Utilization-Based Scheduling of Flexible Mixed-Criticality Real-Time Tasks

Gang Chen, Nan Guan, Di Liu, Qingqiang He, Kai Huang, Todor Stefanov, and Wang Yi

Abstract—Mixed-criticality models are an emerging paradigm for the design of real-time systems because of their significantly improved resource efficiency. However, formal mixed-criticality models have traditionally been characterized by two impractical assumptions: once *any* high-criticality task overruns, *all* low-criticality tasks are suspended and *all other* high-criticality tasks are assumed to exhibit high-criticality behaviors at the same time. In this paper, we propose a more realistic mixed-criticality model, called the flexible mixed-criticality (FMC) model, in which these two issues are addressed in a combined manner. In this new model, only the overrun task itself is assumed to exhibit high-criticality behavior, while other high-criticality tasks remain in the same mode as before. The guaranteed service levels of low-criticality tasks are gracefully degraded with the overruns of high-criticality tasks. We derive a utilization-based technique to analyze the schedulability of this new mixed-criticality model under EDF-VD scheduling. During run time, the proposed test condition serves an important criterion for dynamic service level tuning, by means of which the maximum available execution budget for low-criticality tasks can be directly determined with minimal overhead while guaranteeing mixed-criticality schedulability. Experiments demonstrate the effectiveness of the FMC scheme compared with state-of-the-art techniques.

Index Terms—EDF-VD scheduling, flexible mixed-criticality system, utilization-based analysis

1 Introduction

A mixed-criticality (MC) system is a system in which tasks with different criticality levels share a computing platform. In MC systems, different degrees of assurance must be provided for tasks with different criticality levels. To improve resource efficiency, MC systems [26] often specify different WCETs for each task at all existing criticality levels, with those at higher criticality levels being more pessimistic. Normally, tasks are scheduled with less pessimistic WCETs for resource efficiency. Only when the less pessimistic WCET is violated, the system switches to the high-criticality mode and *only* tasks with higher criticality levels are guaranteed to be scheduled with pessimistic WCETs thereafter.

There is a large body of research work on specifying and scheduling mixed-criticality systems (see [8] for a comprehensive review). However, to ensure the safety of high-criticality tasks, the classic MC model [1], [2], [3], [4], [5] applies conservative restrictions to the mode-switching scheme. In the classic MC model, whenever *any* high-criticality task overruns, *all* low-criticality tasks are immediately abandoned and *all other* high-criticality tasks are assumed to exhibit high-criticality behaviors. This mode-switching scheme is not realistic in the following two important respects.

- First, it is pessimistic to immediately abandon all low-criticality tasks, because low-criticality tasks require a certain timing performance as well [17], [25].
- Second, it is pessimistic to bind the mode switches of all high-criticality tasks together for the scenarios where the mode switches of high-criticality tasks are naturally independent [12], [22].

Although there has been some research on solving the first problem, i.e., *statically* reserving a certain degraded level of service for low-criticality execution [7], [16], [24], [25], to our knowledge, little work has been done to date to address the second problem.

In this paper, we propose a flexible MC model (denoted as **FMC**) on a uni-processor platform, in which the two aforementioned issues are addressed in a combined manner. In FMC, the mode switches of all high-criticality tasks are independent. A single high-criticality task that violates its low-criticality WCET triggers only itself into high-criticality mode, rather than triggering *all* high-criticality tasks. All other high-criticality tasks remain at their previous criticality levels and thus do not require to book additional resources at mode-switching points. In this manner, significant resources can be saved compared with the classic MC model [1],

- G. Chen is with the School of Computer Science and Engineering, Northeastern University, Shenyang 110004, China and with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China. E-mail: chengang@cse.neu.edu.cn.
- N. Guan and Q. He are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China. E-mail: {nan.guan, qianghe}@polyu.edu.hk.
- D. Liu is with the National Pilot School of Software, Yunnan University, Kunming 650221, China. E-mail: d.liu@liacs.leidenuniv.nl.
- K. Huang is with the Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education (Sun Yet-Sen University), Guangzhou 510275, China. E-mail: huangk36@mail.sysu.edu.cn.
- T. Stefanov is with Leiden Institute of Advanced Computer Science, Leiden University, Leiden, EZ 2311, The Netherlands. E-mail: t.p.stefanov@liacs.leidenuniv.nl.
- W. Yi is with Department of Information Technology, Uppsala University, Uppsala 752 36, Sweden and also with the School of Computer Science and Engineering, Northeastern University, Shenyang 110004, China. E-mail: yi@it.uu.se.

Manuscript received 9 Sept. 2016; revised 17 Apr. 2017; accepted 1 Oct. 2017. Date of publication 15 Oct. 2017; date of current version 19 Mar. 2018. (Corresponding author: Gang Chen.) Recommended for acceptance by J. D. Bruguera. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TC.2017.2763133

[2], [3]. On the other hand, these saved resources can be used by low-criticality tasks to improve their service quality. More importantly, the proposed FMC model adaptively tunes the service level for low-criticality tasks to compensate for the overrun of high-criticality tasks, thereby allowing the system workload to be balanced with minimal service degradation for low-criticality tasks. At each independent mode-switching point, the service level for low-criticality tasks is dynamically updated based on the overruns of high-criticality tasks. By doing so, the quality of service (QoS) for low-criticality tasks can be significantly improved.

Since the service level for low-criticality tasks is dynamically determined during run time, the decision-making procedure should be light-weighted. For this purpose, utilization-based scheduling is more desirable for run-time decision-making because of its simplicity. However, using utilization-based scheduling for our FMC model brings new challenges due to the intrinsic dynamics of this model, such as the service level tuning strategy. In particular, utilization-based schedulability analysis relies on whether the cumulative execution time of low-criticality tasks can be *effectively* upper bounded. In FMC, the service levels for low-criticality tasks are dynamically tuned at each mode switching point. Therefore, the cumulative execution time of low-criticality tasks strongly depends on when mode switches occur. In general, such information is difficult to explicitly represent prior to real execution, because the independence of the mode switches in FMC results in a large analysis space. It is computationally infeasible to analyze all possibilities. To resolve this challenge, we propose a novel approach based on mathematical induction, which allows the cumulative execution time of low-criticality tasks to be *effectively* upper bounded.

In this work, we study the schedulability of the proposed FMC model under EDF-VD scheduling. An utilization-based schedulability test condition is derived by integrating the independent triggering scheme and the adaptive service level tuning scheme. A formal proof of the correctness of this new schedulability test condition is presented. Based on this test condition, an EDF-VD-based MC scheduling algorithm, called FMC-EDF-VD, is proposed for the scheduling of an FMC task system. During run time, the optimal service level for low-criticality tasks can be directly determined via this condition with minimum overhead, and mixed-criticality schedulability can be simultaneously guaranteed. In addition, we explore the feasible region of the virtual deadline factor for FMC model. Simulation results show that FMC-EDF-VD provides benefits in supporting low-criticality execution compared with state-of-the-art algorithms.

2 Related Work

Mixed-criticality scheduling is a research field that has received considerable attention in recent years. As stated in [7], much existing research work [1], [2], [3] on MC scheduling makes the pessimistic assumption that all low-criticality tasks are immediately abandoned once the system enters high-criticality mode. Instead of abandoning all low-criticality tasks, some efforts [7], [16], [19], [24], [25] have been made to provide solutions for offering low-criticality tasks a certain degraded service quality when the system is in high-criticality mode. Nevertheless, these studies still use a pessimistic mode-switch triggering scheme in which, whenever one high-criticality task overruns, all other high-criticality tasks are triggered to exhibit high-criticality behavior and book unnecessary resources.

Recent work presented in [12], [15], [22] offers solutions for improving performance for low-criticality tasks by using different mode-switch triggering strategies. Huang et al. [15] proposed an interference constraint graph to specify the execution dependencies between high-criticality and low-criticality tasks. However, this approach still uses high-confidence WCET estimates for all high-criticality tasks when determining system schedulability, and therefore does not address the second problem discussed above. Gu et al. [12] presented a component-based strategy in which the component boundaries offer the isolation necessary to support the execution of low-criticality tasks. Minor overruns can be handled with an internal mode switch by dropping off all low-criticality jobs within a component. More extensive overruns will result in a system-wide external mode switch and the dropping off of all low-criticality jobs. Therefore, the mode switches at the internal and external levels still use pessimistic strategy in which all low-criticality tasks are abandoned once a mode switch occurs at the corresponding level. The two problems mentioned above still exist at both levels. In addition, the system schedulability is tested using a demand bound function (DBF) based approach. The complexity of the schedulability test is exponential in the size of the input [12], resulting in costly computations.

Ren and Phan [22] proposed a partitioned scheduling algorithm based on group-based Pfair-like scheduling [14] for mixed-criticality systems. Within a task group, a single high-criticality task is encapsulated with several low-criticality tasks. The tasks are scheduled via Pfair-like scheduling [14] by breaking them into quantum-length sub-tasks. Sub-tasks that belong to different groups are scheduled on an earliest-pseudo-deadline-first (EPDF) basis. Pfair scheduling is a well-known optimal scheduling method for scheduling periodic real-time tasks on a multiple-resource system. However, Pfair scheduling poses many practical problems [14]. First, the Pfair algorithm incurs very high scheduling overhead because of frequent preemptions caused by the small quantum lengths. Second, the task groups are explicitly required to be well synchronized and to make progress at a steady rate [27]. Therefore, the work presented in [22] strongly relies on the periodic task models. In addition, the system schedulability in [22] is determined by solving a MINLP problem, which in general has NP-hard complexity [11]. Because of this complexity, the scalability problem needs to be carefully considered.

Compared with the existing work [12], [22], the proposed FMC model and its scheduling techniques offer both *simplicity and flexibility*. In particular, our work differs from these approaches in the following respects. Compared with the Pfair-based scheduling method [22] which relies on periodic task models, our paper derives an EDF-VD-based scheduling scheme for sporadic mixed-criticality task systems, that incorporates an independent mode-switch triggering scheme and an adaptive service level tuning scheme. EDF-VD has shown strong competence in both theoretical and empirical evaluations [4]. Compared with the work presented in [12], our approach uses a more flexible strategy that allows a component/system to abandon low-criticality tasks in accordance with run-time demands. Therefore, both of the problems stated above are addressed in our approach. In contrast to the work of [12], [22], our approach is based on a utilization-based schedulability analysis. The system schedulability can be effectively determined. From the designer's perspective, our utilization-based approach requires simpler specifications and reasoning compared with the work of [12], [22]. In terms of flexibility, our approach can efficiently allocate execution budgets for low-criticality tasks during runtime in accordance with demands, whereas the approaches presented in [12], [22] require that low-criticality tasks should be executed in accordance with the dependencies between low-criticality and high-criticality tasks that have been determined in off-line.

3 System Models and Background

3.1 FMC Implicit-Deadline Sporadic Task Model

Task Model. We consider an MC system with two different criticality levels, *HI* and *LO*. The task set γ contains n MC implicit-deadline sporadic tasks which are scheduled on a uni-processor platform. Each task τ_i in γ generates an infinite sequence of jobs and can be specified by a tuple $\{T_i, L_i, \mathcal{C}_i\}$. Here, T_i denotes the minimum job-arrival intervals. $L_i \in \{LO, HI\}$ denotes the criticality level of a task. Each task is either a low-criticality task or high-criticality task. γ_{LO} and γ_{HI} (where $\gamma = \gamma_{LO} \cup \gamma_{HI}$) denote low-criticality task set and high-criticality task set, respectively. $\mathcal{C}_i \in \{C_i^{LO}, C_i^{HI}\}$ is the list of WCETs, where C_i^{LO} and C_i^{HI} denote the low-criticality and high-criticality WCETs, respectively.

For high-criticality tasks, the WCETs satisfy $C_i^{LO} < C_i^{HI}$. For low-criticality tasks, their execution budget is dynamically determined in FMC based on the overruns of high-criticality tasks. To characterize the execution behavior of low-criticality tasks in high-criticality mode, we now introduce the concept of the service level on each mode-switching point, which specifies the guaranteed service quality after the mode switch.

Service Level. Instead of completely discarding all low-criticality tasks, Burns and Baruah in [7] proposed a more practical MC task model in which low-criticality tasks are allowed to statically reserve resources for their execution at a degraded service level in high-criticality mode (i.e., a reduced execution budget). By contrast, in FMC, the execution budget is dynamically determined based on the run-time overruns rather than statically reserved as in [7]. To model this dynamic behavior, the service level concept defined in [7] should be extended to apply to independent mode switches. Therefore, we define the service level z_i^k when the system has undergone k mode switches.

Definition 1 (Service level z_i^k when k mode switches have occurred). *If low-criticality task τ_i is executed at service level z_i^k when the system has undergone k mode switches, up to $z_i^k \cdot C_i^{LO}$ time units can be used for the execution of τ_i in one period T_i. When τ_i runs in low-criticality mode, we say τ_i is executed at service level z_i^0, where $z_i^0 = 1$.*

The service level definition given above is compliant with the concept of the *imprecise computation model* developed by Lin et al. [18] to deal with time-constrained iterative calculations. *Imprecise computation model* is partly motivated by the observation that many real-time computations are iterative in nature, solving a numeric problem by successive approximations. Terminating an iteration early can return useful imprecise results. With this motivation in mind, the imprecise computation model can be used in a natural way to enhance graceful degradation [20]. The practicality of *imprecise computation model* has been deeply investigated and verified in [9]. Imprecise computation model provides an approximate but timely result, which may be acceptable in many application areas. Examples of such applications are optimal control [6], multimedia applications [21], image and speech processing [10], and fault-tolerant scheduling problems [13]. In FMC, when an overrun occurs, low-criticality tasks will be terminated before completion and sacrifice the quality of the produced results to ensure their timing correctness.

Utilization. Low and high utilization for a task τ_i are defined as $u_i^{LO} = \frac{c_i^{LO}}{T_i}$ and $u_i^{HI} = \frac{c_i^{HI}}{T_i}$, respectively. The system-level utilization for task set γ are defined as $u_{LO}^{LO} = \sum_{\tau_i \in \gamma_{LO}} u_i^{LO}$, $u_{HI}^{LO} = \sum_{\tau_i \in \gamma_{HI}} u_i^{LO}$, and $u_{HI}^{HI} = \sum_{\tau_i \in \gamma_{HI}} u_i^{HI}$. The system utilization of low-criticality tasks after k^{th} mode-switching point can be defined as $u_{LO}^k = \sum_{\tau_i \in \gamma_{LO}} z_i^k \cdot u_i^{LO}$. To guarantee the execution of the mandatory portions of low-criticality tasks, the mandatory utilization can be defined as $u_{LO}^{man} = \sum_{\tau_i \in \gamma_{LO}} z_i^{man} \cdot u_i^{LO}$, where z_i^{man} is the mandatory service level for task τ_i as specified by the users.

Assumptions. For the remainder of the paper, we make the following assumptions: (1) Regarding the compensation for the k^{th} overrun of a high-criticality task, we assume that $z_i^k \leq z_i^{k-1}$. After the k^{th} mode-switching point, the allowed execution time budget in one period should thus be reduced from $z_i^{k-1} \cdot c_i^{LO}$ to $z_i^k \cdot c_i^{LO}$. (2) According to [4], if $u_{LO}^{LO} + u_{HI}^{HI} \leq 1$, then all tasks can be perfectly scheduled by regular EDF under the worst-case reservation strategy. Therefore, we here consider meaningful cases in which $u_{LO}^{LO} + u_{HI}^{HI} > 1$.

3.2 Execution Semantics of the FMC Model

The main *differences* between our FMC execution model and the classic MC execution model lie in the independent mode-switch triggering scheme for high-criticality tasks and the dynamic service tuning of low-criticality tasks. In contrast to the classic MC model, the FMC model allows an independent triggering scheme in which the overrun of one high-criticality task triggers only itself into high-criticality mode. Consequently, the high-criticality mode of the system in FMC depends on the number of high-criticality tasks that have overrun. Therefore, we introduce the following definition:

Definition 2 (k-level high-criticality mode). *At a given instant of time, if k high-criticality tasks have entered high-criticality mode, then the system is in k*-level high-criticality mode. *For low-criticality mode, we say that the system is in* 0-level high-criticality mode.

Based on Definition 2, the execution semantics of the FMC model is illustrated in Fig. 1. Initially, the system is in low-criticality mode (i.e., *0-level high-criticality mode*). Then, the overruns of high-criticality tasks trigger the system to

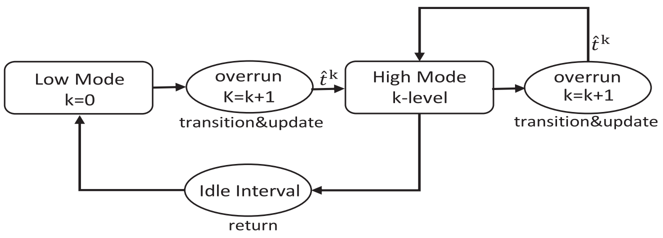


Fig. 1. Execution semantics of the FMC model.

proceed through the high-criticality modes one by one until the condition for transitioning back is satisfied. According to Fig. 1, the execution semantics can be summarized as follows:

- *Low-criticality mode*: All tasks in γ start in 0-level high-criticality mode (i.e., low-criticality mode). As long as no high-criticality task violates its C_i^{LO}, the system remains in 0-level high-criticality mode. In this mode, all tasks are scheduled with C_i^{LO}.
- *Transition*: When one job of a high-criticality task *that is being executed in low-criticality mode* overruns its C_i^{LO}, this high-criticality task immediately switches into high-criticality mode. However, the overrun of this task does not trigger other high-criticality tasks to enter high-criticality mode. All other high-criticality tasks still remain in the *same* mode as before. Correspondingly, the system transitions to a higher-level high-criticality mode.[1]
- *Updates*: At the k^{th} transition point (corresponding to time instant $\hat{t}^k$ in Fig. 1), a new service level z_i^k is determined and updated to provide degraded service for low-criticality tasks τ_i to balance the resource demand caused by the overrun of the high-criticality task. At this time, if any low-criticality jobs have completed more than $z_i^k \cdot c_i^{LO}$ time units of execution (i.e., have used up the updated execution budget for the current period), those jobs will be suspended immediately and wait for the budget to be renewed in the next period. Otherwise, low-criticality jobs can continue to use the remaining time budget for their execution.
- *Return to low-criticality mode*: When the system detects an idle interval [7], [23], the system will transition back into low-criticality mode.

3.3 EDF-VD Scheduling

EDF-VD [4], [5] is a scheduling algorithm for implementing classic preemptive EDF scheduling in MC systems. The main concept of EDF-VD is to artificially reduce the (virtual) deadlines of high-criticality tasks when the system is in low-criticality mode. These virtual deadlines can be used to cause high-criticality tasks to finish earlier to ensure that the system can reserve a sufficient execution budget for the high-criticality tasks to meet their actual deadlines after the system switches into high-criticality mode. In this paper, we study the schedulability under EDF-VD scheduling for the proposed FMC model.

1. Without loss of generality, we assume that the system is in k-level high-criticality mode.

TABLE 1
Example Task Set

	L_i	T_i	C_i^{LO}	C_i^{HI}
$\tau_1, \tau_2, \tau_3, \tau_4$	HI	40	3	8
τ_5	LO	200	30	
τ_6	LO	300	75	

4 FMC-EDF-VD Scheduling Algorithm

In this section, we provide an overview of the proposed EDF-VD-based scheduling algorithm for our FMC model, called FMC-EDF-VD. The proposed scheduling algorithm consists of an *off-line step* and a *run-time step*. We implement the *off-line step* prior to *run time* to select a feasible virtual deadline factor x for tightening the deadlines of high-criticality tasks. During run time, the service levels z_i^k for low-criticality tasks are dynamically tuned based on the overrun of high-criticality tasks. Here, we present the operation flow of FMC-EDF-VD.

Off-Line Step. In accordance with Theorem 1, we first determine x as $\frac{u_{HI}^{LO}}{1-u_{LO}^{LO}}$. Then, to guarantee the schedulability of FMC-EDF-VD, the determined x value should be validated by testing condition Eqn. (24) in Theorem 5. Note that if condition Eqn. (24) is not satisfied, then it is reported that the specified task set cannot be scheduled using FMC-EDF-VD.

Run-Time Step. The run-time behavior follows the execution semantics presented in Section 3.2. In low-criticality mode, all high-criticality tasks are scheduled with their virtual deadlines. At each mode-switching point, the following two procedures are triggered:

- Reset the deadline of overrun high-criticality task from its virtual deadline to the actual deadline. The deadline settings of other high-criticality tasks remain the same as before.
- Update the service levels for low-criticality tasks in accordance with Theorem 2.

Note that various run-time tuning strategies can be specified by the user as long as the condition in Theorem 2 is satisfied. For the purpose of demonstration, a uniform tuning strategy and a dropping-off strategy are discussed in this paper. Complete descriptions of these strategies are provided in Section 6.

4.1 Motivational Example

In this section, we present a motivational example to show how the global triggering scheme in FMC-EDF-VD can efficiently support low-criticality task execution. The uniform tuning strategy (see Theorem 6), in which all low-criticality tasks share the same service level setting z^k during run time (i.e., $\forall \tau_i \in \gamma_{LO}$, $z_i^k = z^k$), is adopted for this demonstration.

Example 1. For clarity of presentation, we consider a task set that contains four identical high-criticality tasks and two low-criticality tasks, as listed in Table 1. We specify $u_{LO}^{man} = 0$ for demonstration. From Table 1, one can derive $u_{LO}^{LO} = \frac{2}{5}$, $u_{HI}^{LO} = \frac{3}{10}$, and $u_{HI}^{HI} = \frac{4}{5}$.

According to Theorem 6, we can compute the uniform service levels z^k for all possible mode-switching scenarios. The results are listed in Table 2.

TABLE 2
Low-Criticality Service Levels

Number of Overrun k	1	2	3	4
Utilization u_{LO}^k	0.3	0.2	0.1	0
Service Level z^k	0.75	0.5	0.25	0
Execution Budget of τ_5	22.5	15	7.5	0
Execution Budget of τ_6	56.25	37.5	18.75	0

As shown in Table 2, FMC-EDF-VD can efficiently support low-criticality task execution by dynamically tuning the low-criticality execution budget based on overrun demand. When only one high-criticality task overruns, low-criticality task τ_5 and τ_6 can use up to 22.5 and 56.25 time units per period for execution. In this case, low-criticality tasks can maintain 75 percent execution. Only when all high-criticality tasks overrun their C_i^L, low-criticality tasks are all dropped. For comparison, the global triggering strategy used in [7], [19] are always required to drop all low-criticality tasks regardless of how many overruns occur during run time because of the overapproximation of the overrun workload. From a probabilistic perspective, the likelihood that all high-criticality tasks will exhibit high-criticality behavior is very low in practice. Therefore, in a typical case, only a few high-criticality tasks will overrun their C_i^L during a busy interval. In most cases, FMC-EDF-VD will only need to schedule resources for a portion of high-criticality tasks based on their overrun demands and can maintain the service level for low-criticality task execution to the greatest possible extent. In this sense, FMC-EDF-VD can provide better and more graceful service degradation.

5 Schedulability Test Condition

In this section, we present a utilization-based schedulability test condition for the FMC-EDF-VD scheduling algorithm. We start by ensuring the schedulability of the system when it is operating in low-criticality mode (Theorem 1). Then, we discuss how to derive a sufficient condition to ensure the schedulability of the algorithm after k mode switches (Theorem 2). Based on several sophisticated new techniques, the correctness of this new schedulability test condition can be proven and the formal proof is provided in Section 5.3. Finally, we derive the region of x that can guarantee the feasibility of the proposed scheduling algorithm.

5.1 Low-Criticality Mode

In low-criticality mode, the system behaviors in FMC are exactly the same as in EDF-VD [4]. Therefore, we can use the following theorem presented in [4] to ensure the schedulability of tasks in low-criticality mode.

Theorem 1. *The following condition is sufficient to ensure that EDF-VD can successfully schedule all tasks in low-criticality mode:*

$$u_{LO}^{LO} + \frac{u_{HI}^{LO}}{x} \leq 1 \tag{1}$$

5.2 High-Criticality Mode After k Mode Switches

In this section, we analyze the schedulability of the FMC-EDF-VD algorithm during the transition phase. With this analysis, we provide the answer to the question of how much execution budget can be reserved for low-criticality tasks while ensuring a schedulable system for mode transitions. Without loss of generality, we consider a general transition case in which the system transitions from $(k-1)$-level high-criticality mode to k-level high-criticality mode. Here, we first introduce the derived schedulability test condition in Theorem 2. Then, the formal proof of the correctness of this schedulability test condition is provided in Section 5.3. Recall that u_{LO}^k denotes the utilization of low-criticality tasks for the k^{th} mode-switching point and is defined as $u_{LO}^k = \sum_{\tau_i \in \gamma_{LO}} z_i^k \cdot u_i^{LO}$.

Theorem 2. *The system is in $(k-1)$-level high-criticality mode. For the k^{th} mode-switching point $\hat{t}^k$, when high-criticality task $\tau_{\hat{t}^k}$ overruns, the system is schedulable at $\hat{t}^k$ if the following conditions are satisfied:*

$$u_{LO}^k \leq u_{LO}^{k-1} + \frac{\frac{u_{\hat{t}^k}^{LO}}{u_{HI}^{LO}}(1-u_{LO}^{LO}) - u_{\hat{t}^k}^{HI}}{(1-x)} \tag{2}$$

$$z_i^k \leq z_i^{k-1} \quad (\forall \tau_i \in \gamma_{LO}) \tag{3}$$

where $u_{\hat{t}^k}^{LO}$ and $u_{\hat{t}^k}^{HI}$ denote low and high utilization, respectively, for the high-criticality task $\tau_{\hat{t}^k}$ that undergoes a mode switch at $\hat{t}^k$.

In Theorem 2, we present a general utilization-based schedulability test condition for the FMC model. Now, let us take a closer look at the conditions specified in Theorem 2. We observe the following interesting properties of FMC-EDF-VD:

- In Theorem 2, the desired utilization balance between low-criticality and high-criticality tasks is achieved. As constrained by Eqn. (3), the utilization of low-criticality tasks should be further reduced when a new overrun occurs. As shown in Eqn. (2), the utilization reduction $u_{LO}^k - u_{LO}^{k-1}$ is bounded by $\frac{\frac{u_{\hat{t}^k}^{LO}}{u_{HI}^{LO}}(1-u_{LO}^{LO}) - u_{\hat{t}^k}^{HI}}{(1-x)}$ for utilization balance.
- Another important observation is that the bound on the utilization reduction is determined *only* by the overrun of high-criticality task $\tau_{\hat{t}^k}$ (as shown in Eqn. (2)). This means that the effects of the overruns on utilization reduction are independent. Moreover, the occurrence sequence of high-criticality task overruns has no impact on the utilization reduction.
- Theorem 2 also provides us with a generic metric for managing the resources of low-criticality tasks when each independent mode switch occurs. In general, various run-time tuning strategies can be applied during the transition phase, as long as the conditions in Theorem 2 are satisfied.

5.3 The Proof of Correctness

We now prove the correctness of the schedulability test condition presented in Theorem 2. We start with the proof by introducing some important concepts. Then, we propose a key technique to obtain the bound of the cumulative execution time for low-criticality and high-criticality tasks (Lemma 1, Lemma 2, and Lemma 3). Based on these derived bounds, the utilization-based test condition can be derived.

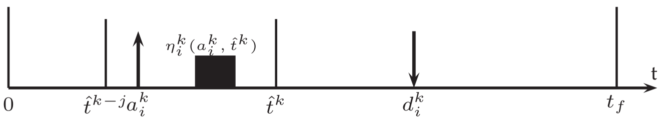


Fig. 2. The execution scenario for a k-*carry-over job*.

5.3.1 *Challenges*

Incorporating the FMC model into a utilization-based EDF-VD scheduling analysis introduces several new challenges. The independent triggering scheme and the adaptive service level tuning scheme in the FMC model allow flexible system behaviors. However, this flexibility also makes the system behavior more complex and more difficult to analyze. In particular, it is difficult to *effectively* determine an upper bound on the cumulative execution time for low-criticality tasks. In the FMC model, the service levels for low-criticality tasks are dynamically tuned at each mode-switching point. Therefore, the cumulative execution time of low-criticality tasks strongly depends on when each mode switch occurs. However, this information is difficult to explicitly represent prior to real execution because the independence of the mode switches in the FMC model results in a large analysis space. This makes it computationally infeasible to analyze all possibilities. Moreover, apart from the timing information of multiple mode switches, the derivation of the cumulative execution time also depends on the service tuning decisions made at previous mode switches. Determining how to extract static information (i.e., utilization) to formulate a feasible sufficient condition from these variables is another challenging task.

5.3.2 *Concepts and Notation*

Before diving into the detailed proofs, we introduce some commonly used concepts and notation that will be used throughout the proofs. To derive a sufficient test, suppose that there is a time interval $[0, t_f]$ such that the system undergoes the k^{th} mode switch and the first deadline miss occurs at t_f. Let J be the minimal set of jobs released from the MC task set γ for which a deadline is missed. This minimality means that if any job is removed from J, the remainder of J will be schedulable. Here, we introduce some notation for later use. $\hat{t}^k$ denotes the time instant of the k^{th} mode switch caused by high-criticality task $\tau_{\hat{t}^k}$. The absolute release time and deadline of the job of $\tau_{\hat{t}^k}$ that overruns at $\hat{t}^k$ are denoted by $a_{\hat{t}^k}$ and $d_{\hat{t}^k}$, respectively. $\eta_i^k(t_1, t_2)$ denotes the cumulative execution time of task τ_i when the system is operating in k-level high-criticality mode during the interval $(t_1, t_2]$. Next, we define a special type of job for low-criticality tasks, called a *carry-over job*, and introduce several important propositions that will be useful for our later proofs.

Definition 3. *A job of low-criticality task τ_i is called a* k-carry-over job *if the k^{th} mode switch occurs in the interval $[a_i^k, d_i^k]$, where a_i^k and d_i^k are the absolute release time and deadline of this job, respectively.*

Fig. 2 shows how a k-*carry-over job* is executed during the interval $[a_i^k, d_i^k]$. The black box represents the cumulative execution time $\eta_i^k(a_i^k, \hat{t}^k)$ of the k-*carry-over job* before the k^{th} mode-switching point $\hat{t}^k$.

Proposition 1. (From [4], [5]) All jobs executed in $[\hat{t}^k, t_f]$ have a deadline $\leq t_f$.

Proposition 2. The k^{th} mode-switching point $\hat{t}^k$ satisfies $\hat{t}^k \leq a_{\hat{t}^k} + x \cdot (t_f - a_{\hat{t}^k})$.

Proof. Since a high-criticality job of $\tau_{\hat{t}^k}$ triggers the k^{th} mode switch at $\hat{t}^k$, its virtual deadline $a_{\hat{t}^k} + x \cdot (d_{\hat{t}^k} - a_{\hat{t}^k})$ must be greater than $\hat{t}^k$. Otherwise, the high-criticality job would have completed its execution before the time instant of the switch. □

Proposition 3. For a k-*carry-over job* of low-criticality task τ_i, if $\eta_i^k(a_i^k, \hat{t}^k) \neq 0$, then the following holds: $d_i^k \leq a_{\hat{t}^k} + x \cdot (t_f - a_{\hat{t}^k})$.

Proof. There are two cases to consider: $a_i^k \geq a_{\hat{t}^k}$ and $a_i^k < a_{\hat{t}^k}$.

Case 1 ($a_i^k \geq a_{\hat{t}^k}$): In this case, for the k-*carry-over job* to be executed after $a_{\hat{t}^k}$, the k-*carry-over job* should have a deadline no later than the virtual deadline $a_{\hat{t}^k} + x(d_{\hat{t}^k} - a_{\hat{t}^k})$ of the high-criticality job that triggered the k^{th} mode switch. As a result, because $d_{\hat{t}^k} \leq t_f$, we have $d_i^k \leq (a_{\hat{t}^k} + x \cdot (t_f - a_{\hat{t}^k}))$.

Case 2 ($a_i^k < a_{\hat{t}^k}$): We prove the correctness of this case by contradiction. Suppose that the k-*carry-over job* of low-criticality task τ_i, with its deadline of $d_i^k > (a_{\hat{t}^k} + x \cdot (t_f - a_{\hat{t}^k}))$, were to be executed before $a_{\hat{t}^k}$. Let t^* denote the latest time instant at which this k-*carry-over job* is executed before $a_{\hat{t}^k}$. At time instant $\hat{t}^k$, all previous $(k-1)$ mode switches are known to the system[2]. At t^*, we know that there should be no pending job with a deadline of $\leq (a_{\hat{t}^k} + x \cdot (t_f - a_{\hat{t}^k}))$. This means that jobs that are released at or after t^* will also suffer a deadline miss at t_f, which contradicts the minimality of J. Therefore, $d_i^k \leq (a_{\hat{t}^k} + x \cdot (t_f - a_{\hat{t}^k}))$. □

Using the propositions and notation presented above, we now derive an upper bound on the cumulative execution time $\eta_i^k(0, t_f)$ for low-criticality tasks (Lemma 1) and high-criticality tasks (Lemma 2 and Lemma 3).

5.3.3 *Bound for Low-Criticality Tasks*

As discussed above, it is difficult to derive an upper bound on the cumulative execution time of low-criticality tasks during the interval $[0, t_f]$ because of the large analysis space. In this section, we propose a novel derivation strategy to resolve this challenge. The overall derivation strategy is based on the specified derivation protocol (Rule 1-Rule 4) and mathematical induction. The purpose of the derivation protocol is to specify unified *intermediate* upper bounds for different execution scenarios. The advantage of introducing these *intermediate* upper bounds is that we can *virtually* hide the influence of the previous $k-1$ mode switches. For instance, in Rule 1 (see Eqn. (4)), the influence of the previous $k-1$ mode switches is hidden in the term $\sup\{\eta_i^k(0, d_i^l)\}$. In this way, the k^{th} mode switch and the previous $k-1$ mode switches are decorrelated.

Throughout the remainder of this section, we will use $\sup\{\eta_i^k(t_1, t_2)\}$ to denote the *intermediate* upper bounds on $\eta_i^k(t_1, t_2)$ for different execution scenarios, which represent the upper bounds under specific conditions. Let $\hat{t}^{k-j}$ $(j > 0)$ denote the last mode-switching point before a_i^k (as shown in

2. At $\hat{t}^k$, all previous $k-1$ mode switches have already occurred.

Fig. 2). z_i^{k-j} denotes the updated service level at $\hat{t}^{k-j}$. d_i^l denotes the absolute deadline for the last job[3] of τ_i during $[0, t_f]$. Now, we present the rules for deriving $\sup\{\eta_i^k(0, t_f)\}$ and $\sup\{\eta_i^k(a_i^k, d_i^k)\}$, as summarized in Eqns. (4) and (5).

$$\sup\{\eta_i^k(0,t_f)\} = \begin{cases} \sup\{\eta_i^k(0,d_i^l)\} + (t_f - \hat{t}^k)\cdot z_i^k \cdot u_i^{LO} & d_i^l < \hat{t}^k \ (\textbf{Rule } 1) \\ \sup\{\eta_i^k(0,d_i^k)\} + (t_f - d_i^k)\cdot z_i^k \cdot u_i^{LO} & \text{Otherwise } (\textbf{Rule } 2) \end{cases} \tag{4}$$

$$\sup\{\eta_i^k(a_i^k,d_i^k)\} = \begin{cases} (d_i^k - a_i^k)\cdot z_i^{k-j}\cdot u_i^{LO} & \eta_i^k(a_i^k,\hat{t}^k) \neq 0 \ (\textbf{Rule } 3) \\ (d_i^k - a_i^k)\cdot z_i^k \cdot u_i^{LO} & \text{Otherwise } (\textbf{Rule } 4) \end{cases} \tag{5}$$

The detailed description and proof are presented in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TC.2017.2763133. In Rule 1-Rule 4, one may notice that there are several different execution scenarios in which only one mode switch is considered. When n mode switches are allowed, the combination space for all execution scenarios will increase exponentially with n. In general, it is very difficult to derive a bound on the cumulative execution time for low-criticality tasks because of this large combination space. To solve this problem, we analyze the difference between $\sup\{\eta_i^k(0, t_f)\}$ and $\sup\{\eta_i^{k-1}(0, t_f)\}$ and find that this difference can be uniformly bounded by a *difference term* ψ_i^k (see Lemma 1). This finding is formally proven in Lemma 1 through mathematical induction. Before the proof, we first present a fact that will be useful for later interpretation.

Fact 1. For the k^{th} mode-switching point $\hat{t}^k$, at time instant t_0 such that $t_0 \le \hat{t}^k$, $\eta_i^k(0,t_0) = \eta_i^{k-1}(0,t_0)$.

Proof. The k^{th} mode switch can only begin to affect low-criticality task execution after the corresponding mode-switching point $\hat{t}^k$. Before $\hat{t}^k$, the k^{th} mode switch has no impact. Thus, we have $\eta_i^k(0,t_0) = \eta_i^{k-1}(0,t_0)$. □

Lemma 1. *For all $k \ge 1$, the cumulative execution time $\eta_i^k(0, t_f)$ can be upper bounded by*

$$t_f \cdot u_i^{LO} + \sum_{j=1}^{k} \psi_i^j \tag{6}$$

where the difference term *ψ_i^j is defined as $(t_f - a_{\hat{t}^j})(1-x)(z_i^j - z_i^{j-1})u_i^{LO}$.*

Proof. Instead of proving the original statement, we will prove an alternative statement $P(k)$, which is defined as follows:

The intermediate upper bounds $\sup\{\eta_i^k(0,t_f)\}$ under different execution scenarios can be uniformly upper bounded by Eqn. (6).

Since $\eta_i^k(0,t_f) \le \sup\{\eta_i^k(0,t_f)\}$, the original statement will be proven correct if the statement $P(k)$ is proven to be correct. Now, we will prove that the statement $P(k)$ is correct for all possible integers k based on mathematical induction. Recall that d_i^l is the absolute deadline for the last job of τ_i during $[0, t_f]$.

Step 1*(base case)*: We will prove that $P(1)$ is correct for $k = 1$. □

3. Here, the last job means the last job with a deadline of $\le t_f$.

Proof. We consider two cases, one in which a *carry-over job* does not exist at the first mode-switching point $\hat{t}^1$ (i.e., $d_i^l < \hat{t}^1$) and one in which such a job does exist (i.e., $d_i^l \ge \hat{t}^1$).

Case 1 ($\mathbf{d_i^l < \hat{t}^1}$): According to Rule 1 and Property 2, we have the following:

$$\begin{aligned} &\sup\{\eta_i^1(0,t_f)\} \\ &= \sup\{\eta_i^1(0,d_i^l)\} + (t_f - \hat{t}^1)\cdot z_i^1 \cdot u_i^{LO} \\ &= d_i^l \cdot u_i^{LO} + (t_f - \hat{t}^1)\cdot z_i^1 \cdot u_i^{LO} \\ &\quad \textit{since} \;\; d_i^l < \hat{t}^1 \le a_{\hat{t}^1} + x\cdot(t_f - a_{\hat{t}^1}) \\ &< t_f \cdot z_i^1 \cdot u_i^{LO} + \hat{t}^1 \cdot u_i^{LO}\cdot(1 - z_i^1) \quad (\textit{replace } d_i^l \textit{ with } \hat{t}^1) \\ &\le t_f \cdot u_i^{LO} + \underbrace{(t_f - a_{\hat{t}^1})(1-x)(z_i^1 - 1)u_i^{LO}}_{\textit{difference term } \psi_i^1} \quad (\textit{replace } \hat{t}^1) \end{aligned}$$

Case 2 ($\mathbf{d_i^l \ge \hat{t}^1}$): In this case, we consider the two following execution scenarios.

S1 ($\boldsymbol{\eta_i^1(a_i^1, \hat{t}^1) \neq 0}$): According to Rule 2, Rule 3, and Property 3, we have the following:

$$\begin{aligned} &\sup\{\eta_i^1(0,t_f)\} \\ &= \sup\{\eta_i^1(0,a_i^1)\} + \sup\{\eta_i^1(a_i^1,d_i^1)\} + (t_f - d_i^1)z_i^1 u_i^{LO} \\ &= a_i^1 u_i^{LO} + (d_i^1 - a_i^1)u_i^{LO} + (t_f - d_i^1)z_i^1 u_i^{LO} \\ &= t_f \cdot u_i^{LO} + (t_f - d_i^1)(z_i^1 - 1)u_i^{LO} \\ &\quad \textit{since} \; d_i^1 \le a_{\hat{t}^1} + x\cdot(t_f - a_{\hat{t}^1}) \\ &\le t_f \cdot u_i^{LO} + \underbrace{(t_f - a_{\hat{t}^1})(1-x)(z_i^1 - 1)u_i^{LO}}_{\textit{difference term } \psi_i^1} \quad (\textit{replace } \; d_i^1) \end{aligned}$$

S2 ($\boldsymbol{\eta_i^1(a_i^1, \hat{t}^1) = 0}$): According to Rule 2, Rule 4, and Property 2, we have the following:

$$\begin{aligned} &\sup\{\eta_i^1(0,t_f)\} \\ &= \sup\{\eta_i^1(0,a_i^1)\} + \sup\{\eta_i^1(a_i^1,d_i^1)\} + (t_f - d_i^1)z_i^1 u_i^{LO} \\ &= a_i^1 u_i^{LO} + (d_i^1 - a_i^1)z_i^1 u_i^{LO} + (t_f - d_i^1)z_i^1 u_i^{LO} \\ &= t_f \cdot u_i^{LO} + (t_f - a_i^1)(z_i^1 - 1)u_i^{LO} \\ &\quad \textit{since} \; a_i^1 < \hat{t}^1 \le a_{\hat{t}^1} + x\cdot(t_f - a_{\hat{t}^1}) \\ &\le t_f \cdot u_i^{LO} + \underbrace{(t_f - a_{\hat{t}^1})(1-x)(z_i^1 - 1)u_i^{LO}}_{\textit{difference term } \psi_i^1} \quad (\textit{replace } \; a_i^1) \end{aligned}$$

Therefore, $P(1)$ is correct for $k = 1$. □

Step 2 *(induction hypothesis)*: Assume that $P(k_0 - 1)$ is correct for some possible integers $k_0 - 1$.

Step 3. *(induction)*: We now prove that $P(k_0)$ is correct by the induction hypothesis.

Proof. Since $\hat{t}^{k_0-1} \le \hat{t}^{k_0}$, we need to consider the following three cases.

Case 1($\mathbf{d_i^l < \hat{t}^{k_0-1} \le \hat{t}^{k_0}}$): In this case, neither a (k_0− 1)-*carry-over job* nor a k_0-*carry-over job* exists. According to Rule 1 and Fact 1, we have the following:

$$\begin{aligned} \sup\{\eta_i^{k_0-1}(0,t_f)\} &= \sup\{\eta_i^{k_0-1}(0,d_i^l)\} + (t_f - \hat{t}^{k_0-1})z_i^{k_0-1}u_i^{LO} \\ \sup\{\eta_i^{k_0}(0,t_f)\} &= \sup\{\eta_i^{k_0}(0,d_i^l)\} + (t_f - \hat{t}^{k_0})z_i^{k_0}u_i^{LO} \\ \sup\{\eta_i^{k_0-1}(0,d_i^l)\} &= \sup\{\eta_i^{k_0}(0,d_i^l)\} \end{aligned}$$

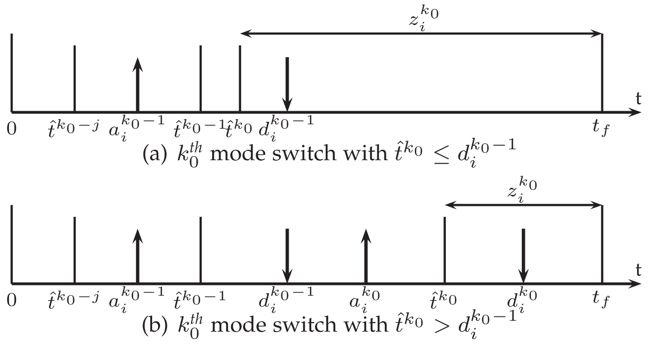


Fig. 3. Mode switch from $\hat{t}^{k_0-1}$ to $\hat{t}^{k_0}$.

Since $\hat{t}^{k_0} \ge \hat{t}^{k_0-1}$ and $z_i^{k_0} \le z_i^{k_0-1}$, we have

$$\sup\{\eta_i^{k_0}(0,t_f)\} \le \sup\{\eta_i^{k_0-1}(0,t_f)\} + (t_f - \hat{t}^{k_0})(z_i^{k_0} - z_i^{k_0-1})u_i^{LO} \tag{7}$$

According to Property 2, we can replace $\hat{t}^{k_0}$ with $a_{\hat{t}^{k_0}} + x(t_f - a_{\hat{t}^{k_0}})$ in Eqn. (7). Then, $\sup\{\eta_i^{k_0}(0,t_f)\}$ can be bounded by

$$\sup\{\eta_i^{k_0-1}(0,t_f)\} + \underbrace{(t_f - a_{\hat{t}^{k_0}}) \cdot (1-x) \cdot (z_i^{k_0} - z_i^{k_0-1}) \cdot u_i^{LO}}_{difference\ term\ \psi_i^{k_0}}$$

Case 2 ($\hat{\mathbf{t}}^{k_0-1} \le \hat{\mathbf{t}}^{k_0} \le \mathbf{d}_\mathbf{i}^\mathbf{l}$): In this case, both a (k_0-1)-*carry-over job* and a k_0-*carry-over job* exist. Recall that $d_i^{k_0-1}$ is the absolute deadline for the (k_0-1)-*carry-over job*. Two sub-cases, one with $\hat{t}^{k_0} \le d_i^{k_0-1}$ and one with $\hat{t}^{k_0} > d_i^{k_0-1}$, as shown in Figs. 3a and 3b, need to be considered.

According to Fact 1, we have the following:

$$\eta_i^{k_0}(0, a_i^{k_0}) = \eta_i^{k_0-1}(0, a_i^{k_0}) \tag{8}$$

• **Case 2-A ($\hat{t}^{k_0} \le d_i^{k_0-1}$)**: This execution scenario is illustrated in Fig. 2. In this case, the (k_0-1)-*carry-over job* and the k_0-*carry-over job* are the same job. Therefore, we have $a_i^{k_0} = a_i^{k_0-1}$ and $d_i^{k_0} = d_i^{k_0-1}$. In the following, we use $a_i^{k_0-1}$ and $d_i^{k_0-1}$ in place of $a_i^{k_0}$ and $d_i^{k_0}$, respectively. In Case 2-A, the following two scenarios are considered:

S1 ($\eta_\mathbf{i}^{\mathbf{k_0}}(\mathbf{a_i^{k_0-1}}, \hat{\mathbf{t}}^{\mathbf{k_0}}) \neq \mathbf{0}$): According to Rule 3 and Rule 4, we have the following[4]:

$$\sup\{\eta_i^{k_0}(a_i^{k_0-1}, d_i^{k_0-1})\} = \sup\{\eta_i^{k_0-1}(a_i^{k_0-1}, d_i^{k_0-1})\}$$
$$= \begin{cases} (d_i^{k_0-1} - a_i^{k_0-1}) \cdot z_i^{k_0-j} \cdot u_i^{LO} & \eta_i^k(a_i^{k_0-1}, \hat{t}^{k_0-1}) \neq 0 \\ (d_i^{k_0-1} - a_i^{k_0-1}) \cdot z_i^{k_0-1} \cdot u_i^{LO} & otherwise \end{cases} \tag{9}$$

According to Rule 2, Eqns. (8) and (9), we have the following:

$$\begin{aligned}
&\sup\{\eta_i^{k_0}(0,t_f)\} \\
&= \sup\{\eta_i^{k_0}(0, a_i^{k_0-1})\} + \sup\{\eta_i^{k_0}(a_i^{k_0-1}, d_i^{k_0-1})\} \\
&\quad + (t_f - d_i^{k_0-1}) z_i^{k_0} u_i^{LO} \\
&= \underline{\sup\{\eta_i^{k_0-1}(0, a_i^{k_0-1})\} + \sup\{\eta_i^{k_0-1}(a_i^{k_0-1}, d_i^{k_0-1})\}} \\
&\quad \underline{+(t_f - d_i^{k_0-1}) z_i^{k_0-1} u_i^{LO}} + (t_f - d_i^{k_0-1})(z_i^{k_0} - z_i^{k_0-1}) u_i^{LO} \\
&= \sup\{\eta_i^{k_0-1}(0,t_f)\} + (t_f - d_i^{k_0-1})(z_i^{k_0} - z_i^{k_0-1}) u_i^{LO}
\end{aligned}$$

According to Property 3, by replacing $d_i^{k_0-1}$ with $a_{\hat{t}^{k_0}} + x \cdot (t_f - a_{\hat{t}^{k_0}})$, $\sup\{\eta_i^{k_0}(0,t_f)\}$ can be bounded by

$$\sup\{\eta_i^{k_0-1}(0,t_f)\} + \underbrace{(t_f - a_{\hat{t}^{k_0}}) \cdot (1-x) \cdot (z_i^{k_0} - z_i^{k_0-1}) \cdot u_i^{LO}}_{difference\ term\ \psi_i^{k_0}} \tag{10}$$

S2 ($\eta_\mathbf{i}^{\mathbf{k_0}}(\mathbf{a_i^{k_0-1}}, \hat{\mathbf{t}}^{k_0}) = \mathbf{0}$): According to Rule 2, Rule 4, and Eqn. (8), we have the following:

$$\begin{aligned}
&\sup\{\eta_i^{k_0}(0,t_f)\} \\
&= \sup\{\eta_i^{k_0}(0, a_i^{k_0-1})\} + \sup\{\eta_i^{k_0}(a_i^{k_0-1}, d_i^{k_0-1})\} \\
&\quad + (t_f - d_i^{k_0-1}) z_i^{k_0} u_i^{LO} \\
&= \underline{\sup\{\eta_i^{k_0-1}(0, a_i^{k_0-1})\} + \sup\{\eta_i^{k_0-1}(a_i^{k_0-1}, d_i^{k_0-1})\}} \\
&\quad \underline{+(t_f - d_i^{k_0-1}) z_i^{k_0-1} u_i^{LO}} + (t_f - a_i^{k_0-1})(z_i^{k_0} - z_i^{k_0-1}) u_i^{LO} \\
&= \sup\{\eta_i^{k_0-1}(0,t_f)\} + (t_f - a_i^{k_0-1})(z_i^{k_0} - z_i^{k_0-1}) u_i^{LO}
\end{aligned}$$

According to Property 2 and $a_i^{k_0-1} < \hat{t}^{k_0}$, $\sup\{\eta_i^{k_0}(0,t_f)\}$ can be bounded by

$$\sup\{\eta_i^{k_0-1}(0,t_f)\} + \underbrace{(t_f - a_{\hat{t}^{k_0}})(1-x)(z_i^{k_0} - z_i^{k_0-1}) u_i^{LO}}_{difference\ term\ \psi_i^{k_0}} \tag{11}$$

• **Case 2-B: ($d_i^{k_0-1} < \hat{t}^{k_0}$)**: This execution scenario is illustrated in Fig. 3. In this case, the (k_0-1)-*carry-over job* and the k_0-*carry-over job* are different jobs. For this case, we will consider the following two scenarios:

S1 ($\eta_\mathbf{i}^{\mathbf{k_0}}(\mathbf{a_i^{k_0}}, \hat{\mathbf{t}}^{k_0}) \neq \mathbf{0}$): According to Rule 2, Rule 3, and Eqn. (8), we have the following:

$$\begin{aligned}
&\sup\{\eta_i^{k_0}(0,t_f)\} \\
&= \sup\{\eta_i^{k_0}(0, a_i^{k_0})\} + \sup\{\eta_i^{k_0}(a_i^{k_0}, d_i^{k_0})\} + (t_f - d_i^{k_0}) z_i^{k_0} u_i^{LO} \\
&= \underline{\sup\{\eta_i^{k_0-1}(0, a_i^{k_0})\} + (t_f - a_i^{k_0}) z_i^{k_0-1} u_i^{LO}} \\
&\quad + (t_f - d_i^{k_0})(z_i^{k_0} - z_i^{k_0-1}) u_i^{LO} \\
&= \sup\{\eta_i^{k_0-1}(0,t_f)\} + (t_f - d_i^{k_0})(z_i^{k_0} - z_i^{k_0-1}) u_i^{LO}
\end{aligned}$$

Again, by replacing $d_i^{k_0}$ in accordance with Property 3, we obtain the following bound:

$$\sup\{\eta_i^{k_0-1}(0,t_f)\} + \underbrace{(t_f - a_{\hat{t}^{k_0}})(1-x)(z_i^{k_0} - z_i^{k_0-1}) u_i^{LO}}_{difference\ term\ \psi_i^{k_0}} \tag{12}$$

S2 ($\eta_\mathbf{i}^{\mathbf{k_0}}(\mathbf{a_i^{k_0}}, \hat{\mathbf{t}}^{k_0}) = \mathbf{0}$): According to Rule 2, Rule 4, and Eqn. (8), we have the following:

$$\begin{aligned}
&\sup\{\eta_i^{k_0}(0,t_f)\} \\
&= \sup\{\eta_i^{k_0}(0, a_i^{k_0})\} + \sup\{\eta_i^{k_0}(a_i^{k_0}, d_i^{k_0})\} + (t_f - d_i^{k_0}) z_i^{k_0} u_i^{LO} \\
&= \underline{\sup\{\eta_i^{k_0-1}(0, a_i^{k_0})\} + (t_f - a_i^{k_0}) z_i^{k_0-1} u_i^{LO}} \\
&\quad + (t_f - a_i^{k_0})(z_i^{k_0} - z_i^{k_0-1}) u_i^{LO} \\
&= \sup\{\eta_i^{k_0-1}(0,t_f)\} + (t_f - a_i^{k_0})(z_i^{k_0} - z_i^{k_0-1}) u_i^{LO}
\end{aligned}$$

Again, according to Proposition 2 and $a_i^{k_0} < \hat{t}^{k_0}$, $\sup\{\eta_i^k(0,t_f)\}$ can be upper bounded by

4. According to the proof of Rule 3 (see Appendix A, available online), we have a similar result: $\sup\{\eta_i^{k_0}(a_i^{k_0-1}, d_i^{k_0-1})\} = (d_i^{k_0-1} - a_i^{k_0-1}) \cdot z_i^{k_0-1} \cdot u_i^{LO}$ because $\eta_i^k(a_i^{k_0-1}, \hat{t}^{k_0-1}) = 0$.

$$\sup\{\eta_i^{k_0-1}(0,t_f)\} + \underbrace{(t_f - a_{\hat{t}^{k_0}})(1-x)(z_i^{k_0} - z_i^{k_0-1})u_i^{LO}}_{difference\ term\ \psi_i^{k_0}} \tag{13}$$

For case 2, we can conclude that $\sup\{\eta_i^k(0,t_f)\}$ can be upper bounded by $\sup\{\eta_i^{k_0-1}(0,t_f)\} + \psi_i^{k_0}$ according to Eqns. (10)-(13).

Case 3 ($\hat{t}^{k_0-1} \le d_i^l < \hat{t}^{k_0}$): In this case, a (k_0-1)-*carry-over job* exists but a k_0-*carry-over job* does not. According to Rule 1, we have the following:

$$\sup\{\eta_i^{k_0}(0,t_f)\} = \sup\{\eta_i^{k_0}(0,d_i^l)\} + (t_f - \hat{t}^{k_0}) \cdot z_i^{k_0} \cdot u_i^{LO}$$

Since $d_i^l < \hat{t}^{k_0} < t_f$, we can derive

$$\sup\{\eta_i^{k_0-1}(0,t_f)\} = \sup\{\eta_i^{k_0-1}(0,d_i^l)\} + (t_f - d_i^l) \cdot z_i^{k_0-1} \cdot u_i^{LO}$$

According to Fact 1 and $d_i^l < \hat{t}^{k_0}$, we have

$$\sup\{\eta_i^{k_0}(0,t_f)\} \le \sup\{\eta_i^{k_0-1}(0,t_f)\} + (t_f - \hat{t}^{k_0})(z_i^{k_0} - z_i^{k_0-1})u_i^{LO}$$

Again, according to Proposition 2, $\sup\{\eta_i^k(0,t_f)\}$ can be upper bounded by

$$\sup\{\eta_i^{k_0-1}(0,t_f)\} + \underbrace{(t_f - a_{\hat{t}^{k_0}})(1-x)(z_i^{k_0} - z_i^{k_0-1})u_i^{LO}}_{difference\ term\ \psi_i^{k_0}}$$

For the three cases above, we can conclude that $\sup\{\eta_i^k(0,t_f)\}$ can be upper bounded by $\sup\{\eta_i^{k_0-1}(0,t_f)\} + \psi_i^{k_0}$. Thus, $P(k_0)$ is correct by the induction hypothesis.

Hence, through mathematical induction, $P(k)$ is proven correct for all possible k. Under different execution scenarios, the cumulative execution time $\eta_i^k(0,t_f)$ can be bounded by the *intermediate* upper bound $\sup\{\eta_i^k(0,t_f)\}$. Since $P(k)$ is correct, the original statement is correct. □

5.3.4 *Bound for High-Criticality Tasks*

Recall that $\tau_{\hat{t}^k}$ is the high-criticality task that suffers an overrun at $\hat{t}^k$. Since the mode switches are independent, the high-criticality tasks can be divided into two sets, namely, the sets of tasks that have and have not already entered high-criticality mode at mode-switching point $\hat{t}^k$, which can be denoted by $\gamma_{HI}^{HI}(\hat{t}^k)$ and $\gamma_{HI}^{LO}(\hat{t}^k)$, respectively. Now, we derive the upper bounds on the cumulative execution time for both types of high-criticality tasks.

Lemma 2. *For high-criticality task $\tau_{\hat{t}^j}$ in task set $\gamma_{HI}^{HI}(\hat{t}^k)$ ($j \le k$), the cumulative execution time $\eta_{\tau_{\hat{t}^j}}^k(0,t_f)$ can be bounded as follows:*

$$\sup\{\eta_{\tau_{\hat{t}^j}}^k(0,t_f)\} = \frac{a_{\hat{t}^j}}{x} \cdot u_{\hat{t}^j}^{LO} + (t_f - a_{\hat{t}^j}) \cdot u_{\hat{t}^j}^{HI} \tag{14}$$

Proof. For the proof, refer to fact 3 in [4]. □

Lemma 3. *For high-criticality task τ_i in task set $\gamma_{HI}^{LO}(\hat{t}^k)$, the cumulative execution time $\eta_i^k(0,t_f)$ can be bounded as follows:*

$$\sup\{\eta_i^k(0,t_f)\} = \frac{t_f}{x} \cdot u_i^{LO} \tag{15}$$

Proof. Since the job has a modified deadline $\le t_f$, the actual deadline is $\le \frac{t_f}{x}$. Therefore, $\sup\{\eta_i^k(0,t_f)\} = \frac{t_f}{x} \cdot u_i^{LO}$. □

5.3.5 *Putting It All Together*

Now, we are ready to establish the schedulability test condition. To prove Theorem 2, we first introduce two auxiliary theorems, Theorems 3 and 4. In Theorem 3, the schedulability test condition is derived based on Lemmas 1, 2, and 3. This test condition should rely on the previous mode switches. Theorem 4 demonstrates the consistency of the test condition, by which the dependences among mode switches can be removed.

Theorem 3. *At the k^{th} mode-switching point $\hat{t}^k$, k ($k \ge 1$) high-criticality tasks $\tau_{\hat{t}^1}, \tau_{\hat{t}^2}, \ldots, \tau_{\hat{t}^k}$ have switched into high-criticality mode. The system is schedulable if the service level z_i^j at $\hat{t}^j$ satisfies the following conditions for* ***all*** *j such that $1 \le j \le k$.*

$$z_i^j \le z_i^{j-1} \tag{16}$$

$$u_{\hat{t}^j}^{HI} + (1-x)(u_{LO}^j - u_{LO}^{j-1}) + \frac{u_{\hat{t}^j}^{LO}}{u_{HI}^{LO}}(u_{LO}^{LO} - 1) \le 0 \tag{17}$$

Proof. The condition $z_i^j \le z_i^{j-1}$ is a basic assumption of our model, which guarantees the satisfaction of Lemma 1, Rule 3, and Rule 4. Therefore, $z_i^j \le z_i^{j-1}$ needs to be satisfied.

Let N_γ^k denote the cumulative execution time of task set γ during the interval $[0,t_f]$ when the k^{th} mode switch occurs. To calculate N_γ^k, let us sum the the cumulative execution time of all tasks over $[0,t_f]$.

For the low-criticality task set γ_{LO}, we can bound $N_{\gamma_{LO}}^k$ according to Lemma 1.

$$N_{\gamma_{LO}}^k \le \sum_{\tau_i \in \gamma_{LO}} \left(t_f u_i^{LO} + \sum_{j=1}^{k} \psi_i^j \right) \tag{18}$$

For the high-criticality task set $\gamma_{HI}^{HI}(\hat{t}^k)$, which contains k high-criticality tasks (i.e., $\|\gamma_{HI}^{HI}(\hat{t}^k)\| = k$), we can derive the cumulative execution time according to Lemma 2.

$$N_{\gamma_{HI}^{HI}(\hat{t}^k)}^k \le \sum_{j=1}^{k} \left(\frac{a_{\hat{t}^j}}{x} \cdot u_{\hat{t}^j}^{LO} + (t_f - a_{\hat{t}^j}) \cdot u_{\hat{t}^j}^{HI} \right) \tag{19}$$

For the high-criticality tasks in $\gamma_{HI}^{LO}(\hat{t}^k)$, which have not entered high-criticality mode at $\hat{t}^k$, we can derive the cumulative execution time according to Lemma 3.

$$N_{\gamma_{HI}^{LO}(\hat{t}^k)}^k \le \sum_{\tau_i \in \gamma_{HI}^{LO}(\hat{t}^k)} \frac{t_f}{x} u_i^{LO} \tag{20}$$

Based on Eqns. (18), (19) and (20), N_γ^k can be bounded as shown in Eqn. (21). The complete derivation is given in Appendix B, available online because of space limitations.

$$\begin{aligned} N_\gamma^k &= N_{\gamma_{LO}}^k + N_{\gamma_{HI}^{HI}(\hat{t}^k)}^k + N_{\gamma_{HI}^{LO}(\hat{t}^k)}^k \\ &\le t_f + \sum_{j=1}^{k} (t_f - a_{\hat{t}^j}) \left(u_{\hat{t}^j}^{HI} + (1-x)(u_{LO}^j - u_{LO}^{j-1}) + \frac{u_{\hat{t}^j}^{LO}}{u_{HI}^{LO}}(u_{LO}^{LO} - 1) \right) \end{aligned} \tag{21}$$

Since the first deadline miss occurs at time instant t_f, the following holds[5]:

$$N_\gamma^k > t_f$$

Therefore,

$$\sum_{j=1}^{k}(t_f - a_{\hat{t}^j})\left(u_{\hat{t}^j}^{HI} + (1-x)(u_{LO}^j - u_{LO}^{j-1}) + \frac{u_{\hat{t}^j}^{LO}}{u_{HI}^{LO}}(u_{LO}^{LO} - 1)\right) > 0$$

Taking the contrapositive, we obtain

$$\sum_{j=1}^{k}(t_f - a_{\hat{t}^j})\underline{\left(u_{\hat{t}^j}^{HI} + (1-x)(u_{LO}^j - u_{LO}^{j-1}) + \frac{u_{\hat{t}^j}^{LO}}{u_{HI}^{LO}}(u_{LO}^{LO} - 1)\right)} \le 0 \tag{22}$$

Since $t_f - a_{\hat{t}^j} > 0$, to guarantee the system schedulability of task set γ at the k^{th} mode switch, it is sufficient to ensure that the term indicated in Eqn. (22) is less than 0 for all j such that $1 \le j \le k$.

$$\begin{aligned}&\forall j \text{ such that } 1 \le j \le k: \\ &u_{\hat{t}^j}^{HI} + (1-x)(u_{LO}^j - u_{LO}^{j-1}) + \frac{u_{\hat{t}^j}^{LO}}{u_{HI}^{LO}}(u_{LO}^{LO} - 1) \le 0\end{aligned} \tag{23}$$

□

In Theorem 3, at the k^{th} mode-switching point, additional conditions are imposed on the previous $k-1$ mode switches. Therefore, to remove this dependence, we require that these imposed conditions should be consistent with the decision-making at the previous mode-switching points $\hat{t}^j$ ($j < k$). We demonstrate this consistency in Theorem 4.

Theorem 4. *The new conditions imposed on $u_{LO}^1, u_{LO}^2, \ldots, u_{LO}^{k-1}$ by the k^{th} mode switch are consistent with the decisions that have been made at the previous mode-switching points.*

Proof. The conditions given in Theorem 3 for decisions that have been made at the previous $k-1$ mode-switching points $\hat{t}^j$ ($1 \le j \le k-1$) are exactly the same as the new conditions imposed on $u_{LO}^1, u_{LO}^2, \ldots, u_{LO}^{k-1}$ with the k^{th} mode switch. Therefore, their consistency is guaranteed.□

FMC Schedulability. Now, we are ready to prove Theorem 2 using Theorem 3 and Theorem 4.

Proof. According to Theorem 4, the constraints in Theorem 3 that are imposed on $u_{LO}^1, u_{LO}^2, \ldots, u_{LO}^{k-1}$ with the k^{th} mode switch have already been covered by the previous $k-1$ mode switches. Therefore, we need to check only two conditions: Eqns. (16) and (17) with $j = k$. □

5.4 Feasibility of Algorithm

In this section, we investigate the region of x values that can guarantee the feasibility of the run-time algorithm. The selection of any x from this region during the off-line phase can guarantee that a feasible solution as determined by Theorem 2 can always be found during run time. To derive this region, we first introduce several definitions and properties that will be useful for the later proof of feasibility.

According to Eqn. (2) in Theorem 2, when $\frac{u_{\hat{t}^k}^{LO}}{u_{HI}^{LO}}(1 - u_{LO}^{LO}) - u_{\hat{t}^k}^{HI} > 0$, we do not need to reduce the utilization of low-criticality tasks. The overrun of the high-criticality task at this mode-switching point is covered by the system resource margin. Only when $\frac{u_{\hat{t}^k}^{LO}}{u_{HI}^{LO}}(1 - u_{LO}^{LO}) - u_{\hat{t}^k}^{HI} \le 0$, u_{LO}^k should be decreased to compensate for the overrun of the high-criticality task. For simplicity, we define a discriminant function $\phi(\tau_i)$ for each high-criticality task τ_i to indicate whether the overrun of τ_i can be covered by the system resource margin.

Definition 4. $\phi(\tau_i) = \frac{u_i^{LO}}{u_{HI}^{LO}}(1 - u_{LO}^{LO}) - u_i^{HI} \quad (\tau_i \in \gamma_{HI})$

Definition 5. *A high-criticality task τ_i is called margin high-criticality task if $\phi(\tau_i) > 0$. Otherwise, τ_i is called compensation high-criticality task.*

Definition 6. *The margin high-criticality task set and the compensation high-criticality task set are defined as $\gamma_{HI}^{\circ} = \{\tau_i \in \gamma_{HI} | \phi(\tau_i) > 0\}$ and $\gamma_{HI}^{*} = \{\tau_i \in \gamma_{HI} | \phi(\tau_i) \le 0\}$, respectively. $\gamma_{HI} = \gamma_{HI}^{\circ} \cup \gamma_{HI}^{*}$.*

With the definitions given above, we can now perform the feasibility analysis for x.

Theorem 5. *Given the mandatory utilization u_{LO}^{man}, any x that satisfies the following condition can guarantee that a feasible solution as determined by Theorem 2 can always be found during run time.*

$$(1-x)(u_{LO}^{LO} - u_{LO}^{man}) + \sum_{\tau_i \in \gamma_{HI}^{*}} \phi(\tau_i) \ge 0 \tag{24}$$

Proof. Recall that $\gamma_{HI}^{HI}(\hat{t}^k)$ is the set of high-criticality tasks that have entered high-criticality mode at $\hat{t}^k$. By iterating the conditions in Theorem 2, a direct solution for u_{LO}^k can be obtained as follows:

$$u_{LO}^k \le u_{LO}^{LO} + \frac{\sum_{\tau_i \in \gamma_{HI}^{*} \cap \gamma_{HI}^{HI}(\hat{t}^k)} \phi(\tau_i)}{(1-x)} \tag{25}$$

To guarantee the execution of the mandatory portions of low-criticality tasks, the following condition should be satisfied for all k:

$$u_{LO}^{man} \le u_{LO}^k \le u_{LO}^{LO} + \frac{\sum_{\tau_i \in \gamma_{HI}^{*} \cap \gamma_{HI}^{HI}(\hat{t}^k)} \phi(\tau_i)}{(1-x)} \tag{26}$$

Since the right-hand side of Eqn. (26) is non-increasing with respect to the number of overrun high-criticality tasks (i.e., k), the worst-case scenario is that all high-criticality tasks in γ_{HI} enter high-criticality mode. If mandatory service can be guaranteed in this worst-case scenario, then the feasibility of the proposed algorithm is ensured. Therefore, condition Eqn. (26) can be rewritten as Eqn. (27).

$$\begin{aligned}&u_{LO}^{LO} + \frac{\sum_{\tau_i \in \gamma_{HI}^{*}} \phi(\tau_i)}{(1-x)} \ge u_{LO}^{man} \\ \Rightarrow\ &(1-x)(u_{LO}^{LO} - u_{LO}^{man}) + \sum_{\tau_i \in \gamma_{HI}^{*}} \phi(\tau_i) \ge 0\end{aligned} \tag{27}$$

5. Note that there is no idle instant within the interval $[0, t_f]$. Otherwise, jobs from set J with release times at or after the latest idle instant could form a smaller job set causing a deadline miss at t_f, which would contradict the minimality of J.

Note that u_{LO}^{man} is the mandatory utilization defined as $u_{LO}^{man} = \sum_{\tau_i \in \gamma_{LO}} z_i^{man} \cdot u_i^{LO}$, where the item $z_i^{man} \cdot u_i^{LO}$ can be considered as a mandatory part which affects the correctness of the result in imprecise computation model [18]. □

Now, we use the following example to illustrate how to test the feasibility of FMC-EDF-VD.

Example 2. Considering the task system in Example 1, we can derive $x = \frac{u_{HI}^{LO}}{1-u_{LO}^{LO}} = \frac{1}{2}$ according to Theorem 1. For high-criticality tasks, one can compute discriminant functions $\phi(\tau_1) = \phi(\tau_2) = \phi(\tau_3) = \phi(\tau_4) = -\frac{1}{20}$ in accordance with Definition 4. The feasibility of x is validated by checking condition Eqn. (24) in Theorem 5.

$$(1-x)(u_{LO}^{LO} - u_{LO}^{man}) + \sum_{\tau_i \in \gamma_{HI}^*} \phi(\tau_i) = \left(1 - \frac{1}{2}\right) \times \frac{2}{5} - 4 \times \frac{1}{20} = 0$$

Thus, we know $x = \frac{1}{2}$ that is feasible for scheduling using FMC-EDF-VD.

6 Service Level Tuning Strategy

Theorem 2 provides an important criterion for run-time service level tuning. By checking the conditions in Theorem 2, one can determine how much utilization can be reserved for low-criticality task execution to compensate for the overruns. In general, various tuning strategies can be specified by the user as long as the condition in Theorem 2 is satisfied during run time. In this paper, we present a uniform tuning strategy and a dropping-off strategy to demonstrate the performance of FMC.

6.1 Dropping-Off Strategy

To compensate for overruns, the dropping-off strategy partially drops low-criticality tasks by assigning $z_i^k = 0$ for dropped tasks. To maximize the utilization of low-criticality tasks, the tasks to be dropped can be selected according to their utilization. At each mode-switching point $\hat{t}^k$, tasks with less utilization are given higher priority for dropping. To implement this selection strategy, we can create a task table TA_{LO} during the off-line phase by sorting the low-criticality tasks in ascending order of their utilization. During run time, the utilization reduction U_R^k that is required to compensate for the kth mode switch is determined according to Theorem 2. Based on TA_{LO}, the set γ_{LO}^k of tasks that are dropped at the kth mode-switching point is determined via binary search. Note that other selection criteria, such as job completion percentage, can also be applied to select the low-criticality tasks to be dropped.

6.2 Uniform Tuning Strategy

In this section, we present a uniform tuning strategy in which $z_i^k = z^k$ holds for all low-criticality tasks. The service levels z_i^k of all low-criticality tasks τ_i are uniformly set to z^k at the kth mode-switching point. By applying $z_i^k = z^k$ in the conditions given in Theorem 2, the uniform service level z^k can be directly computed using Eqn. (28) in Theorem 6.

Theorem 6. *The system is schedulable at the $k-1$th mode-switching point with a uniform z^{k-1}. At the kth mode-switching point $\hat{t}^k$, the system is still schedulable if z^k is determined as follows:*

$$0 \le z^k \le z^{k-1} + \min\left(0, \frac{\frac{u_{\hat{t}^k}^{LO}}{u_{HI}^{LO}}(1-u_{LO}^{LO}) - u_{\hat{t}^k}^{HI}}{(1-x)u_{LO}^{LO}}\right) \tag{28}$$

where $u_{\hat{t}^k}^{LO}$ and $u_{\hat{t}^k}^{HI}$ denote low and high utilization, respectively, of the high-criticality task $\tau_{\hat{t}^k}$ that suffers an overrun at $\hat{t}^k$.

Proof. In the uniform tuning strategy, $z_i^k = z^k$ holds for any low-criticality task τ_i. Recall that $u_{LO}^k = \sum_{\tau_i \in \gamma_{LO}} z_i^k \cdot u_i^{LO}$. Thus, we can obtain Eqn. (28) by combining the two conditions expressed in Eqns. (2) and (3) in Theorem 2. □

6.3 Case Study

In this case study, we first use the task system in Example 1 to illustrate how uniform tuning strategy and dropping-off strategy work in FMC. Then, we implement the uniform tuning strategy in our simulation framework (presented in Appendix C, available online) to demonstrate the graceful low-criticality service degradation of FMC.

First of all, we consider the generalized conditions presented in Theorem 2, which determines how much utilization can be reserved for low-criticality task execution to compensate for the overruns. By applying the task system presented in Example 1 to Theorem 2, we can get the following utilization conditions

$$u_{LO}^k - u_{LO}^{k-1} \le \frac{\frac{u_{\hat{t}^k}^{LO}}{u_{HI}^{LO}}(1-u_{LO}^{LO}) - u_{\hat{t}^k}^{HI}}{(1-x)} = -\frac{1}{10} \tag{29}$$

$$z_i^k \le z_i^{k-1} \quad (\forall \tau_i \in \gamma_{LO}) \tag{30}$$

Since the high-criticality tasks are identical, each overrun will result in identical utilization reduction of $\frac{1}{10}$, as shown in Eqn. (29). Now, we illustrate how uniform tuning strategy and dropping-off strategy work based on these generalized conditions Eqn. (29) and Eqn. (30).

- For dropping-off strategy by assigning $z_i^k = 0$ for dropped tasks, the system is required to drop off a portion of low-criticality tasks to compensate for the overruns of one high-criticality task. For example, when one high-criticality task overruns its C_i^L, low-criticality task τ_5 may decrease its execution budget from 30 to 10, while low-criticality task τ_6 is executed without degradation. By this way, the service degradation of τ_5 results in utilization reduction of $\frac{1}{10}$ to accommodate one high-criticality overrun. The dropping-off process is summarized in Table 3.

TABLE 3
Low-Criticality Service Levels

Number of Overrun k	1	2	3	4
Utilization u_{LO}^k	0.3	0.2	0.1	0
Execution Budget of τ_5	10	0	0	0
Execution Budget of τ_6	75	60	30	0

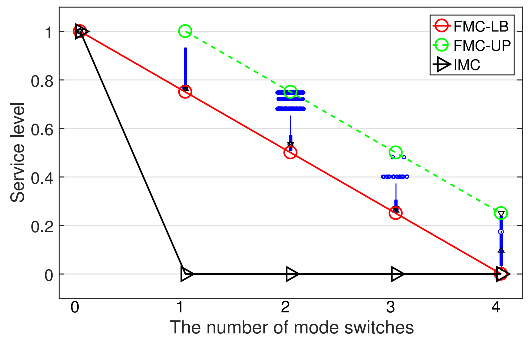


Fig. 4. Service level of low-criticality tasks under the different number of mode switches.

- For uniform tuning strategy by restricting $z_i^k = z^k$ for all low-criticality tasks, each overrun will result in an identical reduction of 0.25 in z_k, such that the condition Eqn. (29) is satisfied. Therefore, the service level z_k for operation in k-level high-criticality mode can be expressed as $z_k = 1 - 0.25 \cdot k$.
- By contrast, if one were to apply IMC [19] to this task set, the guaranteed service level would be 0. This means that any overrun would result in the dropping off of all low-criticality tasks.

Next, we evaluate the implementation of the FMC-EDF-VD run-time system in our simulation framework to demonstrate the graceful low-criticality service degradation of FMC. In this case study, the uniform tuning strategy is applied for demonstration. We ran the simulation for 2×10^6 time units, which contains 5×10^4 high-criticality jobs. We set the high-criticality job behavior probability to 0.1. The simulation process is detailed in Appendix C, available online.

Fig. 4 shows the run-time service levels for both FMC and IMC [19]. The lower bounds on the service levels with different numbers of mode switches, as discussed above, for FMC and IMC are represented by red and black lines, respectively, in Fig. 4. The dashed green line represents service level z^{k-1} for operation in $(k-1)$-level high-criticality mode for FMC. The collected run-time service levels as scheduled by FMC are represented in the form of box-whisker plots with blue dots.

As shown in Fig. 4, FMC can gracefully degrade the low-criticality service level as the number of mode switches increases. By contrast, IMC fails to respond to the variability in the workload. As long as not all high-criticality tasks overrun during run time, the execution budget determined by FMC always outperforms that of IMC.

Another interesting observation is that the collected run-time service levels are bounded by the red and green lines. This observation matches the FMC execution semantics presented in Section 3.2. In the k^{th} transition phase, the execution budget for low-criticality jobs will be reduced from $z^{k-1} \cdot C_i^{LO}$ to $z^k \cdot C_i^{LO}$. According to the FMC execution semantics, two cases can be considered:

- **Case 1:** Low-criticality jobs that have already exhausted their execution budget of $z^k \cdot C_i^{LO}$ at the transition point. Such jobs will be suspended immediately. In addition, these suspended jobs should have an execution time of less than $z^{k-1} \cdot C_i^{LO}$ by the k^{th} transition point. Otherwise, these jobs would have already been suspended when the system entered $(k-1) - level$ high-criticality mode. Therefore, the execution time of these jobs will be bounded in $[z^k \cdot C_i^{LO}, z^{k-1} \cdot C_i^{LO})$.
- **Case 2:** Low-criticality jobs that have not yet exhausted their execution budget $z^k \cdot C_i^{LO}$. Such jobs will continue to run until their remaining time budget is used up. Therefore, these jobs will execute up to $z^k \cdot C_i^{LO}$.

From the above two cases, we can conclude that the execution time of these jobs in k-level high-criticality mode is bounded in $[z^k C_i^{LO}, z^{k-1} C_i^{LO})$, as clearly shown in Fig. 4.

6.4 Run-Time Complexity

According to [4], for a task set containing n tasks, the classic EDF-VD algorithm has a run-time complexity of $O(log\, n)$ per event for job arrival, job completion, and mode switching. Compared with EDF-VD [4], FMC-EDF-VD needs to implement only one additional operation during mode switching, that is, tuning the service levels for low-criticality tasks according to the specified strategy. For the uniform tuning strategy, the uniform service level z_k can be directly computed with a complexity of $O(1)$ according to Theorem 6. For the dropping-off strategy, the dropping-off task can be determined via binary search with a complexity of $O(log\, n)$. Therefore, FMC-EDF-VD still has a run-time complexity of $O(log\, n)$ per event.

7 Evaluation

In this section, simulation experiments are presented to evaluate the performance of FMC. Our experiments are based on randomly generated MC tasks. We randomly generate task sets using the same approach as in [4], [12]. The various parameters are set as follows:

- The period T_i of each task is an integer drawn uniformly at random from $[20, 150]$.
- For each task τ_i, low-criticality utilization u_i^{LO} is a real number drawn at random from $[0.05, 0.15]$.
- R_i denotes the ratio of u_i^{HI}/u_i^{LO}, which is a real number drawn uniformly at random from $[2, 3]$.
- pCri denotes the probability that a task τ_i is a high-criticality task, and we set this probability to 0.5. If τ_i is a low-criticality task, then we set $C_i^{LO} = \lfloor u_i^{LO} \cdot T_i \rfloor$. Otherwise, we set $C_i^{LO} = \lfloor u_i^{LO} \cdot T_i \rfloor$ and $C_i^{HI} = \lfloor u_i^{LO} \cdot R_i \cdot T_i \rfloor$.

Given the utilization bound u_B, we generate one task at a time until the following conditions are both satisfied: (1) $u_B - 0.05 \leq \max\{u_{LO}^{LO} + u_{HI}^{LO}, u_{HI}^{HI}\} \leq u_B$. (2) At least 3 high-criticality tasks have been generated.

The generated task set is evaluated for both off-line schedulability and on-line performance in terms of support for low-criticality task execution under six different schemes. These schemes include FMC with dropping off strategy proposed in this paper('FMC'), Pfair-based scheme with using task grouping [22] ('PF'), component-based scheme [12] ('COM'), advanced EDF-VD scheduling of IMC systems [19] ('IMC'), service adaption strategy that decreases the dispatch frequency of low-criticality tasks based on EDF-VD scheduling [16] ('sA'), classic EDF-VD scheduling [4] ('EDF-VD').

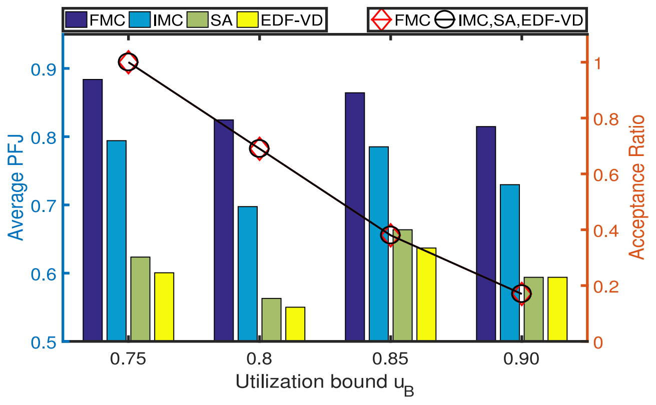


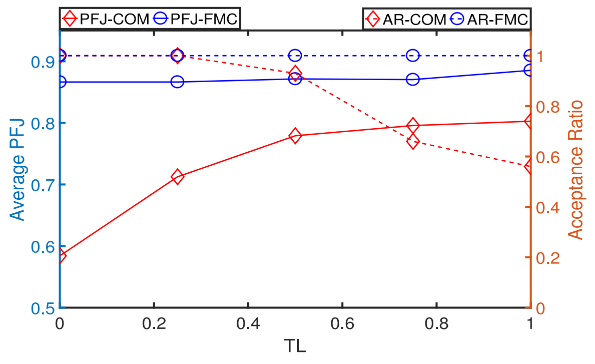
Fig. 5. Comparison between FMC and schemes based on the global triggering.

The on-line low-criticality performance is measured as the percentage of finished LC jobs (denoted by PFJ), which is the same quantitative parameter used in [12]. PFJ is defined as the percentage of low-criticality jobs that are successfully finished by their deadlines. Each simulation is run for 10^6 time units. The execution distribution presented in [23] is used to compute the probability that a high-criticality task τ_i will be executed beyond its low-criticality WCET. Due to schedulability performance differences among the compared schemes, the PFJ is obtained only when the task-set is schedulable for all compared schemes. The simulation process is detailed in Appendix C, available online.

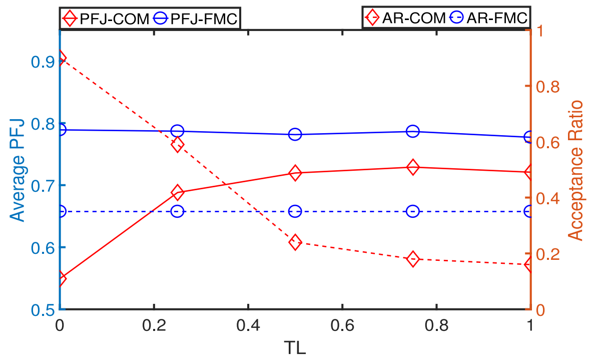
7.1 Comparison with Schemes Based on the Global Triggering Strategy

First, we demonstrate the effectiveness of FMC compared with the IMC, SA, and classic EDF-VD schemes, which use the global triggering strategy. In these three schemes, *any* overrun will trigger low-criticality tasks to statically reserve a constant degraded service level. For IMC and FMC, we consider the mandatory utilization $U_{LO}^{man} = 0$ for the schedulability test. The schedulability test for the SA scheme [16] is a utilization-based test. Therefore, the IMC, SA, and classic EDF-VD schemes have the same schedulability. However, for some schedulable task sets, the SA scheme [16] cannot derive a suitable factor y to increase the period of low-criticality tasks. For this case, we consider y to be infinity, which means that all low-criticality jobs will be dropped when an overrun occurs.

For various utilization bounds $u_B \in \{0.75, 0.8, 0.85, 0.9\}$, the average PFJ and system schedulability are compared. The results are shown in Fig. 5. The left axis shows the PFJ values achieved for low-criticality tasks, represented by the bar graphs, and the right axis shows the acceptance ratios, represented by the line graphs. From Fig. 5, we can observe the following trends: (1) FMC consistently outperforms the three other schemes in terms of support for low-criticality task execution. This is expected because schemes that use the global triggering strategy always consider the worst-case overrun workload, resulting in waste of unnecessary resources. By contrast, FMC can allocate resources based on the true overrun demands. (2) Compared with these three schemes based on the global triggering strategy, FMC achieves almost the same acceptance ratio. This means that FMC can achieve higher on-line low-criticality performance with negligibly reduced schedulability performance.



(a) $u_B = 0.75$



(b) $u_B = 0.85$

Fig. 6. Comparison between component-based scheme and FMC.

7.2 Comparison with the Pfair- and Component-Based Schemes

Next, we will experimentally compare our approach to Pfair- and component-based schemes: PF [22] and COM [12]. For the component-based scheme COM [12], we use the same experiment setting as [12] and consider a two-component system with a high-criticality component $\mathbb{C}_{\mathbb{H}}$ and a low-criticality component $\mathbb{C}_{\mathbb{L}}$. All the high-criticality tasks are allocated to $\mathbb{C}_{\mathbb{H}}$. Each low-criticality task can be allocated to either $\mathbb{C}_{\mathbb{H}}$ or $\mathbb{C}_{\mathbb{L}}$.[6] Since the performance of the scheme presented in [12] depends on a tolerance limit TL, we generate the result of component-based scheme [12] for various values of the tolerance limit $TL = \{0, \lfloor 0.25|H|\rfloor, \lfloor 0.5|H|\rfloor, \lfloor 0.75|H|\rfloor, |H|\}$, where $|H|$ denotes the number of high-criticality tasks. For the Pfair-based scheme [22], a two-phased scheduling strategy[7] is implemented for comparison.

Comparison with Component-Based Scheme COM [12]. The performance results of COM and FMC are presented in Figs. 6a and 6b with different settings on u_B. In these figures, x-axis denotes the varying value of TL, whereas the left and right y-axis present the average PFJ and acceptance ratio, respectively. As shown in Fig. 6, FMC consistently outperforms COM in terms of support for low-criticality execution. This performance gain is achieved by the fact that COM adopts pessimistic dropping-off strategies in internal and

6. Since the work presented in [12] does not specify the settings for low-criticality tasks, we specify one probability to determine if a low-criticality task is allocated to $\mathbb{C}_{\mathbb{H}}$. Here, we choose a relatively low value for probability and set it as 0.25.

7. The version used for evaluation in [22].

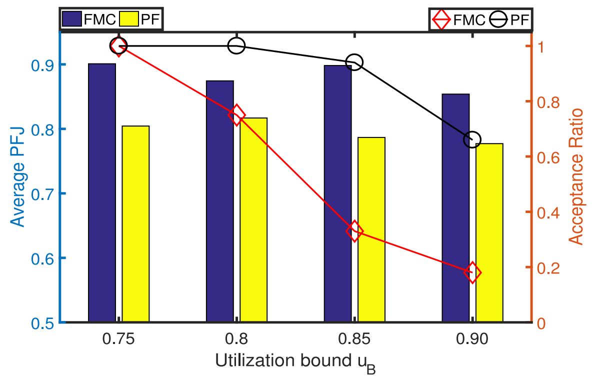


Fig. 7. Comparison between Pfair-based scheme and FMC.

external mode-switch levels. In COM, all low-criticality tasks in $\mathbb{C}_{\mathbb{H}}$ will be abandoned once *any* overrun occurs, which thus results in resource under-utilization. As a comparison, FMC drops off low-criticality tasks as the demand and therefore can achieve better execution support for low-criticality tasks. Besides, we can observe that there is a performance trade-off between PFJ and acceptance ratio in COM. The reason for this trend is that a higher *TL* in COM requires additional resources to support low-criticality executions but generally implies lower schedulability, and the converse also holds. When we consider the TL configuration on which the same scheduability performance can be achieved by FMC and COM, FMC can support more than 25 and 15 percent low-criticality tasks to finish the deadline compared to COM under different u_B settings, respectively.

Comparison with Pfair-Based Scheme PF [22]. Fig. 7 shows the compared results for FMC and PF. Compared with PF, FMC can achieve a better execution support for low-criticality tasks but with inferior schedulability, as shown in Fig. 7. The reason for the gain in low-criticality task execution support is that the Pfair scheduling tends to evenly distribute the quanta of tasks over time, resulting in more unfinished jobs at mode-switching points.

Regarding schedulability inferiority, we mainly attribute this expected inferiority to the theoretical optimality of Pfair scheduling in terms of schedulability performance [14]. In fact, this optimality is achieved at the cost of a high scheduling overhead by quantum-length sub-tasks partitioning and the enforcement of proportional progress. In fact, this

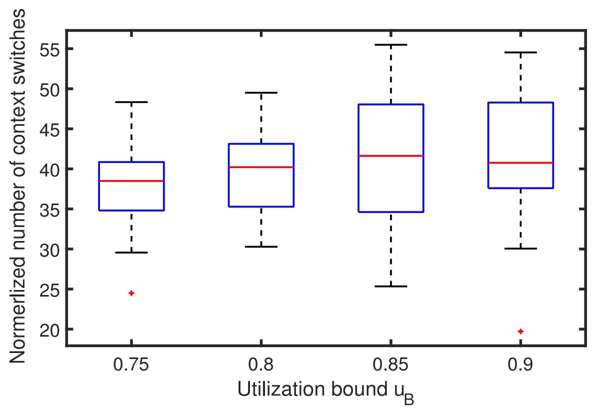


Fig. 8. Context switches for FMC and Pfair-based scheme.

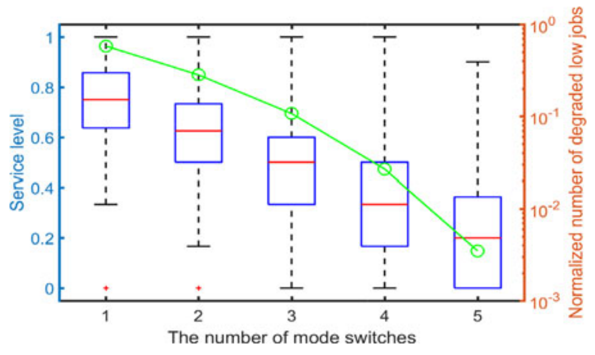


Fig. 9. Service degradation in generic simulation.

schedulability deficit of FMC can be compensated by significantly reduced context-switching overheads compared with PF. Here, we present simulation results to show the compared context-switch numbers. Fig. 8 presents the number of context switches for the Pfair-based scheme, which is normalized with respect to the number for FMC. The results confirm the significant reduction of context switches by FMC. The Pfair-based scheme requires 38.0 to 41.3 times the number of context switches required in FMC for different utilization settings.

7.3 Graceful Service Level Degradation

In the case study presented above, we have demonstrated the gradual service level degradation property of FMC, that is, the degradation in the service levels for low-criticality tasks as the number of mode switch increases. Now, we validate this trend in a generic simulation. The uniform tuning strategy is applied to randomly generated task sets. We use the task generator introduced above to generate 100 task sets in which u_B is randomly selected from $[0.75, 0.9]$. A generated task set is accepted for simulation when the following two additional conditions are satisfied: (1) the task set can be scheduled by FMC, and (2) the task set contains 5 high-criticality tasks. The degraded low-criticality jobs under various task sets are classified according to the number of mode switches.

The simulation results are shown in Fig. 9. The left and right y-axis present the service level and the normalized number of service-degraded low-criticality jobs, respectively. To reveal the distribution of service levels for service-degraded low-criticality jobs, the service levels are represented in the form of box-whisker plots. The results shown in Fig. 9 confirm the observations made in Section 6.3. For almost all low-criticality task jobs, the graceful degradation property is clearly demonstrated except for a few corner cases. Furthermore, the results in terms of the percentages of service-degraded low-criticality jobs also confirm that the likelihood of all high-criticality tasks exhibiting the high-criticality behavior is very low. Only 0.35 percent of service-degraded low-criticality jobs are affected by this worst-case overrun scenario. By contrast, 96.9 percent of service-degraded low-criticality jobs are impacted by mode-switch scenarios with mode switches ≤ 3. For this vast majority of cases, FMC needs to allocate additional resources to only a subset of the high-criticality tasks based on their demands and therefore can provide better and more graceful service degradation.

8 Conclusion and Future Work

Most previous theoretical work on scheduling in mixed-criticality systems has adopted impractical assumptions: once any high-criticality task overruns, all low-criticality tasks are suspended and all other high-criticality tasks are required to exhibit high-criticality behaviors. In this paper, we propose a more flexible MC model (FMC) with EDF-VD scheduling, in which the above issues are addressed. In this model, the transitions of all high-criticality tasks are independent and the service levels of low-criticality tasks can be adaptively tuned in accordance with the true overruns of the high-criticality tasks. A utilization-based schedulability test condition is successfully derived for the FMC systems. Numerical results are presented to illustrate the improved service levels for low-criticality tasks during run time.

For the next step, we are interested in implementing the proposed approach on real-time operating system and evaluating its performance. Furthermore, another interesting future work includes investigations on: (1) integration of FMC and fault tolerance techniques to develop optimal resource allocation strategies for assurances against different types of faults; (2) integrating the slack reclamation schemes into FMC for further performance improvement.

Acknowledgments

This work was partly supported by the National Natural Science Foundation of China (Grant No. 61702085 and 61532007), the RGC of Hong Kong (Grant No. ECS-25204216 and GRF-15204917), and the Fundamental Research Funds for the Central Universities (Grant No. N161604002). Furthermore, we would also like to acknowledge Gu Xiaozhe in Nanyang Technological University for his helps on the implementation of component-based scheme.

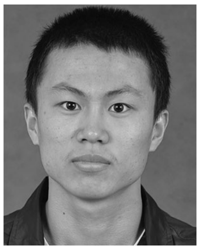
References

[1] S. Baruah, et al., "Towards the design of certifiable mixed-criticality systems," in *Proc. 16th IEEE Real-Time Embedded Technol. Appl. Symp.*, 2010, pp. 13–22.
[2] S. Baruah, et al., "Mixed-criticality scheduling of sporadic task systems," in *Proc. 19th Eur. Conf. Algorithms*, 2011, pp. 555–566.
[3] S. Baruah, et al., "Response-time analysis for mixed criticality systems," in *Proc. IEEE 32nd Real-Time Syst. Symp.*, 2011, pp. 34–43.
[4] S. Baruah, et al., "The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems," in *Proc. 24th Euromicro Conf. Real-Time Syst.*, 2012, pp. 145–154.
[5] S. Baruah et al., "Preemptive uniprocessor scheduling of mixed-criticality sporadic task systems," *J. ACM*, vol. 62, no. 2, 2015, Art. no. 14.
[6] P. Binns, "Incremental rate monotonic scheduling for improved control system performance," in *Proc. 3rd IEEE Real-Time Technol. Appl. Symp.*, 1997, pp. 80–90.
[7] A. Burns, et al., "Towards a more practical model for mixed criticality systems," in *Proc. 1st Int. Workshop Mixed Criticality Syst.*, 2013, pp. 1–6.
[8] A. Burns et al., "Mixed criticality systems-a review," University of York, Heslington, York, U.K., 2015.
[9] H. Chishiro, et al., "Practical imprecise computation model: Theory and practice," in *Proc. IEEE 17th Int. Symp. Object/Component/Service-Oriented Real-Time Distrib. Comput.*, 2014, pp. 198–205.
[10] W. Feng, et al., "An extended imprecise computation model for time-constrained speech processing and generation," in *Proc. IEEE Workshop Real-Time Appl.*, 1993, pp. 76–80.
[11] C. A. Floudas., *Deterministic Global Optimization: Theory, Methods and Applications*. Berlin, Germany: Springer, 2005.
[12] X. Gu, et al., "Resource efficient isolation mechanisms in mixed-criticality scheduling," in *Proc. 27th Euromicro Conf. Real-Time Syst.*, 2015, pp. 13–24.
[13] C. C. Han, et al., "A fault-tolerant scheduling algorithm for real-time periodic tasks with possible software faults," *IEEE Trans. Comput.*, vol. 52, no. 3, pp. 362–372, Mar. 2003.
[14] P. Holman, et al., "Group-based pfair scheduling," *Real-Time Syst.*, vol. 32, pp. 125–168, 2006.
[15] P. Huang, et al., "Interference constraint graph: A new specification for mixed-criticality systems," in *Proc. IEEE 18th Conf. Emerging Technol. Factory Autom.*, 2013, pp. 1–8.
[16] P. Huang, et al., "Service adaptions for mixed-criticality systems," in *Proc. 19th Asia South Pacific Des. Autom. Conf.*, 2014, pp. 125–130.
[17] ISO 26262:Road vehicles. (2012). [Online]. Available: http://www.iso.org/iso/
[18] K.-J. Lin, et al., "Imprecise results: Utilizing partial comptuations in real-time systems," in *Proc. Real-Time Syst. Symp.*, 1987.
[19] D. Liu, et al., "Edf-vd scheduling of mixed-criticality system with degraded quality guarantees," in *Proc. IEEE 32nd Real-Time Syst. Symp.*, 2016, pp. 35–46.
[20] J. W. S. Liu, W. K. Shih, K. J. Lin, R. Bettati, and J. Y. Chung, "Imprecise computations," in *Proc. IEEE*, 1994, vol. 82, no. 1, pp. 83–94.
[21] R. Rajkumar, et al., "A resource allocation model for QoS management," in *Proc. IEEE Real-Time Syst. Symp.*, 1997, pp. 298–307.
[22] J. Ren, et al., "Mixed-criticality scheduling on multiprocessors using task grouping," in *Proc. 27th Euromicro Conf. Real-Time Syst.*, 2015, pp. 25–34.
[23] F. Santy, L. George, P. Thierry, and J. Goossens, "Relaxing mixed-criticality scheduling strictness for task sets scheduled with FP," in *Proc. 24th Euromicro Conf. Real-Time Syst.*, 2012, pp. 155–165.
[24] H. Su, et al., "An elastic mixed-criticality task model and its scheduling algorithm," in *Proc. Des. Autom. Test Europe Conf. Exhib.*, 2013, Art. no. 28.
[25] H. Su, et al., "Service guarantee exploration for mixed-criticality systems," in *Proc. IEEE 20th Int. Conf. Embedded Real-Time Comput. Syst. Appl.*, 2014, pp. 1–10.
[26] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *Proc. 28th IEEE Int. Real-Time Syst. Symp.*, 2007, pp. 239–243.
[27] D. Zhu, et al., "Multiple-resource periodic scheduling problem: How much fairness is necessary?" in *Proc. 24th IEEE Real-Time Syst. Symp.*, 2003, pp. 142–151.

Gang Chen received the BE degree in biomedical engineering, in 2008, the BS degree in mathematics and applied mathematics, in 2008, the MS degree in control science and engineering, in 2011, from Xian Jiaotong University, China, and the PhD degree in computer science from Technical University of Munich, Germany, in 2016. He is currently an associate professor with Northeastern University, China. His research interests include mixed-criticality system, energy-aware real-time scheduling, certifiable cache architecture design, and high-performance computing.

Nan Guan received the PhD degree from Uppsala University, Sweden, in 2013. He is currently an assistant professor with Hong Kong Polytechnic University. His research interests lie in the fields of safe-critical cyber-physical systems, including real-time scheduling theory, worst-case execution time analysis and formal verification techniques. He received the European Design Automation Association (EDAA) Outstanding Dissertation Award in 2014, Best Paper Award of IEEE Real-Time Systems Symposium (RTSS) in 2009, Best Paper Award of Design Automation and Test in Europe conference (DATE) in 2013, and Best Poster Award in the PhD forum of IEEE International Parallel and Distributed Processing Symposium (IPDPS) in 2012 and in IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA) in 2017.

Di Liu received the BA and MS degrees both from Northwestern Pyrotechnical University, China, in 2011, and the PhD degree from Leiden University, in 2017. He is an assistant professor of National Pilot School of Software, Yunnan University, China. His research interests lie in the fields of real-time systems, mixed-criticality systems, and system-level multi-core systems design.



Qingqiang He received the BS degree in computer science and technology from Northeastern University, China, in 2014, and the master's degree in computer software and theory from Northeastern University, China, in 2017. Now he is working as a research assistant with Hong Kong Polytechnic University. His research interests include embedded real-time system, scheduling analysis in mixed-criticality system, and real-time interrupt response technology on xen-based virtualization platform.

Kai Huang received the BSc degree from Fudan University, China, in 1999, the MSc degree from University of Leiden, The Netherlands, in 2005, and the PhD degree from ETH Zurich, Switzerland, in 2010, and He joined Sun Yat-Sen University as a professor in 2015. He was appointed as the director of the Institute of Cyber Physical Systems of School of Data and Computer Science in 2016. He was a senior researcher in the Computer Science Department, the Technical University of Munich, Germany from 2012 to 2015 and a research group leader in fortiss GmbH in Munich Germany in 2011. His research interests include techniques for the analysis, design, and optimization of embedded systems, and particularly in the automotive domain. He was the recipient of Best Paper Awards ESTIMedia 2013, SAMOS 2009, and General Chairs' Recognition Award For Interactive Papers in CDC 2009.

Todor Stefanov received the Dipl. Ing and MS degrees in computer engineering from The Technical University of Sofia, Bulgaria, in 1998 and the PhD degree in computer science from Leiden University, The Netherlands, in 2004. Currently, he is an associate professor in the Leiden Institute of Advanced Computer Science at Leiden University and the head of the Leiden Embedded Research Center (LERC). He is a recipient of the prestigious 2009 IEEE TCAD Donald O. Pederson Best Paper Award in the *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*. Moreover, he serves (has served) on the organizational committees of several leading conferences, symposia, and workshops, such as DATE, ACM/IEEE CODES+ISSS, RTSS, IEEE ICCD, IEEE/IFIP VLSI-SoC, and SAMOS.

Wang Yi received the PhD in computer science from Chalmers University of Technology, Sweden, in 1991. He is a chair professor with Uppsala University. His interests include models, algorithms and software tools for building and analyzing computer systems in a systematic manner to ensure predictable behaviors. He was awarded with the CAV 2013 Award for contributions to model checking of real-time systems, in particular the development of UPPAAL, the foremost tool suite for automated analysis and verification of real-time systems. For contributions to real-time systems, he received Best Paper Awards of RTSS 2015, ECRTS 2015, DATE 2013 and RTSS 2009, Outstanding Paper Award of ECRTS 2012 and Best Tool Paper Award of ETAPS 2002. He is on the steering committee of ESWEEK, the annual joint event for major conferences in embedded systems areas. He is also on the steering committees of ACM EMSOFT (co-chair), ACM LCTES, and FORMATS. He serves frequently on Technical Program Committees for a large number of conferences, and was the TPC chair of TACAS 2001, FORMATS 2005, EMSOFT 2006, HSCC 2011, LCTES 2012 and track/topic Chair for RTSS 2008 and DATE 2012-2014. He is a member of Academy of Europe (Section of Informatics) and a fellow of the IEEE.