2006. *Modular performance analysis and interface-based design for embedded real-time systems.* Ph.D. Dissertation. ETH Zurich.
[20] H. P. Williams. 2013. *Model building in mathematical programming* (5th ed ed.). Wiley, Hoboken, N.J.
[21] W. Xu, Z. Hammadeh, S. Quinton, A. Kröller, and R. Ernst. 2015. Improved Deadline Miss Models for Real-Time Systems Using Typical Worst-Case Analysis. In *ECRTS*. Lund.