Schedulability Analysis for Adaptive Mixed Criticality Systems with Arbitrary Deadlines and Semi-Clairvoyance

Alan Burns
Department of Computer Science, University of York, UK.
Email: alan.burns@york.ac.uk

Robert I. Davis
Department of Computer Science, University of York, UK.
Email: rob.davis@york.ac.uk

***Abstract*—This paper provides analysis of the Adaptive Mixed Criticality (AMC) scheduling scheme for mixed-criticality systems that include tasks with arbitrary deadlines and semi-clairvoyant behavior. An arbitrary deadline task is one that can have a deadline that may be greater than its period. A semi-clairvoyant task is one that upon arrival of each job, reveals which of its two WCET parameters will be respected. This enables an earlier switch to be made from the normal mode of operation to the abnormal mode. The previously published schedulability test AMC-max is modified to cater for both of these extensions. Evaluation shows that there is a significant improvement in schedulability for semi-clairvoyant tasks over non-clairvoyant, and for arbitrary-deadline tasks over considering those deadlines as being constrained by the task's period.**

I. Introduction

Since the publication of Vestal's model [30] there has been a significant number of papers published on the scheduling of Mixed Criticality Systems (MCS); these are summarised in a comprehensive survey [16] published in 2017, and in a more up-to-date review [17]. Many of these papers focus on schemes based on Fixed-Priority Preemptive Scheduling (FPPS). Within FPPS schemes, Adaptive Mixed Criticality (AMC) [8] is widely regarded as the most effective approach [24], and has been built upon to take account of additional aspects including: preemption thresholds [32], [33], deferred preemption [14], multiple criticality levels [22], criticality-specific periods [9], [31], weakly-hard timing constraints [23], probabilistic task models [28], and context switch costs [18]. An exact analysis for AMC has also been developed [3] for periodic task sets with offsets.

In the original paper on AMC [8] two forms of response-time analysis were introduced: AMC-max and a less precise, but computationally more efficient scheme called AMC-rtb. In this paper we extend the schedulability analysis for AMC-max to include two important characteristics of a more general model of MCS, namely: tasks with *arbitrary deadlines* that may be greater than their periods[1] and tasks that exhibit *semi-clairvoyant* behaviour. While most extensions to AMC have been analysed using extensions to AMC-rtb, the two characteristics considered in this paper require the use of the more precise analysis, AMC-max.

Arbitrary-deadline tasks cater for situations where there is some leeway in when a task must execute. For example, a consumer task that reads items from a buffer must, over a long time interval, consume items at the same rate as they are produced; however, the task can have response times that are longer than its period, provided that the buffer has sufficient space to store unread items. A task set may be unschedulable if deadlines are constrained to be less than or equal to task periods, but may meet all of its time constraints if tasks are permitted to have longer, (i.e. arbitrary) deadlines.

A defining property of a MCS is that there is more than one estimate of a task's Worst-Case Execution Time (WCET). Prior work on mixed-criticality scheduling theory assumes that the actual execution time of a given invocation or job of a task is only revealed by actually executing that job. Which WCET estimate applies therefore only becomes known when the job either terminates or has executed for a considerable time. In the semi-clairvoyant model it is instead assumed that upon arrival, a job reveals which of its WCET parameters it will respect. The semi-clairvoyant scheduler thus has access to some limited information about the future behavior of the task.

The notion of Semi-Clairvoyance was introduced by Agrawal et al. [2] as an intermediate step between an ordinary Mixed-Criticality (MC) scheduler and a clairvoyant scheduler that has access to all future behaviors. A fully clairvoyant scheduler is an idealized abstraction that cannot be realized in practice; it serves as an unobtainable upper bound on what any real scheduler can possibly achieve. A semi-clairvoyant scheduler is, by contrast, a practical possibility; it only requires information as to which mode of operation each job of a task will invoke, and only needs that information to be made available when the job arrives.

Agrawal et al. [2], in their introduction to semi-clairvoyance, focused on scheduling finite sets of jobs, rather than on scheduling tasks (i.e. recurring jobs). They proved a number of significant properties. It had been previously shown [6] that determining whether a set of jobs is MC-schedulable is intractable (NP-hard in the strong sense) and that no MC-schedulable algorithm can have a speedup factor smaller than $\left(\sqrt{5}+1\right)/2 \approx 1.618$; a bound that is now known to be tight [1]. This speedup factor is with respect to a clairvoyant scheduler. Agrawal et al. [2] derived a semi-clairvoyant scheduling algorithm (LPSC) with a speedup factor of $\frac{3}{2} = 1.5$, which was shown to be tight. LPSC was proven optimal, with schedulability determined in polynomial time.

In this paper we derive analysis for mixed-criticality *tasks*, and demonstrate that significant improvements in schedulability can be achieved via a semi-clairvoyant AMC approach.

There are a number of potential characteristics of MCS that could benefit from the semi-clairvoyant model:

- System developers may provide *alternative implemen-*

[1] A preliminary version of this analysis appeared in a workshop paper [15] at WMC 2017.

tations of a task: upon arrival of a job, the task knows which implementation to execute under the given circumstances.
- The task may normally be expected to deal with up to say 10 objects in an image, if on arrival it is known that there are more than 10 objects then a larger WCET estimate will apply.
- One or more tasks may have parameterised (by mode) WCET estimates. On arrival of a job, the current mode is known; for example, fault recovery modes may require the execution of extra fault mitigation code.

In general, the benefit of the semi-clairvoyant approach arises when the execution time of the task's code depends on the state of the system at the time the job arrives, rather than on some internal property that emerges as it executes.

The remainder of the paper is organized as follows. Section II outlines the standard MCS model. Section III recaps on the extended form of response-time analysis for arbitrary-deadline tasks under FPPS. Section IV reviews existing schedulability analysis for the Adaptive Mixed Criticality (AMC) scheme, assuming tasks with constrained deadlines. The two main contributions of the paper are presented in Section V, which introduces schedulability analysis for arbitrary-deadline tasks under AMC, and Section VI, which introduces analysis for constrained and arbitrary-deadline semi-clairvoyant tasks again under AMC. Section VII addresses priority assignment, while Section VIII explores the dominance relationships between semi-clairvoyant and non-clairvoyant AMC scheduling. Section IX evaluates the performance of the various schemes and schedulability tests, including comparisons against prior work. Section X concludes.

II. System Model, Terminology and Notation

In this paper, we are interested in the fixed priority preemptive scheduling of a single processor system comprising a static set of n sporadic tasks. Each standard single-criticality task, τ_i, is defined by its period or minimum inter-arrival time, relative deadline, Worst-Case Execution Time (WCET), and unique priority: (T_i, D_i, C_i, P_i). Task deadlines may be arbitrary, i.e. less than, equal to, or greater than their periods.

We assume that each task, τ_i, gives rise to a potentially unbounded sequence of jobs, with the release of each job separated by at least the minimum inter-arrival time from the release of the previous job of the same task. The worst-case response time of task τ_i is denoted by R_i and corresponds to the longest response time from release to completion for any of its jobs. For tasks with arbitrary deadlines, more than one job of the same task may be active at any given time. Among jobs of the same task, those released earlier are executed first.

A Mixed-Criticality System (MCS) is assumed to be defined over two criticality levels, HI and LO. Each LO-criticality task τ_j is assumed to have a single estimate of its WCET: $C_j(LO)$, while each HI-criticality task τ_k has two estimates: $C_k(HI)$ and $C_k(LO)$, with $C_k(HI) \geq C_k(LO)$. (Note we drop the task index when using these and other terms in a generic way, but include the index when referring to the parameters of a specific task). In addition to these criticality-specific execution time values, MCS tasks also have the standard parameters: T_i, D_i, and P_i.

Most scheduling approaches for MCS identify different modes of behavior. In the LO-criticality (or normal) mode, all tasks execute within their $C(LO)$ bounds and all deadlines are required to be met. At all times LO-criticality tasks are constrained by run-time monitoring to execute for no more than their $C(LO)$ bound. In contrast, if a HI-criticality task executes for $C(LO)$ without signalling completion then the system enters the HI-criticality (or abnormal) mode. In this mode only HI-criticality tasks are required to meet their deadlines. HI-criticality tasks are assumed to execute for no more than $C(HI)$. The response time of a task τ_i in the LO-criticality mode is denoted by $R_i(LO)$ and in the HI-criticality mode by $R_i(HI)$.

The system is assumed to execute on a single processor. The approaches developed are, however, applicable to multi-processor platforms that employ partitioned scheduling.

III. Existing Analysis for FPPS

In this section, we recap existing analysis for constrained-deadline tasks ($D_i \leq T_i$) in a single criticality level system, scheduled according to FPPS. The response time of each task τ_i can be computed as follows (see [25], [5] for a full derivation):

$$R_i \;=\; C_i \;+\; \sum_{j \in \mathbf{hp}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \qquad (1)$$

where $\mathbf{hp(i)}$ is the set of tasks with higher priority than τ_i. This and all subsequent response time equations can be solved via fixed-point iteration. In this case, iteration starts with a value of $R_i = C_i$ and continues until convergence or until the computed response time exceeds the task's deadline.

For tasks with arbitrary deadlines ($D_i \geq T_i$) it is possible in a schedulable system that $R_i \geq T_i$. It follows that there can be more than one job of task τ_i active within the same priority level-i busy period[2]; any of these jobs can give rise to the worst-case response time for the task. We use q as an index to denote each job within the busy period, with $q = 0$ indicating the first job. The finish time of each job of the task, $f_i(q)$ ($0 \leq q \leq p$), as measured from the start of the busy period, can be computed as follows (see [29], [26] for a full derivation):

$$f_i(q) \;=\; (q+1)C_i \;+\; \sum_{j \in \mathbf{hp}(i)} \left\lceil \frac{f_i(q)}{T_j} \right\rceil C_j \qquad (2)$$

The last job in the busy period is denoted by p, which is the first value where completion of the job occurs before the next release of the task, i.e. where $f_i(p) \leq (p+1)T_i$. The response-time of each job q is calculated as follows,

$$\forall_{q, 0 \leq q \leq p} :\; R_i(q) = f_i(q) - qT_i \qquad (3)$$

with the worst-case response time of the task given by:

$$R_i \;=\; \max_{\forall q, 0 \leq q \leq p} \{R_i(q)\} \qquad (4)$$

This analysis for FPPS scheduling of tasks can be applied to MCS by simply assuming that all HI-criticality tasks have a single execution time of $C(HI)$ and all LO-criticality tasks have a single execution time of $C(LO)$. Similarly, the more complex analysis for tasks with arbitrary deadlines can be ignored, and (1) applied, via the expedient of reducing the

[2]A priority level-i busy period is a continuous interval of time $[t^1, t^2)$ during which there is always some pending (i.e. uncompleted) workload of priority i or higher that arrived during the interval, and strictly before t^2.

relative deadline of any task with $D > T$ so that $D = T$. In subsequent sections, we derive analyses that significantly outperform these simple approaches.

IV. Existing Analysis for AMC

In this section, we recap existing analysis for Adaptive Mixed Criticality (AMC) [8] scheduling. With AMC, if a HI-criticality task executes for $C(LO)$ without completing, then the system enters the HI-criticality mode. Previously released jobs of LO-criticality tasks may be completed, but subsequent releases of LO-criticality tasks are not started. Only HI-criticality tasks are required to be schedulable in the HI-criticality mode; however, the LO-criticality mode may be re-entered when the processor becomes idle, or indeed earlier [11].

In the original paper on AMC [8], two sufficient schedulability tests were developed. The first approach, called AMC-rtb, takes account of a bound on the duration over which LO-criticality tasks can interfere. The second, more precise approach, called AMC-max, determines the worst-case response time by taking into account all possible times at which the criticality mode change could occur. In order to accommodate the requirements of arbitrary deadlines and semi-clairvoyant scheduling we later build upon the analysis for AMC-max.

The AMC-max analysis first computes the worst-case response time for each task τ_i in the LO-criticality mode:

$$R_i(LO) = C_i(LO) + \sum_{\tau_j \in \mathbf{hp}(i)} \left\lceil \frac{R_i(LO)}{T_j} \right\rceil C_j(LO) \qquad (5)$$

AMC-max then computes the worst-case response time $R_i^s(HI)$ of HI-criticality task τ_i, assuming a mode change at time s, and then takes the maximum of these values over all possible values of s. A formulation for R_i^s is constructed from the different forms of interference that task τ_i can experience,

$$R_i^s(HI) = C_i(HI) + I_L(s) + I_H(s, R_i^s(HI)) \qquad (6)$$

where $I_L(s)$ is the interference from higher priority LO-criticality tasks, and hence is only a function of s. $I_H(s,t)$ is the interference from higher priority HI-criticality tasks, and hence is a function of s and t; the latter being the length of the priority level-i busy period, which here equates to the response-time.

As jobs of higher priority LO-criticality tasks are prevented from being released after the mode change at time s, their worst-case interference is upper bounded by:

$$I_L(s) = \sum_{j \in \mathbf{hpL}(i)} \left(\left\lfloor \frac{s}{T_j} \right\rfloor + 1 \right) C_j(LO) \qquad (7)$$

where $\mathbf{hpL(i)}$ is the set of LO-criticality tasks with higher priority than τ_i; similarly, in the following, $\mathbf{hpH(i)}$ is the set of HI-criticality tasks with higher priority than τ_i.

$I_H(s,t)$ is defined by considering the number of jobs of each higher priority HI-criticality task τ_k that can execute in a priority level-i busy period of length t, with the mode change taking place at time s, with $s < t$. Those jobs that have some part of their execution after time s can contribute interference of $C(HI)$, with the remainder contributing $C(LO)$.

The maximum number of jobs of τ_k with $D_k \leq T_k$ that can fit into an interval of length $t - s$ is bounded by:

$$\left\lceil \frac{t - s + D_k}{T_k} \right\rceil \qquad (8)$$

Equation (8) follows from the fact that the latest a job of task τ_k can execute is at its deadline, while the earliest that subsequent jobs can execute is at their arrival times. Equation (8) can be pessimistic; including more jobs than can actually arrive in an interval of length t. This is taken into account by defining:

$$M(k,s,t) = \min \left\{ \left\lceil \frac{t - s + D_k}{T_k} \right\rceil, \left\lceil \frac{t}{T_k} \right\rceil \right\} \qquad (9)$$

where $M(k,s,t)$ is the maximum number of jobs of task τ_k that can exhibit HI-criticality behavior in a busy period of length t with a transition to the HI-criticality mode at time s. The interference term for higher priority HI-criticality tasks in an interval of length t thus becomes [8]:

$$I_H(s,t) = \sum_{k \in \mathbf{hpH}(i)} \left\lceil \frac{t}{T_k} \right\rceil C_k(LO) + \sum_{k \in \mathbf{hpH}(i)} M(k,s,t) \left(C_k(HI) - C_k(LO) \right) \qquad (10)$$

Hence the worst-case response time with a mode change at time s is given by [8]:

$$R_i^s(HI) = C_i(HI) + \sum_{j \in \mathbf{hpL}(i)} \left(\left\lfloor \frac{s}{T_j} \right\rfloor + 1 \right) C_j(LO) + \sum_{k \in \mathbf{hpH}(i)} \left\lceil \frac{R_i^s(HI)}{T_k} \right\rceil C_k(LO) + \sum_{k \in \mathbf{hpH}(i)} M(k,s,t) \left(C_k(HI) - C_k(LO) \right) \qquad (11)$$

The worst-case response time of the task is then the maximum over all possible values of s:

$$R_i(HI) = \max_{\forall s, s < R_i(LO)} \{ R_i^s(HI) \} \qquad (12)$$

Note, for consistency with the derivations provided in subsequent sections, the terms in (9), (10) and (11) have been simplified or re-arranged with respect to how they appear in [8].

Finally, it is necessary to limit the number of values of s that are considered from the range of all possible values. In (11), the $\mathbf{hpL}$ term increases as a step function with increasing values of s, while the $\mathbf{hpH}$ term decreases. It follows that $R_i^s(HI)$ can only increase at values of s corresponding to multiples of the periods of LO-criticality tasks, hence these are the only values of s that need to be considered. Further, the mode change must occur by $R_i(LO)$, otherwise either task τ_i completes or is itself responsible for causing the mode change at that time. If so, then any LO-criticality job arriving at exactly $R_i(LO)$ will not be allowed to execute, and hence s is restricted in (12) to the interval [0,$R_i(LO)$) [8].

Note that the AMC-max analysis recapped above does not assume a synchronous arrival sequence for all tasks, as that would not necessarily result in the worst-case response time. Rather, the analysis accounts independently for the maximum interference that can occur in two time windows, the first of length s representing LO-criticality mode, and the second of length $t - s$ representing HI-criticality mode. This same basic approach is used in the derivation of further analyses in subsequent sections.

V. Arbitrary-Deadline Analysis for AMC

In this section, we introduce[1] AMC-max-Arb analysis for arbitrary-deadline tasks scheduled according to AMC.

First, we consider each task τ_i in the LO-criticality mode. Building on the analysis for arbitrary-deadline tasks under FPPS [29], [26] (recapped in Section III), the length $f_i^{LO}(q)$ of the priority level-i busy period in LO-criticality mode, up to the completion of job q of τ_i, is given by:

$$f_i^{LO}(q) \;=\; (q+1)C_i(LO) \;+\; \sum_{j \in \mathbf{hp}(i)} \left\lceil \frac{f_i^{LO}(q)}{T_j} \right\rceil C_j(LO) \tag{13}$$

The worst-case response time of each job in the LO-criticality mode is therefore given by:

$$\forall_{q,0 \le q \le p}: \quad R_i^{LO}(q) = f_i^{LO}(q) - qT_i \tag{14}$$

and the worst-case response time of the task in the LO-criticality mode is given by:

$$R_i(LO) \;=\; \max_{\forall_{q,0 \le q \le p}} \left\{ R_i^{LO}(q) \right\} \tag{15}$$

As with the analysis for FPPS, iteration over the values of q ends at p, the smallest value such that $f_i^{LO}(p) \le (p+1)T_i$, indicating that $f_i^{LO}(p)$ corresponds to the end of the priority level-i busy period in the LO-criticality mode.

We now consider the worst-case response-time of HI-criticality tasks in the HI-criticality mode (including the transition to it). It turns out that the derivation, recapped in Section IV for the constrained-deadline case, applies with some simple adaptations.

First, we compute the completion time $f_i^s(q)$ of the qth job of task τ_i when the mode change occurs at time s, as follows:

$$f_i^s(q) = x \cdot C_i(HI) + y \cdot C_i(LO) + I_L(s) + I_H(s, f_i^s(q)) \tag{16}$$

where x is the number of HI-criticality jobs, and y is the number of LO-criticality jobs and hence $x + y = q + 1$. The values of x and y are determined below.

Following the same argument as the original AMC-max analysis [8], recapped in Section IV, we need to determine an upper bound $I_H(s,t)$ on the interference that HI-criticality tasks such as τ_k can cause in a priority level-i busy period of length t if the mode change occurs at time s, with $s < t$. To do so, we maximize the number of jobs of τ_k still potentially active at time s, as all of these jobs can contribute interference of $C_k(HI)$, while all other jobs of τ_k contribute $C_k(LO)$. In an interval of length $t - s$ there can be at most

$$\left\lceil \frac{t - s + D_k}{T_k} \right\rceil \tag{17}$$

active jobs of τ_k.

Equation (17) is identical to (8) that caters for tasks with $D \le T$. Since (9) and (10) are derived from (8), and otherwise depend only on the values t and s, they are also applicable to tasks with $D > T$. Thus, (16) can be used to compute $f_i^s(q)$, with $I_H(s, f_i^s(q))$ given by (10) and $I_L(s)$, the upper bound on interference from LO-criticality tasks, given by (7).

The maximum number, x, of jobs of the task under analysis, τ_i, that contribute $C_i(HI)$ can be derived in a similar way to (9). Accounting for the fact that there are at most $q+1$ jobs of task τ_i in the busy period, we have:

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

and since the total number of jobs is $q + 1$,

$$y = q + 1 - x$$

Note, when x is substituted into (16), the value of t is given by $f_i^s(q)$.

The next step is to consider all possible values for s:

$$f_i^{HI}(q) \;=\; \max_{\forall s, s < f_i^{LO}(q)} \{ f_i^s(q) \} \tag{18}$$

As with the previous AMC-max analysis [8], it is necessary to limit the number of values of s that are considered in (18). The contribution to $f_i^s(q)$ from HI-criticality tasks, $I_H(s,t)$, given by (10) and also the contribution from task τ_i itself are decreasing in s, while the contribution from LO-criticality tasks is an increasing step function. It follows that $f_i^s(q)$ can only increase at values of s corresponding to multiples of the periods of LO-criticality tasks, hence again these are the only values of s that need to be considered. Further, by the time $f_i^{LO}(q)$ (given by (13)), either the busy period has ended or job q of task τ_i causes the mode change itself, hence s can be restricted to the interval [0,$f_i^{LO}(q)$).

Each job's response time is given by:

$$\forall_{q,0 \le q \le v}: \quad R_i^{HI}(q) = f_i^{HI}(q) - qT_i \tag{19}$$

where v is the smallest value such that $f_i^{HI}(v) \le (v+1)T_i$. Finally, the worst-case response time is given by:

$$R_i(HI) \;=\; \max_{\forall_{q,0 \le q \le v}} \left\{ R_i^{HI}(q) \right\} \tag{20}$$

If a value of q considered in (20), (19), and hence in (18), exceeds the maximum number of jobs p that can be present in a LO-criticality mode priority level-i busy period (where p is the smallest value such that $f_i^{LO}(p) \le (p+1)T_i$), then the value $f_i^{LO}(p)$ is used in place of $f_i^{LO}(q)$ to limit the range of values of s that are checked. This holds because $f_i^{LO}(p)$ is the maximum possible continuous interval of LO-criticality mode execution, after which there can only be either a mode change (caused by job p of task τ_i) or an idle instant.

VI. Analysis for Semi-Clairvoyant Scheduling

In this section we develop analysis for mixed-criticality task systems scheduled according to the rules for semi-clairvoyant scheduling, as stated below:

- A job is defined as **normal** if it can execute for no more than $C(LO)$, otherwise it is defined as **abnormal** and can execute for no more than $C(HI)$. Note, LO-criticality tasks are assumed to generate only **normal** jobs, whereas HI-criticality tasks can generate both **normal** and **abnormal** jobs.
- If all jobs that have arrived since the last idle instant are **normal** then the mode of the system is **normal**; otherwise it is **abnormal**.
- When **normal** mode applies, jobs of both LO-criticality and HI-criticality tasks must meet their deadlines.
- When **abnormal** mode applies, jobs of HI-criticality tasks must still meet their deadlines; however, jobs of LO-criticality tasks are not required to meet their deadlines.
- On arrival, each job indicates to the scheduler whether it is **normal** or **abnormal**.

Note a job's actual execution time is not known before it completes execution, all that is known is whether the job is **normal** or **abnormal**, and hence whether a mode change is required. The scheduler can therefore initiate a mode change at the time the job arrives, rather than waiting until its execution time reaches the $C(LO)$ bound. With the semi-clairvoyant AMC scheme, jobs of LO-criticality tasks that arrive during **abnormal** mode are not executed.

The task's indication must be safe, if a job declares that it is **normal** but subsequently executes for more than $C(LO)$, then the analysis developed below would be invalid. However, a job that indicates that it is **abnormal**, but then completes with an execution time of less than $C(LO)$ does not invalidate the analysis. Both the job and the mode are **abnormal**, irrespective of the fact that an **abnormal** job may sometimes complete in less than $C(LO)$.

In terms of implementation, the above rules mean that the semi-clairvoyant AMC model does not necessarily require execution time monitoring. For reasons of fault tolerance, however, the implementation may be required to ensure that no LO-criticality task executes for more than $C(LO)$, and no HI-criticality task executes for more than $C(HI)$. The absence of the requirement to identify when a HI-criticality task has executed for more than $C(LO)$ simplifies the implementation of semi-clairvoyant AMC scheduling, compared to the non-clairvoyant case, potentially reducing its run-time overheads.

The following two subsections introduce analysis for semi-clairvoyant AMC scheduling. First, for constrained-deadline task sets we have AMC-sem, which builds on the existing AMC-max analysis, recapped in Section IV. Second, for arbitrary-deadline task sets, we have AMC-sem-Arb, which builds on the new analysis for arbitrary-deadline tasks sets with non-clairvoyant behavior introduced in Section V.

A. Analysis for Semi-Clairvoyant Constrained-Deadline Tasks

For constrained-deadline tasks, considering **normal** mode, the worst-case response time for all tasks is given by (5), as in the non-clairvoyant case.

To analyse the response time of a HI-criticality task τ_i in **abnormal** mode (including the transition to it), we again consider a priority level-i busy period that starts at time 0, ends at time t, and has a mode change at time s, with $s < t$.

We begin by considering the interference, $I_L(s)$, in the busy period from higher priority LO-criticality tasks, and similarly the interference, $I_H(s,t)$, from higher priority HI-criticality tasks. Semi-clairvoyant scheduling causes no change to the interference from LO-criticality tasks in the interval $[0,s]$, thus $I_L(s)$ is again given by (7). There are, however, differences in the interference, $I_H(s,t)$, generated by higher priority HI-criticality tasks. These tasks can release **normal** jobs that arrive before s and execute for a maximum of $C(LO)$, and **abnormal** jobs that *arrive* at or after s and execute for a maximum of $C(HI)$. Unlike in the non-clairvoyant case, no job that arrives before s can execute for more than $C(LO)$, even if some of its execution takes place after s. Since **abnormal** jobs contribute a larger amount of interference, we maximize the number of these jobs in the interval from $[s,t)$. $I_H(s,t)$ can thus be bounded by assuming interference of $C(LO)$ for the maximum number of jobs that arrive in the interval $[0,t)$, plus an extra $C(HI)-C(LO)$ for the maximum number of jobs that can arrive at or after s, i.e. in the interval $[s,t)$, hence:

$$I_H(s,t) = \sum_{j \in \mathbf{hpH}(i)} \left(\left\lceil \frac{t}{T_j} \right\rceil C_j(LO) + \left\lceil \frac{t-s}{T_j} \right\rceil (C_j(HI) - C_j(LO)) \right) \quad (21)$$

To derive the worst-case response time, we consider two distinct cases for the HI-criticality task τ_i under analysis:

Case 1: Task τ_i gives rise to a **normal** job that executes for, at most, $C_i(LO)$. In this case, the worst-case response time occurs when the job arrives at time 0. (If the job arrived later within the busy period, then it would finish at the same time, but its response time would be smaller). Here, a job of some other HI-criticality task is assumed to arrive at time s and trigger the mode change.

Case 2: Task τ_i gives rise to an **abnormal** job that executes for $C_i(HI)$. By definition of time s, this job arrives no earlier than s. In this case, the worst-case response time occurs when the job arrives at time s and causes the mode change. (If the job arrived later within the busy period than time s, then it would finish at the same time, but its response time would again be smaller).

Note in Case 2 it is still necessary to compute the busy period starting from time 0, rather than from time s when the job of task τ_i arrives. This is because jobs of LO-criticality tasks with higher priorities may "push" jobs of other HI-criticality tasks that arrive before time s into executing after time s, thus causing extra interference on task τ_i, increasing its response time. If this occurs, then these jobs of HI-criticality tasks will nevertheless only execute for a maximum of $C(LO)$, since they are **normal** jobs that arrive before time s.

The two cases impose different constraints on the values of s that need to be considered.

In Case 1, as with AMC-max analysis for non-clairvoyant scheduling recapped in Section IV, only values of s that correspond to the arrival times of higher priority LO-criticality tasks in the interval $[0, R_i(LO))$ need be checked. Here, any value of s greater than or equal to $R_i(LO)$ would mean that τ_i had already completed execution (of its normal job) before the mode change could happen, with a worst-case response time of $R_i(LO)$ that is already accounted for via analysis of the **normal** mode.

In Case 2, since task τ_i arrives at time s, we need only check values of s that correspond to the arrival times of higher priority LO-criticality tasks in the interval $[0, S_i(LO))$, where $S_i(LO)$ corresponds to the worst-case start time of task τ_i in **normal** mode. Here, any value of s greater than or equal to $S_i(LO)$ would mean that at some point in the interval $[0,s]$, the busy period ended (i.e. there was no pending execution left), and hence the interval $[0,t)$ is not a valid busy period that needs to be considered. $S_i(LO)$ can be determined as follows [13]:

$$S_i(LO) = \sum_{\tau_j \in \mathbf{hp}(i)} \left(\left\lfloor \frac{S_i(LO)}{T_j} \right\rfloor + 1 \right) C_j(LO) \quad (22)$$

The two cases give rise to re-formulations of (6) from the original AMC-max analysis [8].

$$\forall_{s, s \le R_i(LO)} \bullet R_i^s(HI)_1 = C_i(LO) + I_L(s) + I_H(s, R_i^s(HI)_1) \quad (23)$$

$$\forall_{s, s \le S_i(LO)} \bullet R_i^s(HI)_2 = C_i(HI) + I_L(s) + I_H(s, R_i^s(HI)_2) \quad (24)$$

where $I_L(s)$ is given by (7), and $I_H(s,t)$ is given by (21).

In Case 2, although the busy period starts at time 0, task τ_i arrives no earlier than time s, and hence its worst-case response time is given by $R_i^s(HI)_2 - s$. It follows that the overall worst-case response time for task τ_i in **abnormal** mode is given by:

$$R_i(HI) = \max\left\{ \max_{s, s \le R_i(LO)} \{R_i^s(HI)_1\}, \max_{s, s \le S_i(LO)} \{R_i^s(HI)_2 - s\} \right\} \tag{25}$$

where $R_i^s(HI)_1$ is defined as follows by substituting $I_L(s)$ from (7), and $I_H(s,t)$ from (21), into (23):

$$\begin{aligned} R_i^s(HI)_1 = C_i(LO) + & \\ \sum_{j \in \mathbf{hpL}(i)} & \left(\left\lfloor \frac{s}{T_j} \right\rfloor + 1 \right) C_j(LO) + \\ \sum_{j \in \mathbf{hpH}(i)} & \left\lceil \frac{R_i^s(HI)_1}{T_j} \right\rceil C_j(LO) + \\ \sum_{j \in \mathbf{hpH}(i)} & \left\lceil \frac{R_i^s(HI)_1 - s}{T_j} \right\rceil (C_j(HI) - C_j(LO)) \end{aligned} \tag{26}$$

$R_i^s(HI)_2$, defined in (24), expands in a similar way, with $C_i(HI)$ in the first line instead of $C_i(LO)$.

B. Analysis for Semi-Clairvoyant Arbitrary-Deadline Tasks

In this section we combine the two main extensions introduced in this paper: analysis for arbitrary-deadline tasks given in Section V, and analysis for semi-clairvoyant constrained-deadline tasks given in Section VI-A.

For semi-clairvoyant arbitrary-deadline tasks, considering the **normal** mode, the worst-case response time for all tasks is given by (15), as in the non-clairvoyant case.

To analyse the response time of a HI-criticality task τ_i in **abnormal** mode (including the transition to it) we consider priority level-i busy periods that start at time 0, end at some time t, and have a mode change at time s, with $0 \le s < t$. In the arbitrary-deadline case, we need to consider a number of jobs of task τ_i. Recall that we use q as the index to denote each job within the busy period, with $q = 0$ indicating the first job. In total, there are $q+1$ jobs of task τ_i in the busy period. The maximum number of **abnormal** jobs is denoted by x and the number **normal** jobs by y, with $x + y = q + 1$. The values for x and y are determined below.

We begin by considering the interference, $I_L(s)$, from higher priority LO-criticality tasks, and similarly the interference, $I_H(s,t)$, from higher priority HI-criticality tasks. Since these terms do not depend on the parameters or jobs of task τ_i, they are given by the same formulation used in the semi-clairvoyant constrained-deadline case, i.e. $I_L(s)$ is given by (7), and $I_H(s,t)$ is given by (21). Building on (26), we therefore have the following formula for the length, $f_i^s(q)$, of the busy period up to the completion of job q of task τ_i.

$$\begin{aligned} f_i^s(q) = x \cdot C_i(HI) + y \cdot C_i(LO) + & \\ \sum_{j \in \mathbf{hpL}(i)} & \left(\left\lfloor \frac{s}{T_j} \right\rfloor + 1 \right) C_j(LO) + \\ \sum_{j \in \mathbf{hpH}(i)} & \left\lceil \frac{f_i^s(q)}{T_j} \right\rceil C_j(LO) + \\ \sum_{j \in \mathbf{hpH}(i)} & \left\lceil \frac{f_i^s(q) - s}{T_j} \right\rceil (C_j(HI) - C_j(LO)) \end{aligned} \tag{27}$$

To derive the worst-case response time for job q of HI-criticality task τ_i with a mode change at time s, we again make a case distinction:

Case 1: All jobs of task τ_i in the busy period are **normal** jobs that execute for $C_i(LO)$. In this case, the worst-case response time for job q occurs when the first of these job arrives at time 0, and subsequent jobs of task τ_i arrive as early as possible thereafter. (If the jobs arrived later, but still within the busy period, then job q would finish at the same time, but its response time would be smaller). Here, a job of some other HI-criticality task is assumed to arrive at time s and trigger the mode change.

Case 2: At least one job of task τ_i is an **abnormal** job that executes for $C_i(HI)$. By definition of time s, this job arrives no earlier than s.

Case 1: Every job of task τ_i contributes $C_i(LO)$, hence we have $x = 0$ and $y = q + 1$. The next step for job q is to consider all possible values for s:

$$f_i^{HI}(q)_1 = \max_{\forall s, s < f_i^{LO}(q)} \{f_i^s(q)\} \tag{28}$$

As with the arbitrary-deadline analysis for the non-clairvoyant case, it is necessary to limit the number of values of s that are considered in (28). The contribution to $f_i^s(q)$ (given by (27)) from HI-criticality tasks is decreasing in s, while the contribution from LO-criticality tasks is an increasing step function. It follows that $f_i^s(q)$ can only increase at values of s corresponding to multiples of the periods of LO-criticality tasks, hence again these are the only values of s that need to be considered. Further, given that in this case all jobs of τ_i are **normal** jobs, then if there is no mode change by time $f_i^{LO}(q)$ (given by (13)) then job q of task τ_i will have already completed execution in **normal** mode, and hence s can be restricted to the interval [0,$f_i^{LO}(q)$).

Each job's response time is given by:

$$\forall_{q, 0 \le q \le v} : \quad R_i^{HI}(q)_1 = f_i^{HI}(q)_1 - qT_i \tag{29}$$

where v is the smallest value such that $f_i^{HI}(v)_1 \le (v+1)T_i$. Finally, the worst-case response time for Case 1 is given by:

$$R_i(HI)_1 = \max_{\forall_{q, 0 \le q \le v}} \left\{ R_i^{HI}(q)_1 \right\} \tag{30}$$

If a value of q considered in (30), (29), and hence in (28), exceeds the maximum number of jobs p that can be present in a **normal** mode priority level-i busy period (where p is the smallest value such that $f_i^{LO}(p) \le (p+1)T_i$), then the value $f_i^{LO}(p)$ is used in place of $f_i^{LO}(q)$ to limit the range of values of s that are checked in (28). This holds because $f_i^{LO}(p)$ is the maximum possible continuous interval of **normal** mode

execution, after which there must be an idle instant (given that all jobs of task τ_i in the busy period are **normal** jobs). Thus any value of $s \geq f_i^{LO}(p)$ implies that there is an idle instant in the interval $[0, t)$ and hence the interval $[0, t)$ is not a valid busy period that needs to be considered.

Case 2: The number of **normal** and **abnormal** jobs of task τ_i are dependent on the values of s, t, and q. Further, by the case distinction, at least one job must be an **abnormal** job. In the following, we maximize the number of **abnormal** jobs for the given values of s and t, in order to obtain the worst-case response time for job q.

The total number of jobs of task τ_i considered in the busy period is $q+1$, at least one of which is **abnormal**. The maximum number x of **abnormal** jobs is therefore given by:

$$x = \max\left\{1, \min\left(\left\lceil \frac{t-s}{T_i} \right\rceil, q+1\right)\right\}$$

and the corresponding number of **normal** jobs is given by:

$$y = q + 1 - x$$

Note, when x is substituted into (27), the value of t is given by $f_i^s(q)$.

Next, all necessary values of s are considered:

$$f_i^{HI}(q)_2 \;=\; \max_{\forall s, s < S_i^{LO}(q)} \{f_i^s(q)\} \tag{31}$$

As with Case 1, it is necessary to limit the number of values of s that are considered in (31). The contribution to $f_i^s(q)$ from HI-criticality tasks, including τ_i, is decreasing in s, while the contribution from LO-criticality tasks is an increasing step function. It again follows that $f_i^s(q)$ can only increase at values of s corresponding to multiples of the periods of LO-criticality tasks, hence these are the only values of s that need to be considered.

By the case distinction, at least one job of task τ_i is **abnormal**, hence in (31) we need only check values of s in the interval $[0, S_i^{LO}(q))$, where $S_i^{LO}(q)$ is the latest possible start time of job q of task τ_i in **normal** mode given by the following equation (derived from the analysis in [13]):

$$S_i^{LO}(q) \;=\; qC_i(LO) \;+\; \sum_{j \in \mathbf{hp}(i)} \left(\left\lfloor \frac{S_i^{LO}(q)}{T_j} \right\rfloor + 1\right) C_j(LO) \tag{32}$$

Any value of s greater than or equal to $S_i^{LO}(q)$ would mean that at some point in the interval $[0, s]$, prior to the start of job q of τ_i, the busy period ended and hence the interval $[0, t)$ is not a valid busy period that need be considered.

Each job's response time is given by:

$$\forall_{q, 0 \leq q \leq v} : \quad R_i^{HI}(q)_2 = \min\left\{f_i^{HI}(q)_2 - qT_i, f_i^{HI}(q)_2 - s\right\} \tag{33}$$

where v is the smallest value such that $f_i^{HI}(v)_2 \leq (v+1)T_i$. Note that by the case distinction, at least one job of task τ_i is **abnormal**, it therefore follows that the final job q cannot be released prior to time s, and hence its response time is given by the minimum of the two terms in (33).

The worst-case response time for Case 2 is given by:

$$R_i(HI)_2 \;=\; \max_{\forall_{q, 0 \leq q \leq v}} \left\{R_i^{HI}(q)_2\right\} \tag{34}$$

If a value of q considered in (34), (33), and hence (31) exceeds the maximum number of jobs p that can be present in a **normal** mode priority level-i busy period (where p is the smallest value such that $f_i^{LO}(p) \leq (p+1)T_i$ and $f_i^{LO}(p)$ is given by (13)), then the value $S_i^{LO}(p)$ is used in place of $S_i^{LO}(q)$ in (31) to limit the range of values of s that are checked. This holds because $S_i^{LO}(p)$ is the maximum possible continuous priority level-i busy period of **normal** mode execution *prior* to starting any job of task τ_i. Thus, for any value of $s \geq S_i^{LO}(p)$ it follows that there is some idle instant in the interval $[0, t)$ and hence that interval is not a valid busy period.

Combining Case 1 and Case 2: The overall worst-case response time is given by combining (30) and (34).

$$R_i(HI) \;=\; \max\{R_i(HI)_1, R_i(HI)_2\} \tag{35}$$

VII. Priority Assignment

To maximize schedulability it is necessary to assign task priorities in an optimal way [20]. For arbitrary-deadline task sets scheduled under FPPS, and for constrained-deadline mixed-criticality task sets scheduled under AMC and analysed using AMC-max, it is known [8] that an optimal priority ordering can be obtained via Audsey's Optimal Priority Assignment (OPA) algorithm [4].

It is proved in [19] that it is both sufficient and necessary to show that a schedulability test meets three simple conditions in order for Audlsey's OPA algorithm to be applicable. These three conditions require that schedulability of a task according to the test is (i) independent of the relative priority order of higher priority tasks, (ii) independent of the relative priority order of lower priority tasks, (iii) cannot get worse if the task is moved up one place in the priority order (i.e. its priority is swapped with that of the task immediately above it in the priority order).

We observe that these three conditions hold for the new analyses derived for arbitrary-deadline task sets scheduled under AMC (Section V), as well as for both constrained-deadline and arbitrary-deadline task sets scheduled using semi-clairvoyant AMC (Section VI), and thus Audsley's OPA algorithm is applicable.

VIII. Dominance Relationships

In this section, we prove dominance relationships between the analyses for semi-clairvoyant and non-clairvoyant mixed criticality scheduling.

Definition 1. A schedulability test A dominates a schedulability test B (denoted by $A \rightarrow B$) if all task sets that are schedulable according to test B are also schedulable according to test A, and there exists at least one task set that is schedulable by test A, but not by test B.

Theorem 1. *The AMC-sem analysis dominates AMC-max.*

Proof: First, we note that the analysis for LO-criticality mode / **normal** mode is the same for both AMC-sem and AMC-max, and is given by (5). Thus, to prove dominance, we need only consider the analysis for HI-criticality mode / **abnormal** mode and the transition to it. Further, in both cases, the analyses of HI-criticality mode / **abnormal** mode are equivalent to considering all possible values for s. Hence, without loss of generality, we can ignore the range of values that s may take, and focus on proving dominance for any arbitrary value of s (i.e. $0 \leq s < t$).

In order to ease comparison with AMC-max (12), we simplify AMC-sem (25), forming a more pessimistic intermediate schedulability test AMC-sem2 that is by construction dominated by AMC-sem. AMC-sem2 is given by:

$$R_i(HI) = \max_{s, s \leq R_i(LO)} \{R_i^s(HI)_2\}$$

with $R_i^s(HI)_2$ given by (26), with $C_i(HI)$ in the first line instead of $C_i(LO)$. Since $R_i^s(HI)_2 \geq R_i^s(HI)_1$ (see (26)), and $R_i(LO) \geq S_i(LO)$ (see (5) and (22)), it follows that the response time given by AMC-sem2 is never less than that given by AMC-sem, and hence AMC-sem dominates AMC-sem2.

Finally, to prove the theorem, we show that AMC-sem2 dominates AMC-max. To transform the analysis for AMC-max, i.e. (11) into that given above for AMC-sem2, we need only replace $M(k, s, t)$ with $\left\lceil \frac{t-s}{T_k} \right\rceil$. Since

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

it follows that AMC-sem2 dominates AMC-max, and hence AMC-sem dominates AMC-max. Strict inequalities are required in the above equation for dominance rather than equality of AMC-sem versus AMC-max, however, these trivially occur for valid parameter settings. ■

Theorem 2. *The AMC-sem-Arb analysis dominates AMC-max-Arb.*

Proof: First, we note that the analysis for LO-criticality mode / **normal** mode is the same for both AMC-sem-Arb and AMC-max-Arb, and is given by (15). Thus, to prove dominance, we need only consider the analysis for HI-criticality mode / **abnormal** mode, including the transition to it. Further, in both cases, the analyses of HI-criticality mode / **abnormal** mode are equivalent to considering all possible values for s and all possible values for q. Hence, without loss of generality, we can ignore the range of values that s and q may take, and focus on proving dominance for any arbitrary value of s ($0 \leq s < t$) and q ($q \geq 0$).

In order to ease comparison with AMC-max-Arb (20), we simplify AMC-sem-Arb (35), forming a more pessimistic intermediate schedulability test AMC-sem-Arb2 that is by construction dominated by AMC-sem-Arb. AMC-sem-Arb2 is formed from the analysis for AMC-sem-Arb as follows: First, we remove the second term from the $\min()$ in (33), thus we have $R_i^{HI}(q)_2 = f_i^{HI}(q)_2 - qT_i$. Second, we increase the range of values of s considered in (31) to $f_i^{LO}(q)$ from $S_i^{LO}(q)$. Now comparing $f_i^{HI}(q)_2$ with $f_i^{HI}(q)_1$ in (28), we note that $f_i^{HI}(q)_2 \geq f_i^{HI}(q)_1$, since $x = 0$ in $f_i^{HI}(q)_1$ whereas $x \geq 1$ in $f_i^{HI}(q)_2$. Hence, we can eliminate any consideration of $f_i^{HI}(q)_1$ and $R_i^{HI}(q)_1$ in AMC-sem-Arb2, simplifying the entire schedulability test to:

$$x = \max \left\{ 1, \min \left(\left\lceil \frac{t-s}{T_i} \right\rceil, q+1 \right) \right\}$$

$$y = q + 1 - x$$

$$f_i^{HI}(q)_2 = \max_{\forall s, s < f_i^{LO}(q)} \{f_i^s(q)\}$$

$$R_i(HI) = \max_{\forall q, 0 \leq q \leq v} \left\{ f_i^{HI}(q)_2 - qT_i \right\}$$

Finally, to prove the theorem, we show that AMC-sem-Arb2 dominates AMC-max-Arb. To transform the analysis for AMC-max-Arb (i.e. (16) to (20)) into that for AMC-sem-Arb2 given above, we need only replace the value of x given by

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

with

$$x = \max \left\{ 1, \min \left(\left\lceil \frac{t-s}{T_i} \right\rceil, q+1 \right) \right\}$$

since the former is greater than or equal to the latter (given $D_i > 0$), it follows that the response time given by AMC-max-Arb is never less than that given by AMC-sem-Arb2. Since AMC-sem-Arb dominates AMC-sem-Arb2, in follows that AMC-sem-Arb dominates AMC-max-Arb. Strict inequalities are required between the above equations for x for dominance rather than equality of AMC-sem-Arb versus AMC-max-Arb, and trivially occur for valid parameters. ■

IX. Evaluation

In this section, we present an empirical evaluation of the schedulability tests introduced in this paper for mixed-criticality tasks. Two groups of experiments were performed. The first group focused on task sets with constrained deadlines (Section IX-B) and the second group on task sets with arbitrary deadlines (Section IX-C).

A. Task set parameter generation

The task set parameters used in our experiments followed the approach described in [8], and were randomly generated as follows:

- Task utilizations ($U_i = C_i/T_i$) were generated using UUnifast [12], providing an unbiased distribution.
- Task periods T_i were generated according to a log-uniform distribution [21] with a factor of 100 difference between the minimum and maximum possible period. This represents a spread of task periods from 10ms to 1 second, as found in many real-time applications.
- Task deadlines D_i were set equal to task periods T_i for the group of experiments on constrained-deadline task sets, and were generated according to a log-uniform distribution in the range $[0.25, 4.0]T_i$ for the experiments on task sets with arbitrary deadlines.
- The LO-criticality execution time of each task was given by: $C_i(LO) = U_i \cdot T_i$.
- The HI-criticality execution time of each task was given by: $C_i(HI) = CF \cdot C_i(LO)$ where CF is the Criticality Factor, default $CF = 2.0$.
- The probability that a generated task was of HI-criticality was given by the Criticality Proportion, default $CP = 0.5$.

In our experiments, the task set utilization was varied from 0.05 to 0.95[3]. For each utilization value, 10000 task sets were generated, (1000 in the case of experiments using the weighted schedulability measure [10]). The default cardinality of the task sets was 20. Note, the graphs are best viewed on-line in colour.

In some figures, we show the *weighted schedulability measure* $W_y(p)$ [10] for each schedulability test y as a function of some parameter p. For each value of p, this measure

[3]Utilization here is computed using the $C(LO)$ values only.

combines results for all of the task sets τ generated for all of a set of equally spaced utilization levels (0.05 to 0.95 in steps of 0.05). Let $S_y(\tau, p)$ be the binary result (1 or 0) of schedulability test y for a task set τ with parameter value p:

$$W_y(p) = (\sum_{\forall \tau} u(\tau) \cdot S_y(\tau, p)) / \sum_{\forall \tau} u(\tau) \quad (36)$$

where $u(\tau)$ is the utilization of task set τ.

The weighted schedulability measure reduces what would otherwise be a 3-dimensional plot to 2 dimensions [10].

B. Experiments with constrained-deadline task sets

The experiments on constrained-deadline task sets investigated the performance of the following schedulability tests:

(i) Clairvoyant: This test checks (using exact analysis of FPPS [25], [5]) if all of the tasks are schedulable in **normal** mode and if all of the HI-criticality tasks are schedulable in **abnormal** mode with no LO-criticality tasks executing. It ignores the mode switch, since the clairvoyant scheduler knows for any given run of the system whether the **abnormal** mode will be entered or not, and hence whether it needs to execute *any* jobs of LO-criticality tasks. This test provides an upper bound on the performance of any fixed-priority fully-preemptive scheduling scheme for mixed-criticality tasks. (We note that clairvoyant corresponds to the UB-H&L necessary test discussed in prior work [8]).
(ii) AMC-sem: introduced in Section VI.
(iii) AMC-max: defined in [8], and recapped in Section IV.
(iv) SMC: Static Mixed Criticality [7]: extends Vestal's original approach [30] with run-time monitoring. Under SMC, LO-criticality tasks continue to be released and to execute in HI-criticality mode; however, they are not required to meet their deadlines in that mode.
(v) FPPS: Standard response time analysis for FPPS defined in [25], [5]. This test requires that both LO-criticality and HI-criticality tasks are schedulable in both modes.

In each case, Audsley's Optimal Priority Assignment (OPA) algorithm [4] was used to assign priorities, ensuring an optimal assignment with respect to each schedulability test.

Figure 1 shows the percentage of task sets generated that were deemed schedulable by each of the above schedulability tests with the default parameters as described in Section IX-A. The dominance relationships between the schedulability tests are evidenced by the lines on the graph.

Observe that AMC-sem outperforms AMC-max by a significant margin, roughly halving the difference in performance between the theoretical and un-achievable Clairvoyant scheduler and AMC-max. This key observation is borne out in the following figures showing how the weighted schedulability measure varies with different task set parameters. Note, in each of the weighted schedulability experiments the default settings given in Section IX-A were used for all of the parameters that were held constant.

Figure 2 shows the results of varying the Criticality Factor ($CF = C(HI)/C(LO)$, from 1.0 to 5.5. Observe that at very low and very high values of CF, all of the mixed-criticality scheduling policies and tests have similar performance. This is because at low values of CF, schedulability is dominated by the behavior in **normal** mode, while at high values it is dominated by the behavior in the **abnormal** mode. In between, the behavior and analysis of the mode change transition becomes important and the differences between the mixed-criticality scheduling policies and analyses becomes apparent.

Figure 3 shows the results of varying the proportion of HI-criticality tasks from 5% to 95%. Observe, that if all of the tasks were LO-criticality, or all were HI-criticality, then all of the schedulability tests would have the same performance. The largest differences occur when the proportion of HI-criticality tasks is in the range 30% to 70%.

Figure 4 illustrates the impact of task set cardinality (varied from 4 to 60) on schedulability test performance. In this experiment, we fixed the number of HI-criticality tasks at exactly 50% of the total, rather than using a probability of 0.5 that each task would be HI-criticality. This was done to avoid a skew in the results for low numbers of tasks. For example, with 4 tasks, there would otherwise be a 1 in 8 chance that all tasks would be LO-criticality. This results in an 8-10% increase in weighted schedulability for all methods for 4 tasks and a 4-5% increase for 8 tasks above the values shown.

Figure 5 shows the effects of varying the range of task periods (ratio of max/min possible task period) from $10^{0.5} \approx 3$ to $10^4 = 10,000$. Here, it is interesting to note that with a small range of task periods (and hence a small range of deadlines), all of the mixed-criticality schedulability tests tend towards the same relatively low level of performance. This is partly because the interference effectively reduces to a single job of each higher priority task regardless of the mixed-criticality scheduling policy used. Nevertheless, one might expect Clairvoyant to have an advantage as it does not have to include both interference of $C(HI)$ from higher priority HI-criticality tasks and $C(LO)$ from higher priority LO-criticality tasks. Further, one might expect SMC to have lower performance, since LO-criticality tasks continue to execute in the **abnormal** mode causing interference on lower priority HI-criticality tasks. However, given that the deadlines are all very similar, the optimal priority assignment algorithm has the scope to place HI-criticality tasks at higher priorities than the LO-criticality tasks. This negates the differences in interference between Clairvoyant, AMC-sem, AMC-max, and SMC. As the range of task periods and deadlines increases, OPA can no longer achieve a priority ordering that also reflects criticality and so the relative performance of SMC degrades.

Finally, Figure 6 shows the results of varying task deadlines from 25% to 100% of the task's period. As expected, schedulability improves for all approaches as task deadlines are uniformly increased.

C. Experiments with arbitrary-deadline task sets

The experiments on arbitrary-deadline task sets investigated the performance of the following schedulability tests:

(i) Clairvoyant: As defined in the previous section, but using the standard arbitrary-deadline analysis for FPPS [29], [26] described in Section III.
(ii) AMC-sem-Arb: introduced in Section VI-B.
(iii) AMC-max-Arb: introduced in Section V.
(iv) SMC-Arb: Analysis for Static Mixed Criticality [7] using a straightforward adaptation of the standard arbitrary-deadline analysis for FPPS [29], [26].
(v) FPPS-Arb: The standard arbitrary-deadline analysis for FPPS [29], [26] described in Section III. This test requires that both LO- and HI-criticality tasks must be schedulable in both modes.

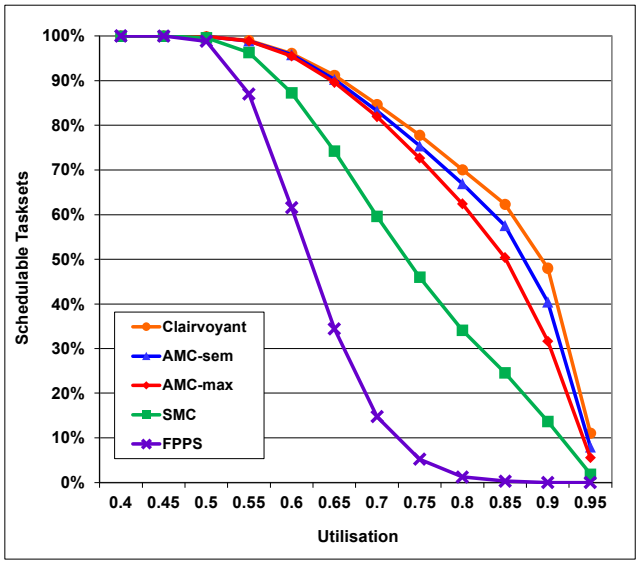

Fig. 1. Success Ratio: Percentage of schedulable task sets with utilization

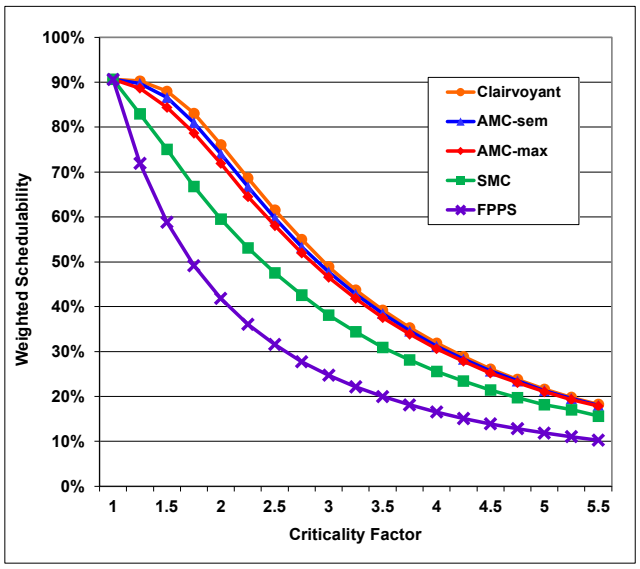

Fig. 2. Weighted Schedulability: Varying the Criticality Factor

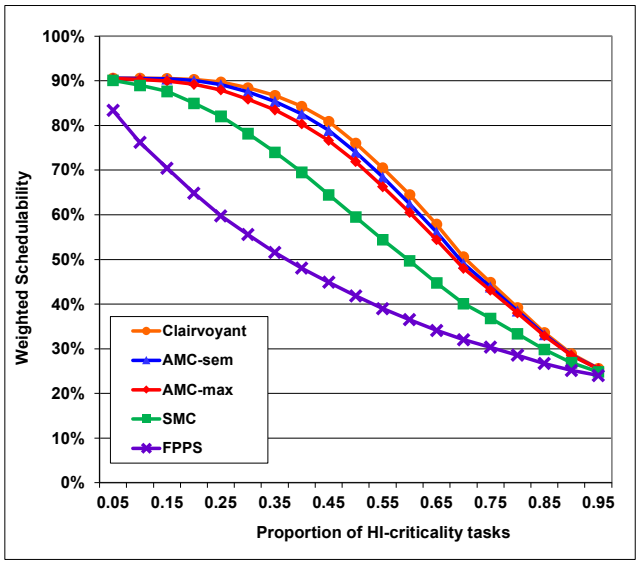

Fig. 3. Weighted Schedulability: Varying the proportion of HI-crit. tasks

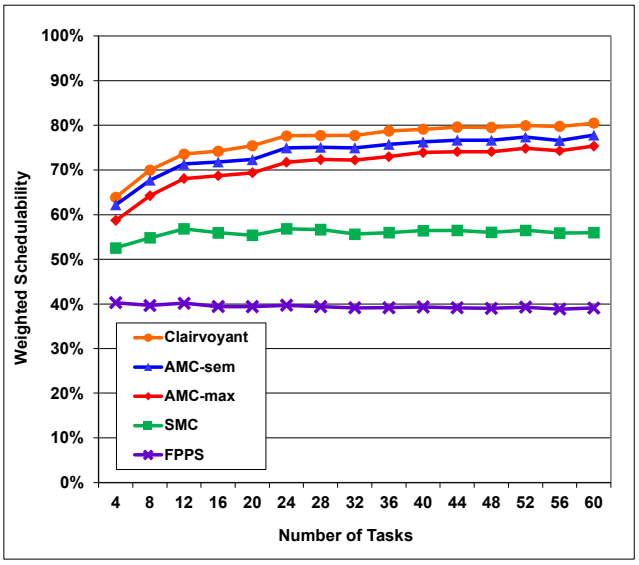

Fig. 4. Weighted Schedulability: Varying the number of tasks

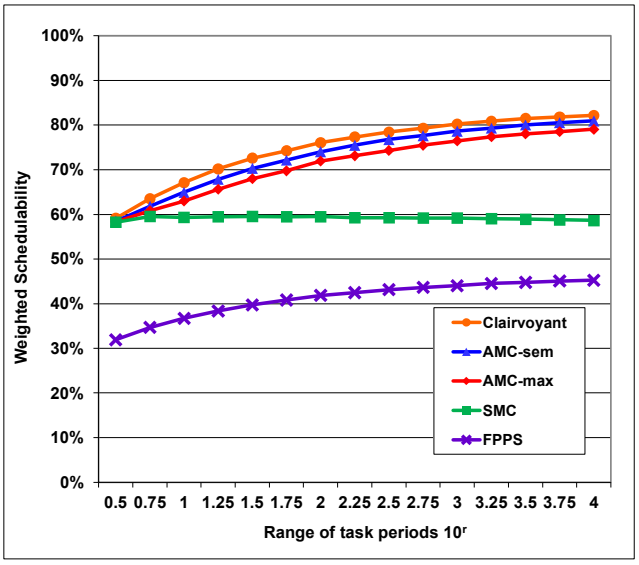

Fig. 5. Weighted Schedulability: Varying the range of task periods

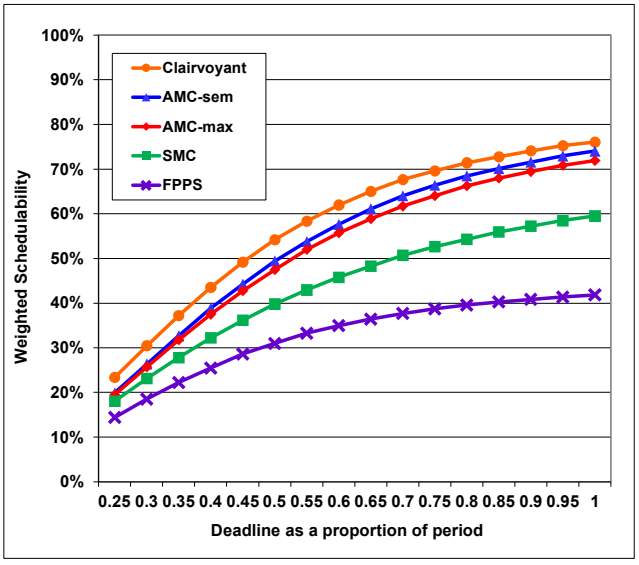

Fig. 6. Weighted Schedulability: Varying the task deadlines

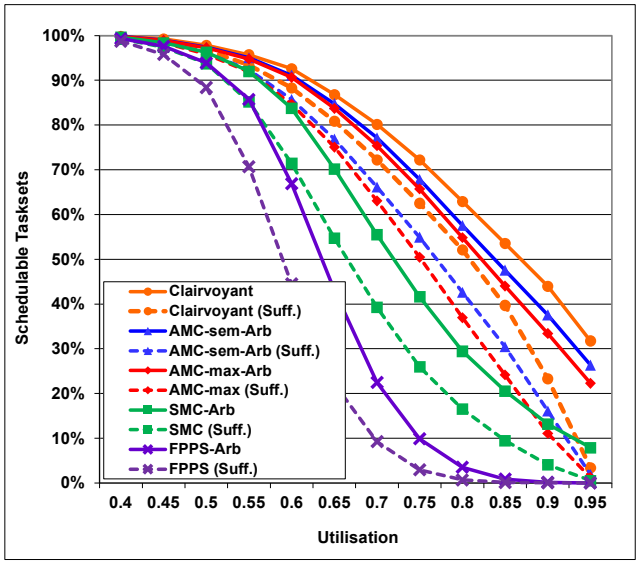

Fig. 7. Success Ratio: Percentage of schedulable task sets with utilization

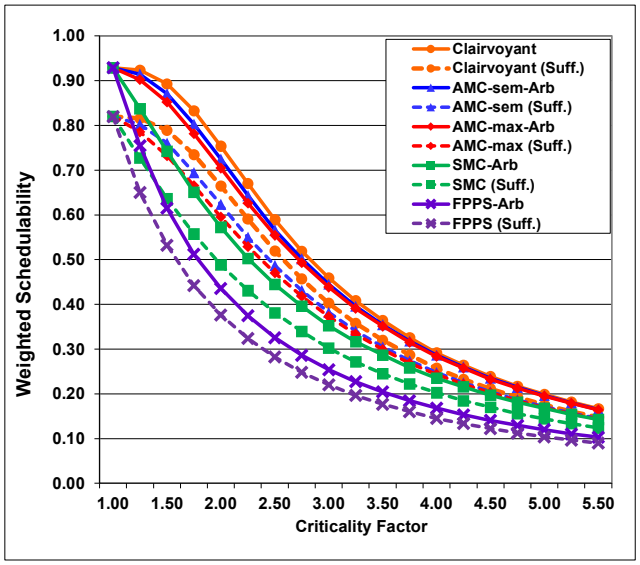

Fig. 8. Weighted Schedulability: Varying the Criticality Factor

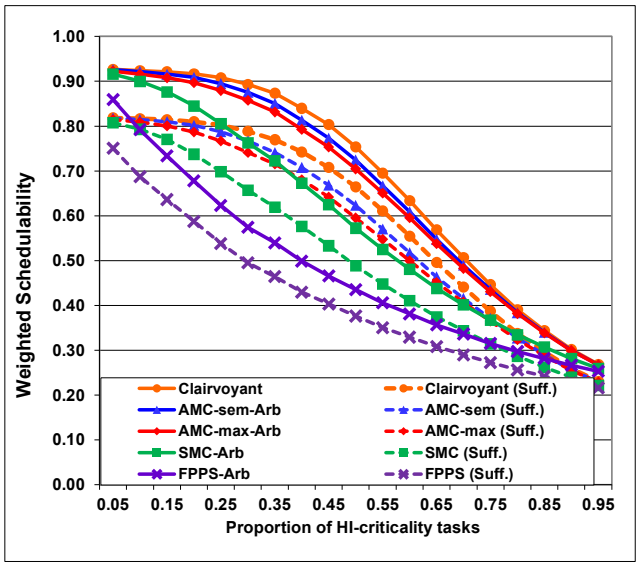

Fig. 9. Weighted Schedulability: Varying the proportion of HI-crit. tasks

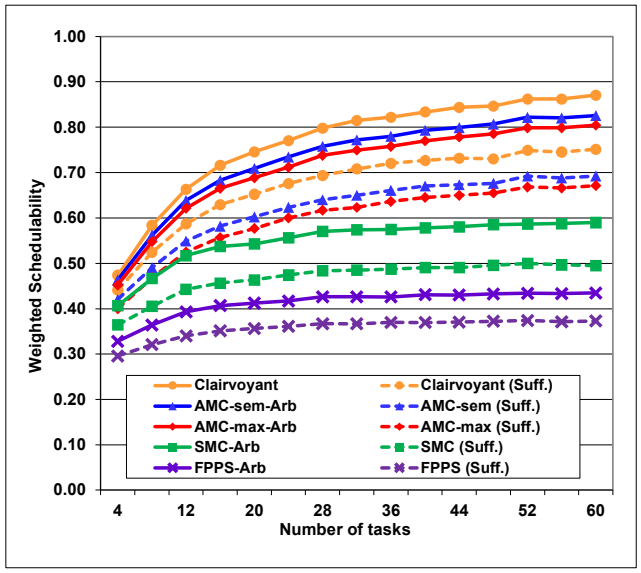

Fig. 10. Weighted Schedulability: Varying the number of tasks

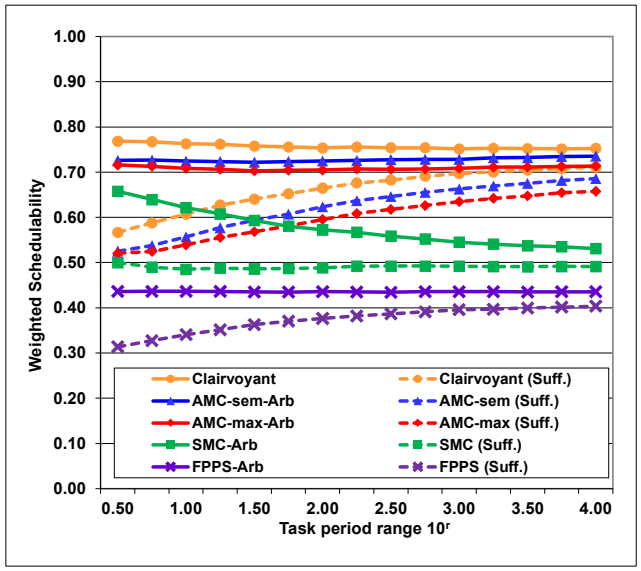

Fig. 11. Weighted Schedulability: Varying the range of task periods

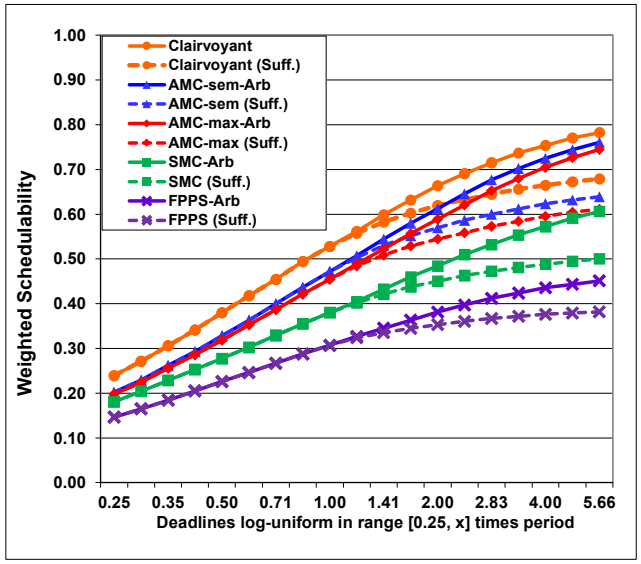

Fig. 12. Weighted Schedulability: Varying the task deadlines

In addition to the above tests for arbitrary-deadline tasks, we also explored the performance that could be obtained by utilizing schedulability tests (e.g. AMC-sem, AMC-max [8], SMC [7], FPPS [25], [5]) designed for constrained-deadline task sets.

These methods can be used to provide approximate, sufficient tests for arbitrary-deadline task sets via the simple expedient of *constraining* any deadline that is greater than the task's period to be equal to that period. In the figures, these methods are denoted by "(Suff.)" indicating a sufficient approximation. They are shown using dotted lines, with the same markers and line colors as the equivalent, more precise, arbitrary-deadline tests.

In each case, Audsley's Optimal Priority Assignment (OPA) algorithm [4] was used to assign priorities, ensuring an optimal assignment with respect to each schedulability test.

Figure 7 shows the percentage of task sets generated that were deemed schedulable by each of the above schedulability tests with the default parameters as described in Section IX-A. The dominance relationships between the schedulability tests are evidenced by the lines on the graph.

There are two important points to note from Figure 7. First, taking arbitrary deadlines into account in the analysis results in substantial improvements in schedulability when compared to an equivalent sufficient test which makes the simplifying approximation of constraining larger deadlines to be no more than the task's period. This is the case for all of the schedulability tests considered. Stated otherwise, for mixed-criticality systems scheduled using AMC-sem, AMC-max, SMC, or FPPS, increasing task deadlines beyond their periods can provide a substantial increase in guaranteed real-time performance when the schedulability tests derived in this and prior papers are employed. Second, as was the case with constrained-deadline task sets, AMC-sem-Arb outperforms AMC-max-Arb by a significant margin, roughly halving the difference in performance between the theoretical and unachievable Clairvoyant scheduler and AMC-max-Arb. Both of these two key observations are borne out in the following figures showing how the weighted schedulability measure varies with different task set parameters. Note, in each of the weighted schedulability experiments the default settings given in Section IX-A were used for all of the parameters that were held constant.

Figure 8 shows the results of varying the Criticality Factor (ratio of $C(HI)/C(LO)$) from 1.0 to 5.5. The case where $CF = 1.0$ corresponds to a single criticality level system. At that point, the performance of all of the schedulability tests that cater explicitly for arbitrary-deadline task sets is the same, similarly for the sufficient approximations assuming constrained deadlines, but at a lower level. This illustrates the underlying advantage of taking arbitrary deadlines into account.

Figure 9 shows the results of varying the proportion of HI-criticality tasks from 5% to 95%. Observe that if all of the tasks were LO-criticality, or all of the tasks were HI-criticality, then the group of more precise schedulability tests, tailored to account for arbitrary deadlines, would have the same performance. Similarly, all of the approximate (Suff.) tests would also have the same, but lower performance.

Figure 10 illustrates the impact of task set cardinality (varied from 4 to 60) on schedulability test performance. As in the constrained-deadline case, we fixed the number of HI-criticality tasks at exactly half of the total to avoid skew in the results with small numbers of tasks. As expected for fixed priority preemptive scheduling schemes, in all cases, schedulability improves with an increasing number of tasks and lower average per task utilization.

Figure 11 shows the effects of varying the range of task periods (ratio of max/min possible task period) from $10^{0.5} \approx 3$ to $10^4 = 10,000$. Observe that in the case of the sufficient approximations, when the range of task periods is small, then once constrained to the range $[0.25, 1.0]T_i$ all of the task deadlines are fairly similar and hence schedulability is low compared to the situation with a much larger range of periods (and deadlines). This is a well-known property of FPPS. It happens when the total interference from higher priority tasks in a given interval is considerably higher than that implied by their utilization [27]. With longer, arbitrary deadlines or with a larger range of task periods, this excess interference reduces and so schedulability improves.

Finally, Figure 12 shows the results of varying the range of task deadlines from $[0.25, 0.25]T_i$ to $[0.25, 5.66]T_i$ (each step on the x-axis increases the upper limit of the range by a factor of $\sqrt[4]{2}$, and hence every 4 steps it increases by a factor of 2). Note in each case the deadlines are chosen at random according to a log-uniform distribution. As expected, in all cases schedulability improves as the range of possible deadlines is expanded. Observe that while the range is no greater than $[0.25, 1.0]T_i$, then each of the arbitrary-deadline methods provides exactly the same results as its constrained-deadline (Suff.) counterpart (i.e. the solid and the dotted lines precisely overlap). Beyond that point, the arbitrary-deadline analysis confers increasingly superior performance. We note that the relative performance of the various schemes (AMC-semi, AMC-max, SMC, and FPPS) remains broadly similar to that shown in the baseline experiment (Figure 7).

X. Conclusions

In this paper, we considered the problem of scheduling mixed-criticality systems on a single processor. We studied the Adaptive Mixed Criticality (AMC) fixed-priority scheduling scheme that provides a flexible platform on which to build analysis for a range of application needs. Two significant new application requirements were addressed: tasks that have arbitrary (i.e unconstrained) deadlines; and tasks that have semi-clairvoyant behavior, where limited performance information is revealed at the time each job of a task arrives. Arbitrary deadlines are useful in increasing the schedulability of systems that are able to cope with transient overloads by employing buffers. Semi-clairvoyant behavior is a realistic option for a class of tasks that have more than one mode of operation, and where the mode of operation is dependent on the state of the system at the time at which each job of the task is released. Schedulability analysis was provided that deals with each of these use-cases, and for systems that have both arbitrary deadlines and semi-clairvoyant behavior. Comprehensive evaluations demonstrated that the new analyses out-perform existing general-purpose analyses for AMC that can be used to provide sufficient test for these two task characteristics.

Acknowledgments

The research in this paper is partially funded by the ESPRC grants, MCCps (EP/K011626/1) and STRATA (EP/N023641/1). EPSRC Research Data Management: No new primary data was created during this study.

REFERENCES

[1] K. Agrawal and S. Baruah. Intractability issues in mixed-criticality scheduling. In *Proc. Euromicro Conference on Real-Time Systems (ECRTS)*, pages 11:1–11:21, 2018.

[2] K. Agrawal, S. Baruah, and A. Burns. Semi-clairvoyance in mixed-criticality scheduling. In *Proc. IEEE Real-Time Systems Symposium (RTSS)*, pages 458–468, 2019.

[3] S. Asyaban and M. Kargahi. An exact schedulability test for fixed-priority preemptive mixed-criticality real-time systems. *Real-Time Systems Journal*, 54:32–90, 2018.

[4] N. Audsley. On priority assignment in fixed priority scheduling. *Information Processing Letters*, 79(1):39–44, 2001.

[5] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. Wellings. Applying new scheduling theory to static priority preemptive scheduling. *Software Engineering Journal*, 8(5):284–292, 1993.

[6] S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, N. Megow, and L. Stougie. Scheduling real-time mixed-criticality jobs. *IEEE Transactions on Computers*, 61(8):1140–1152, 2012.

[7] S. Baruah and A. Burns. Implementing mixed criticality systems in Ada. In *Proc. of Reliable Software Technologies - Ada-Europe*, pages 174–188, 2011.

[8] S. Baruah, A. Burns, and R. I. Davis. Response-time analysis for mixed criticality systems. In *Proc. IEEE Real-Time Systems Symposium (RTSS)*, pages 34–43, 2011.

[9] S. Baruah and B. Chattopadhyay. Response-time analysis of mixed criticality systems with pessimistic frequency specification. In *Proc. IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2013.

[10] A. Bastoni, B. Brandenburg, and J. Anderson. Cache-related preemption and migration delays: Empirical approximation and impact on schedulability. In *Proc. International Workshop on Operating Systems Platforms for Embedded Real-Time Applications*, pages 33–44, 2010.

[11] I. Bate, A. Burns, and R. I. Davis. An enhanced bailout protocol for mixed criticality embedded software. *IEEE Transactions on Software Engineering*, 43(4):298–320, 2016.

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

[13] R. Bril, J. Lukkien, and W. Verhaegh. Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption. *Real-Time Systems*, 42(1-3):63–119, 2009.

[14] A. Burns and R. I. Davis. Adaptive mixed criticality scheduling with deferred preemption. In *Proc. IEEE Real-Time Systems Symposium (RTSS)*, pages 21–30, 2014.

[15] A. Burns and R. I. Davis. Response-time analysis for mixed-criticality systems with arbitrary deadlines. In *Proc. Workshop on Mixed Criticality Systems (WMC)*, pages 13–18, 2017.

[16] A. Burns and R. I. Davis. A survey of research into mixed criticality systems. *ACM Computer Surveys*, 50(6):1–37, 2017.

[17] A. Burns and R. I. Davis. Mixed criticality systems: A review (12th edition). Technical Report MCC-1(M), available at https://www-users.cs.york.ac.uk/~burns/review.pdf, Department of Computer Science, University of York, 2019.

[18] R. I. Davis, S. Altmeyer, and A. Burns. Mixed criticality systems with varying context switch costs. In *Proc. Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2018.

[19] R. I. Davis and A. Burns. Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. In *Real-Time Systems, Volume 47, Issue 1*, pages 1–40, 2010.

[20] R. I. Davis, L. Cucu-Grosjean, M. Bertogna, and A. Burns. A review of priority assignment in real-time systems. *Journal of Systems Architecture*, 65:64–82, 2016.

[21] P. Emberson, R. Stafford, and R. I. Davis. Techniques for the synthesis of multiprocessor tasksets. In *International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, pages 6–11, July 2010.

[22] T. Fleming and A. Burns. Extending mixed criticality scheduling. In *Proc. Workshop on Mixed Criticality Systems (WMC)*, pages 7–12, 2013.

[23] O. Gettings, S. Quinton, and R. I. Davis. Mixed criticality systems with weakly-hard constraints. In *Proc. International Conference on Real-Time Networks and Systems (RTNS))*, pages 237–246, 2015.

[24] H.-M. Huang, C. Gill, and C. Lu. Implementation and evaluation of mixed criticality scheduling approaches for periodic tasks. In *Proc. Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 23–32, 2012.

[25] M. Joseph and P. Pandya. Finding response times in a real-time system. *BCS Computer Journal*, 29(5):390–395, 1986.

[26] J. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Proc. IEEE Real-Time Systems Symposium (RTSS)*, pages 201–209, 1990.

[27] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *JACM*, 20(1):46–61, 1973.

[28] D. Maxim, R. I. Davis, L. Cucu-Grosjean, and A. Easwaran. Probabilistic analysis for mixed criticality systems using fixed priority preemptive scheduling. In *Proc International Conference on Real-Time Networks and Systems (RTNS)*, pages 237–246, 2017.

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

[30] S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *Proc. Real-Time Systems Symposium (RTSS)*, pages 239–243, 2007.

[31] N. Zhang, C. Xu, J. Li, and M. Peng. A sufficient response-time analysis for mixed criticality systems with pessimistic period. *Journal of Computational Information Systems*, 11(6):1955–1964, 2015.

[32] Q. Zhao, Z. Gu, and H. Zeng. PT-AMC: Integrating preemption thresholds into mixed-criticality scheduling. In *Proc. Design Automation and Test in Europe (DATE)*, pages 141–146, 2013.

[33] Q. Zhao, Z. Gu, H. Zeng, and N. Zheng. Schedulability analysis and stack size minimization with preemption thresholds and mixed-criticality scheduling. *Journal of Systems Architecture*, 83:57–74, 2017.