CISTER
Research Center in
Real-Time & Embedded
Computing Systems

Conference Paper

Considerations on the Least Upper Bound for Mixed-Criticality Real-Time Systems

J. Augusto Santos-Jr.

George Lima

Konstantinos Bletsas*

*CISTER Research Center

CISTER-TR-151102

2015/11/03

Considerations on the Least Upper Bound for Mixed-Criticality Real-Time Systems

J. Augusto Santos-Jr., George Lima, Konstantinos Bletsas*

*CISTER Research Center

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: ksbs@isep.ipp.pt

http://www.cister.isep.ipp.pt

Abstract

Real-time mixed-criticality systems (MCS) are designed so that tasks with different criticality levels share the same computing platform. Scheduling mechanisms must ensure that high criticality tasks are safe independently of lower criticality tasks' behaviour. In this paper we provide theoretical schedulability properties for MCS by showing that: (a) the least upper bound on processor utilisation of MCS is in general null for both uniprocessor and multiprocessor platforms; (b) this bound lies in interval $[\ln 2, 2(\sqrt{2}-1)]$ if higher criticality tasks do not have periods larger than lower criticality ones; and (c) if the task of these uniprocessor systems have harmonic periods, the least upper bound reaches 1.

Considerations on the Least Upper Bound for Mixed-Criticality Real-Time Systems

J. Augusto Santos-Jr. and George Lima
Distributed Systems Laboratory (LaSiD)
Federal University of Bahia (UFBA)
Salvador-Bahia, Brazil
Email: {jamjunior,gmlima}@ufba.br

Konstantinos Bletsas
CISTER/INESC-TEC Research Centre, ISEP/IPP
Porto, Portugal
Email: ksbs@isep.ipp.pt

***Abstract*—Real-time mixed-criticality systems (MCS) are designed so that tasks with different criticality levels share the same computing platform. Scheduling mechanisms must ensure that high criticality tasks are safe independently of lower criticality tasks' behaviour. In this paper we provide theoretical schedulability properties for MCS by showing that: (a) the least upper bound on processor utilisation of MCS is in general null for both uniprocessor and multiprocessor platforms; (b) this bound lies in interval $[\ln 2, 2(\sqrt{2}-1)]$ if higher criticality tasks do not have periods larger than lower criticality ones; and (c) if the task of these uniprocessor systems have harmonic periods, the least upper bound reaches 1.**

I. Introduction

A real-time embedded system may consist of components associated with different levels of criticality. For example, take the domain of unmanned aerial vehicles (UAV). Their on-board system functionalities are specified in terms of two criticality levels. At level 1, there are mission-critical components, which are associated with image acquisition, data transfer to base station, surveillance objectives etc. The most critical functionality are at criticality level 2, where the components must ensure a safe flight. Flight permission is given only after Certification Authorities (CA) ensure that level-2 functionalities are safe whereas designers are responsible for ensuring the correctness of mission-critical functionalities.

The design of such real-time embedded systems usually requires that system components are partitioned according to their criticality so that the safety of higher criticality components are preserved independently of the behaviour of lower criticality ones. If physical partitioning is employed, higher design costs may be in place since under this strategy computing resources are not shared, which causes excessive over-provisioning. On the other hand, if the same hardware platform is shared by the system components, one must guarantee that the correctness of higher criticality components are not at stake by the behaviour of lower criticality ones. Systems designed according to this latter partitioning strategy are known as mixed-criticality systems (MSC).

This work was supported by CNPq (grant number 456193/2014-6), CAPES (grant number 99999.005354/2014-05), and National Funds through FCT/MEC (Portuguese Foundation for Science and Technology) and co-financed by ERDF (European Regional Development Fund) under the PT2020 Partnership, within project UID/CEC/04234/2013 (CISTER). Also by FCT/MEC and the EU ARTEMIS JU within projects ARTEMIS/0001/2013 - JU grant number 621429 (EMC2).

Mixed-criticality systems have recently been subject to considerable research efforts [1]. Indeed, several systems commonly found in the automotive and aerospace industries are evolving to adopt the concept of MCS with the aim of optimizing non-functional requirements such as cost, weight, size, energy consumption etc. One of the parameters usually taken into consideration by the research community and industry is the worst-case execution time (WCET) of the system tasks. CA have their own tools, methods and mechanisms to determine WCET values of critical components, which are then used for certification purposes. Designers may take advantage of the fact that these estimates are usually too conservative for implementing less critical components on the same platform. Each task of such a mixed-criticality system is then specified in terms of possibly two or more WCET estimates each one with a degree of conservativeness. If the system behaves as assumed by the designers, schedulability of the whole system is preserved. Otherwise, scheduling mechanisms must guarantee temporal correctness of high criticality level tasks, as required by CA, possibly canceling the execution of low criticality tasks. Indeed, scheduling policies and schedulability analysis play a central role in the design of MCS.

After briefly reviewing recent results in the field of MCS scheduling in Section II, we address this issue by deriving theoretical properties of MCS schedulability in terms of least utilisation bounds. These serve as a way of determining whether or not a given system will be correctly scheduled when subject to a scheduling algorithm. The precise definition used in the paper is given in Section III, which also presents the system model we adopted. We then show in Section IV a negative result stating that the least processor utilisation of MCS can be as low as zero. This holds for both uniprocessor and multiprocessor platforms. Then in Section V we identify conditions under which MCS exhibits positive processor utilisation bounds. More specifically, we show that the least utilisation bound of uniprocessor MCS for which higher criticality tasks do not have periods larger than lower criticality ones lies in interval $[\ln 2, 2(\sqrt{2}-1)]$. Further, we show that if the task of these uniprocessor systems have harmonic periods, the least upper bound on their processor utilisation reaches 100%. We finish the paper presenting our final comments in Section VI.

II. Related work

The mixed criticality scheduling problem was initially addressed by Vestal [2], who described an approach based on fixed-priority scheduling (FPS). With the focus on uniprocessor periodic task systems, this work has shown that the Rate-Monotonic priority assignment (RM) [3] is not optimal for MCS, being Audsley's optimal priority assignment algorithm [4] more suitable for this kind of system.

Baruah and Vestal [5] have extended Vestal's model by considering sporadic tasks. They have shown that the Earliest Deadline First (EDF) scheduling policy [3] does not dominate FP (and vice-versa) in MCS by exhibiting feasible systems that cannot be scheduled by EDF (respectively, FPS) whereas they can be scheduled by FPS (respectively, EDF).

Later on it has been shown that the mixed-criticality scheduling problem is strongly NP-Hard [6], [7], [8], which implies that only sufficient rather than exact analysis is possible. Since then several pieces of work in the field have been focusing on deriving scheduling strategies with a good average behaviour either on uniprocessors [9], [10], [11], [12], [13], [14], [15], [16], [17], [18] or, more recently, considering multiprocessor platforms [19], [20], [19], [21], [22]. The reader may refer to [1] for a good source of information about recent developments in the area.

In this paper we rather focus on deriving utilisation bounds for MCS. Both uniprocessor and multiprocessor platforms are considered. Usually, schedulability limits for MCS are given in terms of speed-up factor, which represents the increase in processing resources for a given system be schedulable in comparison with an optimal scheduling approach. Although such a schedulability characterization is useful, it does not give a direct connection with the schedulability of a given system implemented on a given computing platform. To the best of our knowledge, studying the mixed-criticality scheduling problem in terms of utilisation bounds has not been carried out before.

III. System model

In this section we describe our system model which is based on the Mixed Criticality System (MCS) model proposed by Vestal [2] and explored by Baruah [5] and others, as generalised for multiple criticality levels. We consider a MCS composed of a set Γ of $\mathcal{N}$ independent implicit-deadline sporadic tasks to be scheduled on $\mathcal{M}$ identical processors. Any task τ_i releases a possibly infinite sequence of jobs. As usual, we make the simplifying assumption considering that preemption and migration costs can be neglected. Since our focus is on deriving schedulability bounds, this simplification does not restrict the results presented in this paper. Indeed, the processor utilisation bound for a given system is certainly lower when migration/preemption costs are considered.

Each task τ_i is associated to a pre-specified criticality level. Task τ_i is represented by the tuple $(\overrightarrow{C_i}, T_i, L_i)$, where $\overrightarrow{C_i} = [C_i[1], \ldots, C_i[L_i]]$ is a vector of computation times for each criticality level, T_i is the minimum interarrival time between two successive jobs by the task and $L_i \in \{1, \ldots, \mathcal{L}\}$ is its criticality, where $\mathcal{L}$ is the number of criticality levels of the considered system. In this document T_i may also be called the period of task τ_i for convenience. For any two criticality levels k and l with $k < l \leqslant L_i$, it holds that $C_i[l] \geqslant C_i[k]$; in other words, WCET estimates for the same task are more pessimistic at higher criticality levels. We assume that no task τ_i executes for more than $C_i[L_i] \leqslant T_i$ time units.

We assume that system starts executing in criticality 1, that is, all its tasks always start to execute at the lowest criticality level. At run-time the criticality may increase. A MCS composed of a set of tasks Γ is said to be in criticality $k > 1$ if no job of any its task $\tau_i \in \Gamma$ has yet executed for more than $C_i[k]$ but at least one job by some τ_i has executed for more than $C_i[k-1]$.

Under the MCS assumed in this paper, and consistently with Vestal's model, it is necessary to offer guarantees for tasks with criticality level k or higher when the system is running in criticality k. That is, when analysing the system in criticality k, one has to consider only tasks with criticality k or higher. The schedulability of lower criticality tasks are not taken into consideration, as the definition below states:

Definition 1 (MCS Schedulability): A task set Γ of a MCS with $\mathcal{L}$ criticality levels is schedulable in criticality k if there is an algorithm capable of scheduling Γ such that (i) no task in Γ with criticality k or higher misses its deadline when the system runs in criticality k and (ii) additionally, for the case that $k < \mathcal{L}$, no task in Γ with criticality $k+1$ or higher misses its deadline when the system runs in criticality $k+1$. Further, if Γ is schedulable in all criticality levels $k \in \{1, \ldots, \mathcal{L}\}$, Γ is said to be schedulable.

It is worth observing that the above definition takes into consideration criticality level changes, when the system runs in criticality k and goes to level $k+1$. This is important because when running at level k, enough computing resources must be available to take care of a possible change to criticality $k+1$, which is defined in terms of more pessimistic estimates for WCET, as we now illustrate:

Example 1: Let $\Gamma = \{\tau_1 = ([1], 2, 1), \tau_2 = ([2, 10], 10, 2)\}$ be a task set to be scheduled on $\mathcal{M} = 1$ processor.

According to Definition 1, Γ given in this example is not schedulable although $\{\tau_1, \tau_2\}$ and $\{\tau_2\}$ could be feasibly scheduled if one independently considered criticality levels 1 and 2, respectively, which is illustrated in Figure 1(a) and Figure 1(b). As can be noticed, a schedule of this MCS in criticality 1 should take into consideration the execution of 5 jobs of τ_1 within any time interval of size 10; otherwise there would missed deadlines. More specifically, each of these jobs must execute within an interval of size 2, as illustrated in the Figure 1(a). Considering the system in criticality 2, since there is no slack time available, τ_2 should execute without preemption. The problem arrises when analysing a possible mode change, from criticality 1 to criticality 2. Deciding to schedule the first job of τ_1 before that of τ_2, this latter misses its deadline if the system goes to criticality 2, a scenario illustrated in Figure 1(c). Scheduling the execution of τ_2 before that of τ_1 does not work either: τ_1 would miss its deadline if the system is kept at criticality level 1.

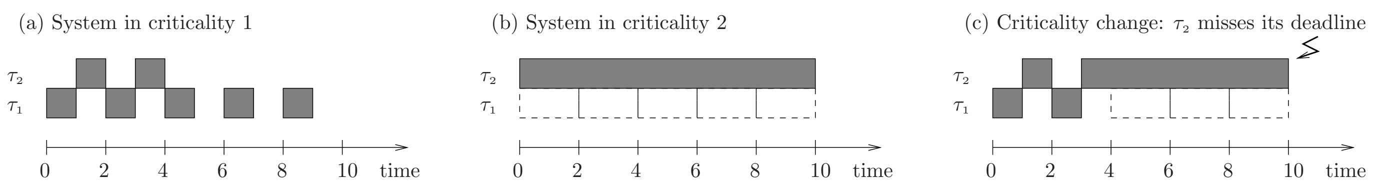

Fig. 1. Possible schedules for Example 1, an unschedulable MCS: (a) feasible schedule in criticality 1; (b) feasible schedule in criticality 2; (c) infeasible schedule when criticality changes at time 4. Solid gray boxes represent task execution whereas dashed white boxes indicate task cancelations.

The processor utilisation of a task τ_i for single-criticality task sets is usually defined as $U(\tau_i) \overset{\text{def}}{=} \frac{C_i}{T_i}$. And the total system utilisation for a single-criticality task set Γ is the sum of all task utilisations, $U(\Gamma) \overset{\text{def}}{=} \sum_{\tau_i \in \Gamma} U(\tau_i)$. For convenience, we extend these definitions for mixed-criticality systems. The utilisation of task τ_i in criticality k is denoted

$$U^k(\tau_i) \overset{\text{def}}{=} \begin{cases} \frac{C_i[k]}{T_i} & \text{if } k \leqslant L_i \\ 0 & \text{otherwise} \end{cases}$$

which makes it possible to denote the system utilisation in a given criticality as

$$U^k(\Gamma) \overset{\text{def}}{=} \sum_{i=1}^{\mathcal{N}} U^k(\tau_i)$$

For the sake of notation, we consider that $\mathcal{L} = 1$ for single-criticality systems. This allows us to consistently denote $U^1(\tau_i)$ and $U^1(\Gamma)$ as the utilisation of single-criticality task and system, respectively. For a given criticality k, we also denote the set of system tasks with criticality greater than k as

$$H^k \overset{\text{def}}{=} \{\forall \tau_i \in \Gamma : L_i > k\}$$

In the literature for single-criticality scheduling, the Least Upper Bound (LUB) on processor utilisation (or, simply, the Least Utilisation Bound) is a traditional metric for evaluating the scheduling potential of a given scheduling algorithm $\mathcal{A}$. Meaningful only in the case of implicit deadline tasks, the LUB is defined as a threshold for the system utilisation such that any system whose utilisation does not exceed that threshold is guaranteed to be schedulable under algorithm $\mathcal{A}$. In this paper we extend (the use of) this metric for the characterisation of scheduling performance of mixed criticality systems, conforming to Vestal's model:

Definition 2 (LUB): The Least Utilisation Bound in criticality $k \in \{1, \ldots, \mathcal{L}\}$ for a task set Γ (LUB(k)) with $\mathcal{L}$ criticality levels is a threshold such that if $U^k(\Gamma) \leqslant \text{LUB}(k)$, then Γ is schedulable in criticality k (according to Definition 1).

Two observations about the above definition must be made. First, the concept of LUB generalises the usual definition of LUB applied to single-criticality systems. If $\mathcal{L} = 1$, it is only required that all tasks in the system meet their deadlines because the system would never be in criticality 2 (recall Definition 1). Second, the above definition is not related to a particular scheduling algorithm. Therefore, when we say that $\text{LUB}(k) = u$, we are implicitly stating that there is some scheduling algorithm capable of feasibly scheduling the system into consideration Γ as long as $U^k(\Gamma) \leqslant u$. Defining LUB independently of specific scheduling algorithms serves for our purposes since we are interested in analysing the properties of MCS and not those of the algorithms to schedule them.

IV. Negative results

In this section characterize LUB(k) for each criticality level k considering the MCS model previously defined. More specifically, we show that $\text{LUB}(k) = 0$ for some k. Before showing this negative result, we establish a necessary condition for preserving schedulability on MCS.

Lemma 1: Assume that the system is in criticality k and may reach criticality up to $k+1$, where $1 \leqslant k \leqslant \mathcal{L} - 1$. If a job by $\tau_i \in H^k$, released at instant t, does not execute for at least $C_i[k]$ time units during interval $[t, t + T_i - (C_i[k+1] - C_i[k])]$ when the system is in criticality k, then the job might not meet its deadlines if the system reaches criticality $k+1$.

Proof: Without loss of generality assume that $\tau_i \in H^k$ arrives at time t, when the system is in criticality k, and that the system switches to criticality $k+1$ at instant $t' = t + T_i - (C_i[k+1] - C_i[k])$. Also, assume that by this time the job has executed for less than $C_i[k]$. Then, even if this job executes continuously from t' until completion, it misses its deadline if it executes for its entire WCET in criticality $k+1$. ■

The following two lemmas give negative results for multiprocessor and uniprocessor systems.

Lemma 2: An MCS composed of a set of periodic tasks Γ and with $\mathcal{M} > 1$ identical processors has $\text{LUB}(k) = 0$ for some $k \in \{1, \ldots, \mathcal{L} - 1\}$ even if Γ is schedulable in any criticality level greater than k.

Proof: We will construct an unschedulable system in criticality k with $\mathcal{M} > 1$ processors and $\mathcal{N}$ tasks, where $\mathcal{N} = \mathcal{L} = \mathcal{M} + k$ and with system utilisation in criticality k barely above zero. Let the system tasks be defined as follows:

$$([\epsilon, \ldots, \epsilon, T'\epsilon], T', i), \quad \underbrace{1 \leqslant i < \mathcal{N} - \mathcal{M}}_{k-1 \text{ tasks with criticality } <k}$$

$$([T'\epsilon, \ldots, T'\epsilon, C'], T', k), \quad \underbrace{i = k = \mathcal{N} - \mathcal{M}}_{1 \text{ task with criticality } =k}$$

$$\left(\left[\epsilon^2, \ldots, \epsilon^2, \underbrace{C''}_{C_i[k]}, 1, \ldots, 1\right], 1, i\right), \quad \underbrace{\mathcal{N} - \mathcal{M} < i \leqslant \mathcal{N}}_{\mathcal{M} \text{ tasks with criticality } >k}$$

where $T' = 1 + \epsilon$, $C' = 1$, $C'' = \epsilon$ and $\epsilon \in (0, 0.5)$. Since H^k contains $\mathcal{M}$ tasks, this system is clearly schedulable in $\mathcal{M}$ processors when in criticality greater than k. We proceed to show that τ_k may miss its deadline when the system runs in criticality k.

Without loss of generality assume that every task in the system arrives at time t. By inspecting the behavior of the sytem over the time interval $[t, t' = t + T']$, it follows that the processor time available in all $\mathcal{M}$ processors within this interval is

$$\sum_{i=1}^{\mathcal{M}} (t' - t) = T'\mathcal{M} = (1+\epsilon)\mathcal{M} \tag{1}$$

Given that during the time interval $[t, t']$ every job of tasks in H^k must execute for up to C'' time units (as follows from Lemma 1) and there are up to two jobs of each such a task during $[t, t']$, the processor time that must be provided to H^k within this time interval to guarantee that the corresponding deadlines will be met cannot be lower than

$$\sum_{\tau_i \in H^k} 2C_i[k] = 2C''\mathcal{M} = 2\epsilon\mathcal{M} \tag{2}$$

From Equations (1) and (2) it follows that the remaining processing capacity that can be used by τ_k does not exceed $(1-\epsilon)\,\mathcal{M} < C'\mathcal{M}$. Nevertheless, note that the $\mathcal{M}$ tasks in H^k cannot be interfered during their execution, otherwise they would not be able to meet their deadlines when the system goes to criticality greater than k. This means that all these $\mathcal{M}$ tasks must execute in parallel in $\mathcal{M}$ processors. Therefore a deadline miss may occur with the system utilisation being

$$U^k(\Gamma) = \sum_{\tau_i \in \Gamma} U^k(\tau_i) = 1 + \epsilon\mathcal{M} \quad \Rightarrow \quad \frac{U^k(\Gamma)}{\mathcal{M}} = \frac{1}{\mathcal{M}} + \epsilon$$

Assuming that $\epsilon = \frac{1}{\mathcal{M}+1}$, the above equation yields

$$\lim_{\mathcal{M}\to\infty} \frac{U^k(\Gamma)}{\mathcal{M}} = \lim_{\mathcal{M}\to\infty} \left(\frac{1}{\mathcal{M}} + \frac{1}{\mathcal{M}+1}\right) = 0$$

■

Interestingly, the result given by Lemma 2 is independent of the scheduling algorithm. We next show that similar conclusions can be drawn for uniprocessor systems.

Lemma 3: A uniprocessor MCS composed of a set of periodic tasks Γ has $\text{LUB}(k) = 0$ for some $k \in \{1, \ldots, \mathcal{L}-1\}$ even if Γ is schedulable in any criticality level greater than k.

Proof: Once again, it suffices defining an unschedulable task set in criticality k, which is schedulable in criticality $k+1$ or higher but whose utilisation in criticality k can be as close to zero as we wish. Let Γ contain $\mathcal{N} = \mathcal{L}$ tasks, defined as:

$$\left(\left[\epsilon^4, \ldots, \epsilon^4, \epsilon^3\right], \epsilon, i\right), \qquad \underbrace{1 \leqslant i < k}_{k-1 \text{ tasks with criticality } <k}$$

$$\left(\left[\epsilon^3, \ldots, \epsilon^3, 2\epsilon^2\right], \epsilon, k\right), \qquad \underbrace{i = k}_{1 \text{ task with criticality } k}$$

$$\left(\left[\epsilon^2, \ldots, \epsilon^2, \epsilon, 1-\epsilon^2\right], 1, i\right), \qquad \underbrace{i = k+1}_{1 \text{ task with criticality } k+1}$$

$$\left(\left[\frac{\epsilon^2}{\mathcal{N}}, \ldots, \frac{\epsilon^2}{\mathcal{N}}, C'\right], 1, i\right), \qquad \underbrace{k+1 < i \leqslant \mathcal{N}}_{\mathcal{N}-k-1 \text{ tasks with criticality } >k+1}$$

where $C' = \frac{\epsilon^2}{\mathcal{N}-k}$ and $\epsilon \in (0, 0.5)$. We first observe that when this system is in criticality higher than k, at most $\mathcal{N} - k$ must run and all these tasks have the same deadline, which is equal to 1. The computation time jointly required by these tasks cannot be greater than

$$(\mathcal{N} - k - 1)\frac{\epsilon^2}{\mathcal{N} - k} + 1 - \epsilon^2 < 1$$

Hence, this system is schedulable in criticality $k+1$ or higher. Now let us turn our attention to the schedulability of τ_k.

Without loss of generality, assume that a job by each task in Γ arrives at time instant t and that the system is in criticality k at t. Let us analyse the system behaviour during time interval $[t, t' = t + \epsilon^2 + \epsilon]$. Note that during the this time interval task τ_{k+1} must execute for up to $C_{k+1}[k] = \epsilon$ time units otherwise it may miss its deadline if the system goes to criticality $k+1$ within $[t, t']$ (recall the necessary condition stated in Lemma 1). This means that the time left for executing other tasks is no more than $t' - t - C_{k+1}[k] = \epsilon^2 + \epsilon - \epsilon = \epsilon^2$. As the processor time that τ_k requires during time interval $[t, t']$ can be as much as $2\epsilon^2$ time units, a deadline miss may occur with the system utilisation being

$$U^k(\Gamma) = \sum_{\tau_i \in \Gamma} U^k(\tau_i) = \frac{\epsilon}{1} + \frac{2\epsilon^2}{1} + \frac{\epsilon^2}{\mathcal{N}}(\mathcal{N} - k - 1) < 3\epsilon^2 + \epsilon$$

And so $\lim_{\epsilon\to 0} U^k(\Gamma) = 0$, as required. ■

We now generalize the results stated in Lemmas 2 and 3:

Theorem 1: An MCS composed of a set of periodic/sporadic tasks Γ to be scheduled on $\mathcal{M} \geqslant 1$ identical processors has $\text{LUB}(k) = 0$ for some $k \in \{1, \ldots, \mathcal{L}-1\}$.

Proof: From Lemmas 2 and 3 we know that the theorem follows for periodic tasks even when Γ meets its deadlines at criticality level $k+1$. Since the periodic task model is a special case of the sporadic task model, we conclude that in these cases the theorem also follows for sporadic tasks.

The claim trivially holds when Γ misses a deadline at criticality level $k+1$. Assume for example that $U^{k+1}(\Gamma) = \mathcal{M} + \epsilon$ and that $U^k(\Gamma) = \epsilon$, where ϵ is a small positive constant. That is, Γ is not schedulable in criticality k and $k+1$ due to the assumption that the system misses a deadline at criticaliy level $k+1$. This completes the proof, since $\lim_{\epsilon\to 0} U^k(\Gamma) = 0$. ■

A way of circumventing this negative result is examined in next section, which gives a positive upper bound on $\text{LUB}(k)$ for uniprocessor systems if additional conditions are satisfied.

V. Positive Results

We now focus on uniprocessor MCS characterized by:

Hypothesis 1: For any two tasks τ_i and τ_j, if $L_i > L_j$ then $T_i \leqslant T_j$.

We consider systems conforming to Hypothesis 1 because of their interesting theoretical properties. We notice that not all systems may be in line with this hypothesis, though. This is because, unlike task priorities, which are controlled by the designer, the task interarrival times and criticalities are typically specified as input parameters to the design process and may not be changed by designers.

We next show that even under Hypothesis 1, no algorithm is able to schedule uniprocessor systems with LUB higher than $2(\sqrt{2}-1)$:

Theorem 2: No uniprocessor MCS composed of a set of tasks Γ has $\mathrm{LUB}(k)$ greater than $2\left(\sqrt{2}-1\right)$ for some $k \in \{1,\ldots,\mathcal{L}-1\}$.

Proof: We will show that Γ is not schedulable in criticality k although it has utilisation barely above $2\left(\sqrt{2}-1\right)$. For this, we will consider that Γ is schedulable in any criticality greater than k. If at criticality level $k+1$, Γ misses a deadline, the theorem could be verified by simply presenting such a task set so that $U^k(\Gamma) < 2\left(\sqrt{2}-1\right)$. Hence, let Γ contain $\mathcal{N} = \mathcal{L}$ tasks defined as follows:

$$\begin{aligned}
&\left(\left[\epsilon,\ldots,\epsilon,\sqrt{2}\epsilon\right],\sqrt{2},i\right), && \underbrace{1 \leqslant i < k}_{k-1 \text{ tasks with criticality } <k}\\
&\left(\left[\sqrt{2}\epsilon,\ldots,\sqrt{2}\epsilon,C'\right],\sqrt{2},k\right), && \underbrace{i = k}_{1 \text{ task with criticality } =k}\\
&\left(\left[\epsilon,\ldots,\epsilon,C'',1-\epsilon\right],1,i\right), && \underbrace{i = k+1}_{1 \text{ task with criticality } =k+1}\\
&\left(\left[\frac{\epsilon}{\mathcal{N}},\ldots,\frac{\epsilon}{\mathcal{N}},\frac{\epsilon}{\mathcal{N}-k}\right],1,i\right), && \underbrace{k+1 < i \leqslant \mathcal{N}}_{\mathcal{N}-k-1 \text{ task with criticality } >k+1}
\end{aligned}$$

where $C' = 2-\sqrt{2}+2\sqrt{2}\epsilon$, $C'' = \sqrt{2}-1-\epsilon$ and $\epsilon \in (0, 0.5)$. As for criticality higher than k, note that the deadline of all $\mathcal{N}-k$ tasks with criticality $k+1$ or higher have deadline equal to 1. Their joint computation time can be bounded by

$$(\mathcal{N}-k-1)\frac{\epsilon}{\mathcal{N}-k} + 1 - \epsilon < 1$$

which means that this system is schedulable in criticality $k+1$ or higher. The schedulability of τ_k, however, cannot be guaranteed, as we now show.

Without loss of generality, assume that a job by each task in Γ arrives at time instant t and that the system is in criticality k at t. Analyzing the system behaviour during time interval $[t, t' = t+\sqrt{2}]$, we note that task τ_{k+1} arrives up to two times during the interval $[t,t']$ and the second job of the task must complete up to C'' time units of execution until time instant t' (by Lemma 1). Thus, τ_{k+1} must execute for up to $2C''$ time units otherwise it may miss its deadline if the system goes to criticality $k+1$ within $[t,t']$. This means that the time left for executing other tasks is no more than $t'-t-2C'' = 2-\sqrt{2}+2\epsilon < C'$. As the processor time that τ_k requires during time interval $[t,t']$ can be as much as C' time units, a deadline miss may occur with the system utilisation being

$$U^k(\Gamma) = \frac{C'}{\sqrt{2}} + \frac{C''}{1} + \frac{(\mathcal{N}-k-1)\epsilon}{\mathcal{N}} < 2\left(\sqrt{2}-1\right) + 2\epsilon$$

Since $\lim_{\epsilon\to 0} U^k(\Gamma) = 2\left(\sqrt{2}-1\right)$, the Theorem holds. ■

Although Theorem 2 gives us an upper bound on $\mathrm{LUB}(k)$, its result alone is not of any help given that we know from the previous section that $\mathrm{LUB}(k)$ can be as low as zero. However, under Hypothesis 1, we show that $\mathrm{LUB}(k) \geqslant \ln 2$ by studying the schedulability of the MCS into consideration. This is done based on properties associated with single-criticality systems Γ_k whose tasks are obtained from the tasks with criticality at least k of an MCS task set Γ. More formally,

$$\Gamma_k \overset{\text{def}}{=} \bigcup_{\tau_i \in H^{k-1}} \{\tau_i' = (C_i[k], T_i)\} \tag{3}$$

A first (naïve) attempt to use Γ_k to infer the schedulability of Γ would be to check whether or not each Γ_k, $k \in \{1,\ldots,\mathcal{L}\}$, is schedulable. As we have seen in Example 1, such an approach does not work; changes in the system criticality during execution must be taken into account. We address the criticality change issue turning our attention to a specific scheduling algorithm, namely Rate-Monotonic (RM) [3], and considering MCS systems complying with Hypothesis 1. Interestingly, in this scenario the RM priority assignment equals criticality monotonic priority assignment [11]. Under these conditions, the schedulability of Γ can indeed be verified via checking the schedulability of Γ_k, as the following theorem states:

Theorem 3: Let Γ be an MCS task set with $\mathcal{L}$ criticality levels in line with Hypothesis 1. Γ is schedulable by RM if Γ_k, as defined by (3), is schedulable by RM for all $k \in \{1,\ldots,\mathcal{L}\}$.

Proof: We first observe that if the system always runs in criticality $k = 1$, the theorem trivially holds; otherwise, Γ_1 would not be schedulable by RM. Now consider the case where the system is in criticality $k > 1$. For this case, we proceed by contradiction assuming that some task in Γ misses its deadline at some instant t when Γ is scheduled by RM with the system running at criticality $k \in \{2,\ldots,\mathcal{L}\}$. This task must belong to H^{k-1} since the schedulability of lower criticality tasks is not taken into consideration when the system runs in criticality k.

From Hypothesis 1, we know that tasks in $\Gamma \backslash H^{k-1}$ cannot interfere in the execution of tasks in H^{k-1} since lower criticality tasks have greater periods and, therefore, lower priorities according to RM. From our contradiction assumption, we also know that any task τ_i that executes before t requires no more than $C_i[k]$. Furthermore, by definition, the execution time of all tasks τ_i' in Γ_k is also upper bounded by $C_i[k]$. This means that some task in Γ_k would also miss its deadline at or before t if Γ_k was scheduled by RM, which is a contradiction.

As the above reasoning holds all $k \in \{1,\ldots,\mathcal{L}\}$, it follows that no task in H^{k-1} can miss its deadline when the system runs in criticality k. Therefore, Γ is schedulable by RM. ■

An interval that characterizes $\mathrm{LUB}(k)$ can now be given:

Theorem 4: If uniprocessor MCS with $\mathcal{L}$ criticality levels is schedulable by RM and Hypothesis 1 holds, then $\mathrm{LUB}(k) \in \left[\ln 2, 2\left(\sqrt{2}-1\right)\right]$ for all $k \in \{1,\ldots,\mathcal{L}\}$.

Proof: Let Γ be the considered task set. The fact that Γ is schedulable by RM means that Γ_k (defined according to (3)) is schedulable by RM for all $k \in \{1,\ldots,\mathcal{L}\}$ (by Theorem 3), where $U^1(\Gamma_k) = U^k(\Gamma)$. We know from [3] that any task set schedulable by RM has a least utilisation upper bound not lower $\ln 2$. The upper bound of $2\left(\sqrt{2}-1\right)$ follows directly from Theorem 2. ■

Our final observation comes from the fact that for some systems, task periods have an harmonic relation [23]. That is, task periods are multiple from one another. Since it is known that under this condition RM can feasibly schedule uniprocessor systems with utilisation as high as 100%, MCS schedulability also exhibits such a high bound if Hypothesis 1 holds:

Theorem 5: Let Γ be the task set of a uniprocessor MCS for which Hypothesis 1 holds. If the periods of tasks in Γ exhibit an harmonic relation, then $\mathrm{LUB}(k) = 1$ for all $k \in \{1, \ldots, \mathcal{L}\}$.

Proof: Consider that $U^k(\Gamma) \leqslant 1$ for all $k \in \{1, \ldots, \mathcal{L}\}$. Based on Definition 2, we need to show that any such a Γ is schedulable by some scheduling policy (we use RM for this purpose) provided that the harmonic relation between task periods holds. Since we are interested in determining $\mathrm{LUB}(k)$, task sets for which utilisation values are greater than 1 are not to be considered since they are not schedulable anyway.

Let Γ_k be defined according to (3) for all $k \in \{1, \ldots, \mathcal{L}\}$. By construction, both the task periods in Γ_k are harmonic and $U^1(\Gamma_k) \leqslant 1$. We know from [24] that all these Γ_k are schedulable by RM. By Theorem 3, this implies that Γ is also schedulable by RM. ■

VI. Conclusion

We have studied in this paper schedulability properties for mixed-criticality systems. After defining the concept of least upper bound on processor utilisation in a given criticality k, $\mathrm{LUB}(k)$, we have shown that both uniprocessor and multiprocessor systems exhibit null $\mathrm{LUB}(k)$ in general. We have also shown that $\mathrm{LUB}(k)$ lies within interval $\left[\ln 2, 2\left(\sqrt{2}-1\right)\right]$ for uniprocessor systems whose lower criticality tasks have periods not lower than those of higher criticality ones; and that if those systems are made of tasks with harmonic task periods, $\mathrm{LUB}(k)$ reaches 1.

An interesting issue for further investigation is about whether or not some scheduling policy can reach the theoretical limit of $2\left(\sqrt{2}-1\right)$ when non-harmonic tasks are considered. As for multiprocessor systems, it would be also important to identify conditions under which positive values for $\mathrm{LUB}(k)$ can be guaranteed. Future research steps may explore these and other related questions.

References

[1] A. Burns and R. Davis, "Mixed criticality systems: A review," Department of Computer Science, University of York, Tech. Rep. MCC-1(e), February 2015. [Online]. Available: http://www-users.cs.york.ac.uk/burns/review.pdf

[2] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*, Dec 2007, pp. 239–243.

[3] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogram in a hard real-time environment," *Journal of ACM*, vol. 20, no. 1, pp. 46 – 61, 1973.

[4] N. C. Audsley, "On priority asignment in fixed priority scheduling," *Inf. Process. Lett.*, vol. 79, no. 1, pp. 39–44, May 2001. [Online]. Available: http://dx.doi.org/10.1016/S0020-0190(00)00165-4

[5] S. Baruah and S. Vestal, "Schedulability analysis of sporadic tasks with multiple criticality specifications," in *Real-Time Systems, 2008. ECRTS '08. Euromicro Conference on*, July 2008, pp. 147–155.

[6] S. Baruah, "Mixed criticality schedulability analysis is highly intractable," Tech. Rep., 2009. [Online]. Available: http://www.cs.unc.edu/˜baruah/Submitted/02cxty.pdf

[7] S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, N. Megow, and L. Stougie, "Scheduling real-time mixed-criticality jobs," *Computers, IEEE Transactions on*, vol. 61, no. 8, pp. 1140–1152, Aug 2012.

[8] S. Baruah, "Semantics-preserving implementation of multirate mixed-criticality synchronous programs," in *Proceedings of the 20th International Conference on Real-Time and Network Systems*, ser. RTNS '12. New York, NY, USA: ACM, 2012, pp. 11–19. [Online]. Available: http://doi.acm.org/10.1145/2392987.2392989

[9] S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, S. Van der Ster, and L. Stougie, "The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems," in *Real-Time Systems (ECRTS), 2012 24th Euromicro Conference on*, July 2012, pp. 145–154.

[10] S. K. Baruah, V. Bonifaci, G. D'Angelo, A. Marchetti-Spaccamela, S. Van Der Ster, and L. Stougie, "Mixed-criticality scheduling of sporadic task systems," in *Proceedings of the 19th European Conference on Algorithms*, ser. ESA'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 555–566. [Online]. Available: http://dl.acm.org/citation.cfm?id=2040572.2040633

[11] S. Baruah, A. Burns, and R. Davis, "Response-time analysis for mixed criticality systems," in *Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd*, Nov 2011, pp. 34–43.

[12] D. de Niz, K. Lakshmanan, and R. Rajkumar, "On the scheduling of mixed-criticality real-time task sets," in *Real-Time Systems Symposium, 2009, RTSS 2009. 30th IEEE*, Dec 2009, pp. 291–300.

[13] N. Guan, P. Ekberg, M. Stigge, and W. Yi, "Effective and efficient scheduling of certifiable mixed-criticality sporadic task systems," in *Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd*, Nov 2011, pp. 13–23.

[14] K. Lakshmanan, D. de Niz, and R. Rajkumar, "Mixed-criticality task synchronization in zero-slack scheduling," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2011 17th IEEE*, April 2011, pp. 47–56.

[15] H. Li and S. Baruah, "Load-based schedulability analysis of certifiable mixed-criticality systems," in *Proceedings of the Tenth ACM International Conference on Embedded Software*, ser. EMSOFT '10. New York, NY, USA: ACM, 2010, pp. 99–108. [Online]. Available: http://doi.acm.org/10.1145/1879021.1879035

[16] F. Santy, L. George, P. Thierry, and J. Goossens, "Relaxing mixed-criticality scheduling strictness for task sets scheduled with FP," in *Real-Time Systems (ECRTS), 2012 24th Euromicro Conference on*, July 2012, pp. 155–165.

[17] 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*, March 2013, pp. 147–152.

[18] S. Baruah, V. Bonifaci, G. D'angelo, H. Li, A. Marchetti-Spaccamela, S. Van Der Ster, and L. Stougie, "Preemptive uniprocessor scheduling of mixed-criticality sporadic task systems," *J. ACM*, vol. 62, no. 2, pp. 14:1–14:33, May 2015. [Online]. Available: http://doi.acm.org/10.1145/2699435

[19] H. Li and S. Baruah, "Outstanding paper award: Global mixed-criticality scheduling on multiprocessors," in *Real-Time Systems (ECRTS), 2012 24th Euromicro Conference on*, July 2012, pp. 166–175.

[20] R. Pathan, "Schedulability analysis of mixed-criticality systems on multiprocessors," in *Real-Time Systems (ECRTS), 2012 24th Euromicro Conference on*, July 2012, pp. 309–320.

[21] S. Baruah, B. Chattopadhyay, H. Li, and I. Shin, "Mixed-criticality scheduling on multiprocessors," *Real-Time Systems*, vol. 50, no. 1, pp. 142–177, 2014. [Online]. Available: http://dx.doi.org/10.1007/s11241-013-9184-2

[22] Z. Al-bayati, Q. Zhao, A. Youssef, H. Zeng, and Z. Gu, "Enhanced partitioned scheduling of mixed-criticality systems on multicore platforms," in *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*, Jan 2015, pp. 630–635.

[23] J. Herman, C. Kenna, M. Mollison, J. Anderson, and D. Johnson, "RTOS support for multicore mixed-criticality systems," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2012 IEEE 18th*, April 2012, pp. 197–208.

[24] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: exact characterization and average case behavior," in *Real Time Systems Symposium, 1989., Proceedings.*, Dec 1989, pp. 166–171.