On the Analysis of EDF-VD Scheduled Mixed-Criticality Real-Time Systems

Tianyu Zhang[1], Nan Guan[1,2], Qingxu Deng[1] and Wang Yi[1,2]

[1] Northeastern University, China
[2] Uppsala University, Sweden

***Abstract*—Real-time embedded systems usually integrate multiple functionalities of different criticality levels on a shared hardware platform. For these mixed-criticality real-time systems it is a challenging problem to efficiently utilize system resource to satisfy all the timing constraints on different criticality levels. A simple yet efficient algorithm EDF-VD has recently been proposed to schedule mixed-criticality real-time systems, and shown promising real-time performance. However, the competency of EDF-VD has not been fully exploited due to the imprecise underlying analysis techniques. In this paper, we develop new schedulability analysis methods for EDF-VD. Different from previous analysis methods that separate the analysis on each individual criticality level, our new analysis looks into system behavior crossing multiple criticality levels to obtain more precisely analysis results. Experiments show that our new analysis method can significantly improve guaranteed schedulability of EDF-VD, especially for systems with more criticality levels. The price paid for improved schedulability is higher analysis complexity, but a combination of our new techniques and previous methods can obtain a good balance between the analysis precision and efficiency.**

I. Introduction

An increasing trend in embedded systems is to integrate functionalities of different criticality levels on a shared platform for better cost and power efficiency. In such *mixed-criticality* systems, we need to guarantee the temporal correctness of high-criticality functionalities at a higher level of assurance, using very pessimistic assumptions (e.g., with worst-case execution time upper bounds obtained by static analysis) which are unlikely to occur in reality. At a lower level of assurance, we want to guarantee the temporal correctness of all functionalities, but under less pessimistic assumptions (e.g., with measured worst-case execution times).

Mixed-criticality systems bring significant challenges to the design of real-time systems. Traditional real-time scheduling algorithms like EDF and RM may lead to great resource waste. In recent years, several new scheduling algorithms have been proposed to improve the resource usage efficiency by systematically exploring the asymmetry among different criticality levels. Among those, Earliest-Deadline-First with Virtual Deadlines (EDF-VD) has shown clearly better performance than others in both schedulability and run-time efficiency. The idea of EDF-VD, as suggested by its name, is to use EDF as the underlying scheduling decision policy, and balances the schedulability on different criticality levels by setting different virtual deadlines of a task when it executes in on different criticality modes.

Although EDF-VD has shown better performance than traditional real-time scheduling algorithms and other newly developed algorithms, its advantage actually has not been fully exploited. The state-of-the-art analysis techniques for EDF-VD are still rather pessimistic, which may reject a considerable amount of task systems that are indeed schedulable. The aim of this paper is to develop new techniques to more precisely analyze the schedulability of EDF-VD.

The challenge of analyzing EDF-VD is raised by the *criticality mode switch* behavior at run time, which is triggered by the overrun of certain tasks. It is generally unknown when the criticality mode switch happens and what is the system state at the mode switch point (e.g., how much work has been accomplished of the current instance of each task). It is computationally infeasible to enumerate all the possibilities, and people resort to over-approximation to trade precision for analysis efficiency. Ekberg and Yi in [11], [12] proposed schedulability analysis techniques for EDF-VD, which we called the EY analysis in this paper. The main idea of EY analysis is to separate the analysis of different criticality levels by approximating the workload that is carried from a lower criticality level to a higher one. However, such an approximation is very inaccurate and the workload estimation error accumulates over multiple criticality levels. Therefore, the analysis results are rather pessimistic, especially for systems with more criticality levels.

In this paper, we develop new analysis techniques for EDF-VD. Different from the EY analysis that separates the analysis of different criticality levels, our analysis considers the system behavior crossing multiple criticality levels to more precisely bound the system workload and yields more precisely analysis results.

Experiments show that the new analysis presented in this paper promises much better performance over the EY analysis. And as the number of criticality levels increases, the superiority of our new analysis method is more significant. The price we pay for the improving analysis precision is a higher analysis complexity. However, by combining our new analysis and the EY analysis we can gain a good balance between the analysis accuracy and efficiency.

II. Related Work

The mixed-criticality scheduling problem was first identified and formalized by Vestal in [18], where he proposed a fixed-priority algorithm to schedule such systems and a response time analysis technique to analyze its schedulability. Dorin et. al. [9] proved that the algorithm in [18] is optimal in the scope of fixed-task-priority preemptive algorithms. However, as pointed out by Baruah and Vestal [5], the algorithm in [18] is not optimal if we are not restricted to fixed-taskpriority preemptive algorithms, and is actually incomparable with the EDF algorithm.

Baruah et. al. conducted a series of fundamental works on a simpler model consisting of a finite set of jobs with fixed release times. They showed that deciding the feasibility of such job sets is strongly NP-hard even if all the jobs are released at the same time [1]. They proposed an effective heuristic algorithm OCBP [4], which has a quantitative performance guarantee in terms of the golden ratio speedup factor. OCBP is then extended to mixed-criticality *sporadic* task systems by Li and Baruah [17] with a runtime adjustment algorithm with pseudo-polynomial complexity, and later is improved with more efficiency algorithms with quadratic and linear complexity [15], [13]. The response time analysis problem of mixed criticality systems is studied in [3], where the effect of different runtime support to the schedulability is evaluated.

Baruah et. al. [2] proposed an efficient algorithm EDF-VD (EDF with Virtual Deadlines) to schedule mixed-criticality task systems. The idea of this algorithm is to set different virtual deadlines for a task on different criticality levels, such that a higher-criticality task expect to finish its low-criticality workload early and save enough time to finish the extra workload before its deadline if criticality mode switch occurs. Ekberg and Yi [11] developed techniques for EDF-VD (the EY analysis) to bound the demand bound of each task on different criticality levels, quantified the negative effects and positive effects of shortening the virtual deadline of a task in the low criticality level, and proposed an algorithm to search for suitable virtual deadline configurations. This analysis technique is then generalized in [12] to the case of more general models with more criticality levels with other criticality-sensitive task parameters. Recently, Easwaran improves the EY analysis for dual-criticality systems [10]. The idea of Easwaran's work inspire the new analysis method of this paper. While Easwaran's analysis is only applicable to dual-criticality systems, the main contribution of this paper is to improve the schedulability of systems with more than two criticality levels. The EDF-VD algorithm has also been applied to multiprocessor platforms, with both global and partitioned scheduling algorithms [6], [16], [14].

Despite a relatively short history since Vestal's first paper formalizing the mixed-criticality real-time scheduling problem, a surprisingly large amount of attentions have been drawn in the real-time systems research community to this important yet challenging problem. Many important literatures are not listed here due to the space limit, and we finally refer to [8], a comprehensive survey of the work on real-time scheduling for mixed-criticality systems.

III. Preliminaries

A. Mixed-Criticality Task Systems

The mixed-criticality task system τ, as a generalization of traditional sporadic task systems, consists of a set of independent recurring tasks. Each mixed-criticality task $\tau_i \in \tau$ is characterized by a tuple (T_i, D_i, ℓ_i, C_i):

- T_i is τ_i's minimum inter-release separation time.
- D_i is τ_i's relative deadline.
- $\ell_i \in \{1, \cdots, L\}$ is the criticality level, where L is the total number of criticality levels of the system. We use a small number to denote a lower criticality level.
- $C_i = (C_i^1, \cdots, C_i^L)$ is τ_i's WCET vector, where C_i^l denotes the task's WCET on the l^{th} criticality level. We assume $x < y \Rightarrow C_i^x \leq C_i^y$

Each task τ_i releases potentially infinitely many *jobs*. We use J_i^k to denote the k^{th} job released by τ_i. If τ_i releases a job at time r, then this job's absolute deadline is at time $d = r + D_i$. For simplicity of notation, we also use J_i to denote a job of task τ_i when the context is clear.

The system starts with criticality mode 1. If any job of a task τ_i executes for C_i^1 without signaling that it has finished, the system will immediately switch to the 2^{nd} criticality mode. After that, it is not required to meet any deadlines for tasks with $\ell_i = 1$, but jobs of other tasks may instead execute for up to their C_i^2. Similarly, if any job of a task τ_i executes for C_i^2 without signaling that it has finished, the system will immediately switch to the 3^{rd} criticality mode. In practice, the system can switch back from a higher criticality mode to a lower criticality mode when, e.g., the processor is idle, but from the modeling perspective we simply view tasks of criticality l or lower as being discarded along with its active jobs at the time when the system switches from the criticality l to $l+1$. Since τ_i is immediately aborted as soon as the system is switched to criticality mode l, there is no need to define C_i^l for task τ_i for all $l: \ l > \ell_i$. We call criticality mode $l-1$ and $l+1$ the *preceding* and *succeeding* mode of criticality mode l, respectively.

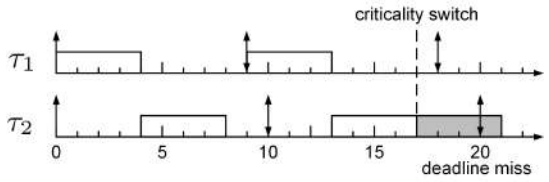
We use $\tau(l)$ to denote the set of tasks of criticality l, i.e., $\tau(l) = \{\tau_i | \ell_i = l\}$. Then we can define the schedulability of the system:

Definition 3.1 (Schedulability): The mixed-criticality task set is schedulable if and only if all tasks in $\bigcup_{k \geq l} \tau(k)$ meet their deadlines when the system is running in criticality mode l.

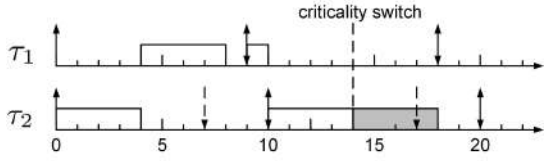
For clarity, in the special case where the system has dual-criticality (i.e., $\ell_i \in \{1, 2\}$), we use LO instead of 1 and HI instead of 2 to represent the low and high criticality, respectively.

TABLE I
MIXED-CRITICALITY TASK SET EXAMPLE

tasks	C_i^{LO}	C_i^{HI}	$T_i(D_i)$	ℓ_i	D_i^{LO}	D_i^{HI}
τ_1	4	–	9	LO	9	–
τ_2	4	8	10	HI	7	10



(a) unschedulable by EDF



(b) schedulable by EDF-VD

Fig. 1. Illustration of the task set in Table I scheduled by EDF-VD.

B. The EDF-VD Algorithm

EDF is optimal to schedule traditional sporadic task systems (with only one criticality level). However, EDF may lead to poor performance when applied to mixed-criticality task systems introduced in above. For example, we schedule the dual-criticality task set in Table I by EDF, assuming both tasks synchronously release the first jobs at time 0. The second job of the HI-criticality task τ_2 has executed for its $C_2^{\text{LO}} = 4$ without signalling completion, which triggers the criticality mode switch at time 17. Then τ_1 is aborted, but τ_2 has to execute for up to $C_2^{\text{HI}} = 8$, which causes a deadline miss. The example illustrates that a HI-criticality job may need to be prioritized for execution in the LO-criticality if even its absolute deadline is later, in order to save enough slack time to finish its extra workload on the HI-criticality level in case of a mode switch occurs.

EDF-VD [2] uses different "virtual" deadlines for a task when the system executes in different criticality modes. The scheduler uses these virtual deadlines to decide the priority order of active jobs. We use D_i^{HI} to D_i^{LO} to denote the HI-criticality and LO-criticality virtual deadline of τ_i in a dual-criticality system, and use D_i^l to denote the virtual deadline of τ_i in criticality mode l. In criticality mode l, the virtual deadline of a task with $\ell_i < l$ is undefined since this task has been aborted. If we set $D_2^{\text{LO}} = 7$ (and keep all other virtual deadlines the same as the original deadlines), the scheduling sequence of the task set is shown in Figure 1-(b), where the second job of τ_2 is prioritized for execution over the second job of τ_1. When τ_2 overruns and triggers the criticality mode switch, there are enough slack time for it to finish C_2^{HI} by its real absolute deadline.

By using the *virtual deadlines*, we can obtain extra slack time for jobs crossing the mode switch point and decrease the workload of a HI-criticality task τ_i "carried over" into the HI-mode. On the other hand, if it becomes more difficult to successfully schedule the system in the LO-criticality mode we enforce shorter virtual deadlines of some tasks. So it is a tradeoff between the schedulability in different criticality modes when choosing appropriate virtual deadlines.

C. Existing Analysis Methods of EDF-VD

For a mixed-criticality system, we want to answer the question that whether we can find a configuration of the task virtual deadlines on each criticality level such that the system is schedulable by EDF-VD. Two problems need to be solved to answer the above question:

1) *Schedulability Test*. Given a fixed virtual deadline configuration, how to decide whether the system is schedulable under EDF-VD.
2) *Tuning Virtual Deadlines*. Given a certain schedulability test method, how to configure the task virtual deadlines to make the system schedulable.

Schedulability Test

Ekberg and Yi studied the schedulability test problem of EDF-VD in [11]. They assumes a dual-criticality system. The schedulability of the LO-criticality mode is tested by the standard demand bound function analysis method [7]. To analyze the schedulability of the HI-criticality mode, each HI-criticality job that may cross the criticality switch point is divided into two parts, the sub-job before its HI-criticality virtual deadline, and the sub-job (called the carry-over job) between its HI-criticality virtual deadline and original deadline. The analysis then only looks into the time intervals starting at or after the system switches to HI-criticality mode.

Applying the EY analysis described above to the task set in Table I, the carry-over job has execution time lager than the time interval between its "release" and its deadline ($C_2^{\text{HI}} - C_2^{\text{LO}} > D_2^{\text{HI}} - D_2^{\text{LO}}$), so the task system is determined as unschedulable, as shown in Figure 2. The main reason why the EY analysis makes such a pessimistic decision is that it assumes the LO-criticality computation requests are finished just by before D_i^{LO} and its extra workload in HI-criticality mode ($C_i^{\text{HI}} - C_i^{\text{LO}}$) all need to be executed in the time interval between the two virtual deadlines D_i^{LO} and D_i^{HI}. At runtime the LO-criticality computation request actually may be finished much earlier than D_i^{LO}, which cannot be explored by the EY analysis.

The schedulability test in [2] (referred to as BA test) uses the similar approach with the EY analysis, but is based on utilization bound rather than demand bound function. The BA test suffers the same problem as the EY analysis as described above, and is even more pessimistic (due to the workload approximation to derive an elegant utilization bound).

Tuning Virtual Deadlines

In [2] the virtual deadlines are chosen by setting fixed ratios between the virtual deadlines on different criticality

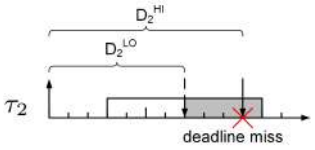


Fig. 2. Illustration of EY analysis

levels for *all* HI-criticality tasks. A finer grained tuning strategy is used in [11], [12], which set task virtual deadlines from the highest to the lowest criticality, and greedily shortens individual task's virtual deadline in a lower criticality mode to improve the schedulability of the higher criticality modes.

IV. New Schedulability Analysis

As we introduced in the last section, the EY analysis method in [11], [12] suffers the inaccurate estimation of the workload of jobs crossing the criticality mode switch point. Particularly, the analysis of a particular criticality mode assumes that each crossing job performs its execution in the preceding mode as late as possible. However, at runtime some of these crossing jobs may already have finished earlier. The separate analysis cannot explore this as it only looks into intervals that start from the mode switch points.

To solve the above problem and improve the analysis accuracy, we propose a new schedulability analysis method. The new analysis method looks into time intervals crossing the criticality mode switch points, to obtain a more comprehensive view of the system behavior and thus provide a more accurate schedulability analysis.

In the following, we first present the new analysis method with dual-criticality systems (only have two criticality modes LO and HI) and then generalize it to the case of more criticality levels.

A. Dual-Criticality Systems

To prove the schedulability of a dual-criticality task system, we need to show that:

1) All tasks (of both criticality level LO and HI) meet their deadlines when the system is executing in the LO criticality mode.
2) All the HI-criticality tasks (tasks in $\tau(\text{HI})$) meet their deadlines when the system is executing in the HI criticality mode.

We can simply use the standard demand bound function based schedulability test condition [7] to check whether 1) holds or not:

$$\forall x : \sum_{\tau_i \in \tau} \mathsf{dbf}_i^{\text{LO}}(x) \leq x$$

where

$$\mathsf{dbf}_i^{\text{LO}}(x) = n_i(x) \times C_i^{\text{LO}} \tag{1}$$

is the demand bound function of task τ_i in the LO-criticality mode, where $n_i(x)$ is the maximal number of jobs of τ_i with both release times and absolute deadlines in a time interval of length x [7]:

$$n_i(x) = \max(\lfloor (x - D_i)/T_i \rfloor + 1, 0)$$

However, it is more difficult to check whether 2) holds or not. It is not sufficient to use $n_i(x) \times C_i^{\text{HI}}$ as the demand bound of task τ_i in the HI-criticality mode, since there exists a "transition phase" after the criticality mode switch point. In the following we focus on the analysis of 2).

We look into a time interval $[t_1, t_d]$, where t_d is an arbitrary time point after the mode switch point t_s, and t_1 is the earliest time point no later than t_d such that at any time point in $[t_1, t_s]$ $\tau(\text{HI})$ has at least one active job (released but not finished yet) with deadline no later than t_d. If t_1 is later than t_s, we know all the jobs released before t_s have finished by t_1, and the workload in time interval $[t_1, t_d]$ are all contributed by jobs release after t_1, which is bounded by the standard demand bound function dbf. In the following, we focus on the more interesting case of $t_1 \leq t_s$. We define $x = t_d - t_1$ and $y = t_d - t_s$.

The total number of jobs of a HI-criticality task τ_i with both release times and deadlines within a time interval of size x is bounded by $n_i(x)$. If a job is finished before t_s, it can only execute for at most C_i^{LO}, since otherwise the criticality mode switch should have been triggered before t_s. So our target is to bound the number of jobs, among these $n_i(x)$ jobs, that can execute in $[t_s, t_d]$. This number, not only depends on the value of x, but also depends on y, so we denote it by $m_i(x, y)$. Therefore, the total workload of task τ_i in $[t_1, t_d]$ is calculated by

$$\mathsf{dbf}_i^{\text{HI}}(x, y) = m_i(x, y) \times C_i^{\text{HI}} + \big(n_i(x) - m_i(x, y)\big) \times C_i^{\text{LO}} \tag{2}$$

It is clear that the jobs that are released after the criticality mode switch point should be counted in $m_i(x, y)$. The question is whether we should count the *crossing job*, which is defined as follows.

Definition 4.1 (Crossing Job): A *crossing job* has its release time before and absolute deadline after the criticality mode switch point, respectively.

In the following we will show conditions to safely rule out the crossing job from $m_i(x, y)$, by which upper bounds of $m_i(x, y)$ are derived.

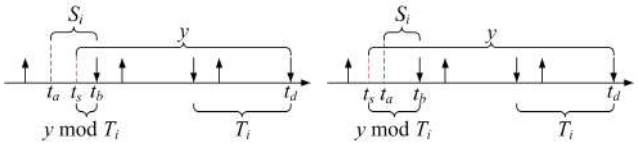
Lemma 4.2: If a HI-criticality task τ_i can meet its LO-criticality deadline D_i^{LO} in the LO-criticality mode, then $m_i(x, y)$ is bounded by $p_i(x, y)$:

$$p_i(x, y) = \begin{cases} n_i(y), & y \bmod T_i < S_i \\ \min(n_i(y) + 1, n_i(x)), & \text{otherwise} \end{cases}$$

where $S_i = D_i^{\text{HI}} - D_i^{\text{LO}}$.

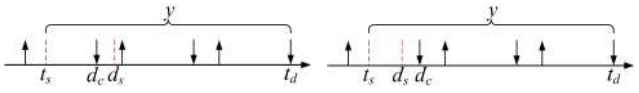
Proof: Let t_a be the time point at which the crossing job J_i either signalled completion or had executed for C_i^{LO}, and let t_b be the absolute deadline of J_i.

If $t_a < t_s$, as shown in Figure 3-(a), the crossing job must have signalled completion by t_a (otherwise the criticality mode switch occurs before t_s). In this case only normal



(a) $t_a < t_s$ (b) $t_a \geq t_s$

Fig. 3. Illustration of the proof of Lemma 4.2



(a) $d_c < d_s$ (b) $d_c \geq d_s$

Fig. 4. Illustration of the proof of Lemma 4.3

jobs of τ_i can execute in $[t_s, t_d]$, the number of which is bounded by $n_i(y)$.

If $t_a \geq t_s$, as shown in Figure 3-(b), the crossing job may be unfinished by t_s. In this case both the crossing job and the normal jobs can execute in $[t_s, t_d]$, and the total number of them is bounded by $n_i(y)+1$. Furthermore, this bound can be tightened by $n_i(x)$ to exclude the case that the crossing job is released before t_1. In summary, in this case $m_i(x, y)$ is bounded by $\min(n_i(y) + 1, n_i(x))$.

The remaining of the proof is to show how to decide whether it holds $t_a < t_s$. We consider the worst-case release pattern[1] of task τ_i where its jobs are released as fast as possible and one of its job's deadline aligns with t_d, as illustrated in Figure 3, by which we conclude that

$$y \bmod T_i < t_b - t_a \Rightarrow t_a < t_s \tag{3}$$

Since τ_i can meet its LO-criticality deadline D_i^{LO} in the LO-criticality mode, we know $t_a \leq t_r + D_i^{\text{LO}}$ where t_r is the release time of J_i. On the other hand we have $t_b = t_r + D_i^{\text{HI}}$. So it holds

$$t_b - t_a \geq D_i^{\text{HI}} - D_i^{\text{LO}} \tag{4}$$

Combining (3) and (4) we have $y \bmod T_i < D_i^{\text{HI}} - D_i^{\text{LO}} \Rightarrow t_a < t_s$ ∎

Lemma 4.3: Let $\mathcal{S}$ be defined as

$$\mathcal{S} = \min_{\tau_i \in \tau(\text{HI})} \{D_i^{\text{HI}} - D_i^{\text{LO}}\}$$

then $m_i(x, y)$ is bounded by $q_i(x, y)$:

$$q_i(x, y) = \max\left(\min\left(\left\lceil \frac{y - \mathcal{S}}{T_i} \right\rceil, n_i(x)\right), n_i(y)\right). \tag{5}$$

Proof: Assume τ_s is the task in $\tau(\text{HI})$ that triggers the criticality mode switch, i.e., a job J_s of τ_s has executed for its LO-criticality WCET without signalling completion. We let d_s be J_s's absolute deadline. If $d_s > t_d$, then any job with deadline in $[t_1, t_d]$ has higher priority than J_s, so only jobs released after the criticality mode switch point t_s (the normal jobs) can execute in HI-criticality mode. The number of normal jobs, i.e., jobs with both released time and deadline in $[t_s, t_d]$ is simply bounded by $n_i(y)$. In the following we focus on the case of $d_s \leq t_d$.

Again, we consider the worst-case release pattern of task τ_i where its jobs are released as fast as possible and one of its job's deadline aligns with t_d. We use J_c to denote the crossing job, and let d_c be the crossing job's absolute deadline.

If $d_c < d_s$, as illustrated in Figure 4-(a), the crossing job has priority higher than J_s, and must have been finished when J_s is executing at t_s. Therefore, this crossing job cannot execute in HI-criticality mode and thus should be excluded from $m_i(x, y)$. Otherwise, as illustrated in Figure 4-(b), the crossing job may execute after J_s, and cannot be excluded from $m_i(x, y)$.

By the above discussions, we know that a job in $[t_1, t_d]$ can execute in HI-criticality mode if it holds any of the following two conditions: (1) its deadline is in $[d_s, t_d]$; (2) it is a normal job. The number of jobs satisfying condition (1) is bounded by $\lceil (t_d - d_s)/T_i \rceil$, and can be further tightened by $n_i(x)$ to exclude the case that the crossing job is released before t_1. In summary, the number of jobs satisfying (1) is bounded by $\min(\lceil (t_d - d_s)/T_i \rceil, n_i(x))$. The number of jobs satisfying condition (2) (i.e., the normal jobs) is bounded by $n_i(y)$.

Summarizing the above discussions, we conclude that

$$m_i(x, y) \leq \max\left(\min\left(\left\lceil \frac{t_d - d_s}{T_i} \right\rceil, n_i(x)\right), n_i(y)\right). \tag{6}$$

The last step of the proof is to find an upper bound for $t_d - d_s$. Since J_s triggers criticality mode switch at t_s, i.e., J_s has executed for C_i^{LO} at t_s, so the distance between t_s and its absolute deadline is no smaller than $D_i^{\text{HI}} - D_i^{\text{LO}}$, i.e.,

$$\begin{aligned} & d_s - t_s \geq D_i^{\text{HI}} - D_i^{\text{LO}} \\ \Leftrightarrow \quad & t_d - d_s \leq y - (D_i^{\text{HI}} - D_i^{\text{LO}}) \\ \Rightarrow \quad & t_d - d_s \leq y - \mathcal{S} \end{aligned}$$

Applying this to (6) completes the proof. ∎

Using the upper bounds for $m_i(x, y)$ in the above lemmas, we can establish the schedulability test condition for a dual-criticality task system τ:

Theorem 4.4: The dual-criticality task system τ is schedulable if it holds

$$\forall x \geq y \geq 0 : \sum_{\tau_i \in \tau(\text{LO})} \mathsf{dbf}_i^{\text{LO}}(x-y) + \sum_{\tau_i \in \tau(\text{HI})} \mathsf{dbf}_i^{\text{HI}}(x, y) \leq x \tag{7}$$

where $\mathsf{dbf}_i^{\text{LO}}(x - y)$ is calculated by (1) and $\mathsf{dbf}_i^{\text{HI}}(x, y)$ is calculated by (2) with

$$m_i(x, y) = \min(p_i(x, y), q_i(x, y)) \tag{8}$$

Proof: By the definition of $\mathsf{dbf}_i^{\text{HI}}(x, y)$, $\mathsf{dbf}_i^{\text{HI}}(x, y) = \mathsf{dbf}_i^{\text{LO}}(x)$ when $y = 0$, so condition (7) can be rewritten as

$$\forall x \geq 0 : \sum_{\tau_i \in \tau} \mathsf{dbf}_i^{\text{LO}}(x) \leq x.$$

[1]This leads to the worst-case scenario for upper-bounding $m_i(x, y)$, which can be proved by comparing the bound obtained under this particular pattern with that obtained by "shifting" the release times to earlier or later time points. Details are omitted due to space limit.

This is exactly the standard EDF schedulability test formula based on demand bound function [7], which implies the schedulability of the whole task system in the LO-criticality mode.

In the following we will prove the HI-criticality tasks all meet their deadlines in the HI-criticality model. We prove by contradiction, assuming J_i is the first job missing its deadline at time t_d. We reuse the definition of t_1 and t_s as defined earlier in this section. There are two cases regarding the order between t_1 and t_s: (1) $t_1 > t_s$, (2) $t_1 \leq t_s$.

We first consider case (1). Since $t_1 > t_s$, all LO-criticality tasks have been aborted by time t_1. So only jobs of HI-criticality tasks with release times and absolute deadlines in $[t_1, t_d]$ can execute in $[t_1, t_d]$. The total workload of such jobs of task τ_j is bounded by

$$W_j = n_j(t_d - t_1) \times C_j^{\text{HI}}.$$

Since J_i misses its deadline, we know

$$\sum_{\tau_j \in \tau(\text{HI})} W_j > t_d - t_1.$$

On the other hand, by the definition of $p_i(x, y)$ we have

$$n_j(t_d - t_1) \leq p_j(t_d - t_1, t_d - t_1)$$

so W_j is bounded by $\mathsf{dbf}_j^{\text{HI}}(t_d - t_1, t_d - t_1)$. Putting these together we have

$$\sum_{\tau_j \in \tau(\text{HI})} \mathsf{dbf}_j^{\text{HI}}(t_d - t_1, t_d - t_1) > t_d - t_1$$

which contradicts condition (7).

Now we consider case (2). Since a LO-criticality task is aborted at time t_s, only jobs that are released in $[t_1, t_s)$ can execute in $[t_1, t_d]$. So the workload of a LO-criticality task τ_j in $[t_1, t_d]$ is bounded by $n_j(t_s - t_1) \times C_j^{\text{LO}}$. For each HI-criticality task, the number of its jobs with deadline no later than t_d is bounded by $n_j(t_d - t_1)$. By Lemma 4.2 and 4.3, among these jobs, the number of those executing for C_j^{HI} is bounded by $m_j(t_d - t_1, t_d - t_s)$ as defined by (8), so the total workload of jobs with both release times and deadlines in $[t_1, t_d]$ is bounded by $\mathsf{dbf}_i^{\text{HI}}(t_d - t_1, t_d - t_s)$. And since J_i misses its deadline at t_d, so the total workload of jobs with release times and deadlines in $[t_1, t_d]$ of the whole task system is strictly larger than $t_d - t_1$.

$$\sum_{\tau_i \in \tau(\text{LO})} \mathsf{dbf}_i^{\text{LO}}(t_s - t_1) + \sum_{\tau_i \in \tau(\text{HI})} \mathsf{dbf}_i^{\text{HI}}(t_d - t_1, t_d - t_s) > t_d - t_1$$

which contradicts (7). In summary, both cases lead to contradictions, and so the assumption that a HI-criticality job misses deadline cannot be true. ■

Later in Section IV-C we will discuss the range of x and y values to be checked in test condition (7).

B. Multi-Criticality Systems

This subsection generalizes the schedulability test introduced above to the case of more than two criticality levels.

We generalize $\mathsf{dbf}_i^{\text{HI}}(x, y)$ to $\mathsf{dbf}_i^l(x_1, x_2, \cdots, x_l)$ to bound the total demand of all jobs of task τ_i in time interval $[t_1, t_d]$. $\mathsf{dbf}_i^l(x_1, x_2, \cdots, x_l)$ is calculated by

$$\mathsf{dbf}_i^l(x_1, x_2, \cdots, x_l) = \sum_{j=1}^{l} N_i^j \times C_i^j \tag{9}$$

where N_i^j is iteratively computed by

$$N_i^j = \max(\overline{N_i^j} - \overline{N_i^{j+1}}, 0)$$

with $\overline{N_i^{l+1}} = 0$ and

$$\overline{N_i^1} = n_i(x_1)$$
$$\overline{N_i^j} = \min(p_i^j(x_1, x_j), q_i^j(x_1, x_j))$$

$$p_i^j(x_1, x_j) = \begin{cases} n_i(x_j), \\ \qquad \text{if } \ x_j \bmod T_i < D_i^j - D_i^{j-1} \\ \min(n_i(x_j) + 1, n_i(x_1)), \quad \text{otherwise} \end{cases}$$

$$q_i^j(x_1, x_j) = \max\left(\min\left(\left\lceil \frac{x_j - \mathcal{S}^j}{T_i} \right\rceil, n_i(x_1)\right), n_i(x_j)\right).$$

$$\mathcal{S}^j = \min_{\tau_i \in \bigcup_{l \geq j} \tau(l)} \{D_i^j - D_i^{j-1}\}$$

Before stating and proving the schedulability test condition, we first introduce an auxiliary lemma:

Lemma 4.5: Given three sequences of number $\{\mu_1, \mu_2, \cdots, \mu_K\}$, $\{N_1, N_2, \cdots, N_K\}$ and $\{C_1, C_2, \cdots, C_K\}$ satisfying the following conditions:

1) $\forall 1 \leq k \leq K: \ \sum_{l=k}^{K} \mu_l \leq \sum_{l=k}^{K} N_l$
2) $C_1 \leq C_2 \leq \cdots \leq C_K$

then it holds:

$$\sum_{l=1}^{K} \mu_l \times C_l \leq \sum_{l=1}^{K} N_l \times C_l \tag{10}$$

Proof: We prove the lemma by induction.

Base Case: When $l = K$, as $\sum_{l=K}^{K} \mu_l \leq \sum_{l=K}^{K} N_l$, i.e.,

$$\mu_K \leq N_K \ \Rightarrow \ \mu_K \times C_K \leq N_K \times C_K$$

which satisfies (10).

Inductive Step: Given the inductive hypothesis that (10) holds for $l = k+1, k+2, \cdots, K$, we shall prove that (10) still holds for $l = k$. Consider two cases:

(a) $\mu_k \leq N_k$:

$$\sum_{l=k+1}^{K} \mu_l \times C_l \leq \sum_{l=k+1}^{K} N_l \times C_l$$

$$\Rightarrow \ \mu_k \times C_k + \sum_{l=k+1}^{K} \mu_l \times C_l \leq N_k \times C_k + \sum_{l=k+1}^{K} N_l \times C_l$$

$$\Leftrightarrow \ \sum_{l=k}^{K} \mu_l \times C_l \leq \sum_{l=k}^{K} N_l \times C_l$$

which satisfies (10).

(b) $\mu_k > N_k$: Let $\lambda_l = N_l - \mu_l$, then $\lambda_k < 0$. We first claim $\sum_{l=k+1}^{K} \lambda_l > 0$. This is because, if this is not true: $\sum_{l=k+1}^{K} \mu_l \geq \sum_{l=k+1}^{K} N_l$ and then

$$\sum_{l=k+1}^{K} \mu_l \geq \sum_{l=k+1}^{K} N_l \Rightarrow \mu_k + \sum_{l=k+1}^{K} \mu_l > N_k + \sum_{l=k+1}^{K} N_l$$
$$\Leftrightarrow \sum_{l=k}^{K} \mu_l > \sum_{l=k}^{K} N_l$$

which contradicts the condition 1) in the lemma.

So we have $\sum_{l=k+1}^{K} \lambda_l > 0$, and by $\sum_{l=k}^{K} \mu_l \leq \sum_{l=k}^{K} N_l$ we have:

$$\sum_{l=k}^{K} \lambda_l = \lambda_k + \sum_{l=k+1}^{K} \lambda_l \geq 0$$

and since $C_k \leq C_{k+1} \leq \cdots \leq C_K$, we know

$$\lambda_k \times C_k + \sum_{l=k+1}^{K} \lambda_l \times C_l \geq 0$$
$$\Leftrightarrow N_k \times C_k + \sum_{l=k+1}^{K} N_l \times C_l \geq \mu_k \times C_k + \sum_{l=k+1}^{K} \mu_l \times C_l$$
$$\Leftrightarrow \sum_{l=k}^{K} \mu_l \times C_l \leq \sum_{l=k}^{K} N_l \times C_l$$

which satisifies (10). ∎

Now we are ready to establish the schedulability test condition for a task set τ with L criticality levels:

Theorem 4.6: A mixed-criticality task set τ of L criticality levels is schedulable by EDF if it holds:

$$\forall x_1 \geq ... \geq x_L \geq 0 : \sum_{l=1}^{L} \sum_{\tau_i \in \tau(l)} \mathsf{dbf}_i^l(x_1, \cdots, x_l) \leq x_1 \tag{11}$$

where $\mathsf{dbf}_i^l(x_1, \cdots, x_l)$ is calculated by (9)

Proof: We prove the theorem by contradiction, assuming a job J_i of criticality l misses its deadline at time t_d. Let t_l be the time point when the system criticality switches from mode $l-1$ to l. t_b is the earliest time point no later than t_d such that at any time point in $[t_b, t_d]$ $\tau(l)$ has at least one active job (released but not finished yet) with deadline no later than t_d. If we divide the time interval $[t_1, t_d)$ into segments $[t_1, t_2), [t_2, t_3), \cdots, [t_\beta, t_d)$, and we let $t_b \in [t_\alpha, t_{\alpha+1})$[2]. We let $x_l = t_d - t_l$. In the following, we shall prove that the workload of jobs of a task τ_j in $\tau(\gamma)$ with both release times and absolute deadlines in time interval $[t_1, t_d]$ is bounded by $\Delta = \mathsf{dbf}_i^\beta(x_1, ..., x_\beta)$ where $x_1 = x_2 = \cdots = x_\alpha$ and $x_\gamma = x_{\gamma+1} = \cdots = x_\beta$. If this is true, it must hold $\Delta > x_1$, which contradicts (11). We assume at runtime the workload of a task in $\tau(\gamma)$ in time interval $[t_1, t_d]$ is greater than Δ. We assume at runtime the number of jobs of task τ_j with both release times and deadlines in $[t_1, t_d]$ that execute for more than C_i^l but no more than C_i^{l+1} is μ_i^{l+1}. Then the total workload of $\tau(\gamma)$ is bounded by $\sum_{l=\alpha}^{\gamma} \mu_i^l \times C_i^l$. On the other hand, by the definition of dbf_i^l, $\Delta = \sum_{l=\alpha}^{\gamma} N_i^l \times C_i^l$. So we have

$$\sum_{l=\alpha}^{\gamma} \mu_i^l \times C_i^l > \sum_{l=\alpha}^{\gamma} N_i^l \times C_i^l \tag{12}$$

We let $\mu = \sum_{l=\alpha}^{\gamma} \mu_i^l$ and $N = \sum_{l=\alpha}^{\gamma} N_i^l$, then one of the following two conditions must be true in order to satisfy (12): In the following we will prove for both cases (i) $\mu > N$ and (ii)$\mu \leq N$ there are contradictions.

(i)$\mu > N$. By the definition of N_i^l, we have $N = n_i(x_1)$, which is the maximal number of jobs of task τ_i with both release times and deadlines in a time interval of length x_1. It is obvious that $\mu > N$ cannot be true.

(ii) $\mu \leq N$. Let δ be the largest index such that $\mu_i^\delta \neq N_i^\delta$. Consider two cases: (1) $\mu_i^\delta > N_i^\delta$: In this case, $\overline{\mu_i^\delta} = \mu_i^\delta + \overline{\mu_i^{\delta+1}}$ and $\overline{N_i^\delta} = N_i^\delta + \overline{N_i^{\delta+1}}$. By the definition of δ, for any $j > \delta, \mu_i^j = N_i^j$ then $\overline{\mu_i^{\delta+1}} = \overline{N_i^{\delta+1}}$. So we have $\overline{\mu_i^\delta} > \overline{N_i^\delta}$. Using the same agrement as in the proof of Lemma 4.2 and 4.3, $p_i^\delta(x_1, x_\delta)$ and $q_i^\delta(x_1, x_\delta)$ both bound the number of jobs in which τ_i execute in mode δ of higher, so $\overline{N_i^\delta} = \min(p_i^\delta(x_1, x_\delta), q_i^\delta(x_1, x_\delta))$ is a safe upper bound of the number of jobs in which τ_i execute in mode δ of higher. So we know $\overline{\mu_i^\delta} > \overline{N_i^\delta}$ cannot be true, and thus $\mu_i^\delta > N_i^\delta$ is also not true.

(2)$\mu_i^\delta < N_i^\delta$: By lemma 4.5, we have

$$\sum_{l=\alpha}^{\gamma} \mu_i^l \times C_i^l \leq \sum_{l=\alpha}^{\gamma} N_i^l \times C_i^l$$

which contradicts (12).

In summary, both cases lead to contradictions to (12), and the assumption that a job J_i of criticality l misses its deadline cannot be true.

∎

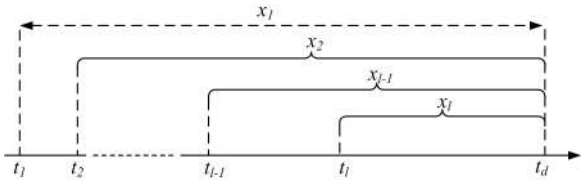


Fig. 5. Illustration of the notations

C. Complexity and Efficiency

We first analyze the complexity of our new analysis method for dual-criticality systems, i.e., the schedulability test in (7). We first define some notations:

$$\mathcal{U}_{\text{LO}}^{\text{LO}} = \sum_{\tau_i \in \tau(\text{LO})} U_i^{\text{LO}}, \ \mathcal{U}_{\text{HI}}^{\text{LO}} = \sum_{\tau_i \in \tau(\text{HI})} U_i^{\text{LO}}, \quad \mathcal{U}_{\text{HI}}^{\text{HI}} = \sum_{\tau_i \in \tau(\text{HI})} U_i^{\text{HI}}$$
$$\mathcal{C}_{\text{LO}}^{\text{LO}} = \sum_{\tau_i \in \tau(\text{LO})} C_i^{\text{LO}}, \ \mathcal{C}_{\text{HI}}^{\text{LO}} = \sum_{\tau_i \in \tau(\text{HI})} C_i^{\text{LO}}, \quad \mathcal{C}_{\text{HI}}^{\text{HI}} = \sum_{\tau_i \in \tau(\text{HI})} C_i^{\text{HI}}$$

As a common assumption in scheduling of sporadic tasks, we assume the total utilization is bounded by a

[2]We omit the proof for the easy case $t_b \in [t_\beta, t_d)$, where no criticality switch occurs during the busy period.

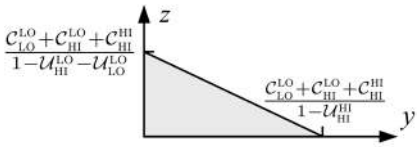


Fig. 6. Illustration of the region of (y, z) that needs to be tested

constant strictly smaller than the long-term proportion of the allocated resource, i.e., $\mathcal{U}_{\rm HI}^{\rm LO} + \mathcal{U}_{\rm LO}^{\rm LO} \leq \epsilon < 1$ and $\mathcal{U}_{\rm HI}^{\rm HI} \leq \epsilon < 1$. The following gives the range of x and y values to be checked in (7).

Theorem 4.7: Let $x = y + z$. (7) must hold if no (y, z) satisfying the following condition violates it:

$$(1-\mathcal{U}_{\rm HI}^{\rm HI})\times y+(1-\mathcal{U}_{\rm HI}^{\rm LO}-\mathcal{U}_{\rm LO}^{\rm LO})\times z \leq \mathcal{C}_{\rm LO}^{\rm LO}+\mathcal{C}_{\rm HI}^{\rm LO}+\mathcal{C}_{\rm HI}^{\rm HI} \quad (13)$$

Proof: A job executes in LO-mode must have its release time in $[t_1, t_s)$, so the total workload of such jobs of τ_i is no larger than $(\lceil \frac{t_s-t_1}{T_i} \rceil) \times C_i^{\rm LO}$, which is bounded by $z \times U_i^{\rm LO} + C_i^{\rm LO}$. A job executes in HI-mode must have its absolute deadline in $(t_s, t_d]$, so the total workload of such jobs of τ_i is no larger than $(\lceil \frac{t_d-t_s}{T_i} \rceil) \times C_i^{\rm HI}$, which is bounded by $y \times U_i^{\rm HI} + C_i^{\rm HI}$. So we know

$$\sum_{\tau_i \in \tau({\rm LO})} \mathsf{dbf}_i^{\rm LO}(z) + \sum_{\tau_i \in \tau({\rm HI})} \mathsf{dbf}_i^{\rm HI}(y+z, y) \leq$$
$$y \times \mathcal{U}_{\rm HI}^{\rm HI} + z \times (\mathcal{U}_{\rm HI}^{\rm LO} + \mathcal{U}_{\rm LO}^{\rm LO}) + \mathcal{C}_{\rm LO}^{\rm LO} + \mathcal{C}_{\rm HI}^{\rm LO} + \mathcal{C}_{\rm HI}^{\rm HI}$$

Since (7) is violated, i.e., there exists y, z such that

$$\sum_{\tau_i \in \tau({\rm LO})} \mathsf{dbf}_i^{\rm LO}(z) + \sum_{\tau_i \in \tau({\rm HI})} \mathsf{dbf}_i^{\rm HI}(y+z, y) \geq y + z$$
$$\Rightarrow \; y \times \mathcal{U}_{\rm HI}^{\rm HI} + z \times (\mathcal{U}_{\rm HI}^{\rm LO} + \mathcal{U}_{\rm LO}^{\rm LO}) + \mathcal{C}_{\rm LO}^{\rm LO} + \mathcal{C}_{\rm HI}^{\rm LO} + \mathcal{C}_{\rm HI}^{\rm HI} \geq y + z$$

by which the theorem is proved. ■

The region of y and z to be checked is illustrated in Figure 6. By examining the definitions, we can see that $\mathsf{dbf}_i^{\rm HI}(x, y)$ is a stair-case function non-decreasing with respect to both y and z. Therefore, it is sufficient to check (7) only with discrete points in the region of Figure 6 at which $\mathsf{dbf}_i^{\rm HI}(x, y)$ is non-differentiable, and the overall complexity of the schedulability test is pseudo-polynomial.

In the same way, we can get the conclusion for multi-criticality systems with following notations:

$$\forall j \leq l: \; \mathcal{U}_l^j = \sum_{\tau_i \in \tau(l)} U_i^j = \sum_{\tau_i \in \tau(l)} C_i^j / T_i^j,$$
$$\forall j \leq l: \; \mathcal{C}_l^j = \sum_{\tau_i \in \tau(l)} C_i^j$$

Theorem 4.8: Let $z_j = x_j - x_{j+1}$ and $z_L = x_L$. (11) must be true if all $\{z_1, \cdots, z_L\}$ satisfying the following condition does not violates (11):

$$\sum_{j=1}^{l}(1 - \sum_{i=j}^{l} \mathcal{U}_i^j) \times z_j \leq \sum_{j=1}^{l}(\sum_{i=j}^{l} \mathcal{C}_i^j) \quad (14)$$

The proof of the theorem is similar to that of theorem 4.7, and is omitted due to space limit.

While for dual-criticality systems the region of the y and z values to be checked is a two-dimensional area, the values of $\{z_1, \cdots, z_L\}$ to be checked forms an L-dimensional space. Given that L is a constant, the computational complexity is still pseudo-polynomial, but in practise may be very time consuming for systems with a large L.

To solve this problem, we can use a combination of the schedulability test in (11) and the EY analysis to balance the precision and efficiency of the analysis. More specifically, we set a certain limit ζ for the number of criticality modes to be covered by our analysis. For example, consider a task system with 4 criticality levels, and we set $\zeta = 2$. Then we analyze the schedulability of the first two criticality levels and last two criticality levels separately. When we analyze the last two criticality levels, we shall bound workload of the carry-over jobs from the 2^{nd} criticality level to the 3^{rd} criticality level as in the EY analysis.

If we set $\zeta = 3$ for a system with 5 criticality levels, then there are two options to group the criticality levels: (i) $\{1, 2\}$ and $\{3, 4, 5\}$, (ii) $\{1, 2, 3\}$ and $\{4, 5\}$. The performance comparison of different options will be evaluated and discussed in Section VI.

V. Tuning Virtual Deadlines

In the previous sections we presented the new analysis technique for EDF-VD scheduled mixed-criticality systems using different "virtual" deadlines for a task when the system executes in different criticality modes. In this section we present a new algorithm to choose suitable virtual deadlines for each task on different criticality levels.

An important observation in [11], [12] is that, shortening the relative deadline for a task on criticality level l is potentially beneficial to the schedulability of criticality levels $l+1, l+2, \cdots$, at the price of decreasing the schedulability of criticality level l. Therefore, the major problem to solve in tuning the virtual deadlines is to balance the schedulability on different criticality levels.

In [11], [12], EY separates the analysis of different criticality levels and approximate the workload of criticality mode j by considering the workload carried from its preceding mode $j-1$. Because of this, [11], [12] designed strategy to tune the relative deadlines from the highest to lowest criticality level in order: on each criticality level j, it tunes the relative deadlines in its preceding criticality level $j-1$, to reduce the carry-over workload from criticality $j-1$ as much as possible until the criticality level j is guaranteed to be schedulable.

Different with EY, dbf_i^j computed by our new analysis method presented in this paper contains workload crossed multiple the criticality modes lower than j. So we have no idea about on which criticality level shall we shorten the relative deadlines, to improve the schedulability of the currently considered criticality level.

The key idea of our new relative deadline tuning strategy is to design an effective guidance to choose the criticality level in which we shall shorten the relative deadline of

some task in each step of the tuning procedure. Again, shortening a relative deadline on criticality level j may hurt the schedulability of this criticality level itself, so we shall choose the criticality level that are "easier" to be schedulable, i.e., the one with lower workload. We use the metric $load(j)$ to represent the workload of criticality level j contributed by all the tasks with criticality no lower than j:

Definition 5.1 (load): Let load(j) be defined as

$$load(j) = \max_{\forall x>0} \frac{\sum_{\tau_i \in \bigcup_{k \geq j} \tau(j)} \mathsf{dbf}_i^j(x)}{x}$$

where $\mathsf{dbf}_i^j(x)$ is the ordinary demand bound function [7] using C_i^j as the worst-case execution time.

With the metric *load* we design a new deadline tuning algorithm, TuneSys(τ), the pseudo-code of which is shown in Algorithm 1. TuneSys(τ) checks the schedulability of all L modes in a decreasing ciriticality order. So at each step, we know that all succeeding modes turned so far are schedulable as long as we can successfully tune the remaining modes. As soon as TuneSys(τ) finds τ is not schedulable in mode l it selects mode j which has the minimal load from all modes lower than l and applies TuneMode(j). After that it recalculates load(j) and repeats the tuning above until τ is schedulable in the current modes. The algorithm fails if in a certain mode it is not possible to shorten the deadlines in any of the lower criticality levels without violating their schedulability, which is determined by a fast load-based test condition ($load(j) \geq 1$). The algorithm terminates with SUCCESS if and only if τ is schedulable on all modes according to (11). The routine TuneMode(j) randomly select one task and shorten its relative deadlines in mode j by one time unit.

```
for l ∈ {L, ··· , 1} do
  while τ is not schedulable in mode l do
    ∀j < l, calculate load(j)
    if ∀j < l, load(j)≥ 1 then
      return FAILURE
    end if
    select j such that load(j) is minimal for ∀j < l
    TuneMode(j)
  end while
end for
return SUCCESS
```

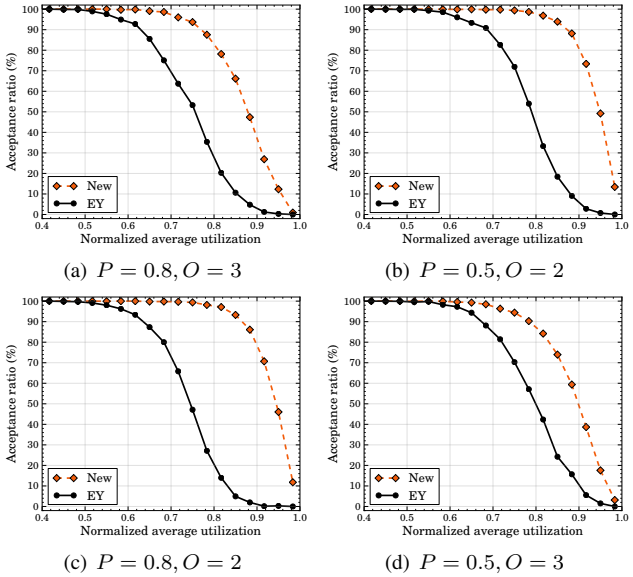
Algorithm 1: TuneSys(τ)

VI. Experimental Evaluation

In this section we conduct experiments to evaluate the performance of the new analysis method presented in this paper, for task systems with *more than two criticality levels* in both its precision and efficiency.

We follow the approach in [11], [12] to generate random mixed-criticality task sets: P is the probability for a task to be a higher-criticality task (if τ_i is a j-criticality task, then the probability for the next generated task τ_{i+1} to be a $(j+1)$ criticality task is P), and O is the maximal ratio between the execution time parameter of two consecutive criticality modes. L is the total number of criticality levels. For each task τ_i, C_i^1 is uniformly distributed over $[1, 10]$, T_i is uniformly distributed over $[10, 100]$, and $D_i = T_i$. Task sets are randomly generated and grouped with different scope of their average utilization $\mathcal{U}_{avg}$, which is defined as

$$\mathcal{U}_{avg} = \sum_{j=1}^{L} \mathcal{U}^j / L, \qquad \mathcal{U}^j = \sum_{\tau_i \in \tau} U_i^j$$



(a) $P = 0.8, O = 3$ (b) $P = 0.5, O = 2$

(c) $P = 0.8, O = 2$ (d) $P = 0.5, O = 3$

Fig. 7. Experiment results with different P and O for triple-criticality task systems

We compare the following two schedulability tests:

- **New**. The schedulability test in Theorem 4.6 with the tuning approach presented in Section V.
- **EY**. The analysis presented in [12] which separates the analysis of different criticality levels.

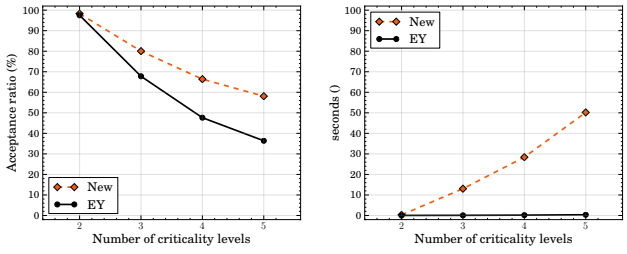
We first evaluate the analysis precision improvement of New over EY. Figure 7 shows the *acceptance ratio* as a function of the average utilization for task sets with three criticality levels ($L = 3$) and different combinations of P and O values. For each utilization range at least 10000 task sets are generated and tested, and the acceptance ratio is defined by the ratio between the number of task sets that is determined to be schedulable by a certain test and the total number of generated task sets. The results in Figure 7 show that our new analysis can significantly improve the analysis precision over EY with different settings.

We also evaluate the analysis precision improvement of New for systems with more ciriticality levels. In Figure 8-(a), we plot the weighted acceptance ratio as a function of the number of criticality levels L varying from 2 to 5. We can see that as the number of criticality levels increases, it becomes more difficult to guarantee the schedulability of

the system, but at the same time the precision improvement of New over EY is more significant.

The precision improvement of the New analysis comes at the price of a lower analysis efficiency. Figure 8-(b) shows the average time consumption for analyzing each randomly generated task set. We can see that the New analysis becomes more and more expensive as the number of criticality levels increases, while the efficiency degradation of EY is much slower. For task systems with 5 criticality levels, the analysis of each task set on average takes 50 seconds. This maybe unacceptable for larger-scale systems, especially those with a large number of criticality levels.

To solve the efficiency problem of the New analysis, we proposed to use a hybrid analysis technique combining the New analysis and the EY analysis, as introduced in the end of Section IV-C. The hybrid analysis balances the analysis precision and efficiency by setting a upper limit ζ for the number of criticality levels to be covered in the analysis. Figure 9 shows the analysis precision of the hybrid analysis with $\zeta = 3$ for task systems with $L = 5$. The curve "New" is the original New analysis that covers all the five criticality levels, the curve "EY" is the EY analysis. The curves "New$_{23}$" and "New$_{32}$" are the hybrid analysis with two different criticality grouping strategies $\{1, 2\}+\{3, 4, 5\}$ and $\{1, 2, 3\} + \{4, 5\}$, respectively. From the experiments we see that in general the two strategies are incomparable (i.e., there exist task sets schedulable under one strategy but unscheduale under the other one, and vice versa), but on average the strategy grouping more criticality levels at lower criticality levels ("New$_{32}$" in this experiment) is more precise than the other option ("New$_{23}$" in this experiment).



(a) acceptance ratio (b) time overhead

Fig. 8. Analysis precision and efficiency, with respect to the number of criticality levels.

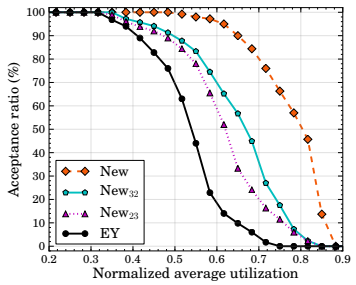


Fig. 9. Evaluation of the hybrid analysis method with different criticality grouping strategies. $L = 5$, $P = 0.8$, $O = 3$.

VII. Conclusion

In this paper, we present new schedulability analysis methods for mixed-criticality sporadic task systems scheduled by EDF-VD. The new analysis improves precision over the state-of-the-art technique EY [11], [12] for systems with multiple criticality levels (more than two levels). The performance improvement of our new analysis comes at the price of higher complexity. In order to solve the scalability problem we also proposed to use a hybrid method combining our new analysis technique and state-of-the-art EY analysis, to balance the precision and efficiency. We use randomly generated task systems to evaluate the performance of the new analysis method. Experiment results show that the new method indeed significantly improve the analysis precision over the state-of-the-art.

As the future work, we will investigate heuristic algorithms to automatically decide how to group the criticality levels in the hybrid approach. We also plan to study how to apply the new analysis in this paper to the partitioned multiprocessor scheduling problem. The challenge is to develop an efficient partitioning strategy without iteratively invoking the expensive schedulability test.

References

[1] S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, N. Megow, and L. Stougie. Scheduling Real-Time Mixed-Criticality Jobs. In *MFCS*, 2010.

[2] 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 *ESA*, 2011.

[3] S. Baruah, A. Burns, and R. I. Davis. Response-time analysis for mixed criticality systems. In *RTSS*, 2011.

[4] S. Baruah, H. Li, and L. Stougie. Towards the Design of Certifiable Mixed-criticality Systems. In *RTAS*, 2010.

[5] S. Baruah and S. Vestal. Schedulability analysis of sporadic tasks with multiple criticality specifications. In *ECRTS*, 2008.

[6] S. K. Baruah, B. Chattopadhyay, H. Li, and I. Shin. Mixed-criticality scheduling on multiprocessors. In *Real-Time Systems*, 2014.

[7] S. K. Baruah, A. Mok, and L. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proceedings of the 11th Real-Time Systems Symposium (RTSS)*, 1990.

[8] R. Davis and A. Burns. Mixed criticality systems - a review. In *Technical Report, University of York*, 2013.

[9] F. Dorin, P. Richard, M. Richard, and J. Goossens. Schedulability and sensitivity analysis of multiple criticality tasks with fixed-priorities. In *Real-Time Systems*, 2010.

[10] A. Easwaran. Demand-based mixed-criticality scheduling of sporadic tasks on one processor. In *RTSS*, 2013.

[11] P. Ekberg and W. Yi. Bounding and shaping the demand of mixed-criticality sporadic tasks. In *ECRTS*, 2012.

[12] P. Ekberg and W. Yi. Bounding and shaping the demand of generalized mixed-criticality sporadic task systems. In *Real-Time Systems*, 2014.

[13] C. Gu, N. Guan, Q. Deng, and W. Yi. Improving ocbp-based scheduling for mixed-criticality sporadic task systems. In *RTCSA*, 2013.

[14] C. Gu, N. Guan, Q. Deng, and W. Yi. Partitioned scheduling of mixed-criticality task systems on multiprocessors. In *DATE*, 2014.

[15] N. Guan, P. Ekberg, M. Stigge, and W. Yi. Effective and Efficient Scheduling of Certifiable Mixed-Criticality Sporadic Task Systems. In *RTSS*, 2011.

[16] S. K. B. Haohan Li. Global mixed-criticality scheduling on multiprocessors. In *ECRTS*, 2012.

[17] H. Li and S. Baruah. An algorithm for scheduling certifiable mixed-criticality sporadic task systems. In *RTSS*, 2010.

[18] S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *RTSS*, 2007.