EDF-VD Scheduling of Flexible Mixed-Criticality System with Multiple-Shot Transitions

ABSTRACT

The existing mixed-criticality real-time task models assume that once any high-criticality task overruns, *all* high-criticality jobs execute up to their *most pessimistic WCET estimations simultaneously in a one-shot manner.* This is very pessimistic in the sense of unnecessary resource overbooking. In this paper, we propose a more generalized mixed-critical real-time task model, called FMC-MST, to address this problem. In FMC-MST, high-criticality tasks can transit multiple intermediate levels to handle less pessimistic overruns independently and to non-uniformly scale the deadline on each level. We develop a run-time schedulability analysis for FMC-MST under EDF-VD scheduling, in which a better trade-off between the penalties of low-criticality tasks and the overruns of high-criticality tasks is achieved to improve the service quality of low-criticality tasks. We also develop a resource optimization technique to find resource-efficient level-insertion configurations for FMC-MST task systems under MC timing constraints. Experiments demonstrate the effectiveness of FMC-MST compared with the state-of-the-art techniques.

1. INTRODUCTION

Integrating applications with different criticality levels on a shared computing platform has increasingly become a common trend in the design of real-time embedded systems. Such a trend has been observed in the automotive [11] and avionics [16] industries and has led to the emergence of mixed-criticality (MC) systems. An MC task model was proposed by Vestal in his seminal paper [19] about ten years ago, wherein different WCETs are specified for each task on all existing criticality levels, with the one on a higher criticality level being more pessimistic. Since then, many techniques for analyzing and scheduling MC systems have been proposed in the real-time literature (see [6] for a comprehensive review). However, these approaches proposed in nearly a decade still share very impractical assumptions on MC task execution behavior. Specifically, once any high criticality task overruns, the following behaviors are assumed:

- First, all low-criticality tasks are abandoned. It is pessimistic to immediately abandon all low-criticality tasks because low-criticality tasks require a certain timing performance as well [11, 18].
- Second, all high-criticality tasks are assumed to exhibit high criticality behaviors. It is overly pessimistic to bind the mode switches of all high-criticality tasks together in the analysis, as the mode switches of high-criticality tasks are naturally independent.
- Third, high-criticality tasks are directly transited to the most pessimistic level. This will result in unnecessary resource overbooking because high-criticality tasks rarely reach its most pessimistic WCET estimation during run-time.

Table 1: Comparison of the existing solutions.

	P_1 Graceful Degradation	P_2 Independent Mode-Switches	P_3 Multiple-Shot Transition
[18, 5, 14]	✓	×	×
[9, 17, 8, 13]	✓	✓	×
[3, 4]	×	×	✓
Our Work	✓	✓	✓

Related Work. Some solutions have been proposed to partly resolve the above problems. In Tab. 1, we summarize the existing solutions in relation to the three problems described above. These solutions can be broadly categorized into the following classes. The first category of research offers low-criticality tasks a certain degraded service quality when the system is in high-criticality mode. Assumptions of abandoning all low-criticality tasks are relaxed by reducing the dispatch frequency of jobs [18] or by reducing the execution budget of jobs [5, 14]. However, these studies still apply a pessimistic mode-switch strategy.

To address the first and second problems, the second category of studies offer solutions for improving performance for low-criticality tasks by using group-based mode-switch strategies [9, 17]. However, these mode-switch strategies are not flexible enough because the dependencies between low-criticality and high-criticality tasks are statically determined. To relax such dependencies, a new MC model, called flexible MC (FMC) model, was recently proposed in [8], where mode-switches of high-criticality tasks are independent and the service degradation of low-criticality is dynamically updated based on the overruns of high-criticality tasks. Lee et al. [13] proposed an MC-ADAPT framework supporting online adaptive task dropping under task-level mode switch that involves using a similar technique. However, the third problem is not addressed in these two state-of-the-art work. In [8, 13], high-criticality tasks always directly transit to the *most* pessimistic level, in which very pessimistic design parameters are applied.

To support multi-shot transitions, EDF-VD scheduling algorithm is extended to support a K-level implicit-deadline task system in [3, 4]. However, the K-level MC task model in [3, 4] still applies impractical assumptions. Specifically, when the system switches the mode to level k, all the tasks of criticality at least k are assumed to exhibit k-level criticality behaviors (i.e., assumption P_2). All other tasks of criticality less than k are discarded (i.e., assumption P_1).

To the best of our knowledge, no work to date has addressed the above three problems collectively. Compared to existing studies, the motivation of this work is to find a more fine-grained transition scheme for overrun handling that captures the varying execution behaviors of high-criticality tasks. Instead of always transiting to the *most* pessimistic level, the proposed MC system can undergo intermediate levels to handle overruns with less pessimistic design parameters, such that unnecessary resource overbooking can be avoided. By doing so, a better run-time trade-off between the penalty of low-criticality tasks and the overruns of high-criticality tasks can be achieved to improve the service quality of low-criticality tasks.

Contributions. In this paper, we propose a flexible MC model with multiple-shot transitions (denoted as FMC-MST) operating on a uni-processor platform. Rather than always switching to the *most* pessimistic level (the strategy used in [8, 13]), the new model allows each high-criticality task to progress over multiple less pessimistic intermediate levels and to scale the deadline non-uniformly on each criticality level. Since high-criticality tasks rarely reach their pessimistic WCET estimations, FMC-MST can present unnecessary resource overbooking for overruns by switching high-criticality tasks to less pessimistic intermediate levels. Furthermore, FMC-MST provides a fine-grained transition scheme where mode-switches are independent with these intermediate criticality levels. The overrun of a high-criticality task only raises its own criticality level while others remain at their previous criticality levels. The minimum required low-criticality service degradation is calculated to maintain the balanced system utilization, so as to secure the additional resources requested by a level-transiting task. The contributions of this work can be summarized as follows:

- We propose a new EDF-VD based scheduling for an MC model with multiple-shot transition schemes. Compared to the state-of-the-art work [8, 13], our work provides a more generalized flexible MC model that allows high-criticality tasks to progress through multiple criticality levels and to scale deadlines non-uniformly.
- We develop a run-time schedulability analysis for each independent mode-switch. To improve the service quality of low-criticality tasks, the utilization balance between low-criticality and high-criticality tasks serves as a basic principle for finding an optimal service degradation strategy for low-criticality tasks to compensate for the additional resources requested by multi-shot overruns of high-criticality tasks.
- We formally prove the correctness of run-time schedulability analysis for this fine-grained transition scheme.
- We develop a resource optimization technique that can find resource-efficient level-insertion configurations for FMC-MST task systems under MC timing constraints.

Our evaluation on randomly generated task systems shows that the performance of FMC-MST outperforms the state-of-the-art MC scheduling approaches.

2. BACKGROUND

2.1 FMC implicit-deadline sporadic task model with multiple criticality levels

We consider an MC sporadic task system γ as consisting of a finite collection $\{\tau_1, \tau_2, \cdots, \tau_n\}$ of n MC implicit-deadline sporadic tasks with multiple criticality levels. Each task τ_i in γ generates an infinite sequence of jobs and can be specified by a tuple $\{T_i, \chi_i, \mathcal{C}_i\}$, where:

- T_i is the minimum job-arrival intervals.
- χ_i is the total number of criticality levels.
- $\mathcal{C}_i = (C_i(0), C_i(1), \cdots, C_i(\chi_i - 1))$ is a vector of the worst-case execution times (WCETs). We assume that $C_i(0) \le C_i(1) \le \cdots \le C_i(\chi_i - 1)$.

For the classic dual-criticality system, high-criticality task has two criticality levels with $\chi_i = 2$ and low-criticality task has one criticality level with $\chi_i = 1$. In this paper, we consider an extended dual-criticality task system in which the concepts of high-criticality task and low-criticality task are presented as follows:

Definition 1. *In an MC system with multiple criticality levels, tasks with $\chi_i >= 2$ and $\chi_i = 1$ are called high-criticality and low-criticality tasks, respectively.*

According to Def. 1, we can divide task set γ into low-criticality task set γ_L and high-criticality task set γ_H. In an MC system with multiple criticality levels, high-criticality tasks are allowed to have several overrun scenarios during run-time. We denote l_i as the criticality level whereby τ_i stays during run-time, and we have $l_i = \{0, 1, 2, \cdots, \chi_i - 1\}$. The mode-switch from level $l_j - 1$ to level l_j can be defined as follows:

Definition 2. *(Mode-switch $M_j^{l_j}$ and $\hat{M}_j^{l_j}$). When high-criticality task τ_j executes for its $C_j(l_j - 1)$ time units without signaling completion, high-criticality task τ_j immediately switches from level $l_j - 1$ to level l_j. This procedure is denoted as mode-switch $M_j^{l_j}$. The closest mode-switch*[1] *occurring before $M_j^{l_j}$ is denoted as $\hat{M}_j^{l_j}$. For the special case of $l_j = 0$, M_j^0 denotes high-criticality task τ_j executes at level* 0*.*

In FMC-MST, each mode-switch $M_j^{l_j}$ is independent. Mode-switch $M_j^{l_j}$ does not require other high-criticality tasks to exhibit high-criticality behavior. For low-criticality tasks, their execution budget is updated dynamically in accordance with $M_j^{l_j}$. To model the degradation of low-criticality tasks on the point of mode-switch $M_j^{l_j}$, we now introduce the concept of the service level as follows:

Definition 3. *(Service level $z_i(M_j^{l_j})$). When the system has undergone mode switch $M_j^{l_j}$, up to $z_i(M_j^{l_j}) \cdot C_i(0)$ time units can be used for the execution of τ_i in one period T_i.*

In this paper, we consider implicit-deadline task systems with task period being equal to the relative deadline (i.e, $T_i = d_i$). The utilization of a task denotes the ratio of its WCET to its period. We define the utilization of task τ_i at level l_i as:

$$u_i(l_i) = \frac{C_i(l_i)}{T_i} \qquad l_i = \{0, 1, 2, \cdots, \chi_i - 1\}$$

The total utilization of low-criticality task set in the initial mode (i.e., all high-criticality tasks stay at crticality level 0) is defined as $u_L(0) = \sum_{\tau_i \in \gamma_L} u_i(0)$. According to Def. 3, the degraded utilization of low-criticality tasks on mode-switch $M_j^{l_j}$ can be defined as $u_L(M_j^{l_j}) = \sum_{\tau_i \in \gamma_L} z_i(M_j^{l_j}) \cdot u_i(0)$.

For the remainder of the manuscript, we assume that the condition of $z_i(M_j^{l_j}) \le z_i(\widehat{M_j^{l_j}})$ should hold to accommodate the resource overbooking of mode-switch $M_j^{l_j}$. Correspondingly, the system utilization reduction $\Delta u_L(M_j^{l_j})$ of low-criticality tasks on mode-switch $M_j^{l_j}$ can be computed as $u_L(M_j^{l_j}) - u_L(\widehat{M_j^{l_j}})$.

Remark 1. *Note that, in FMC-MST, $\Delta u_L(M_j^{l_j})$ is off-line determined to guarantee a schedulable MC system (see Section 3.3). In general, we do not need to specify the*

[1] Note that, this closest mode-switch is triggered by a specific task, which is known to system.

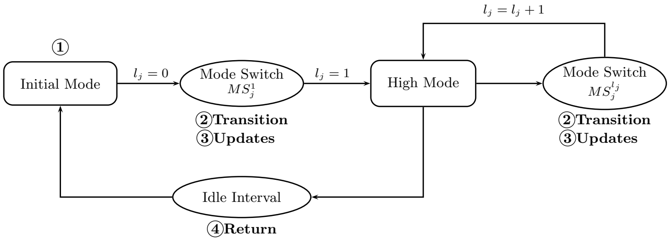

Figure 1: Execution semantics.

settings of $z_i(M_j^{l_j})$ *during off-line stage. Any on-line strategy on tuning* $z_i(M_j^{l_j})$ *can be applied as long as it can achieve the required utilization reduction* $\Delta u_L(M_j^{l_j})$.

2.2 EDF-VD scheduling with nonuniform virtual deadlines

In this paper, we study the schdulability for FMC-MST tasks model under EDF-VD scheduling. The main idea of EDF-VD is to use reduced virtual deadlines to obtain extra slack time for jobs and further decrease the workload of high-criticality tasks after mode-switch.

In EDF-VD [2], the virtual deadlines are uniformly scaled by a single deadline scaling factor x and can be defined uniformly by $d_j^v = x \cdot d_j$. In FMC-MST, we allow *non-uniform* deadline scaling factor $x_j^{l_j}$, where $x_j^{l_j} \in (0, 1)$ is a task and criticality level dependent scaling parameter , to non-uniformly set the virtual deadline as $d_j^v(l_j) = x_j^{l_j} \cdot d_j$.

2.3 Execution semantic

The execution semantic of a high-criticality task is illustrated in Fig. 1. Compared to the classic MC execution model, FMC-MST model allows independent mode-switches for high-criticality tasks and dynamic service tuning for low-criticality tasks. As shown in Fig. 1, the system initially operates at level 0 (i.e., ①). An overrun of a high-criticality task only triggers itself to shift its criticality level (i.e., ②) and degrades low-criticality service to accommodate this overruns (i.e., ③). A sequence of overruns trigger the system to proceed through multiple criticality levels one by one independently (i.e., ② and ③) until the condition for transiting back is satisfied (i.e., ④). The execution semantics can be summarized as follows:

① **Initial mode**: All tasks in γ start in level 0 (i.e., $\forall \tau_i, l_i = 0$). As long as no high-criticality task violates its $C_i(0)$, the system remains in level 0. All tasks are scheduled with $C_i(0)$.

② **Transition**: When one job of a high-criticality task τ_j *that is being executed in level* $l_j - 1$ overruns its $C_j(l_j - 1)$ without signaling completion, τ_j only trigger itself to switches into level l_j and update virtual deadline as $d_j^v(l_j)$. However, all other high-criticality tasks still stay in the **same** criticality level as before.

③ **Updates**: To balance the additional resource demand caused by mode-switch $M_j^{l_j}$, a new service level $z_i(M_j^{l_j})$ is determined and updated to provide degraded service for low-criticality tasks τ_i. At this updating instant, if any low-criticality jobs have completed more than $z_i(M_j^{l_j}) \cdot c_i(0)$ time units of execution, 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.

④ **Return to low-criticality mode**: When the system detects an idle interval [5], the system transits back to the low-criticality mode.

2.4 An Illustrative Example

Table 2: Example Task Set

	χ_i	T_i	$C_i(0)/d_i^v(0)$	$C_i(1)/d_i^v(1)$	$C_i(2)/d_i^v(2)$
τ_1	1	6	3		
τ_2	3	15	3/10	4.5/12.5	5.75/15
τ_3	3	10	1/5	1.5/8	2.5/10

Table 3: Degraded Utilization

Mode Swithch	M_2^1	M_2^2	M_3^1	M_3^2
$\Delta u_L(M_j^{l_j})$	$-\frac{1}{6}$	$-\frac{1}{6}$	$-\frac{1}{12}$	$-\frac{1}{12}$
Budget Reduction	-1	-1	-0.5	-0.5

Now, we give an example to illustrate the related concepts and execution semantics of FMC-MST . Tab. 2 gives three tasks, one low-criticality task ($\chi_1 = 1$) and two high-criticality tasks ($\chi_2 = 3$ and $\chi_3 = 3$). For high-criticality tasks, each criticality level l_j ($j = 2, 3$) associates with one virtual deadline $d_j^v(l_j)$, where $l_j \in \{0, 1, 2\}$. Tab. 3 gives the required utilization degradation $\Delta u_L(M_j^{l_j})$ for each mode switch to guarantee a schedulable MC system[2]. Fig. 2 depicts the scheduling of MC tasks under execution semantics of FMC-MST , where the symbol ∇ is used to indicate mode-switch occurrence point. In Fig. 2, the jobs are operated under the following rules:

(1) Low-criticality task is scheduled with their real deadlines. In Fig. 2, τ_1 are scheduled with $d_1 = 6$.

(2) At each mode switch point ∇, operation ② is triggered to update the virtual deadline while operation ③ is triggered to update the execution budget. Now we take the first mode switch as example for illustration. At $t = 1$, the first mode-switch M_3^1 occurs. τ_3 switches its criticality level from $l_3 = 0$ to $l_3 = 1$ with extended virtual deadline $d_3^v(1) = 8$, while τ_2 stay in the same criticality level as before (i.e., ②). This deadline extension (i.e.,$d_3^v(1) = 8$) results in the preemption of τ_1 at $t = 1$. The execution budget of low-criticality task τ_1 are decreased from 3 to 2.5 to achieve the required $\Delta u_L(M_3^1)$. τ_1 completes its execution at time instant 3.5 due to using up the budget (i.e., ③).

(3) During a busy interval in which multiple overruns occur, the effects of the overruns on budget reduction are independent. For example, during $[0, 15]$, three mode switches ($M_3^1 \triangleright M_2^1 \triangleright M_2^2$) occurs sequentially. By Tab. 3, the required budget reduction can be simply calculated as the sum of the one of these three mode switches, that is -2.5. Therefore, τ_1 only left 0.5 time unit for execution.

3. SCHEDULABILITY ANALYSIS AND RESOURCE OPTIMIZATION

Our FMC-MST model is a more generalized model that allows multiple less pessimistic criticality levels and non-uniform deadline scaling. In this section, we present a utilization-based schedulability analysis for FMC-MST scheduling algorithm. We first analyze online schedulability for a single mode switch $M_j^{l_j}$, by which the minimum low-criticality service degradation can be derived to accommodate the resource overbooking of a mode switch. In Section 3.1, we provide a high-level overview for this online schedulability analysis and attempt to communicate the intuition behind the algorithm design by means of an example. We then provide a more comprehensive

[2] The derivations for determining $\Delta u_L(M_j^{l_j})$ are illustrated in Ex. 1.

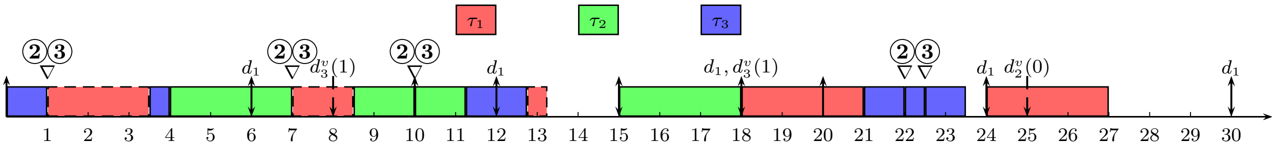

Figure 2: Illustrative example

description in Section 3.2 to prove the correctness of Thm. 1. In Section 3.3, we check whether a task set is schedulable by FMC-MST under arbitrary sequences of mode switches. In Section 3.4, we develop an intermediate level insertion technology and attempt to solve the problem of how to determine intermediate levels for high-criticality tasks to minimize the penalties of low-criticality tasks without sacrificing MC schedulability. We finally prove some important properties of FMC-MST. Tab. 4 shows the notation used throughout this paper.

Table 4: Table of Notations

Symbol	Meaning in the paper
$x_j^{l_j}$	Virtual deadline factor of task τ_j at level l_j
$u_j(l_j)$	Utilization of task τ_j at level l_j
$u_L(M_j^{l_j})$ $(u_L(\hat{M}_j^{l_j}))$	Total utilization of low-criticality tasks after (before) mode switch $M_j^{l_j}$
$\Delta u_L(M_j^{l_j})$	Utilization reduction of low-criticality tasks required to accommodate mode switch $M_j^{l_j}$
γ_H^H	mode-switched task set $\gamma_H^H = \{\tau_j \in \gamma_H \mid l_j \geq 1\}$
γ_H^L	non-mode-switched task set $\gamma_H^L = \{\tau_j \in \gamma_H \mid l_j = 0\}$
$a_j^{l_j}$ $(d_j^{l_j})$	Absolute release time (deadline) of the job of high-criticality τ_j that switches to $M_j^{l_j}$

3.1 Sufficient schedulability test on transition case $M_j^{l_j}$

In this section, we provide a high-level overview of online schedulability analysis for one mode switch, and introduce the derived schedulability test condition in Thm. 1. With these conditions, we can adaptively determine how much of execution budget can be reserved for low-criticality tasks to handle each intermediate overrun while ensuring a schedulable system during run-time. Without loss of generality, we consider a general transition case $M_j^{l_j}$ where high-criticality task τ_j switches from level $l_j - 1$ to l_j, and assume the system is MC-schedulable on level $l_j - 1$. To accommodate $M_j^{l_j}$, the minimum required utilization reduction $\Delta u_L(M_j^{l_j})$ can be determined by Thm. 1.

Theorem 1. *For mode-switch $M_j^{l_j}$ with $l_i \geq 1$, when high-criticality task τ_j overruns its $C_j(l_j - 1)$, the system is schedulable when the following conditions are satisfied:*

$$\Delta u_L(M_j^{l_j}) + \frac{u_j(l_j)}{x_j^{l_j}} - \frac{u_j(l_j-1)}{x_j^{l_j-1}} \leq 0 \tag{1}$$

$$\Delta u_L(M_j^{l_j}) + \frac{u_j(l_j) - u_j(l_j-1) + p_j(l_j)}{1 - x_j^{l_j-1}} \leq 0 \tag{2}$$

$$\Delta u_L(M_j^{l_j}) \leq 0 \tag{3}$$

$$\frac{u_j(l_j)}{x_j^{l_j}} \leq u_j(\chi_j - 1) \tag{4}$$

where $p_j(l_j)$ are constrained by:

$$p_j(l_j) \leq 0 \tag{5}$$

$$\sum_{l_j=1}^{\chi_j-1} p_j(l_j) = u_j(0) - \frac{u_j(0)}{x_j^0} \tag{6}$$

with the initial utilization condition on criticality level 0:

$$u_L(0) + \sum_{\tau_j \in \gamma_H} \frac{u_j(0)}{x_j^0} \leq 1 \tag{7}$$

$$\frac{u_j(0)}{x_j^0} \leq u_j(\chi_j - 1) \tag{8}$$

Intuition. The intuition behind Thm. 1 is to maintain balanced system utilization during the transitions. The conditions can be explained as follows. Eqn. (7) ensures MC schedulability when the system stays in initial mode [2]. According to [13], high-criticality task with $\frac{u_j(l_j)}{x_j^{l_j}} \geq u_j(\chi_j - 1)$ will produce schedulability loss. We use contraints Eqn. (4) and Eqn. (8) to avoid the performance loss during transitions. An event of overrun of high-criticality task normally results in an increase in virtual and overrun utilization due to resource overbooking. By analyzing the difference in virtual and overrun utilization, Eqn. (1) and Eqn. (2) serve as an efficient way to represent the resource balance between the penalty of low-criticality tasks and the overruns of high-criticality tasks. In order to provide an intuition of how the proposed analysis works, we apply Thm. 1 on a simple task set and calculate the required utilization degradation for guaranteeing MC schedulability of a single mode switch.

Table 5: Feasible Settings

Mode Swithch	M_2^1	M_2^2	M_3^1	M_3^2
$p_j(l_j)$	$-\frac{7}{45}$	$-\frac{1}{18}$	$-\frac{1}{60}$	$-\frac{1}{12}$
$x_j^{l_j-1}$	$\frac{2}{3}$	$\frac{5}{6}$	$\frac{1}{2}$	$\frac{4}{5}$

Example 1. *Consider a task set in Tab. 2. Feasible settings*[3] *on $p_j(l_j)$ and $x_j^{l_j}$ is listed in Tab. 5, so that conditions Eqns. (4)-(8) are satisfied. In the following, we take the mode switch M_2^1 as an example to illustrate the derivation process of the required utilization degradation $\Delta u_L(M_2^1)$. According to Eqns. (1)-(3) in Thm. 1, utilization degradation $\Delta u_L(M_2^1)$ should satisfy the following conditions to accommodate a feasible mode switch M_2^1.*

$$\Delta u_L(M_2^1) \leq \min\left(\underbrace{-(\frac{u_2(1)}{x_2^1} - \frac{u_2(0)}{x_2^0})}_{virtual\ utilization}, \underbrace{-\frac{u_2(1) - u_2(0) + p_2(1)}{1 - x_2^0}}_{overrun\ utilization}, 0\right)$$
$$= -\frac{1}{6}$$

The similar derivation can be operated to obtain utilization degradation for other mode switches.

3.2 Proof of the correctness

We now prove the correctness of the schedulability test condition presented in Thm. 1. The proof process involves three steps. We first determine the initial conditions to ensure the schedulability of tasks in initial mode (Eqn. (7)) and to satisfy the boundary constraints (Eqn. (3), Eqn. (4),

[3] Feasible settings can be off-line determined by formulated CSP problem presented in Section 3.3.

and Eqn. (8)). In the second step, we prove the correctness of the sufficient condition (i.e., Eqn. (1)) to ensure MC schedulability after mode switch $M_j^{l_j}$. In the third step, we propose a sufficient schedulability condition (i.e., Eqn. (2)) to maintain balanced overrun utilization as the system undergoes mode transition $M_j^{l_j}$.

3.2.1 Initial conditions

The basic assumption $z_i(M_j^{l_j}) \leq z_i(\hat{M}_j^{l_j})$ implies Eqn. (3). According to [2], we can use Eqn. (7) to ensure MC schedulability of in level 0. Eqn. (4) and Eqn. (8) restrict resource utilization to levels less than those achieved in the most pessimistic level (i.e., level $\chi_j - 1$). Otherwise, tasks can directly execute in level $\chi_j - 1$ for efficient resource use [13].

3.2.2 Virtual utilization balance equation

We now show how to ensure MC schedulability after mode switch $M_j^{l_j}$ occurs. This is achieved via virtual utilization balance analysis before and after mode switch $M_j^{l_j}$. By replacing the period as virtual deadline, virtual utilization of each high-criticality task τ_j on level l_j is computed as $\frac{u_j(l_j)}{x_j^{l_j}}$. $u_\gamma^v(\hat{M}_j^{l_j})$ and $u_\gamma^v(M_j^{l_j})$ denote the virtual utilization of task set γ before and after mode switch $M_j^{l_j}$, respectively. To ensure the correctness of system behaviors after mode switch $M_j^{l_j}$, system virtual utilization $u_\gamma^v(M_j^{l_j})$ must meet the following condition:

$$u_\gamma^v(M_j^{l_j}) = u_L(M_j^{l_j}) + \sum_{\tau_j \in \gamma_H} \frac{u_j(l_j)}{x_j^{l_j}} \leq 1 \tag{9}$$

After mode switch $M_j^{l_j}$, high-criticality task τ_j overruns $C_j(l_j - 1)$ and shifts from level $l_j - 1$ to level l_j. With the exception of high-criticality task τ_j, all other high-criticality tasks remain at their respective criticality levels without changing the utilization. Therefore, an increase in the virtual utilization of high-criticality tasks can be determined as $\frac{u_j(l_j)}{x^{l_j}} - \frac{u_j(l_j-1)}{x_j^{l_j-1}}$. For low-criticality tasks, low-criticality utilization is degraded from $u_L(\hat{M}_j^{l_j})$ to $u_L(M_j^{l_j})$ due to resource overbooking of overruns. Therefore, the difference in system virtual utilization can be formulated as:

$$\begin{aligned}
&u_\gamma^v(M_j^{l_j}) - u_\gamma^v(\hat{M}_j^{l_j}) \\
=&\underbrace{u_L(M_j^{l_j}) - u_L(\hat{M}_j^{l_j})}_{Utilization\ Reduction} + \underbrace{\frac{u_j(l_j)}{x_j^{l_j}} - \frac{u_j(l_j - 1)}{x_j^{l_j-1}}}_{Utilization\ Increment} \\
=&\underbrace{\Delta u_L(M_j^{l_j}) + \frac{u_j(l_j)}{x_j^{l_j}} - \frac{u_j(l_j - 1)}{x_j^{l_j-1}}}_{Eqn.\ (1)}
\end{aligned} \tag{10}$$

As the system is schedulable before mode switch $M_j^{l_j}$, we have $u_\gamma^v(\hat{M}_j^{l_j}) \leq 1$. Hence, we find that Eqn. (1) ensures the correctness of $u_\gamma^v(M_j^{l_j}) \leq u_\gamma^v(\hat{M}_j^{l_j}) \leq 1$ to guarantee MC schedulability after the mode-switch.

3.2.3 Overrun utilization balance equation

As the third step, we prove that the condition presented in Eqn. (2) is sufficient to ensure the MC schedulability during the transition phase. We adopt the similar proof strategy based on [3, 8] and prove it by contradiction. Suppose that there is a time interval $[0, t_f]$ such that the system undergoes mode switch $M_j^{l_j}$ and the first deadline miss occurs at t_f. Let J denote the minimal set[4] of jobs released from task set γ for which a deadline is missed. $\eta_i^{l_j}(t_1, t_2)$ denotes cumulative execution time of task τ_i when the system undergoes the mode-switch $M_j^{l_j}$ during the interval $(t_1, t_2]$. $N_\gamma^{l_j}$ denotes the sum of $\eta_i^{l_j}(0, t_f)$ for all tasks in γ. Since the first deadline miss occurs at t_f, we have $N_\gamma^{l_j} > t_f$. In the following, we will show the upper bound of $N_\gamma^{l_j}$ is less than t_f, which leads to a contradiction.

To calculate the upper bound of $N_\gamma^{l_j}$, we start the proof by introducing auxiliary lemmas to analyze the upper bound of cumulative execution time for high-criticality tasks (i.e., Lem. 1 and Lem. 2) and low-criticality tasks (i.e., Lem. 3).

High Criticality Tasks: Since the mode switches are independent, high-criticality tasks can be divided into mode-switched task set γ_H^H and non-mode-switched task set γ_H^L. Now, we derive upper bounds of the cumulative execution time for both types of high-criticality tasks.

Lemma 1. *For high-criticality task τ_j of task set γ_H^H, the cumulative execution time $\eta_j^{l_j}(0, t_f)$ can be bounded by:*

$$\frac{a_j^1}{x_j^0} \cdot u_j(0) + (t_f - a_j^1)u_j(1) + \sum_{r_j=2}^{l_j} (t_f - a_j^{r_j})\Delta u_j(r_j) \tag{11}$$

where $\Delta u_j(r_j) = u_j(r_j) - u_j(r_j - 1)$.

Proof. Recall that $a_j^{r_j}$ is the absolute release time of the job executed on level r_j. High-criticality task τ_j progresses though l_j levels. Therefore, the analysis duration can be divided into $l_j + 1$ time segments, as shown in Fig. 3. During time segment $[a_j^{r_j}, a_j^{r_j+1}]$, the execution requirement per job is bounded by $c_j(r_j)$. For ease of presentation, we use $a_j^{l_j+1} = t_f$. Considering l_j time segments shown in Fig. 3, the cumulative execution time $\eta_j^{l_j}(0, t_f)$ can be bounded as:

$$\begin{aligned}
\eta_j^{l_j}(0, t_f) &\leq a_j^1 \cdot u_j(0) + \sum_{r_j=1}^{l_j} (a_j^{r_j+1} - a_j^{r_j}) \cdot u_j(r_j) \\
&\leq \frac{a_j^1}{x_j^0} u_j(0) + (a_j^2 - a_j^1)u_j(1) + \underline{\sum_{r_j=2}^{l_j} (a_j^{r_j+1} - a_j^{r_j}) \cdot u_j(r_j)}
\end{aligned} \tag{12}$$

Since $u_j(r_j) = \sum_{k=2}^{r_j}(u_j(k) - u_j(k-1)) + u_j(1)$, we have:

$$\begin{aligned}
&\sum_{r_j=2}^{l_j} (a_j^{r_j+1} - a_j^{r_j}) \cdot u_j(r_j) \\
=&(a_j^{r_j+1} - a_j^2)u_j(1) + \sum_{r_j=2}^{l_j} \sum_{k=2}^{r_j} (a_j^{r_j+1} - a_j^{r_j})(u_j(k) - u_j(k-1)) \\
=&(a_j^{r_j+1} - a_j^2)u_j(1) + \sum_{k=2}^{l_j} \sum_{r_j=k}^{l_j} (a_j^{r_j+1} - a_j^{r_j})(u_j(k) - u_j(k-1)) \\
=&(a_j^{r_j+1} - a_j^2)u_j(1) + \sum_{k=2}^{l_j} (a_j^{l_j+1} - a_j^k)(u_j(k) - u_j(k-1))
\end{aligned} \tag{13}$$

Substituting the marked item in Eqn. (12) with Eqn. (13), $\eta_j^{l_j}(0, t_f)$ can be reformulated as:

$$\frac{a_j^1}{x_j^0} u_j(0) + (a_j^{l_j+1} - a_j^1)u_j(1) + \sum_{k=2}^{l_j} (a_j^{l_j+1} - a_j^k)(u_j(k) - u_j(k-1)) \tag{14}$$

Therefore, $\eta_j^{l_j}(0, t_f)$ can be bounded as Eqn. (11) by replacing $a_j^{l_j+1}$ and k with t_f and r_j, respectively. □

Lemma 2. *(From [8]) High-criticality task τ_j in task set γ_H^L has*

$$\eta_j^0(0, t_f) \leq \frac{t_f}{x_j^0} u_j(0) \tag{15}$$

[4] This minimality means that if any job is removed from J, the remainder of J will be schedulable.

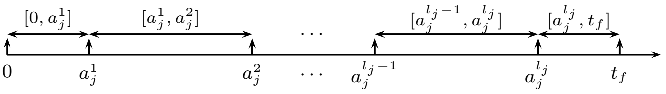

Figure 3: Time segments.

Low Criticality Tasks: We now derive an upper bound on the cumulative execution time $\eta_i^{l_j}(0,t_f)$ for low-criticality tasks using a proof strategy similar to that used in [8].

Lemma 3. *For low-criticality task τ_i, the cumulative execution time $\eta_i^{l_j}(0,t_f)$ can be upper bounded by*

$$t_f \cdot u_i(0) + \sum_{\tau_j \in \gamma_H^H} \sum_{r_j=1}^{l_j} \psi_i^{r_j} \tag{16}$$

with difference term $\psi_i^{r_j} = (t_f - a_j^{r_j})(1 - x_j^{r_j - 1})\Delta u_i(M_j^{r_j})$.

Proof. We will only sketch the proof here as it is similar to the proof in [8]. The detailed proof is presented in Appendix-A. Following the proof strategy in [8], we analyze the difference of the cumulative execution time before and after mode-switch $M_j^{l_j}$ and prove that the difference can be uniformly upper bounded by *difference term* $\psi_i^{l_j}$. By visiting all mode switches $M_j^{r_j}$, the upper bound of $\eta_i^{l_j}(0,t_f)$ can be obtained. □

Total Cumulative Requirements: Now, we sum the cumulative requirements over all tasks given as Eqn. (17) and prove the sufficient condition Eqn. (2). The complete derivation of $N_\gamma^{l_j}$ is given in Appendix-B.

$$N_\gamma^{l_j} = \sum_{\tau_i \in \gamma_L} \eta_i^{l_j}(0,t_f) + \sum_{\tau_j \in \gamma_H^L} \eta_j^{0}(0,t_f) + \sum_{\tau_j \in \gamma_H^H} \eta_j^{l_j}(0,t_f) \tag{17}$$

$$\leq t_f + \sum_{\tau_j \in \gamma_H^H} (t_f - a_j^1) \left((1 - x_j^0)\Delta u_L(M_j^1) + \Delta u_j(1) + \underbrace{u_j(0) - \frac{u_j(0)}{x_j^0}}_{Eqn.\ (6)} \right)$$

$$+ \sum_{\tau_j \in \gamma_H^H} \sum_{r_j=2}^{l_j} (t_f - a_j^{r_j}) \Big((1 - x_j^{r_j-1})\Delta u_L(M_j^{r_j}) + \Delta u_j(r_j) \Big)$$

$$= t_f + \sum_{\tau_j \in \gamma_H^H} (t_f - a_j^1) \left((1 - x_j^0)\Delta u_L(M_j^1) + \Delta u_j(1) + \sum_{l_j=1}^{\chi_j - 1} p_j(l_j) \right)$$

$$+ \sum_{\tau_j \in \gamma_H^H} \sum_{r_j=2}^{l_j} (t_f - a_j^{r_j}) \Big((1 - x_j^{r_j-1})\Delta u_L(M_j^{r_j}) + \Delta u_j(r_j) \Big)$$

$$Since\ a_j^1 \leq a_j^2 \leq \cdots \leq a_j^{l_j} < t_f\ and\ p_j(r_j) \leq 0$$

$$\leq t_f + \sum_{\tau_j \in \gamma_H^H} \sum_{r_j=1}^{l_j} (t_f - a_j^{r_j}) \Big((1 - x_j^{r_j-1})\Delta u_L(M_j^{r_j}) + \Delta u_j(r_j) + p_j(r_j) \Big)$$

The assumed deadline miss implies $N_\gamma > t_f$. That is:

$$\sum_{\tau_j \in \gamma_H^H} \sum_{r_j=1}^{l_j} (t_f - a_j^{r_j}) \Big((1 - x_j^{r_j-1})\Delta u_L(M_j^{r_j}) + \Delta u_j(r_j) + p_j(r_j) \Big) > 0$$

Taking the contrapositive, we have:

$$\sum_{\tau_j \in \gamma_H^H} \sum_{r_j=1}^{l_j} (t_f - a_j^{r_j}) \left(\underbrace{(1 - x_j^{r_j-1})\Delta u_L(M_j^{r_j}) + \Delta u_j(r_j) + p_j(r_j)}_{Eqn.\ (2)} \right) \leq 0 \tag{18}$$

Since $t_f - a_j^{r_j} > 0$, it is sufficient to ensure the system schedulability of task set γ by guaranteeing Eqn. (2) holds for each mode switch $M_j^{r_j}$. In Eqn. (18), the constraints imposed on each mode switch $M_j^{r_j}$ are consistent to each other. Based on this property, the constraints imposed on current mode switch $M_j^{l_j}$ imply the condition Eqn. (2), guaranteeing MC schedulability during the transition phase.

3.3 Feasibility of algorithm

Thm. 1 gives an online schedulability test condition *only* for a single transition. It is yet unclear how to off-line determine whether a task set is schedulable by FMC-MST under arbitrary sequences of mode switches. In this section, we present the off-line schedulability test conditions for a task set with specified criticality levels. To guarantee schedulability, we must ensure that FMC-MST can successfully schedule the task set under any execution scenario during run-time. Therefore, to show that the task set is MC-schedulable, we need to satisfy the following two conditions:

Condition A: We need to guarantee the feasibility of each mode-switch. Therefore, constraints (1)-(8) for each mode-switch must be satisfied.

Condition B: We must ensure the system-wide feasibility. As shown in Thm. 1, each overrun will result in a decreased low-criticality utilization. For low-criticality tasks, we must show remaining low-criticality utilization should not fall below a level of 0 under the worst-case overrun scenario, that is each high-criticality task τ_j reaches criticality level $\chi_j - 1$. Therefore, we require:

$$\sum_{\tau_j \in \gamma_H} \sum_{l_j=1}^{\chi_j - 1} \Delta u_L(M_j^{l_j}) + u_L(0) \geq 0 \tag{19}$$

For condition A, constraints (1)-(5) must be subjected to all mode switches with $\forall \tau_j \in \gamma_H$ and $l_j = 1, \cdots, \chi_j - 1$, while constraint (6) should be subjected to all high tasks with $\forall \tau_j \in \gamma_H$. By combining all of these conditions, we can formulate the offline schedulability problem as a constraint satisfaction problem (CSP). Any insertion solution of intermediate levels whose states satisfy a number of constraints in the derived CSP problem can guarantee a feasible scheduling system. We use the following example to illustrate how to evaluate the schedulability of the given insertion solution.

Example 2. *Consider the example task set with the dedicated insertion solution given in Tab. 2 and the settings listed in Tab. 5. We have already demonstrated condition A is satisfied, as illustrated in Ex. 1. For condition B, we know it is also satisfied by simply checking:*

$$\sum_{\tau_j \in \gamma_H} \sum_{l_j=1}^{2} \Delta u_L(M_j^{l_j}) + u_L(0) = -\frac{1}{6} - \frac{1}{6} - \frac{1}{12} - \frac{1}{12} + \frac{3}{6} = 0$$

Therefore, the example task set in Tab. 2 is MC-schedulable.

3.4 Resource optimization

Above, we prove a metric for evaluating the schedulability of an MC task set with specified level-insertion configurations. However, for an MC task set with two bounded criticality levels (i.e., $C_j(0)$ and $C_j(\chi_i - 1)$ are known), how to specify a reasonable level-insertion configuration for each high-criticality task is still not known yet. In this section, we will study the off-line resource optimization problem with the aim of finding the resource-efficient level-insertion configuration for the FMC-MST task system within MC timing constraints.

In general, the probability that the execution time of high-criticality task reaches its most pessimistic WCET estimation is quite low. However, in EDF-VD scheduling, high-criticality tasks always transit from low-criticality level to the *most* pessimistic level once an overrun occurs. To avoid unnecessary resource over-booking, we can insert several intermediate levels to handle the less pessimistic overruns. The intermediate level to take depends on the real execution time of high-criticality tasks. In this paper, we use the distribution of the execution time of high-criticality task τ_j to compute the probability of overruns. The cumulative distribution function $F_j(t)$ is used to model the diversity of execution time of high-criticality task τ_j during run-time. Hence, the probability of the overrun $M_j^{l_j}$ that the execution time of high-criticality task τ_j falls in $[c_j(l_j), c_j(l_j+1)]$ can be represented as $F_j(c_j(l_j+1)) - F_j(c_j(l_j))$. When high-criticality task reaches criticality level l_j, the utilization of low-criticality tasks require to decrease $\sum_{r_j=0}^{l_j} \Delta u_L(M_j^{r_j})$. In the off-line stage, we introduce a QoS function Eqn. (20) with the aim to minimize the average low-criticality utilization decrease. Based on this objective and the aforementioned constraints, the resource optimization problem (ROP) is formulated as:

$$\textbf{ROP: min} \sum_{\tau_j \in \gamma_H} \sum_{l_j=1}^{\chi_j - 1} (F_j(c_j(l_j+1)) - F_j(c_j(l_j))) \sum_{r_j=0}^{l_j} \Delta u_L(M_j^{r_j}) \tag{20}$$

$$s.t. \begin{cases} \textbf{Condition A} : Eqn.\ (1) - (8)\ for\ all\ mode\ switches \\ \textbf{Condition B} : Eqn.\ (19) \end{cases}$$

The objective function shown above is subjected to the constraints listed in the above CSP formulation (conditions A and B). Given an MC task set where two bounded execution times $[C_j(0), c_j(\chi_j - 1)]$ are specified for each high-criticality task, the developed resource optimization formulation can automatically generate a feasible level-insertion configuration with intermediate execution time $c_j(l_j)$ and deadline scaling factor $x_j^{l_j}$ for each high-criticality task.

Complexity. Due to non-linear items in the constraints, the ROP (Eqn. (20)) is a nonlinear optimization problem (NLP). For a task set with M high-criticality tasks and L criticality levels, then NLP problem has $4M(L-1)+M+2$ constraints and $4M(L-2)+3$ real variables. Hence, the number of variables and constraints is polynomially bounded to the size of the input problem, and it can be solved by a polynomial-time heuristic [10].

Properties. We now provide important properties to show the efficiency of FMC-MST.

Property 1. *Criticality level insertions operated by ROP do not degrade the schedulability of FMC-MST.*

Proof. We consider a general case that a task set is MC-schedulable by FMC-MST with L criticality levels. ROP formulation generates level-insertion configuration $[\Delta u_L(M_j^{l_j}), u_j(l_j), x_j^{l_j}, p_j(l_j)]$ for each criticality level l_j of high-criticality task τ_j. In general, without changing the previous configurations of L levels, one can insert $L+1^{th}$ level with the following configuration:

$$\Delta u_L(M_j^{l_j+1}) = 0, u_j(l_j+1) = u_j(l_j), x_j^{l_j+1} = x_j^{l_j}, p_j(l_j+1) = 0 \tag{21}$$

The new configuration still satisfies the CSP. Therefore, the task set is still MC-schedulable. □

Property 2. *FMC-MST with* 2 *criticality levels dominates EDF-AD-E [13] in terms of MC-schedulability.*

Proof. For FMC-MST with 2 criticality levels (i.e., $\chi_j = 2$), $u_j(0)$ and $u_j(1)$ are equivalent to low-criticality and high-criticality utilization in EDF-AD-E, respectively. For task set γ, the high-criticality task set γ_H can be divided into HI-mode-preferred task set $\gamma_H^F = \{\tau_j \in \gamma_{HI} | \frac{u_j(0)}{x_j^0} \geq u_j(1)\}$ and non-HI-mode-preferred task set $\gamma_H - \gamma_H^F$, respectively. Assume task set γ is MC-schedulable by EDF-AD-E [13]. Therefore, the following conditions must be satisfied to ensure MC schedulability according to [13].

$$u_L(0) + \min_{\tau_j \in \gamma_H} (\frac{u_j(0)}{x}, u_j(1)) \leq 1 \tag{22}$$

$$x \cdot u_L(0) + u_H(1) \leq 1 \tag{23}$$

In general, we can always find a lower-bound factor $\hat{x}$ that satisfies $u_L(0) + \min_{\tau_j \in \gamma_{HI}} (\frac{u_j(0)}{\hat{x}}, u_j(1)) = 1$ and Eqn. (23).

To achieve equivalent behavior, we assign $x_j^0 = \frac{u_j(0)}{u_j(1)}$ for HI-mode-preferred tasks and $\hat{x}$ for non-HI-mode-preferred tasks when applying FMC-MST. By this equivalence transformation, we can make the following observations for the CSP formulation:

- Eqn. (7) and Eqn. (22) are equivalent.
- $\Delta u_L(M_j^0) = 0$ holds for HI-mode-preferred tasks.
- For non-HI-mode-preferred tasks, the constraints Eqn. (1)-(6) can be equivalently merged as Eqn. (2).

Based on above observations, by Eqn. (2) and Eqn. (19), one can derive Eqn. (23) and guarantee a feasible CSP problem for FMC-MST.

$$-u_L(0) \leq \sum_{\tau_j \in \gamma_H} \Delta u_L(M_j^1) \leq \frac{\sum_{\tau_j \in \gamma_H - \gamma_H^F} \left(\frac{u_j(0)}{\hat{x}} - u_j(1)\right)}{1 - \hat{x}} \tag{24}$$

$$\begin{aligned}
&\rightleftharpoons -u_L(0) \leq \frac{\sum_{\tau_j \in \gamma_H - \gamma_H^F} \left(\frac{u_j(0)}{\hat{x}} - u_j(1)\right)}{1 - \hat{x}} \\
&\rightleftharpoons \hat{x} \cdot u_L(0) + u_H(1) \leq u_L(0) + \sum_{\tau_j \in \gamma_H - \gamma_H^F} \frac{u_j(0)}{\hat{x}} + \sum_{\tau_j \in \gamma_H^F} u_j(1) \\
&\quad From\ the\ definition\ of\ \gamma_H^F\ and\ \gamma_H - \gamma_H^F \\
&\rightleftharpoons \hat{x} \cdot u_L(0) + u_H(1) \leq u_L(0) + \min_{\tau_j \in \gamma_H} (\frac{u_j(0)}{\hat{x}}, u_j(1)) \\
&\quad From\ the\ definition\ of\ \hat{x} \\
&\rightleftharpoons \hat{x} \cdot u_L(0) + u_H(1) \leq 1
\end{aligned}$$

Therefore, we can conclude when any task set γ is MC-schedulable by EDF-AD-E [13], it is also MC-schedulable by FMC-MST with 2 criticality levels. □

Property 3. *FMC-MST with L criticality levels inserted by ROP dominates EDF-AD-E [13] in terms of MC-schedulability.*

Proof. This can be directly proved by Prop. 1 and Prop. 2. □

4. EVALUATION

Experiment Setup. In this section, we conduct the simulation experiments to evaluate the effectiveness of

FMC-MST by an extensive comparison to state-of-the-art approaches: EDF-AD-E [13], FMC [8], IMC [14], EDF-VD [2]. Our experiments were conducted based on randomly-generated MC task systems. We adopt the same workload generation algorithm as that used in [7, 2, 9] to randomly generate task sets with two criticality levels. In FMC-MCL, two criticality levels act as the lowest and highest criticality levels (i.e., $l_j = 0$ and $l_j = \chi_j - 1$). The resource optimization approach presented in Section 3.4 will automatically insert the intermediate levels between these two levels. For ease of presentation, we denote these two criticality levels as LO and HI levels during the generation process. In particular, the various parameters[5] of each task are generated in the following ways:

- For each task τ_i, low-criticality utilization u_i^{LO} is a real number drawn at random from $[0.05, 0.15]$ [6].
- R_i denotes the ratio of u_i^{HI}/u_i^{LO} for every high-criticality task, which is a real number drawn uniformly at random from [1, 5].
- Task period T_i of each task is an integer drawn uniformly at random from $[100, 1000]$.
- pCri denotes the probability that a task τ_i is a high-criticality task, and we set it as 0.5. When τ_i is a low-criticality task, then set $C_i^{LO} = \lfloor u_i^{LO} \cdot T_i \rfloor$. Otherwise, set $C_i^{LO} = \lfloor u_i^{LO} \cdot T_i \rfloor$ and $C_i^{HI} = \lfloor u_i^{LO} \cdot R_i \cdot T_i \rfloor$ [7].

One task is generated at a time until $u_B - 0.05 \le \max\{u_{LO}^{LO} + u_{HI}^{LO}, u_{HI}^{HI}\} \le u_B$.

As stated in Rem. 1, FMC-MST provides a generalized degradation strategy. For the evaluation, we adopt dropping-off strategy where low-criticality tasks are partly dropped by assigning $z_i(M_j^{l_j}) = 0$ for dropped tasks. We quantitatively compare FMC-MST with above state-of-the-art approaches in terms of offline schedulability and online performance. Following [8, 9], online low-criticality performance is measured by the percentage of finished LC jobs (denoted by PFJ). PFJ defines the ratio of the number of finished jobs of LO-critical tasks over the total number of jobs released in a given time interval. During the simulation, the execution distribution with an exponential tail [15] is used to generate the actual execution time for jobs of high-criticality tasks. The system takes the intermediate level according to the actual execution time. To ensure fair comparisons, we generate a job trace for each generated task set in off-line and use this unified job trace to obtain the PFJ for all compared schemes during run-time.

Results. We first demonstrate the effectiveness of FMC-MST compared with state-of-the-art approaches: FMC [8], EDF-AD-E [13], IMC [14], and EDF-VD [2], in which high-criticality tasks always directly enter the most pessimistic execution mode once overrun occurs. We vary utilization bounds u_B from 0.7 to 0.95 with step size of 0.05, to evaluate offline schedulability and online performance. For FMC-MST, each high-criticality task are inserted with 3 intermediate levels. Each data-point was obtained by randomly generating 1000 task sets. Fig. 4 shows the acceptance ratio and average PFJ for the compared approaches. The left axis shows PFJ values achieved for low-criticality tasks represented by the bar graphs, and the right axis shows acceptance ratios represented by the line graphs.

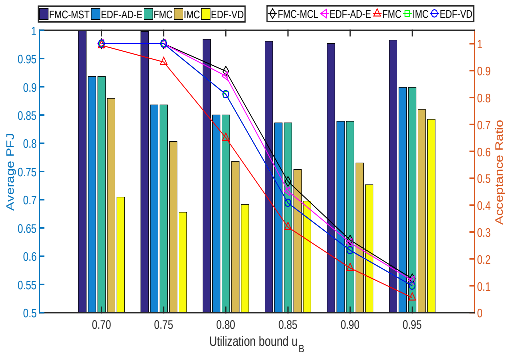

Figure 4: Performance with varying utilization bound.

As shown in Fig. 4, FMC-MST can provide more low-criticality service without sacrifice in the MC schedulability. We can observe the following trends: (1) FMC-MST outperforms all the compared approaches in terms of support for low-criticality execution. This is expected because one-shot transition scheme based approaches always switch to the level with applying the most pessimistic design parameters. In contrast, FMC-MST can capture the varying execution behavior of high-criticality tasks and can penalize low-criticality tasks more precisely according to the overrun demands of high-criticality tasks. (2) FMC-MST dominates all the EDF-VD based scheduling algorithms with one-shot transition scheme. This schedulability performance gain is attributed to the fact that FMC-MST provides a generalized MC model where a non-uniform deadline scaling is a relaxation of EDF-VD based schedulings [13, 8] and might cause more task sets to be deemed schedulable.

Next, we will show how the number of intermediate levels L will impact the effectiveness of FMC-MST. In this experiment, R_i is randomly selected from four different uniform distributions $\{[1, 2], [2, 3], [3, 4], [4, 5]\}$. Varying L from 2 to 11, we conduct the simulation on random MC task sets with $u_B = 0.85$. Fig. 5 shows online low-criticality performance under different settings on L and R_i. As shown in Fig. 5, the average PFJ increases with the number of insertion levels L. The reason for this trend is that the more insertion levels generally imply more opportunities for handling the less pessimistic overruns during run-time, which can avoid overbooking unnecessary resources.

We finally evaluate the computation time for deriving automatic intermediate level insertion by solving the formulated optimization problem presented in Section 3. According to the parameters of task sets presented above, we can automatically generate an optimization problem and use the APMonitor optimization suite [1] to solve it. For all task set tested above, the selected optimization tool can generate results within 8 seconds. The results show that the formulated optimization problem can be solved efficiently.

5. CONCLUSION

We present a generalized flexible MC model that enables independent multiple-shot transitions for high-criticality tasks. A run-time schedulability test condition is successfully derived, which serves as a basis principle to find

[5] We also follows [12] to evaluate the performance under different settings. More results is available online []

[6] In FMC-MCL, u_i^{LO} and u_i^{HI} correspond to $u_i(0)$ and $u_i(\chi_j - 1)$, respectively

[7] In FMC-MCL, C_i^{LO} and C_i^{HI} correspond to $C_i(0)$ and $C_i(\chi_j - 1)$, respectively

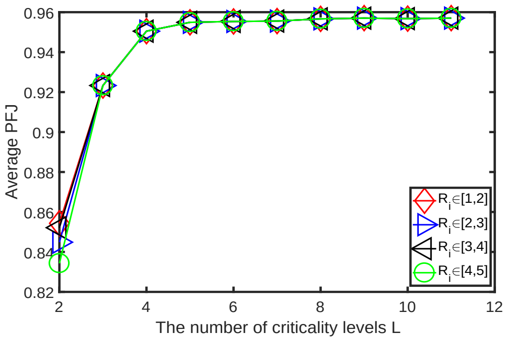

Figure 5: The impact of the number of criticality levels L.

an optimal service degradation strategy for low-criticality tasks. We develop a resource optimization formulation to maximize the run-time low-criticality service quality without sacrificing MC schedulability. Experimental results illustrate the efficiency of the proposed approach.

6. REFERENCES

[1] APMonitor Optimization Suite. http://apmonitor.com/.

[2] S. Baruah et al. The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems. In *2012 24th Euromicro Conference on Real-Time Systems*, 2012.

[3] S. Baruah et al. Preemptive uniprocessor scheduling of mixed-criticality sporadic task systems. *Journal of the ACM*, 2015.

[4] S. Baruah et al. Scheduling of mixed-criticality sporadic task systems with multiple levels. In *12th Workshop on Models and Algorithms for Planning and Scheduling Problems*, 2015.

[5] A. Burns and S. Baruah. Towards a more practical model for mixed criticality systems. In *1st International Workshop on Mixed Criticality Systems*, pages 1–6, 2013.

[6] A. Burns and R. I. Davi. Mixed criticality systems-a review. *University of York, Technical Report*, 2018.

[7] A. Easwaran. Demand-based scheduling of mixed-criticality sporadic tasks on one processor. In *the 2013 IEEE 34th Real-Time Systems Symposium (RTSS)*, 2013.

[8] C. Gang et al. Utilization-based scheduling of flexible mixed-criticality real-time tasks. *IEEE Transaction on Computers*, 2018.

[9] X. Gu et al. Resource efficient isolation mechanisms in mixed-criticality scheduling. In *2015 27th Euromicro Conference on Real-Time Systems*, 2015.

[10] D. S. Hochbaum. Complexity and algorithms for nonlinear optimizationproblems. *Annals of Operations Research*, 2007.

[11] ISO 26262:Road vehicles. `http://www.iso.org/iso/`.

[12] N. Kim et al. Attacking the one-out-of-m multicore problem by combining hardware management with mixed-criticality provisioning. In *2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2016.

[13] J. Lee et al. Mc-adapt: Adaptive task dropping in mixed-criticality scheduling. *ACM Transactions on Embedded Computing Systems*, 2017.

[14] D. Liu et al. Edf-vd scheduling of mixed-criticality system with degraded quality guarantees. In *32nd IEEE Real-Time Systems Symposium*, 2016.

[15] D. Maxim et al. Probabilistic analysis for mixed criticality systems using fixed priority preemptive scheduling. In *Proceedings of the 25th International Conference on Real-Time Networks and Systems (RTNS)*, 2017.

[16] P. J. Prisaznuk. Integrated modular avionics. In *Proceedings of the IEEE 1992 National Aerospace and Electronics Conference*, 1992.

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

[18] H. Su, N. Guan, and D. Zhu. Service guarantee exploration for mixed-criticality systems. In *2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications*, 2014.

[19] S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *2007 28th IEEE International Real-Time Systems Symposium*, 2007.

Appendix-A: Proof of Lem. 3.

Now, we derive a *difference term* to bound the difference of the cumulative execution time obtained before and after mode-switch $M_j^{l_j}$. Due to space limitations, some conclusions in [3, 8] are reused to keep the proof as brief as possible. Following [8], we start the proof by extending the concept of *carry-over job* and several important propositions to FMC-MST model.

Definition 4. *A job of low-criticality task τ_i is called a l_j-carry-over job if mode switch $M_j^{l_j}$ occurs in the interval $[\hat{a}_i^{l_j}, \hat{d}_i^{l_j}]$, where $\hat{a}_i^{l_j}$ and $\hat{d}_i^{l_j}$ are the absolute release time and deadline of this job, respectively.*

With this extended definition of *carry-over job*, we will show the propositions presented in [8] can be also extended to FMC-MST task system. $\hat{t}^{l_j}$ denotes the time instant of mode switch $M_j^{l_j}$ caused by high-criticality task τ_j. The extended propositions can be written as follows:

Proposition 1. *(From [2, 3]) All jobs executed in $[\hat{t}^{l_j}, t_f]$ have a deadline $\leq t_f$.*

Proposition 2. *The mode-switching point $\hat{t}^{l_j}$ satisfies $\hat{t}^{l_j} \leq a_j^{l_j} + x_j^{l_j} \cdot (t_f - a_j^{l_j})$.*

Proof. The virtual deadline $a_j^{l_j} + x_j^{l_j}(t_f - a_j^{l_j})$ must be greater than $\hat{t}^{l_j}$. Otherwise, the high-criticality job would have completed its execution before the mode-switch. □

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

Proof. Similar to the proof of Proposition 3 in [8]. □

Based on above propositions, we now present two auxiliary derivation rules that uniformly specify the upper bounds of $\eta_i^{l_j}(0, t_f)$ and $\eta_i^{l_j}(\hat{a}_i^{l_j}, \hat{d}_i^{l_j})$ for different execution scenarios. Throughout the proof, $\sup\{\eta_i^{l_j}(t_1, t_2)\}$ is used to denote the upper bounds on $\eta_i^{l_j}(t_1, t_2)$ for low-criticality task τ_i under $M_j^{l_j}$. Let $M_p^{l_p}$ denote the last mode-switch occurring before $\hat{a}_i^{l_j}$, which occurs at the time instant $\hat{t}^{l_p}$. $z_i(M_p^{l_p})$ denotes the updated service level at $\hat{t}^{l_p}$. $d_i^{t_f}$ denotes the absolute deadline for the last job of τ_i during $[0, t_f]$. Now, we present rules for deriving $\sup\{\eta_i^{l_j}(0, t_f)\}$ and $\sup\{\eta_i^{l_j}(\hat{a}_i^{l_j}, \hat{d}_i^{l_j})\}$ as follows[8]:

$$\sup\{\eta_i^{l_j}(0, t_f)\} = \begin{cases} \sup\{\eta_i^{l_j}(0, d_i^{t_f})\} + (t_f - \hat{t}^{l_j}) z_i(M_j^{l_j}) u_i(0) & d_i^{t_f} < \hat{t}^{l_j} \\ \sup\{\eta_i^{l_j}(0, \hat{d}_i^{l_j})\} + (t_f - \hat{d}_i^{l_j}) z_i(M_j^{l_j}) u_i(0) & \text{Otherwise} \end{cases} \tag{25}$$

$$\sup\{\eta_i^{l_j}(\hat{a}_i^{l_j}, \hat{d}_i^{l_j})\} = \begin{cases} (\hat{d}_i^{l_j} - \hat{a}_i^{l_j}) z_i(M_p^{l_p}) u_i(0) & \eta_i^{l_j}(\hat{a}_i^{l_j}, \hat{t}^{l_j}) \neq 0 \\ (\hat{d}_i^{l_j} - \hat{a}_i^{l_j}) z_i(M_j^{l_j}) u_i(0) & \text{Otherwise} \end{cases} \tag{26}$$

$M_q^{l_q}$ denotes the closest mode switch[9] before $M_j^{l_j}$ which occurs at $\hat{t}^{l_q}$. With above auxiliary conclusions, we analyze the difference between $\sup\{\eta_i^{l_j}(0, t_f)\}$ and $\sup\{\eta_i^{l_q}(0, t_f)\}$ based on the proof strategy given in [8]. As $\hat{t}^{l_q} \leq \hat{t}^{l_j}$, we consider the following three cases:

[8] Due to space limitations, we omit detailed derivations here, which can be found in [8].

[9] $M_q^{l_q}$ is equivalent to $\hat{M}_j^{l_j}$.

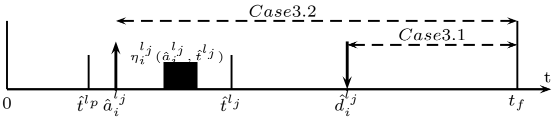

Figure 6: The execution scenarios

- *Case 1* ($d_i^{t_f} < \hat{t}^{l_q} \le \hat{t}^{l_j}$): In this case, neither a *l_q-carry-over job* nor a *l_j-carry-over job* exists. From Eqn. (25) and $\hat{t}^{l_q} \le \hat{t}^{l_j}$, we have:

$$\sup\{\eta_i^{l_j}(0,t_f)\} \le \sup\{\eta_i^{l_q}(0,t_f)\} + (t_f - \hat{t}^{l_j})(z_i(M_j^{l_j}) - z_i(M_q^{l_q}))u_i(0)$$

- *Case 2* ($\hat{t}^{l_q} \le d_i^{t_f} < \hat{t}^{l_j}$): In this case, a *$l_q$-carry-over job* exists but a *l_j-carry-over job* does not. We know $\sup\{\eta_i^{l_q}(0,d_i^{t_f})\}$ bounds $\eta_i^{l_q}(0,t_f)$. As $d_i^{t_f} < t_f$, we have:

$$\sup\{\eta_i^{l_q}(0,t_f)\} = \sup\{\eta_i^{l_q}(0,d_i^{t_f})\} + (t_f - d_i^{t_f}) \cdot z_i(M_q^{l_q}) \cdot u_i(0)$$

From Eqn. (25) and $d_i^{t_f} < \hat{t}^{l_j}$, we have:

$$\sup\{\eta_i^{l_j}(0,t_f)\} \le \sup\{\eta_i^{l_q}(0,t_f)\} + (t_f - \hat{t}^{l_j})(z_i(M_j^{l_j}) - z_i(M_q^{l_q}))u_i(0)$$

For above two cases, applying Prop. 2 and $z_i(M_j^{l_j}) \le z_i(M_q^{l_q})$, we can conclude that $\sup\{\eta_i^{l_j}(0,t_f)\}$ is bounded by:

$$\sup\{\eta_i^{l_q}(0,t_f)\} + \underbrace{(t_f - a_j^{l_j}) \cdot (1 - x_j^{l_j}) \cdot (z_i(M_j^{l_j}) - z_i(M_q^{l_q})) \cdot u_i(0)}_{difference\ term\ \psi_i^{l_j}}$$

- *Case 3* ($\hat{t}^{l_q} \le \hat{t}^{l_j} \le d_i^{l}$): In this case, both a *l_q-carry-over job* and a *l_j-carry-over job* exist. Two sub-cases needs to be considered, as shown in Fig. 6.

– *Case 3.1* ($\eta_i^{l_j}(\hat{a}_i^{l_j},\hat{t}^{l_j}) \neq 0$): The basic assumption $z_i(M_j^{l_j}) \le z_i(\hat{M}_q^{l_q})$ implies:

$$\sup\{\eta_i^{l_j}(\hat{a}_i^{l_j},\hat{d}_i^{l_j})\} \le \sup\{\eta_i^{l_q}(\hat{a}_i^{l_j},\hat{d}_i^{l_j})\} \tag{27}$$

From Eqn. (25), we have the following

$$\begin{aligned}
&\sup\{\eta_i^{l_j}(0,t_f)\}\\
&= \sup\{\eta_i^{l_j}(0,\hat{a}_i^{l_j})\} + \sup\{\eta_i^{l_j}(\hat{a}_i^{l_j},\hat{d}_i^{l_j})\} + (t_f - \hat{d}_i^{l_j})z_i(M_j^{l_j})u_i(0)\\
&\le \underline{\sup\{\eta_i^{l_q}(0,\hat{a}_i^{l_j})\} + \sup\{\eta_i^{l_q}(\hat{a}_i^{l_j},\hat{d}_i^{l_j})\} + (t_f - \hat{d}_i^{l_j})z_i(M_q^{l_q})u_i(0)}\\
&\quad + (t_f - \hat{d}_i^{l_j})(z_i(M_j^{l_j}) - z_i(M_q^{l_q}))u_i(0)\\
&= \sup\{\eta_i^{l_q}(0,t_f)\} + (t_f - \hat{d}_i^{l_j})(z_i(M_j^{l_j}) - z_i(M_q^{l_q}))u_i(0)
\end{aligned}$$

By replacing $\hat{d}_i^{l_j}$ via Prop. 3, $\sup\{\eta_i^{l_j}(0,t_f)\}$ is bounded as:

$$\sup\{\eta_i^{l_q}(0,t_f)\} + \underbrace{(t_f - a_j^{l_j}) \cdot (1 - x_j^{l_j}) \cdot (z_i(M_j^{l_j}) - z_i(M_q^{l_q})) \cdot u_i(0)}_{difference\ term\ \psi_i^{l_j}}$$

– *Case 3.2* ($\eta_i^{l_j}(\hat{a}_i^{l_j},\hat{t}^{l_j}) = 0$): From Eqn. (25) and Eqn. (26), we have:

$$\begin{aligned}
&\sup\{\eta_i^{l_j}(0,t_f)\}\\
&= \sup\{\eta_i^{l_j}(0,\hat{a}_i^{l_j})\} + \sup\{\eta_i^{l_j}(\hat{a}_i^{l_j},\hat{d}_i^{l_j})\} + (t_f - \hat{d}_i^{l_j})z_i(M_j^{l_j})u_i(0)\\
&= \sup\{\eta_i^{l_j}(0,\hat{a}_i^{l_j})\} + (t_f - \hat{a}_i^{l_j})z_i(M_j^{l_j})u_i(0)\\
&= \underline{\sup\{\eta_i^{l_q}(0,t_f)\}} + (t_f - \hat{a}_i^{l_j})(z_i(M_j^{l_j}) - z_i(M_q^{l_q}))u_i(0)
\end{aligned}$$

According to Prop. 2 and $\hat{a}_i^{l_j} < \hat{t}^{l_j}$, $\sup\{\eta_i^{k_0}(0,t_f)\}$ can be bounded by:

$$\sup\{\eta_i^{l_q}(0,t_f)\} + \underbrace{(t_f - a_j^{l_j}) \cdot (1 - x_j^{l_j}) \cdot (z_i(M_j^{l_j}) - z_i(M_q^{l_q})) \cdot u_i(0)}_{difference\ term\ \psi_i^{l_j}}$$

For the three cases shown above, we can conclude that $\sup\{\eta_i^{l_j}(0,t_f)\}$ can be upper bounded by $\sup\{\eta_i^{l_q}(0,t_f)\} + \psi_i^{l_j}$. By visiting all mode switches $M_j^{r_j}$, the upper bound of $\eta_i^{l_j}(0,t_f)$ can be obtained as Eqn. (16).

Appendix-B: The derivation for $N_\gamma^{l_j}$ in Eqn. (17).

According to Lem. 1, Lem. 2, and Lem. 3, the derivation of $N_\gamma^{l_j}$ in Eqn. (17) is detailed as follows:

$$\begin{aligned}
N_\gamma^{l_j} &= \sum_{\tau_i \in \gamma_L} \eta_i^{l_j}(0,t_f) + \sum_{\tau_i \in \gamma_H^L} \eta_j^{0}(0,t_f) + \sum_{\tau_j \in \gamma_H^H} \eta_j^{l_j}(0,t_f) \qquad (28)\\
&\le \sum_{\tau_i \in \gamma_L} \Big(t_f \cdot u_i(0) + \sum_{\tau_j \in \gamma_H^H} \sum_{r_j=1}^{l_j} \psi_i^{r_j}\Big) + \sum_{\tau_i \in \gamma_H^L} \frac{t_f}{x_j^0} u_j(0)\\
&\quad + \sum_{\tau_j \in \gamma_H^H} \Big(\frac{a_j^1}{x_j^0} \cdot u_j(0) + (t_f - a_j^1)u_j(1) + \sum_{r_j=2}^{l_j} (t_f - a_j^{r_j})\Delta u_j(r_j)\Big)\\
&= \sum_{\tau_i \in \gamma_L} \Big(t_f \cdot u_i(0) + \sum_{\tau_j \in \gamma_H^H} (t_f - a_j^1)(1 - x_j^0)\Delta u_i(M_j^1)\Big)\\
&\quad + \sum_{\tau_j \in \gamma_H^H} \sum_{r_j=2}^{l_j} (t_f - a_j^{r_j})(1 - x_j^{r_j - 1})\Delta u_L(M_j^{r_j})\\
&\quad + \sum_{\tau_i \in \gamma_H^L} \frac{t_f}{x_j^0} u_j(0) + \sum_{\tau_j \in \gamma_H^H} \Big(\frac{a_j^1}{x_j^0} \cdot u_j(0) + (t_f - a_j^1)u_j(1)\Big)\\
&\quad + \sum_{\tau_j \in \gamma_H^H} \sum_{r_j=2}^{l_j} (t_f - a_j^{r_j})\Delta u_j(r_j)\\
&= \underline{\sum_{\tau_i \in \gamma_L} \Big(t_f \cdot u_i(0) + \sum_{\tau_j \in \gamma_H^H} (t_f - a_j^1)(1 - x_j^0)\Delta u_i(M_j^1)\Big)}\\
&\quad \underline{+ \sum_{\tau_i \in \gamma_H^L} \frac{t_f}{x_j^0} u_j(0) + \sum_{\tau_j \in \gamma_H^H} \Big(\frac{a_j^1}{x_j^0} \cdot u_j(0) + (t_f - a_j^1)u_j(1)\Big)}\\
&\quad + \sum_{\tau_j \in \gamma_H^H} \sum_{r_j=2}^{l_j} (t_f - a_j^{r_j})\Big((1 - x_j^{r_j - 1})\Delta u_L(M_j^{r_j}) + \Delta u_j(r_j)\Big)
\end{aligned}$$

For ease of presentation, let us denote $\hat{N}_\gamma$ as the underline item in (28).

$$\begin{aligned}
&\hat{N}_\gamma = t_f \cdot u_L(0) + \sum_{\tau_j \in \gamma_H^H} (t_f - a_j^1)(1 - x_j^0)\Delta u_L(M_j^1)\\
&+ \sum_{\tau_i \in \gamma_H^L} \frac{t_f}{x_j^0} u_j(0) + \sum_{\tau_j \in \gamma_H^H} \Big(\frac{a_j^1}{x_j^0} \cdot u_j(0) + (t_f - a_j^1)u_j(1)\Big)\\
&Since\ u_L(0) + \sum_{\tau_j \in \gamma_H} \frac{u_j(0)}{x_j^0} \le 1\ with\ Eqn.\ (7)\\
&\le t_f \cdot (1 - \underline{\sum_{\tau_j \in \gamma_H} \frac{u_j(0)}{x_j^0}}) + \sum_{\tau_j \in \gamma_H^H} (t_f - a_j^1)\Big((1 - x_j^0)\Delta u_L(M_j^1) + u_j(1)\Big)\\
&\underline{+ \sum_{\tau_i \in \gamma_H^L} \frac{t_f}{x_j^0} u_j(0) + \sum_{\tau_j \in \gamma_H^H} \frac{a_j^1}{x_j^0} \cdot u_j(0)}\\
&= t_f - \underline{\sum_{\tau_i \in \gamma_H^H} (t_f - a_j^1)\frac{u_j(0)}{x_j^0}} + \sum_{\tau_j \in \gamma_H^H} (t_f - a_j^1)\Big((1 - x_j^0)\Delta u_L(M_j^1) + u_j(1)\Big)\\
&= t_f + \sum_{\tau_j \in \gamma_H^H} (t_f - a_j^1)\Big((1 - x_j^0)\Delta u_L(M_j^1) + u_j(1) - \frac{u_j(0)}{x_j^0}\Big)\\
&Since\ u_j(1) = \Delta u_j(1) + u_j(0)\\
&= t_f + \sum_{\tau_j \in \gamma_H^H} (t_f - a_j^1)\Big((1 - x_j^0)\Delta u_L(M_j^1) + \Delta u_j(1) + u_j(0) - \frac{u_j(0)}{x_j^0}\Big)
\end{aligned}$$

By substituting $\hat{N}_\gamma$ into (28), we have:

$$\begin{aligned}
N_\gamma^{l_j} \le t_f &+ \sum_{\tau_j \in \gamma_H^H} (t_f - a_j^1)\Big((1 - x_j^0)\Delta u_L(M_j^1) + \Delta u_j(1) + u_j(0) - \frac{u_j(0)}{x_j^0}\Big)\\
&+ \sum_{\tau_j \in \gamma_H^H} \sum_{r_j=2}^{l_j} (t_f - a_j^{r_j})\Big((1 - x_j^{r_j - 1})\Delta u_L(M_j^{r_j}) + \Delta u_j(r_j)\Big)
\end{aligned}$$