Scheduling Analysis of Imprecise Mixed-Criticality Real-Time Tasks

Liu, D.; Guan, N.; Spasic, J.; Chen, G.; Liu, S.; Stefanov, T.P.; Yi, W.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2018.2789879, IEEE Transactions on Computers

Scheduling Analysis of Imprecise Mixed-Criticality Real-Time Tasks

Di Liu[1, 2], Nan Guan[1], Jelena Spasic[3], Gang Chen[4], Songran Liu[4], Todor Stefanov[3], Wang Yi[4, 5]
[1] Hong Kong Polytechnic University, Hong Kong
[2] Yunnan University, China
[3] Leiden University, The Netherlands
[4] Northeastern University, China
[5] Uppsala University, Sweden

Abstract—In this paper, we study the scheduling problem of the *imprecise mixed-criticality model* (IMC) under *earliest deadline first with virtual deadline* (EDF-VD) scheduling upon uniprocessor systems. Two schedulability tests are presented. The first test is a concise utilization-based test which can be applied to the implicit deadline IMC task set. The suboptimality of the proposed utilization-based test is evaluated via a widely-used scheduling metric, *speedup factors*. The second test is a more effective test but with higher complexity which is based on the concept of demand bound function (DBF). The proposed DBF-based test is more generic and can apply to constrained deadline IMC task set. Moreover, in order to address the high time cost of the existing deadline tuning algorithm, we propose a novel algorithm which significantly improve the efficiency of the deadline tuning procedure. Experimental results show the effectiveness of our proposed schedulability tests, confirm the theoretical suboptimality results with respect to speedup factor, and demonstrate the efficiency of our proposed algorithm over the existing deadline tunning algorithm. In addition, issues related to the implementation of the IMC model under EDF-VD are discussed.

1 Introduction

As safety-critical systems with diverse functionalities have been emerging, besides real-time constraints, many real-time applications in safety-critical systems also feature another important property, called *criticality levels*. For example, unmanned aerial vehicles (UAVs) have two types of applications, safety-critical applications, such as flight control, and mission-critical applications, such as surveillance and video streaming. The safety-critical applications (e.g., the flight control) have higher criticality level because they are essentially crucial to the operational safety of the whole system and failure (i.e., violating timing properties) of the safety-critical applications will lead to a catastrophic consequence, such as loss of UAV which may injure a human-being. On the other hand, the mission-critical applications have lower criticality level because they are not coupled to the operational safety of the whole system, so failure of mission-critical applications will not threaten the operational safety of the system but will only affect the system service quality. In different industrial contexts, different standards are deployed to guide the design of systems with different criticality-level applications, such as IEC61508 for electrical/electronic/programmable electronic safety-related systems, ISO26262 for automotive systems, and DO-178B/C for avionic systems.

With the rapid development of complex and sophisticated safety-critical systems, increasing number of applications with different criticality and complex functionality are incorporated into a system, thus requiring a plentiful of processing units. For instance, modern premium cars typical contain around 70-100 computers, around 100 electronic motors and 2 km of wire [1]. This complicated and sometimes redundant hardware leads to a system with large system size and very high power consumption. Therefore, to reduce Size, Weight, and Power (SWaP), the emerging trend in the development of safety-critical systems is to integrate applications with different criticality into a shared computing platform. We call such systems *mixed-criticality systems*. A formal definition of a *mixed-criticality system* is given as follows:

Definition 1 ([2]). *A mixed-criticality system is an integrated suite of hardware, operating system and middleware services, and application software that supports the execution of safety-critical, mission-critical, and non-critical software within a single, secure compute platform.*

To ensure the timing correctness of a *mixed-criticality* (MC) system, highly critical tasks are subject to certification by Certification Authorities (CAs). In order to guarantee the safety and correctness of highly critical tasks in all cases, CAs consider very pessimistic situations which even rarely occur in practice. As a consequence, this conservativeness leads to a large overestimation of worst-case execution time (WCET) for these highly critical tasks and in turn to resource wastage. To deal with this overestimation, Vestal proposed in [3] to characterize a highly critical task with different WCETs corresponding to different criticality levels. Besides the WCET determined by the CAs, each highly critical task is specified with several smaller WCETs which are determined by system designers at lower assurance levels, i.e., considering less pessimistic situations. Scheduling highly critical tasks using their low assurance WCETs can better utilize hardware resource, and in most cases all tasks can be safely and successfully scheduled with their low assurance WCETs, and then the system is deemed to operate in *low*-criticality mode.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2018.2789879, IEEE Transactions on Computers

Then, if a rare case occurs, i.e., any highly critical task cannot complete its execution within its low assurance WCET, the system discards all less critical tasks and schedules only highly critical tasks with their certified (very pessimistic) WCETs. When any highly critical tasks overrun, the system is deem to transit to *high*-criticality mode and operate in this mode. The challenge in scheduling MC systems is to simultaneously guarantee the timing correctness of (1) only high-criticality tasks under very pessimistic assumptions, and (2) all tasks, including low-critical ones, under less pessimistic assumptions such that resource efficiency is achieved.

The scheduling problem of MC systems has been intensively studied in recent years (see Section 2 for a brief review and [4] for a comprehensive review). The MC model proposed by Vestal in [3] receives the most attention from the real-time scheduling community such as [5]–[8]. In the reminder of this article, we refer to the MC model proposed by Vestal in [3] as the *classical* MC model. However, the classical MC model seriously disturbs the service of low-criticality tasks as it discards low criticality tasks completely when the system switches to *high*-criticality mode. This is actually not acceptable in many practical systems, so the Vestal MC model receives some criticisms from system designers [9] [10].

Several new MC models have been proposed to improve execution of low criticality tasks in *high*-criticality mode, e.g., [9], [10], etc. Burns and Baruah in [9] introduced an imprecise mixed-criticality (IMC) task model [4] [11] where low-criticality tasks reduce their execution budgets (i.e., short execution time) to guarantee their execution with regular execution frequency (i.e., the same period) in *high*-criticality mode. This IMC model is highly beneficial to those low criticality tasks which feature the *imprecise* property defined in the widely known and studied *imprecise computation model* [12] [13]. In the *imprecise computation model*, the output quality of a task is related to its execution time. The longer a task executes, the better quality results it produces. Then, if there is an overload in the system, tasks can trade off the quality of the produced results (i.e., reduce the execution time) to ensure their timing correctness. In [14], Ravindran *et al.* gave several real-life applications with this imprecise feature in different domains, e.g., video encoding, robotic control, cyber-physical systems, and planetary rover.

However, the IMC model does not receive sufficient attention, only few works studying the scheduling problem of the IMC model [9] [15]. Earliest-deadline-first with virtual deadlines (EDF-VD) scheduling algorithm [5] has shown strong competence for the classical MC model by both theoretical and empirical evaluations [5], [7], [8], where the classical EDF scheduling algorithm is enhanced by a deadline adjustment mechanism to compromise the resource requirement on different criticality levels. Although EDF-VD is an effective MC scheduling algorithm, the scheduling analysis and performance of the IMC model under EDF-VD scheduling has not been addressed and known yet. Therefore, in this paper, we study EDF-VD scheduling of the IMC model and demonstrate the scheduling performance through comprehensive comparison with other state of the art scheduling algorithms for the IMC model. The main technical contributions of this paper include:

- We propose a utilization-based sufficient test for the IMC model under EDF-VD, - see Theorem 3 in Section 4. This concise utilization-based test is applicable to the case where the IMC tasks with *implicit deadlines* are considered and virtual deadlines of all *high*-criticality tasks are tuned uniformly;
- With our proposed utilization-based test, we quantify the EDF-VD scheduling for the IMC model via a scheduling metric, namely speedup factor. We derive a speedup factor function with respect to the utilization ratios of high criticality tasks and low criticality tasks - see Theorem 4 in Section 5. The derived speedup factor function enables us to quantify the suboptimality of EDF-VD and evaluate the impact of the utilization ratios on the speedup factor. We also compute the maximum value $4/3$ of the speedup factor function, which is equal to the speedup factor bound for the classical MC model [5].
- We propose a demand bound function (DBF) based test for the IMC model. The DBF-based test is a good complement to the utilization-based test and can be used for the more generic case where constrained deadline IMC tasks can be considered and virtual deadlines of high-criticality tasks can be tuned individually.
- Along with the DBF-based test, we propose a novel deadline tune algorithm which significantly improves the efficiency of the deadline tuning procedure in comparison with the existing algorithm [7] [8].
- We carry out extensive experiments on synthetic IMC task sets. The experimental results show the effectiveness of the proposed schedulability tests over the existing approaches. Moreover, the experimental results validate the observations we obtained for speedup factor and demonstrates the efficiency of our proposed deadline tunning algorithm.
- We present a possible implementation of IMC model under EDF-VD based on Linux OS with LITMUS-rt extension [16] and discuss the run-time overhead.

The remainder of this paper is organized as follows: Section 2 discusses the related work. Section 3 gives the preliminaries and describes the IMC task model and its execution semantics. Section 4 presents our sufficient test for the IMC model and Section 5 derives the speedup factor function for the IMC under EDF-VD. Section 6 presents our DBF based test and gives the new deadline tunning algorithm. Section 7 shows our experimental results and Section 8 discusses the implementation and overhead of the IMC model. Finally, Section 9 concludes this paper.

2 Related Work

Burns and Davis in [4] gave a comprehensive review of work on real-time scheduling for MC systems. Many of these literatures, e.g., [5] [7] [8], considered the classical MC model in which all low criticality tasks are discarded if the system switches to the high criticality mode. Several models or approaches are proposed to improve the execution low criticality tasks when there is an overrun occurred to any high criticality tasks. In [9], Burns and Baruah discussed three

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2018.2789879, IEEE Transactions on Computers

approaches to keep some low criticality tasks running in *high*-criticality mode. The first approach is to change the priority of low criticality tasks. However, for fixed-priority scheduling, deprioritizing low criticality tasks cannot guarantee the execution of the low criticality tasks with a short deadline after the mode switches. [9]. Similarly, for EDF, lowering priority of low criticality tasks leads to a degraded service [10]. In this paper, we consider the IMC model which improves the schedulability of low criticality tasks in *high*-criticality mode by reducing their execution time. The IMC model can guarantee the regular service of a system by trading off the quality of the produced results. For some applications given in [12] [13] [14], such trade-off is preferred.

The second approach in [9] is to extend the periods of low criticality tasks when the system mode changes to *high*-criticality mode such that the low criticality tasks execute less frequently to ensure their schedulability. Su *et al.* [17] [18] and Jan *et al.* [19] both consider this model. However, some applications might prefer an on-time result with a degraded quality rather than a delayed result with a perfect quality. Some example applications can be seen in [20] [12] [13]. Then, the approach of extending periods is less useful for this kind of applications. The last approach proposed in [9] is to reduce the execution budget of low criticality tasks when the system mode switches, i.e., the use of the IMC model studied in this paper. In [9], the authors extend the AMC [6] approach to test the schedulability of an IMC task set under fixed-priority scheduling. Comparing to the AMC, EDV-VD scheduling provides better schedulability and to our best knowledge this is the first work addressing the schedulability analysis of the IMC model under EDF-VD scheduling. In [15], Baruah *et al.* analyzed the schedulability of the IMC model under MC-fluid scheduling [21]. However, in practice, MC-fluid scheduling algorithm suffers from extremely high scheduling overhead due to the frequent context switching, so the scheduling performance is affected seriously when the scheduling overhead is taken into account. Moreover, in Section 7, the experimental results show on uniprocessor systems the EDF-VD scheduling is even slightly better than the MC-fluid scheduling.

Some works tried to drop a subset of *low*-criticality tasks instead of all *low*-criticality tasks [22], [23] in *high*-criticality mode. Comparing to the IMC model, these studies have two shortcomings: 1) both works consider a hierarchy scheduling which may suffer from much higher scheduling overhead, e.g., context-switch; 2) there is no service guarantee for *low*-criticality tasks in *high*-criticality mode. In this paper, EDF-VD scheduling algorithm is considered, where EDF-VD only causes negligibly additional overhead than the original EDF scheduling. In addition, as long as the system are schedulable with the specified parameters, a minimum quality of service is guaranteed to each *low*-criticality tasks. [24] is similar to our work in providing a guaranteed service to *low*-criticality tasks in *high*-criticality mode, but their approach relies on a run-time budget allocator. Therefore, it is difficult for their approach to provide any theoretical bound on the scheduling performance such as the speedup factor we obtain in this paper.

The execution semantics of the classical MC model and IMC model are very similar to the systems operating with several modes [25]. The multi-mode system usually executes in one mode and may change the mode during the runtime. The crucial and main difference of the existing multi-mode protocols and the MC models is that the multi-mode protocols only guarantee the schedulability in each mode, however the schedulability of the mode transition/switch is not considered [26]. For the classical MC model and IMC model, even the schedulability of the mode transition is required to be ensured. Therefore, the existing multi-mode protocols or analysis cannot be applied to the MC model.

3 PRELIMINARIES

This section first introduces the IMC task model and its execution semantics. Then, we give a brief explanation for EDF-VD scheduling [5] and an example to illustrate the execution semantics of the IMC model under EDF-VD scheduling.

3.1 Imprecise Mixed-Criticality Task Model

We consider the *sporadic* task model given in [9] where a task set γ includes n tasks which are scheduled on a uniprocessor system. Without loss of generality, all tasks in γ are assumed to start at time 0. Each task τ_i in γ generates an infinite sequence of jobs $\{J_i^1, J_i^2 ...\}$ and is characterized by $\tau_i = \{T_i, D_i, L_i, \mathcal{C}_i\}$:

- T_i is the period or the minimal separation interval between two consecutive jobs;
- D_i denotes the relative task deadline;
- $L_i \in \{LO, HI\}$ denotes the criticality (*low or high*) of a task. In this paper, like in many previous research works [17] [10] [5] [7] [8], we consider a duel-criticality MC model. Then, we split tasks into two task sets, $\gamma_{LO} = \{\tau_i | L_i = LO\}$ and $\gamma_{HI} = \{\tau_i | L_i = HI\}$;
- $\mathcal{C}_i = \{C_i^{LO}, C_i^{HI}\}$ is a list of WCETs, where C_i^{LO} and C_i^{HI} represent the WCET in *low*-criticality mode and the WCET in *high*-criticality mode, respectively. For a *high*-criticality task, it has $C_i^{LO} \leq C_i^{HI}$, whereas $C_i^{LO} \geq C_i^{HI}$ *for a low-criticality task, i.e., low-criticality task* τ_i *has a reduced WCET in high-criticality mode*.

WCET Estimation: The *high*-criticality tasks are subject to certification by Certification Authorities (CAs), so CAs usually provide *high*-criticality WCET estimation (C_i^{HI}) for high criticality tasks. On the other hand, the system designers estimate WCETs, C_i^{LO} for both *low*-criticality and *high*-criticality tasks, and C_i^{HI} for *low*-criticality tasks. For *low*-criticality tasks, C_i^{HI} is estimated by system designers according to their expected Quality of Service (QoS) requirement (i.e., degraded) when the system executes in *high*-criticality mode. This is analogous to the *imprecise computation model*, where tasks have a mandatory part which could guarantee an acceptable output when the system overloads. *Therefore, as long as a given IMC task set is schedulable, we consider that the IMC task set could guarantee a normal QoS in low-criticality mode and a degraded QoS in high-criticality mode.* It is worth noting that *low*-criticality criticality tasks could have several WCETs in *high*-criticality mode such that the system could select

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2018.2789879, IEEE Transactions on Computers

appropriate WCETs according to the system total workload of *high*-criticality mode. But in this paper, we mainly consider the schedulability test of the IMC model under EDF-VD scheduling and our proposed schedulability tests will serve as the critical foundation for this optimization problem. We leave this problem for our future work.

Every task $\tau_i \in \gamma$ could generate infinite jobs during system operation. Then each job J_i is characterized by $J_i = \{a_i, d_i, L_i, \mathcal{C}_i\}$, where a_i is the absolute release time and d_i is the absolute deadline. Note that if *low*-criticality task τ_i has $C_i^{HI} = 0$, it will be immediately discarded at the time of the switch to *high*-criticality mode. In this case, the IMC model behaves like the classical MC model. Notice that in Section 4, we consider the *implicit deadline sporadic* IMC model, i.e., $\forall \tau_i \in \gamma, D_i = T_i$. In Section 6, we consider a more general task model in which task's deadline is smaller than or equal to its period, i.e., $\forall \tau_i \in \gamma, D_i \leq T_i$, widely known as the *constrained deadline* task model.

In real-time theories, the utilization of a task is used to denote the ratio between its WCET and its period. We define the following utilizations for an IMC task set γ:

- For every task τ_i, it has $u_i^{LO} = \frac{C_i^{LO}}{T_i}$, $u_i^{HI} = \frac{C_i^{HI}}{T_i}$;
- For all *low*-criticality tasks, we have total utilizations
$$U_{LO}^{LO} = \sum_{\forall \tau_i \in \gamma_{LO}} u_i^{LO}, \ U_{LO}^{HI} = \sum_{\forall \tau_i \in \gamma_{LO}} u_i^{HI}$$
- For all *high*-criticality tasks, we have total utilizations
$$U_{HI}^{LO} = \sum_{\forall \tau_i \in \gamma_{HI}} u_i^{LO}, \ U_{HI}^{HI} = \sum_{\forall \tau_i \in \gamma_{HI}} u_i^{HI}$$
- For an IMC task set, we have
$$U^{LO} = U_{LO}^{LO} + U_{HI}^{LO}, \quad U^{HI} = U_{LO}^{HI} + U_{HI}^{HI}$$

3.2 Execution Semantics of the IMC Model

The execution semantics of the IMC model are similar to those of the classical MC model. The **major difference** occurs after a system switches to *high*-criticality mode. *Instead of discarding all low-criticality tasks, as it is done in the classical MC model, the IMC model tries to schedule low-criticality tasks with their reduced execution times* C_i^{HI}. The execution semantics of the IMC model are summarized as follows:

- The system starts in *low*-criticality mode, and remains in this mode as long as no *high*-criticality job overruns its *low*-criticality WCET C_i^{LO}. If any job of a *low*-criticality task tries to execute beyond its C_i^{LO}, the system will suspend it and launch a new job at the next period;
- If any job of *high*-criticality task executes for its C_i^{LO} time units without signaling completion, the system immediately switches to *high*-criticality mode;
- As the system switches to *high*-criticality mode, *if jobs of low-criticality tasks have completed execution for more than their* C_i^{HI} *but less than their* C_i^{LO}*, the jobs will be suspended till the tasks release new jobs for the next period. However, if jobs of low-criticality tasks have not completed their* C_i^{HI} $(\leq C_i^{LO})$ *by the switch time instant, the jobs will complete the left execution to* C_i^{HI} *after the switch time instant and before their deadlines. Hereafter, all jobs are scheduled using* C_i^{HI}. For *high*-criticality tasks, if their jobs have not completed their C_i^{LO} $(\leq C_i^{HI})$ by the switch time instant, all jobs will continue to be scheduled to complete C_i^{HI}. After that, all jobs are scheduled using C_i^{HI}.

Task	L	C_i^{LO}	C_i^{HI}	T_i	$\hat{D}_i$
τ_1	LO	4	2	9	
τ_2	HI	4	7	10	7

Table 1: Illustrative example

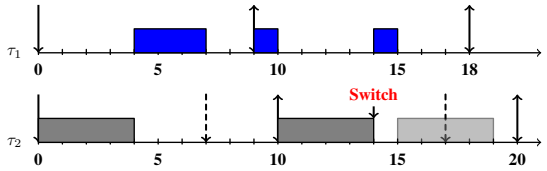


Figure 1: Scheduling of Example 1

Santy *et al.* [27] have shown that the system can switch back from *high*-criticality mode to *low*-criticality mode when there is an idle period and no *high*-criticality job awaits for execution. For the IMC model, we can use the same scenario to trigger the switch-back. In this paper, we focus on the switch from *low*-criticality mode to *high*-criticality mode.

3.3 EDF-VD Scheduling

The challenge to schedule MC tasks with EDF scheduling algorithm [28] is to deal with the overrun of *high*-criticality tasks when the system switches from *low*-criticality mode to *high*-criticality mode. Baruah *et al.* in [5] proposed to artificially tighten (i.e., tune down) deadlines of jobs of *high*-criticality tasks in *low*-criticality mode such that the system can preserve execution budgets for the *high*-criticality tasks across mode switches. This approach is called *EDF with virtual deadlines* (EDF-VD).

3.4 An Illustrative Example

Here, we give a simple example to illustrate the execution semantics of the IMC model under EDF-VD. Table 1 gives two tasks, one *low*-criticality task τ_1 and one *high*-criticality task τ_2, where $\hat{D}_i$ is the virtual deadline. Figure 1 depicts the scheduling of the given IMC task set, where we assume that the mode switch occurs in the second period of τ_2. When the system switches to *high*-criticality mode, τ_2 will be scheduled by its original deadline 10 instead of its virtual deadline 7. Hence, τ_1 preempts τ_2 at the switch time instant. Since in *high*-criticality mode τ_1 only has execution budget of 2 , i.e., C_1^{HI}, τ_1 executes one unit and suspends. Then, τ_2 completes its left execution 4 $(C_2^{HI} - C_2^{LO})$ before its deadline.

4 Utilization Based Test

In this section, we consider the *implicit deadline sporadic* IMC model and assume that virtual deadlines of all *high*-criticality tasks are tuned uniformly by a scaling factor x. We propose a utilization-based sufficient test for the IMC model under EDF-VD. To aim so, we need to ensure the timely correctness of the IMC model under two modes, i.e., *low*-criticality mode and *high*-criticality mode. Following, we analyze the behaviors of the IMC model under the two modes, respectively.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2018.2789879, IEEE Transactions on Computers

4.1 Low Criticality Mode

We first ensure the schedulability of tasks when they are in *low*-criticality mode. When in *low*-criticality mode, the tasks can be considered as traditional real-time tasks scheduled by EDF with virtual deadlines (VD). The following theorem is given in [5] for tasks scheduled in *low*-criticality mode.

Theorem 1 (Theorem 1 from [5])**.** *The following condition is sufficient for ensuring that EDF-VD successfully schedules all tasks in low-criticality mode:*

$$x \geq \frac{U_{HI}^{LO}}{1 - U_{LO}^{LO}} \tag{1}$$

where $x \in (0, 1)$ is used to uniformly modify the relative deadline of high-criticality tasks.

Since the IMC model behaves as the classical MC model in *low*-criticality mode, Theorem 1 holds for the IMC model as well.

4.2 High Criticality Mode

For *high*-criticality mode, the classical MC model discards all *low*-criticality jobs after the switch to *high*-criticality mode. In contrast, the IMC model keeps *low*-criticality jobs running but with degraded quality, i.e., a shorter execution time. So the schedulability condition in [5] does not work for the IMC model in the *high*-criticality mode. Thus, we need a new test for the IMC model in *high*-criticality mode.

To derive the sufficient test in *high*-criticality mode, suppose that there is a time interval $[0, t_2]$, where a first deadline miss occurs at t_2 and t_1 denotes the time instant of the switch to *high*-criticality mode in the time interval, where $t_1 < t_2$. Assume that $\mathcal{J}$ is a minimal set of jobs generated from task set γ which leads to the first deadline miss at t_2. The minimality of $\mathcal{J}$ means that removing any job in $\mathcal{J}$ guarantees the schedulability of the rest of $\mathcal{J}$. Here, we introduce some notations for our later interpretation. Let variable η_i denote the cumulative execution time of task τ_i in the interval $[0, t_2]$. J_1 denotes a special *high*-criticality job which has switch time instant t_1 within its period (a_1, d_1), i.e, $a_1 < t_1 < d_1$. Furthermore, J_1 is the job with the earliest release time amongst all *high*-criticality jobs in $\mathcal{J}$ which execute in $[t_1, t_2)$. Moreover, we define a special type of job for *low*-criticality tasks which is useful for our later proofs.

Definition 2. *A job J_i^{IC} from low-criticality task τ_i is a imprecise carry-over (IC) job, if its absolute release time a_i is before and its absolute deadline d_i is after the switch time instant, i.e., $a_i < t_1 < d_i$.*

With the notations introduced above, we have the following,

Proposition 1 (Fact 1 from [5])**.** *All jobs in $\mathcal{J}$ that execute in $[t_1, t_2)$ have deadline $\leq t_2$.*

It is easy to observe that only jobs which have deadlines $\leq t_2$ are possible to cause a deadline miss at t_2. If a job has its deadline $> t_2$ and is still in set $\mathcal{J}$, it will contradict the minimality of $\mathcal{J}$.

Proposition 2. *The switch time instant t_1 has*

$$t_1 < (a_1 + x(t_2 - a_1)) \tag{2}$$

Proof: Let us consider a time instant $(a_1 + x(d_1 - a_1))$ which is the virtual deadline of job J_1. Since J_1 executes in time interval $[t_1, t_2)$, its virtual deadline $(a_1 + x(d_1 - a_1))$ must be greater than the switch time instant t_1. Otherwise, it should have completed its *low*-criticality execution before t_1, and this contradicts that it executes in $[t_1, t_2)$. Thus, it holds that

$$\begin{aligned} & t_1 < (a_1 + x(d_1 - a_1)) \\ \Rightarrow & t_1 < (a_1 + x(t_2 - a_1)) \quad (\text{since } d_1 \leq t_2) \end{aligned}$$

□

Proposition 3. *If a IC job J_i^{IC} has its cumulative execution equal to $(d_i - a_i)u_i^{LO}$ and $u_i^{LO} > u_i^{HI}$, its deadline d_i is $\leq (a_1 + x(t_2 - a_1))$.*

Proof: For a IC job J_i^{IC}, if it has its cumulative execution equal to $(d_i - a_i)u_i^{LO}$ and $u_i^{LO} > u_i^{HI}$, it should complete its C_i^{LO} execution before t_1. Otherwise, if job J_i^{IC} has executed time units $C_i \in [C_i^{HI}, C_i^{LO})$ at time instant t_1, it will be suspended and will not execute after t_1.

Now, we will show that when job J_i^{IC} completes its C_i^{LO} execution, its deadline is $d_i \leq (a_1 + x(t_2 - a_1))$. We prove this by contradiction. First, we suppose that J_i^{IC} has its deadline $d_i > (a_1 + x(t_2 - a_1))$ and release time a_i. As shown above, job J_i^{IC} completes its C_i^{LO} execution before t_1. Let us assume a time instant t^* as the latest time instant at which this IC job J_i^{IC} starts to execute before t_1. This means that at this time instant all jobs in $\mathcal{J}$ with deadline $\leq (a_1 + x(t_2 - a_1))$ have finished their executions. This indicates that these jobs will not have any execution within interval $[t^*, t_2]$. Therefore, jobs in $\mathcal{J}$ with release time at or after time instant t^* can form a smaller job set which causes a deadline miss at t_2. Then, it contradicts the minimality of $\mathcal{J}$. Thus, IC job J_i^{IC} with its cumulative execution time equal to $(d_i - a_i)u_i^{LO}$ and $u_i^{LO} > u_i^{HI}$ has its deadline $d_i \leq (a_1 + x(t_2 - a_1))$. □

With the propositions and notations given above, we derive an upper bound of the cumulative execution time η_i of *low*-criticality task τ_i.

Lemma 1. *For any low-criticality task τ_i, it has*

$$\eta_i \leq (a_1 + x(t_2 - a_1))u_i^{LO} + (1 - x)(t_2 - a_1)u_i^{HI} \tag{3}$$

Proof: If $u_i^{LO} = u_i^{HI}$, it is trivial to see that Lemma 1 holds. Below we focus on the case when $u_i^{LO} > u_i^{HI}$. If a system switches to *high*-criticality mode at t_1, then we know that *low*-criticality tasks are scheduled using C_i^{LO} before t_1 and using C_i^{HI} after t_1. To prove this lemma, we need to consider two cases, where τ_i releases a job within interval $(a_1, t_2]$ or it does not. We prove the two cases separately.

Case A (task τ_i releases a job within interval $(a_1, t_2]$): There are two sub-cases to be considered.

- **Sub-case 1 (No IC job):** The deadline of a job of *low*-criticality task τ_i coincides with switch time instant t_1. The cumulative execution time of *low*-criticality task τ_i within time interval $[0, t_2]$ can be bounded as follows,
$$\eta_i \leq (t_1 - 0) \cdot u_i^{LO} + (t_2 - t_1) \cdot u_i^{HI}$$
Since $t_1 < (a_1 + x(t_2 - a_1))$ according to Proposition 2

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2018.2789879, IEEE Transactions on Computers

and for *low*-criticality task τ_i it has $u_i^{LO} > u_i^{HI}$, then

$$\eta_i < \big(a_1 + x(t_2 - a_1)\big)u_i^{LO} + \big(t_2 - \big(a_1 + x(t_2 - a_1)\big)\big)u_i^{HI}$$
$$\Leftrightarrow \eta_i < (a_1 + x(t_2 - a_1))u_i^{LO} + (1-x)(t_2 - a_1)u_i^{HI}$$

- **Sub-case 2 (with IC job):** In this case, before the IC job, jobs of τ_i are scheduled with its C_i^{LO}. After the IC job, jobs of τ_i are scheduled with its C_i^{HI}. It is trivial to observe that for a IC job its maximum cumulative execution time can be obtained when it completes its C_i^{LO} within its period $[a_i, d_i]$, i.e., $(d_i - a_i)u_i^{LO}$. Considering the maximum cumulative execution for the IC job, we then have for *low*-criticality task τ_i,

$$\eta_i \le (a_i - 0)u_i^{LO} + (d_i - a_i)u_i^{LO} + (t_2 - d_i)u_i^{HI}$$
$$\Leftrightarrow \eta_i \le d_i u_i^{LO} + (t_2 - d_i)u_i^{HI}$$

Proposition 3 shows as J_i^{IC} has its cumulative execution equal to $(d_i - a_i) \cdot u_i^{LO}$, it has $d_i \le (a_1 + x(t_2 - a_1))$. Given $u_i^{LO} > u_i^{HI}$ for *low*-criticality task, we have

$$\eta_i \le d_i u_i^{LO} + (t_2 - d_i)u_i^{HI}$$
$$\Rightarrow \eta_i \le \big(a_1 + x(t_2 - a_1)\big)u_i^{LO} + \big(t_2 - \big(a_1 + x(t_2 - a_1)\big)\big)u_i^{HI}$$
$$\Leftrightarrow \eta_i \le (a_1 + x(t_2 - a_1))u_i^{LO} + (1-x)(t_2 - a_1)u_i^{HI}$$

Case B (task τ_i does not release a job within interval $(a_1, t_2]$): In this case, let J_i^{IC} denote the last release job of task τ_i before a_1 and a_i and d_i are its absolute release time and absolute deadline, respectively. If $d_i \le t_1$, we have

$$\eta_i = (a_i - 0)u_i^{LO} + (d_i - a_i) \cdot u_i^{LO} = d_i u_i^{LO}$$

If $d_i > t_1$, J_i^{IC} is a IC job. As we discussed above, the maximum cumulative execution time of IC job J_i^{IC} is $(d_i - a_i)u_i^{LO}$, so we have

$$\eta_i \le (a_i - 0)u_i^{LO} + (d_i - a_i) \cdot u_i^{LO} \Leftrightarrow \eta_i \le d_i u_i^{LO}$$

Similarly, according to Proposition 3, we obtain,

$$\eta_i \le d_i \cdot u_i^{LO} \le (a_1 + x(t_2 - a_1))u_i^{LO}$$
$$\Rightarrow \eta_i < (a_1 + x(t_2 - a_1))u_i^{LO} + \big(t_2 - \big(a_1 + x(t_2 - a_1)\big)\big)u_i^{HI}$$
$$\Leftrightarrow \eta_i < (a_1 + x(t_2 - a_1))u_i^{LO} + (1-x)(t_2 - a_1)u_i^{HI}$$

□

Lemma 1 gives the upper bound of the cumulative execution time of a *low*-criticality task in *high*-criticality mode. In order to derive the sufficient test for the IMC model in *high*-criticality mode, we need to upper bound the cumulative execution time of *high*-criticality tasks.

Proposition 4 (Fact 3 from [5])**.** *For any high-criticality task τ_i, it holds that*

$$\eta_i \le \frac{a_1}{x}u_i^{LO} + (t_2 - a_1)u_i^{HI} \tag{4}$$

Proposition 4 is used to bound the cumulative execution of the *high*-criticality tasks. Since in the IMC model the *high*-criticality tasks are scheduled as in the classical MC model, Proposition 4 holds for the IMC model as well. With Lemma 1 and Proposition 4, we can derive the sufficient test for the IMC model in *high*-criticality mode.

Theorem 2. *The following condition is sufficient for ensuring that EDF-VD successfully schedules all tasks in high-criticality mode:*

$$xU_{LO}^{LO} + (1-x)U_{LO}^{HI} + U_{HI}^{HI} \le 1 \tag{5}$$

Proof: Let N denote the cumulative execution time of all tasks in $\gamma = \gamma_{LO} \cup \gamma_{HI}$ over interval $[0, t_2]$. We have

$$N = \sum_{\forall \tau_i \in \gamma_{LO}} \eta_i + \sum_{\forall \tau_i \in \gamma_{HI}} \eta_i$$

By using Lemma 1 and Proposition 4, N is bounded as follows

$$N \le \sum_{\forall \tau_i \in \gamma_{LO}} \Big(\big(a_1 + x(t_2 - a_1)\big)u_i^{LO} + (1-x)(t_2 - a_1)u_i^{HI} \Big) + \sum_{\forall \tau_i \in \gamma_{HI}} \Big(\frac{a_1}{x}u_i^{LO} + (t_2 - a_1)u_i^{HI} \Big)$$
$$\Leftrightarrow N \le (a_1 + x(t_2 - a_1))U_{LO}^{LO} + (1-x)(t_2 - a_1)U_{LO}^{HI} + \frac{a_1}{x}U_{HI}^{LO} + (t_2 - a_1)U_{HI}^{HI}$$
$$\Leftrightarrow N \le a_1(U_{LO}^{LO} + \frac{U_{HI}^{LO}}{x}) + x(t_2 - a_1)U_{LO}^{LO} + (1-x)(t_2 - a_1)U_{LO}^{HI} + (t_2 - a_1)U_{HI}^{HI} \tag{6}$$

Since the tasks must be schedulable in *low*-criticality mode, the condition given in Theorem 1 holds and we have $1 \ge (U_{LO}^{LO} + \frac{U_{HI}^{LO}}{x})$. Hence,

$$N \le a_1 + x(t_2 - a_1)U_{LO}^{LO} + (1-x)(t_2 - a_1)U_{LO}^{HI} + (t_2 - a_1)U_{HI}^{HI} \tag{7}$$

Since time instant t_2 is the first deadline miss, it means that there is no idle time instant within interval $[0, t_2]$. Note that if there is an idle instant, jobs from set $\mathcal{J}$ which have release time at or after the latest idle instant can form a smaller job set causing deadline miss at t_2 which contradicts the minimality of $\mathcal{J}$. Then, we obtain

$$N = \Big(\sum_{\forall \tau_i \in \gamma_{LO}} \eta_i + \sum_{\forall \tau_i \in \gamma_{HI}} \eta_i \Big) > t_2$$
$$\Rightarrow a_1 + x(t_2 - a_1)U_{LO}^{LO} + (1-x)(t_2 - a_1)U_{LO}^{HI} + (t_2 - a_1)U_{HI}^{HI} > t_2$$
$$\Leftrightarrow x(t_2 - a_1)U_{LO}^{LO} + (1-x)(t_2 - a_1)U_{LO}^{HI} + (t_2 - a_1)U_{HI}^{HI} > t_2 - a_1$$
$$\Leftrightarrow xU_{LO}^{LO} + (1-x)U_{LO}^{HI} + U_{HI}^{HI} > 1$$

By taking the contrapositive, we derive the sufficient test for the IMC model when it is in *high*-criticality mode:

$$xU_{LO}^{LO} + (1-x)U_{LO}^{HI} + U_{HI}^{HI} \le 1$$

□

Note that if $U_{LO}^{HI} = 0$, i.e., no *low*-criticality tasks are scheduled after the system switches to *high*-criticality mode, our Theorem 2 is the same as the sufficient test (Theorem 2 in [5]) for the classical MC model in *high*-criticality mode. Hence, our Theorem 2 actually is a generalized schedulability condition for (I)MC tasks under EDF-VD.

By combining Theorem 1 (see Section 4.1) and our Theorem 2, we prove the following theorem,

Theorem 3. *Given an IMC task set, if*

$$U_{HI}^{HI} + U_{LO}^{LO} \le 1 \tag{8}$$

then the IMC task set is schedulable by EDF; otherwise, if

$$\frac{U_{HI}^{LO}}{1 - U_{LO}^{LO}} \le \frac{1 - (U_{HI}^{HI} + U_{LO}^{HI})}{U_{LO}^{LO} - U_{LO}^{HI}} \tag{9}$$

where

$$U_{HI}^{HI} + U_{LO}^{HI} < 1 \text{ and } U_{LO}^{LO} < 1 \text{ and } U_{LO}^{LO} > U_{LO}^{HI} \tag{10}$$

then this IMC task set can be scheduled by EDF-VD with a deadline scaling factor x arbitrarily chosen in the following

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2018.2789879, IEEE Transactions on Computers

range

$$x \in \left[\frac{U_{HI}^{LO}}{1-U_{LO}^{LO}}, \quad \frac{1-(U_{HI}^{HI}+U_{LO}^{HI})}{U_{LO}^{LO}-U_{LO}^{HI}} \right]$$

Proof: Total utilization $U \leq 1$ is the exact test for EDF on a uniprocessor system. If the condition in (8) is met, the given task set is *worst-case reservation* [5] schedulable under EDF, i.e., the task set can be scheduled by EDF without deadline scaling for *high*-criticality tasks and execution budget reduction for *low*-criticality tasks. Now, we prove the second condition given by (9). From Theorem 1, we have,

$$x \geq \frac{U_{HI}^{LO}}{1-U_{LO}^{LO}}$$

From Theorem 2, we have

$$xU_{LO}^{LO} + (1-x)U_{LO}^{HI} + U_{HI}^{HI} \leq 1$$
$$\Leftrightarrow x \leq \frac{1-(U_{HI}^{HI}+U_{LO}^{HI})}{U_{LO}^{LO}-U_{LO}^{HI}}$$

Therefore, if $\frac{U_{HI}^{LO}}{1-U_{LO}^{LO}} \leq \frac{1-(U_{HI}^{HI}+U_{LO}^{HI})}{U_{LO}^{LO}-U_{LO}^{HI}}$, the schedulability conditions of both Theorem 1 and 2 are satisfied. Thus, the IMC tasks are schedulable under EDF-VD. □

5 Speedup Factor

The speedup factor bound is a useful metric to compare the worst-case performance of different MC scheduling algorithms. The following is the definition of the speedup factor for an MC scheduling algorithm.

Definition 3 (from [5])**.** *The speedup factor of an algorithm $\mathcal{A}$ for scheduling MC systems is the smallest real number $f \geq 1$ such that any task system that is schedulable by a hypothetical optimal clairvoyant scheduling algorithm*[1] *on a unit-speed processor is correctly scheduled by algorithm $\mathcal{A}$ on a speed-f processor.*

Generally speaking, by increasing a processor's speed a non-optimal scheduling algorithm is able to schedule the task sets which are deemed to be unschedulable by the non-optimal scheduling algorithm but schedulable by an optimal scheduling algorithm on the processor without speed increase. The speedup factor actually computes how much the processor needs to speed up such that the non-optimal scheduling algorithm achieves the same scheduling performance as an optimal scheduling algorithm.

For the sake of understanding, we give a simple example.

Example 1. *Given a task set which is presumptively schedulable under an optimal scheduling algorithm on a platform, we have two scheduling algorithms, A and B, which cannot schedule the task set on the same platform. To successfully schedule the task set by using algorithms A and B, we can speed up the execution frequency of the platform (because the execution time of tasks will be reduced). If algorithms A and B need to speed up the platform at least by 1.5 and 2 times, respectively, to ensure the schedulability of the task set, then algorithm A is said to be better than algorithm B in terms of scheduling performance due to the lower hardware cost, i.e., the smaller scaling factor. It is evident to see that if we speed up the platform more than the minimal scaling number, their schedulability will always be guaranteed but unnecessary.*

As seen from the example, a smaller speedup factor requires a lower hardware cost and in turn indicate the better scheduling performance for a non-optimal scheduling algorithm. The speedup factor bound for the classical MC model under EDF-VD is known to be $4/3$ [5].

Following, we prove the speedup factor of the IMC model under EDF-VD scheduling. For notational simplicity, we define

$$U_{HI}^{HI} = c, \quad U_{HI}^{LO} = \alpha \times c$$
$$U_{LO}^{LO} = b, \quad U_{LO}^{HI} = \lambda \times b$$

where $\alpha \in (0,1]$ and $\lambda \in [0,1]$. α denotes the utilization ratio between U_{HI}^{LO} and U_{HI}^{HI}, while λ denotes the utilization ratio between U_{LO}^{HI} and U_{LO}^{LO}.

First, let us analyze the speedup factor of two corner cases. When $\alpha = 1$, i.e., $U_{HI}^{LO} = U_{HI}^{HI}$, this means that there is no mode-switch. Therefore, the task set is scheduled by the traditional EDF, i.e., the task set is schedulable if $U_{LO}^{LO} + U_{HI}^{LO} \leq 1$. Since EDF is the optimal scheduling algorithm on a uniprocessor system, the speedup factor is 1. When $\lambda = 1$, i.e., $U_{LO}^{LO} = U_{LO}^{HI}$, if the task set is schedulable in *high*-criticality mode, it must hold $U_{HI}^{HI} + U_{LO}^{LO} \leq 1$ by Theorem 2. Then it is scheduled by the traditional EDF and thus the speedup factor is 1 as well.

In this paper, instead of generating a single speedup factor bound, we derive a speedup factor function with respect to (α, λ). This speedup factor function enables us to quantify the suboptimality of EDF-VD for the IMC model in terms of speedup factor (by our proposed sufficient test) and evaluate the impact of the utilization ratio on the schedulability of an IMC task set under EDF-VD.

First, we strive to find a minimum speed s ($\leq$1) for a clairvoyant optimal MC scheduling algorithm such that any implicit-deadline IMC task set which is schedulable by the clairvoyant optimal MC scheduling algorithm on a speed-s processor can satisfy the schedulability test given in Theorem 3, i.e., schedulable under EDF-VD on a unit-speed processor.

Lemma 2. *Given $b, c \in [0,1]$, $\alpha \in (0,1)$, $\lambda \in [0,1)$, and*

$$\max\{b + \alpha c, \lambda b + c\} \leq S(\alpha, \lambda) \tag{11}$$

where

$$S(\alpha,\lambda) = \frac{(1-\alpha\lambda)((2-\alpha\lambda-\alpha)+(\lambda-1)\sqrt{4\alpha-3\alpha^2})}{2(1-\alpha)(\alpha\lambda-\alpha\lambda^2-\alpha+1)}$$

then it guarantees

$$\frac{\alpha c}{1-b} \leq \frac{1-(c+\lambda b)}{b-\lambda b} \tag{12}$$

Proof: The complete proof is given in [11]. □

Lemma 2 shows that any IMC task set that is schedulable by an optimal clairvoyant MC scheduling algorithm on a speed-$S(\alpha,\lambda)$ is schedulable by EDF-VD on a unit-speed processor. Therefore, the speedup factor of EDF-VD is $1/S(\alpha,\lambda)$.

Theorem 4. *The speedup factor of EDF-VD with IMC task sets is*

$$f = \frac{2(1-\alpha)(\alpha\lambda-\alpha\lambda^2-\alpha+1)}{(1-\alpha\lambda)((2-\alpha\lambda-\alpha)+(\lambda-1)\sqrt{4\alpha-3\alpha^2})}$$

1. A 'clairvoyant' scheduling algorithm knows all run-time information, e.g., when the mode switch will occur, prior to run-time.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2018.2789879, IEEE Transactions on Computers

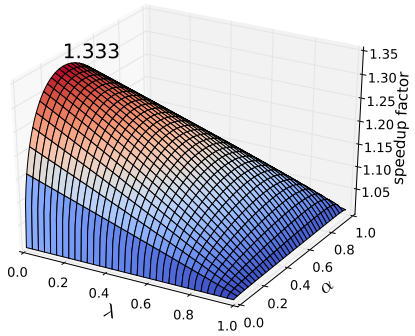


Figure 2: 3D image of speedup factor w.r.t α and λ

λ \ α	0.1	0.3	1/3	0.5	0.7	0.9	1
0	1.254	1.332	**1.333**	1.309	1.227	1.091	1
0.1	1.231	1.308	1.310	1.293	1.219	1.090	1
0.3	1.183	1.256	1.259	1.254	1.201	1.087	1
0.5	1.134	1.195	1.200	1.206	1.174	1.083	1
0.7	1.082	1.126	1.130	1.143	1.133	1.074	1
0.9	1.028	1.046	1.048	1.056	1.061	1.048	1
1	1	1	1	1	1	1	1

Table 2: Speedup factor w.r.t α and λ

The speedup factor is shown to be a function with respect to α and λ. Figure 2 plots the 3D image of this function and Table 2 lists some of the values with different α and λ. By doing some calculus, we obtain the maximum value 1.333 $(4/3)$ of the speedup factor function when $\lambda = 0$ and $\alpha = \frac{1}{3}$, which is highlighted in Figure 2 and Table 2. We see that the speedup factor bound is achieved when the task set is a classical MC task set. From Figure 2 and Table 2, we observe different trends for the speedup factor with respect to α and λ.

- First, given a fixed λ, the speedup factor is not a monotonic function with respect to α. The relation between α and the speedup factor draws a downward parabola. Therefore, a straightforward conclusion regarding the impact of α on the speedup factor cannot be drawn.
- Given a fixed α, the speedup factor is a monotonic decreasing function with respect to increasing λ. It is seen that increasing λ leads to a smaller value of the speedup factor. *This means that a larger λ brings a positive effect on the schedulability of an IMC task set*.

Note that the schedulability test and speedup factor results of this paper also apply to the *elastic mixed-criticality* (EMC) model proposed in [17], where the periods of low-criticality tasks are scaled up in high-criticality mode. The detailed proof is provided in [11].

6 DBF-Based Test

Section 4 provides a utilization based sufficient test and the speedup factor derived in Section 5 quantifies the worst-case scheduling performance of EDF-VD with our proposed utilization-based test[2]. The utilization-based test is concise and easy to check the schedulability for the *implicit deadline* IMC model, but it also has some shortcomings. 1) The proposed utilization-based test is not applicable to the constrained deadline IMC model, where $D_i \leq T_i$. 2) The virtual deadlines of *high*-criticality tasks cannot be tunned individually and in turn this uniformly deadline settings hurts the scheduling performance of EDF-VD scheduling algorithm [7].

In this section, we propose a DBF-based schedulability test to address the shortcomings of the utilization-based test. Demand bound function (DBF) was proposed in [29] to test the schedulability of conventional real-time tasks (i.e., only one criticality level) under preemptive EDF. Basically, DBF computes the maximum cumulative execution time of a task within a time interval.

Definition 4. *For a task τ_i and a time interval t, $dbf(\tau_i, t)$ determines the maximum cumulative execution time of jobs generated by task τ_i and with both release time and deadline within the time interval $[0, t)$.*

For a task set γ, its total demand requirement within a time interval is the summation of demand requirement of all individual tasks in γ.

$$dbf(\gamma, t) = \sum_{\tau_i \in \gamma} dbf(\tau_i, t)$$

To check the schedulability of a task set γ, it just needs to check whether for any time instant within a sufficient long time interval $t_{\max}$ the following holds,

$$\forall t \leq t_{\max}, \quad dbf(\gamma, t) \leq t$$

If the above condition holds, then the task set is said to be *schedulable*. Otherwise, it reports unschedulable.

To extend the DBF analysis framework to the IMC model, we need to analyze the schedulability of the IMC model in *low*-criticality and *high*-criticality mode, respectively.

6.1 Schedulability Analysis in Low-Criticality Mode

If there is no overrun occuring to any *high*-criticality task, the schedulability of task set γ can be checked by using the existing test.

Proposition 5 ([7] [29]). *An IMC Task set γ is schedulable in low-criticality mode iff*

$$\forall 0 \leq t \leq t_{max}, \quad \sum_{\tau_i \in \gamma} dbf(\tau_i, t) \leq t \tag{13}$$

6.2 Schedulability Analysis in High-Criticality Mode

To compute the maximum demand requirement in *high*-criticality mode, we need to take into account the system switch behavior. For the sake of simplicity, we also use $\text{dbf}(\tau_i, t_s, t)$ to denote the demand bound function of a task τ_i in *high*-criticality mode where $t_s (\leq t)$ is the time instant at which the system switches to *high*-criticality mode within a time interval t.

6.2.1 Low-Criticality Tasks

To precisely derive the demand requirement of a *low*-criticality task and eliminate pessimism, we need to accurately depict the execution status of an IC job (Definition 2 in Section 4.2). For sake of concise interpretation, in the remainder of paper, we define $\text{mod}(\tau_i, t_s)$

$$\text{mod}\ (\tau_i, t_s) = t_s - \left\lfloor \frac{t_s}{T_i} \right\rfloor T_i \tag{14}$$

2. See the definition of the speedup factor.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2018.2789879, IEEE Transactions on Computers

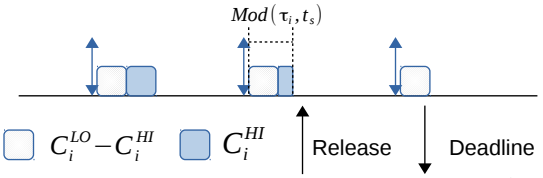


Figure 3: Case 2: $C_i^{HI} \leq \text{mod}(\tau_i, t_s) \leq C_i^{LO}$

$\text{mod}\ (\tau_i, t_s)$ actually computes the time interval between time instant t_s and the release time of the IC job generated by task τ_i. Additionally, we use $[\![A]\!]_0$ to denote $\max(A, 0)$.

The following proposition determines the demand requirement of the IC job of a *low*-criticality task.

Proposition 6. *For a low-criticality task τ_i, the DBF of its IC job J_i^{IC} can be computed as follows,*

$$dbf(J_i^{IC}, t_s, t) = \begin{cases} C_i^{HI}; & mod(\tau_i, t_s) \leq C_i^{HI} \\ & \& \quad r(J_i^{IC}) + D_i \leq t \\ mod(\tau_i, t_s); & C_i^{HI} < mod(\tau_i, t_s) < C_i^{LO} \\ & \& \quad r(J_i^{IC}) + D_i \leq t \\ C_i^{LO}; & mod(\tau_i, t)s) >= C_i^{LO} \\ & \& \quad r(J_i^{IC}) + D_i \leq t \\ 0; & r(J_i^{IC}) + D_i > t \end{cases} \tag{15}$$

where $r(J_i^{IC})$ denotes the release time of IC job J_i^{IC}.

Proof: According to the definition of DBF, a task or job that demands execution within a time interval must have both release time and deadline within this time interval. Therefore, if $r(J_i^{IC}) + D_i > t$, job J_i^{IC} is deemed to have no demand requirement within the time interval.

Then, with condition $r(J_i^{IC}) + D_i \leq t$, we can compute the demand requirement of an IC job by considering three different cases,

- **Case 1:** $\text{mod}(\tau_i, t_s) < C_i^{HI}$
  We see that the interval between the switch time instant and the release time of IC job J_i^{IC} is $< C_i^{HI}$. For IC job J_i^{IC}, if it does not complete its C_i^{HI} execution before the switch time, it will continue to complete the left execution to C_i^{HI} after switch time instant t_s. Therefore, $dbf(J_i^{IC}, t_s, t) = C_i^{HI}$.
- **Case 2:** $C_i^{HI} \leq \text{mod}(\tau_i, t_s) < C_i^{LO}$
  The interval between switch time instant t_s and the release time of IC job J_i^{IC} is $\geq C_i^{HI}$ but $< C_i^{LO}$. In this case, the demand of J_i^{IC} is maximized if job J_i^{IC} starts its execution immediately at its release time and continuously executes until the switch time instant t_s, i.e., $\text{mod}(\tau_i, t_s)$. Fig. 3 depicts this scenario. Therefore, $dbf(J_i^{IC}, t_s, t) = \text{mod}(\tau_i, t_s)$
- **Case 3:** $C_i^{LO} \leq \text{mod}(\tau_i, t_s)$
  The interval between switch time instant t_s and the release time of IC job is $> C_i^{LO}$. In this case, the demand of job J_i^{IC} is maximized if job J_i^{IC} completes its C_i^{LO} before or at switch time instant t_s. Therefore, $dbf(J_i^{IC}, t_s, t) = C_i^{LO}$

With the three cases explained above, the DBF of IC job J_i^{IC} can be computed by Eq. (15). □

Given the DBF of IC job J_i^{HI}, we can compute the demand requirement of *low*-criticality task τ_i as follows,

Lemma 3. *Given a time interval $[0, t]$ and a switch time t_s, the demand requirement of low-criticality task τ_i is computed by*

$$\begin{aligned} dbf^{LO}(\tau_i, t_s, t) = &\left\lfloor \frac{t_s}{T_i} \right\rfloor C_i^{LO} + dbf(J_i^{IC}, t_s, t) \\ &+ \left(\left[\!\left[\left\lfloor \frac{t - D_i}{T_i} \right\rfloor \right]\!\right]_0 - \left\lfloor \frac{t_s}{T_i} \right\rfloor \right) C_i^{HI} \end{aligned} \tag{16}$$

where $dbf(J_i^{IC}, t_s, t)$ is computed by Eq. (15).

Proof: First, $\lfloor \frac{t_s}{T_i} \rfloor$ computes how many jobs *low*-criticality task τ_i has generated before the IC job J_i^{IC}, and $\lfloor \frac{t_s}{T_i} \rfloor C_i^{LO}$ determines the demand requirement of those jobs which are executed before J_i^{IC}.

After J_i^{IC}, jobs from *low*-criticality task τ_i are scheduled with C_i^{HI}. Then, $\lfloor \frac{t-D_i}{T_i} \rfloor$ computes how many jobs with deadlines before time instant t are generated by *low*-criticality task τ_i excluding J_i^{IC}. If $t - D_i < 0$, there is no any demand from τ_i within $(0, t)$. Hence, $([\![\lfloor \frac{t-D_i}{T_i} \rfloor]\!]_0 - \lfloor \frac{t_s}{T_i} \rfloor) C_i^{HI}$ determines the total demand requirement of jobs executed after J_i^{IC} with deadlines smaller than t.

With $\lfloor \frac{t_s}{T_i} \rfloor$, $([\![\lfloor \frac{t-D_i}{T_i} \rfloor]\!]_0 - \lfloor \frac{t_s}{T_i} \rfloor) C_i^{HI}$ and the demand requirement of IC job J_i^{IC} determined by Proposition 6, if the system switches to *high*-criticality mode at time instant t_s, the demand requirement of *low*-criticality task τ_i within interval $[0, t]$ can be computed by Eq (16). □

6.2.2 High-Criticality Tasks

The *high*-criticality tasks in the IMC model behave exactly the same as in the classical MC model, so the existing method of computing the demand requirement for *high*-criticality tasks can be reused in this case. Here, we refer to the method proposed in [8] because it provides the better schedulability for the classical MC model. The computing method proposed in [8] is resummarized in [22] as follows,

Proposition 7 ([22]). *If the system switches to high-criticality mode at time instant t_s within time interval $[0, t]$, the demand requirement of a high-criticality task τ_i within time interval $(0, t]$ can be computed by the following equation.*

$$dbf^{HI}(\tau_i, t_s, t) = \begin{cases} \lfloor \frac{t_s}{T_i} \rfloor C_i^{LO} + dbf(J_i^A, t_s, t) \\ \qquad if\ t - t_i \leq D_i - D_i^{LO} \\ b_i C_i^{LO} + dbf(J_i^A, t_s, t) + a_i C_i^{HI} \\ \qquad if\ t - t_i \geq D_i \\ \max\left\{dbf^{HI}(1), dbf^{HI}(2)\right\} \\ \qquad if\ D_i - D_i^{LO} < t - t_s < D_i \end{cases} \tag{17}$$

where D_i and D_i^{LO} denote the actual deadline and virtual deadline of task τ_i, respectively. b_i and a_i in Case 2 are computed as follows,

$$b_i = \lfloor \frac{t_s - (t - D_i - \lfloor (t - D_i)/T_i \rfloor \times T_i}{T_i} \rfloor$$

$$a_i = \lfloor \frac{t - D_i}{T_i} \rfloor - b_i$$

$dbf^{HI}(1)$ and $dbf^{HI}(2)$ denote the first and second case given in Eq. (17), respectively.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2018.2789879, IEEE Transactions on Computers

6.3 DBF Schedulability Test

Given Lemma 3 and Proposition 7, we derive the DBF-based schedulability test for the IMC model.

Theorem 5. *An IMC task set γ is schedulable under EDF-VD, if* $\forall 0 \leq t \leq t_{\max}, \quad \forall 0 \leq t_s \leq t$

$$\sum_{\forall \tau_i \in \gamma^{LO}} dbf^{LO}(\tau_i, t_s, t) + \sum_{\forall \tau_j \in \gamma^{HI}} dbf^{HI}(\tau_j, t_s, t) \leq t \quad (18)$$

where $dbf^{LO}(\tau_i, t_s, t)$ and $dbf^{HI}(\tau_j, t_s, t)$ are given in Eq. (16) and Eq. (17), respectively. $t_{\max} = lcm\{T_1, T2, \cdots, T_n\}$.

Here, *lcm* is the least common multiply operation and t_{max} is also called hyper-period. Although EDF-VD is a variation of EDF, many optimization and improvement techniques proposed for EDF are not applicable to EDF-VD, like more efficient $t_{\max}$ given in [30]. Therefore, we use the most generic bound of *lcm* to test the schedulability and in future it is worth investigating a more efficient bound for our test. When a switch from *low*-criticality mode to *high*-criticality mode occurs at t_s within time interval $(0, t]$, the left-hand-side of inequality (18) computes the maximum cumulative demand of task set γ. And for any time interval within a sufficient long time interval $t_{\max}$, if Theorem 5 holds, then all jobs from task set τ_i are ensured to receive enough time to complete the execution by their deadlines. Thus, task set γ is schedulable under EDF-VD. In the implementation of this test, since a miss (i.e., the demand requirements are greater than the time interval) only happens at a task's deadline [30], the complexity can be reduced by only checking at all absolute virtual deadlines and real deadlines within this interval.

6.4 Deadline Tuning Algorithm

In Section 6.3, we establish the DBF-based schedulability test for the IMC model. In the DBF-based test, properly setting virtual deadlines of *high*-criticality tasks is crucially important to shape the demand requirement of tasks within time interval such that the schedulability of the task set can be ensured.

The existing deadline tunning algorithm, such as the one in [7] [8], gradually reduces the virtual deadlines of *high*-criticality tasks until the schedulability of the task set is guaranteed in both *low*-criticality and *high*-criticality mode. Basically, if the algorithm finds the task set is unschedulable at a time instant in *high*-criticality mode, it terminates and tunes down the virtual deadline of a selected *high*-criticality task. However, tunning down virtual deadlines may cause unschedulability of the task set in *low*-criticality mode. When this happens, a new virtual deadline setting has to be found and the complex *high*-criticality DBF test has to be carried out again. Typically, this procedure repeats many times until a valid virtual deadline configuration is retained, thus leading to high time cost.

To address this issue, we propose a new and efficient deadline tunning algorithm, namely DTA, which can significantly reduce the time cost in setting proper virtual deadlines for *high*-criticality tasks. Two features in our algorithm boost the efficiency of the tunning procedure. *The first feature* is that in DTA algorithm, instead of reducing virtual deadlines gradually, we strive to find the minimum virtual deadlines configuration which can ensure the schedulability of the IMC task set in *low*-criticality mode. Then, when we check the schedulability of an IMC task set in *high*-criticality mode with the minimum virtual deadline settings, our DTA algorithm just goes through the complex test in Theorem 5 only **once**.

The second feature of our DTA algorithm is that we integrate the Quick-convergence Processor-demand Analysis (QPA) [30] into our algorithm. QPA approach has shown that it can effectively reduce the timing complexity of DBF test. For the details of QPA, interesting readers are referred to [30]. With the two features, our algorithm provides a very efficient approach to tune virtual deadlines for *high*-criticality tasks. In addition, experimental results show that our algorithm achieves this high efficiency with negligible schedulability loss.

Algorithm 1: Deadline Tuning Algorithm (DTA)

```
   input : An IMC taskset γ
   output: schedulability of taskset γ
1  for ∀τ_i ∈ γ do
2      if L_i = HI then
3          D'_i = C_i^LO
4      else
5          D'_i = D_i
6  d_min = min{d_i | d_i = D'_i}
7  t = max{d_i | d_i = kT_i + D'_i ∧ d_i < t_max}
8  while t > d_min do
9      if dbf^LO(γ, t) < t then
10         t = dbf^LO(γ, t)
11     else
12         if t = dbf^LO(γ, t) then
13             t = max{d_i | d_i = kT_i + D'_i ∧ d_i < t}
14         else
15             τ_c = findTask(γ, t)
16             if τ_c = ⊥ then
17                 return Failure
18             D'_c = t − ⌊t/T_c⌋T_c + 1
19             t = max{d_i | d_i = kT_i + D'_i ∧ d_i < t_max}
20 if dbf^LO(γ, t) ≤ d_min and Pass Theorem 5 then
21     return Schedulable
22 return Failure
```

The pseudo-code of our DTA algorithm is given in Algorithm 1. Basically, the algorithm first sets all *high*-criticality tasks' virtual deadlines equal to C_i^{LO} (Line 1-5). From Line 6-20, we use the QPA approach to check the schedulability of *low*-criticality mode. If it reports unschedulable at time instant t, function findTask selects a *high*-criticality task τ_c which contributes the most demand requirement in time interval $(0, t]$ and then the algorithm increases the virtual deadline of *high*-criticality task τ_i such that the demand requirement within $(0, t]$ is reduced. The process repeats until a valid virtual deadline configuration is obtained for *low*-criticality mode. Then, the algorithm proceeds to check the schedulability in *high*-criticality mode. If it passes the *high*-criticality test proposed in Theorem 5, it reports schedulable. Otherwise, a failure is detected. *Note that since Theorem 5 also works with the classical MC model, DTA algorithm can be applied to the classical MC model.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2018.2789879, IEEE Transactions on Computers

Complexity: In our DTA algorithm, the *while loop* in the worst case needs to execute $\sum_{\forall \tau_i \in \gamma^{HI}}(D_i - C_i)$ times, where function findTask goes through all *high*-criticality tasks to find a candidate and thus the complexity of function findTask is $|\gamma^{HI}|$. The complexity of the test in Theorem 5 is t^2_{max}. Therefore, the complexity of our DTA algorithm is $-O(|\gamma^{HI}| \cdot \sum_{\forall \tau_i \in \gamma^{HI}}(D_i - C_i) + t^2_{max})$, whereas the complexity of the deadline tunning algorithm in [8] is $O(t^2_{max} \cdot \sum_{\forall \tau_i \in \gamma^{HI}}(D_i - C_i))$.

7 Experimental Evaluation

In this section, we conduct experiments to evaluate the effectiveness of the proposed sufficient tests for the IMC model in terms of schedulable task sets (acceptance ratio). Moreover, we conduct experiments to verify the observations obtained in Section 5 regarding the impact of α and λ on the average acceptance ratio. Finally, we carry out experiments to show the efficiency of our proposed deadline tunning algorithm. The experiments are based on randomly generated IMC tasks. We use a task generation approach, similar to that used in [8] [7], to randomly generate IMC task sets for the evaluation. Each task τ_i is generated based on the following procedure,

- pCriticality is the probability that the generated task is a *high*-criticality task; pCriticality$\in [0, 1]$.
- Period T_i is randomly selected from range $[100, 1000]$.
- In order to have sufficient number of tasks in a task set, utilization u_i is randomly drawn from the range$[0.05, 0.2]$.
- For any task τ_i, $C_i^{LO} = u_i * T_i$.
- $R \geq 1$ denotes the ratio C_i^{HI}/C_i^{LO} for every *high*-criticality task. If $L_i = HI$, we set $C_i^{HI} = R * C_i^{LO}$. It is easy to see that α used in the speedup factor function is equal to $\frac{1}{R}$;
- $\lambda \in (0, 1]$ denotes the ratio C_i^{HI}/C_i^{LO} for every *low*-criticality task. If $L_i = LO$, we set $C_i^{HI} = \lambda * C_i^{LO}$.

In the experiment, we generate IMC task sets with different target utilization U. Each task set is generated as follows. Given a target utilization U, we first initialize an empty task set. Then, we generate task τ_i according to the task generation procedure introduced above and add the generated task to the task set. The task set generation stops as we have

$$U - 0.05 \leq U_{avg} \leq U + 0.05$$

where

$$U_{avg} = \frac{U^{LO} + U^{HI}}{2}$$

is the average total utilization of the generated task set. If adding a new task makes $U_{avg} > U + 0.05$, then the added task will be removed and a new task will be generated and added to the task set till the condition is met.

7.1 Schedulability Evaluation

In this section, we thoroughly compare the two proposed schedulability tests, i.e., Theorem 3 and Theorem 5, to all existing works considering the IMC model, the AMC approach [9] and the MC-Fluid approach [15] in terms of average acceptance ratio. In this experiment, AMC denotes the AMC approach [9], MCF denotes the MC-Fluid approach [15], UTIL and DBF denote the tests given in Theorem 3 and 5, respectively.

We setup this experiment as follows. R is randomly selected from a uniform distribution $[1.5, 2.5]$. With different λ and pCriticality settings, we vary U_{avg} from 0.4 to 0.95 with step of 0.05, to evaluate the effectiveness of the proposed sufficient test in terms of the average acceptance ratios. We generate 10,000 task sets for each given U_{avg}. Due to space limitations, we only present the experimental results when pCriticality$= 0.3$ and pCriticality$= 0.5$. Experimental results with different settings show the similar trend as we observe from these two experimental configurations. The results are plotted in Figure 4 and 5, where the x-axis denotes the varying U_{avg} and the y-axis denotes the acceptance ratio. We observe the following:

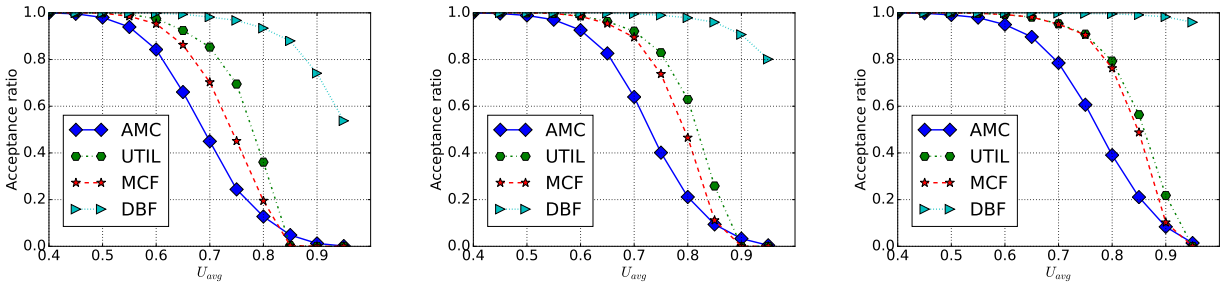
1) DBF performs the best among the four approaches in terms of average acceptance ratio. In particular, with the increasing U_{avg}, the acceptance ratio of UTIL and MCF drops rapidly, whereas DBF still remains a relatively high acceptance ratio.
2) Although MC-Fluid scheduling is more effective than EDF-VD scheduling on multiprocessor systems, we see that on the uniprocessor system the UTIL achieves the same acceptance ratio as MCF. Even in some cases UTIL is slightly better than MCF.
3) Although we can briefly deem that AMC corresponds to fixed-priority scheduling and EDF-VD corresponds to EDF in the traditional real-time literature, the introduced concept of mode switch in MC model actually undermines the optimality of EDF and therefore we cannot really ensure whether EDF-VD is always better than AMC in terms of acceptance ratios, i.e., schedulability. When $U_{avg} \in [0.5, 0.8]$, EDF-VD (UTIL) always outperforms AMC in terms of acceptance ratio. However, if $U_{avg} > 0.8$ and $\lambda = 0.3$ or 0.5, AMC performs better than EDF-VD. The same trend is also found for the classical MC model under EDF-VD and AMC, see the comparison in [7].
4) By comparing sub-figures in Figure 5, we see that the average acceptance ratio improves when λ increases. This confirms the observation for the speedup factor we obtained in Section 5. The increasing λ leads to a smaller speedup factor. As a result, it provides a better schedulability. When λ increases, we surprisingly notice that not only UTIL improves its acceptance ratio but the acceptance ratios of other approaches [9] also improve.

In Figure 4 and 5, we evaluate different approach with different values of λ and pCriticality. Next, we study the effect of varying period selection range. We use a *weighted acceptance ratio* to plot the experimental results. The *weighted acceptance ratio* is computed as follows [7],

$$A(\mathcal{U}) \stackrel{\text{def}}{=} \frac{\sum_{U \in \mathcal{U}}(U \cdot A(U))}{\sum_{U \in \mathcal{U}} U}$$

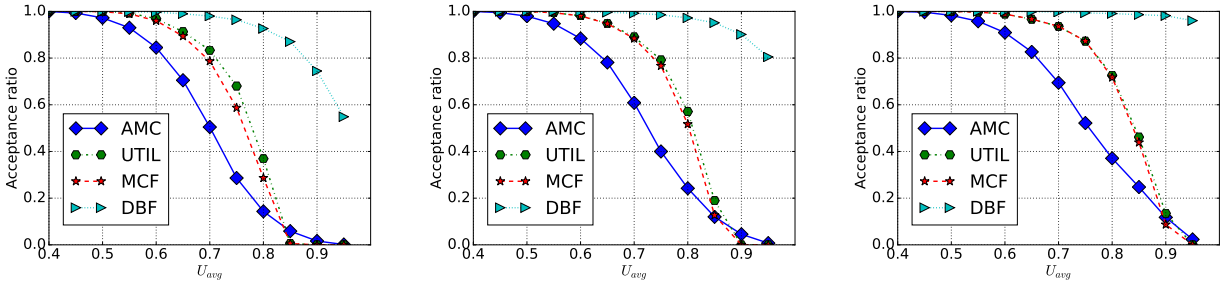
where $\mathcal{U}$ is a set of average utilizations, in this experiment, we have $\mathcal{U} = [0.5, 0.6, 0.7, 0.8]$. $U \in \mathcal{U}$ is one average utilization generated. $A(U)$ is the acceptance ratio when U is given. The plots with the weighted acceptance ratio allow to reduce one

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2018.2789879, IEEE Transactions on Computers



(a) $\lambda = 0.3$ (b) $\lambda = 0.5$ (c) $\lambda = 0.7$

Figure 4: Varying U_{avg} with different λ and pcriticality=0.3



(a) $\lambda = 0.3$ (b) $\lambda = 0.5$ (c) $\lambda = 0.7$

Figure 5: Varying U_{avg} with different λ and pcriticality=0.5

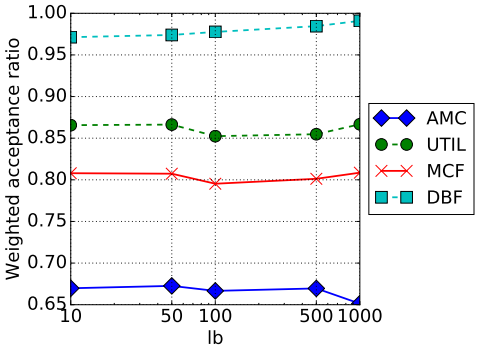


Figure 6: Varying the lower bound of period range

dimension in plots and can evaluate the effect of a varying parameter. We can notice that from the equation above, in the plots with the weighted acceptance ratio, the more importance is given to the acceptance ratio for a larger utilization value. To vary the period, we set the period range $[lb, 10000]$ and change the value of lb such that the effect of different period range can be evaluated. The experimental results are plotted in Figure 6. We can see that for all approaches the weighted acceptance ratios remain steady over different period ranges, so we can conclude that the different period ranges do not affect the scheduling performance of MC scheduling algorithms.

7.2 Impact of α and λ

Above, we compare our proposed sufficient test to the existing AMC approach and MC-Fluid approach. In this section, we conduct experiments to further evaluate the impact of λ and α $(1/R)$ on the acceptance ratio with respect to the utilization based test given in Theorem 3. In this experiment, we select

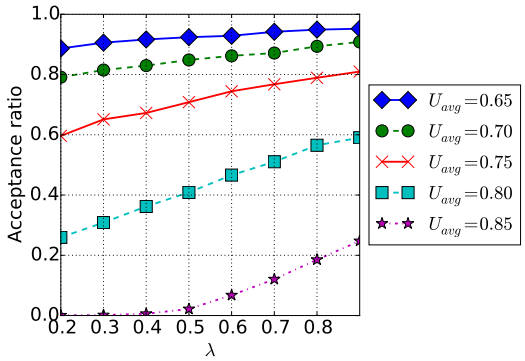


Figure 7: Impact of λ

$U_{avg} = \{0.65, 0.7, 0.75, 0.8, 0.85\}$ to conduct experiments. We fix U_{avg} to a certain utilization and vary λ and α to evaluate the impact.

We first show the results for λ. The results are depicted in Figure 7, where the x-axis denotes the value of λ from 0.2 to 0.9 with step of 0.1 and the y-axis denotes the average acceptance ratio. R is randomly selected from a uniform distribution $[1.5, 2.5]$ and pCriticality$= 0.5$. Similarly, 10,000 task sets are generated for each point in the figures. A clear trend can be observed that the acceptance ratio increases as λ increases. This trend confirms the positive impact of increasing λ on the schedulability which we have observed in Section 5.

Next we conduct experiments to evaluate the impact of α on the schedulability. Similarly, we fix U_{avg} and vary α to carry out the experiments. Due to $\alpha = \frac{1}{R}$, if α is given, we compute the corresponding R to generate task sets. The

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2018.2789879, IEEE Transactions on Computers

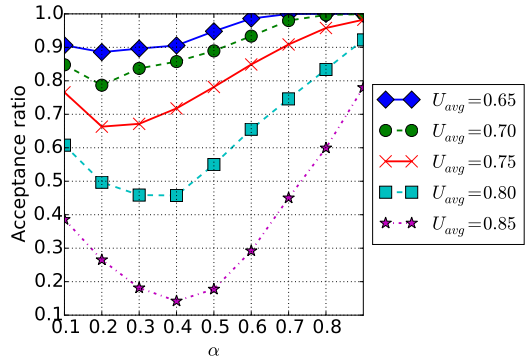


Figure 8: Impact of α

U_{avg}	Acceptance Ratio		Efficiency ratio	
	EYE	DTA	Average	Max
0.6	1	0.999	40	1.8×10^3
0.65	0.998	0.994	88	6.4×10^3
0.7	0.997	0.994	59	6.2×10^3
0.75	0.997	0.984	63	4×10^3
0.8	0.984	0.951	8.8×10^3	4×10^6
0.85	0.968	0.949	4.1×10^3	1.8×10^6
average	0.0127		2.2×10^3	9.6×10^5

Table 3: The effectiveness and efficiency comparison for IMC task sets

results are depicted in Figure 8, where $\lambda = 0.5$. The x-axis denotes the varying α from 0.1 to 0.9 with step of 0.1. while the y-axis denotes the average acceptance ratio. First, from Table 2, we see that with increasing α the speedup factor first increases till a point. This means within this range the scheduling performance of EDF-VD gradually decreases. After that point, the speedup factor decreases which means the scheduling performance of EDF-VD gradually improves. The experimental results confirm what we have observed for α in Section 5. The acceptance ratio gradually decreases till a point and then it increases.

7.3 DTA Evaluation

As analyzed at the end of Section 6, our DTA algorithm has a lower complexity than the deadline tuning algorithm used in [7] [8], denoted as EYE. In this section, we evaluate the effectiveness and efficiency of DTA in comparison with EYE through extensive experiments. The two comparison metrics are the average acceptance ratio and execution efficiency. The experiment is setup as before, where R is randomly selected from a uniform distribution $[1.5, 2.5]$. pCriticality and λ are set to 0.5 and 0.7, respectively, and we vary U_{avg} from 0.6 to 0.85 with step of 0.05. Since EYE is really time consuming, we only generate 1000 task sets for each given U_{avg}. All experiments are performed on an Intel i7 dual-core processor running at 2.7GHz with 4 GB RAM.

The experimental results are summarized in Table 3. The first column is the varying utilizations, while the second and third columns are the acceptance ratios gained by EYE and DTA, respectively. To evaluate the efficiency of our DTA, we use the metric *efficiency ratio* which is computed as follows,

$$\text{efficiency ratio} = \frac{\text{time cost of EYE}}{\text{time cost of DTA}}$$

The four column is the average efficiency ratio of 1000 task sets and the fifth column is the maximum efficiency obtained from 1000 task sets.

U_{avg}	Acceptance Ratio		Efficiency ratio	
	EYE	DTA	Average	Max
0.6	0.995	0.996	74	1.9×10^3
0.65	0.987	0.98	2.1×10^3	7.6×10^5
0.7	0.979	0.968	3.5×10^3	1.4×10^6
0.75	0.923	0.919	2.5×10^4	1.0×10^7
0.8	0.855	0.843	4.7×10^4	3.2×10^6
0.85	0.748	0.72	9.5×10^4	9.7×10^6
average	0.0102		2.9×10^4	4.2×10^6

Table 4: The effectiveness and efficiency comparison for classical MC task sets

From the schedulability perspective, EYE performs slightly better than DTA in terms of the average acceptance ratio. In the worst case, EYE achieves 3% higher acceptance ratio than DTA at $U_{avg} = 0.8$. On the average, we see that our DTA loses only 1.3% acceptance ratio than the EYE. For the execution efficiency, we see that when $U_{avg} \leq 0.75$, the majority of generated tasks are schedulable with a simple deadline tunning procedure, on the average our DTA does not improve the execution efficiency too much but still in the best case DTA improves the efficiency more than 10^3 times. When $U_{avg} \geq 0.8$, since the increasing number of generated task sets goes through a more complex procedure to find a valid deadline setting, even on the average DTA can achieve more 10^3 times execution efficiency than EYE and in the best case DTA can speed up the deadline tuning by 4×10^6 times. Usually, DTA takes only a couple of minutes in the worst case, whereas EYE, in the worst case, takes several hours to tune deadlines. On average, our DTA can speed up the deadline tuning procedure by 2.2×10^3 times. In the best case, DTA can speed up the procedure by 4×10^6 times. As we see, the experimental results show that the proposed DTA considerably improves the efficiency of deadline tunning procedure with a negligible schedulability loss.

We further evaluate the effectiveness and efficiency of DTA on the classical MC task sets. The experiment setup is analogous to the IMC experimental setup and the only modification is to set $\lambda = 0$. The experimental results are summarized in Table 4. The experimental results show a larger improvement in terms of the execution efficiency when the classical MC tasks are considered. Only when $U_{avg} = 0.6$, the average efficiency improvement is 74. For other experimental settings, the average efficiency ratio is more than 10^3. In the best case, the efficiency ratio can achieve 1.0×10^7 at $U_{avg} = 0.75$.

8 Discussion

This section presents some discussion related to the IMC model in terms of the application of the proposed schedulability tests and the implementation of the IMC model under EDF-VD scheduling.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2018.2789879, IEEE Transactions on Computers

8.1 Discussion of The schedulability Tests

Two schedulability tests for the IMC model under EDF-VD are proposed in this paper. The utilization-based test provides a concise and polynomial-time test, but only applicable to the *implicit deadline task model*. On the other hand, the DBF-based test can be applied to a more general task model, *constrained deadline task model*, at the expense of high complexity. Hence, in practical, the two sets can be first selected to use according to the type of task models, i,e, *implicit* or *constrained*. For the *implicit task model*, if the utilization-based test reports unschedulable, the DBF-test and the deadline tunning algorithm can be applied to further check the schedulability. However, for the *constrained deadline task model*, only the DBF-test can be deployed.

In this work, our analysis is mainly based on the model with two criticality levels. In industry standards, it can be up to 5 criticality level, e.g., DO-178B/C for aviation industry. In this case, at different criticality levels, different virtual deadlines might be assigned to tasks according to their inherent criticality level. Then, all tasks start to be scheduled at the lowest criticality level. When any tasks with criticality level higher than the operating criticality level overrun, the system switches to a higher criticality mode similar to what we have in the two-criticality level model. The criticality mode of the system could be switched one by one and may finally reach the highest criticality level. The execution semantics of the multiple criticality levels is still analogous to the one with two criticality levels, and the only difference is that the system may occur several mode switches. But also, some tasks with lower criticality levels might be completely discarded during consecutive switches in order to compensate tasks with higher criticality levels. Baruah *et al.* in [31] have shown that the similarly utilization-based test for the classical MC model can be scaled up to 5 criticality levels at the cost of the scheduling performance, so our utilization-based test should follow the same trend as in [31] when more criticality levels are considered. However, for the DBF-based test, although we have improved the deadline tunning algorithm, the complexity is still pseudo-polynomial time. As Burns *et al.* observed for the classical MC model in [4], it is not clear whether the DBF-based test can be scaled up to more than two criticality levels, due to the high complexity of the deadline tunning algorithm.

8.2 Discussion of The Implementation

Previously, we have theoretically shown some results for the IMC model. In this section, we discuss the IMC mode under EDF-VD from the practical perspective, i.e., the implementation on real systems. To implement the IMC model on a real platform, we have two parts to consider, the implementation of the imprecise feature of *low*-criticality tasks, and the implementation of EDF-VD scheduling.

In [32], two approaches to return the imprecise results were proposed, the milestone approach and the sieve approach. The milestone approach is suitable for the applications which have several distinct stages. At the end of every distinct stage, the currently computed results are recorded. If the system overloads and needs to make the deadline, the latest recorded results are returned as the imprecise results. The sieve approach is applicable to the applications whose results are improved over time. That is, at a certain point the raw results can be generated but the further refinement can be conducted on the initially raw results to obtain the refined results (more precise results). When the applications are terminated to ensure the deadline, the latest refined results are returned as the imprecise results. Both milestone and sieve approaches can be implemented with simple primitive functions, therefore they occur no significant overhead. To guarantee the temporal correctness of the system, the implementation overhead should be added to the WCET estimation of *low*-criticality tasks.

Following, we discuss the implementation solution of EDF-VD scheduling based on the latest version of LITMUS-RT [16]. LITMUS-RT is a real-time extension of Linux kernel and has been widely used in the real-time systems research to evaluate the performance of different scheduling algorithms. The first step to implement EDF-VD needs to extend the parameters of real-time tasks. In LITMUS-RT, a data structure, called rt_task, is given to specify the real-time parameters of a process/thread, such as WCET, period, and deadline. To enable the MC property, more parameters should be given to account for *high*-criticality WCET, virtual deadline, and criticality level. A possible solution is to define a new data structure to include all these MC-related parameters and add this MC-related data structure to rt_task. The following is one possible implementation of the MC-related data structure.

```
struct mc_extension {
    crit_level_t          criticality;
    unsigned long long    deadline_offset;
    unsigned long long    wcet_high; };
```

where crit_level_t is an enum type to specify the criticality level, deadline_offset is the difference of the actual deadline and the virtual deadline of a task.

The implementation of EDF-VD can be done with a slight modification on the EDF implementation. The modification is to implement the mode switch in the MC/IMC model. In LITMUS-RT, every task is assigned an execution budget, and different mechanism can be deployed when the assigned budget is exhausted. In the context of MC systems, the task is assigned an execution budget equivalent to its *low*-criticality WCET, and the *high*-criticality wcet is assigned to the element, wcet_high, in **struct** mc_extension. When a budget overrun is detected for a *low*-criticality task, the *low*-criticality task will be throttled and wait for the next release. On the other hand, if a *high*-criticality task overruns its *low*-criticality budget, instead of throttling the task, we update the parameters like changing the execution budgets for all tasks and deadlines for all *high*-criticality tasks. In LITMUS-RT, the ready queue and the release queue are both implemented based on binomial heap (bheap) which could provide a priority queue according to a certain scheduling algorithm. However, it is very difficult to traverse all nodes in bheap. So a linked list data structure would be preferred to store references of all tasks such that updating tasks' parameters could be achieved by quickly traversing all tasks. Once the parameters are all updated, the build-in function bheap_decrease which increase the priority of a task within

```
int bheap_decrease(
bheap_prio_t edf_ready_order,
struct bheap_node* node)
```

a queue can be used to update the scheduling order of all tasks. where bheap_prio_t is to specify how to order tasks in the heap and bheap_node is a bheap node with one task in it. The complexity of the update parameter should be $O(|\gamma|)$, $|\gamma|$ indicates the number of tasks on the system. For the step of reordering tasks, since deadlines of *high*-criticality tasks will be increased (i.e., priority of *high*-criticality tasks will remain the same or become lower after the mode switches), only calling bheap_decrease function for all *low*-criticality tasks could reorder all released tasks on the system. The complexity of bheap_decrease is $O(\log|\gamma|)$, so the complexity of the reordering step should be $O(|\gamma_{LO}|\log|\gamma|)$, where $|\gamma_{LO}|$ is the number of *low*-criticality tasks. On a 2.7GhZ core, one execution of bheap_decrease just takes 15 μs. We can see that this implementation does not occur significant overhead, and this overhead is only added to the *high*-criticality tasks because the mode switch always happens when one *high*-criticality task overruns.

9 Conclusions

In this paper, we comprehensively study the scheduling problem of *imprecise mixed-criticality* (IMC) model from [9] under EDF-VD scheduling. A utilization-based schedulability test is first proposed. Based on the utilization-based sufficient test, we derive a speedup factor function with respect to the utilization ratio α of all *high*-criticality tasks and the utilization ratio λ of all *low*-criticality tasks. This speedup factor function provides a good insight to observe the impact of α and λ on the speedup factor and thus quantifies the suboptimality of EDF-VD for the IMC model in terms of speedup factor. To further improve the schedulability of the IMC model under EDF-VD and consider more general constrained deadline task model, we propose a complex schedulability test based on the demand bound function and a new algorithm to improve the efficiency of the deadline tuning procedure is also provided. Our experimental results demonstrate the effectiveness of our proposed schedulability tests in terms of average acceptance ratio and show the efficiency of our deadline tunning algorithm in comparison with the existing algorithm. Moreover, the extensive experiments also confirm the theoretical observations we obtained for the speedup factor. From practical perspective, the possible implementation of the IMC model under EDF-VD scheduling is discussed at the end.

References

[1] H. Thompson, "Mixed criticality systems," February 2012.

[2] J. Barhorst, T. Belote, P. Binns, J. Hoffman, J. Paunicka, P. Sarathy, J. S. P. Stanfill, D. Stuart, and R. Urzi., "White paper: A research agenda for mixed-criticality systems," April 2009.

[3] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *Proceedings of the 28th IEEE International Real-Time Systems Symposium (RTSS)*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 239–243.

[4] A. Burns and R. Davis, "Mixed criticality systems-a review," *University of York, Tech. Rep*, 2015.

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

[6] S. K. Baruah, A. Burns, and R. I. Davis, "Response-time analysis for mixed criticality systems," in *Proceedings of the 32nd IEEE Real-Time Systems Symposium*. Washington, DC, USA: IEEE Computer Society, 2011, pp. 34–43.

[7] P. Ekberg and W. Yi, "Bounding and shaping the demand of generalized mixed-criticality sporadic task systems," *Real-time systems*, vol. 50, no. 1, pp. 48–86, 2014.

[8] A. Easwaran, "Demand-based scheduling of mixed-criticality sporadic tasks on one processor," in *Proceedings of the 2013 IEEE 34th Real-Time Systems Symposium (RTSS)*. Washington, DC, USA: IEEE Computer Society, 2013, pp. 78–87.

[9] A. Burns and S. Baruah, "Towards a more practical model for mixed criticality systems," in *Proceedings of Workshop on Mixed Criticality, IEEE Real-Time Systems Symposium (RTSS)*, 2013, pp. 1–6.

[10] P. Huang, G. Giannopoulou, N. Stoimenov, and L. Thiele, "Service adaptions for mixed-criticality systems," in *Proceedings of the 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2014, pp. 125–130.

[11] D. Liu, J. Spasic, N. Guan, G. Chen, S. Liu, T. Stefanov, and W. Yi, "Edf-vd scheduling of mixed-criticality systems with degraded quality guarantees," in *2016 IEEE Real-Time Systems Symposium (RTSS)*, Nov 2016, pp. 35–46.

[12] J. W. Liu, K.-J. Lin, W.-K. Shih, A. C.-s. Yu, J.-Y. Chung, and W. Zhao, "Algorithms for scheduling imprecise computations," *Computer*, vol. 24, no. 5, pp. 58–68, 1991.

[13] J. W. Liu, W.-K. Shih, K.-J. Lin, R. Bettati, and J.-Y. Chung, "Imprecise computations," *Proceedings of the IEEE*, vol. 82, no. 1, pp. 83–94, 1994.

[14] R. Ravindran, C. M. Krishna, I. Koren, and Z. Koren, "Scheduling imprecise task graphs for real-time applications," *International Journal of Embedded Systems (IJES)*, vol. 6, 2014.

[15] S. Baruah, A. Burns, and Z. Guo, "Scheduling mixed-criticality systems to guarantee some service under all non-erroneous behaviors," in *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)*, July 2016, pp. 131–138.

[16] B. B. Brandenburg, "Scheduling and locking in multiprocessor real-time operating systems," Ph.D. dissertation, Chapel Hill, NC, USA, 2011, aAI3502550.

[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] H. Su, N. Guan, and D. Zhu, "Service guarantee exploration for mixed-criticality systems," in *Proceedings of IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2014.

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

[20] J.-Y. Chung, J. W. S. Liu, and K.-J. Lin, "Scheduling periodic jobs that allow imprecise results," *IEEE Trans. Comput.*, vol. 39, no. 9, pp. 1156–1174, Sep. 1990.

[21] J. Lee, K. M. Phan, X. Gu, J. Lee, A. Easwaran, I. Shin, and I. Lee, "Mc-fluid: Fluid model-based mixed-criticality scheduling on multiprocessors," in *2014 IEEE Real-Time Systems Symposium*, Dec 2014, pp. 41–52.

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

[23] J. Ren and L. T. X. Phan, "Mixed-criticality scheduling on multiprocessors using task grouping," in *2015 27th Euromicro Conference on Real-Time Systems*, July 2015, pp. 25–34.

[24] X. Gu and A. Easwaran, "Dynamic budget management with service guarantees for mixed-criticality systems," in *2016 IEEE Real-Time Systems Symposium (RTSS)*, Nov 2016, pp. 47–56.

[25] J. Real and A. Crespo, "Mode change protocols for real-time systems: A survey and a new proposal," *Real-Time Syst.*, vol. 26, no. 2, pp. 161–197, Mar. 2004.

[26] K. W. Tindell, A. Burns, and A. J. Wellings, "Mode changes in priority preemptively scheduled systems," in *[1992] Proceedings Real-Time Systems Symposium*, Dec 1992, pp. 100–109.

[27] F. Santy, L. George, P. Thierry, and J. Goossens, "Relaxing mixed-criticality scheduling strictness for task sets scheduled with fp," in

Proceedings of the 24th Euromicro Conference on Real-Time Systems (ECRTS), July 2012.

[28] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, 1973.

[29] S. K. Baruah, A. K. Mok, and L. E. Rosier, "Preemptively scheduling hard-real-time sporadic tasks on one processor," in *[1990] Proceedings 11th Real-Time Systems Symposium*, Dec 1990, pp. 182–190.

[30] F. Zhang and A. Burns, "Schedulability analysis for real-time systems with edf scheduling," *IEEE Trans. Comput.*, vol. 58, no. 9, pp. 1250–1258, Sep. 2009.

[31] 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.

[32] K.-J. Lin, S. Natarajan, and J. W.-S. Liu, "Imprecise results: Utilizing partial computations in real-time systems," 1987.