On Bounding Execution Demand under Mixed-Criticality EDF

Mitra Mahdiani and Alejandro Masrur
Department of Computer Science, TU Chemnitz, Germany
mitra.mahdiani@informatik.tu-chemnitz.de
a.masrur@cs.tu-chemnitz.de

ABSTRACT

This paper is concerned with mixed-criticality systems where a set of low-criticality (LO) and high-criticality (HI) tasks share one processor and are scheduled under the EDF algorithm. Basically, the system operates in two modes: LO and HI mode. In LO mode, one HI task may exceed its execution budget, which then causes a change to HI mode in the system. HI tasks are assigned larger execution budgets in HI mode at the cost of the LO tasks — which are assumed to be discarded. Since these mode changes may happen at arbitrary points in time, it is difficult to find an accurate bound on the amount of *carry-over* execution demand. That is the execution demand by HI jobs that were released before, but did not finish executing at the point of changing to HI mode. As a consequence, the resulting characterization of the overall execution demand becomes pessimistic. In this paper, to overcome this problem, a technique is proposed that works around the computation of carry-over execution demand and results in a more accurate bound on execution demand under mixed-criticality EDF. In principle, the proposed technique consists in separating the schedulability analysis of *stable* HI mode from that of the *transition* between modes and deriving a separate demand bound function for the latter case. The proposed technique results not only in a considerably simpler, but also tighter bound on execution demand under mixed-criticality EDF, in particular, as the number of HI tasks increases. We illustrate this analytically and by a large set of experiments based on synthetic data.

CCS CONCEPTS

• **Computer systems organization** → *Real-time systems*; *Real-time operating systems*; *Real-time system specification*;

KEYWORDS

Real-time scheduling, admission control, mixed criticality, EDF-VD, resource efficiency

ACM Reference Format:
Mitra Mahdiani and Alejandro Masrur. 2018. On Bounding Execution Demand under Mixed-Criticality EDF. In *26th International Conference on Real-Time Networks and Systems (RTNS '18), October 10–12, 2018, Chasseneuil-du-Poitou, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3273905.3273930

RTNS '18, October 10–12, 2018, Chasseneuil-du-Poitou, France

ACM ISBN 978-1-4503-6463-8/18/10…$15.00
https://doi.org/10.1145/3273905.3273930

1 INTRODUCTION

There is a trend towards consolidating software functions onto fewer processors in different domains such as automotive systems and avionics. Hence applications with different levels of criticality that used to run in isolation now start sharing processors. As a result, there is a need for techniques that allow designing such mixed-criticality (MC) systems and, at the same time, complying with certification requirements in the different domains.

In this paper, we study the problem of scheduling MC systems under the earliest deadline first (EDF) algorithm. In particular, we consider that a mix of low-criticality (LO) and high-criticality (HI) tasks are scheduled on one processor. While LO tasks can be modeled by minimum inter-arrival time, deadline, and worst-case execution time (WCET), HI tasks are characterized by two WCET parameters: an optimistic and a conservative one.

The system then distinguishes between two operation modes: HI and LO mode. In LO mode, HI tasks require executing for no longer than their optimistic WCETs and are scheduled together with the LO tasks. A switch to HI mode occurs when one HI task executes for longer than its optimistic WCET (but still less than its conservative one). Similar to other approaches from the literature, we consider that LO tasks are immediately discarded in HI mode, which then allows accommodating this increase in HI execution demand.

According to EDF, HI tasks can be scheduled arbitrarily close to their deadlines in LO mode. As a result, they may miss their deadlines after switching to HI mode in spite of discarding LO tasks. To overcome this problem, a common approach is to assign HI tasks *virtual deadlines* $x_i \cdot D_i$ that are shorter than their real deadlines D_i, where x_i is in $(0, 1]$. This way, HI tasks are scheduled within their virtual deadlines in LO mode such that they can always meet their *real deadlines* after switching to HI mode.

The problem of testing schedulability in this setting reduces to finding valid x_i for each of the HI tasks in the system. So far, there have been different approaches to this, which are usually based on approximating the execution demand by MC task sets [1][2][3]. However, since a switch from LO to HI mode may occur at an arbitrary point in time, it remains difficult to accurately bound the execution demand by so-called *carry-over jobs*, i.e., HI jobs that have been released before but have not finished executing at the point in time of switching. As a result, known demand bound for mixed-criticality EDF are pessimistic.

Contributions. In this paper we address the above problem and propose a technique that allows better bounding the execution demand under mixed-criticality EDF. The proposed technique is based on separating the schedulability analysis of transitions to HI mode from that of stable HI mode. If the system switches to HI mode at an arbitrary t', the transition to HI mode happens in

the interval $[t', t'']$, where t'' is the point in time at which the processor first idles after t'. In contrast, stable HI mode starts from t'' onwards. Our contributions can be summarized as follows:

- Based on the above idea, we derive a separate demand bound function for transitions from LO to HI mode and prove its validity. This technique allows us to work around the computation of carry-over execution demand and, hence, to reduce the amount of pessimism in characterizing mixed-criticality EDF.
- We further show that the proposed technique strictly dominates the one by Ekberg and Yi [2], i.e., it leads to a tighter bound on the execution demand under mixed-criticality EDF. In addition, this is provably tighter than that of Easwaran [3] for most cases.
- Based on the proposed technique, we derive a schedulability test for mixed-criticality systems based on EDF. The resulting schedulability test is considerably simpler than those by Ekberg and Yi [2] and Easwaran [3]. This makes it particularly interesting for *design space exploration*, where often a large set of different configurations needs to be tested.
- We present evaluation results by a large set of experiments based on synthetic data illustrating the benefits of the proposed technique.

Structure of the paper. The rest of this paper is structured as follows. Related work is discussed in Section 2. Section 3 introduces the task model and assumptions used, whereas the principles of the proposed technique are explained in Section 4. In Section 5, we perform an analytical comparison of the proposed technique with those of [2] and [3]. Our schedulability test for mixed-criticality EDF is discussed in Section 6. In Section 7 we present experimental results, whereas Section 8 concludes the paper.

2 RELATED WORK

MC scheduling was first proposed under this name by Vestal [4]. Baruah *et al.* later analyzed per-task priority assignments and the resulting worst-case response times [5]. In [6], Baruah *et al.* proposed the EDF-VD algorithm to schedule a mix of HI and LO tasks. EDF-VD introduces two operation modes and uses a *priority-promotion* scheme by uniformly scaling deadlines of HI tasks.

A speed-up factor for EDF-VD was first obtained in [6] as $(\sqrt{5}+1)/2$. Later this speed-up factor was improved to $4/3$ [1]. Baruah *et al.* further proposed extensions to EDF-VD, where, in particular, a per-task deadline scaling is used [7]. However, they also concluded that the speed-up factor of $4/3$ cannot be improved [7].

A more flexible approach with per-task deadline scaling was presented by Ekberg and Yi [2][8]. Ekberg and Yi characterized the execution demand of MC systems under EDF by deriving demand bound functions for the LO and the HI mode. In [3], Easwaran presented a similar technique and showed that it strictly dominates that of Ekberg and Yi.

This paper follows this line of work. In particular, we propose separating the analysis of transitions from stable HI mode. Based on this, we derive the corresponding demand bound functions, which are shown to be more accurate than the ones by Ekberg and Yi and of Easwaran.

Recently, Huang *et al.* proposed speeding up processors to account for increases in HI execution demand when switching to HI mode [9]. Huang *et al.* made use of Ekberg and Yi's demand bound functions to compute the necessary speed-up factors that guarantee meeting all deadlines. It should be noted that the proposed technique of this paper can also be combined with that of [9] to compute more accurate speed-up factors.

Improvements to the original EDF-VD have also been proposed by other authors. In [10], Su and Zhu used an *elastic* task model [11] to improve resource utilization in MC systems. In [12], Zhao *et al.* applied *preemption thresholds* [13] in MC scheduling in order to better utilize the processing unit. In [14], a technique consisting of two scaling factors is proposed for a admission control in MC systems.

For multiprocessors, a partitioned and a global scheduling approach both based on EDF-VD were proposed in [15]. According to this work, partitioned behaves better than global scheduling in the context of MC systems.

Further, in [16], Pathan studies global MC scheduling with task-level fixed priorities and gives a schedulability test based on response time analysis for more than two criticality levels. Finally, a comprehensive overview of mixed-criticality systems is further given in [17].

3 MODELS AND ASSUMPTIONS

We basically adopt the task model originally used in [2]. We denote by τ the set of n independent sporadic tasks τ_i that run on one processor under preemptive EDF scheduling.

The minimum separation between any two jobs or instances of a τ_i is denoted by T_i. We assume *constrained* deadlines, i.e., $\forall i : D_i \leq T_i$ where D_i denotes a task's relative deadline. There is no self-suspension, and context-switch overheads are assumed to be negligible.

As already stated, we are concerned with dual-criticality systems with two levels of criticality, namely LO and HI. The *criticality* of a task τ_i is denoted by χ_i with:

$$\chi_i \in \{LO, HI\}.$$

A LO task is associated with only its WCET C_i^{LO}. Opposed to this, a HI task is characterized by its optimistic WCET estimate C_i^{LO} and its conservative WCET estimate C_i^{HI} with:

$$C_i^{LO} \leq C_i^{HI} \leq D_i \leq T_i.$$

Basically, the system operates in two modes denoted by m: LO and HI mode. In LO mode, HI tasks execute for no longer than C_i^{LO}, whereas these might require executing for up to C_i^{HI} in HI mode. The system initializes in LO mode where all LO and HI tasks need to meet their deadlines. As soon as one job of a HI task executes for longer than its C_i^{LO}, the system switches to HI mode where only the HI tasks are allowed to run – LO tasks are immediately discarded.

We denote the *utilization* by LO and HI tasks in the LO and HI mode respectively as follows:

$$U_\chi^m := \sum_{\chi_i=\chi} \frac{C_i^m}{T_i},$$

where again χ and m can assume values in $\{LO, HI\}$. Note that only U_{LO}^{LO}, U_{HI}^{LO} and U_{HI}^{HI} exist. U_{LO}^{HI} is effectively zero, since LO tasks are dropped and, hence, do not run in HI mode.

Finally, in contrast to [2], we are not constrained to integer numbers, but rather use real numbers for all above parameters which gives us more flexibility in modeling MC workloads.

Mixed-Criticality EDF. A common approach when scheduling MC systems is to shorten the deadlines of HI jobs in LO mode. This way, processor capacity can be *reserved* for a potential switch to HI mode — where HI tasks require more execution demand. In other words, we assign a *virtual deadline* equal to $x_i \cdot D_i$ with $x_i \in (0, 1]$ to all τ_i where $\chi_i = HI$.

This virtual deadline is used instead of D_i — the *real* deadline — to schedule HI tasks in LO mode. The parameter x_i is the so-called *deadline scaling factor*. There is no deadline scaling for LO tasks such that they are scheduled (only in LO mode) using their D_i.

When the system switches to HI mode, HI tasks start being scheduled according to their real deadlines D_i whereas LO tasks are discarded immediately. In this paper, we consider that tasks are scheduled under EDF in both modes and refer to this scheme as *mixed-criticality* EDF.

Clearly, whereas schedulability of separated modes can be easily tested, i.e., when the system is stable in either LO or HI mode, it is difficult to test schedulability of transitions between modes. In particular, careful analysis is required when the system switches from LO to HI mode.

In this paper, similar to other approaches from the literature, transitions from HI back to LO mode are disregarded. The reason is that, in contrast to changes from LO to HI, a change from HI to LO mode can be programmed or postponed to a suitable point in time, e.g., at which the processor idles after all HI tasks have run again for their optimistic WCETs, and does not require further analysis.

4 BOUNDING EXECUTION DEMAND

In this section, we introduce the proposed technique. Basically, similar to [2] and [3], we characterize the execution demand of τ by applying the concept of *demand bound function* [18]. In [2] and [3], a demand bound function was derived for each mode in the system, i.e., one for the LO mode and one for the HI mode.

In contrast to this, as mentioned above, we derive a third demand bound function for the transition between modes. This allows working around the computation of carry-over execution demand, reducing the amount of pessimism and, hence, relaxing schedulability conditions in HI mode as we illustrate later.

Schedulability in LO mode. In LO mode, LO tasks need to scheduled together with HI tasks, while the latter are assigned virtual deadlines $x_i \cdot D_i$. As a consequence, the demand bound function $\mathrm{dbf}_{LO}(t)$ in LO mode is given by:

$$\mathrm{dbf}_{LO}(t) = \sum_{\chi_i=LO} \left(\left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1 \right) C_i^{LO} + \sum_{\chi_i=HI} \left(\left\lfloor \frac{t - x_i \cdot D_i}{T_i} \right\rfloor + 1 \right) C_i^{LO}. \quad (1)$$

Here $t \geq 0$ is a real number representing time, i.e., $\mathrm{dbf}_{LO}(t)$ returns a τ's maximum execution demand in LO mode in an interval of length t. Note that $\mathrm{dbf}_{LO}(t)$ is always greater than or equal to zero, since $D_i \leq T_i$ holds for all τ_i and x_i has values in $(0, 1]$.

The system is schedulable in LO mode, if $\mathrm{dbf}_{LO}(t) \leq t$ holds for all possible t until the processor first idles [18], i.e., until a point in time $\hat{t}_{LO}$ by which $\mathrm{dbf}_{LO}(\hat{t}_{LO}) = \hat{t}_{LO}$ holds. Following the technique in [18], we can remove the floor function in (1) to obtain an upper bound on $\hat{t}_{LO}$:

$$\hat{t}_{LO} \leq \frac{\sum_{\chi_i=LO} (T_i - D_i) \frac{C_i^{LO}}{T_i}}{1 - U_{LO}^{LO} - U_{HI}^{LO}} + \frac{\sum_{\chi_i=HI} (T_i - x_i \cdot D_i) \frac{C_i^{LO}}{T_i}}{1 - U_{LO}^{LO} - U_{HI}^{LO}}. \quad (2)$$

The bound in (2) depends on the values of x_i, which need to be computed and are not known a priori. On the other hand, to resolve this dependency, note that this bound maximizes for all $x_i = 0$ which then leads to the following:

$$\hat{t}_{LO} \leq \frac{\sum_{\chi_i=LO} (T_i - D_i) \frac{C_i^{LO}}{T_i}}{1 - U_{LO}^{LO} - U_{HI}^{LO}} + \frac{\sum_{\chi_i=HI} C_i^{LO}}{1 - U_{LO}^{LO} - U_{HI}^{LO}}, \quad (3)$$

Clearly, $U_{LO}^{LO} + U_{HI}^{LO}$ — the utilization in LO mode — must be strictly less than one in order that (2) and (3) return valid and finite values.

Schedulability in HI mode. In HI mode, again, LO tasks do not run and HI tasks run for their corresponding C_i^{HI} leading to the following demand bound function:

$$\mathrm{dbf}_{HI}(t) = \sum_{\chi_i=HI} \left(\left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1 \right) C_i^{HI}, \quad (4)$$

where again $t \geq 0$ is a real number representing time, i.e., $\mathrm{dbf}_{HI}(t)$ returns the maximum execution demand in a time interval of length t.

The system is schedulable in *stable* HI mode, if $\mathrm{dbf}_{HI}(t) \leq t$ for all possible t until the processor first idles, i.e., until a point in time $\hat{t}_{HI}$ is reached where $\mathrm{dbf}_{HI}(\hat{t}_{HI}) = \hat{t}_{HI}$. We again can remove the floor function in (4) to obtain an upper bound on $\hat{t}_{HI}$:

$$\hat{t}_{HI} \leq \frac{\sum_{\chi_i=HI} (T_i - D_i) \frac{C_i^{HI}}{T_i}}{1 - U_{HI}^{HI}}. \quad (5)$$

U_{HI}^{HI} — the utilization in HI mode — must be strictly less than one in order that (5) returns a valid and finite upper bound on $\hat{t}_{HI}$.

Schedulability in the transition from LO to HI mode. The transition from LO to HI mode may happen at an arbitrary point in time when one HI job exceeds its LO execution budget C_i^{LO}.

Unfinished LO jobs are discarded at that time; however, the problem arises with HI jobs that are released but have not finished executing their C_i^{LO}, i.e., carry-over jobs. Since it is difficult to accurately bound the execution demand by carry-over jobs, usually, pessimistic assumptions need to taken.

The following theorem is a generalization of a theorem in [14] and allows us to work around carry-over jobs constituting the main contribution by this paper. In other words, this theorem allows us to guarantee schedulability without computing carry-over execution demand at the point of switching from LO to HI mode, which considerably reduces the amount of pessimism.

Theorem 1. *Given a set τ of MC tasks, let us assume that the following two conditions hold: (i) $dbf_{LO}(t) \leq t$ and (ii) $dbf_{HI}(t) \leq t$ hold for $0 < t \leq \hat{t}_{LO}$ and $0 < t \leq \hat{t}_{HI}$ respectively, i.e., τ is schedulable in LO and stable HI mode. The transition from LO to HI mode is schedulable under mixed-criticality EDF, if $dbf_{SW}(t) \leq t$ also holds for $0 < t \leq \hat{t}_{SW}$, where $dbf_{SW}(t)$ is given by:*

$$dbf_{SW}(t) = \sum_{\chi_i=HI} \left(\left\lfloor \frac{t - \Delta D_i}{T_i} \right\rfloor + 1 \right) \Delta C_i, \tag{6}$$

with $\Delta D_i = D_i - x_i \cdot D_i$, $\Delta C_i = C_i^{HI} - C_i^{LO}$, and $\hat{t}_{SW}$ is upper bounded by the following expression:

$$\hat{t}_{SW} \leq \frac{\sum_{\chi_i=HI} \Delta C_i}{1 - U_{HI}^{SW}}, \tag{7}$$

where U_{HI}^{SW} is given by $\sum_{\chi_i=HI} \frac{\Delta C_i}{T_i}$.

Proof. Let us consider that the system *switches* to HI mode at time t' and that the processor idles for the first time thereafter at t'' with $t' < t''$, i.e., all jobs released prior to t'' finish executing at latest by t''. Clearly, jobs that are released after t'' are guaranteed schedulable by assumption (ii).

Let us now assume that a deadline is missed for the first time at t_{miss} by a job of any τ_i that we denote τ_{miss}. Clearly, t_{miss} must be in the interval $[t', t'']$ and the following must hold for $\delta_{miss} = t_{miss} - t'$:

$$\delta_{miss} < CO + \sum_{\chi_i=HI} \left(\left\lfloor \frac{\delta_{miss} - \phi_i - D_i}{T_i} \right\rfloor + 1 \right) C_i^{HI},$$

where $\phi_i = r_i' - t'$ is the *phase* of a τ_i at t', i.e., the release time r_i' of its first job after t' minus t'. Note that ϕ_i is in the interval $[0, T_i)$. In addition, CO denotes the *carry-over* execution demand at t'. This is the amount of execution in $[t', t_{miss}]$ by HI jobs that are released prior to t', but have not finished executing at t'.

As already discussed, it is difficult to determine CO in an accurate manner. Hence, to work around CO, let us first divide each τ_i, whose jobs have both release times and deadlines in $[t', t_{miss}]$, into two subtasks. The first subtask — denoted by τ_i^{LO} — is released for the first time at ϕ_i and requires executing C_i^{LO} within $x_i \cdot D_i$ every T_i time, i.e., this represents τ_i's execution demand in LO mode. The second subtask — denoted by τ_i^{SW} — is released for the first time at $\phi_i' = \phi_i + x_i \cdot D_i$ and requires executing $\Delta C_i = C_i^{HI} - C_i^{LO}$ within $\Delta D_i = D_i - x_i \cdot D_i$ every T_i time, i.e., this presents τ_i's increase in execution demand incurred in HI mode. Note that, in spite of this modification, the total amount of execution demand in $[t', t_{miss}]$ does not change, i.e., a deadline is still missed at t_{miss} as per assumption, and we can reshape the above inequality to:

$$\delta_{miss} < CO + \sum_{\chi_i=HI} \left(\left\lfloor \frac{\delta_{miss} - \phi_i - x_i \cdot D_i}{T_i} \right\rfloor + 1 \right) C_i^{LO} + \sum_{\chi_i=HI} \left(\left\lfloor \frac{\delta_{miss} - \phi_i' - \Delta D_i}{T_i} \right\rfloor + 1 \right) \Delta C_i. \tag{8}$$

Note that t_{miss} coincides with the deadline of the corresponding job of τ_{miss}^{SW}, which misses its deadline. (Recall that τ_{miss} is now divided into the subtasks τ_{miss}^{LO} and τ_{miss}^{SW}.) Now, there are two possible cases to consider in order to prove this theorem: The set of *only* subtasks τ_i^{SW} is either (1) unschedulable or (2) schedulable in isolation.

Case (1): This is a rather trivial case. If the set of only τ_i^{SW} is unschedulable in isolation, i.e., when scheduled alone on a single processor, $\text{dbf}_{SW}(t) > t$ must hold for some t in $[0, \hat{t}_{SW}]$ with $\text{dbf}_{SW}(t)$ given as per (6). As a result, we will be able to detect a deadline miss in the transition between LO and HI mode by only testing the set of all τ_i^{SW}.

To this end, we need to find an upper bound on $\hat{t}_{SW}$ making $\text{dbf}_{SW}(\hat{t}_{SW}) = \hat{t}_{SW}$ and removing the floor function as before:

$$\hat{t}_{SW} \leq \frac{\sum_{\chi_i=HI} (T_i - \Delta D_i) \frac{\Delta C_i}{T_i}}{1 - U_{HI}^{SW}}.$$

Here, $U_{HI}^{SW} = \sum_{\chi_i=HI} \frac{\Delta C_i}{T_i}$ is the utilization of the set of only τ_i^{SW}. Since $U_{HI}^{SW} < 1$ holds by assumption (ii), the above inequality returns a valid bound on $\hat{t}_{SW}$. This depends on $\Delta D_i = D_i - x_i \cdot D_i$ and, therefore, on the values of x_i, which we do not know in advance. However, we can make $x_i = 1$ for all i leading to the upper bound in (7). The theorem follows.

Case (2): If $\text{dbf}_{SW}(t) \leq t$ holds for all t in $[0, \hat{t}_{SW}]$, we show that assuming that a deadline is missed at t_{miss} leads to a contradiction.

We have assumed that a deadline is missed for the first time at t_{miss}, hence, all previous jobs in $[t', t_{miss})$ can actually finish executing in time. Since now τ_{miss} is divided into the subtasks τ_{miss}^{LO} and τ_{miss}^{SW}, the τ_{miss}^{LO}'s job with a deadline equal to $t_{miss} - \Delta D_{miss}$ must push *carry-over* τ_i^{SW}'s jobs (i.e., those that are released prior to and have not finished executing at $t_{miss} - \Delta D_{miss}$ and that have deadlines prior to t_{miss}) by at least Δ_{miss} (being Δ_{miss} the amount of the deadline miss at t_{miss}). Otherwise, no deadline can be missed at t_{miss}, since again $\text{dbf}_{SW}(t) \leq t$ is assumed to hold for all t. In addition, note that the processor does not idle in $[t_{miss} - \Delta D_{miss}, t_{miss}]$.

Case (2.a): Let us assume that there is only one carry-over job of an arbitrary τ_i^{SW} and that $\Delta D_i \leq \Delta D_{miss}$ holds. Note that, after moving this job forward to force its release time to coincide at $t_{miss} - \Delta D_{miss}$, τ_{miss}^{SW}'s job continues to miss its deadline by Δ_{miss} at t_{miss}, since the deadline of this carry-over τ_i^{SW}'s job remains within the interval $[t_{miss} - \Delta D_{miss}, t_{miss}]$. As a result, the amount of execution demand in $[t_{miss} - \Delta D_{miss}, t_{miss}]$ does not change after this displacement.

The above analysis leads to a contradiction, since τ_{miss}^{SW}'s job and its carry-over τ_i^{SW}'s job are now released in synchrony at $t_{miss} - \Delta D_{miss}$. Consequently, τ_{miss}^{LO} cannot push any additional execution demand into $[t_{miss} - \Delta D_{miss}, t_{miss}]$ and, hence, if a deadline is missed at t_{miss}, $\text{dbf}_{SW}(t) \leq t$ cannot hold for all t.

Case (2.b): Let us now assume that there is again only one carry-over job of an arbitrary τ_i^{SW}, however, $\Delta D_i > \Delta D_{miss}$ holds this time. Note that we can displace this carry-over τ_i^{SW}'s job forward until its deadline occurs at $t_{miss} + \varepsilon$, where ε is an infinitesimally small number greater than zero. As a result, this τ_i^{SW}'s job starts missing its deadline by an amount equal to $\Delta_{miss} - \varepsilon$, since now the original execution demand in $[t_{miss} - \Delta D_{miss}, t_{miss}]$ starts being executed in $[t_{miss} - \Delta D_{miss}, t_{miss} + \varepsilon]$.

It is easy to see that we can now apply the analysis of Case (2.a) where the carry-over τ_i^{SW}'s job of this case misses its deadline at $t_{miss} + \varepsilon$ by an amount equal to $\Delta_{miss} - \varepsilon$ and the τ_{miss}^{SW}'s job of this case becomes the carry-over job in Case (2.a). As a result, Case (2.b) also leads to a contradiction, i.e., if a deadline is missed at t_{miss}, $\text{dbf}_{SW}(t) \leq t$ cannot hold for all t.

Clearly, there can be several carry-over jobs whose execution demands are pushed (at least partially) by τ_{miss}^{LO} into $[t_{miss} - \Delta D_{miss}, t_{miss}]$, however, the total amount of execution demand pushed by τ_{miss}^{LO} remains Δ_{miss}. In this latter case, again, it is easy to see that we can apply the analysis of Case (2.a) and Case (2.b) to each individual carry-over job. Consequently, if a deadline is missed at t_{miss}, $\text{dbf}_{SW}(t) > t$ must hold for some t and the theorem follows. □

Theorem 1 allows characterizing the *additional* execution demand in the transitions from LO to HI mode in a more accurate manner. Based on it, we test whether deadlines are met or not in $[t', t'']$, i.e., from the time t' of switching to HI mode to the time t'' at which the processor first idles after switching. Next we perform an analytical comparison with the known approaches from the literature.

5 ANALYTICAL COMPARISON

In this section, we compare the proposed demand bound functions for mixed-criticality EDF with those used by Ekberg and Yi in the GREEDY algorithm [2] and by Easwaran in the ECDF algorithm [3]. We show, for most cases, that the proposed ones result in tighter bounds on the execution demand than the other mentioned approaches.

5.1 The GREEDY algorithm

In LO mode, note that $\text{dbf}_{LO}(t)$ in (1) is identical to that of Ekberg and Yi — denoted by $\text{gdbf}_{LO}(t)$ in this paper. That is, $\text{dbf}_{LO}(t) = \text{gdbf}_{LO}(t)$ for all $0 < t \leq \hat{t}_{LO}$.

In HI mode, Ekberg and Yi proposed a demand bound function — denoted $\text{gdbf}_{HI}(t)$ in this paper — which is given by the following expression [2]:

$$\text{gdbf}_{HI}(t) = \sum_{\chi_i = HI} \left(\left\lfloor \frac{t - \Delta D_i}{T_i} \right\rfloor + 1 \right) C_i^{HI} - \sum_{\chi_i = HI} \text{done}_i(t), \quad (9)$$

with $\Delta D_i = D_i - x_i \cdot D_i$ and $\text{done}_i(t)$ is given by:

$$\text{done}_i(t) = \begin{cases} \max\left(0, C_i^{LO} - \text{mod}\left(\frac{t}{T_i}\right) + \Delta D_i\right), \\ \qquad \text{if } D_i > \text{mod}\left(\frac{t}{T_i}\right) \geq \Delta D_i, \\ 0, \quad \text{otherwise.} \end{cases}$$

Note that $\text{gdbf}_{HI}(t)$ bounds the execution demand in HI mode taking transitions into account. In our case, as discussed above, we derive different bounds on the execution demand at transitions and in stable HI mode, viz., $\text{dbf}_{SW}(t)$ and $\text{dbf}_{HI}(t)$ respectively.

Now, since $\text{done}_i(t) \leq C_i^{LO}$ holds for all valid values of t and x_i, $\text{dbf}_{SW}(t) \leq \text{gdbf}_{HI}(t)$ also holds for all t. In other words, if the transition to HI mode is safe by $\text{gdbf}_{HI}(t)$, it will also be safe by the proposed $\text{dbf}_{SW}(t)$. However, this does not hold the other way around, i.e., $\text{dbf}_{SW}(t)$ results in a tighter bound on the execution demand at transitions from LO to HI mode than $\text{gdbf}_{HI}(t)$.

On the other hand, if transitions are safe, it is guaranteed that no deadlines are missed after switching to HI mode at a t' and until the processor first idles at a t''. From t'' onwards, it is easy to see that our proposed $\text{dbf}_{HI}(t)$ is sufficient and necessary. That is, if $\text{dbf}_{HI}(t) \leq t$ does not hold for some t with $0 \leq t \leq \hat{t}_{HI}$, then the system is not feasible, i.e., it will be neither be feasible by $\text{gdbf}_{HI}(t)$.

5.2 The ECDF algorithm

Similar to the case of the GREEDY algorithm, note that $\text{dbf}_{LO}(t)$ in (1) is identical to that used in the ECDF algorithm in LO mode. We denote this latter by $\text{edbf}_{LO}(t)$ in this paper. That is, $\text{dbf}_{LO}(t) = \text{edbf}_{LO}(t)$ for all $0 < t \leq \hat{t}_{LO}$.

In HI mode, the ECDF algorithm uses a demand bound function — denoted $\text{edbf}_{HI}(t)$ in this paper — which is given by the following expression [3]:

$$\begin{aligned} \text{edbf}_{HI}(t_1, t_2) = \; & \min\left(t_1, \sum_{j=1}^{3} \text{dbf}_{L_j}(t_1, t_2)\right) \\ & + \sum_{\substack{\chi_i = HI \\ \text{and case 2 or 3}}} \text{dbf}_i^{HI}(t_1, t_2) \\ & + \sum_{\substack{\chi_i = HI \\ \text{and case 2}}} (CO(t_2 - t_1) + \Delta C_i), \end{aligned} \quad (10)$$

where ΔC_i is defined as $C_i^{HI} - C_i^{LO}$. In addition, $0 \leq t_2 \leq \hat{t}_{HI}$ and $0 \leq t_1 \leq t_2 - \min_{\chi_i = HI}(\Delta D_i)$ hold with again $\Delta D_i = D_i - x_i \cdot D_i$.

Here t_1 represents the point in time at which the system switches to HI mode (i.e., $t_1 = t'$ in this paper's notation) and t_2 is the point in time at which a deadline is potentially missed (i.e., $t_2 = t_{miss} \leq t''$ in this paper). Note that $\text{dbf}_i^{HI}(\cdot)$ is a τ_i's contribution to $\text{dbf}_{HI}(\cdot)$ shown in (4). HI tasks in (10) are classified into three cases: case 1 which plays a role in computing $\text{dbf}_{L_j}(\cdot)$, case 2, and case 3. For details on how to compute $\text{dbf}_{L_j}(\cdot)$ for $1 \leq j \leq 3$ and how to compute $CO(\cdot)$ we refer to [3].

The system is schedulable in HI mode, if $\text{edbf}_{HI}(t_1, t_2) \leq t_2$ holds. (Here again no distinction is made between transition and stable HI mode.) Let us now consider that t_1 is less than or equal to

Algorithm 1 Schedulability test for mixed-criticality EDF

Require: τ

Require: τ_{HI} /* subset of HI tasks */

1: X_{LW}=testLO(τ)
2: **if** testHI(τ_{HI})='Passed' **and** $\mathrm{X}_{LW} \neq \emptyset$ **then**
3: X_{UP}=testSW(τ_{HI})
4: **if** $\mathrm{X}_{UP} \neq \emptyset$ **and** $\mathrm{X}_{LW} \leq \mathbf{1} - \mathrm{X}_{UP}$ **then**
5: Return ('Passed')
6: **else**
7: Return ('Not passed')
8: **end if**
9: **end if**

$\sum_{j=1}^{3} \mathrm{dbf}_{L_j}(t_1, t_2)$, such that the schedulability condition by ECDF now becomes:

$$\sum_{\substack{\chi_i=HI \\ \text{and case 2 or 3}}} \mathrm{dbf}_i^{HI}(t_1, t_2) + \sum_{\substack{\chi_i=HI \\ \text{and case 2}}} (CO(t_2 - t_1) + \Delta C_i) \leq t_2 - t_1. \quad (11)$$

Easwaran proved that the left-hand side of the above condition is equal to $\mathrm{gdbf}_{HI}(t_2 - t_1)$ [3]. As a consequence, the proposed $\mathrm{dbf}_{SW}(t)$ results in a tighter bound than (11), since $\mathrm{dbf}_{SW}(t) \leq \mathrm{gdbf}_{HI}(t)$ holds for all t — see again the above Section 5.1.

In the case where t_1 is greater than $\sum_{j=1}^{3} \mathrm{dbf}_{L_j}(t_1, t_2)$, the analytical comparison between $\mathrm{edbf}_{HI}(\cdot)$ and $\mathrm{dbf}_{SW}(\cdot)$ becomes difficult. This is the case where some amount of the execution demand given by $\mathrm{edbf}_{HI}(\cdot)$ starts being executed before t_1, at $t_1 - \sum_{j=1}^{3} \mathrm{dbf}_{L_j}(t_1, t_2)$ to be more precise. Whether a proof of dominance exists (in either way) remains an open problem.

At least, from the above discussion, we can assert that the proposed $\mathrm{dbf}_{SW}(\cdot)$ is tighter for the more stringent case, i.e., when none of the execution demand by $\mathrm{edbf}_{HI}(\cdot)$ can be executed before t_1. Our experiments, based on a large number of synthetic task sets, present evidence that this also holds on average, i.e., $\mathrm{dbf}_{SW}(\cdot)$ usually results in tighter bounds, particularly, when the number of HI task increases.

6 FINDING VALID x_i

In this section, we propose an algorithm to find valid values of x_i for each HI task in τ. Clearly, this is closely related to the technique used to *tighten* deadlines in LO mode. In this paper, we do not aim to improve deadline tightening. The contribution is rather a new technique for bounding demand execution, which can be combined with existing deadline tightening techniques, e.g., from [2] or [3].

The proposed algorithm shown in Alg. 1 essentially tests τ's schedulability in the LO mode (line 1), and in HI mode (line 2). If τ is schedulable in LO mode, the function testLO() returns a vector

Algorithm 2 Function testLO()

Require: τ

1: Compute $\hat{t}_{LO}$ by (3)
2: $\mathrm{X}_{LW} = \mathbf{1}$
3: **while** $t \leq \hat{t}_{LO}$ **do**
4: **if** $\mathrm{dbf}_{LO}(t) > t$ **then**
5: **if** χ_i=LO **or** $\mathrm{dbf}_{LO}(t) - r_i > D_i$ **then**
6: $\mathrm{X}_{LW} = \emptyset$
7: Return
8: **end if**
9: **end if**
10: **if** χ_i=HI **then**
11: **if** Computed(i)='false' **or** $\mathrm{X}_{LW}(i) < \frac{\mathrm{dbf}_{LO}(t)-r_i}{D_i}$ **then**
12: $\mathrm{X}_{LW}(i) = \frac{\mathrm{dbf}_{LO}(t)-r_i}{D_i}$ /* r_i = job's release time */
13: **end if**
14: **end if**
15: (t, i)=getNextDeadline()
16: **end while**
17: Return

X_{LW} with the minimum values of x_i that could be found to be valid. If this vector is not empty, i.e., valid x_i values could be found, and τ is schedulable in HI mode, Alg. 1 tests schedulability at the transitions from LO to HI mode (line 3).

Further, if the set of HI tasks in τ — denoted by τ_{HI} — is schedulable at transitions from LO to HI, the function testSW() returns a vector X_{UP} with the minimum values of $1 - x_i$ that are also valid. That is, if X_{UP} is neither empty, the whole τ will be schedulable under mixed-criticality EDF provided that $\mathrm{X}_{LW} \leq \mathbf{1} - \mathrm{X}_{UP}$ holds (line 4). Here, $\mathbf{1}$ denotes a unity vector (where all elements are equal to one). That is, for each element in the vectors X_{LW} and X_{UP}, the following condition has to hold:

$$\begin{aligned} \mathrm{X}_{LW}(i) &\leq x_i, \\ \mathrm{X}_{UP}(i) &\leq 1 - x_i, \\ \implies x_i &\leq 1 - \mathrm{X}_{UP}(i). \end{aligned}$$

As already mentioned, the functions testLO() and testSW() — shown in Alg. 2 and Alg. 3 — test schedulability in LO mode and at the transitions from LO to HI mode. These two functions are very similar — apart from testLO() dealing with the whole τ and testSW() with the subset τ_{HI} — and return lower bounds on x_i and on $1 - x_i$ respectively. Thus, the following discussion of testLO() also applies to testSW().

Basically, testLO() computes $\mathrm{dbf}_{LO}(t)$ for all $0 \leq t \leq \hat{t}_{LO}$ starting from $x_i = 1$ for all HI tasks. If the current t corresponds to a deadline of a HI task (lines 10 to 14), its (relative) virtual deadline $x_i \cdot D_i$

Algorithm 3 Function testSW()

Require: τ_{HI}

1: Compute $\hat{t}_{SW}$ by (7)
2: $X_{UP} = \mathbf{1}$
3: **while** $t \leq \hat{t}_{SW}$ **do**
4: **if** $\mathrm{dbf}_{SW}(t) > t$ **and** $\mathrm{dbf}_{SW}(t) - r_i > D_i$ **then**
5: $X_{UP} = \emptyset$
6: Return
7: **end if**
8: **if** Computed(i)='false' **or** $X_{UP}(i) < \frac{\mathrm{dbf}_{SW}(t)-r_i}{D_i}$ **then**
9: $X_{UP}(i) = \frac{\mathrm{dbf}_{SW}(t)-r_i}{D_i}$ /* r_i = job's release time */
10: **end if**
11: (t, i)=getNextDeadline()
12: **end while**
13: Return

is adjusted such that its absolute deadline $r_i + x_i \cdot D_i$ is equal to $\mathrm{dbf}_{LO}(t)$ (i.e., the total execution demand at t).

Note that the execution demand of jobs with prior deadlines to t is contained in $\mathrm{dbf}_{LO}(t)$. As a result, the computed x_i in line 12 can never compromise schedulability of these previous jobs. In addition, the currently computed x_i can only replace a previously computed x_i, if it is greater than this latter (lines 11 to 13). This deadline tightening reduces the number of possibilities for x_i, but it also reduces the complexity of the algorithm.

Computed(i) in line 11 returns 'false', if no x_i has been computed yet for the current i. The function getNextDeadline() in line 15 returns the point in time t at which the next deadline occurs and the index i of the task corresponding to that deadline. Clearly, this function has to take the computed values of x_i into account.

The function testLO() succeeds if it finishes testing $\mathrm{dbf}_{LO}(t)$ for $0 \leq t \leq \hat{t}_{LO}$ and it could find a value of x_i in $(0, 1]$ for each HI task in τ. On the other hand, testLO() fails, if $\mathrm{dbf}_{LO}(t) > t$ holds for some t and either t corresponds to a deadline of a LO task — whose deadline cannot be adjusted by the used tightening technique — or the resulting x_i becomes greater than 1 (lines 4 to 8).

Analogous to testLO(), testSW() computes $\mathrm{dbf}_{SW}(t)$ for all $0 \leq t \leq \hat{t}_{SW}$ starting from $1 - x_i = 1$ for all HI tasks – recall that deadlines in $\mathrm{dbf}_{SW}(t)$ are equal to $(1 - x_i) \cdot D_i$. Otherwise, as mentioned above, testLO() and testSW() are very similar and, hence, the above explanation for testLO() also applies to testSW(). Finally, the function testHI() in Alg. 1 is the known schedulability test for EDF from the literature [18] and, hence, does not require further discussion.

7 EXPERIMENTAL EVALUATION

In this section, we evaluate the proposed technique from Section 4 based on synthetic data and compare it to the most prominent approaches from the literature. The intention of this section is to show how the different algorithms roughly behave with respect to each other (and not to prove any particular behavior, in contrast, to the previous sections).

In particular, we compare the proposed technique in form of Alg. 1 with EDF-VD [1], with the GREEDY algorithm by Ekberg and Yi [2], and with ECDF by Easwaran [3].

It should be noted that the proposed Alg. 1 as well as GREEDY and ECDF essentially perform two inter-related functions: (i) selection of x_i parameters by some deadline tightening technique, (ii) schedulability test for a given set of x_i. Clearly, the more accurate the schedulability test is, the better the selection of x_i is and vice versa.

As discussed above, the aim of this paper is not to improve the deadline tightening, but to propose a new technique to bound execution demand of mixed-criticality EDF. As a result, the proposed Alg. 1 makes use of a rather rudimentary (though less complex) tightening technique compared to those of the GREEDY and the ECDF algorithms. However, on average, experimental results evidence that benefits overcome drawbacks by the proposed Alg. 1.

Finally, note that we had to modify EDF-VD to consider the deadlines D_i of tasks instead of their inter-arrival times or periods T_i, i.e., to consider the tasks' densities instead of their utilizations, to account for the case of constrained deadlines $D_i \leq T_i$ and be meaningfully compared with the other algorithms in this section.

Obtaining test data. Now, we explain how we obtained test data for our experiments. The description below is common to all curves presented in the paper. Details concerning a specific curve will be given as it becomes necessary.

Basically, we used the algorithm *UUniFast* [19][20] to generate sets of 10 and 20 tasks for a varying *LO utilization*, i.e., for a varying $U_{LO}^{LO} + U_{HI}^{LO}$. For each curve, a total number of 5,000 different task sets were created.

For a given value of LO utilization, UUniFast returns a vector of (individual) task utilizations. We then generate periods T_i randomly and use the task utilization to obtain the values of C_i^{LO}. Now, given a percentage of HI tasks, which we vary in our experiments, we assumed that HI task experience a random increase in execution demand in HI mode of either 10% or 100% more of their $\frac{C_i^{LO}}{T_i}$. With this, we then obtained the values of C_i^{HI}. Further we randomly selected D_i in $[C_i^{HI}, T_i]$ for HI tasks and in $[C_i^{LO}, T_i]$ for LO tasks.

7.1 Comparison for sets of 10 tasks

Fig. 1 to Fig. 6 show the results of our experiments for 10 tasks and a varying number of HI tasks. In Fig. 1, Fig. 2 and Fig. 3, only one HI task was considered (10% of n) for an increasing amount of HI execution demand.

As we can see, in this case, the proposed algorithm and ECDF behave almost the same with ECDF being slightly better for an increase in HI execution demand that goes from 10% in Fig. 1 to 100% in Fig. 3, i.e., the HI task doubles its execution demand in HI mode. The GREEDY algorithm is outperformed by the proposed algorithm and by ECDF by around 10%, i.e., the proposed algorithm and ECDF constantly find around 10% more tasks sets that are feasible under mixed-criticality EDF than the GREEDY algorithm.

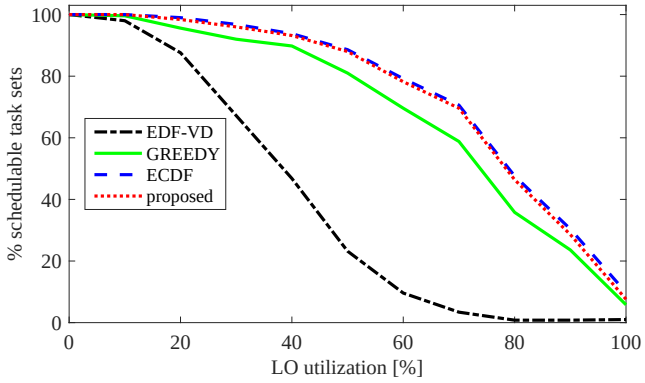

Figure 1: Schedulability vs. LO utilization for $n = 10$ and 10% of HI tasks with 10% increase of HI execution demand

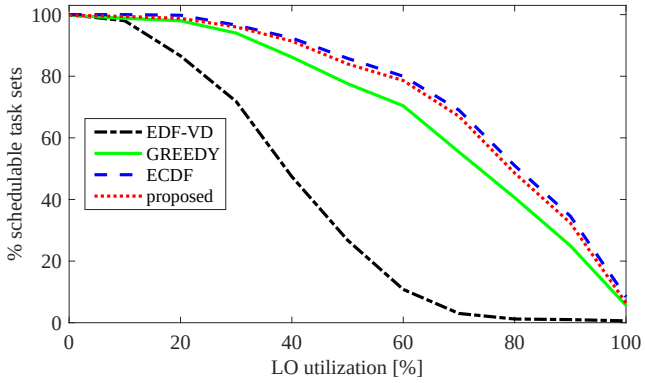

Figure 2: Schedulability vs. LO utilization for $n = 10$ and 10% of HI tasks with 50% increase of HI execution demand

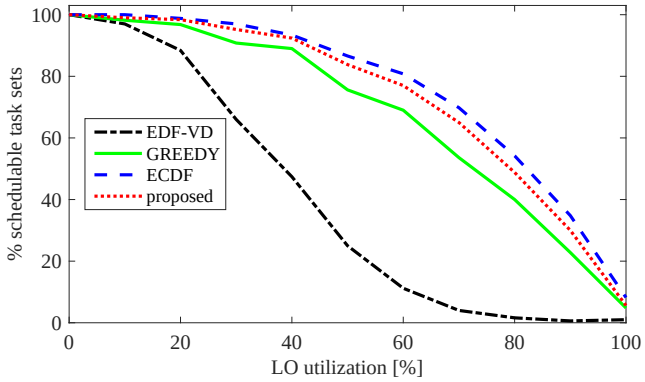

Figure 3: Schedulability vs. LO utilization for $n = 10$ and 10% of HI tasks with 100% increase of HI execution demand

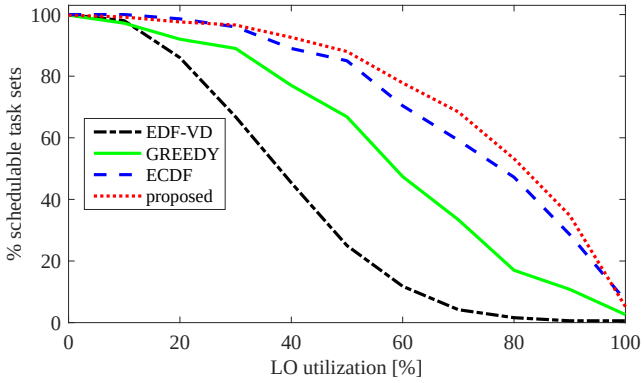

Figure 4: Schedulability vs. LO utilization for $n = 10$ and 30% of HI tasks with 10% increase of HI execution demand

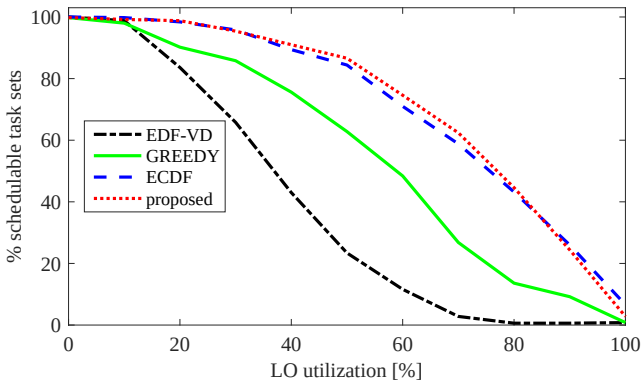

Figure 5: Schedulability vs. LO utilization for $n = 10$ and 30% of HI tasks with 50% increase of HI execution demand

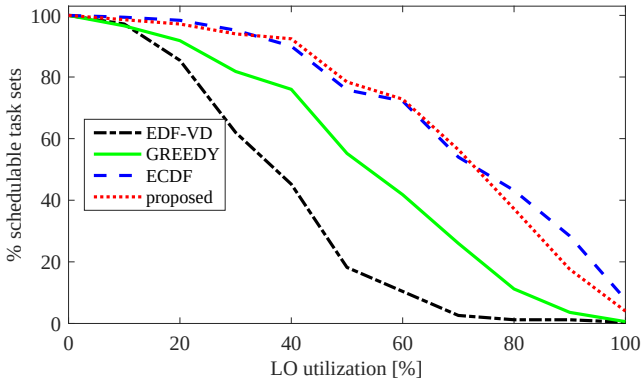

Figure 6: Schedulability vs. LO utilization for $n = 10$ and 30% of HI tasks with 100% increase of HI execution demand

In Fig. 4, Fig. 5 and Fig. 6, three HI tasks were considered (30% of n) again for an increasing amount of HI execution demand. In this case, the proposed algorithm and ECDF behave almost the same; however, the proposed algorithm is most of the time slightly better than ECDF for an increase in HI execution demand that goes from 10% in Fig. 4 to 100% in Fig. 6, i.e., the three HI tasks double their execution demand in HI mode.

The GREEDY algorithm is outperformed by the proposed and ECDF by around 20% this time, i.e., the proposed algorithm and

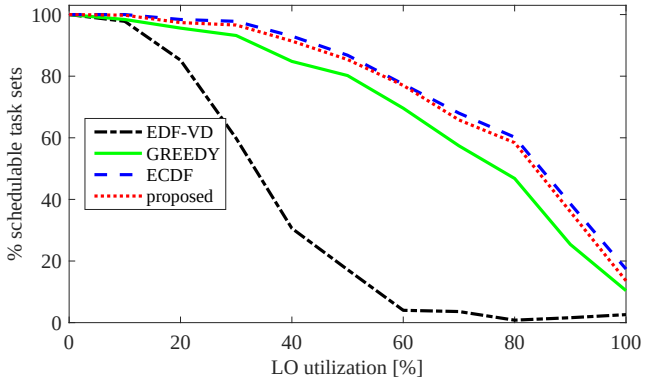

Figure 7: Schedulability vs. LO utilization for $n = 20$ and 10% of HI tasks with 10% increase of HI execution demand

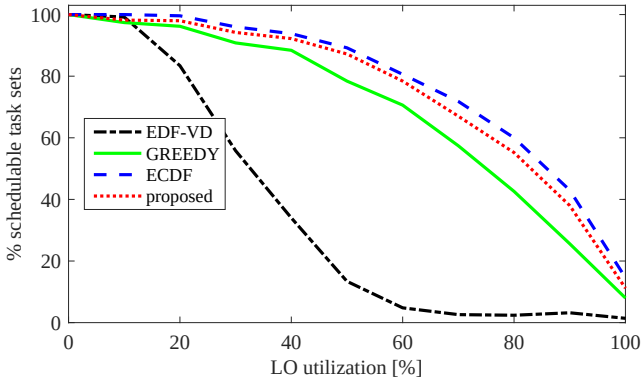

Figure 8: Schedulability vs. LO utilization for $n = 20$ and 10% of HI tasks with 50% increase of HI execution demand

ECDF constantly find around 20% more tasks sets that are feasible under mixed-criticality EDF.

Even though there is not much difference between the proposed algorithm and ECDF, it should be noted that ECDF has a more sophisticated deadline tightening technique than Alg. 1. Tightening deadlines in more efficient manner will certainly improve Alg. 1's performance.

7.2 Comparison for sets of 20 tasks

Fig. 7 to Fig. 12 show the results of our experiments for 20 tasks and a varying number of HI tasks. In Fig. 7, Fig. 8 and Fig. 9, two HI tasks were considered (10% of n) for an increasing amount of HI execution demand.

Here, it can be seen that the proposed algorithm and ECDF behave almost the same with ECDF being again slightly better for an increase in HI execution demand that goes from 10% in Fig. 7 to 100% in Fig. 9. The GREEDY algorithm is again outperformed by the proposed and the ECDF algorithm by around 10%, i.e., these latter find around 10% more tasks sets that are feasible under mixed-criticality EDF.

In Fig. 10, Fig. 11 and Fig. 12, six HI tasks were considered (30% of n) for an increasing amount of HI execution demand. This time the proposed algorithm considerably outperforms ECDF by around 10% to 20% more accepted task sets. The GREEDY algorithm is still outperformed by ECDF and the proposed one.

The performance of the proposed algorithm is the highest for the experiments in Fig. 10, i.e., where there are a relatively big number of HI tasks, each of which experiences a relatively small increase in execution demand in HI mode. Again, the performance of the proposed Alg. 1 can be further improved by using a more sophisticated deadline tightening scheme.

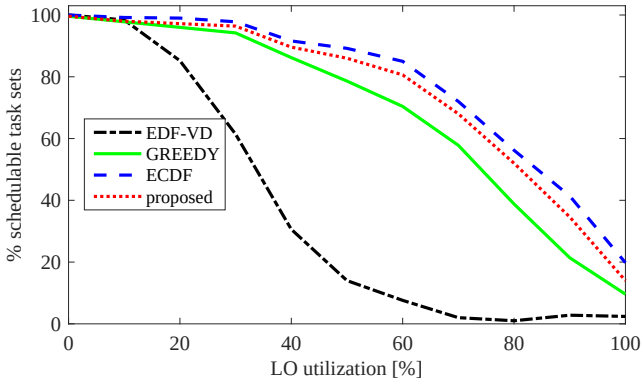

Figure 9: Schedulability vs. LO utilization for $n = 20$ and 10% of HI tasks with 100% increase of HI execution demand

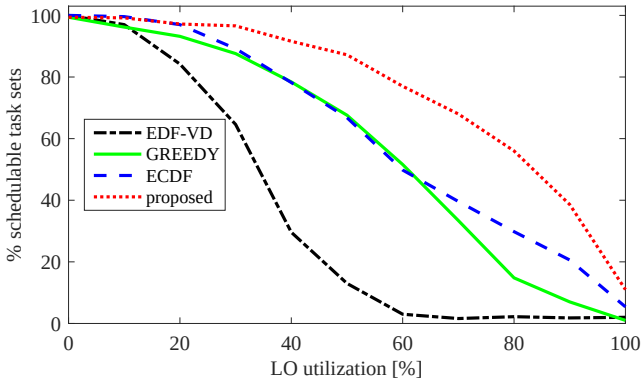

Figure 10: Schedulability vs. LO utilization for $n = 20$ and 30% of HI tasks with 10% increase of HI execution demand

8 CONCLUDING REMARKS

In this paper, we studied the problem of mixed-criticality scheduling under EDF, where a mix of low-criticality (LO) and high-criticality (HI) tasks share the processor. Similar to the literature, we characterize the execution demand of a mixed-criticality task set by deriving demand bound functions in the different operation modes, viz., HI and LO mode.

However, it was shown that treating the transitions from LO to HI mode separately from the stable HI mode allows working around carry-over jobs and, therefore, reducing pessimism in estimating the execution demand under mixed-criticality EDF. Carry-over jobs are

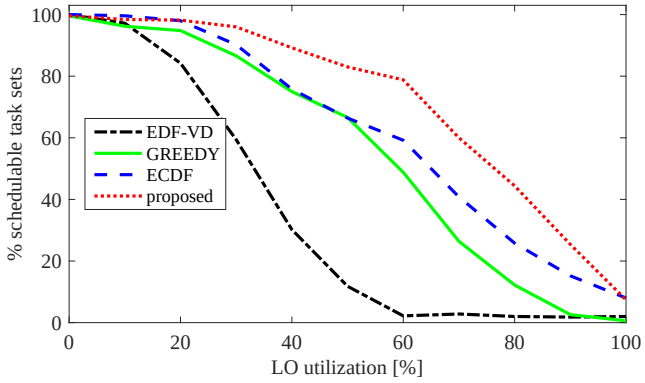

Figure 11: Schedulability vs. LO utilization for $n = 20$ and 30% of HI tasks with 50% increase of HI execution demand

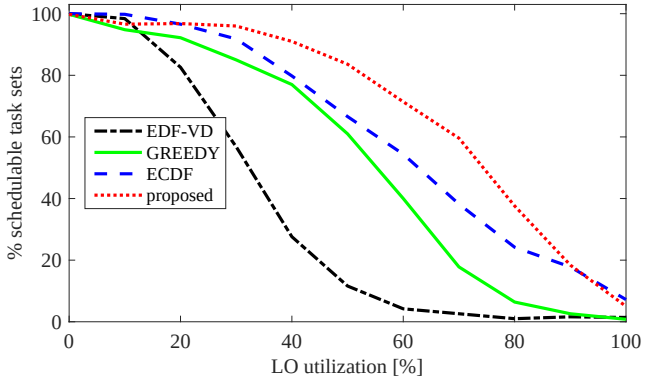

Figure 12: Schedulability vs. LO utilization for $n = 20$ and 30% of HI tasks with 100% increase of HI execution demand

those HI jobs that start prior to, but have not yet finished executing at the moment of switching from LO mode to HI mode.

It is interesting to notice that the proposed technique reduces the problem of testing schedulability under mixed-criticality EDF to testing schedulability of three *almost* unrelated task sets: the one in LO mode, the one in HI mode and the equivalent task set for transitions between LO and HI mode. This leads to a considerably simpler schedulability test and improves our understanding of this problem.

We illustrated the performance of the proposed technique by an analytical comparison with the known approaches from the literature and by means of experiments based on a large number of synthetic task sets.

ACKNOWLEDGMENT

Mitra Mahdiani was funded by the German Academic Exchange Service (DAAD). The authors would like to thank RTNS reviewers for the valuable comments and suggestions.

REFERENCES

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

[2] P. Ekberg and W. Yi, "Bounding and shaping the demand of mixed-criticality sporadic tasks," in *Proc. of Euromicro Conference on Real-Time Systems (ECRTS)*, 2012.

[3] A. Easwaran, "Demand-based scheduling of mixed-criticality sporadic tasks on one processor," in *Proc. of Real-Time Systems Symposium (RTSS)*, Dec. 2013.

[4] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *Proc. of Real-Time Systems Symposium (RTSS)*, 2007.

[5] S. Baruah, A. Burns, and R. Davis, "Response-time analysis for mixed criticality systems," in *Proc. of Real-Time Systems Symposium (RTSS)*, 2011.

[6] S. Baruah, V. Bonifaci, G. D'Angelo, A. Marchetti-Spaccamela, S. Van Der Ster, and L. Stougie, "Mixed-criticality scheduling of sporadic task systems," in *Proc. of European Symposium on Algorithms (ESA)*, 2011.

[7] 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," *Journal of the ACM (JACM)*, vol. 62, no. 2, 2015.

[8] P. Ekberg and W. Yi, "Bounding and shaping the demand of generalized mixed-criticality sporadic task systems," *Real-Time Systems (RTS)*, vol. 50, no. 1, 2014.

[9] P. Huang, P. Kumar, G. Giannopoulou, and L. Thiele, "Run and be safe: Mixed-criticality scheduling with temporary processor speedup," in *Proc. of Design, Automation and Test in Europe (DATE)*, March 2015.

[10] H. Su and D. Zhu, "An elastic mixed-criticality task model and its scheduling algorithm," in *Proc. of Design, Automation and Test in Europe (DATE)*, 2013.

[11] T.-W. Kuo and A. K. Mok, "Load adjustment in adaptive real-time systems," in *Proc. of Real-Time Systems Symposium (RTSS)*, 1991.

[12] Q. Zhao, Z. Gu, and H. Zeng, "PT-AMC: Integrating Preemption Thresholds into Mixed-Criticality Scheduling," in *Proc. of Design, Automation and Test in Europe (DATE)*, 2013, pp. 141–146.

[13] Y. Wang and M. Saksena, "Scheduling fixed-priority tasks with preemption threshold," in *Proc. of Real-Time Computing Systems and Applications (RTCSA)*, 1999.

[14] A. Masrur, D. Müller, and M. Werner, "Bi-level deadline scaling for admission control in mixed-criticality systems," in *Proc. of IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, Aug. 2015.

[15] S. Baruah, B. Chattopadhyay, H. Li, and I. Shin, "Mixed-criticality scheduling on multiprocessors," *Real-Time Systems (RTS)*, vol. 50, 2013.

[16] R. Pathan, "Schedulability analysis of mixed-criticality systems on multiprocessors," in *Proc. of Euromicro Conference on Real-Time Systems (ECRTS)*, 2012.

[17] A. Burns and R. Davis, "Mixed criticality systems - a review," Department of Computer Science, University of York, Tech. Rep., 2015.

[18] S. Baruah, A. Mok, and L. Rosier, "Preemptively scheduling hard-real-time sporadic tasks on one processor," in *Proc. of Real-Time Systems Symposium (RTSS)*, Dec. 1990.

[19] E. Bini and G. Buttazzo, "Biasing effects in schedulability measures," in *Proc. of Euromicro Conference on Real-Time Systems (ECRTS)*, 2004.

[20] ——, "Measuring the performance of schedulability tests," *Real-Time Systems (RTS)*, vol. 30, no. 1-2, 2005.