

# Response Time Analysis of Multiframe Mixed-Criticality Systems with arbitrary deadlines

Ishfaq Hussain · Muhammad Ali Awan ·  
Pedro F. Souto · Konstantinos Bletsas ·  
Benny Akesson · Eduardo Tovar

Received: date / Accepted: date

## 1 Abstract

The well-known model of Vestal aims to avoid excessive pessimism in the quantification of the processing requirements of mixed-criticality systems, while still guaranteeing the timeliness of higher-criticality functions. This can bring important savings in system costs, and indirectly help meet size, weight and power constraints. This efficiency is promoted via the use of multiple worst-case execution time (WCET) estimates for the same task, with each such estimate characterized by a confidence associated with a different criticality level. However, even this approach can be very pessimistic when the WCET of successive instances of the same task can vary greatly according to a known pattern, as in MP3 and MPEG codecs or the processing of ADVB video streams.

---

Ishfaq Hussain  
CISTER Research Center and ISEP/IPP, Porto, Portugal.  
E-mail: [hussa@isep.ipp.pt](mailto:hussa@isep.ipp.pt)

Muhammad Ali Awan  
CISTER Research Center and ISEP/IPP, Porto, Portugal.  
E-mail: [awa@isep.ipp.pt](mailto:awa@isep.ipp.pt)

Pedro F. Souto  
University of Porto, FEUP-Faculty of Engineering and CISTER Research Centre, Porto, Portugal.  
E-mail: [pfs@fe.up.pt](mailto:pfs@fe.up.pt)

Konstantinos Bletsas  
CISTER Research Centre and ISEP/IPP, Porto, Portugal.  
E-mail: [ksbs@isep.ipp.pt](mailto:ksbs@isep.ipp.pt)

Benny Akesson  
ESI (TNO), Eindhoven & University of Amsterdam, Amsterdam, the Netherlands.  
E-mail: [benny.akesson@tno.nl](mailto:benny.akesson@tno.nl)

Eduardo Tovar  
CISTER Research Centre and ISEP/IPP, Porto, Portugal.  
E-mail: [emt@isep.ipp.pt](mailto:emt@isep.ipp.pt)

In this paper, we present a schedulability analysis for the new multiframe mixed-criticality model, which allows tasks to have multiple, periodically repeating, WCETs in the same mode of operation. Our work extends both the analysis techniques for Static Mixed-Criticality scheduling (SMC) and Adaptive Mixed-Criticality scheduling (AMC), on one hand, and the schedulability analysis for multiframe task systems on the other. A constrained-deadline model is initially targeted, and then extended to the more general, but also more complex, arbitrary-deadline scenario. The corresponding optimal priority assignment for our schedulability analysis is also identified. Our proposed worst-case response time (WCRT) analysis for multiframe mixed-criticality systems is considerably less pessimistic than applying the static and adaptive mixed-criticality scheduling tests oblivious to the WCET variation patterns. Experimental evaluation with synthetic task sets demonstrates up to 20% and 31.4% higher scheduling success ratio (in absolute terms) for constrained-deadline analyses and arbitrary-deadline analyses, respectively, when compared to the best of their corresponding frame-oblivious tests.

## 2 Introduction

Recent trends in many real-time embedded domains (e.g., automotive and avionics) favor mixed-criticality systems, where computing tasks of different criticalities co-exist on the same processor. A task's criticality is a measure of the severity of the consequences of that task failing, in conjunction with the probability of such a failure. Accordingly, tasks of higher criticalities are developed according to stricter (and costlier) methodologies, and the same holds for the techniques for estimating their worst-case execution times (WCETs). Additionally, scheduling arrangements have to ensure that, even when sharing system resources, a misbehavior of a lower-criticality application cannot affect the timing behavior of a higher-criticality application. One arrangement that ensures that, while also promoting efficient platform utilization, is the mixed-criticality task model of Vestal. The two most established variants of that model are the static variant with execution monitoring [1] and the adaptive mode-based variant [2].

In the simpler static variant of Vestal's model, each task has a criticality level and a WCET estimate for each criticality level lower or equal to its own. At run-time, the execution times of all tasks are monitored and any job that exceeds the WCET estimate corresponding to the degree of confidence appropriate to its criticality level is killed. However, future jobs of the same task will still arrive as normal. For schedulability analysis, the classic worst-case response time (WCRT) recurrence is employed [3], assuming, for all tasks, WCET estimates with the degree of confidence appropriate for the lowest criticality level of the interfering task and of the task under analysis.

In the simplest case, the adaptive model involves two criticality levels and two modes (L and H) of operation. In L-mode, all tasks are present and WCETs are assumed for them which are *probably*, but not *provably*, safe. In case any

task executes for its assumed WCET estimate without completing, a mode switch is triggered. Then, low-criticality tasks are discarded, and provably safe, but potentially very pessimistic, WCETs are assumed for the remaining tasks. In each mode, all tasks present must provably meet their deadlines, assuming the respective WCET estimates for that mode. This arrangement considerably mitigates the inefficiency in platform utilization (and commensurate over-engineering) that results from the overwhelming pessimism in the derivation of provably safe WCET estimates for high-criticality tasks.

Both of the above scheduling arrangements promote efficient processor utilization. However, there is another source of inefficient resource use, which the present article intends to remedy. Namely, when the WCETs of successive jobs of the same task vary greatly by design, according to a known pattern. For such systems, assuming the maximum of the WCETs for all instances of the task (even the ones for which we *know* it is an overestimation) would grossly inflate the processor requirements. For example, in MPEG codecs [4], different kinds of frames (P, I or B) appear in a repeating pattern, with very different worst-case processing requirements. Moreover, Avionics Digital Video Bus (ADVB) [5] frames are transmitted uncompressed, to minimize encoding/encoding delays, but the fact that distinct types of ADVB frames exist (e.g., data, audio or video) implies different end-node processing requirements for each type. In other real-time industrial applications [6, 7], small amounts of data are collected periodically and then summarized and stored in batch after  $N$  periods, in an operation that involves costlier processing than that in the preceding periods. The *multiframe task model*, invented by Mok and Chen [8], and its analysis provide a way for efficiently dealing with patterned WCET variations in the single-criticality scheduling. However, until now this model and its existing analysis did not consider the scheduling of mixed-criticality tasks.

Under the multiframe model, a task with  $N$  frames is described by  $N$  different frame WCETs that repeat, in a cyclic manner, in the sequence of its jobs. The well-known Liu and Layland task model [9] then becomes a special case, where the number of frames is one for all tasks. The schedulability analysis for the multiframe model leverages the information about the pattern of frame WCET variation and achieves greater accuracy, compared to the frame-agnostic application of analysis for the Liu and Layland task model.

**The present work combines the mixed-criticality model of Vestal with the multiframe model**, in order to achieve similar improvements in the schedulability testing of mixed-criticality systems whose tasks' WCETs vary according to known patterns. Our four main contributions are the following:

1. The multiframe task-model is combined with the mixed-criticality Vestal model. It is termed the *multiframe Vestal model* for mixed-criticality systems, in its static and mode-based variants.
2. Based on the principles of mixed-criticality scheduling and established schedulability tests (SMC, AMC-rtb and AMC-max), we developed *adaptive multiframe mixed-criticality schedulability analyses*, SMMC, AMMC-

rtb and AMMC-max, for fixed-priority-scheduled mixed-criticality tasks deployed on a uniprocessor hardware platform. We initially target a constrained-deadline model, and subsequently extend the analysis for the more general but more complex scenario of arbitrary deadlines.

3. We identify the optimal priority assignment scheme, for use in conjunction with our schedulability analysis.
4. In experiments with synthetic workloads, the proposed analyses are compared in terms of scheduling success ratio, against the frame-agnostic analyses for the corresponding variants of the Vestal model.

The article is organized in eleven sections. Related work is discussed in Section 3. The system model is presented in Section 4. Section 5 discusses some important background results on mixed-criticality scheduling of constrained-deadline tasks. The schedulability analysis techniques for the static and mode-based variants of the multiframe Vestal model for mixed-criticality systems are presented in Section 6 assuming a constrained deadline model. The existing schedulability analyses on mixed-criticality scheduling of arbitrary-deadline tasks are presented in Section 7. The multiframe mixed-criticality analyses designed for constrained-deadline tasks are then generalized for arbitrary task deadlines, in Section 8. In Section 9, we provide an example of the analyses applied to a task-set. In Section 10, we identify an optimal priority assignment scheme for multiframe mixed-criticality systems, using our analysis as the schedulability test. Section 11 provides an experimental evaluation, using synthetic task sets, of the schedulability performance of the proposed analyses against those devised for the original (non-multiframe) Vestal model. Finally, Section 12 concludes this work.

### 3 Related Work

The two main variants (static [1] and adaptive mode-based [2]) of Vestal’s original mixed-criticality model [10] characterize each task with multiple worst-case task execution time (WCET) estimates, with a corresponding degree of confidence associated with a different criticality level (up to the task’s own criticality). This reflects the fact that, in the industry, the enormous costs of proving the safety of a WCET estimate (and the associated pessimism/overestimation) beyond doubt are justified only for high-criticality tasks [10]. For other tasks, less rigorously derived, WCET estimates are used – probably, but not provably, safe. Based on this model, several mixed-criticality scheduling arrangements have been devised for a variety of hardware platforms in the last decade. The majority of the work is summarized in a survey paper by Burns and Davis [11]. Here, we restrict ourselves to fixed-priority mixed-criticality scheduling algorithms on a uniprocessor platform.

Baruah and Burns [1] first proposed the use of execution-time monitoring, already supported in most embedded platforms. Any job by a lower-criticality task requiring more execution time than its most conservative WCET assumes, is terminated. Baruah, Burns and Davis [2] showed how standard worst-case

response time analysis can be applied to this model (termed “Static Mixed-Criticality” or SMC), by assuming, for all tasks, WCET estimates associated with a criticality level never higher than that of the task under analysis. Baruah, Burns and Davis also [2] proposed an adaptive mixed-criticality (AMC) scheduling technique that adds modes of operation. When operating on a mode different from the highest, if any task exceeds its assumed WCET for that mode, the tasks whose criticality level is equal to the current mode of operation are discarded and the system switches to a next higher mode. Two schedulability tests for AMC were devised: AMC-rtb and the tighter, but more complex, AMC-max. The AMC-rtb schedulability test derives a simple-to-compute upper bound, while AMC-max checks all the possible mode-switch instants to derive the worst-case. The AMC-max test is more accurate but not exact. This work is recently extended for arbitrary-deadline tasks [12]. Audsley’s priority assignment algorithm is also identified as being optimal for both SMC and AMC, in these works [2, 12].

Asyaban and Kargahi [13] developed exact analysis for AMC at the cost of losing optimality in the priority ordering. The author also derived the feasibility interval for mixed-criticality periodic tasks with offset [13]. Fleming and Burns [14] extended the dual-criticality AMC schedulability analysis to an arbitrary number of criticality levels and showed that AMC-rtb approximates AMC-max reasonably well. The later AMC-IA schedulability test [15, 16] may slightly outperform AMC-max in some cases. Zhao et al. [17] improve on AMC-max via a slightly different preemption model.

All aforementioned efforts have been developed in the context of a single-frame task model, where a single WCET per criticality level of a task is used for all its jobs. These analyses are pessimistic for applications whose execution requirement may vary from one instance (job/frame) to another, but the variation follows a repetitive pattern. Such applications are better modeled with a multiframe task model, which is a generalization of the single-frame task model. In the multiframe task model, the schedulability analysis takes advantage of the varying execution requirement to improve the scheduling performance, potentially reducing system cost (size, weight and power).

The multiframe task model was initially introduced by Mok and Chen [8] as a generalization of Liu and Layland’s task model. This model assumes that the WCETs of the successive jobs of the same task can vary with a repetitive pattern. The length of the pattern, in which each job may have a distinct WCET estimate is called the task’s “frame size”. The schedulability analysis of Mok and Chen [8] showed considerable improvements over the conservative schedulability analyses that use the highest task’s WCET estimate as the WCET of each job. Baruah et al. [18] improved the schedulability analysis of multiframe task model by considering the actual frame pattern rather than an accumulatively monotonic reordered pattern used by Mok and Chen [8]. The multiframe task model was further generalized by Baruah et al. [19] by considering additional task attributes, other than WCET, differing among frames of a task. Zuhily and Burns [20] proposed an exact worst-case response time analysis for non-accumulatively monotonic multiframe tasks scheduled

with a fixed-priority scheduling scheme. An idea similar to the multiframe task model has also been proposed for modeling signal processing applications using data-flow graphs. Each actor in a cycle-static data-flow (CSDF) graph has a number of phases, periodically repeating, each with its own execution time. This allows the throughput of the application to be more accurately predicted [21]. Note that these systems are typically statically scheduled and don't consider criticalities.

The schedulability analysis for the multiframe task model has not been formulated for mixed-criticality systems. This prevents the benefits of that model from being exploited in the context of mixed-criticality scheduling. The present work therefore introduces schedulability analyses based on the principles of SMC, AMC-rtb and AMC-max, which eliminate the latent pessimism and improve the scheduling performance by leveraging the properties of the multiframe task model. We further extend these proposed multiframe mixed-criticality analyses for the more generic arbitrary-deadline task model.

## 4 System Model

In this section, we formalize the mixed-criticality multiframe task model, in both of its variants, static and adaptive mode-based.

Consider a set  $\tau$  of  $n$  mixed-criticality sporadic tasks  $(\tau_1, \dots, \tau_n)$  on a uniprocessor system. Each task  $\tau_i$  has a relative deadline  $D_i$ , a minimum inter-arrival time  $T_i$ , a criticality level  $\kappa_i$ , which is either high (H) or low (L). We assume that there is a total order on the criticality levels, i.e. that L has lower criticality than H. For the sake of conciseness, we denote a low-criticality and a high-criticality task as L-task and H-task, respectively. Unlike jobs of conventional (i.e., single frame) tasks, successive jobs of the same multiframe task can differ in their WCETs, in a pattern that repeats after  $F_i$  successive jobs. Accordingly, any  $F_i$  successive jobs of  $\tau_i$  are called a *superframe* and we often refer to a job as a frame, especially when we want to distinguish between jobs in a superframe. In any schedule, the  $k^{th}$ ,  $(k + F_i)^{th}$ ,  $(k + 2F_i)^{th}$ , ... jobs of  $\tau_i$  all have the same worst-case execution time behavior. However, even the same frame of the same task has multiple estimates for its WCET, in our model, with different degree of confidence. Each job has a WCET estimate per criticality level less than or equal to its task's criticality level. Thus, job  $j$  of L-task  $\tau_i$  has a single WCET estimate,  $C_{i,j}^L$ , i.e., a low-criticality WCET, or L-WCET for short. On the other hand, job  $k$  of H-task  $\tau_i$ , has two WCET estimates, one L-WCET,  $C_{i,k}^L$ , and one high-criticality WCET, or H-WCET for short,  $C_{i,k}^H$ . L-WCETs can be optimistic. Thus,  $C_{i,j}^L \leq C_{i,j}^H$ . We do not assume that  $C_{i,j}^L > C_{i,k}^L$  implies  $C_{i,j}^H < C_{i,k}^H$ , for any  $j$  or  $k$ .

The semantics vary slightly, according to the particular variant of the model. The choice of scheduling algorithm is orthogonal to the above task model. In this paper, we assume fixed-priority scheduling, i.e., each task has a fixed priority.

Table 1 summarizes most symbols used in the worst-case response time analysis. Some other symbols are defined later in Sections 6 and 8.

#### 4.1 Static variant

In this variant, Whenever a job of some L-task  $\tau_i$ , corresponding to its  $j^{th}$  frame, executes for its entire L-WCET,  $C_{i,j}^L$ , without completing, then that particular job is terminated. This does not suppress the arrival of future jobs of  $\tau_i$ , subject to interarrival time constraints, and respecting the actual frame sequence of the task. Under this model variant, the system is schedulable if (i) no L-task misses its deadline as long as all jobs of all tasks, irrespective of their criticality, execute for up to their  $C_{i,j}^L$ , and also (ii) no H-task misses its deadline, as long as every job of every task  $\tau_i$  executes for up to its  $C_{i,j}^{\kappa_i}$ , where  $\kappa_i$  is the criticality of  $\tau_i$ , i.e., as long as L-tasks execute up to  $C_{i,j}^L$  and H-tasks execute up to  $C_{i,j}^H$ .

#### 4.2 Adaptive mode-based variant

As in the mode-based variant of Vestal’s model introduced by Baruah and Burns [2], the system operates in different modes – two in this paper, which is also the simplest case. The system boots in the default L-mode, where all tasks (i.e., both of low and high criticality) are present. For the L-mode, WCET estimates are assumed, for all task frames, that are most likely but not necessarily safe. If at any point in time, any job attempts to exceed its assumed WCET estimate for the L-mode, a mode switch is triggered. This means that all low-criticality tasks (L-tasks) are discarded and only high-criticality tasks (H-tasks) are allowed to execute. The system is mixed-criticality-schedulable if (i) all tasks meet their deadlines in the L-mode, assuming that their jobs execute in accordance to the WCETs assumed for that mode; and (ii) all H-tasks (including any jobs thereof caught up in the mode switch) meet their deadlines in the H-mode, assuming that their jobs can execute for as long as their respective H-mode WCET estimates. The latter are provably safe and typically very pessimistic.

### 5 Background on schedulability analyses of constrained-deadline tasks

Our analysis builds on the worst-case response time analysis for the multi-frame task model by Baruah et al. [18] and on the different mixed-criticality schedulability analysis techniques by Baruah, Burns and Davis [2]. In this section, we summarize those existing results.

Table 1: Symbols used in the analysis

Symbols	Description
$F_i$	Number of frames of a task $\tau_i$
$C_{i,j}^x$	WCET of $j^{th}$ frame of $\tau_i$ in mode $x \in \{L, H\}$
$g^x(\tau_i, j)$	Cumulative worst-case execution requirement of $j$ successive jobs of $\tau_i$ in mode $x \in \{L, H\}$ , where $1 \leq j \leq F_i$
$G^x(\tau_i, t)$	Cumulative worst-case processor request of $\tau_i$ in mode $x$ in any time interval of $t$ time units
$R_i^L$	WCRT of $\tau_i$ in L-mode
$R_i^H$	WCRT of $\tau_i$ in H-mode
$R_i^s$	WCRT of $\tau_i$ , if caught in a mode switch at time $s$
$r_i^L(q), r_i^H(q)$	Completion time of the $q^{th}$ job of $\tau_i$ in a level- $i$ busy period in L-mode and H-mode, respectively.
$D_i, T_i$	Deadline and period of a task $\tau_i$

### 5.1 Multiframe Task Model

To determine the schedulability of a multiframe task, it suffices to check if the WCRT of the frame with the largest WCET is smaller than the task deadline (which, for now can be assumed to be constrained, i.e.,  $D_i \leq T_i$ )<sup>1</sup>. Unlike what holds for the “single-frame” model, it is not enough to compute the number of job releases of each interfering task in the WCRT of the task under analysis. This is because each multiframe task  $\tau_i$  is characterized by a vector of WCET  $(C_{i,0}, C_{i,1}, \dots, C_{i,(F_i-1)})$ , and the phasing of the released jobs affects the amount of interference. This is illustrated in Fig. 1, which shows the 3 possible phasings of jobs of a task  $\tau_i$  with period  $T_i = 10$  and WCET vector  $(2, 4, 1)$ , with respect to a frame under analysis whose WCRT is between 14 and 20 time units. During this interval there are at most 2 jobs of the interfering task, but the amount of interference depends on the first job in that sequence. As illustrated, this interference is worst, 6, if the first job in the sequence is job 0,  $F_i, \dots$ . On the other hand, if the first job in the sequence is job 2,  $2 + F_i, \dots$ , the interference is the least, 3.

To efficiently compute the interference exerted by a multiframe task, Baruah et al. [18] define the  $g$  function:

$$g(\tau_i, k) = \begin{cases} 0 & \text{if } k = 0 \\ \max \left\{ \sum_{\ell=j}^{j+k-1} C_{i,(\ell \bmod F_i)} : 0 \leq j < F_i \right\} & \text{if } 1 \leq k \leq F_i \\ q \cdot g(\tau_i, F_i) + g(\tau_i, r) & \text{otherwise} \\ \text{where } q = k \operatorname{div} F_i \text{ and } r = k \operatorname{mod} F_i & \end{cases} \quad (1)$$

<sup>1</sup> The more general case of arbitrary deadlines is considered later in Section 8.

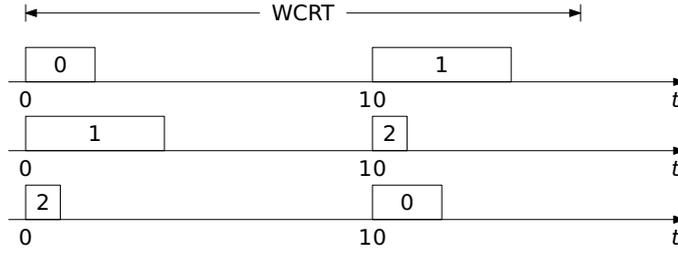


Fig. 1: Interference of a multiframe task depends on the phasing of its jobs w.r.t the job under analysis. (The numbers inside the rectangles are the frame numbers of the interfering task.) In this example, we assume that the WCRT of the frame under analysis is between 14 and 20, and that the interfering task has a period of 10 and its WCET vector is (2, 4, 1).

which bounds the cumulative WCET of any sequence of  $k$  jobs of task  $\tau_i$ . This function is then used to define the  $G$  function:

$$G(\tau_i, t) = g\left(\tau_i, \left\lceil \frac{t}{T_i} \right\rceil\right) \quad (2)$$

which bounds the cumulative WCET of task  $\tau_i$  over any time interval of duration  $t$ . Finally, the worst-case response time recurrence is:

$$R_i = g(\tau_i, 1) + \sum_{\tau_j \in hp(i)} G(\tau_j, R_i) \quad (3)$$

where  $hp(i)$  is the set of tasks with priority higher than  $\tau_i$ .

## 5.2 Static Mixed Criticality (SMC)

In the SMC task model [1], each task has a criticality level and it may have a different WCET estimate for each level less than or equal to its criticality level. The conservativeness of those estimates increases with the criticality level. When the job of a task exceeds the WCET estimate corresponding to its own level, it is terminated. With just two levels, this ensures that the interference of any job of a low-criticality task does not exceed  $C_i^L$ .

Thus, assuming only two criticality levels, the WCRT recurrence for a single-frame low-criticality task under SMC is

$$R_i = C_i^L + \sum_{\tau_j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j^L \quad (4)$$

and for a high-criticality task it is

$$R_i = C_i^H + \sum_{\tau_j \in hpL(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j^L + \sum_{\tau_k \in hpH(i)} \left\lceil \frac{R_i}{T_k} \right\rceil C_k^H \quad (5)$$

where  $hpL(i)$  and  $hpH(i)$  are the sets of higher-priority low- and high-criticality tasks, respectively, for  $\tau_i$ .

### 5.3 Adaptive Mixed Criticality (AMC)

Baruah, Burns and Davis [2] proposed a fixed-priority uniprocessor adaptive mixed-criticality scheduling algorithm (AMC). In the proposed model, the system operates in two different modes. During the default L-mode, tasks execute with their L-WCET estimates. If a task tries to execute for more than its L-WCET estimate, a mode switch occurs. Thereafter, only high-criticality tasks are allowed to execute, for up to their corresponding H-WCET estimates, while all the low-criticality tasks are discarded. In [2], two sufficient schedulability analyses (AMC-rtb and AMC-max) are devised. Both analyses verify the schedulability of all tasks in L-mode, and the schedulability of H-tasks in H-mode and when they are affected by a mode switch, under the AMC model. In this section, we briefly review the latter, because the analyses in L-mode and steady-state H-mode are essentially standard fixed-priority response time analysis.

#### 5.3.1 AMC-rtb

In this analysis, the WCRT recurrence of a job of task  $\tau_i$  that is affected by a mode switch is:

$$R_i^* = C_i^H + \sum_{\tau_j \in hpH(i)} \left\lceil \frac{R_i^*}{T_j} \right\rceil C_j^H + \sum_{\tau_k \in hpL(i)} \left\lceil \frac{R_i^L}{T_k} \right\rceil C_k^L \quad (6)$$

This analysis makes two conservative assumptions. First, that the number of L-jobs for each interfering L-task is maximum:  $R_i^L$  is the latest time the mode switch may occur. Second, that all jobs of each interfering H-task take their H-WCET,  $C_j^H$ . However, these two assumptions cannot simultaneously hold. The big advantage of this analysis is that the calculation of  $R_i^*$  is independent of the instant when the mode switch occurs.

#### 5.3.2 AMC-max

This analysis reduces the pessimism by taking into account the instant  $s$ , relative to the release of the job under analysis, at which the mode switch occurs. This allows for a more accurate accounting of the number of jobs of an interfering L-task  $\tau_j$ , and therefore of its interference:

$$IL_j(s) = \left( \left\lceil \frac{s}{T_j} \right\rceil + 1 \right) C_j^L \quad (7)$$

As explained in [2], Equation (7) assumes  $\lfloor \frac{s}{T_j} \rfloor + 1$  interfering jobs of  $\tau_j$  (instead of  $\lceil \frac{s}{T_j} \rceil$ ) in order to accommodate the possibility of any L-jobs released up to the mode switch instant (inclusive) being allowed to execute to completion<sup>2</sup>.

By taking into account  $s$ , it is also possible to be less conservative in the estimate of the number of jobs of a H-task  $\tau_k$  that complete after the mode switch.

$$M(k, s, t) = \min \left\{ \left\lceil \frac{t - s - (T_k - D_k)}{T_k} \right\rceil + 1, \left\lceil \frac{t}{T_k} \right\rceil \right\} \quad (8)$$

The number of jobs of a H-task that complete before a mode switch is obtained by subtracting this value from an upper bound on the total number of jobs of task  $\tau_k$  that may be released in time interval  $t$ , i.e.  $\lceil t/T_k \rceil$ . This allows the interference of H-task  $\tau_k$  to be safely upper-bounded because  $C_k^L \leq C_k^H$ . Therefore, the interference of H-task  $\tau_k$  in a time interval of duration  $t$ , when the mode switch occurs  $s$  time units after the beginning of that interval, is given by:

$$IH_k(s, t) = M(k, s, t) \cdot C_k^H + \left( \left\lceil \frac{t}{T_k} \right\rceil - M(k, s, t) \right) C_k^L \quad (9)$$

Thus, the WCRT recurrence used to compute the response time of a job of a H-task  $\tau_i$  that is affected by a mode switch is:

$$R_i^{(n)}(s) = C_i^H + \sum_{\tau_j \in hpL(i)} IL_j(s) + \sum_{\tau_k \in hpH(i)} IH_k(s, R_i^{(n-1)}(s)) \quad (10)$$

The solution of this recurrence,  $R_i(s)$  depends on the mode switch instant,  $s$ . Thus, the response time of  $\tau_i$  when affected by a mode switch is given by:

$$R_i^* = \max\{R_i(s) : 0 < s \leq R_i^L\}$$

However, Baruah, Burns and Davis [2] argue that it is enough to compute  $R_i(s)$  only for values of  $s$  that correspond to the release of jobs of higher-priority L-tasks, when the first job of all these tasks is released at the same time as the job of the task under analysis and these tasks' jobs arrive as soon as possible. This is because  $IH_k$ , as given by (9), is non-increasing as  $s$  increases and  $IL_j$ , as given by (7), is a step function whose value increases when jobs of task  $\tau_j$  are released.

---

<sup>2</sup> We note that, if such jobs are instead immediately discarded at the mode switch, then assuming  $\lceil \frac{s}{T_j} \rceil$  interfering jobs is safe. In [22], when discussing the original AMC-max, we inadvertently used the expression with  $\lfloor \frac{s}{T_j} \rfloor$  for  $IL_j(s)$ , instead of Equation (7). That typo did not propagate to the experiments in that paper.

## 6 Multiframe mixed-criticality scheduling with constrained-deadline tasks

### 6.1 Analysis for static multiframe mixed-criticality systems (SMMC)

In this section, we extend the SMC analysis to the multiframe model. Like in the multiframe model, there is a single mode of operation in SMC. However tasks may have different levels of criticality and different WCET-estimates, one per criticality-level less than or equal to its own, see Section 4.1. Therefore, depending on the level of criticality of the task under analysis, different WCET-estimates are used in the WCRT-recurrence, see (4) and (5). For example, the WCRT of a H-task, (5), uses the L-WCET estimate to compute the interference by L-tasks and the H-WCET estimate to compute the interference by H-tasks. Therefore, we need to define additional  $g/G$  functions that take into account the criticality level of the WCET estimate.

Let  $g^L(\tau_i, k)$  be the cumulative L-WCET of any sequence of  $k$  jobs of task  $\tau_i$ . This function is defined as [18]'s  $g$  function, see (1), except that for job  $j$  it uses its L-WCET,  $C_{i,j}^L$  rather than its criticality-oblivious WCET,  $C_i$ :

$$g^L(\tau_i, k) = \begin{cases} 0 & \text{if } k = 0 \\ \max \left\{ \sum_{\ell=j}^{j+k-1} C_{i,(\ell \bmod F_i)}^L : 0 \leq j < F_i \right\} & \text{if } 1 \leq k \leq F_i \\ q \cdot g^L(\tau_i, F_i) + g^L(\tau_i, r) & \text{otherwise} \\ \text{where } q = k \operatorname{div} F_i \text{ and } r = k \operatorname{mod} F_i & \end{cases} \quad (11)$$

Analogously, we define  $g^H(\tau_i, k)$ , the cumulative H-WCET of any sequence of  $k$  jobs of task  $\tau_i$ . Furthermore, we define the corresponding  $G$  functions, see (2):

$$G^L(\tau_i, t) = g^L \left( \tau_i, \left\lceil \frac{t}{T_i} \right\rceil \right) \quad (12)$$

$$G^H(\tau_i, t) = g^H \left( \tau_i, \left\lceil \frac{t}{T_i} \right\rceil \right) \quad (13)$$

With these definitions, it is straightforward to generalize the SMC's WCRT recurrences to the multiframe model. For L-tasks, the recurrence becomes similar to that for the single-criticality task model (3):

$$R_i = g^L(\tau_i, 1) + \sum_{\tau_j \in hp(i)} G^L(\tau_j, R_i) \quad (14)$$

Indeed, like in the original multiframe analysis [18], it suffices to compute the response time for the frame with largest L-WCET,  $g^L(\tau_i, 1)$ . Furthermore, each of the terms in the summation,  $G^L(\tau_j, R_i)$ , is an upper-bound of the interference by jobs of higher-priority task  $\tau_j$  in the response time  $R_i$  of the task under analysis,  $\tau_i$ . This is because it represents the cumulative L-WCET,

i.e., computed with same level of confidence as that of  $g^L(\tau_i, 1)$ , of any sequence of  $\tau_j$ 's jobs in time interval,  $R_i$ .

However, for H-tasks, the WCRT-recurrence must use the  $g/G$  functions corresponding to the criticality level of each task:

$$R_i = g^H(\tau_i, 1) + \sum_{\tau_j \in hpL(i)} G^L(\tau_j, R_i) + \sum_{\tau_k \in hpH(i)} G^H(\tau_k, R_i) \quad (15)$$

Again, it is enough to compute the response time of the job with the largest H-WCET,  $g^H(\tau_i, 1)$ . The interference by higher-priority H-task  $\tau_k$  is bound by  $G^H(\tau_k, R_i)$ , the cumulative H-WCET of a sequence of jobs of  $\tau_k$  in the response time interval,  $R_i$ , of the task under analysis,  $\tau_i$ . The interference by higher-priority L-task  $\tau_j$  is bound by  $G^L(\tau_j, R_i)$ , because in SMC the system terminates a job of an L-task as soon as its execution time exceeds the respective L-WCET estimate.

## 6.2 Analysis for adaptive multiframe mixed criticality systems (AMMC)

In this section, we develop a response time analysis for the adaptive mode-based variant of the multiframe mixed-criticality task model (as presented in Section 4.2). The idea is straightforward; we use AMC to count the number of jobs of each interfering task, and use Multiframe  $g/G$  functions, or similar functions, to compute the cumulative WCET of these jobs and therefore the WCRT of the different tasks.

As usual, we need to check the schedulability in:

L-mode: This can be done for all tasks using Baruah's WCRT recurrence, see (3), and replacing  $g$  and  $G$  functions with  $g^L$  and  $G^L$ , respectively.

H-mode: Again, we can reuse Baruah's WCRT recurrence and replace  $g$  and  $G$  functions with  $g^H$  and  $G^H$ , respectively, for all H-tasks.

Mode switch: In this case, we must consider, for each H-task, the response time of each job in one of its superframes when it is caught by a mode switch, i.e. when it is released before the mode switch but completes only after.

We now focus on the response time of jobs caught by a mode switch. Whereas in the single frame model, all jobs have the same parameters, in the multiframe model, different jobs have different parameters. Therefore, for each task under analysis,  $\tau_i$ , we need to:

1. analyse the WCRT of each job in a superframe of  $\tau_i$ ;
2. consider the phasing of the interfering tasks, i.e., for each higher-priority task, what is the first job in the sequence of interfering jobs of that task.

Figure 1 illustrates the need for the latter. With respect to (1), consider two different jobs  $j$  and  $k$  ( $j \neq k$ ) of a superframe of  $\tau_i$ , it may be the case that  $C_{i,j}^L > C_{i,k}^L$  and  $C_{i,j}^H < C_{i,k}^H$ . Thus, the schedulability of job  $j$  cannot be inferred from the response time of job  $k$ , nor vice-versa. Of course, it may not

be necessary to compute the response time of all jobs. E.g., if  $C_{i,j}^L > C_{i,k}^L$  and  $C_{i,j}^H \geq C_{i,k}^H$ , clearly it is enough to analyze the response time of job  $j$  of  $\tau_i$ . Task  $\tau_i$  is schedulable, if the WCRT of all jobs in one of its superframes does not exceed  $D_i$ .

In each of the following subsections, we describe how to generalize each of the AMC analyses for constrained-deadline tasks to the multiframe model.

### 6.2.1 AMMC-rtb: AMC-rtb to multiframe tasks

As summarized in Section 5.3.1, AMC-rtb considers that each higher-priority L-task may interfere up to  $R_i^L$ . In the multiframe model, for each higher-priority L-task  $\tau_k$ , we need to consider the cumulative WCET of a sequence of  $\tau_k$ 's jobs. Thus, the interference of a higher-priority L-task,  $\tau_k$ , on job  $j$  of H-task  $\tau_i$  is given by:

$$G^L(\tau_k, R_{i,j}^L)$$

where  $R_{i,j}^L$  is the WCRT in L-mode of job  $j$  of task  $\tau_i$ . For each higher-priority H-task,  $\tau_k$ , AMC-rtb assumes that every job of  $\tau_k$  executes for its H-mode WCET (H-WCET for short) over the entire WCRT of task  $\tau_i$ 's job  $j$  when it is caught by a mode switch,  $R_{i,j}^*$ . Thus the interference of  $\tau_k$ , on that job is given by:

$$G^H(\tau_k, R_{i,j}^*)$$

Accordingly, the WCRT of job  $j$  of a H-task  $\tau_i$  released before the mode switch, but completing after it, is upper-bounded by:

$$R_{i,j}^* = C_{i,j}^H + \sum_{k \in hpL(i)} G^L(\tau_k, R_{i,j}^L) + \sum_{k \in hpH(i)} G^H(\tau_k, R_{i,j}^*) \quad (16)$$

The WCRT of H-task  $\tau_i$  is therefore the maximum of the WCRT of all frames in a superframe of  $\tau_i$ :

$$R_i^* = \max\{R_{i,j}^* : 0 \leq j < F_i\} \quad (17)$$

We can trade-off the computation cost of  $R_i^*$  for its pessimism, by bounding the mode switch instant to  $R_i^L$  rather than to  $R_{i,j}^L$ , where

$$R_i^L = \max\{R_{i,j}^L : 0 \leq j < F_i\} \quad (18)$$

In this case, it suffices to compute  $R_{i,j}^*$  for the job  $j$  with the maximum  $C_{i,j}^H$ , therefore the WCRT of H-task  $\tau_i$  released before the mode switch but, completing after it, can be upper bounded by:

$$R_i^* = g^H(\tau_i, 1) + \sum_{j \in hpL(i)} G^L(\tau_j, R_i^L) + \sum_{k \in hpH(i)} G^H(\tau_k, R_i^*) \quad (19)$$

### 6.2.2 AMMC-max: AMC-max to multiframe tasks

AMC-max reduces the pessimism in AMC-rtb by explicitly considering the mode switch instant,  $s$ , as summarized in Section 5.3.2. In particular, rather than assuming that all jobs of a H-task complete after the mode switch, AMC-max provides a tighter upper-bound on the number of jobs of a H-task that complete after the mode switch. As a consequence, AMC-max may underestimate the number of jobs of that task that complete before the mode switch. (This is also done in AMC-rtb, where the number of jobs of a H-task that complete before the mode switch is assumed to be 0.) Thus, critical for the safety of AMC-max is that the cumulative WCET of a task in a time interval comprising the mode switch instant does not decrease, if the number of jobs of that task that complete after the mode switch is overestimated.

To safely apply AMC-max's accounting of H-jobs that complete after the mode switch to the multiframe model, the following must hold:

**Lemma 1** *Consider a sequence of successive jobs by a multiframe task  $\tau_i$  with a fixed length  $n$  such that the last  $n^H < n$  of these jobs terminate after the mode switch. Let  $n'^H$  ( $n \geq n'^H \geq n^H$ ) be an overestimation of  $n^H$ . Then assuming the termination of the last  $n'^H$  jobs after the mode switch does not decrease the cumulative WCET.*

*Proof* The proof is by induction on the number of jobs that terminate after the mode switch.

*Base step:*  $n'^H = n^H$  and therefore the two job sequences are the same and so are their cumulative WCET.

*Induction step* Consider a sequence of these  $n$  jobs such that the last  $h$  ( $n > h \geq n^H$ ) terminate after the mode switch. Let  $j$  be the last job in that sequence that terminates before the mode switch (such a job exists because  $h < n$ ). If we assume that job  $j$  terminates after the mode switch, the difference between the cumulative WCET of the new and the previous sequences is  $C_{i,j}^H - C_{i,j}^L$ . Because by assumption  $C_{i,j}^H \geq C_{i,j}^L$  this difference is non-negative, and therefore, by the induction step assumption, the cumulative WCET is not smaller than that of the sequence with  $n^H$  jobs terminating after the mode switch.

Thus, for multiframe tasks (just as for single-frame tasks), it is safe for the analysis to overestimate, in a given sequence of jobs by an interfering task, the number of jobs that execute for their H-WCET. In particular, this means that we can use AMC-max's upper-bound on the number of jobs that complete after the mode switch, (8), and use it to derive the number of jobs that complete before the mode switch to be used in the WCRT recurrence.

The challenge is that the phasing of the interfering task affects the amount of interference. The difference with respect to Baruah's multiframe analysis is that now there may be two subsequences, one before the mode switch and another after the mode switch. Figure 2 illustrates this. It shows an interfering

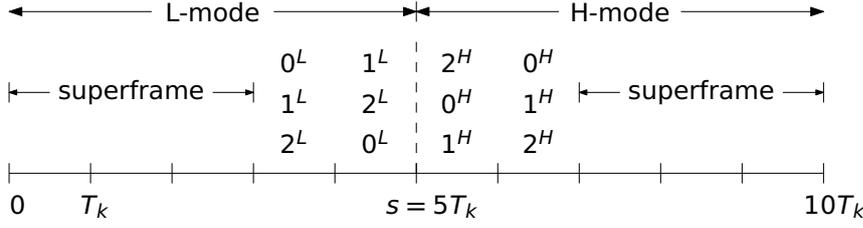


Fig. 2: Interference of a H-task depends on the phasing of its jobs.

task  $\tau_k$  with 3 frames. The time interval under consideration is 10 times  $T_k$  and we assume that the mode switch occurs exactly at the middle of that interval. Thus, in this interval, there are 10 releases of jobs of  $\tau_k$ , 5 before the mode switch and another 5 after the mode switch. Hence, there is a superframe plus 2 frames, both before and after the mode switch. The cumulative WCET of a super-frame is independent of the initial job in the superframe. Therefore, we need only to find the worst-case cumulative WCET of a sequence of 4 frames, such that the two first frames take their L-WCET, whereas the other 2 frames take their H-WCET. One of the following frame sequences leads to the worst-case cumulative execution time:  $(0^L, 1^L, 2^H, 0^H)$ ,  $(1^L, 2^L, 0^H, 1^H)$ ,  $(2^L, 0^L, 1^H, 2^H)$ , where we denote by  $i^L$  ( $i^H$ ) a frame whose L-WCET (H-WCET) is equal to the L-WCET (H-WCET) of frame  $i$ .

Formally, we define function  $g^*(\tau_i, \ell^L, \ell^H)$  that computes the cumulative WCET of a sequence of jobs of H-task  $\tau_i$ , which is composed by a subsequence of  $\ell^L$  jobs that take their L-WCET followed by a subsequence of  $\ell^H$  that take their H-WCET:

$$g^*(\tau_i, \ell^L, \ell^H) = \begin{cases} g^H(\tau_i, \ell^H) & \text{if } \ell^L = 0 \\ g^L(\tau_i, \ell^L) & \text{if } \ell^L \neq 0 \wedge \ell^H = 0 \\ \max \left\{ \begin{array}{l} \sum_{k=j}^{j+\ell^L-1} C_{i,(k \bmod F_i)}^L \\ + \sum_{k=j+\ell^L}^{j+\ell^L+\ell^H-1} C_{i,(k \bmod F_i)}^H : 0 \leq j < F_i \end{array} \right\} & \text{if } 1 \leq \ell^L < F_i \wedge 1 \leq \ell^H < F_i \\ q^L \cdot g^L(\tau_i, F_i) + g^*(\tau_i, r^L, r^H) + q^H \cdot g^H(\tau_i, F_i) & \text{otherwise} \end{cases} \quad (20)$$

where  $q^L = (\ell^L \text{ div } F_i)$ ,  $r^L = (\ell^L \bmod F_i)$ ,  $q^H = (\ell^H \text{ div } F_i)$  and  $r^H = (\ell^H \bmod F_i)$ .

Furthermore, we define:

$$G^{L+}(\tau_i, t) = g^L \left( \tau_i, \left\lfloor \frac{t}{T_i} \right\rfloor + 1 \right) \quad (21)$$

to account for the fact that AMC-max upper-bounds the number of jobs of higher-priority L-task  $\tau_i$  by  $\left\lceil \frac{s}{T_i} \right\rceil + 1$  rather than  $\left\lceil \frac{s}{T_i} \right\rceil$ , where  $s$  is the mode switch instant relative to the release of the job under analysis<sup>3</sup>.

Thus the WCRT of job  $j$  of H-task  $\tau_i$  is upper bounded by:

$$R_{i,j}^*(s) = C_{i,j}^H + \sum_{k \in hpL(i)} G^{L+}(\tau_k, s) \quad (22)$$

$$+ \sum_{k \in hpH(i)} g^*(\tau_k, \ell^L(k, R_{i,j}^*(s), s), \ell^H(k, R_{i,j}^*(s), s))$$

where:

$$\ell^L(k, t, s) = \left\lceil \frac{t}{T_k} \right\rceil - \ell^H(k, t, s) \quad (23)$$

$$\ell^H(k, t, s) = M(k, s, t) \quad (24)$$

Therefore:

$$R_{i,j}^* = \max\{R_{i,j}^*(s) : 0 < s \leq R_{i,j}^L\}$$

Like in AMC-max, we need to compute the WCRT of job  $j$  of task  $\tau_i$  only for the values of  $s$  that correspond to the release of jobs of higher-priority L-tasks, when the first job of each of these tasks is released at the same time as the job under analysis and the other jobs arrive as soon as possible.

Like in AMMC-rtb, to compute the WCRT of task  $\tau_i$ , we need to compute the response time of all jobs in a superframe and take the maximum of these values:

$$R_i^* = \max\{R_{i,j}^* : 0 \leq j < F_i\}$$

Again, we can trade-off computation cost for pessimism, by always considering all the  $s$  values smaller than  $R_i^L$ , see (18), i.e.:

$$R_i^* = \max\{R_i^*(s) : 0 < s \leq R_i^L\} \quad (25)$$

In this case, rather than computing the WCRT for all the jobs in a superframe it suffices to compute the WCRT for the longest H-mode job in a superframe. Thus (22) becomes:

$$R_i^*(s) = g^H(\tau_i, 1) + \sum_{j \in hpL(i)} G^{L+}(\tau_j, s) \quad (26)$$

$$+ \sum_{k \in hpH(i)} g^*(\tau_k, \ell^L(k, R_i^*(s), s), \ell^H(k, R_i^*(s), s))$$

where  $\ell^L$  and  $\ell^H$  are given by (23) and (24), respectively.

<sup>3</sup> In Equation (18) of [22] (which corresponds to Equation (22)), the regular  $G^L$  function was inadvertently used, instead of  $G^{L+}$ , to upper-bound the number of such jobs. This is still safe, as long as all jobs by low-criticality tasks are terminated immediately at mode switch, but unsafe, if they are allowed to continue executing to completion for up to their L-WCET. In any case, the typo did not propagate to the experiments in [22], which were in fact consistent with Equation (22).

### 6.2.3 Implementation of the $g^*$ function

For an efficient implementation of the analyses described in the previous section, it is important to be able to efficiently implement the  $g^*$  function.

An efficient implementation of the  $g$  function is given in [18] that relies on an array  $M_k$  of length  $F_k + 1$  per task  $\tau_k$ , such that  $M_k[i] = g(\tau_k, i)$ , for  $0 \leq i \leq F_k$ . This array is computed before initiating the response time analysis, and the complexity of its computation is  $\mathcal{O}(F_k^2)$ . Analogously, in order to implement the  $g^*$  function, we define a 2-dimensional  $(F_k + 1) \times (F_k + 1)$  matrix,  $M_k$  per H-task  $\tau_k$ , where  $M_k[i][j] = g^*(\tau_k, i, j)$  for  $0 \leq i \leq F_k$  and  $0 \leq j \leq F_k$ . Note that element  $M_k[0][j] = g^H(\tau_k, j)$ , and therefore  $M_k[0][F_k]$  is the H-WCET of a superframe of  $\tau_k$ . Likewise, element  $M_k[i][0] = g^L(\tau_k, i)$ , and therefore  $M_k[F_k][0]$  is the L-WCET of a superframe of  $\tau_k$ . Furthermore, all other elements of the last column and of the last line are not needed and therefore need not be computed. Indeed, the value of  $g^*(\tau_k, \ell, h)$  can be efficiently computed from the elements of the  $M_k$  matrix, as follows:

$$(\ell \operatorname{div} F_k) \cdot M_k[F_k][0] + M_k[\ell \bmod F_k][h \bmod F_k] + (h \operatorname{div} F_k) \cdot M_k[0][F_k]$$

The computation of the matrix  $M_k$  is very similar to the computation of array  $M_k$  in [18]. The first step is to compute two  $F_k \times F_k$  matrices  $L_k$  and  $H_k$ . These matrices are just like  $F_k \times F_k$  matrix  $D_k$  in [18], except that  $L_k$  and  $H_k$  use the L-WCET and H-WCET of  $\tau_k$ 's frames, respectively. More specifically, element  $L_k[i][j] = \sum_{\ell=i}^{i+j} C_{k,(\ell \bmod F_k)}^L$ , i.e. element  $L_k[i][j]$  is the cumulative WCET of a sequence of  $(i+1)$  L-frames that starts with frame  $j$ . Algorithm 1 shows the code segment presented in [18], adapted to initialize the elements of matrix  $L_k$ . Except for the notation, the only difference is the use of the L-WCET in Lines 2 and 5. The initialization of  $H_k$  is identical. The complexity of this code segment is  $\mathcal{O}(F_k^2)$ .

---

#### Algorithm 1 Computation of auxiliary matrix $L_k$ .

---

//  $L_k$  is a  $F_k \times F_k$  matrix, such that  $L_k[i][j] = \sum_{\ell=i}^{i+j} C_{k,(\ell \bmod F_k)}^L$

```

1: for ( $j:=0; j < F_k; j:=j+1$ ) // First line
2:    $L_k[0][j] := C_{k,j}^L$ ;
3: for ( $i:=1; i < F_k; i:=i+1$ )
4:   for ( $j:=0; j < F_k; j:=j+1$ )
5:      $L_k[i][j] := L_k[i-1][j] + C_{i,(i+j) \bmod F_k}^L$ ;

```

---

Algorithm 2 shows the code segment used for computing matrix  $M_k$  from matrices  $L_k$  and  $H_k$ . For the sake of readability, we use the  $\max(m, n)$  function, which returns the maximum of its two integer arguments. The algorithm uses the elements of  $L_k$  and  $H_k$  in a very similar way to the computation of the corresponding array in [18] from the elements of the  $D_k$  matrix. As a result, the complexity for computing each element of the  $M_k$  matrix,  $\mathcal{O}(F_k)$ , is the same as the complexity for computing each element of the array in [18]. However

because the  $M_k$  matrix has  $(F_k + 1)^2$  elements rather than  $(F_k + 1)$  elements, the complexity of the computation of matrix  $M_k$  is  $\mathcal{O}(F_k^3)$  rather than  $\mathcal{O}(F_k^2)$ . Just like in Algorithm 1, note that all elements of the last row except for the first one are not computed, as they are not needed. The same for all elements of the last column except the first one.

---

**Algorithm 2** Computation of  $(F_k + 1) \times (F_k + 1)$  matrix  $M_k$ .  $M_k[i][j]$  is the cumulative WCET of any sequence of  $i$  L-jobs followed by  $j$  H-jobs of task  $\tau_k$

---

```

1:  $M_k[0][0] := 0$ 
   // First row is  $g^H$ : this is the algorithm in [18]
2: for ( $h:=1; h \leq F_k; h:=h+1$ ) { //  $h$  is the length of the sequence
3:    $M_k[0][h] := 0;$ 
4:   for ( $i:=0; i < F_k; i:=i+1$ ) //  $i$  is the first frame
5:      $M_k[0][h] := \max(M_k[0][h], H_k[h-1][i]);$ 
6: }
   // First column is  $g^L$ : this is the algorithm in [18]
7: for ( $\ell:=1; \ell \leq F_k; \ell:=\ell+1$ ) { //  $\ell$  is the length of the sequence
8:    $M_k[\ell][0] := 0;$ 
9:   for ( $i:=0; i < F_k; i:=i+1$ ) //  $i$  is the first frame
10:     $M_k[\ell][0] := \max(M_k[\ell][0], L_k[\ell-1][i]);$ 
11: }
   // Remaining elements: based on the algorithm in [18]
12: for ( $\ell:=1; \ell < F_k; \ell:=\ell+1$ ) //  $\ell$  is the length of the L-sequence
13:   for ( $h:=1; h < F_k; h:=h+1$ ) { //  $h$ : H-sequence's length
14:      $M_k[\ell][h] := 0;$ 
15:     for ( $i:=0; i < F_k; i:=i+1$ ) //  $i$ : first frame in L-sequence
16:        $M_k[\ell][h] := \max(M_k[\ell][h],$ 
17:          $L_k[\ell-1][i] + H_k[h-1][(i+\ell) \bmod F_k]);$ 

```

---

## 7 Background on the schedulability analysis of arbitrary-deadline tasks

So far (From sections 6.1 to 6.2, as also published in [22]), we have formulated analysis for multiframe mixed-criticality systems under the assumption that task deadlines are constrained. We now extend those derivations for a more general *arbitrary-deadline* model, whereby it does not necessarily hold that  $D_i \leq T_i$  for a task  $\tau_i$ . The previously formulated analysis no longer holds in that case, because a job by a task with  $D_i > T_i$  may additionally suffer interference from earlier-released jobs *of the same task*. To address this, we draw from the work by Tindell et al. [23] on the fixed-priority preemptive scheduling of arbitrary-deadline single-frame tasks.

Under that approach [23], the schedulability of a task  $\tau_i$  with arbitrary deadline ( $D_i > T_i$ ) can be computed using the concept of level- $i$  busy period [24], i.e., a time interval while the processor is busy executing jobs of tasks with priority equal or higher than that of  $\tau_i$ . Indeed, Lehoczky proved that the longest response time for a job of a such a task  $\tau_i$  occurs during a

level- $i$  busy period initiated by a critical instant, i.e., a time instant when all tasks with priority equal or higher than  $\tau_i$  are released. However, unlike for constrained deadline tasks, the worst-case response time may not occur for the first job of  $\tau_i$  after the critical instant. Therefore, to determine whether  $\tau_i$  is schedulable, one needs to check that the response time of all jobs of  $\tau_i$  in a level- $i$  busy period (that starts at a critical instant) does not exceed  $D_i$ .

### 7.1 Analysis for FPPS for single criticality tasks with arbitrary deadlines

To compute the response time of job  $q$  in a level- $i$  busy period, i.e., of a job that starts  $q \cdot T_i$  time units after the beginning of the busy period, Tindell [23] extended standard worst-case response time analysis for constrained deadline tasks [3] to compute the completion time of job  $q + 1$  with respect to the beginning of the level- $i$  busy period:

$$r_i(q) = (q + 1)C_i + \sum_{j \in hp(i)} \left\lceil \frac{r_i(q)}{T_j} \right\rceil C_j \quad (27)$$

Indeed, the left hand side of (27) is the cumulative processor requirement by all jobs of tasks with higher priority than  $\tau_i$  and of the first  $q + 1$  jobs after the critical instant. Because the last of these jobs is released  $q \cdot T_i$  time units after the beginning of the level- $i$  busy period, its response time is given by:

$$R_i(q) = r_i(q) - qT_i \quad (28)$$

As mentioned above, a task  $\tau_i$  is schedulable if  $R_i(q) \leq D_i$ , for all jobs  $q$  in a level- $i$  busy period. Note that we need not know *a priori* how many jobs there are in a level- $i$  busy period. The computation of  $r_i(q)$  can start for  $q = 0$  and stop when either  $R_i(q) > D_i$ , in which case  $\tau_i$  is not schedulable, or when  $r_i(q) \leq (q + 1) \cdot T_i$ , i.e., at the last job of a level- $i$  busy period, in which case  $\tau_i$  is schedulable.

### 7.2 Mixed-criticality scheduling with arbitrary deadline tasks

Burns and Davis [12] extended SMC, AMC-rtb and AMC-max for task sets with constrained deadlines to task sets with arbitrary deadlines. Actually, SMC's extension is straightforward. To compute  $r_i(q)$ , (27), instead of using  $C_i$  and  $C_j$ , one should use  $C_i^{L_i}$  and  $C_j^{\min(L_i, L_j)}$ , where  $L_i$  and  $L_j$  are the criticality levels of tasks  $\tau_i$  and  $\tau_j$ , respectively. I.e. if the task under analysis is an L-task, then the worst-case completion time of its job  $q$ ,  $r_i(q)$ , is computed using the L-WCET estimates of higher-priority tasks. If the task under analysis is a H-task, then  $r_i(q)$  is computed using the L-WCET for higher-priority L-tasks and H-WCET for higher-priority H-tasks.

### 7.2.1 AMC-rtb-Arb analysis

In AMC, the analysis must ensure schedulability in both L- and H-mode as well as upon a mode switch.

The completion time of a job in L-mode of a task  $\tau_i$  in a level- $i$  busy period released  $qT_i$  time units after the beginning of that busy period can be adapted from (27) by using the appropriate WCET for the different jobs:

$$r_i^L(q) = (q+1)C_i^L + \sum_{j \in hp(i)} \left\lceil \frac{r_i^L(q)}{T_j} \right\rceil C_j^L \quad (29)$$

and:

$$R_i^L(q) = r_i^L(q) - qT_i \quad (30)$$

Task  $\tau_i$  is schedulable in L-mode, if  $R_i^L(q) < D_i$  for all jobs  $q$  in a level- $i$  busy period in L-mode.

Likewise, we can obtain an expression for the steady H-mode. However, the steady H-mode is dominated by H-mode upon a mode switch.

For AMC-rtb-Arb, the completion time of a job of a H-task  $\tau_i$  affected by a mode switch is given by [12]:

$$r_i^H(q) = (q+1)C_i^H + \sum_{j \in hpH(i)} \left\lceil \frac{r_i^H(q)}{T_j} \right\rceil C_j^H + \sum_{k \in hpL(i)} \left\lceil \frac{r_i^L(\min(q,p))}{T_k} \right\rceil C_k^L \quad (31)$$

where  $p$  is the last job number in a level- $i$  busy period in L-mode.

The main issue in the analysis upon a mode switch is to bound the duration of the L-mode interval before the mode switch. Although,  $r_i^L(q)$  appears to be a straightforward choice, the level- $i$  busy period upon a mode switch may be longer than in L-mode. If  $\tau_i$  is schedulable in L-mode, then the level- $i$  busy period must be bounded, and there is a last job from  $\tau_i$  in that period, job  $p$ . Clearly, the mode switch must occur before the completion time of that job, otherwise that job would not have been affected by the mode switch, nor any job that follows it. Therefore, (31) uses  $r_i^L(\min(p,q))$  rather than  $r_i^L(q)$ .

As before:

$$R_i^H(q) = r_i^H(q) - qT_i \quad (32)$$

and H-task  $\tau_i$  is schedulable in H-mode (including mode switch), if  $R_i^H(q) < D_i$  for all jobs  $q$  in a level- $i$  busy period that comprises a mode switch.

### 7.2.2 AMC-max-Arb analysis

AMC-max-Arb differs from AMC-rtb-Arb only on the completion time expression for jobs of a H-task  $\tau_i$  that are affected by a mode switch. Assuming the mode switch occurs at time  $s$  (measured with respect to the beginning of the level- $i$  busy period), the completion time of job  $q$  of such a task is given by [12]:

$$r_i(q, s) = X(q, s, r_i(q, s))C_i^H + Y(q, s, r_i(q, s))C_i^L + IL(s) + IH(s, r_i(q, s)) \quad (33)$$

where  $X(q, s, t)$  and  $Y(q, s, t)$  are the number of the first  $q + 1$  jobs, i.e. until job  $q$ , of  $\tau_i$  that complete until instant  $t$  after and before the mode switch, respectively. Thus  $X(q, s, t) + Y(q, s, t) = q + 1$ . AMC-max-Arb conservatively (over-)estimates  $X(q, s, t)$  as follows:

$$X(q, s, t) = \min \left( \left\lceil \frac{t - s + (D_i - T_i)}{T_i} \right\rceil + 1, q + 1 \right) \quad (34)$$

and derives  $Y(q, s, t)$  from  $X(q, s, t) + Y(q, s, t) = q + 1$ .

$IL(s)$  and  $IH(s, t)$  in (33) are given by the same summations as in AMC-max (for constrained deadline task sets), respectively (7) and (9).

Similarly to AMC-max,  $r_i(q, s)$  must be computed for every value of  $s \in [0, r_i^L(q)]$  equal to the release time of L-tasks with higher priority than  $\tau_i$ . Let,  $r_i^*(q)$  be the maximum of all these values. Then, the worst-case response time for job  $q$  in a level- $i$  busy period that comprises a mode switch can be computed as:

$$R_i^*(q) = r_i^*(q) - qT_i \quad (35)$$

and task  $\tau_i$  is schedulable in H-mode (including mode switch), if  $R_i^*(q) < D_i$  for all jobs  $q$  in a level- $i$  busy period that comprises the mode switch.

## 8 Generalisation of multiframe mixed-criticality scheduling analyses for arbitrary-deadline tasks

We now formulate the generalized schedulability analyses techniques (SMMC-Arb, AMMC-rtb-Arb and AMMC-max-Arb) for arbitrary-deadline multiframe mixed-criticality tasks.

### 8.1 Static multiframe mixed criticality with arbitrary deadline (SMMC-Arb)

In this section, we extend the static multiframe mixed-criticality analysis for the arbitrary-deadline task model.

**L-task analysis:** The response time of an L-task in SMMC (and in SMC) considers the L-WCETs of all tasks, independently of their criticality. Therefore, the worst-case completion time of job  $q$  of L-task  $\tau_i$  in a level- $i$  busy period can be computed as follows:

$$r_i(q) = g^L(\tau_i, (q + 1)) + \sum_{j \in hp(i)} G^L(\tau_j, r_i(q)) \quad (36)$$

Indeed,  $g^L(\tau_i, (q + 1))$  bounds the cumulative processing demand of any sequence of  $q + 1$  jobs, i.e. up to job  $q$ , of the task under analysis, using its own criticality level WCET-estimate. Furthermore, each of the terms in the summation,  $G^L(\tau_j, r_i(q))$ , is an upper bound of the interference of any sequence of jobs of higher-priority task  $\tau_j$  in time interval  $r_i(q)$ , using their L-mode WCET

estimate, i.e., computed with a confidence corresponding to the criticality level of the task under analysis.

Just as for SMC-Arb, the corresponding worst-case response time is:

$$R_i(q) = r_i(q) - qT_i \quad (37)$$

Also like in SMC-Arb, a multiframe L-task  $\tau_i$  is schedulable under SMMC-Arb, if the response time of all its jobs in a level- $i$  busy period does not exceed its deadline,  $D_i$ .

**H-task analysis:** In SMMC (and in SMC), the response time analysis for a H-task assumes that each task executes for the WCET estimate corresponding to its criticality level. Therefore, the worst-case completion time of job  $q$  of H-task  $\tau_i$  in a level- $i$  busy period can be computed as follows: Since all the tasks are allowed to execute at all times, the completion time of the task under analysis  $\tau_i$  can be computed as follows:

$$r_i(q) = g^H(\tau_i, q + 1) + \sum_{j \in hpL(i)} G^L(\tau_j, r_i(q)) + \sum_{k \in hpH(i)} G^H(\tau_k, r_i(q)) \quad (38)$$

Indeed, the first term on the right-hand side of this equation,  $g^H(\tau_i, q + 1)$  bounds the cumulative WCET of any sequence of  $q + 1$  jobs, i.e. up to job  $q$ , of the task under analysis, using their H-WCET estimates. Furthermore, each term in the first summation on the right-hand side,  $G^L(\tau_j, r_i(q))$ , bounds the interference of the largest sequence of jobs of higher-priority L-task  $\tau_j$  in a time interval of duration  $r_i(q)$ , using their L-WCET estimates. Finally, each term in the second summation on the right-hand side,  $G^H(\tau_k, r_i(q))$ , bounds the interference of the largest sequence of jobs of higher-priority H-task  $\tau_k$  in a time interval of duration  $r_i(q)$ , using their H-WCET estimates. Note that the two summations are similar to those used in the estimate of the worst-case response time for SMMC (with constrained deadlines) (15),  $R_i$ , except that time interval considered is  $r_i(q)$ , i.e. the completion time of job  $q$  of  $\tau_i$ , rather than  $R_i$ .

The worst-case response time for such a job is determined as usual for a task with an arbitrary deadline, i.e. using (37). Furthermore, like in SMC-Arb or for a multiframe L-task under SMMC-Arb, a multiframe H-task  $\tau_i$  is schedulable under SMMC-Arb, if the response time of all its jobs in a level- $i$  busy period does not exceed its deadline,  $D_i$ .

## 8.2 Analysis for adaptive multiframe mixed criticality systems (AMMC)

In this section, we extend the AMMC analysis for tasks with constrained deadlines in Section 6.2 to tasks with arbitrary deadlines, by combining those analyses with the analyses for mixed-criticality systems with arbitrary deadlines (AMC-rtb-Arb and AMC-max-Arb) in [12]. We refer to these two new analyses as AMMC-rtb-Arb and AMMC-max-Arb.

8.2.1 Adaptive multiframe mixed criticality-rtb with arbitrary deadline  
(AMMC-rtb-Arb)

In AMMC the system operates in two modes. Therefore, we need to analyse both L- and H-modes.

**L-mode analysis:** In L-mode, independently of a task's level, AMC, and therefore AMMC, uses only L-mode WCET estimates. Therefore, the completion time of job  $q$  of a multiframe task  $\tau_i$ , whether it is an L-task or a H-task, is given by:

$$r_i^L(q) = g^L(\tau_i, (q+1)) + \sum_{j \in hp(i)} G^L(\tau_j, r_i^L(q)) \quad (39)$$

The first term bounds the processing requirement of any sequence of  $q+1$  jobs of the task under analysis,  $\tau_i$ , using their L-WCET estimates. Also, each of the terms of the summation,  $G^L(\tau_j, r_i^L(q))$ , upper bounds the processing requirement in L-mode of any sequence of jobs of higher-priority task  $\tau_j$  over a time interval equal to the completion time in L-mode of any sequence of  $q+1$  jobs of  $\tau_i$ , the task under analysis.

**H-mode analysis:** AMC-rtb-Arb (and AMC-rtb) computes the worst-case response time of a job of a H-task  $\tau_i$  by bounding the number of jobs of each higher-priority task, both L- and H-tasks, and assuming that all jobs of H-tasks execute for their H-mode WCET. In AMMC-rtb-Arb we do the same, but we need to consider not only the number of jobs of each interfering task, see (31), but also their order. Therefore, the completion time of job  $q$  of a multiframe H-task  $\tau_i$  in a level- $i$  busy period comprising the mode switch instant is given by:

$$r_i^H(q) = g^H(\tau_i, (q+1)) + \sum_{j \in hpL(i)} G^L(\tau_j, r_i^L(\min(p, q))) + \sum_{k \in hpH(i)} G^H(\tau_k, r_i^H(q)) \quad (40)$$

where  $p$  is the last job of  $\tau_i$  in a level- $i$  busy period in L-mode operation. Indeed, the first term on (40)'s right-hand side upper-bounds the processing requirement of any sequence of  $q+1$  jobs of task  $\tau_i$ , assuming their H-WCET. Each term of the second summation, upper-bounds the interference from the largest sequence of jobs of higher-priority H-task  $\tau_k$  over  $r_i^H(q)$ , i.e., the worst-case completion time for job  $q$  of  $\tau_i$  in a level- $i$  busy period, assuming that each of these jobs takes their H-WCET. Finally, each term of the first summation, upper-bounds the interference from the largest sequence of jobs of higher-priority L-task  $\tau_j$  over the longest completion time of either job  $q$  or job  $p$  of  $\tau_i$ , whichever is smaller, assuming that each of these jobs takes their L-WCET. Indeed, as observed in [12], if  $p < q$ , the mode switch must occur before  $r_i^L(p)$ . Otherwise,  $\tau_i$  would have not been caught by a mode switch.

Note that in this analysis we are trading-off computation time for pessimism. Indeed,  $r_i^L(\min(p, q))$  is the maximum response time in L-mode for

any job sequence with the specified number of jobs, but this may occur for a sequence that starts with a job that is different from the job that leads to the maximum cumulative H-WCET of a sequence of  $q + 1$  jobs. This is similar to the trade-off described in Subsection 6.2.1. A tighter analysis would require to upper-bound the completion time of the  $F_i$  sequences of  $q + 1$  jobs, one per job in a superframe. For each sequence, the worst-case completion time would be computed using the cumulative H-WCET of the jobs in that sequence and also the respective L-mode completion time to upper-bound the interference by L-mode tasks. Finally, the response time would be computed by taking the maximum of these completion times and subtracting  $qT_i$ .

### 8.2.2 Adaptive multiframe mixed criticality-max with arbitrary deadlines (AMMC-max-Arb)

We now extend AMMC-max for arbitrary deadlines (AMMC-max-Arb). AMC-max and AMC-max-Arb differ from AMC-rtb and AMC-rtb-Arb, respectively, only in the analysis of H-mode operation. So do AMMC-max-Arb and AMMC-rtb-Arb. That is, AMMC-rtb-Arb analysis for L-mode operation (see Subsection 8.2.1) is also applicable to AMMC-max-Arb in L-mode operation. Therefore, in this section, we will only discuss the analysis of H-tasks in H-mode, including mode switch.

**H-mode analysis:** The completion time of job  $q$  of a H-task  $\tau_i$  in a level- $i$  busy period that comprises a mode switch, depends on:

1. The interference from jobs of higher-priority L-tasks,  $\tau_j$ , released before the mode switch. This can be computed as for AMMC-max with constrained deadlines, in Subsection 6.2.2, using the  $G^{L+}(\tau_j, s)$  function, (21).
2. The interference from jobs of higher-priority H-tasks,  $\tau_k$ , released during the level- $i$  busy period before completion of that job. Again, this can be computed as for AMMC-max with constrained deadlines, using the  $g^*(\tau_k, \ell^L(k, t, s), \ell^H(k, t, s))$  function, (20), and appropriate values for  $t$  and  $s$ .
3. The interference from the previous jobs of the task under analysis, released during the level- $i$  busy period before job  $q$ .

To compute the last interference component, we observe that some of the jobs of  $\tau_i$  before job  $q$  complete before the mode switch instant  $s$ , whereas the remaining ones complete after the mode switch. We observe that Lemma 1 is still valid for arbitrary-deadline tasks, because neither it nor its proof depend on  $D_i \leq T_i$ . Thus, to upper bound the interference, we upper bound the number of jobs that complete after the mode switch just like for AMC-max-Arb. Therefore we can use  $X(q, s, t)$ , given by (34). The number of jobs completed before the mode switch can also be estimated as for AMC-max-Arb:  $Y(q, s, t) = q + 1 - X(q, s, t)$ . Thus, the completion time of job  $q$  of a H-task  $\tau_i$  in a level- $i$  busy period comprising a mode switch can be computed using

the  $g^*$  function as follows:

$$r_i^*(q, s) = g^*(\tau_i, Y(q, s, r_i^*(q, s)), X(q, s, r_i^*(q, s))) + \quad (41)$$

$$+ \sum_{j \in hpL(i)} G^{L+}(\tau_j, s) + \sum_{k \in hpH(i)} g^*(\tau_k, l^L(k, r_i^*(q, s), s), l^H(k, r_i^*(q, s), s))$$

As for AMC-max, AMC-max-Arb and AMMC-max, it suffices to compute  $r_i^*(q, s)$  only for values of  $s$  that are equal to the release time of L-tasks with higher priority than  $\tau_i$ . The justification is that  $g^*(\tau_i, l, h)$  is non-increasing as  $s$  increases and  $G^{L+}(\tau_j, s)$  is a step function whose value increases for values of  $s$  corresponding to releases of jobs of task  $\tau_j$ . However, in a slight optimisation over [12], and similarly as in AMC-rtb-Arb and AMMC-rtb-Arb, we only need to consider such values of  $s$  from the interval  $[0, r_i^L(\min(p, q))]$ , instead of the potentially larger interval  $[0, r_i^L(q)]$ , where  $p$  is the job number of the last job in a level- $i$  busy period in L-mode. The justification is that  $r_i^L(p)$  is the longest i-level busy period in L-mode, thus if  $\tau_i$  is affected by a mode switch, this must occur before  $r_i^L(p)$ .

Let  $r_i^*(q)$  be the maximum of all the  $r_i^*(q, s)$  computed values. Then, the worst-case response time for job  $q$  in a level- $i$  busy period that comprises a mode switch can be computed as for AMC-max-Arb, i.e.:

$$R_i^*(q) = r_i^*(q) - qT_i \quad (42)$$

Again, in this analysis (41), we are trading-off computation time for pessimism. Indeed,  $r_i^L(\min(p, q))$  is the maximum completion time in L-mode over all possible sequences with  $\min(p, q) + 1$  jobs of  $\tau_i$ . So, as described above, we must compute the completion time of  $q + 1$  jobs for every value of  $s$  smaller than  $r_i^L(\min(p, q))$  that is equal to the release time of an L-job with higher priority than  $\tau_i$ . This value of  $s$  will affect every term on the RHS of (41), and in particular the first term, which is the maximum cumulative execution of  $q + 1$  jobs, such that some of them complete before the mode switch,  $s$ , and others complete after the mode switch. Assume that  $r_i^*(q, s)$  takes the maximum value for some sequence of  $q + 1$  jobs of  $\tau_i$  such that  $y$  of them complete before the mode switch and  $x$  of them complete after the mode switch. It may be the case that the completion time of that same sequence of  $y$  jobs in L-mode is shorter than the value of  $s$ . Thus, the maximum completion time computed for that value of  $s$  has some pessimism. A tighter analysis would require to upper-bound the completion time of the  $F_i$  sequences of  $q + 1$  jobs of  $\tau_i$ , one per frame in a superframe. For each sequence, the worst-case completion time would be computed using its cumulative WCET and also the respective L-mode completion time to upper-bound the values of  $s$  that would have to be used. Finally, the response time would be computed by taking the maximum of these completion times and subtracting  $qT_i$ .

Table 2: Example multiframe task set  $\tau$  with three tasks.

task	$\kappa$	$C^L$	$C^H$	T	D	Priority
$\tau_1$	L	{1,2,6,4 }	{ }	10	10	H
$\tau_2$	H	{3,5,2 }	{6,10,4 }	20	20	M
$\tau_3$	H	{1,2 }	{ 2,4 }	30	40	L

## 9 Example

In this section, we illustrate the application of the different analyses described in the previous section to the task set shown in Table 2. This task set comprises one L-task and two H-tasks. The tasks are listed in decreasing priority order, as found by Audsley’s priority assignment, which is discussed in Section 10. We focus on the response time at mode switch. Furthermore, we consider only the response time analysis of task  $\tau_3$ , which is the most interesting, as it is interfered with by the other two tasks. The response time analysis of the L- and H-modes, and also for the remaining tasks, is similar.

### 9.1 SMMC-Arb

For this analysis, the completion time of job  $q$  of task  $\tau_i$  is given by recurrence (38), which we repeat here for convenience:

$$r_i(q) = g^H(\tau_i, q + 1) + \sum_{j \in hpL(i)} G^L(\tau_j, r_i(q)) + \sum_{k \in hpH(i)} G^H(\tau_k, r_i(q))$$

Thus, the recurrence for completion time of the first job ( $q = 0$ ) of  $\tau_3$  of the example task set becomes:

$$r_3(0) = 4 + G^L(\tau_1, r_3(0)) + G^H(\tau_2, r_3(0))$$

which converges to the value 33. Thus, the response time of the first job of  $\tau_3$ , see (28), is:

$$R_3(0) = 33 - 0 * T_3 = 33$$

which is shorter than its deadline,  $D_3 = 40$ . However, it is larger than its period,  $T_3 = 30$ , thus we need to compute the response time for the 2nd job. The completion time of the second job, relative to the release of the first job, is obtained by solving the recurrence:

$$r_3(1) = 4 + 2 + G^L(\tau_1, r_3(1)) + G^H(\tau_2, r_3(1))$$

which converges to the value 35. Thus its response time is:

$$R_3(1) = 35 - 1 * T_3 = 5$$

Again, this is smaller than  $\tau_3$ ’s deadline. Furthermore, it is smaller than  $\tau_3$ ’s period, therefore there is no need to consider further jobs, and we can conclude that  $\tau_3$  is schedulable when it is assigned the lowest priority.

## 9.2 AMMC-rtb-Arb

For this analysis, we need first to analyze the response time in L-mode for the different jobs in a  $i$ -busy period, because the corresponding L-mode completion times are used in the H-mode completion times recurrences.

The recurrence for computing the completion time of job  $q$  of task  $\tau_i$  in L-mode is given by recurrence (39), which we repeat here for convenience:

$$r_i^L(q) = g^L(\tau_i, (q+1)) + \sum_{k \in hp(i)} G^L(\tau_k, r_i^L(q))$$

Thus, the completion time of  $\tau_3$ 's first job in L-mode can be obtained by solving the recurrence:

$$r_3^L(0) = 2 + G^L(\tau_1, r_3^L(0)) + G^L(\tau_2, r_3^L(0))$$

which converges to 17. Therefore,  $R_3^L(0) = 17$ . Since this is shorter than both the deadline and the period of  $\tau_3$ , we are done with L-mode analysis.

The recurrence (40) is used for computing the completion time of job  $q$  of task  $\tau_i$  in H-mode, which we repeat here for convenience:

$$r_i^H(q) = g^H(\tau_i, (q+1)) + \sum_{j \in hpL(i)} G^L(\tau_j, r_i^L(\min(p, q))) + \sum_{k \in hpH(i)} G^H(\tau_k, r_i^H(q))$$

Thus, the completion time of  $\tau_3$ 's first job upon a mode switch is given by:

$$r_3^H(0) = 4 + G^L(\tau_1, 17) + G^H(\tau_2, r_3^H(0)) \quad (43)$$

which converges to 30. Thus the response time of the first job is  $R_3^H(0) = 30$ , which is not greater than the deadline. Furthermore, it is also not greater than  $\tau_3$ 's period. So, this means that we have completed the response time analysis and that  $\tau_3$  is schedulable when it is assigned the lowest priority.

## 9.3 AMMC-max-Arb

As already mentioned, the AMMC-max-Arb analysis differs from AMMC-rtb-Arb only in the WCRT recurrence for the case of a mode switch. To reduce the pessimism of AMMC-rtb-Arb, the AMMC-max-Arb completion time recurrence for job  $q$  in a level- $i$  busy period, (41), which we repeat below for convenience, takes into account the mode switch instant,  $s$ .

$$r_i^*(q, s) = g^*(\tau_i, Y(q, s, r_i^*(q, s)), X(q, s, r_i^*(q, s))) + \sum_{j \in hpL(i)} G^{L+}(\tau_j, s) + \sum_{k \in hpH(i)} g^*(\tau_k, l^L(k, r_i^*(q, s), s), l^H(k, r_i^*(q, s), s))$$

As mentioned, it suffices to compute  $r_i^*(q, s)$  only for values of  $s$  that are equal to the release time of L-tasks with higher priority than  $\tau_i$  in  $[0, r_i^L(\min(p, q))]$ , where  $p$  is the job number of the last job in a level- $i$  busy period in L-mode.

Table 3: Response time of all the tasks.

task	$\kappa$	D	WCRT		
			SMMC	AMMC-rtb-Arb	AMMC-max-Arb
$\tau_1$	L	10	6	(6, -, -)	(6, -, -)
$\tau_2$	H	20	20	(15, 20, 10)	(15, 20, 10)
$\tau_3$	H	40	$R_3(0) = 33$ $R_3(1) = 5$	(17, 30, 14)	(17, 24, 14)

The WCRT triples in the AMMC-rtb-Arb and AMMC-max-Arb columns are for L-mode, mode switch and H-mode, respectively.

Once the completion time of job  $q$  has been determined, its worst-case response time can be easily computed:  $R_i^*(q) = r_i^*(q) - q \cdot T_i$ , where  $r_i^*(q)$  is the maximum of all  $r_i^*(q, s)$  values computed.

Thus to compute the completion time of the first job,  $q = 0$ , of  $\tau_3$ ,  $r_3^*(0)$ , we must consider the values of  $s$  in the interval  $[0, r_3^L(0)) = [0, 17)$ , corresponding to the release of task  $\tau_1$ , the only L-task with a priority higher than  $\tau_3$ . Because  $T_1 = 10$ , there are two possible values for  $s$ : 0 and 10.

Considering  $s = 0$  first, we get the recurrence:

$$r_3^*(0, 0) = g^*(\tau_3, Y(0, 0, r_3^*(0, 0)), X(0, 0, r_3^*(0, 0))) \\ + G^{L+}(\tau_1, 0) + g^*(\tau_2, l^L(2, r_3^*(0, 0), 0), l^H(2, r_3^*(0, 0), 0))$$

which converges to 20. Since the corresponding response time is not greater than the  $D_3$ , we now compute the completion time of the first job for  $s = 10$ , using the recurrence:

$$r_3^*(0, 10) = g^*(\tau_3, Y(0, 0, r_3^*(0, 10)), X(0, 0, r_3^*(0, 10))) \\ + G^{L+}(\tau_1, 10) + g^*(\tau_2, l^L(2, r_3^*(0, 10), 10), l^H(2, r_3^*(0, 10), 10))$$

which converges to 24. Thus:  $r_3^*(0) = \max\{20, 24\} = 24$ , and its worst-case response time is:  $R_3^*(0) = r_3^*(0) - 0 \cdot T_3 = 24$ . Since this value is smaller than  $D_3$ , the first job of  $\tau_3$  meets the deadline. Furthermore, because  $R_3^*(0) = 24 \leq T_3 = 30$ , there is at most one active job of  $\tau_3$  in any 3-level busy period. Thus, we can conclude that  $\tau_3$  is schedulable under AMMC-max-Arb when assigned the lowest priority.

Table 3 summarizes the worst-case response times of all tasks of the example task set. Since for all tasks the WCRTs are not greater than the respective deadline, the task set is schedulable by all analyses.

## 10 Priority Assignment

Deadline monotonic priority assignment is not optimal for AMC [2], and therefore it is not optimal for AMMC either; this applies to both AMMC-rtb and AMMC-max. (The same is true also for SMC and therefore SMMC.) However, Audsley's priority assignment algorithm [25] is optimal for AMC (and SMC)

and, as we will proceed to show, it is also optimal for their multi-frame extensions, AMMC, both AMMC-rtb and AMMC-max, and SMMC. Furthermore, this is true also for their arbitrary-deadline variants.

Indeed, it has been shown [26] that Audsley’s priority assignment is optimal for a given schedulability test  $S$ , if the following 3 conditions hold:

- Condition 1 The schedulability of a task  $\tau$  may, according to test  $S$ , depend on any independent properties of tasks with higher priority than  $\tau$ , but not on any properties of those tasks that depend on their relative priority ordering.
- Condition 2 The schedulability of a task  $\tau$  may, according to test  $S$ , depend on any independent properties of tasks with lower priority than  $\tau$ , but not on any properties of those tasks that depend on their relative priority ordering.
- Condition 3 When the priorities of any two tasks of adjacent priority are swapped, the task being assigned the higher priority cannot become unschedulable according to test  $S$ , if it was previously schedulable at the lower priority.

where by *independent property* we mean a property that is independent of the priority assigned to a task. (Actually, the wording we are using for these conditions is by Davis and Burns [27].)

By analysing the different worst-case response time analysis algorithms for multiframe tasks, it is straightforward to check that these conditions hold. We provide the arguments for the AMMC-max analysis, but they apply to the other multiframe mixed-criticality analyses, independently of whether the deadlines are constrained or arbitrary.

Our schedulability test is based on response-time analysis. In AMMC-max, for each task  $\tau_i$ , we check whether its worst-case response time in L-mode is not greater than its (relative) deadline. Furthermore, if  $\tau_i$  is an H-task, we check if its worst-case response time in H-mode, including the case of mode switch, is not greater than its deadline. These checks can be applied to each task independently of the relative priorities assigned to the tasks. In addition, they depend only on  $\tau_i$ ’s deadline and on the response time. The deadline is a fixed attribute of a task and independent of the priority assigned to a task. Also, the worst-case response times of a task in both L- and H-mode do not depend on the relative ordering of higher-priority tasks or on the relative ordering of lower-priority tasks.

Indeed, in AMMC-max, the L-mode response time of task  $\tau_i$  is computed with (14) (which is also valid for SMMC and AMMC-rtb), that we repeat here for convenience:

$$R_i^L = g^L(\tau_i, 1) + \sum_{\tau_j \in hp(i)} G^L(\tau_j, R_i^L)$$

We observe that  $g^L(\tau_i, 1)$ , which is given by (11), depends only on the L-WCETs of the jobs in  $\tau_i$ ’s superframe, and therefore does not depend on the relative priority ordering of the tasks. Likewise,  $G^L(\tau_j, R_i^L)$ , which is given by (12), depends on the L-WCETs of the jobs in  $\tau_j$ ’s superframe,  $\tau_j$ ’s period and

$R_i^L$ . Again,  $\tau_j$ 's attributes used in  $G^L$  function do not depend on the relative priority ordering of the tasks.  $R_i^L$  is also not dependent on the relative ordering among tasks with priority higher than  $\tau_i$  or among tasks with priority lower than  $\tau_i$ . Indeed,  $R_i^L$  is obtained by summing  $g^L(\tau_i, 1)$  and  $G^L(\tau_j, R_i^L)$  for each task  $\tau_j$  with a priority higher than  $\tau_i$ , and therefore does not depend on the relative ordering of higher-priority tasks, or even on lower-priority tasks.

With respect to Condition 3, it is straightforward to check that, by swapping the priorities of any two tasks of adjacent priorities, the second term of the right-hand side of the schedulability test, (14), for the task being assigned the higher priority cannot be larger than what it was in the previous priority assignment. Therefore, if that task was previously schedulable at the lower priority, it will remain schedulable at the higher priority.

In AMMC-max, the response time of H-task  $\tau_i$  in H-mode, including mode switch, is computed with (25), which we repeat here for convenience:

$$R_i^* = \max\{R_i^*(s) : 0 < s \leq R_i^L\}$$

where  $s$  is the mode switch instant relative to the release instant of  $\tau_i$ . As mentioned, the analysis needs to be performed only for values of  $s$  corresponding to the release of higher-priority L-tasks. These instants are independent of the relative priority order among higher-priority tasks or of the relative priority order among lower-priority tasks.  $R_i^*(s)$  is computed with (26), which we repeat here for convenience:

$$\begin{aligned} R_i^*(s) = & g^H(\tau_i, 1) + \sum_{j \in hpL(i)} G^{L+}(\tau_j, s) \\ & + \sum_{k \in hpH(i)} g^*(\tau_k, \ell^L(k, R_i^*(s), s), \ell^H(k, R_i^*(s), s)) \end{aligned}$$

We observe that  $g^H(\tau_i, 1)$  is similar to  $g^L(\tau_i, 1)$ , the only difference is that it uses the H-mode WCET of the jobs in a superframe instead of their L-mode WCET. Therefore it is also independent of the relative priority ordering among the other tasks. Following a line of argumentation similar to the one we presented above for  $G^L(\tau_j, R_i^L)$ , and taking into account that the values of  $s$  do not depend on the relative ordering of lower-priority tasks or even on higher-priority tasks, we conclude that  $G^{L+}(\tau_j, s)$  is also independent of the relative priority ordering among higher-priority tasks or among lower-priority tasks. Finally,  $g^*(\tau_k, \ell^L, \ell^H)$ , which is given by (20), depends on  $\ell^L(k, R_i^*(s), s)$  and  $\ell^H(k, R_i^*(s), s)$  and also on attributes of H-task  $\tau_k$ , i.e. its period and also the L- and H-WCET of the jobs in its superframe. These attributes of H-task  $\tau_k$  are independent of the priorities of the other tasks. The values  $\ell^L(k, R_i^*(s), s)$  and  $\ell^H(k, R_i^*(s), s)$  are given by (23) and (24), respectively, and depend only on the period and the deadline of task  $\tau_k$ , in addition to  $s$  and  $R_i^*(s)$ , which are also independent of the relative ordering among tasks with priority higher than  $\tau_i$ , or among tasks with priority lower than  $\tau_i$ . Indeed,  $R_i^*(s)$  is obtained by summing  $g^H(\tau_i, 1)$ ,  $G^{L+}(\tau_j, s)$  for each L-task  $\tau_j$  with higher priority than  $\tau_i$ , and  $g^*(k, \ell^L(k, R_i^*(s), s), \ell^H(k, R_i^*(s), s))$  for each H-task  $\tau_k$  with higher

priority than  $\tau_i$ , and therefore does not depend on the relative ordering of higher-priority tasks or on the relative ordering of lower-priority tasks.

With respect to Condition 3, we need to consider two cases, depending on the criticality of the task whose priority is lower after the priority swap. If the latter is an L-task, then the argument is exactly the same as presented above, for L-mode. If the task whose priority is lower after the priority swap is a H-task, then the third term of the right-hand side of the schedulability test, (26), for the task being assigned the higher priority cannot be larger than what it was in the previous priority assignment. Therefore, if that task was previously schedulable at the lower priority, it will remain schedulable at the higher priority. This concludes our argument that our schedulability test for constrained-deadline task-sets based on AMMC-max satisfies the 3 conditions stated above.

Finally, we note that the following theorem, which holds for mixed-criticality analyses, SMC and both AMC variants, with constrained deadlines [2] is also valid for the multiframe mixed-criticality analyses, SMMC and both AMMC variants, with constrained deadlines:

**Theorem 1** *An optimal priority ordering exists that has all tasks with the same criticality assigned priorities in deadline monotonic priority order.*

The proof follows the standard argument of the optimality of deadline monotonic priority assignment for tasks with constrained deadlines. The idea is that if there is some schedulable assignment such that the priority assigned to two tasks is not according to their deadlines, then the assignment obtained by swapping the priorities of these two tasks is also schedulable.

Consider two tasks  $\tau_i$  and  $\tau_j$  with the same criticality level such that  $D_i < D_j$ . Assume that there is a priority assignment such that the priority of  $\tau_j$  is higher than that of  $\tau_i$ , which is deemed schedulable. If we swap the priority of  $\tau_i$  and  $\tau_j$ , obviously  $\tau_i$ 's response time under this new priority assignment will be shorter than under the original priority assignment, and therefore  $\tau_i$  remains schedulable. Consider now  $\tau_j$ , whose priority is lower in the new priority assignment. The cumulative processing demand by the completion of  $\tau_j$ 's job under the new priority assignment is equal to the cumulative processing demand by the completion of  $\tau_i$  under the initial priority assignment. Indeed, because  $\tau_i$  and  $\tau_j$  are both constrained deadline tasks, there is only one job of each of these tasks in both schedules. Therefore, if  $\tau_i$ , which has an earlier deadline than  $\tau_j$ , was deemed schedulable under the initial priority assignment, so will  $\tau_j$  be in the new priority assignment.

This theorem allows us to use a variant of Audsley optimal assignment [2] that in each step considers at most 2 tasks: the H- and L- tasks not yet assigned with largest deadline, thus reducing the worst-case number of tests required to  $2n - 1$ , rather than  $n(n + 1)/2$  in the general case.

## 11 Evaluation

### 11.1 Experimental setup

The proposed analyses are implemented in a Java tool [28] to evaluate their scheduling performance. This Java tool has two modules. The first module generates the synthetic workload for the specified input parameters. The second module implements the different schedulability analyses. The generation of the synthetic task sets is controlled through the following parameters.

- Task periods are generated in the range of 10 msec to 1 sec using a log-uniform distribution.
- The deadlines of the tasks are generated within a range of  $[0.25, 4]T_i$  using a log-uniform distribution [12]. The analyses presented in Sections 5 and 6 hold for only constrained-deadlines, so their deadlines are selected with log-uniform distribution within a range of  $[0.25, 1]T_i$ .
- The UUnifast algorithm [29,30] is used to generate the L-mode utilization ( $U_{i,1}^L$ ) for the first frame of a task  $\tau_i$  in an unbiased way. The L-WCET of the first frame<sup>4</sup> of each task is then  $C_{i,1}^L = T_i \times U_{i,1}^L$ . The number of frames for each task is selected randomly with a uniform distribution within a range of  $[1, \alpha]$ , where  $\alpha > 1$  is a user-defined integer parameter. The L-WCETs of other frames of a task are randomly selected within an interval of  $[\beta \times C_{i,1}^L, C_{i,1}^L]$  with uniform distribution, where  $\beta \in (0, 1]$  is a task generation parameter limiting the L-WCET variation among a task's frames
- The user-defined fraction of H-tasks  $\xi \in (0, 1)$  in the task set.
- The H-WCET of the  $j^{th}$  frame of H-task  $\tau_i$  ( $C_{i,j}^H$ ) is derived by linearly scaling up that frame's L-WCET ( $C_{i,j}^L$ ) with a user-defined factor of  $\kappa$ , i.e.,  $C_{i,j}^H = \kappa \times C_{i,j}^L$ .

We used Audsley's optimal priority assignment [26] for all schedulability tests. The target utilization is varied within a range of  $(0, 1]$  with a step size of 0.1. Different random class objects are defined for utilization, minimum inter-arrival time, number of frames, deadline and L-WCET of each frame. These random class objects are seeded with different odd numbers and reused in successive replications [31]. For each set of input parameters, 1000 random task sets are generated. The parameters used in this evaluation are summarized in Table 4.

To avoid generating a huge number of plots, in each of our experiments, we only vary one parameter at a time, with the other parameters conforming to their default values, displayed in Table 4. The triples in the Values column are respectively the minimum value, the step size and the maximum value

---

<sup>4</sup> For convenience, without loss of generality (since shift-rotating the order of the frames results in an equivalent multiframe task), the first frame has the biggest L-WCET. Please note that our analyses make no assumptions regarding the WCET of the first frame being the greatest.

Table 4: Overview of Parameters

Parameters	Values	Default
H-WCET scale-up factor ( $\kappa$ )	{2 : 0.5 : 6}	3
Task set size ( $n$ )	{8 : 4 : 32}	16
Fraction of H-tasks in $\tau$ ( $\xi$ )	{0.2 : 0.05 : 0.7}	0.4
Upper bound on # of frames ( $\alpha$ )	{3 : 1 : 10}	5
Lower bound on L-WCET ( $\beta$ ) variation	{0.1 : 0.1 : 0.8}	0.2
Inter-arrival time ( $T_i$ )	10ms to 1s	N/A
Deadline ( $D_i$ )	$LoqUni f\{0.25 \quad 4 T_i\}$	N/A

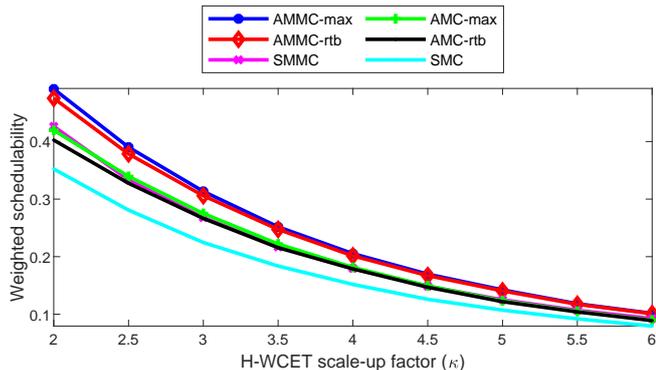


Fig. 3: Effect of H-WCET variation on constrained-deadline analyses

for the respective parameter. The number of plots is further reduced by plotting weighted schedulability rather than schedulability success ratio for each possible combination of input parameters. The weighted schedulability representation condenses three dimensional-plots to two-dimensional plots [32,33] by eliminating the axis of task set utilization. This performance metric gives more weight to task sets with higher utilization. Using notation from [33], let  $S_y(\tau, p)$  represent the result (0 or 1) of the schedulability test  $y$  for a given task set  $\tau$  with an input parameter  $p$ . Then  $W_y(p)$ , the weighted schedulability for that test  $y$  as a function  $p$ , is presented in (44), where  $U^L(\tau)$  is the nominal system utilization of the task set  $\tau$  in the L-mode.

$$W_y(p) = \frac{\sum_{\forall \tau} (U^L(\tau) \cdot S_y(\tau, p))}{\sum_{\forall \tau} U^L(\tau)} \quad (44)$$

## 11.2 Results

We compared the weighted schedulability of the six proposed schedulability analyses (SMMC, AMMC-rtb, AMMC-max, SMMC-Arb, AMMC-rtb-Arb, AMMC-max-Arb) against the existing ones (SMC, AMC-rtb, AMC-max, SMC-Arb, AMC-rtb-Arb and AMC-max-Arb). For the SMC, AMC-rtb, AMC-max,

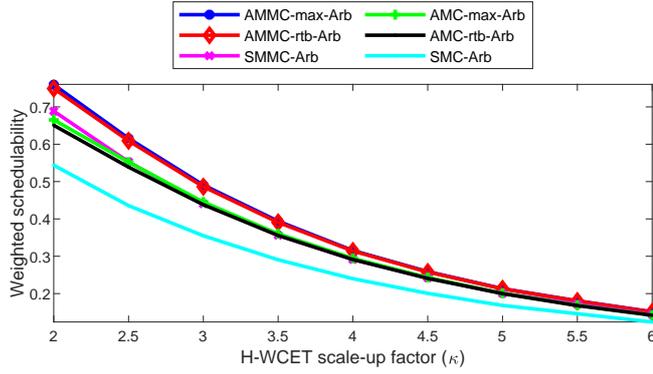


Fig. 4: Effect of H-WCET variation on arbitrary-deadline analyses

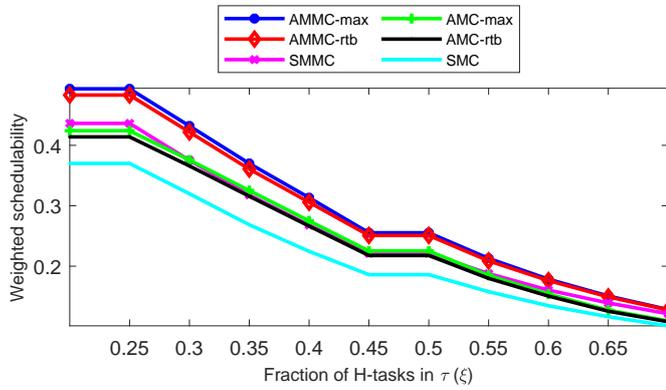


Fig. 5: Weighted Schedulability vs. H-task share (constrained-deadline analyses)

SMC-Arb, AMC-rtb-Arb and AMC-max-Arb tests, the multiframe task is transformed into an instance of the classic (i.e., single-frame) mixed-criticality task-model by pessimistically discarding frame information. In this transformation, for any task  $\tau_i$ , its L-WCET and H-WCET are set to  $C_i^L = \max_{j=0}^{F_i-1} C_{i,j}^L$  and  $C_i^H = \max_{j=0}^{F_i-1} C_{i,j}^H$ , respectively.

Please note that the implicit-deadline model ( $D_i = T_i$ ) was used in the evaluation of the constrained-deadline schedulability analyses presented in our previous work [22]. The use of the constrained deadlines in this paper for the algorithms presented in Sections 5 and 6 (constrained-deadline schedulability analyses) has led to a slight decrease in their weighted scheduling success ratio when compared to the results presented in [22].

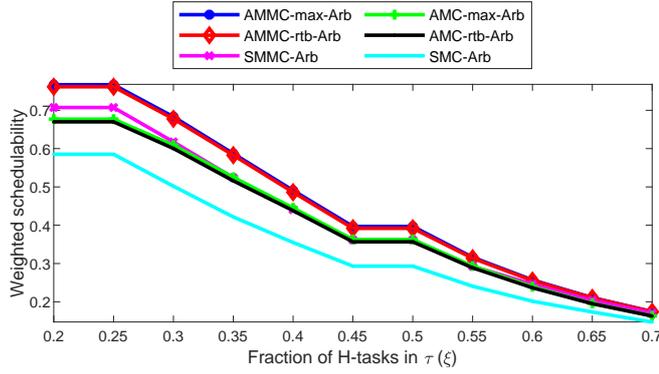


Fig. 6: Weighted Schedulability vs. H-task share (arbitrary-deadline analyses)

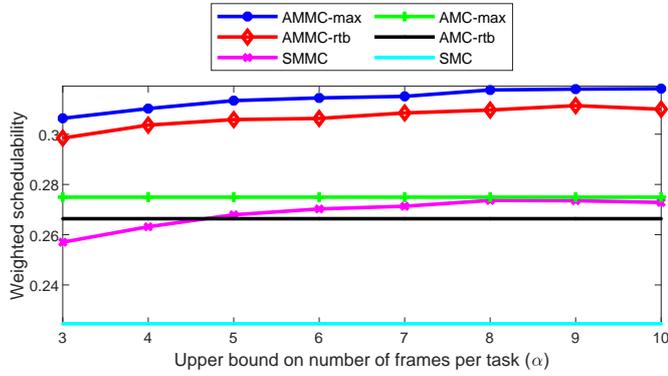


Fig. 7: Weighted schedulability vs. max number of frames in a task (constrained-deadline analyses)

Figures 3 and 4 present the effect of varying the H-WCET scale-up factor ( $\kappa$ ) on all analyses. A higher value for  $\kappa$  increases the H-mode processor requirement of high-criticality tasks, hence the weighted schedulability of all analyses decreases accordingly. The multiframe-based analyses perform better than their corresponding frame-agnostic conventional analyses by leveraging frame information. As expected, AMC-max outperforms AMC-rtb and AMMC-max outperforms AMMC-rtb, albeit by very small margins in both cases. The difference between adaptive and static variants comes from the fact that static analyses do not drop L-tasks, this makes the WCRT of H-tasks computed with SMC more likely to exceed their deadlines. SMMC performs better at low values of  $\kappa$  compared to SMC, AMC-rtb and AMC-max, but the

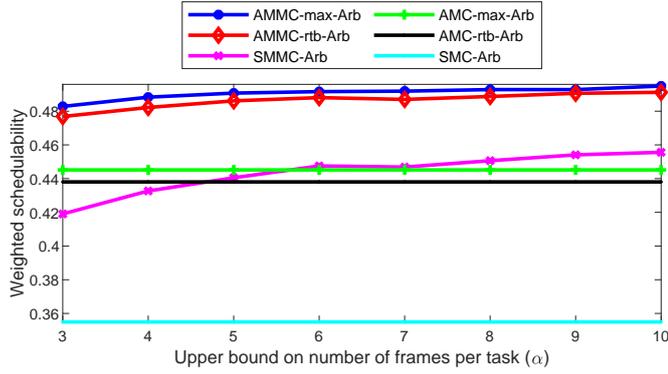


Fig. 8: Weighted schedulability vs. max. number of frames in a task (arbitrary-deadline analyses)

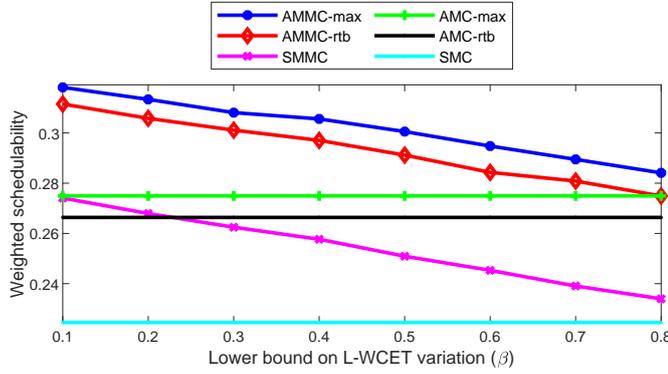


Fig. 9: Comparison of constrained-deadline analyses for L-WCET variation

difference among them decreases with an increase in  $\kappa$ , because task sets become harder to schedule. The observations mentioned for constrained-deadline analyses are also true for the arbitrary-deadline analyses (see Figure 4). The constrained-deadline analyses have more processing requirement when compared to arbitrary-deadline analyses as  $D_i \leq T_i$ . Hence, the arbitrary-deadline analyses when compared to their corresponding constrained-deadline analyses show better weighted schedulability ratio. Please note that the difference between the proposed arbitrary-deadline frame-based analyses and their corresponding single-frame based analyses is lower when compared to a similar comparison in constrained-deadline analyses. A higher  $\kappa$  reduces the overall

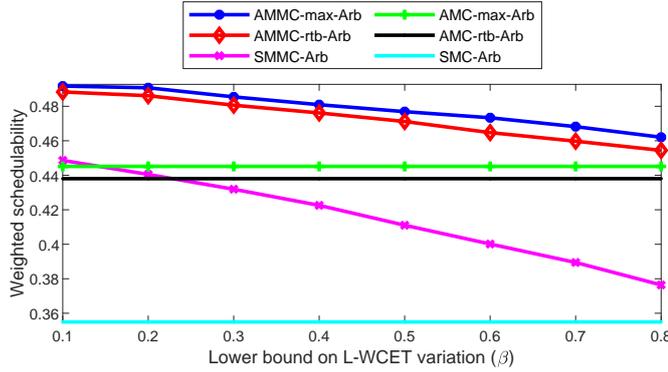


Fig. 10: Comparison of arbitrary-deadline analyses for L-WCET variation

Table 5: Maximum absolute difference in schedulability success ratio of each multiframe analysis from its corresponding single-frame analysis.

Analyses	Varied parameters				
	$\kappa$	$ \tau $	$\xi$	$\alpha$	$\beta$
AMMC-max	<b>14.9%</b>	14.3%	14.6%	9.2%	8.6%
AMMC-rtb	<b>16.6%</b>	15.7%	15.1%	10.8%	10.5%
SMMC	19.3%	<b>20.0%</b>	15.6%	13.5%	14.2%
AMMC-max-Arb	29.1%	12.2%	<b>31.4%</b>	8.2%	8.2%
AMMC-rtb-Arb	28.1%	11.4%	<b>30.9%</b>	8.1%	8.1%
SMMC-Arb	28.4%	<b>29.6%</b>	26.3%	18.8%	18.0%

schedulability ratio for each analysis, therefore, the differences in performance between the multiframe- and single-frame-based analyses also decrease.

A higher fraction of H-tasks ( $\xi$ ) increases the processor requirement of the system in H-mode. Hence, the weighted schedulability of all analyses decreases (Figures 5 and 6). Their absolute difference in terms of weighted schedulability also decreases, as the number of feasible task sets decreases with a higher fraction of H-tasks.

The potential for improvements in weighted schedulability for SMMC, AMMC-rtb, AMMC-max, SMMC-Arb, AMMC-rtb-Arb and AMMC-max-Arb over single-frame-based analyses increases when the number of frames per task is higher. A larger value of upper bound for the number of frames per tasks results in more frames per task on average as well. With more frames in a task, the probability of having low-WCET frames increases, which in turn magnifies the improvement from leveraging the frame information (or, equivalently, magnifies the pessimism when disregarding it). Hence, the weighted schedulability of the multiframe-based analyses improves when the upper bound on the number of frames per task is higher, as shown in Figures 7 and 8. The classic SMC, AMC-rtb, AMC-max, SMC-rtb, AMC-rtb-Arb and AMC-max-

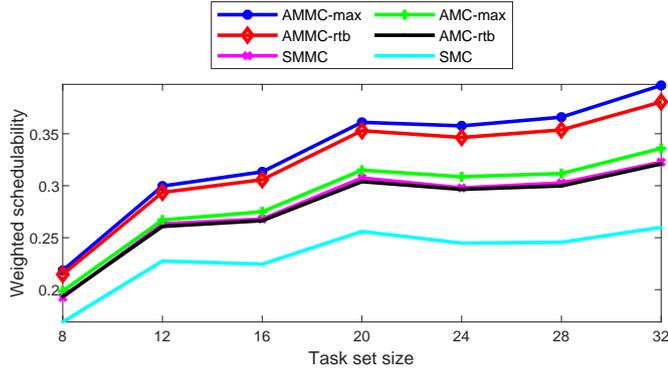


Fig. 11: Effect of task set size variation over weighted schedulability of constrained-deadline analyses

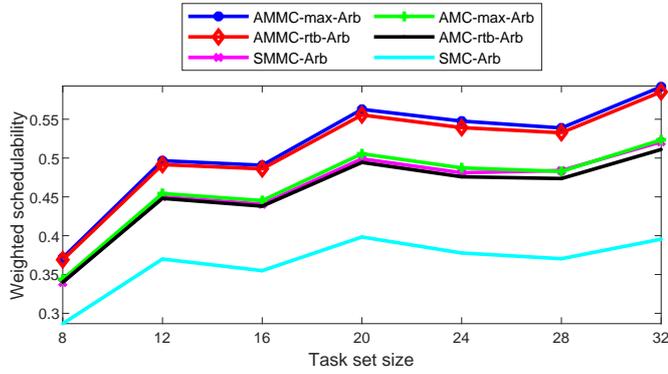


Fig. 12: Effect of task set size variation over weighted schedulability of arbitrary-deadline analyses

Arb are insensitive to this parameter, as these analyses assume the maximum estimates for L-WCET and H-WCET over all frames. For low  $\alpha$ , leveraging frame information does not compensate for the pessimism of static multiframe analyses (SMMC and SMMC-Arb), causing it to under perform compared to conventional single-frame adaptive analyses (AMC-rtb, AMC-max, AMC-rtb-Arb and AMC-max-Arb).

The WCET variation limiting parameter  $\beta$  defines a lower bound for the ratio between the smallest frame L-WCET and the greatest frame WCET of a given task. A higher value of  $\beta$  decreases this range, and consequently, increases the average execution requirement of task's frames, all other things

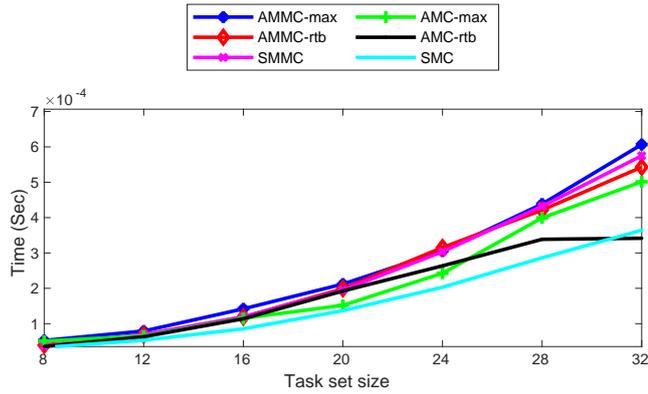


Fig. 13: Impact of task set size variation on the average running time of constrained-deadline analyses

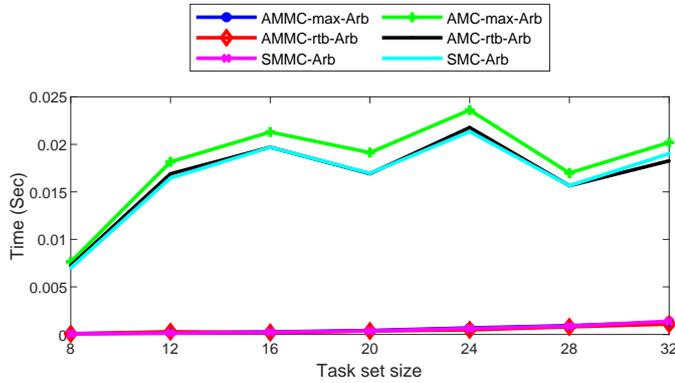


Fig. 14: Impact of task set size variation on the average running time of arbitrary-deadline analyses

remaining equal. Hence, the weighted schedulability of the multiframe-based analyses decreases with an increase in  $\beta$  (see Figures 9 and 10). One important observation is that the absolute difference among the weighted schedulability of AMMC-max, AMMC-rtb and SMMC increases with an increase in  $\beta$ . The pessimism in AMMC-rtb against AMMC-max becomes a major differentiator when the potential of gaining from the multiframe properties becomes limited. The same behavior is shown by SMMC against SMC. These effects also holds true for arbitrary-deadline analyses (see Figure 10). Similarly to the bound on the maximum number of frames per task,  $\beta$  has no effect on SMC, AMC-rtb,

Table 6: Average running time (in seconds) of all constrained-deadline analyses with default parameters.

AMMC-max	AMMC-rtb	SMMC	AMC-max	AMC-rtb	SMC
$2.6233 \times 10^{-4}$	$2.4362 \times 10^{-4}$	$2.4725 \times 10^{-4}$	$2.1804 \times 10^{-4}$	$1.9268 \times 10^{-4}$	$1.6601 \times 10^{-4}$

AMC-max, SMC-Arb, AMC-rtb-Arb and AMC-max-Arb, as the maximum of the WCET estimates over all frames are assumed in each mode.

The effect of variation in the number of tasks on the weighted schedulability is presented in Figures 11 and 12. The important observation is that, similarly to single-criticality systems, the weighted schedulability improves with larger task set sizes, as more low-utilization tasks are easier to schedule compared to fewer high-utilization tasks. The fact that during task generation, the number of H-tasks is rounded up to the nearest integer as needed (i.e., when the target fraction of H-tasks would result in a non-integer number) explains the saw-tooth shape of the weighted schedulability plots of each analysis.

To quantify the benefits in terms of *non-weighted* schedulability success ratio, Table 5 shows the maximum absolute difference in non-weighted schedulability success ratio of each multiframe analysis over its corresponding single-frame analysis. In the best case, the AMMC-max, AMMC-rtb, SMMC, AMMC-max-Arb, AMMC-rtb-Arb and SMMC-Arb analyses achieve up to 14.9%, 16.6%, 20%, 31.4%, 30.9% and 29.6% higher schedulability successes ratio, respectively, compared to their corresponding single-frame analyses. The bold values in Table 5 shows the highest difference achieved by each analysis among all experiments.

Finally, we experimentally explore the running time of each analysis. The platform (PowerEdge-T320) used for these experiments has 32 GB RAM and eight Intel(R) Xeon(R) E5-2420 v2 CPUs, each with a maximum frequency of 2.20 GHz and running Linux Mint 18.3 Sylvia. Except for the task set size, the running time is virtually insensitive to other parameters. Figure 13 presents the average running times of all constrained-deadline analyses for different task set sizes in seconds. AMMC-max takes longer than other analyses as it requires more computation while computing the processor requirement in each iteration. As expected, the running times of all tests increase with an increase in task set size as the feasibility testing has to be performed for each task. Figure 14 presents the average running time of the arbitrary-deadline task analyses. The frame-oblivious analyses are pessimistic and the length of the busy interval in the feasibility testing of such analyses becomes much longer when compared to multiframe analyses. Hence, in Figure 14 (contrary to Figure 13), the frame-oblivious analyses take more time than multiframe analyses. For the default parameters, the average running times of all analyses are also presented in Tables 6 and 7. The observations presented above for Figures 13 and 14 are consistent with the data shown in Tables 6 and 7.

Table 7: Average running time (in seconds) of all arbitrary-deadline analysis with default parameters.

AMMC-max-Arb	AMMC-rtb-Arb	SMMC-Arb	AMC-max-Arb	AMC-rtb-Arb	SMC-Arb
$5.49 \times 10^{-4}$	$4.70 \times 10^{-4}$	$5.30 \times 10^{-4}$	$1.8144 \times 10^{-2}$	$1.6632 \times 10^{-2}$	$1.6589 \times 10^{-2}$

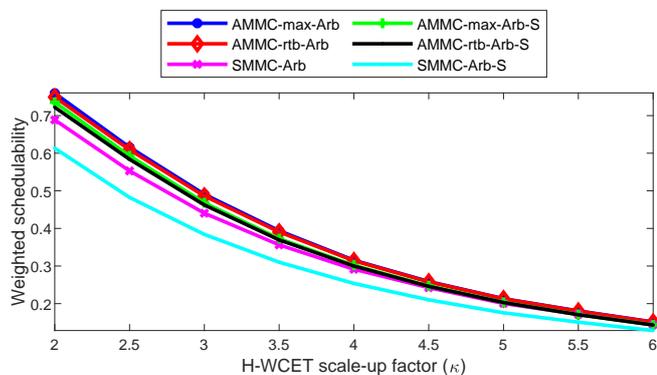


Fig. 15: Weighted Schedulability vs. H-WCET variation – case study

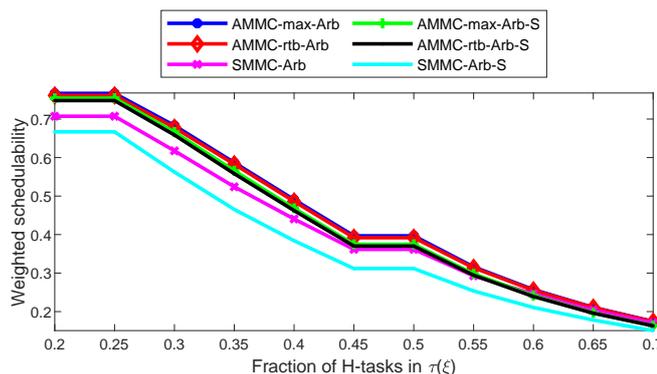


Fig. 16: Weighted Schedulability vs. H-task share – case study.

### 11.3 Case Study

In safety-related applications, the computation of H-WCET and its certification for each frame is expensive and time consuming. This case study explores a scenario (also referred to as a special case) with multiple L-WCET estimates (multiple frames in L-mode) and a single H-WCET estimate (single frame in H-mode), and analyzes its scheduling performance degradation over

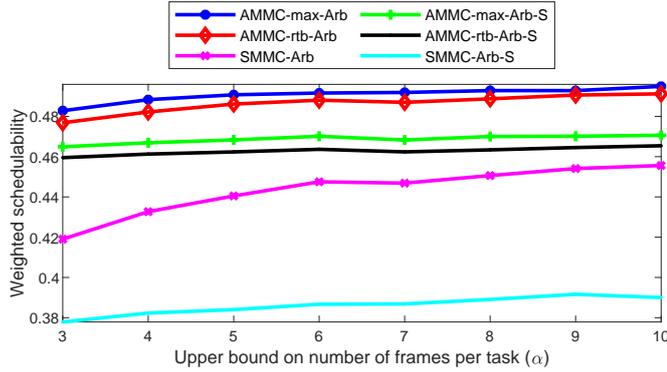


Fig. 17: Weighted schedulability vs. max. number of frames in a task – case study.

the generic multiframe setting. In our results, we append “-S”, from single (H-frame), as suffix to the acronym of each multiframe analysis to differentiate this special case from generic multiframe analyses. Only multiframe arbitrary-deadline task schedulability analyses are compared in this case study. In our experimental setup, the single H-WCET estimate of any H-task in the special case is computed to be  $C_i^H = \kappa \times \max_{j=0}^{F_i} C_{i,j}^L$ .

Figures 15 to 19 compare the weighted schedulability of the generic multiframe analyses with those of the special case by varying different parameters. As expected, the special case analyses lead to a lower weighted schedulability than the respective generic analyses. Indeed, because we set  $C_i^H$  to the product of the scale-up factor,  $\kappa$ , by the largest WCET of all L-frames of task  $\tau_i$ , the execution demand of every task in the special case analysis is never lower than in the generic analysis, and is almost always larger.

Independently of the factor of the experiment, the reduction in weighted schedulability is larger for SMMC-Arb-S than for the two AMMC variants. Furthermore, the reduction in weighed schedulability for AMMC-rtb-Arb-S is approximately equal to that for AMMC-max-Arb-S. These observations can be justified, a posteriori, by the differences in these analyses. Indeed, in SMMC-Arb-S, there is only one single mode of operation, which is affected by the increase in the WCET of the frames of H-tasks, whereas in the AMMC variants, the execution demand in L-mode is not affected, only the execution demand in H-mode and upon mode switch.

Another interesting observation, is that the effect of each factor on the weighted schedulability of a special case analysis is similar to that observed on the weighted schedulability of the respective generic analysis. Again this was somewhat expected since we can view each special case analysis as just the generic analysis of a task set with slightly different parameters, namely in terms of the WCET of H-tasks: in the case of the AMMC variants only the

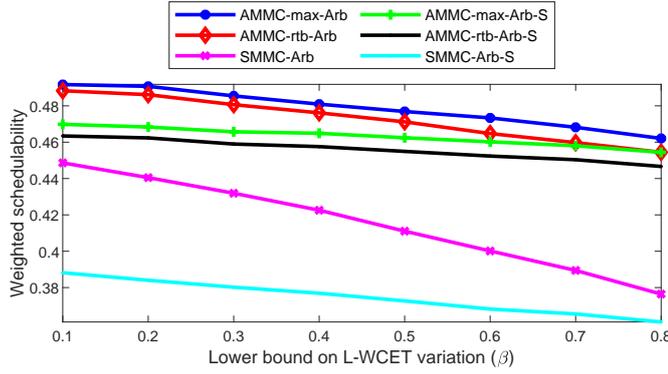


Fig. 18: Weighted schedulability vs. L-WCET variation – case study.

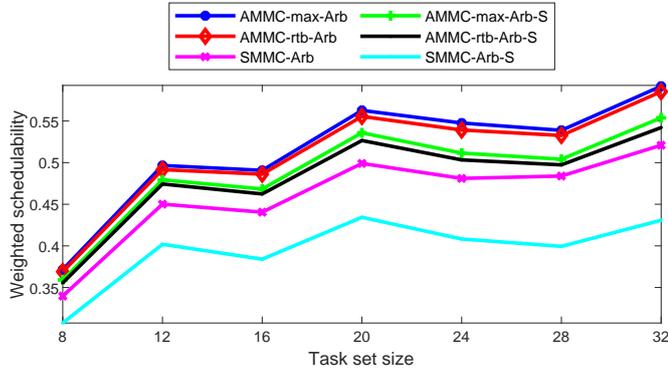


Fig. 19: Comparison of case study for L-WCET variation

H-WCET of the H-tasks is different. Thus the observed effects of each factor on the weighted schedulability of each special case analysis can be justified in the same way as we have done in Section 11.2 for the respective generic analysis.

Finally, because weighted schedulability is biased towards task sets with higher utilization, Table 8 summarizes the results of this case study in terms of (non-weighted) schedulability ratio. The maximum absolute reduction was found to be 21.1%, 7.8% and 6.9% for SMMC-*arb*-S, AMMC-*rtb*-S and AMMC-*max*-S, respectively, when compared to the respective generic analyses.

Table 8: Maximum absolute reduction in non-weighted schedulability ratio of each special case analysis compared with the respective generic analysis.

Analyses	Varied parameters				
	$\kappa$	$ \tau $	$\xi$	$\alpha$	$\beta$
AMMC-max-Arb-S	5.3%	6.4%	<b>6.9%</b>	4.8%	4.3%
AMMC-rtb-Arb-S	5.2%	7.7%	<b>7.8%</b>	4.8%	4.3%
SMMC-Arb-S	15.2%	<b>21.1%</b>	14.6%	13.2%	12.3%

## 12 Conclusions

In this work, and in the context of uniprocessor fixed-priority-scheduled platforms, we extended the mode-based mixed-criticality model of Vestal, which assumed a single worst-case execution time per task per mode. This extension accommodates multiframe tasks that have different WCETs for their jobs, in a repeating pattern. For this extended model, we formulated schedulability analyses (SMMC, AMMC-rtb and AMMC-max) that leverage the frame information, resulting in greater accuracy over the state-of-the-art mixed-criticality schedulability tests (SMC, AMC-rtb and AMC-max) that, pessimistically, have to discard frame information and use a single WCET per task per mode, in order to be applied. These tests, formulated for constrained-deadline systems, were later generalized for arbitrary-deadline tasks (as SMMC, AMMC-rtb and AMMC-max), drawing from the existing results on arbitrary-deadline mixed-criticality (but single-frame) scheduling. Finally, we reason about task priority assignment for the scheduling problem considered, when the new analyses are used for schedulability testing, and prove the optimality of Audsley’s algorithm for the general case, and of a faster variant thereof, in the case of constrained deadline systems. Experimental evaluation with synthetic task sets confirms the benefits (up to 31.4% in schedulability success ratio) of the new model and its analysis, compared to schedulability tests that disregard frame information. In the future, we intend to consider a partitioned multi-core arrangement and incorporate the effects of memory stalls under memory access regulation into the schedulability analysis to make it more realistic and more applicable to real-world systems.

## Acknowledgements

This work was partially supported by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UIDB/04234/2020); also by the Operational Competitiveness Programme and Internationalization (COMPETE 2020) under the PT2020 Partnership Agreement, through the European Regional Development Fund (ERDF), and by national funds through the FCT, within project POCI-01-0145-FEDER-029119 (PREFECT). This work was also supported by the Netherlands Organization for Applied Scientific Research TNO.

## References

1. S. Baruah and A. Burns, "Implementing mixed criticality systems in Ada," in 16th Ada-Europe Conference, 2011, pp. 174–188.
2. S. K. Baruah, A. Burns, and R. I. Davis, "Response-time analysis for mixed criticality systems," in Proceedings of the 32nd IEEE Real-Time Systems Symposium, 2011, pp. 34–43.
3. M. Joseph and P. Pandya, "Finding Response Times in a Real-Time System," The Computer Journal, vol. 29, no. 5, pp. 390–395, 1986.
4. D. Le Gall, "Mpeg: A video compression standard for multimedia applications," Communications of the ACM, vol. 34, no. 4, pp. 46–58, Apr. 1991.
5. ARINC specification 818-2 Avionics Digital Video Bus (ADVB) High Data Rate, 818th ed., AERONAUTICAL RADIO, INC., 2013.
6. C. Bailey, A. Burns, A. Wellings, and C. Forsyth, "Keynote paper: A performance analysis of a hard real-time system," Control Engineering Practice, vol. 3, no. 4, pp. 447–464, 1995.
7. J. P. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: Exact characterization and average case behavior," in Proceedings of the 10th IEEE Real-Time Systems Symposium, 1989, pp. 166–171.
8. A. K. Mok and D. Chen, "A multiframe model for real-time tasks," IEEE Transactions on Software Engineering, vol. 23, no. 10, pp. 635–645, Oct 1997.
9. C. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," Journal of the ACM, vol. 20, pp. 46–61, 1973.
10. S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in Proceedings of the 28th IEEE Real-Time Systems Symposium, 2007.
11. A. Burns and R. I. Davis, "A survey of research into mixed criticality systems," ACM Computing Surveys, vol. 50, no. 6, pp. 82:1–82:37, Nov. 2017.
12. —, "Response time analysis for mixed criticality systems with arbitrary deadlines," in 5th International Workshop on Mixed Criticality Systems (WMC 2017). York, 2017.
13. S. Asyaban and M. Kargahi, "Feasibility interval for fixed-priority scheduling of mixed-criticality periodic tasks with offsets," IEEE Embedded Systems Letters, vol. 11, no. 1, pp. 17–20, March 2019.
14. T. Fleming and A. Burns, "Extending mixed criticality scheduling," in Proc. WMC, RTSS, 2013, pp. 7–12.
15. H.-M. Huang, C. Gill, and C. Lu, "Implementation and evaluation of mixed-criticality scheduling approaches for sporadic tasks," ACM Transactions on Embedded Computing Systems, vol. 13, no. 4s, pp. 126:1–126:25, Apr. 2014.
16. T. Fleming, H.-M. Huang, A. Burns, C. Gill, S. Baruah, and C. Lu, "Corrections to and discussion of "implementation and evaluation of mixed-criticality scheduling approaches for sporadic tasks"," ACM Transactions on Embedded Computing Systems, vol. 16, no. 3, pp. 77:1–77:4, 2017.
17. Q. Zhao, Z. Gu, and H. Zeng, "Pt-amc: Integrating preemption thresholds into mixed-criticality scheduling," in Proceedings of the 50th ACM/IEEE Conference on Design Automation Conference, March 2013, pp. 141–146.
18. S. K. Baruah and A. Mok, "Static-priority scheduling of multiframe tasks," in Proceedings of the 11th Euromicro Conference on Real-Time Systems, June 1999, pp. 38–45.
19. S. Baruah, D. Chen, S. Gorinsky, and A. Mok, "Generalized multiframe tasks," Journal of Real-Time Systems, vol. 17, no. 1, pp. 5–22, Jul 1999.
20. A. Zuhily and A. Burns, "Exact scheduling analysis of non-accumulatively monotonic multiframe tasks," Real-Time Systems, vol. 43, no. 2, pp. 119–146, 2009.
21. G. Bilsen, M. Engels, R. Lauwereins, and J. Peperstraete, "Cycle-static dataflow," IEEE Transactions on signal processing, vol. 44, no. 2, pp. 397–408, 1996.
22. I. Hussain, M. A. Awan, P. F. Souto, K. Bletsas, B. Akesson, and E. Tovar, "Response time analysis of multiframe mixed-criticality systems," in Proceedings of the 27th International Conference on Real-Time Networks and Systems, 2019, pp. 8–18.

23. K. W. Tindell, A. Burns, and A. J. Wellings, "An extendible approach for analyzing fixed priority hard real-time tasks," Real-Time Systems, vol. 6, no. 2, pp. 133–151, 1994.
24. J. P. Lehoczky, "Fixed priority scheduling of periodic task sets with arbitrary deadlines," in [1990] Proceedings 11th Real-Time Systems Symposium. IEEE, 1990, pp. 201–209.
25. N. Audsley, "Optimal priority assignment and feasibility of static priority tasks with arbitrary start times: Technical report ycs164," Department of Computer Science, University of York, 1991.
26. N. C. Audsley, "On priority assignment in fixed priority scheduling," Information Processing Letters, vol. 79, no. 1, pp. 39–44, 2001.
27. R. I. Davis and A. Burns, "Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems," Real-Time Systems, vol. 47, no. 1, pp. 1–40, 2011.
28. B. Nikolic, M. A. Awan, and S. M. Petters, "SPARTS: Simulator for power aware and real-time systems," in Proceedings of the 8th IEEE International Conference on Embedded Software and Systems. Changsha, China: IEEE, Nov. 2011, pp. 999–1004.
29. E. Bini and G. Buttazzo, "Measuring the performance of schedulability tests," Journal of Real-Time Systems, vol. 30, no. 1-2, pp. 129–154, 2009.
30. R. I. Davis and A. Burns, "Priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems," in Proceedings of the 30th IEEE Real-Time Systems Symposium, 2009, pp. 398–409.
31. R. Jain, The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling., ser. Wiley professional computing. Wiley, 1991.
32. A. Bastoni, B. Brandenburg, and J. Anderson, "Cache-related preemption and migration delays: Empirical approximation and impact on schedulability," Proceedings of OSPERT, pp. 33–44, 2010.
33. A. Burns and R. Davis, "Adaptive mixed criticality scheduling with deferred preemption," in Proceedings of the 35th IEEE Real-Time Systems Symposium, Dec 2014, pp. 21–30.