Mixed-Criticality with Integer Multiple WCETs and Dropping Relations: New Scheduling Challenges

Federico Reghenzani
federico.reghenzani@polimi.it
Politecnico di Milano
Milano, Italy

William Fornaciari
william.fornaciari@polimi.it
Politecnico di Milano
Milano, Italy

ABSTRACT

Scheduling Mixed-Criticality (MC) workload is a challenging problem in real-time computing. Earliest Deadline First Virtual Deadline (EDF-VD) is one of the most famous scheduling algorithm with optimal speedup bound properties. However, when EDF-VD is used to schedule task sets using a model with additional or relaxed constraints, its scheduling properties change. Inspired by an application of MC to the scheduling of fault tolerant tasks, in this article, we propose two models for multiple criticality levels: the first is a specialization of the MC model, and the second is a generalization of it. We then show, via formal proofs and numerical simulations, that the former considerably improves the speedup bound of EDF-VD. Finally, we provide the proofs related to the optimality of the two models, identifying the need of new scheduling algorithms.

CCS CONCEPTS

• **Computer systems organization** → *Embedded software*; **Real-time systems**; *Reliability*.

KEYWORDS

mixed-criticality, real-time, scheduling, fault tolerance

ACM Reference Format:
Federico Reghenzani and William Fornaciari. 2023. Mixed-Criticality with Integer Multiple WCETs and Dropping Relations: New Scheduling Challenges. In *28th Asia and South Pacific Design Automation Conference (ASPDAC '23), January 16–19, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3566097.3567851

1 INTRODUCTION

The real-time computing community, since the paper by Vestal in 2007 [10], researched how to analyze and implement Mixed-Criticality (MC) systems. The motivation behind the use of MC is the excessive pessimism of the Worst-Case Execution Time (WCET) estimations. Indeed, modern computing platforms have complex characteristics (e.g., multi-core, multi-level caches, advanced pipelines) which make a tight WCET estimation very difficult. Indeed, WCET analyses, that use the software and hardware descriptions to build a timing model of the task, cannot cope with such complexity and need to introduce several approximations to compute a solution in a feasible amount of time. These approximations led to very large and over-estimated WCET. The MC model tries to mitigate this problem by modeling the WCET of each task as a set of estimations performed at different assurance levels, from the most pessimistic (but safe) to the less pessimistic (but possibly under-estimated) WCET. This idea comes from the *criticality* concept of many safety-critical standards. A criticality level is assigned to each function of a safety-critical system depending on the level of safety required (in real-time computing terms, the *function* can be mapped to the *task* concept). For instance, in avionics, the Design Assurance Levels (DALs) are used to classify the functions, from A to E. Functions with a DAL of A are the most critical functions, whose malfunction can have catastrophic consequences, while DAL E includes all the functions which have no effect on flight safety.

1.1 MC Systems in the Real World

Industrial systems need to satisfy regulations in order to obtain the certification by the authorities, especially if they are safety-critical systems. Unfortunately, certifying MC systems is problematic due to several issues. Focusing on timing, the certifiability of MC systems exploiting scheduling strategies developed by the scientific community in the last 15 years is controversial. Indeed, the majority of MC scheduling algorithms are based on the following concept: if any task overruns a WCET estimation (for instance, the less pessimistic one), all the tasks with lower criticality are killed or are scheduled with only a best-effort policy, in order to accommodate the execution requirements of the higher criticality tasks. While this behavior appears to make sense, it does not comply with most of the safety-critical standards [6], that instead require all the tasks to be functional independent [9], regardless of their criticality levels.

Besides timing requirements, standards require a system to meet well-defined goals of dependability. Due to the intrinsic unreliability of hardware components, fault tolerance methods must be taken into account to satisfy the dependability constraints. Software techniques to detect and recover from hardware faults are attractive from an industrial perspective because reduce the amount of custom hardware needed to achieve the dependability goals. However, such fault tolerance techniques introduce additional challenges for hard real-time schedulability. We will show that software mechanisms for fault tolerance and the MC model are closely interrelated.

1.2 State of the Art

The seminal paper on the MC real-time model is the article by Vestal [10] in 2007. Burns et al. [5] surveyed all the following papers on the mixed-criticality topic, including the scheduling analyses. Among them, we recall two scheduling algorithms, i.e., the Own

ASPDAC '23, January 16–19, 2023, Tokyo, Japan

ACM ISBN 978-1-4503-9783-4/23/01…$15.00
https://doi.org/10.1145/3566097.3567851

Criticality-Based Priorities (OCBP) [3] and the Earliest Deadline First Virtual Deadlines (EDF-VD) [1]. The latter has been thoroughly analyzed in a subsequent article by the same authors [2], which also proved EDF-VD to be speedup optimal for 2 criticality levels, as later described in Section 2. For this reason, this is also the main scheduling algorithm considered in this work. Regarding the certifiability problem, Völp [11], Esper et al. [7], and Ernst et al. [6] analyzed (and criticized) the MC scheduling theory with a careful look at the certification process, i.e., the compliance with safety-critical standards, highlighting the issues previously described in Section 1.1. In the last years, several approaches tried to solve the problem by providing graceful degradation [8] or sacrificing different metrics such as energy instead of killing the tasks [4]. A recent article [9] proposed to exploit the MC model and theory to schedule task sets containing fault recovery tasks, claiming that the approach should comply with safety-critical standards.

1.3 Contributions

In this article, we provide theoretical contributions to the scheduling of real-time task sets satisfying two novel models, derived from the traditional MC model and inspired by the MC fault tolerance challenge proposed by Reghenzani et al. [9]. The first model, named Mixed-Criticality with Integer Multiple WCETs (MC-IMW), exploits a specific derivation of the WCET values. We show that EDF-VD has better speedup bounds when applied to MC-IMW rather than a traditional MC. Nonetheless, we prove that EDF-VD remains speedup optimal for 2 criticality levels. We also run a numerical simulation to obtain the results when criticality levels are large and difficult to compute analytically. The second model is, instead, a generalization of the MC model that uses the *Dropping Relation (DR)* concept. In this case, we show that state-of-the-art solutions, including one based on EDF-VD, are not optimal. These results make room for the development of novel scheduling algorithms and strategies.

2 BACKGROUND

An MC system is composed of a set of tasks $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$. A task τ_i is an abstract entity representing a program and identified by the following tuple: $(T_i, \bar{C}_i, D_i, \chi_i)$, where T_i is the period or inter-arrival time, $\bar{C}_i$ the list of WCETs, D_i the relative deadline, and χ_i the criticality level. In this article, we focus on implicit deadline task sets, i.e., $D_i = T_i$ for all the tasks. At each period, a new job of each task is released and must complete its execution by the following period. The criticality level $\chi_i \in \{1, 2, ..., L\}$ represents the level of assurance to guarantee for the task, where 1 is the lowest level (least critical) and L the highest (most critical) levels. In MC systems, conversely to traditional real-time systems, the WCET is not a single value but a list of χ_i WCET values $\bar{C}_i = \{C_i(1), C_i(2), ..., C_i(\chi_i)\}$, estimated at difference assurance levels. Only the highest WCET value $C_i(L)$ is guaranteed to not under-estimate the real WCET. The system, at each time instant, runs in a *mode* $M(t) \in \{1, 2, ..., L\}$ with $M(0) = 1$. At time t, only tasks with $\chi_i \geq M(t)$ are admitted to scheduling, while the others are killed or dropped[1]. When a job of a task τ_i overruns its current WCET $C_i(M(t))$ and $\chi_i > M(t)$, then the system switches to the next mode $M(t+1) = M(t) + 1$. This concept of system mode is the one most criticized for not complying with safety-critical standards (see Section 1.1).

[1] In this article, we interchangeably use the terms *kill* and *drop* to identify the action of terminating all the jobs of a task or preventing the release of new jobs of a task.

Utilization. The *utilization* value represents the execution requirement in percentage. In particular, in MC systems:

Definition 2.1 (Task utilization). The utilization of a task τ_i at a given criticality level ℓ is defined as: $u_i(\ell) = \frac{C_i(\ell)}{T_i}$. This value is well-defined only if $\ell \leq \chi_i$.

Definition 2.2 (System utilization). The system utilization at a given criticality level ℓ is defined as: $U_i(\ell) = \sum_{\tau_i : \chi_i \leq \ell} u_i(\ell)$.

When the system has only one criticality level, it degenerates to normal real-time scheduling. In such a case, to verify the schedulability, it is possible to use the test of the optimal scheduling algorithm EDF: $U_1(1) \leq 1$.

Scheduling MC task sets. The *necessary* condition for the schedulability of any MC task set under any scheduling algorithm is [2]:

$$\max_{k=1,...,L} \sum_{\ell=k}^{L} U_\ell(k) \leq 1 \tag{1}$$

The rationale is immediate: at any system mode, the set of non-dropped tasks, with WCET at that criticality level, must be at least schedulable by considering only that single mode, i.e. the utilization must be ≤ 1. This condition is also a *sufficient* condition for a clairvoyant scheduling algorithm: if the scheduling algorithm knows in advance whether the system switches or not to a given system mode, the algorithm will schedule the tasks active at that mode according to EDF, and Eq. (1) guarantees schedulability.

Speedup bounds. The Earliest Deadline First (EDF) scheduling algorithm is able to schedule any MC task set provided that the system runs on a processor with speedup $\sigma = L$. More precisely, EDF would be able to schedule all the MC task sets of criticality level L if the tasks run on a processor L times as fast, i.e., the original values $C_i(j)$ become $C_i^*(j) = C_i(j)/L$. Instead, an optimal scheduling algorithm would be able to schedule all the schedulable task sets satisfying Eq. (1) with $\sigma = 1$. It has been proved that an optimal non-clairvoyant scheduling algorithm for MC systems does not exist [2]. For this reason, researchers introduced the concept of *speedup optimal*.

Definition 2.3 (Speedup Optimal Scheduling Algorithm). An MC scheduling algorithm is said *speedup optimal* if it is able to schedule all the schedulable task sets according to Eq. (1) with a processor having speedup $\sigma = \bar{\sigma}$ and no other non-clairvoyant scheduling algorithm with $\sigma = \bar{\sigma}' < \bar{\sigma}$ exists.

EDF-VD. EDF-VD [1] is a speedup optimal scheduling algorithm for a generic MC task set with 2 criticality levels. If the task set has implicit deadlines, the speedup bound is $\sigma = \frac{4}{3}$ for 2 criticality levels. EDF-VD works in a 2-criticality system as follows:

(1) A scaling parameter $\lambda < 1$ is computed as $\lambda = \frac{U_2(1)}{1 - U_1(1)}$.
(2) Tasks are scheduled with EDF, however, tasks of criticality level 2 are scheduled according to a modified deadline, i.e., $\hat{D}_i = \lambda \cdot D_i$. This new virtual deadline gives them a higher priority than they would have had.

(3) If any 2-level task overruns its $C_i(1)$, then the system switches to higher mode, and EDF schedules only 2-level tasks according to their unmodified deadlines D_i.

For an arbitrary number of criticality levels, the scheduling algorithm is similar, with the exception that multiple system mode switches can occur. The article of Baruah et al. [2] provides all the details of the scheduling algorithm in such a case, including how to compute λ when $L > 2$.

2.1 Fault Tolerance via Re-Execution

The re-execution mechanism is a fault recovery technique that deals mainly with transient faults, for instance, bit flips. This mechanism works as follows: the job execution is monitored via appropriate fault detection algorithms (such as control-flow-graph signatures, acceptance tests, plausibility checks, etc.); if a transient fault is detected, then the job is restarted to re-perform the same computation[2]. From a real-time perspective, the fault is detected, in the worst-case, at the end of the computation, i.e., after C_i time units. After the job restart, the job can run for other C_i time units, and if another fault occurs, the job has to execute for other C_i time units, and so on. The maximum number of allowed re-executions depends on the failure requirements for the specific tasks, which are usually mapped to the criticality level and certification requirements.

3 MIXED-CRITICALITY WITH INTEGER MULTIPLE WCETS MODEL

In this section, we propose a specialization of the MC traditional model: the Mixed-Criticality with Integer Multiple WCETs (MC-IMW) model. The model is identical with the MC model but the WCET values cannot be freely chosen. Indeed, the WCET values must be multiples of the lowest-criticality WCET and the number of the criticality level. For instance, if $C_i(1) = 10$, then $C_i(2) = 20$, $C_i(3) = 30$, and so on. Therefore, this model adds the constraint $C_i(k) = k \cdot C_i(1)$ for all $k \in 1, ..., \chi_i$. Note that the computation of $C_i(k)$ can also be performed in the opposite direction: let us assume that a task τ_i with criticality level $\chi_i = 3$, for which we have a pessimistic estimation of the WCET $C_i(3)$. We can then compute $C_i(2)$ and $C_i(1)$ by properly scaling $C_i(3)$, i.e., $C_i(2) = C_i(3) \cdot 2/3$ and $C_i(1) = C_i(3)/3$. The constraints $C_i(k) = k \cdot C_i(1)$ of the MC-IMW model allows us to rewrite the task and system utilizations:

$$u_i(k) = \frac{c_i(k)}{T_i} = k\frac{c_i(1)}{T_i} \tag{2}$$

$$\begin{aligned} U_i(k) &= \sum_{\tau_j:\chi_j=i} u_j(k) = \sum_{\tau_j:\chi_j=i} k u_j(1) \\ &= k \sum_{\tau_j:\chi_j=i} u_j(1) = kU_i(1) \end{aligned} \tag{3}$$

This rewriting of the system utilization with the integer multiple version allows us, in the following sections, to simplify the analytical calculus and improve the EDF-VD speedup bound.

Previous articles left unspecified the engineering process to determine the exact values for the WCET computed at different criticality levels. MC-IWM enforces this choice and perfectly fits fault tolerance systems running re-execution tasks as fault recovery mechanisms, previously explained in Section 2.1. Indeed, C_i is the WCET of the single computation including all the overheads. When re-executions occur, the total time is $2C_i$, then $3C_i$, etc. In this scenario, χ_i represents the maximum number of restarts possible or, in other words, the number of faults to tolerate.

Because the MC-IMW model is a specialization of the MC model, EDF-VD can schedule MC-IMW task sets. However, its speedup bounds need to be revised. We will show that EDF-VD improves its speedup bounds and remains optimal for $L = 2$.

3.1 Analytical bound for EDF-VD(2)

In a traditional MC model, EDF-VD(2) has a speedup bound of $\sigma = \frac{4}{3}$ [2, Theorem 3.8]. By following the original proof of this theorem, we prove the following:

Theorem 3.1. *In a MC-IMW system with 2 criticality levels, any task set Γ satisfying Eq. (1) is schedulable by EDF-VD on a processor of speed $\sigma = \frac{1}{2}(1+\phi) \approx 1.309$, where ϕ is the golden ratio.*

Proof. The schedulability condition of EDF-VD is [2, Eq. (3)]:

$$\frac{U_2(1)}{1-U_1(1)} \leq \frac{1-U_2(2)}{U_1(1)} \tag{4}$$

According to Eq. (3), we can rewrite this equation by replacing $U_2(2)$ with $2U_2(1)$. In addition, let us assume we have a processor of speed $\sigma \geq 1$ and replace the corresponding $U_i(j)$ with $\frac{U_i^*(j)}{\sigma}$ to obtain:

$$\frac{U_2^*(1)}{\sigma - U_1^*(1)} \leq \frac{\sigma - 2U_2^*(1)}{U_1^*(1)} \tag{5}$$

To prove the claim, we find the largest $\frac{1}{\sigma}$ such that the inequality holds, and σ will be our speedup bound. EDF-VD must be able to schedule all the schedulable task sets, i.e., satisfying Eq. (1), by considering a processor of speed σ:

$$U_1(1) + U_2(1) \leq \frac{1}{\sigma} \iff U_1^*(1) + U_2^*(1) \leq 1 \tag{6}$$

$$2U_2(1) \leq \frac{1}{\sigma} \iff 2U_2^*(1) \leq 1 \tag{7}$$

The tightest situation condition occurs when equal signs in Eq. (6) and Eq. (7) hold, i.e., when $U_1^*(1) = U_2^*(1) = 1/2$: $U_2^*(1)$ appears in the numerator with the negative sign on the right-hand side, and with the positive sign on the left-hand side; vice versa, $U_1^*(1)$ appears in the denominator with the positive sign on the right-hand side, and with the negative sign on the left-hand side. By replacing these limits into Eq. (5), we obtain:

$$\sigma^2 - \frac{3}{2}\sigma + \frac{1}{4} \rightarrow \sigma = \frac{3 \pm \sqrt{5}}{4} \tag{8}$$

We exclude the solution with the negative sign because we defined $\sigma \geq 1$, therefore $\sigma = \frac{3+\sqrt{5}}{4} = \frac{1}{2}(1+\phi)$. □

This result shows that, with the introduced restriction on the WCET of the MC-IMW model, the speedup bound for 2 criticality levels slightly improves from $\frac{4}{3}$ down to $\frac{1}{2}(1+\phi)$. The new bound implicitly violates the speedup-optimality theorem [2, Theorem 3.12], which states that a non-clairvoyant scheduling algorithm with a speedup factor lower than $\frac{4}{3}$ cannot exist. We will later show in Section 3.4 that EDF-VD(2) is still optimal.

[2]This mechanism, to correctly work, requires several assumptions, for instance, that the fault must not affect the input data or persistent data that are not recomputed by the job itself. However, this discussion is outside the scope of this paper.

3.2 Analytical bound for EDF-VD(3)

In a traditional MC model, EDF-VD(3) has a speedup bound of $\sigma = 2$ [2, Theorem 3.9]. We prove the following for MC-IMW:

Theorem 3.2. *In a MC-IMW system with 3 criticality levels, any task set Γ satisfying Eq. (1) is schedulable by EDF-VD on a processor of speed $\sigma = \frac{11+\sqrt{61}}{12} \approx 1.567$.*

Proof. The proof follows the same approach of the proof of Theorem 3.1. In this case, two conditions are in logical disjunction [2, Eq. (3)] to guarantee schedulability:

$$\frac{U_2(1) + U_3(1)}{1 - U_1(1)} \leq \frac{1 - U_2(2) - U_3(3)}{U_1(1)} \tag{9}$$

$$\frac{U_3(2)}{1 - U_1(1) - U_2(2)} \leq \frac{1 - U_3(3)}{U_1(1) + U_2(2)} \tag{10}$$

To prove the claim, we find the largest $0 < \frac{1}{\sigma} \leq 1$ such that at least one of the two inequality holds, and σ will be our speedup bound. The value σ must satisfy Eq. (1):

$$U_1(1) + U_2(1) + U_3(1) \leq \frac{1}{\sigma} \tag{11}$$

$$2U_2(1) + 2U_3(1) \leq \frac{1}{\sigma} \tag{12}$$

$$3U_3(1) \leq \frac{1}{\sigma} \tag{13}$$

In both Eq. (9) and Eq. (10), increasing $U_3(1)$, $U_2(1)$, or $U_1(1)$ makes the bound tighter. Therefore, the worst-case bound occurs when Eq. (11), Eq. (12), and Eq. (13) have strict equal sign, i.e., when $U_3^*(1) = \frac{1}{3}$, $U_2^*(1) = \frac{1}{6}$, and $U_1^*(1) = \frac{1}{2}$. See proof of Theorem 3.1 for the definition of $U_i^*(j)$. By replacing these values, we obtain for Eq. (9):

$$\frac{1/2}{\sigma - 1/2} \leq \frac{\sigma - 4/3}{1/2} \Rightarrow \sigma^2 - \frac{11}{6}\sigma + \frac{5}{12} = 0 \tag{14}$$

which leads to $\sigma = \frac{11}{12} + \frac{\sqrt{61}}{12} \approx 1.56$ (the other solution is not feasible because $\sigma < 1$). Instead, for the second condition of Eq. (10):

$$\frac{2/3}{\sigma - 5/6} \leq \frac{\sigma - 1}{5/6} \Rightarrow \sigma^2 - \frac{11}{6}\sigma + \frac{5}{18} = 0 \tag{15}$$

which leads to $\sigma = \frac{5}{3} \approx 1.67$ (the other solution is not feasible because $\sigma < 1$). The best speedup is the lowest of the two, thus the best speedup bound is $\sigma = \frac{11+\sqrt{61}}{12}$. □

3.3 Numerical bounds for EDF-VD(L)

The speedup bound for EDF-VD(L) – i.e., for $L > 2$ criticality levels – is difficult to be analytically computed. The original paper [2] runs a non-linear programming solver to obtain the result. We also followed a numerical approach. Luckily, MC-IMW makes the numerical algorithm simpler because it does not need to use a non-linear programming solver. Instead, we run a numerical simulation as follows:

(1) Start with $L = 2$ and $\sigma_{\text{prev}} = 1$.
(2) The worst-case conditions in term of tightness can be easily obtained simplifying Eq. (1) with the result of Eq. (3) and observing that the EDF-VD(L) schedulability test formula [2, Theorem 3.4, Eq. (3)] is tight when the equality of Eq. (1) holds.
(3) Compute the worst-case $U_i^*(1)$ values, and explore all the possible σ from σ_{prev} upwards[3], until the schedulability test formula [2, Theorem 3.4, Eq. (3)] is satisfied with $U_i(1) = U_i^*(1)/\sigma$.
(4) The obtained $\bar{\sigma}$ is the best speedup bound for EDF-VD with L criticality levels in MC-IMW systems.
(5) Set $\sigma_{\text{prev}} = \bar{\sigma}$, increment L and go to step (2) until the desired criticality level is reached.

n	MC	MC-IWF
2	1.3333	1.309017
3	2.0000	1.567521
4	2.6180	1.778826
5	3.0811	1.948280
6	3.7321	2.066997
7	4.2361	2.173933
8	4.7913	2.270963
9	5.3723	2.359626
10	5.8551	2.441166
11	6.4641	2.507181
12	6.9487	2.567371
13	7.5311	2.624127

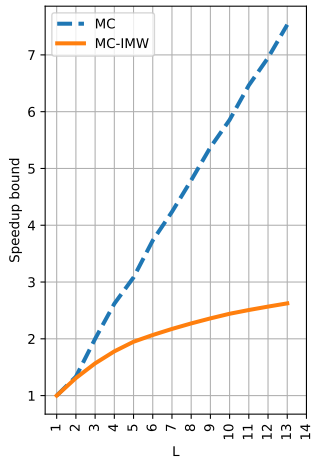


Figure 1: EDF-VD(L) schedulability: comparison between MC and MC-IMW model.

The open source code is publicly available[4] and the exploration of the first 13 levels required a few minutes on a standard PC.

Figure 1 shows the results and the comparison of the MC and the MC-IMW models. On the right plot, we see that the curve of MC-IMW has a logarithmic trend, while the traditional MC is almost linear. This means that increasing the number of levels requires a linear increase of the processor speed to allow EDF-VD to schedule the task sets, while MC-IMW requires only a logarithmic increase. We can conclude that while the speedup bound improvement of MC-IMW models is minimal for low values of L, the use of MC-IMW makes the scheduling problem easier for large L values.

3.4 EDF-VD(2) remains speedup optimal

After the proof that the MC lower-bound condition on speedup is no more valid for MC-IMW, the question is whether EVD-VD(2) is still speedup optimal or not. The answer is affirmative:

Theorem 3.3 (EDF-VD(2) is speedup optimal in MC-IMW). *In a MC-IMW system, the speedup bound of any non-clairvoyant scheduling algorithm is at least $\sigma = \frac{1}{2}(1 + \phi)$.*

Proof. By Theorem 3.1, any MC-IMW schedulable task set is schedulable by EDF-VD on a processor with speed $\sigma = \frac{1}{2}(1 + \phi) = \frac{3+\sqrt{5}}{4}$. We will prove that no scheduling algorithm can do better, i.e., that no scheduling algorithm has a lower speedup bound σ. To prove this we proceed by contradiction. Let us assume that a

[3] We have chosen steps of 10^{-7} and a 128-bit floating point numbers for all the calculus.
[4] https://doi.org/10.5281/zenodo.7253716

scheduling algorithm having speedup $\sigma = \frac{3+\sqrt{5}}{4+\varepsilon}$ exists, where $\varepsilon > 0$ is an arbitrarily-small constant. The following task set must be then schedulable because satisfying Eq. (6) and Eq. (7):

Task	χ_i	C_i	T_i
τ_1	1	$\sqrt{5}-1$	$\sqrt{5}+1$
τ_2	2	$2+\varepsilon/2$	$\sqrt{5}+3$

Indeed, $U_1(1) = \frac{\sqrt{5}-1}{\sqrt{5}+1} = \frac{2}{3+\sqrt{5}}$ and $U_2(1) = \frac{2+\varepsilon/2}{3+\sqrt{5}}$. For the first jobs of τ_1 and τ_2, any scheduler can proceed in only two ways:

- Schedule first τ_1 and then τ_2. In this case, because the scheduler is not clairvoyant, we cannot guarantee the HI-mode deadline: indeed if τ_1 terminates at $t = \sqrt{5}-1$ unit of times, then there are only 4 time units left to run the τ_2 which may potentially require $4+\varepsilon$ unit of times to meet the deadline if the mode switch occurs. Thus scheduling first τ_1 and then τ_2 is not a viable option.
- Schedule first τ_2 and then τ_1. In this case, the system violates the LO-mode assumption: indeed if τ_2 terminates at $t = 2+\varepsilon/2$ unit of times, then there is no time for τ_1 to meet the deadline, because $2+\varepsilon/2+\sqrt{5}-1 > \sqrt{5}+1$. Thus scheduling first τ_2 and then τ_1 is not a viable option.

In conclusion, we proved that this task set is not schedulable under any non-clairvoyant algorithm on a processor with $\sigma < \frac{1}{2}(1+\phi)$. This contradicts the hypothesis that a non-clairvoyant algorithm exists with a better bound than EDF-VD(2). □

It is not known whether EDF-VD(L), with $L > 2$, is speedup-bound optimal or not. The large values of the speedup bounds for large L suggest it is not. Interestingly, in MC-IMW, the logarithmic speedup bound trend of Figure 1 suggests that it may be speedup bound optimal. However, no formal results are currently available.

4 A MORE GENERAL PARADIGM THAN MC

A recent article [9] proposed a generalization of the MC model that introduces the concept of *Dropping Relations* (DRs). In the traditional Vestal's model, tasks are grouped according to their criticality level. Instead, in the DR model, when a job overruns one of its WCETs, a set of specific tasks linked to this event is dropped. This model was designed for fault tolerance with re-execution as the fault recovery mechanism (see Section 2.1). Consequently, the DR model, as proposed in [9], is very similar to the MC-IMW model of Section 3, with the exception that tasks are duplicated instead of considering different WCETs. In the next section, we extend the concept of the DRs, by removing the integer restriction of MC-IMW, and making the DR model a generalization of the MC model.

4.1 DR model description

We reshape the original DR definition to make it more general. Subsequently, we make it specific for the MC and MC-IMW cases. We write $\tau_i^{(j)}$ to identify the i-th task running in the j-th mode, i.e., when a job of τ_i overran the $(j-1)$-th WCET.

Definition 4.1 (Dropping Relation). A dropping relation

$$d(\tau_i^{(j)}) = \{\tau_a^{(b)}, \tau_c^{(d)}, \ldots\}$$

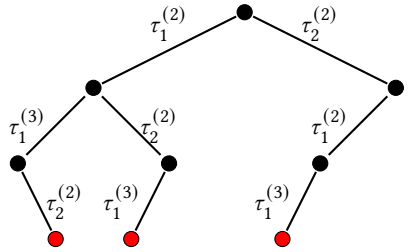


Figure 2: The event tree of Example 4.2. Red nodes identify conditions that could not occur due to the presence of DRs.

with $1 < j < \chi_i, 0 < b < \chi_a, 0 < d < \chi_c, \ldots$ specifies the following behavior: When task τ_i executes in j-th mode – i.e., it overruns its WCET $C_i(j-1)$ – then the task τ_a is immediately dropped as soon as it enters in b-th mode, the task τ_c is immediately dropped as soon as it enters in d-th mode, etc.

Example 4.2. A task set with re-execution as fault recovery strategy is composed of τ_1, that potentially needs to re-execute 2 times, and τ_2, that potentially needs to re-execute 1 extra time. The criticality level of τ_1 is then $\chi_1 = 3$ with $C_1(3) = 3C_1(1)$ and $C_1(2) = 2C_1(1)$, while the criticality level of τ_2 is then $\chi_2 = 2$ with $C_2(2) = 2C_2(2)$. Let us assume that, according to a failure analysis, we add the following DRs: $d(\tau_1^{(3)}) = \{\tau_2^{(1)}, \tau_2^{(2)}\}$, $d(\tau_2^{(2)}) = \{\tau_1^{(3)}\}$. These DRs mean: if τ_1 overruns $C_1(2)$, τ_2 is dropped; if τ_2 overruns $C_2(1)$, τ_1 cannot execute for more than $C_1(2)$ or it will be dropped. These constraints cannot be expressed with the traditional MC model: τ_1 overrunning $C_2(1)$ drops τ_2, so χ_1 should be larger than χ_2, however, when τ_2 overruns $C_2(1)$, it prevents τ_1 entering in the 3-rd mode, and therefore χ_2 should be larger than χ_1 which contradicts the previous result.

The presence of DRs replaces the system mode concept because the dropping is performed more fine-grained than in a traditional MC system. The DR model is a generalization of the MC model because it is possible to map any MC model to the DR model:

Lemma 4.3 (DR is a generalization of MC). *An MC task set with maximum criticality level L, defined according to the notation of Section 2, is equivalent to a DR model with dropping relations satisfying the following relation:*

$$\forall \ell \forall \tau_i \forall \tau_j : 1 < \ell \leq L \wedge \chi_i = \ell \wedge \chi_j < \ell \Rightarrow \tau_j \in d(\tau_i)$$

Proof. The proof follows directly from the two definitions. □

From this Lemma, we can state that DR is also a generalization of MC-IMW, being the latter a specialization of the MC model.

The event tree. The original paper proposed a tree as the online representation of the system status according to the DRs. Let us call this tree the *event tree*, because it represents the possible chain of events potentially occurring at run-time. The event tree of Example 4.2 is depicted in Figure 2. The system is initially in the root node. If τ_1 overruns $C_1(1)$, i.e., enters in the 2-nd criticality mode, then the system moves to the left node on the second level of the tree, following the edge $\tau_1^{(2)}$, which represents the overrun event. If then τ_1 overruns $C_1(2)$ or τ_2 overruns $C_2(1)$, the system moves

to another node on level 3, and the same applies to the other nodes and events. The red nodes are invalid nodes, because, for instance, τ_2 is dropped if it tries to overrun $C_2(2)$ after τ_2 had overran $C_1(2)$. We can, in this way, redefine the concept of *system mode* which is no more a single criticality level value, but a node in the event tree, with a set of active tasks and a set of dropped tasks.

4.2 The need for new scheduling algorithms

Reghenzani et al. [9] proposed a scheduling strategy based on the event tree and the EDF-VD scheduling test. The authors performed a simulation and showed how the schedulability ratio improved compared to normal EDF or EDF-VD. This scheduling strategy consists in extracting all the root-to-leaf paths (excluding red nodes) from the tree; each of the paths represents a single scheduling problem coherent with the MC model. Therefore, if all the paths are schedulable with EDF-VD and all the paths have one single common scaling parameter λ, then the whole tree is schedulable according to that scaling parameter. Online, depending on the occurring events, a specific instance of MC problem is "selected". This strategy is, however, sub-optimal. Indeed, we will prove in the next theorem that the scheduling strategy is not only sub-optimal due to the use of the EDF-VD test, but it would be sub-optimal even using a clairvoyant scheduling test for the single path.

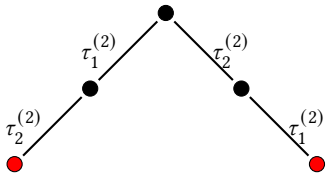
Theorem 4.4. *The scheduling strategy is based on the event tree and schedulability of all the single paths, described in [9, Section V.B], is sub-optimal regardless of the considered scheduling test.*

Proof. Let us consider the following MC-IMW task set and DRs:

Task	χ_i	C_i	T_i
τ_1	2	2	4
τ_2	2	1	2

$$d(\tau_1^{(2)}) = \{\tau_2^{(1)}, \tau_2^{(2)}\} \qquad d(\tau_2^{(2)}) = \{\tau_1^{(1)}, \tau_1^{(2)}\}$$

The event tree is then composed of only two paths: the overrun of the first WCET of τ_1 which prevents τ_2 to execute, and the overrun of the first WCET of τ_2 which prevents τ_1 to execute:



In order to verify the schedulability of the event tree, we have to check the schedulability of each path and then merge the scheduling decisions into a single one[5]. Let us assume that we use, for each path, an optimal MC clairvoyant scheduling test, i.e., having speedup $\sigma = 1$, which is the best theoretical scheduling algorithm. The test, when applied to each path, outputs *schedulable* for all paths:

- Path 1: if τ_1 will overrun 2, then schedule τ_1 first, otherwise schedule τ_2 so that it does not miss the deadline.
- Path 2: Always schedule τ_2 first.

[5]For instance, in the case of EDF-VD, we need to find a common value for λ.

However, we cannot merge these scheduling decisions for the two paths into a single decision. Therefore, the task set is not schedulable but is schedulable by a global clairvoyant scheduling algorithm:

- Always schedule τ_2 first, unless τ_1 will overrun 2.

We proved that a scheduling algorithm correctly scheduling more task sets than the single-path approach [9, Section V.B] exists. □

Future scheduling algorithms, reasoning in a global perspective on the tree, can potentially improve these results by exploiting the flexibility added by the DR model. How to create the DRs is also an open problem, because it involves both failure and scheduling requirements, thus it is integrated in the scheduling analysis.

5 CONCLUSION

Inspired by the fault tolerance re-execution strategy, we presented an approach to determine the WCET values, namely the MC-IMW model. This model simplifies the scheduling problem and improves the speedup bound of EDF-VD. We also proved that EDF-VD(2) remains speedup optimal. Then, we generalized the MC-IMW and MC models into the DR model and highlighted the open problems. Future works can follow different paths. Regarding MC-IMW, we need to prove if EDF-VD(L) with $L > 2$ is speedup optimal or not. If not, can the restrictions of MC-IMW help in developing new scheduling algorithms? The same research question applies to the DR model: can future scheduling algorithms, specifically developed for the DR model, have better properties, e.g., schedulability?

ACKNOWLEDGMENTS

This article has received funding from the EU H2020 Research and Innovation Programme, project TEXTAROSSA (grant no. 956831).

REFERENCES

[1] S. Baruah, V. Bonifaci, G. DAngelo, H. Li, A. Marchetti-Spaccamela, S. van der Ster, and L. Stougie. 2012. The Preemptive Uniprocessor Scheduling of Mixed-Criticality Implicit-Deadline Sporadic Task Systems. In *2012 24th Euromicro Conference on Real-Time Systems*. IEEE, Pisa, Italy, 145–154.

[2] S. Baruah, V. Bonifaci, G. D'angelo, H. Li, A. Marchetti-Spaccamela, S. Van Der Ster, and L. Stougie. 2015. Preemptive Uniprocessor Scheduling of Mixed-Criticality Sporadic Task Systems. *J. ACM* 62, 2, Article 14 (may 2015), 33 pages.

[3] S. Baruah, H. Li, and L. Stougie. 2010. Towards the Design of Certifiable Mixed-criticality Systems. In *2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE, Stockholm, Sweden, 13–22.

[4] A. Bhuiyan, F. Reghenzani, W. Fornaciari, and Z. Guo. 2020. Optimizing Energy in Non-Preemptive Mixed-Criticality Scheduling by Exploiting Probabilistic Information. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 11 (2020), 3906–3917.

[5] A. Burns and R.I. Davis. 2017. A Survey of Research into Mixed Criticality Systems. *ACM Comput. Surv.* 50, 6, Article 82 (nov 2017), 37 pages.

[6] R. Ernst and M. Di Natale. 2016. Mixed Criticality Systems—A History of Misconceptions? *IEEE Design & Test* 33, 5 (2016), 65–74.

[7] A. Esper, G. Nelissen, V. Nélis, and E. Tovar. 2015. How Realistic is the Mixed-Criticality Real-Time System Model?. In *Proceedings of the 23rd International Conference on Real Time and Networks Systems*. ACM, Lille, France, 139–148.

[8] Z. Guo, K. Yang, S. Vaidhun, S. Arefin, S.K. Das, and H. Xiong. 2018. Uniprocessor Mixed-Criticality Scheduling with Graceful Degradation by Completion Rate. In *IEEE Real-Time Systems Symposium (RTSS)*. IEEE, Nashville, TN, USA, 373–383.

[9] F. Reghenzani, Z. Guo, L. Santinelli, and W. Fornaciari. 2022. A Mixed-Criticality Approach to Fault Tolerance: Integrating Schedulability and Failure Requirements. In *2022 IEEE 28th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, Milan, Italy, 27–39.

[10] S. Vestal. 2007. Preemptive Scheduling of Multi-criticality Systems with Varying Degrees of Execution Time Assurance. In *28th IEEE International Real-Time Systems Symposium (RTSS)*. IEEE, Tucson, AZ, USA, 239–243.

[11] M. Völp. 2014. What if we would degrade LO tasks in mixed-criticality systems?. In *Work-in-Progress Session of the 20th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, Berlin, Germany, 2 pages.