Mixed Criticality Systems with Weakly-Hard Constraints

Oliver Gettings
Dept. of Computer Science
University of York, UK
oliver@cs.york.ac.uk

Sophie Quinton
INRIA Grenoble
Rhône-Alpes, France
sophie.quinton@inria.fr

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

ABSTRACT

Current adaptive mixed criticality scheduling policies assume a high criticality mode in which all low criticality tasks are descheduled to ensure that high criticality tasks can meet timing constraints derived from certification approved methods. In this paper we present a new scheduling policy, Adaptive Mixed Criticality - Weakly Hard, which provides a guaranteed minimum quality of service for low criticality tasks in the event of a criticality mode change. We derive response time based schedulability tests for this model. Empirical evaluations are then used to assess the relative performance against previously published policies and their schedulability tests.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based**]: Real-time and embedded systems

General Terms

Algorithms, Performance, Design, Theory, Verification

Keywords

Mixed Criticality, Real-Time Systems, Scheduling Theory

1. INTRODUCTION

Mixed Criticality Systems (MCS) contain components of at least two criticality levels which execute on a common hardware platform. While the sharing of hardware resources may result in a more efficient implementation over traditional isolated systems, there is a significant conflict between the required certification of components and the exploitation of temporal properties to more effectively utilise the underlying platform.

High criticality components may be required by a certification authority (CA) to meet a particular standard (e.g DO-178B/C or ISO26262) which dictates the methods used to determine the timing behaviour of tasks. The Worst-Case Execution Time (WCET) estimate of a task determined by the methods required for certification may be overly conservative compared to the values determined by a system designer using less rigorous methods. It is this pessimism that can be exploited to more efficiently utilise hardware, provided that safeguards are in place to ensure that each component in the system is guaranteed to meet its designated level of assurance.

One way in which this issue has been approached is the concept of criticality modes. Criticality modes are ordered from the lowest to the highest level of assurance. A system starts in the lowest criticality mode with timing behaviour assumed to be that determined by the system designer. Provided that each component does not exceed its allocated execution time budget, the system will remain in this mode. However, should an over-run be detected, the system will increase the criticality mode and all components assigned a criticality below this level will either be descheduled or permitted to miss their deadlines. This form of mode change is an extreme response and despite ensuring the timing constraints determined by the certification process, could lead to a loss of functionality so severe that it is not acceptable in the design of the system.

For example, consider a unmanned aerial vehicle (UAV) which contains three components, one of High (HI) criticality (e.g the flight control system) and two of Low (LO) criticality (e.g surveillance systems). The process required for certification determines that the system utilisation of the HI criticality component is 0.6. The system designer has determined this value is 0.5 in practice. The two LO criticality components which do not require certification are determined by the system designer to have utilisation of 0.25 each. Under the system designer's WCET assumptions, the system is schedulable with a utilisation of 1.0. Should an over-run of the HI criticality component be detected, the system will increase its criticality mode, the HI criticality component will be permitted to execute with a utilisation of up to 0.6 and the two LO criticality components will be descheduled. While this ensures that the HI criticality component meets the timing requirements needed for certification, the LO criticality components are still critical, as opposed to non-critical, and it may not be acceptable to lose their functionality completely. Furthermore, the utilisation in this HI criticality mode is now only 0.6, leaving unused system capacity in which LO criticality components could execute with a reduced Quality of Service (QoS), for example meeting $m-s$ out of m deadlines, by running surveillance

RTNS 2015, November 04-06, 2015, Lille, France
Copyright 2015 ACM. ISBN 978-1-4503-3591-1/15/11...$15.00.
DOI: http://dx.doi.org/10.1145/2834848.2834850

jobs less frequently or by skipping s out of m jobs.

In this paper we introduce a new approach called *Adaptive Mixed Criticality Weakly-Hard* (AMC-WH) to provide graceful degradation of low criticality tasks in the event of a criticality mode change, avoiding a complete loss of low criticality functionality. Section 2 reviews the current state of the art in terms of MCS scheduling and the low criticality abandonment problem. Section 3 outlines the system model assumed in this paper. Section 4 reviews existing schedulability analysis for MCS based on fixed priority scheduling. Section 5 introduces our new algorithm based on an existing policy called Adaptive Mixed Criticality (AMC) [3]. In section 6 we evaluate the relative performance of the new policy. Section 7 concludes with a summary and directions for future research.

2. RELATED WORK

Research into MCS verification was stimulated by Vestal's seminal paper [29] in 2007. Vestal outlined a task model based on the assumption that a task's WCET is dependent on the criticality level. That is, a task of higher criticality will have a larger, more conservative WCET estimate using methods appropriate for certification than if the task were of a lower criticality. This results in multiple WCET estimates for the same task, one for each criticality level.

Vestal's model was extended by Baruah and Vestal [4] in 2008 to lift the restriction of supporting only periodic tasks by using a sporadic task model. The major limitation of Vestal's approach is that the WCET of *all* tasks must be known for *all* criticality levels. This is not always possible in practice. For example, the process to obtain a WCET value for a task at the highest criticality level may involve expensive static code analysis and on-target timing measurements. It was recognised in the paper that a mechanism is required to deal with LO criticality jobs that attempt to exceed their designated WCET budgets to ensure that HI criticality tasks meet their deadlines.

Baruah *et al.* [3] extended Vestal's analysis to produce Static Mixed Criticality (SMC). Using run-time monitoring, overrunning LO criticality tasks are descheduled and so no longer need to be verified up to the highest system criticality level. Building further on this analysis, Baruah *et al.* [3] developed Adaptive Mixed Criticality (AMC). This new algorithm was shown to dominate all previous fixed priority preemptive scheduling algorithms for mixed criticality systems. The initial model supported two levels of criticality, however Fleming and Burns [17] demonstrated that AMC could be generalised to n-criticality levels at the expense of increased complexity.

In 2014, Burns and Davis [12] showed how AMC could use final non-preemptive regions [13]. Empirical evaluations show that this scheme improves standard AMC in terms of schedulability; however, the method requires host Real-Time Operating System (RTOS) support to control deferred preemption behaviour in addition to the basic run-time monitoring of AMC.

One of the major issues hindering the real-world application of AMC is abandonment of all LO criticality jobs in the event of a criticality mode change. This problem has been addressed in various ways. Burns and Baruah [11] proposed a revised system model that would reduce the priority of LO criticality task to background level. While this permits the execution of LO criticality tasks in the HI criticality mode, it does not guarantee a minimum level of service (i.e all deadlines can still be missed). Santy *et al.* [27] proposed a different approach for letting some LO criticality tasks execute after the system has switched to HI criticality mode, as long as their execution does not compromise the schedulability of HI criticality tasks. In effect Santy *et al.*'s algorithm delays the suspension of LO criticality tasks until the latest possible time, rather than descheduling them at the mode change, reducing the number of dropped jobs. However, as it is unknown until run-time how much slack will be available, offline analysis cannot be used to determine LO criticality task behaviour in the event of an overload.

For dynamic scheduling, Su and Zhu [28] addressed the issue of abandoning LO criticality jobs by using an elastic mixed criticality task model. They introduced a policy called Early-Release EDF (ER-EDF) which allows LO criticality tasks to be released early on the slack generated by HI criticality tasks executing for less than their certification process determined HI criticality WCET estimate.

Jan *et al.* [22] also use an elastic task model in the context of MCS by *stretching* a LO criticality task's period in order to decrease the load on the system in HI criticality mode. This permits LO criticality functionality with a reduced frequency at the expense of using online-decision algorithms to compute the required *stretch factor*.

Erickson *et al.* [16] approached the abandonment problem from the perspective of their mixed criticality multicore framework (MC^2). Using a virtual-time mechanism to alter the period of lower criticality tasks, they demonstrated the ability to provide a scalable recovery time from overload conditions without the complete loss of all LO criticality functionality.

In 2014, Fleming and Burns [18] introduced the notion of *importance* in mixed criticality systems, allowing the system designer to specify which LO criticality tasks are suspended first in the event of a criticality mode change. This provides more control over how the system should degrade. Fleming and Burns also highlighted that this approach could be used to group a number of LO criticality tasks together (as applications), providing a more realistic system model.

Gu *et al.* [20] presented a policy where the system is isolated into components with each assigned a tolerance value. If the number of HI criticality tasks executing with their HI criticality behaviour within a component does not exceed this tolerance, than the criticality mode change within the component has no effect on the schedulability of LO criticality tasks executing in other components.

Previous methods aimed at allowing LO criticality tasks to execute after a criticality mode change have mostly been best effort with limited or no guarantees over the level of service provided. For traditional (single criticality level) real-time systems there is a concept called *weakly-hard* that could help provide stronger guarantees. This concept assumes that some hard real-time tasks are in fact permitted to miss some deadlines however the number of missed deadlines must be strictly bounded.

In 1995 Hamdaoui and Ramanathan [26] introduced (m, k)-firm deadlines for streams where at least m deadlines in k consecutive invocations must be met. Priorities are dynamically assigned to streams based on the number of recently missed deadlines relative to their (m, k)-firm values. However, their (m, k)-firm scheduling algorithm does not guarantee a minimum level of service and all streams are assigned a global (m, k) value. Also in 1995, Koren and Shasha [24] worked on a different approach called *skip factor* where, when a system is overloaded, one in s invocations of a task are dropped (skipped). Bernat *et al.* [8, 7] later presented a *weakly-hard* model which in some ways is similar to the mixed criticality model of mode changes. Here, a system runs under one set of scheduling assumptions, but reverts to a *panic mode*, which guarantees that deadlines are met according to some prior offline analysis, should deadlines be at risk of being missed. The offline analysis for Bernat's model [7] is complex however as the entire hyper-period[1] of a taskset needs to be assessed.

Similar concepts have been developed in the domain of Control Systems Theory. In the event of a system overload or transient error, a control algorithm may tolerate a bounded number of missed deadlines; however, the system requires a number of met deadlines to return to stable operation [19]. *Typical Worst-Case Analysis* [21] (TWCA) exploits the fact that such overload situations are rare and usually do not occur for consecutive executions. Two response-time analyses are then performed: the standard worst-case analysis and a second, more optimistic analysis based on a model ignoring the overload. In addition, TWCA solves an ILP problem to study fined-grained effects of overload at the input of different tasks and establish how often the optimistic bound can be exceeded, expressed as a set of weakly-hard guarantees. Unlike Bernat *et al.*'s approach TWCA is not based on an analysis of the system hyper-period.

It is clear that a guaranteed, reduced Quality of Service (QoS) for LO criticality tasks in a HI criticality mode would be desirable. While weakly-hard analysis can become complex, imposing the restriction of allowing only consecutive skips in a fixed cycle prevents the problem from becoming intractable. In this paper we incorporate weakly-hard constraints into the existing AMC [3] policy to provide graceful degradation of LO criticality tasks in HI criticality mode by ensuring that they meet $m - s$ out of m deadlines, skipping s to relieve the load on HI criticality tasks.

3. SYSTEM MODEL AND TERMINOLOGY

A system consists of a single processor executing a sporadic taskset, τ, comprising N tasks, under a fixed priority preemptive scheduling policy. Each task is assigned a unique priority, P_i, according to some policy.

In traditional real-time systems, each task has a number of attributes associated with it such that $\tau_i = (T_i, D_i, C_i)$. Where T_i represents the period or minimum inter-arrival time of the task. D_i represents the relative deadline, the maximum time allowed from the task being released to completing its execution and C_i is the task's Worst-Case Execution Time (WCET). An invocation of a task is called a job. An unbounded number of consecutive jobs of task τ_i may be released at a maximum rate of $1/T_i$.

In a mixed criticality system, there is also the concept of a criticality level. Let $\mathcal{L}$ denote an ordered, finite set of criticality levels, $\{L1, L2, ..., Ln\}$ where $L1 > L2 > \cdots > Ln$, and let L_i be the designed criticality level for task τ_i. For simplicity in this paper will only consider dual criticality systems where $\mathcal{L} = \{HI, LO\}$.

The WCET value assumed for task τ_i at a specific criticality level is expressed as C_i^l where $l \in \mathcal{L}$. Vestal [29] highlighted that the WCET value is dependent on the criticality of the task; the higher the criticality the more conservative and pessimistic the estimate, that is $C_i^{HI} \geq C_i^{LO}$ for task τ_i. A task in a mixed criticality system can therefore be defined by $\tau_i = (T_i, D_i, \vec{C_i}, L_i)$ where $\vec{C_i}$ is the ordered set of C_i^l values. The utilisation of a task τ_i at criticality level l is defined as $U_i^l = \frac{C_i^l}{T_i}$, similarly the total utilisation of a taskset, τ, is defined as $U^l = \sum_{i=1}^{N}(\frac{C_i^l}{T_i})$.

The worst-case response time of a task τ_i is denoted by R_i. This represents the maximum time from the release of the task until the completion of its execution. R_i^{LO} denotes the worst-case response time of task τ_i in LO criticality mode whereas R_i^{HI} is the worst-case response time of task τ_i in HI criticality mode. R_i^* represents the worst-case response time of task τ_i during a criticality mode change $LO \rightarrow HI$. The set $hp(i)$ represents the set of tasks with a higher priority than τ_i. $hpHI(i)$ represents a subset of $hp(i)$ which contains tasks that are of higher priority and higher criticality than τ_i. Similarly, $hpLO(i)$ is the subset of $hp(i)$ that contains tasks that are of higher priority and lower criticality than τ_i. We assume a discrete time model where the time granularity $\Delta = 1$; this can be considered equivalent to one processor clock cycle.

A taskset is said to be *schedulable* with respect to a scheduling algorithm if all possible valid sequences of jobs which may be generated by the taskset can be scheduled by the algorithm without missing any deadlines.

Scheduling algorithm $\mathcal{A}$ is said to *dominate* scheduling algorithm $\mathcal{B}$ if all tasksets that are schedulable under $\mathcal{B}$ are also schedulable under $\mathcal{A}$ and there exists some taskset that is schedulable under $\mathcal{A}$, but not under $\mathcal{B}$.

In section 5 we build upon this system model to incorporate weak-hard constraints for LO criticality tasks executing in HI criticality mode.

4. EXISTING ANALYSIS

This section reviews existing scheduling policies and analysis for MCS based on fixed priority scheduling, in particular we recapitulate the analysis for AMC-rtb and AMC-max since we later build on these to provide analysis for the AMC-WH policy introduced in this paper.

4.1 Fixed Priority Preemptive Scheduling

Fixed Priority Preemptive Scheduling (FPPS) is an established scheduling policy for real-time systems. Formal analysis was first developed by Liu and Layland [25]. Response time analysis was later developed by Joseph and Pandya [23] and Audsley *et al.* [2] under the assumption of a sporadic task model.

$$R_i = C_i^{L_i} + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j^{L_j} \tag{1}$$

Equation (1) can be used to calculate the worst-case response time for each task in a MCS. This form of analysis assumes that the host Real-Time Operating System (RTOS) supports run-time monitoring to prevent LO criticality jobs from executing for longer than their C^{LO} values. For the taskset to be schedulable the following condition must hold, $\forall \tau_i : R_i \leq D_i$.

While Deadline Monotonic Priority Ordering (DMPO) has been proven optimal for uniprocessor fixed priority preemptive systems [25], Vestal showed that it is not optimal for

[1]The hyper-period is the least common multiple of all task periods.

MCS [29] in the case where run-time monitoring is not supported. This is due to an issue where a higher priority, LO criticality job executing for more than its assumed C^{LO} value may prevent a HI criticality task from executing, causing it to miss its deadline. This is referred to as criticality inversion and highlights the conflict between criticality (functional importance) and priority (scheduling importance).

4.2 Criticality Monotonic Priority Ordering

It is clear that the standard approach to scheduling fixed priority preemptive real-time systems, with no run-time monitoring support, does not suit MCS due to the challenges in dealing with multiple WCET estimates. To address the issue of criticality inversion, Criticality Monotonic Priority Ordering (CrMPO) was devised. Tasks are first partitioned by criticality and then ordered by DMPO within criticality levels. This results in higher criticality tasks being prioritised over lower criticality tasks. Equation (1) can be used to perform response time analysis, where each task assumes the execution time for its designated criticality level.

4.3 Static Mixed Criticality - NO

Static Mixed Criticality - No Run-time Support (SMC-NO) is the name given by Baruah *et al.* [3] to Vestal's original analysis for MCS [29]. As the analysis assumes no run-time support all higher priority, LO criticality tasks need to be verified up to the highest criticality level of any task to which they may cause interference.

$$R_i = C_i^{L_i} + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j^{L_i} \tag{2}$$

Equation (2) is used to perform response time analysis. Note that the WCET value for an interfering task τ_j is the same level as the task being assessed, L_i, rather than L_j as in (1). Priority assignment for SMC-NO is performed using Audsley's Optimal Priority Assignment (OPA) algorithm [1], which was proved to be optimal for SMC-NO by Dorin *et al.* [15].

4.4 Static Mixed Criticality

Static Mixed Criticality (SMC) is an extension of Vestal's analysis [29] by means of run-time monitoring of task execution times [10, 3]. If a LO criticality job attempts to execute for longer than its C^{LO} budget, it is either aborted or suspended. The response time equation of Vestal's SMC-NO is modified to become:

$$R_i = C_i^{L_i} + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \min(C_j^{L_i}, C_j^{L_j}) \tag{3}$$

The result is that, unlike Vestal's original approach, LO criticality tasks do not need to be verified to the highest criticality level of the system nor do HI criticality WCET values for LO criticality tasks need to be known. As with Vestal's version of SMC, priorities are assigned using Audsley's OPA algorithm [1].

4.5 Adaptive Mixed Criticality - rtb

Adaptive Mixed Criticality (AMC) builds upon the SMC [3] notion of run-time monitoring. That is if a job of a HI criticality task does not signal completion by its allocated C^{LO} budget, then a criticality mode change will occur. Further LO criticality jobs are descheduled and HI criticality jobs are assumed to execute for at most their C^{HI} values. Baruah *et al.* [3] developed two sufficient schedulability tests for this policy, the first being AMC - response time bound (AMC-rtb). Equation (4) is used to assess all tasks, using their C^{LO} values in LO criticality mode. The condition $\forall \tau_i \mid R_i^{LO} \leq D_i$ must hold for the system to be schedulable in LO-criticality mode.

$$R_i^{LO} = C_i^{LO} + \sum_{j \in hp(i)} \left\lceil \frac{R_i^{LO}}{T_j} \right\rceil C_j^{LO} \tag{4}$$

Equation (5) considers only HI criticality tasks using their C^{HI} values. Recall that *hpHI(i)* is the set of HI criticality tasks with higher priority than τ_i. The condition $\forall \tau_i : L_i = HI \mid R_i^{HI} \leq D_i$ must hold true for the system to be schedulable in HI criticality mode.

$$R_i^{HI} = C_i^{HI} + \sum_{j \in \mathbf{hpHI}(i)} \left\lceil \frac{R_i^{HI}}{T_j} \right\rceil C_j^{HI} \tag{5}$$

The analysis to assess the schedulability of the criticality change is a little more complex. Since a change in criticality for a HI criticality task τ_i must occur before R_i^{LO}, the interference from higher priority, LO criticality tasks (*hpLO(i)*) is bounded, as after this time LO criticality jobs would be descheduled. R^* is the response time of a HI task during a criticality change ($LO \to HI$). The first summation term represents the interference from higher priority, HI criticality tasks. The second summation term is interference from higher priority, LO criticality tasks that arrive *before* the criticality change (i.e before R_i^{LO}).

$$R_i^* = C_i^{HI} + \sum_{j \in \mathbf{hpHI}(i)} \left\lceil \frac{R_i^*}{T_j} \right\rceil C_j^{HI} + \sum_{k \in \mathbf{hpLO}(i)} \left\lceil \frac{R_i^{LO}}{T_k} \right\rceil C_k^{LO} \tag{6}$$

Audsley's OPA algorithm [1] can be used to find a priority ordering that allows a taskset τ to be schedulable according AMC-rtb analysis, if such a priority ordering exits.

4.6 Adaptive Mixed Criticality - max

AMC-rtb suffers from pessimism when considering the mode change, since it assumes that all jobs of HI criticality tasks up to time R^* may execute with their C^{HI} values.

Initially, a HI criticality job may execute in LO criticality mode. During its execution, a criticality mode change may occur and so the job must be assumed to execute up to its C^{HI} value. As it is not known exactly when the mode change will occur, AMC-rtb pessimistically assumes the worst-case, is that all jobs of all HI criticality tasks execute with their C^{HI} values before and after the criticality change. The AMC-max analysis removes this pessimism [3] by observing that there is a bounded interval in which a criticality change may affect the response time of a HI criticality job of task τ_i. That is between 0 and R_i^{LO}. If a criticality change occurs after R_i^{LO} then the job has already completed its execution and so will not be affected by the mode change. Further invocations of the task in HI criticality mode can be verified using (5).

Let y represent the time of the criticality mode change. Figure 1 illustrates a criticality mode change at time y, affecting a HI criticality task τ_i under AMC-max scheduling analysis assumptions. Note that if a criticality change is signalled while a job of τ_i is executing, it will be assumed to execute with its C_i^{HI} value.

$$R_i^y = C_i^{HI} + \sum_{k \in \mathbf{hpLO}(i)} \left(\left\lfloor \frac{y}{T_k} \right\rfloor + 1 \right) C_k^{LO} +$$

$$\sum_{j \in \mathbf{hpHI}(i)} \left(M(j, y, R_i^y) C_j^{HI} + \left(\left\lceil \frac{R_i^y}{T_j} \right\rceil - M(j, y, R_i^y) \right) C_j^{LO} \right) \tag{7}$$

$$M(j, y, t) = min \left\{ \left\lceil \frac{t - y + D_j}{T_j} \right\rceil, \left\lceil \frac{t}{T_j} \right\rceil \right\} \tag{8}$$

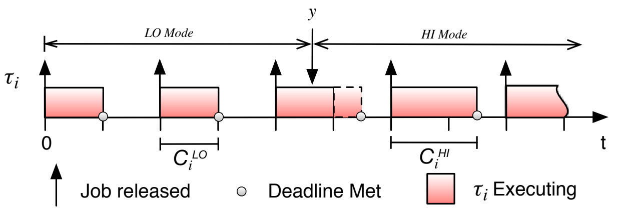

Figure 1: Criticality mode change under AMC-max

The worst-case response time of HI criticality task τ_i is calculated assuming interference from higher priority, LO criticality tasks released *before* the criticality change, y, plus interference from higher priority, HI criticality jobs *active at or after* the criticality change executing with up to their C^{HI} values and those completing *before* the criticality change with their C^{LO} values. Equation (8) calculates the maximum number of releases of a task, τ_j, after the criticality change occurs at y, up to time t [3].

The values of y that need to be assessed are bounded by 0 and R_i^{LO}, however the number of values can be large. Baruah *et al.* [3] observed that the values of y where the interference can increase correspond only to the releases of higher priority, LO criticality tasks. The worst case response time of a HI criticality task during a criticality mode change is therefore given by: $R_i^* = max(R_i^y) \forall y$ where $y \in kT_j \mid \forall j \in hpLO(i) \wedge y \leq R_i^{LO} \mid \forall k : \mathbb{N}$.

5. AMC - WEAKLY HARD

In this section we introduce Adaptive Mixed Criticality - Weakly Hard (AMC-WH). This new scheduling policy allows a number of consecutive jobs of LO critically tasks to be skipped when in HI criticality mode. This reduces the load on the system, freeing up capacity for HI criticality tasks while also providing a degraded service for LO criticality tasks, which are guaranteed to meet $m - s$ out of m deadlines, where s is the number of skips and m is the length of the cycle.

The number of skips permitted and the number of subsequent deadlines that must be met $(m - s)$ may be a requirement from the design of a control algorithm [19] or it may derive from physical properties of the system, for example with a radar altimeter it may be acceptable to drop some readings, but not to lose them altogether.

As an illustrative example consider Figure 2 which describes a LO criticality task initially executing in LO criticality mode. Task τ_k has been assigned the weakly-hard constraints that state that it must skip every 2 consecutive jobs on every 4 releases. Upon entering the HI criticality mode, the task observes these constraints, starting skipping at the first release in HI criticality mode. This cycle of skipping will repeat indefinitely providing the system remains in HI criticality mode.

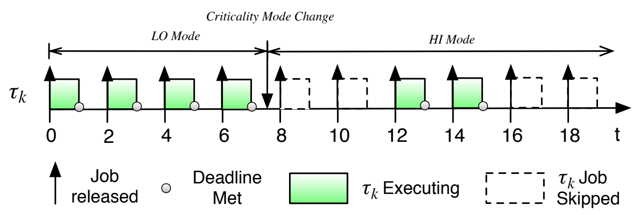

Figure 2: Example AMC-WH Execution

Building on the model in section 3, let s_k equal the number of skips assigned for the task τ_k and m_k equal the assigned cycle length in periods. Let n equal the position of a skipped job of task τ_k from the end of the cycle such that the release of a skipped job is at $m_k T_k - nT_k$, where n may take values from 1 to m_k.

In the transition from $LO \rightarrow HI$ criticality, the jobs of LO criticality task τ_k released before the mode change will continue to completion as assumed with AMC, however the next release of τ_k will be the start of the consecutive skips s_k in the cycle m_k (as shown in Figure 3).

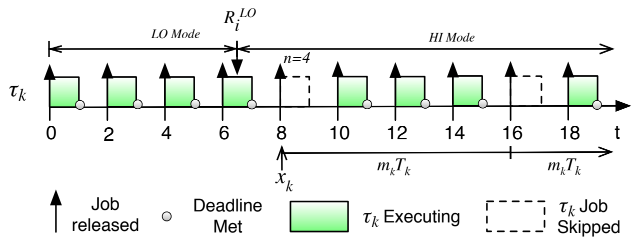

Figure 3: Criticality Change of τ_k

The maximum amount of execution of the task τ_k in an interval of length t (see Figure 4) can be expressed as the number of jobs of τ_k assuming no skips, minus the number of skipped jobs in each cycle. Equation (9) computes this value, accounting for a number of consecutive skips, s_k, where $1 \leq s_k \leq m_k$.

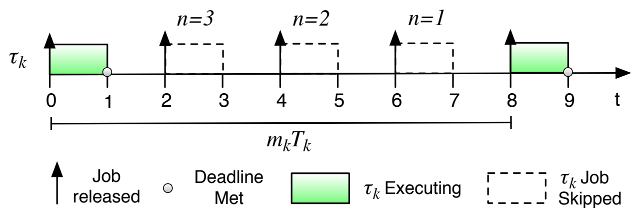

Figure 4: Cycle of τ_k

$$\left(\left\lceil \frac{t}{T_k} \right\rceil - \sum_{n=1}^{s_k} \left\lceil \frac{t - (m_k - n) T_k}{m_k T_k} \right\rceil \right) C_k \tag{9}$$

We note that the maximum amount of execution in an interval of length t occurs when the phasing of consecutive skips is at the end of a cycle i.e $n = 1, 2.., s_k$.

5.1 AMCrtb-WH

We now extend the AMC schedulability analysis (subsection 4.5) to account for these weakly hard constraints as follows.

1) Schedulability of the LO Criticality Mode

In LO criticality mode, the AMC-WH model behaves the same as AMC. HI criticality and LO criticality tasks are assumed to execute with their C^{LO} values and so the worst-case response time of each task can be calculated using (4).

2) Schedulability of the HI Criticality Mode

The worst-case response time occurs when skips s_k, are at the end of the cycle m_k for each higher priority, LO criticality task τ_k. Therefore in the HI criticality mode, the worst-case response time of a task τ_i can be expressed as its computation time, plus the interference from higher priority, HI criticality tasks executing with their C^{HI} values, plus the interference from higher priority, LO criticality tasks, minus the interference from skipped jobs hence;

$$R_i^{HI} = C_i^{L_i} + \sum_{j \in \mathbf{hpHI}(i)} \left\lceil \frac{R_i^{HI}}{T_j} \right\rceil C_j^{HI} + \sum_{k \in \mathbf{hpLO}(i)} \left(\left\lceil \frac{R_i^{HI}}{T_k} \right\rceil - \sum_{n=1}^{s_k} \left\lceil \frac{R_i^{HI} - (m_k - n)T_k}{m_k T_k} \right\rceil \right) C_k^{LO} \tag{10}$$

where $s_k < m_k$. Unlike AMC, both HI and LO criticality tasks need to be assessed using the above analysis. In the event that 100% skipping for a LO criticality task is used (i.e $s_k = m_k$), the LO criticality task in question does not need to be assessed. In addition the task will produce zero interference to lower priority tasks in HI criticality mode.

3) Schedulability of the Criticality Mode Change

Consider Figure 3 which shows the execution of a LO criticality task τ_k. If a HI criticality task τ_i reaches its R_i^{LO} without completing then a criticality mode change will be triggered. If there is a job of τ_k executing at this time then it will be allowed to complete; however the next s_k releases of τ_k, starting at time x_k, will be skipped. This is in contrast to standard weakly-hard [7] systems and aimed at increasing schedulability during the criticality mode change.

Equation (6) is modified to become (11) to assess the schedulability of the mode change for HI criticality tasks. The worst-case response time includes the interference from higher priority, HI criticality tasks, assuming C^{HI} values for all releases from $t = 0$ to R^*, plus interference from higher priority, LO criticality tasks, minus the interference from skipped jobs between x_k and R^*.

$$R_i^* = C_i^{HI} + \sum_{j \in \mathbf{hpHI}(i)} \left\lceil \frac{R_i^*}{T_j} \right\rceil C_j^{HI} + \sum_{k \in \mathbf{hpLO}(i)} \left(\left\lceil \frac{R_i^*}{T_k} \right\rceil - \sum_{n=s_k}^{m_k} \left\lceil \frac{R_i^* - (m_k - n)T_k - x_k}{m_k T_k} \right\rceil_0 \right) C_k^{LO} \tag{11}$$

where $x_k = \left\lceil \frac{R_i^{LO}}{T_k} \right\rceil T_k$ and $s_k < m_k$.

Note that following the criticality mode change, the phasing of skips of LO criticality jobs occurs at the beginning of the cycle, hence $n \in [s_k, m_k]$ is used in the summation term, rather than $n \in [1, s_k]$. During the fixed point iteration, when $R^* < x_k$, the use of $\lceil a \rceil_0$ denoting $max(\lceil a \rceil, 0)$ lower bounds $\lceil a \rceil$ by 0 to avoid including a negative number of skips. We note that no change to HI criticality mode earlier than R_i^{LO} could result in more interference according to the final term in (11). Thus (11) gives a valid upper-bound on R_i^*.

We now consider the schedulability analysis for LO criticality tasks across the criticality mode change. As it is unknown when the criticality mode change may occur, the worst-case response time for LO criticality tasks may be upper bounded using (12) which assumes that there are no skips of LO criticality jobs up to R^*.

$$R_i^* = C_i^{LO} + \sum_{j \in \mathbf{hpHI}(i)} \left\lceil \frac{R_i^*}{T_j} \right\rceil C_j^{HI} + \sum_{k \in \mathbf{hpLO}(i)} \left\lceil \frac{R_i^*}{T_k} \right\rceil C_k^{LO} \tag{12}$$

5.2 AMCmax-WH

For a HI criticality task τ_i, AMCrtb-WH is pessimistic with regards to the mode change due to assuming that all higher priority, HI criticality tasks execute with their C^{HI} values up to R_i^* (11) and also that there is no skipping of LO criticality jobs up to R_i^{LO} (4). Using the same principles as AMC-max, AMCmax-WH addresses this pessimism by taking into account the points at which a criticality change may occur. Figure 5 illustrates how a criticality mode change at time y may affect a LO criticality task (Figure 1 shows how a HI criticality task may be affected). LO criticality mode

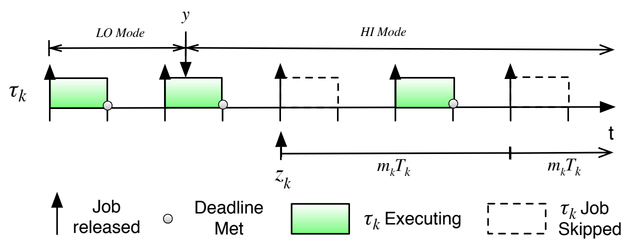

Figure 5: Criticality Change of τ_k

and HI criticality mode schedulability for all tasks can be assessed using the same approach as AMCrtb-WH (see (4) and (10)).

1) Schedulability of the Criticality Mode Change

The AMC-max analysis embodied in (7), can be modified to incorporate weakly-hard LO criticality tasks. The function $M(j, y, t)$ is the same as that used in AMC-max (8). This approach removes the pessimism in AMCrtb-WH by assuming HI criticality jobs execute with their C^{LO} values up to the criticality change at time y, at which point active and subsequent HI criticality jobs will execute up to their C^{HI} values. In addition, jobs of each LO criticality task τ_k are assumed to exhibit their weakly-hard behaviour, starting consecutive skips at z_k, the first release after the criticality change at y.

For HI criticality tasks, the points at which the triggering of a criticality change y, may affect the response time of a job are bounded by $y = [0, R^{LO})$. If the criticality change were to occur after R_i^{LO} then the job would have already completed its execution. For LO criticality tasks being assessed, y should be increased until R^* converges below the current value of y. Once $R^* < y$, increasing the time of the criticality mode change will have no effect on the job and therefore the worst-case response time must have already been obtained.

$$R_i^y = C_i^{L_i} + \sum_{k\in \mathbf{hpLO}(i)} \left(\left\lceil \frac{R_i^y}{T_k} \right\rceil - \sum_{n=s_k}^{m_k} \left\lceil \frac{R_i^y - (m_k - n)T_k - z_k}{m_k T_k} \right\rceil_0 \right) C_k^{LO}$$
$$+ \sum_{j\in \mathbf{hpHI}(i)} \left(M(j,y,R_i^y) C_j^{HI} + \left(\left\lceil \frac{R_i^y}{T_j} \right\rceil - M(j,y,R_i^y) \right) C_j^{LO} \right) \quad (13)$$

where $z_k = \left\lceil \frac{y}{T_k} \right\rceil T_k$ and $s_k < m_k$.

The worst-case for the response time for the criticality mode change can be calculated by: $R_i^* = max(R_i^y) \forall y$ where $y \in kT_j \mid \forall j \in hpLO(i) \ \wedge \ y \leq R_i^{LO} \mid \forall k : \mathbb{N}$, since these are the only values of y where the expression on the right-hand side of (13) can increase.

We note that AMC *dominates* AMC-WH since the former effectively skips all jobs of LO criticality tasks in HI criticality mode. However this means that AMC provides no service for LO criticality tasks. Thus the dominance relationships are as follows; AMC-max (AMC-rtb) dominates AMCmax-WH (AMCrtb-WH).

5.3 Priority Assignment for AMC-WH

Davis and Burns [14] formalised three Conditions for a schedulability test to be compatible with Audsley's Optimal Priority Assignment (OPA) algorithm [1]:

1. The schedulability of a task τ_k may, according to test $\mathcal{S}$, depend on any independent properties of tasks with priorities higher than τ_k, but not on any properties of those tasks that depend on their relative priority ordering.
2. The schedulability of a task τ_k may, according to test $\mathcal{S}$, depend on any independent properties of tasks with priorities lower than τ_k, but not on any properties of those tasks that depend on their relative priority ordering.
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 $\mathcal{S}$, if it was previously schedulable at the lower priority.

THEOREM 1. *AMCrtb-WH and AMCmax-WH schedulability tests comply with the above Conditions [14] and hence Audsley's OPA algorithm can be used to obtain optimal priority assignment ordering.*

PROOF. Inspection of equations (10) to (13) in section 5 shows that the schedulability of τ_k under AMC-WH depends only on independent properties of higher priority tasks. As interference is not caused by lower priority tasks, both Conditions (1) and (2) are satisfied. Consider two tasks, τ_j and τ_k with priorities, $P(v)$ and $P(w)$ respectively, where $P(v) > P(w)$. If τ_k is schedulable with priority $P(w)$ under AMC-WH and is swapped with τ_j to acquire priority $P(v)$, interference from higher priority tasks, as calculated by the summation terms, would decrease and so the worst-case response time of τ_k would become less than at $P(w)$, hence τ_k will remain schedulable. If τ_j is not schedulable at $P(v)$ and is swapped with τ_k to acquire priority $P(w)$, interference from higher priority tasks would increase and so τ_j will remain unschedulable. This satisfies the two scenarios of Condition (3) and therefore, as with SMC and AMC [3], Audsley's OPA algorithm can be used to find an optimal priority ordering with respect to both AMCrtb-WH and AMCmax-WH schedulability tests. $\square$

6. EXPERIMENTAL EVALUATION

In this section we report on an empirical evaluation used to examine the relative performance of the new scheduling policy, AMC-WH, and the associated schedulability tests introduced in section 5. A number of experiments were devised in which the new policy is compared to the previous policies reviewed in section 4.

6.1 Taskset Generation

A set of uniformly distributed utilisation values were generated using the UUnifast algorithm [9]. Periods were then assigned to each task with a log-uniform distribution between 10 and 1000. Task deadlines were assigned the same values as their periods ($D = T$). C_i^{LO} values were calculated using the utilisation equation $C_i^{LO} = U_i/T_i$ and C_i^{HI} values were assigned by multiplying the C_i^{LO} value by a criticality factor (CF). CP denotes a criticality probability, that is the probability that a particular task will be designated as HI criticality rather than LO criticality. For the success ratio experiments, 2500 tasksets were generated per utilisation level. For the weighted schedulability [5] tests, a total of 1000 tasksets were generated per utilisation level and value of the varied parameter. By default, each taskset contained 20 tasks with $CP = 0.5$ and $CF = 2.0$.

6.2 Schedulability Tests

UB-H&L - is a composite upper-bound schedulability test. Schedulability is assessed for all tasks assuming they execute with their C^{LO} values with DMPO. Separately, all HI criticality tasks are assessed, assuming their C^{HI} values, also using DMPO.

AMC-max - (subsection 4.6) is a tighter analysis to AMC-rtb, taking into account a finite set of points when the criticality change may occur [3].

AMC-rtb - (subsection 4.5) is the response time bound analysis for AMC [3].

SMC - (subsection 4.4) is SMC with run-time monitoring that deschedules overrunning LO criticality jobs.

SMC-NO - Vestal's original analysis [29] which does not have support for run-time monitoring.

AMCmax-WH - Adaptive Mixed Criticality max - Weakly Hard (subsection 5.2).

AMCrtb-WH - Adaptive Mixed Criticality response time bound - Weakly Hard (subsection 5.1).

FPPS - Fixed Priority Preemptive Scheduling (subsection 4.1). Tasks are in DMPO, ignoring criticality levels. We note that this may lead to criticality inversion however for the purpose of these experiments, run-time monitoring is assumed which prevents LO criticality tasks from exceeding their C^{LO} values. Each task assumes its $C_i^{L_i}$ value.

CrMPO - Criticality Monotonic Priority Ordering (see subsection 4.2) is where tasks are partitioned by criticality level and then DMPO is used within criticality levels. Response time analysis is then carried out on the tasks using FPPS analysis, where each task assumes its $C_i^{L_i}$ value. As HI criticality tasks have higher priorities, no run-time monitoring is needed.

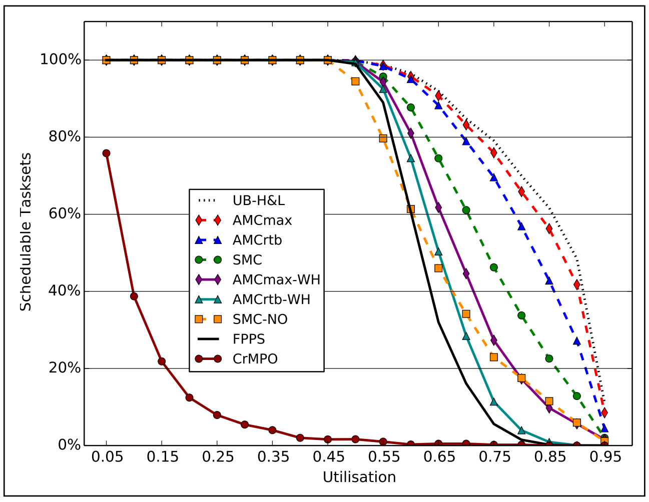

Figure 6: **Expt.1** - Percentage of Schedulable Tasksets

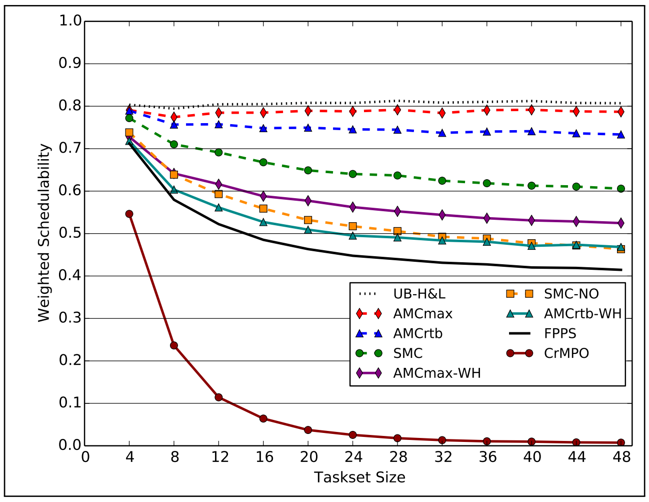

Figure 9: **Expt.4** - Varying the Number of Tasks

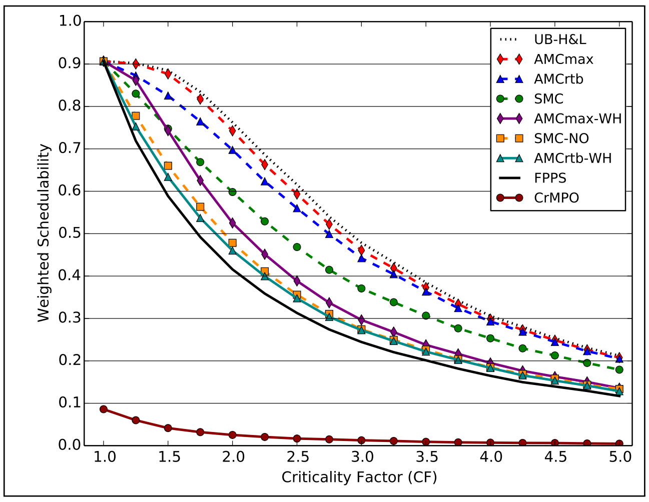

Figure 7: **Expt.2** - Varying the Criticality Factor

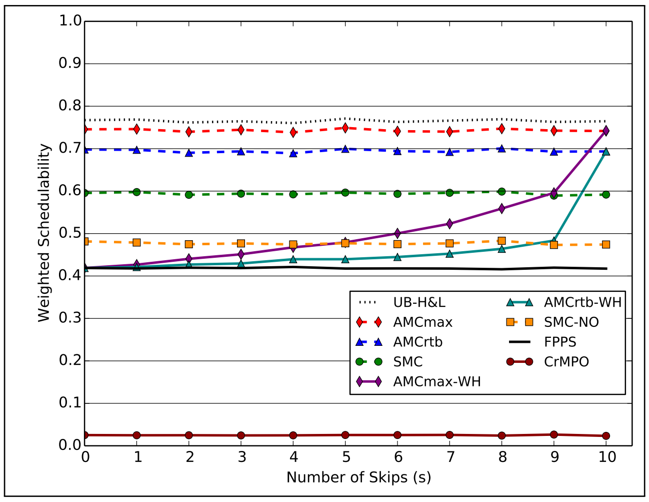

Figure 10: **Expt.5** - Varying the Number of Skips, $m = 10$

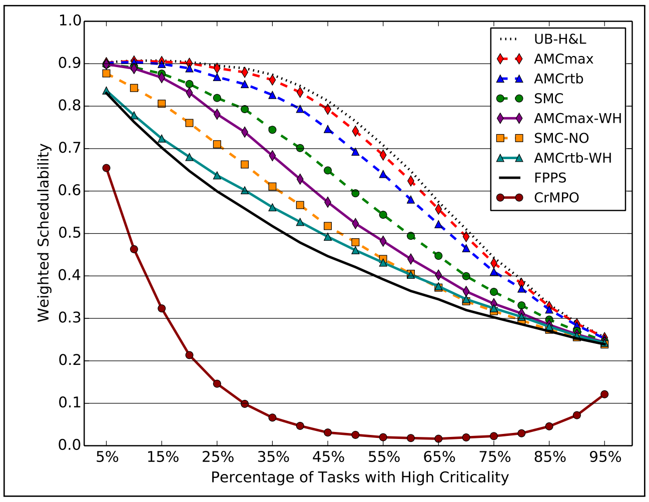

Figure 8: **Expt.3** - Varying the Criticality Mix

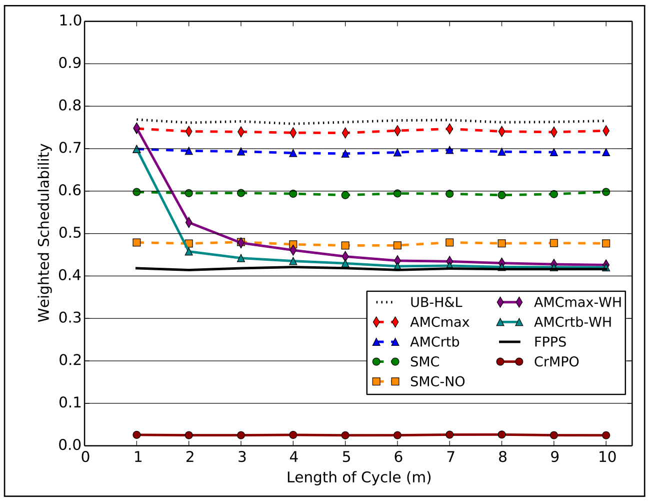

Figure 11: **Expt.6** - Varying the Cycle Length, $s = 1$

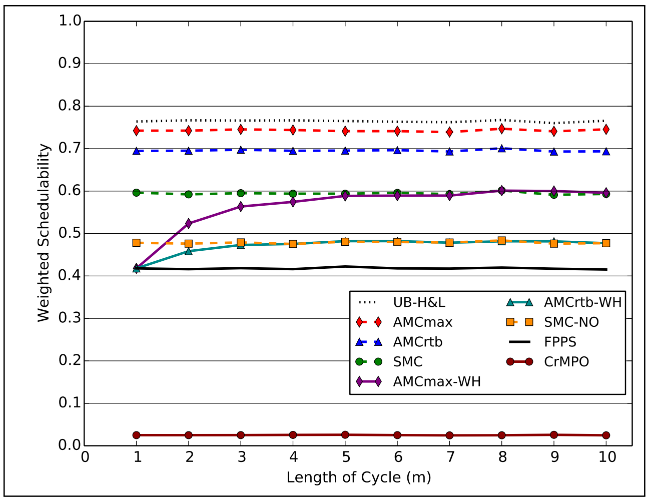

Figure 12: **Expt.7** - Varying the Cycle Length where $s = m - 1$

6.3 Experiments

Expt.1 (Figure 6) - illustrates the schedulability of the different tests at taskset utilisations.
Expt.2 (Figure 7) - shows the result of altering the Criticality Factor (CF) on schedulability. That is the multiplier between a task's C^{LO} value and its C^{HI} value.
Expt.3 (Figure 8) - varies the Criticality Probability (CP) of tasks in a taskset.
Expt.4 (Figure 9) - investigates the effect that larger tasksets have on the schedulability tests.
Expt.5 (Figure 10) - varies the number of skips in a cycle of fixed length, $m = 10$.
Expt.6 (Figure 11) - varies the cycle length, m while keeping the number of skips constant at 1.
Expt.7 (Figure 12) - varies the cycle length and skips such that $(s, m) = (m - 1, m)$.

For experiments 1 to 5, the weakly-hard constraints are set at $(s, m) = (1, 2)$ for all LO criticality tasks. This is equivalent to doubling the period while keeping the deadline the same for LO criticality tasks executing in the HI criticality mode. For experiments 2 to 4 and 6 to 8, weighted schedulability [5] is used to flatten the data from 3 dimensions to 2. Weighted schedulability is calculated via (14) where $S_\phi(\tau, p)$ is a binary test of schedulability of taskset τ with test ϕ and parameter p. Higher utilisation tasksets that are schedulable with test ϕ are more heavily weighted than lower utilisation tasksets.

$$W_\phi(p) = \big(\sum_{\forall\tau} U(\tau) * S_\phi(\tau, p)\big) / \sum_{\forall\tau} U(\tau) \qquad (14)$$

6.4 Discussion of Results

The schedulability tests are grouped into three categories. Solid lines represent tests that guarantee that at least some jobs of LO criticality tasks, assumed to execute up to their C^{LO} estimates, will meet all their deadlines in HI criticality mode. Tests that permit LO criticality tasks to miss their deadlines or deschedule LO criticality tasks in HI criticality mode are represented by dashed lines. The dotted lines on the graphs represent the upper-bounds on schedulability.

A number of points can observed by inspection of the results (Figure 6 to Figure 12). AMC dominates AMC-WH due to AMC dropping all LO criticality jobs when in HI criticality mode, therefore being schedulable at higher utilisations. The dominance of AMCmax-WH over AMCrtb-WH can be seen across all of the experiments.

Comparing the AMC-WH schedulability tests directly with tests which guarantees LO criticality task deadlines in HI criticality mode, namely CrMPO and FPPS (with run-time monitoring), there is a clear dominance. When AMC-WH is assigned a global value of 100% skips for all LO criticality tasks in HI criticality mode, that is $s = m$, the scheduling behaviour becomes that of AMC. Examining the equations in section 5 shows that the schedulability tests for AMCmax-WH and AMCrtb-WH reduce to the equations of AMC-max and AMC-rtb under this condition. This is illustrated in **Expt.5** (Figure 10) and **Expt.6** (Figure 11). At the opposite extreme, where there are 0% skips for all LO criticality tasks in HI criticality mode, both AMC-WH schedulability tests reduce to the behaviour of FPPS which can be seen in **Expt.5** (Figure 10) and **Expt.7** (Figure 12). AMC-WH is therefore a compromise between AMC and FPPS, providing scalable performance trade-offs between the quality of service of LO criticality tasks in HI criticality mode and the schedulability of the system. **Expt.7** (Figure 12) illustrates a behaviour similar to the work of Yip *et al.* [30] in terms of AMC-WH, where a task's period is extended while its relative deadline remains constant, resulting in lower utilisation.

7. CONCLUSIONS

The main contribution of this paper is the scheduling policy, AMC-WH, introduced in section 5. This policy can be used to ensure a minimum Quality of Service (QoS) for LO criticality tasks in the event of a criticality mode change, an imperative issue if such a policy is to be deployed on real systems. Empirical evaluations demonstrated that AMC-WH performs favourably with respect to existing policies, exhibiting a reasonable reduction in schedulability to accommodate the continued execution of LO criticality tasks without compromising the assurance of HI criticality tasks.

The approach provided by AMC-WH offers the flexibility for per-task constraints, allowing a system designer to dictate the level of QoS required for a particular component in the event of entering the HI criticality mode. This opens up the possibility of combining AMC-WH with the notion of importance developed by Fleming and Burns [18].

In future we aim to generalise AMC-WH to n-criticality levels, as has been done with AMC-rtb and AMC-max [17]. We also expect to generalise the model to allow all tasks to exhibit weakly-hard behaviour in any criticality mode, where each task is assigned a set of constraints (one per criticality level). This will offer additional flexibility to the system designer when deciding how the system should degrade after a criticality mode change. Another interesting avenue of research is to explore the use of sensitivity analysis to derive the weakly-hard parameters needed for schedulability under AMC-WH. This approach would allow the highest possible quality of service for LO criticality tasks. We also intend to investigate the integration of methods for the rapid recovery to LO criticality mode [6] with the AMC-WH policy.

Acknowledgements

This research was supported by EPSRC(UK) LSCITS Grant EP/F501374/1, EPSRC(UK) MCC Grant EP/K011626/1 and Rapita Systems Ltd. This collaboration was partly due to the Dagstuhl Seminar on Mixed Criticality (15121). EPSRC(UK) Research Data Management: No new primary data created.

8. REFERENCES

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

[2] N. Audsley, A. Burns, M. F. Richardson, and A. J. Wellings. Hard real-time scheduling: The deadline-monotonic approach. In *Proceedings of IEEE Workshop on Real-Time Operating Systems and Software*, pages 133–137, 1991.

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

[4] S. Baruah and S. Vestal. Schedulability analysis of sporadic tasks with multiple criticality specifications. In *Proceedings of Euromicro Conference on Real-Time Systems (ECRTS)*, pages 147–155, July 2008.

[5] A. Bastoni, B. Brandenburg, and J. Anderson. Cache-related preemption and migration delays: Empirical approximation and impact on schedulability. *In Proceedings of OSPERT*, pages 33–44, 2010.

[6] I. Bate, A. Burns, and R. I. Davis. A bailout protocol for mixed criticality systems. In *Proceedings 27th Euromicro Conference on Real-Time Systems (ECRTS) 2015*, pages 259–268, 2015.

[7] G. Bernat, A. Burns, and A. Liamosi. Weakly hard real-time systems. *IEEE Transactions on Computers*, 50(4):308–321, Apr 2001.

[8] G. Bernat and R. Cayssials. Guaranteed on-line weakly-hard real-time systems. In *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, Dec 2001.

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

[10] A. Burns and S. Baruah. Timing faults and mixed criticality systems. In *Dependable and Historic Computing*, volume 6875 of *Lecture Notes in Computer Science*, pages 147–166. Springer Berlin Heidelberg, 2011.

[11] A. Burns and S. Baruah. Towards a more practical model for mixed criticality systems. In *Workshop on Mixed-Criticality Systems (colocated with RTSS)*, 2013.

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

[13] R. Davis and M. Bertogna. Optimal fixed priority scheduling with deferred pre-emption. In *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, pages 39–50, 2012.

[14] R. Davis and A. Burns. Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. *Real-Time Systems*, 47(1):1–40, 2011.

[15] F. Dorin, P. Richard, M. Richard, and J. Goossens. Schedulability and sensitivity analysis of multiple criticality tasks with fixed-priorities. *Real-Time Systems*, 46(3):305–331, 2010.

[16] J. Erickson, N. Kim, and J. Anderson. Recovering from overload in multicore mixed-criticality systems. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2015.

[17] T. Fleming and A. Burns. Extending mixed criticality scheduling. In *Proceedings of Workshop on Mixed Criticality, IEEE Real-Time Systems Symposium (RTSS)*, pages 7–12, 2013.

[18] T. Fleming and A. Burns. Incorporating the notion of importance into mixed criticality systems. In *Proceedings of Workshop on Mixed Criticality, IEEE Real-Time Systems Symposium (RTSS)*, pages 33–38, 2014.

[19] G. Frehse, A. Hamann, S. Quinton, and M. Woehrle. Formal analysis of timing effects on closed-loop properties of control software. In *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, pages 53–62, 2014.

[20] X. Gu, A. Easwaran, K.-M. Phan, and I. Shin. Resource efficient isolation mechanisms in mixed-criticality scheduling. In *27th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 13–24, July 2015.

[21] Z. A. H. Hammadeh, S. Quinton, and R. Ernst. Extending typical worst-case analysis using response-time dependencies to bound deadline misses. In *2014 International Conference on Embedded Software, EMSOFT 2014, New Delhi, India, October 12-17, 2014*, pages 10:1–10:10, 2014.

[22] M. Jan, L. Zaourar, and M. Pitel. Maximizing the execution rate of low-criticality tasks in mixed criticality system. In *Proceedings of Workshop on Mixed Criticality, IEEE Real-Time Systems Symposium (RTSS)*, pages 43–48, 2013.

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

[24] G. Koren and D. Shasha. Skip-over: algorithms and complexity for overloaded systems that allow skips. In *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, pages 110–117, Dec 1995.

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

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

[27] F. Santy, L. George, P. Thierry, and J. Goossens. Relaxing mixed-criticality scheduling strictness for task sets scheduled with fp. In *Proceedings of Euromicro Conference on Real-Time Systems (ECRTS)*, pages 155–165, July 2012.

[28] H. Su and D. Zhu. An elastic mixed-criticality task model and its scheduling algorithm. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, pages 147–152, March 2013.

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

[30] E. Yip, M. Kuo, P. Roop, and D. Broman. Relaxing the synchronous approach for mixed-criticality systems. In *Proceedings of Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 89–100, April 2014.