Received 4 August 2024, accepted 28 August 2024, date of publication 3 September 2024, date of current version 11 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3454164

RESEARCH ARTICLE

Component-Based Mixed-Criticality Real-Time Scheduling on a Single Processor System

DANDI MA, (Member, IEEE)
College of Computer and Information Science, Southwest University, Chongqing 400715, China
e-mail: mdd115@email.swu.edu.cn

ABSTRACT This paper focuses on a two-level hierarchical system with dual-criticality components scheduled on a single processor. To address the independent scheduling of mixed-criticality (MC) components using fixed-priority (FP) servers, we introduce the mixed-criticality deferrable server (MC-DS), featuring multiple server capacities within a fixed server period. Within the MC-DS, we employ the earliest deadline first (EDF) scheduler to manage mixed-criticality tasks for each component. We perform schedulability analysis based on the worst-case response time (WCRT) for MC-DSs. Augmented by the WCRT analysis, we derive a new supply bound function (SBF) to precisely evaluate the processor execution time guaranteed by the server for components over specific time intervals. Furthermore, a more effective schedulability test based on the concept of demand bound function – supply bound function (DBF-SBF) is presented. Through extensive experiments, we demonstrate the effectiveness of our proposed component-based schedulability analysis. Specifically, our method improves the acceptance ratio of task sets by an average of 17% in our experiments.

INDEX TERMS Earliest deadline first (EDF), fixed-priority (FP), hierarchical system, real-time server, mixed-criticality scheduling, supply bound function (SBF), schedulability analysis.

I. INTRODUCTION

Embedded real-time systems are trending toward consolidating functionalities into fewer but more powerful microprocessors. This paradigm shift is evident not only in the automotive electronics sector but also in avionics and other fields [1]. As the number of electronic devices in avionics systems increases, system architectures become more complex, complicating integration, testing, and maintenance. Running several avionics applications or modules on a single computing device can simplify system complexity and reduce weight. However, integrating multiple real-time applications on a single microprocessor introduces challenges in resource allocation and partitioning. To address these challenges, there is growing interest in real-time servers [2], which can distribute resources from the hardware platform to associated applications and schedule each application independently. In the automotive and aviation domains, the use of servers based on fixed-priority scheduling is particularly important [3].

In server-based hierarchical scheduling systems, each component is managed by a dedicated server. During scheduling, a global scheduler allocates computational resources from the physical computing platform to these servers. Subsequently, each server's local scheduler distributes the allocated resources to the component tasks according to its local scheduling algorithm. Existing servers typically have fixed replenishment periods and a single server capacity [4], [5], [6], [7], which are suitable for scheduling non-mixed-criticality components. However, real-time systems increasingly integrate components with different criticality levels on the same hardware platform [8], known as mixed-criticality (MC) systems.

In MC systems, high-criticality tasks must undergo certification by certification authorities (CAs), leading to extremely pessimistic worst-case execution time (WCET) estimates. In contrast, low-criticality tasks only need to meet the system designer's requirements, resulting in more optimistic

The associate editor coordinating the review of this manuscript and approving it for publication was Alba Amato.

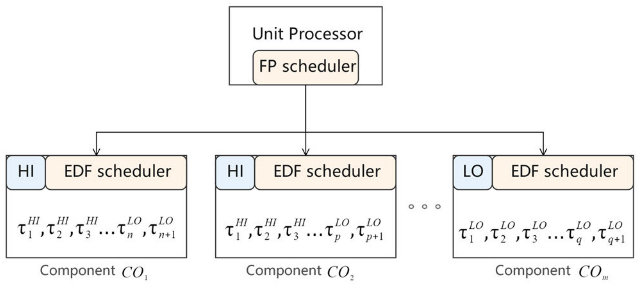

Unit Processor
FP scheduler
HI
EDF scheduler
$\tau_1^{HI},\tau_2^{HI},\tau_3^{HI}\ldots\tau_n^{LO},\tau_{n+1}^{LO}$
Component CO_1
HI
EDF scheduler
$\tau_1^{HI},\tau_2^{HI},\tau_3^{HI}\ldots\tau_p^{LO},\tau_{p+1}^{LO}$
Component CO_2
LO
EDF scheduler
$\tau_1^{LO},\tau_2^{LO},\tau_3^{LO}\ldots\tau_q^{LO},\tau_{q+1}^{LO}$
Component CO_m

FIGURE 1. A motivating scenario.

WCET estimates. If the system is validated at the highest level of assurance, all tasks will be scheduled under the most pessimistic WCET estimates, causing resource wastage. Therefore, Vestal [9] proposed multiple system modes to correspond to different assurance levels. For example, when the system executes in low-criticality mode, all tasks are scheduled according to optimistic WCET estimates. If any high-criticality task exceeds its optimistic WCET estimate, high-criticality tasks are then scheduled according to the most pessimistic WCET estimates.

The scheduling problem in component-based mixed-criticality systems focuses on resource distribution strategies for components when the mode switch occurs. Yang and Dong [10] proposed a degraded mixed-criticality resource model for the low-criticality components. They assume that the budgets for low-criticality components degrade to a specific value when a mode switch occurs. However, their study did not address mixed-criticality workloads. Building upon this foundation, Arafat et al. [11] considered mixed-criticality task models for low-criticality components and assumed independence between the criticality levels of resource models and workloads. Their research also focused on low-criticality components. Gu et al. [8] introduced an interface-based mixed-criticality periodic resource (MCPR) model and developed a schedulability test for high-criticality components under the EDF algorithm. However, their supply bound function (SBF) analysis in the schedulability test was pessimistic.

In this paper, we focus on a scenario where the system comprises a set of dual-criticality components and follows a two-level hierarchical scheduling framework, as depicted in FIGURE 1. At the global level, we employ a fixed-priority (FP) scheduler. Within the servers (also known as the local level), we use the earliest deadline first (EDF) schedulers to manage tasks within components. Each low-criticality component includes a set of low-criticality tasks, while each high-criticality component contains a set of dual-criticality tasks [8]. To reduce the pessimism in the existing schedulability test, we propose the mixed-criticality deferrable server (MC-DS) to guarantee the processor time for MC components. With MC-DS, the minimum processor time received for components within a specific time interval can be calculated using the server's WCRT, thereby enhancing accuracy.

A. CONTRIBUTION OF THIS PAPER

In this paper, we integrate the proposed mixed-criticality deferrable server into the hierarchical system depicted in FIGURE 1. Hierarchical scheduling follows two distinct research paths [12]: one involves schedulability analysis of the system given known server parameters, and the other focuses on designing server parameters. This paper adheres to the first research path. Specifically, we first ensure the schedulability of the mixed-criticality deferrable server in the system. Building upon this, we develop a schedulability analysis of component tasks based on the demand bound function (DBF) and supply bound function (SBF). The technical contributions of this paper can be summarized as follows:

- We propose an FP-based mixed-criticality deferrable server (MC-DS) for allocating computational resources from the physical computing platform to MC components. The MC-DS has two different server capacities, each linked to a distinct criticality level. To ensure that MC-DSs guarantee sufficient processor time for MC components, we conduct the worst-case response time (WCRT) analysis for the MC-DS in different modes and derive the schedulability analysis for mixed-criticality deferrable servers accordingly.
- Based on the WCRT of the MC-DS, we derive a precise computation of the minimum execution time guaranteed by the incomplete server period. This yields a new SBF that allows for a less pessimistic schedulability analysis of mixed-criticality components.
- Through rigorous experiments, we validate the effectiveness of the proposed server and schedulability test on scheduling component tasks. Our experimental results show that our method enhances the acceptance ratio of task sets by an average of 17%.

The remainder of this paper is organized as follows: Section II introduces related work. Section III provides a detailed description of the scheduling model. In Section IV, we introduce the existing SBF computation. Section V conducts a schedulability analysis of the proposed mixed-criticality deferrable server. In Section VI, we derive the server-based SBF computation and use it for the schedulability test of component tasks. The experimental results are presented in Section VII. In section VIII, we conclude the paper.

II. RELATED WORK

Over the past few decades, mixed-criticality systems and hierarchical systems have been extensively studied. In 2007, Vestal [9] first introduced the concept of mixed-criticality systems. He focuses on scheduling mixed-criticality tasks on a uniprocessor using the preemptive fixed-priority algorithm. Following Vestal's model, Baruah et al. [13] proposed the earliest deadline first with the virtual deadlines (EDF-VD) algorithm for mixed-criticality task models with implicit deadlines. To extend the idea of setting virtual deadlines to mixed-criticality task models with constrained deadlines, some studies have introduced deadline tightening algorithms based on the demand bound function [14], [15]. To address the conflict between Vestal's model and practical applications, where low-criticality tasks are completely discarded when the system switches to high-criticality mode, some studies have proposed reducing the worst-case execution time (WCET) of low-criticality tasks in high-criticality mode [16], [17], [18]. The elastic task model [19], [20], [21], [22] or multi-rate [23], [24] retains the execution of low-criticality tasks by extending their periods and relative deadlines. Dynamic models [25], [26] that improve system utilization are proposed in MC systems. Additionally, graceful degradation for low-criticality tasks [27], [28], [29] and mixed-criticality systems executing on varying-speed processors [30], [31], [32], [33] has also been studied. Burns and Davis [34] provided a comprehensive summary of various studies on mixed-criticality systems in 2022.

In research on hierarchical scheduling for non-mixed-criticality systems, Sha et al. [35] proposed the periodic server, which is invoked at fixed periods and has a fixed server capacity for each period. The deferrable server was first introduced in [36], enhancing the periodic server by allowing the server capacity to be preserved when the server is invoked but no tasks associated with it are ready to execute. Lehoczky et al. [36] conducted a schedulability analysis of deferrable servers based on the preemptive fixed-priority algorithm. Subsequently, Strosnider et al. [6] derived the worst-case response time of tasks running on a deferrable server under the preemptive FP algorithm. For both the periodic server and the deferrable server, Hamann et al. [37] proposed an exact response time analysis method for tasks under preemptive FP algorithms. Martinez et al. [3] presented an exact response time analysis method for tasks on a sporadic server, considering hierarchical systems with multiple levels. Insik and Insup [4] proposed a periodic resource model based on component interfaces and conducted schedulability analysis for tasks under both the FP and EDF algorithms.

Research on mixed-criticality systems with hierarchical frameworks has primarily focused on uniprocessor platforms. Gu et al. [8] proposed an interface-based mixed-criticality periodic resource (MCPR) model for mixed-criticality components. They derived a DBF-based schedulability test for component tasks using the EDF-VD algorithm. Yang and Dong [10] introduced a resource degradation model for low-criticality components executed on a virtual processor. This model reduces the resource budget for each resource period when the execution time requirements of low-criticality components degrade in high-criticality mode. In their work, the task model within the low-criticality components was non-mixed-criticality. Building on Yang and Dong's work [10], Arafat et al. [11] considered a mixed-criticality task model within low-criticality components and assumed that the criticality of the resource model and the workload were independent. They proposed four execution modes for the system based on this assumption.

III. SCHEDULING MODEL

In this paper, we focus on the preemptive scheduling of a set of MC tasks on the mixed-criticality deferrable server. Additionally, we assume discrete time throughout this paper. Therefore, all parameters representing time units are integers.

A. TASK AND COMPONENT

Each task τ_i in the system is represented by a quintuple, $\tau_i = \left(C_i^{LO}, C_i^{HI}, D_i, T_i, \chi_i\right)$. Task τ_i releases two consecutive jobs with the minimal separation interval T_i. Each job of τ_i must complete execution within D_i time units after its release, where $D_i \leq T_i$. The worst-case execution time (WCET) of τ_i has two estimates: C_i^{LO} is the estimate under less pessimistic system behavior and C_i^{HI} is the estimate under the most pessimistic system behavior. In this paper, we assume that the tasks are of dual criticality. For high-criticality tasks (for which $\chi_i = HI$), $C_i^{LO} < C_i^{HI}$. Conversely, for low-criticality tasks (which have $\chi_i = LO$), the WCETs meet $C_i^{LO} > C_i^{HI}$ and $= 0$. For simplicity, we refer to low-criticality tasks as LO-Tasks and high-criticality tasks as HI-Tasks.

Our system consists of m independent components with dual criticality. Each low-criticality component comprises a set of low-criticality tasks as described above, while each high-criticality component consists of a set of mixed-criticality tasks. Notably, there is at least one high-criticality task in each high-criticality component. To independently schedule tasks within components, we allocate a mixed-criticality deferrable server to each component to ensure isolation between components. The real-time scheduling requirements for dual-criticality components can be summarized as follows:

- Initially, the system starts in a low-criticality mode (referred to as LO-Mode), where the execution time of jobs in all components does not exceed C_i^{LO} time units. If all jobs within a component meet their deadlines, the component is considered to meet its real-time requirements in LO-Mode.
- If a job of a high-criticality task executes for C_i^{LO} time units without reporting completion, the system switches to high-criticality mode (referred to as HI-Mode), where the execution time of all tasks does not exceed C_i^{HI} time units. In HI-Mode, if every job of each HI-Task within a high-criticality component meets its deadline, the component's real-time requirements are satisfied.

B. MIXED-CRITICALITY DEFERRABLE SERVER

In our system, each mixed-criticality deferrable server DS is assigned a unique priority and is scheduled by a global FP scheduler. The DS is represented by a quadruple $DS = \left(\Theta_{DS}^{LO}, \Theta_{DS}^{HI}, \Pi_{DS}, \Upsilon_{DS}\right)$, where Π_{DS} denotes the replenishment period of the server, and $\Upsilon_{DS} \in \{HI, LO\}$

indicates the criticality of DS. Θ_{DS}^{LO} is the server capacity in low-criticality mode (LO-Mode), and Θ_{DS}^{HI} is the server capacity in high-criticality mode (HI-Mode). The server capacity represents the maximum execution time that the server can consume within one server period [5]. The MC-DS is based on the deferrable server and uses the same capacity preservation mechanism as ordinary deferrable servers [6]. Specifically, when there are no ready tasks to execute on the server, the MC-DS preserves its capacity until the end of the server period.

Initially, the system starts in LO-Mode. In this mode, the MC-DS replenishes its capacity by Θ_{DS}^{LO} time units for each server period. When a high-criticality task triggers a mode switch at time instant s, the low-criticality servers cease operation. High-criticality servers then receive an additional capacity replenishment for the server period during which the mode switch occurred (defined as a mode-switch server period). From the next server period onward, the server replenishes its capacity by Θ_{DS}^{HI} time units for each server period. Importantly, when a task triggers a mode switch, all mixed-criticality servers in the system exhibit high-criticality behavior simultaneously.

Definition 1 (Mode-Switch Server Period): If the arrival time a_e of a server period is less than or equal to the mode-switch time instant s and its end time a_h is greater than s, we refer to it as a mode-switch server period.

Definition 2 (Incomplete Server Period): If the arrival time a_{in} of a server period is less than or equal to t and its end time $a_{in} + \Pi_{DS}$ is greater than t, we refer to it as an incomplete server period during the time interval $[0, t]$.

Now, we detail the rules for supplementing additional capacity in a mode-switch server period. To better illustrate this mechanism, we define L_{se} as the gap between the mode-switch time instant s and the start time a_e of the mode-switch server period, i.e., $L_{se} = s - a_e$. The rules for the additional replenishment mechanism are as follows:

If $L_{se} < \Theta_{DS}^{LO}$, we supplement $\Theta_{DS}^{HI} - \Theta_{DS}^{LO}$ time units of additional server capacity at time instant s for the mode-switch server period. Otherwise, if $L_{se} \geq \Theta_{DS}^{LO}$, no additional capacity replenishment is performed for this period. Starting from the next server period, each server period will replenish Θ_{DS}^{HI} time units of server capacity. Notably, this replenishment mechanism is the same as that of MCPR [8].

C. SCHEDULING ALGORITHM

In two-level hierarchical systems, a global scheduling algorithm is employed to allocate execution resources to components, while a local scheduling algorithm selects ready tasks within the selected component for execution. At the global level, we utilize the preemptive FP algorithm to schedule servers. The preemptive FP algorithm offers flexibility and is sufficiently straightforward to allow for the construction of efficient embedded real-time operating systems [38]. At the local level, we employ the earliest deadline first with virtual deadlines (EDF-VD) algorithm [13], [14], [15], [16] to determine which task to execute on the server. This algorithm is widely adopted for scheduling mixed-criticality tasks.

1) EDF-VD SCHEDULING STRATEGY

In LO-Mode, each task τ_i is assigned a virtual relative deadline denoted as D_i^v. Specifically, for each HI-Task, $D_i^v \leq D_i$ whereas for each LO-Task, $D_i^v = D_i$. When the system switches to HI-Mode, the relative deadlines of all tasks are updated to their actual relative deadlines. Setting virtual deadlines ensures that all executing high-criticality jobs have enough remaining time to fulfill additional execution demands caused by the mode switch. We set virtual deadlines for tasks according to ECDF algorithm [15].

IV. EXISTING COMPUTATION METHODS FOR SBF

In this paper, we use the EDF scheduler to manage tasks within each component. Under the EDF algorithm, there are two main approaches for performing schedulability analysis: the utilization-based approach and the demand bound function-supply bound function (DBF-SBF) analysis. We focus on the DBF-SBF analysis.

The DBF describes the maximum processor execution time required by the component tasks over a specific time interval. The SBF defines the minimum processor execution time that can be guaranteed for a component over a specific time interval. Since the calculation of the DBF for mixed-criticality tasks has been extensively studied [14], [15], [39], we can directly apply them in the schedulability test of component tasks. Our research focuses on providing new SBF calculations.

A. THE SBF OF THE NON-MIXED-CRITICALITY RESOURCE MODEL

The periodic resource model $R(\Theta, \Pi)$ [4] was proposed to abstract the execution of non-mixed criticality components. The SBF of $R(\Theta, \Pi)$ during a specific time interval Δ is defined as:

$$SBF_R(\Delta) = \begin{cases} 0, & \text{if} \Delta \leq 2 \cdot (\Pi - \Theta) \\ \left\lfloor \dfrac{\Delta - (\Pi - \Theta)}{\Pi} \right\rfloor \cdot \Theta + \varepsilon, & \text{otherwise} \end{cases} \tag{1}$$

where

$$\varepsilon = \max \left\{ 0, \Delta - 2(\Pi - \Theta) - \Pi \left\lfloor \dfrac{\Delta - (\Pi - \Theta)}{\Pi} \right\rfloor \right\} \tag{2}$$

$SBF_R(\Delta)$ assumes that the resources are exhausted exactly at 0, and the workload must wait $\Pi - \Theta$ time units for resources to be supplied again. ε calculates the resource supply of R in its last period, assuming that the resource budget for this period is consumed $\Pi - \Theta$ time units after the period's release.

The schedulability test for non-mixed-criticality components under the EDF scheduling algorithm based on DBF-SBF is described in Theorem 1.

Theorem 1 (Theorem 1 in [4]): A non-mixed-criticality workload W scheduled by the EDF algorithm receives execution resources from Γ. W is schedulable if and only if, for every time interval within a hyperperiod, the resource demand of W does not exceed the resource supply of Γ,i.e.,

$$\forall 0 < t \leq 2 \cdot LCM_W : DBF_W(t) \leq SBF_\Gamma(t) \tag{3}$$

where LCM_W is the hyperperiod of the workload, $DBF_W(t)$ represents the maximum execution time required by workload W over the interval t, and $SBF_\Gamma(t)$ represents the minimum execution time provided by Γ over the same interval.

A non-mixed-criticality system executes in a single system mode, so it is sufficient to test schedulability for every $t \in (0, 2 \cdot LCM_W]$. However, in a mixed-criticality system, the mode switching can occur at any time. It is necessary to verify the schedulability for all potential mode switch points within the time interval t.

B. THE SBF OF MIXED-CRITICALITY RESOURCE MODEL

To schedule mixed-criticality components, Gu et al. [8] proposed the mixed-criticality periodic resource (MCPR) model based on the periodic resource model [4]. The MCPR can be abstracted as a periodic task $I = (T, L, C)$, where T represents the period, $L \in \{LO, HI\}$ denotes the criticality level of the corresponding component, and $C = \{C^L, C^H\}$ represents the resource budget for each period. In the low-criticality mode (or high-criticality mode), I provides at least C^L(or C^H) time units of the budget every T time units.

The execution time supplied by the mixed-criticality periodic resource I during the time interval $[0, t]$ is discussed in two scenarios: the first scenario is when I operates in a stable low-criticality mode, meaning that the mode switch time instant s equals t; the second scenario is when I undergoes a mode switch during the interval $[0, t]$, meaning that $0 \leq s < t$. The SBF of I in LO-Mode is defined by (1). They focused on the SBF of I when it experienced a mode switch.

Gu et al. [8] considered two possible supply patterns of I that guarantee the minimal execution time for high-criticality components when $0 \leq s < t$, namely *pattern A* and *pattern B*. The minimum resource supply in *pattern A*, $SBF_I^A(s,t)$, and the minimum resource supply in *pattern B*, $SBF_I^B(s,t)$, are computed by (4) and (5), respectively:

$$SBF_I^A(s,t) = \begin{cases} n_A^{LO} \cdot C^L + \left(n_A - n_A^{LO}\right) \cdot C^{HI} + sbf^{IN}(n_A, t) \\ , \quad if\ s - a < C^L \\ \left(n_A^{LO} + 1\right) \cdot C^L + \left(n_A - n_A^{LO} - 1\right) \cdot C^{HI} \\ +sbf^{IN}(n_A, t), if\ s - a \geq C^L and\ a \neq b \\ n_A^{LO} \cdot C^L + \min\left(sbf^{IN}(n_A, t), C^L\right) \\ , \quad if\ s - a \geq C^L and\ a = b \end{cases} \tag{4}$$

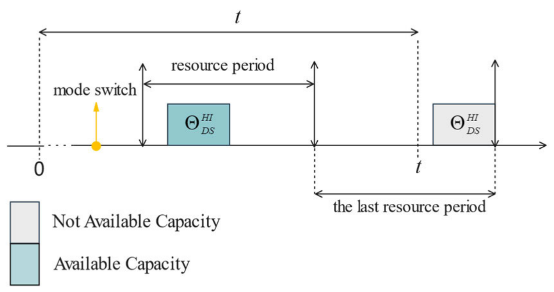


FIGURE 2. The worst-case resource supply for MCPR's last resource period.

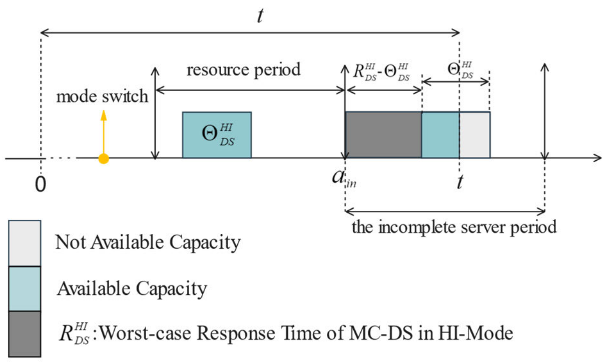

t
resource period
R_{DS}^{HI}-Θ_{DS}^{HI}
Θ_{DS}^{HI}
mode switch
Θ_{DS}^{HI}
0
a_{in}
t
the incomplete server period
Not Available Capacity
Available Capacity
R_{DS}^{HI}:Worst-case Response Time of MC-DS in HI-Mode

FIGURE 3. The worst-case resource supply for MC-DS's incomplete server period.

where n_A^{LO} represents the number of complete resource periods within the interval $[0, s]$, and n_A indicates the number of complete resource periods within the interval $[0, t)$. The term $sbf^{IN}(n, t)$ denotes the processor execution time provided by the last resource period within interval $[0, t]$, where $sbf^{IN}(n_A, t) = [\![t - (2T - C^L - C^H) - n_A \cdot T]\!]_0$.

$$SBF_I^B(s,t) = \begin{cases} \left(n_B^{LO} + 1\right) \cdot C^L + \left(n_B - n_B^{LO} - 1\right) \cdot C^{HI} \\ +sbf^{IN}(n_B, t, l), \qquad if\ a \neq b \\ n_B^{LO} \cdot C^L + \min\left(sbf^{IN}(n_B, t, l), C^L\right), \quad if\ a = b \end{cases} \tag{5}$$

where $sbf^{IN}(n_B, t, l)$ denotes the processor execution time provided by the last resource period in pattern B, where $sbf^{IN}(n_B, t, l) = [\![t - l - (T - C^H) - n_B \cdot T]\!]_0$. l denotes the length of shifting the time interval $[0, t]$ in pattern A to the right, becoming pattern B. This shift length l is given by $l = \lceil s/T \rceil \cdot T - s$.

Therefore, a safe lower bound of SBF when $0 \leq s < t$ is defined as follows:

$$SBF_I(s,t) = \min\left(SBF_I^A(s,t), SBF_I^B(s,t)\right) \tag{6}$$

C. PESSIMISM IN THE EXISTING METHOD

According to the SBF calculation method in [8], they assume that the resource budget in the incomplete resource period is not available until the end of the resource period. As shown in FIGURE 2, if the time instant t is close to the start time of the last resource period, the processor time guaranteed by this period within the time interval $[0, t]$ is 0. This is a pessimistic assumption.

To address this issue, we propose the MC-DS to guarantee processor time for MC components. With the worst-case response time analysis for MC-DSs, we determine the longest time for a server period to guarantee the server capacity in different system modes, which makes the lower bound of processor supply for the last server period less pessimistic. As shown in FIGURE 3, the last (incomplete) server period executes in HI-Mode and the WCRT of this server period is R_{DS}^{HI}. Therefore, processor time guaranteed by this period within the time interval $[0, t]$ is $t - a_{in} - \left(R_{DS}^{HI} - \Theta_{DS}^{HI}\right)$.

Here, we examine the case where the incomplete server period runs in HI-Mode. This approach applies to server periods in other modes as well.

V. WORST-CASE RESPONSE TIME ANALYSIS FOR MIXED-CRITICALITY DEFERRABLE SERVERS

To meet the varying processor requirements of mixed-criticality components and independently verify each component, we propose the mixed-criticality deferrable server (see details in Section III). We now develop the worst-case response time (WCRT) analysis for the MC-DS. In subsection A, we perform WCRT analysis for mixed-criticality deferrable servers in stable modes. In subsection B, we analyze the WCRTs for mixed-criticality servers that experience a mode switch. Combining the results from subsections A and B, we present the schedulability test based on the WCRT for MC deferrable servers in Theorem 2.

A. WORST-CASE RESPONSE TIME IN STABLE MODES

In stable modes, the interference caused by an MC-DS on lower-priority servers is the same as the interference caused by a non-mixed-criticality deferrable server. The maximum interference generated by a non-mixed-criticality deferrable server S within the interval t is calculated as follows [1]:

$$\Theta_S + \left\lceil \frac{t - \Theta_S}{\Pi_S} \right\rceil \Theta_S \tag{7}$$

In stable LO-Mode, each MC-DS DS (including high-criticality servers and low-criticality servers) guarantees Θ_{DS}^{LO} time units of capacity for each server period. Therefore, the maximum interference generated by each higher priority MC-DS X within the interval t is $\Theta_X^{LO} + \left\lceil \left(t - \Theta_X^{LO}\right) / \Pi_X \right\rceil \Theta_X^{LO}$. The WCRT for DS in stable LO-Mode is as follows:

$$R_{DS}^{LO,n+1} = \Theta_{DS}^{LO} + \sum_{\forall X \in hp(DS)} \left(\Theta_X^{LO} + \left\lceil \frac{R_{DS}^{LO,n} - \Theta_X^{LO}}{\Pi_X} \right\rceil \Theta_X^{LO} \right) \tag{8}$$

where $hp(DS)$ denotes the MC deferrable servers (including high-criticality servers and low-criticality servers) with higher priority than DS. R_{DS}^{LO} represents the WCRT of DS executing Θ_{DS}^{LO} time units of capacity in stable LO-Mode. R_{DS}^{LO} starts recursion with $R_{DS}^{LO,0} = R_{DS}^{LO}$, and ends when $R_{DS}^{LO,n} > \Pi_{DS}$ or $R_{DS}^{LO,n+1} = R_{DS}^{LO,n}$. If $R_{DS}^{LO,n} > \Pi_{DS}$, then DS is not schedulable. Otherwise, if $R_{DS}^{LO,n+1} = R_{DS}^{LO,n}$, $R_{DS}^{LO,n+1}$ is the WCRT of DS.

In stable HI-Mode, each high-criticality server DS guarantees Θ_{DS}^{HI} time units of capacity for each server period, while low-criticality servers stop executing. Therefore, DS suffers interference only from high-criticality servers with higher priority than itself. The WCRT for DS in stable HI-Mode is as follows:

$$R_{DS}^{HI,n+1} = \Theta_{DS}^{HI} + \sum_{\forall X \in hpH(DS)} \left(\Theta_X^{HI} + \left\lceil \frac{R_{DS}^{HI,n} - \Theta_X^{LO}}{\Pi_X} \right\rceil \Theta_X^{HI} \right) \tag{9}$$

where $hpH(DS)$ denotes the high-criticality servers with higher priority than DS. R_{DS}^{HI} represents the WCRT of DS executing Θ_{DS}^{HI} time units of capacity in stable HI-Mode. The calculation steps of R_{DS}^{HI} is similar to the calculation of R_{DS}^{LO}.

B. WORST-CASE RESPONSE TIME DURING A MODE SWITCHING

When a server period of high-criticality MC servers DS experiences a mode switch, it is affected by interference from the two factors below:

(1). I_{DSL} : the interference caused by the low-criticality server DSL, which has higher priority than DS. I_{DSL} is defined in Lemma 1.

(2). I_{DSH} : the interference caused by the high-criticality server DSH, which has higher priority than DS. I_{DSH} is defined in Lemma 2.

Lemma 1: When the mode switch occurs at time instant s, where $0 \le s < t$, the maximum execution time demand I_{DSL} of DSL is:

$$I_{DSL} = \left(1 + \left\lceil \frac{s - \Theta_{DSL}^{LO}}{\Pi_{DSL}} \right\rceil \right) \Theta_{DSL}^{LO} \tag{10}$$

where $s = \Theta_{DS}^{LO}$ when $s < \Theta_{DS}^{LO}$, and $s = t$ when $\Theta_{DS}^{LO} \le s < t$.

Proof: Since low-criticality servers stop executing when a mode switch occurs, the execution time demand of DSL during the time interval $[0, t]$ is restricted to interval $[0, s]$. During the interval $[0, s]$, there are $\left\lceil \left(s - \Theta_{DSL}^{LO}\right) / \Pi_{DSL} \right\rceil + 1$ periods that demand Θ_{DSL}^{LO} time units within each period. Since the mode-switch server period has two types of capacities, we consider I_{DSL} in two cases and maximize it under these two cases by setting the value of s to its maximum value. The first case is when $s < \Theta_{DS}^{LO}$, where the mode-switch period supplements extra capacity. Another case is when $\Theta_{DS}^{LO} \le s < t$.

Lemma 2: When the mode switch occurs at time instant s, where $0 \le s < t$, the maximum execution time demand I_{DSH}

of *DSH* is:

$$I_{DSH} = \left(1 + \left\lceil \frac{t - \Theta_{DSH}^{LO}}{\Pi_{DSH}} \right\rceil\right)\Theta_{DSH}^{HI} \tag{11}$$

Proof: For the high-criticality server DSH, the server period that ends before s demands at most Θ_{DSH}^{LO} time units of execution and the server period that ends after s demands at most Θ_{DSH}^{HI} time units of execution. To upper bound I_{DSH}, we assume that $s = 0$. In this case, there are $\lceil (t - \Theta_{DSH}^{LO}) / \Pi_{DSH} \rceil + 1$ periods that demand Θ_{DSH}^{HI} time units within each period.

Based on Lemma 1 and Lemma 2, we derive the WCRT of the high-criticality server DS as follows:

$$\begin{aligned} & R_{DS}^{MC,n+1} \\ &= \Theta + \sum_{\forall X \in hpL(DS)} \left(1 + \left\lceil \frac{s - \Theta_X^{LO}}{\Pi_X} \right\rceil\right)\Theta_X^{LO} \\ &\quad + \sum_{\forall Y \in hpH(DS)} \left(1 + \left\lceil \frac{R_{DS}^{MC,n} - \Theta_Y^{LO}}{\Pi_Y} \right\rceil\right)\Theta_Y^{HI} \end{aligned} \tag{12}$$

where $\Theta = \Theta_{DS}^{LO}$ and $s = R_{DS}^{MC,n}$, if the mode-switch period of DS does not provide additional capacity. However, if the mode-switch period supplements additional capacity, then $\Theta = \Theta_{DS}^{HI}$ and $s = \Theta_{DS}^{LO}$. The calculation steps of R_{DS}^{MC} is similar to those of R_{DS}^{LO}.

Then, we derive the WCRT-based schedulability analysis of a set of MC deferrable servers in Theorem 2.

Theorem 2: MC systems consisting of MC-DSs are schedulable at the global level under the FP algorithm if each MC-DS DS satisfies the following condition:

$$R_{DS}^{LO} \le \Pi_{DS} \wedge R_{DS}^{HI} \le \Pi_{DS} \wedge R_{DS}^{MC} \le \Pi_{DS} \tag{13}$$

where R_{DS}^{LO} represents the WCRT of DS in LO-Mode, which is defined by (8). R_{DS}^{HI} represents the WCRT of DS in HI-Mode, which is defined by (9). And R_{DS}^{MC} represents the WCRT of DS during a mode switch, which is defined as the maximum value in the two cases described in (12).

VI. SCHEDULABILITY ANALYSIS FOR COMPONENT TASKS

In this section, we develop a schedulability test for tasks within each component based on the DBF-SBF method. We assume that MC-DSs in the system are schedulable. Based on this assumption, we focus on task scheduling within individual components. Since our task scheduling strategy is the same as in [15], the formulas from that reference are directly used to calculate DBF. Our main focus is on deriving the SBF.

In section A, we introduce two possible arrival patterns that the MC-DS guarantees to provide a minimum SBF. In sections B and C, we perform specific calculations for the SBF in these two patterns. In subsection D, we derive the upper bound t_{MAX} of the time interval that needs to be verified in the schedulability analysis (see Theorem 3). To make the derivation process more concise, we use notations listed in Table 1.

TABLE 1. Notations.

Notation	Explanation
a_f^A	start time of the first server period during the time interval $[0,t]$, under arrival pattern A
a_f^B	start time of the first server period during the time interval $[0,t]$, under arrival pattern B
a_e^A	start time of the mode-switch server period under arrival pattern A
a_e^B	start time of the mode-switch server period under arrival pattern B
a_{in}^A	start time of the incomplete server period under arrival pattern A
a_{in}^B	start time of the incomplete server period under arrival pattern B
$sbf^{LO}(a,b)$	minimum execution time guaranteed by MC-DS during the interval $[a,b]$, where MC-DS executes in LO-Mode
$sbf^{HI}(a,b)$	minimum execution time guaranteed by MC-DS during the interval $[a,b]$, where MC-DS executes in HI-Mode
$sbf_{mcp}(a,b)$	minimum execution time guaranteed by the mode-switch server period.
$sbf_{inp}(s,t,a,b)$	minimum execution time guaranteed by the incomplete server period
x	The time shift applied to the interval $[0,t]$ from Pattern A to the right.

A. WORST-CASE ARRIVAL PATTERN

To establish a lower bound of the resource supply for mixed-criticality deferrable servers during a specific time interval. We initially determine the worst-case arrival patterns of component tasks on the MC servers. These arrival patterns ensure that the MC-DS guarantees the minimum execution time. Given that we follow the same capacity replenishment rules as MCPR [8], we consider applying the arrival patterns proposed in that work. Subsequently, we demonstrate that these arrival patterns are suitable for our analysis.

The first arrival pattern, referred to as *pattern A*, is shown in FIGURE 4. In this pattern, tasks arrive at time instant 0, precisely when the server capacity of MCS has just exhausted. The tasks must wait for $\Pi_{MCS} - \Theta_{MCS}^{LO}$ time units to receive available server capacity making $a_f^A = \Pi_{MCS} - \Theta_{MCS}^{LO}$. In pattern A, if we shift the time interval $[0, t]$(including time instant s) to the left, it increases the SBF. However, if we shift the interval $[0, t]$ to the right by x time units, the mode-switch server period no longer receives additional capacity replenishment (as shown in interval $\left[a_e^B, a_h^B\right]$ in FIGURE 5), which may lead to a decrease in the SBF. We refer to this situation as *pattern B*. In pattern B, tasks arriving at 0 must wait for

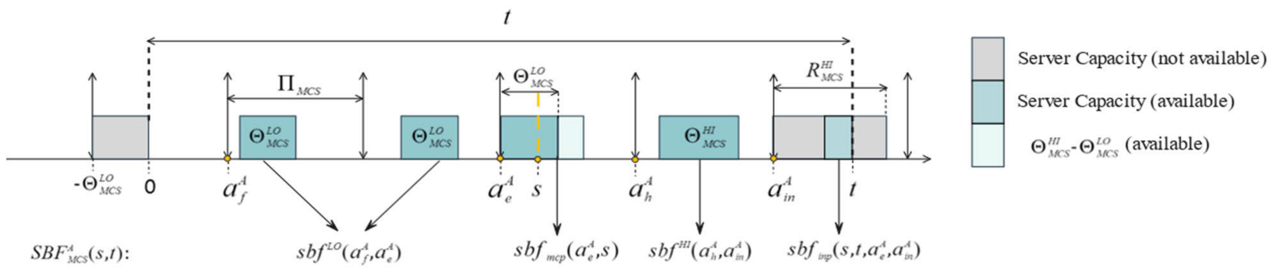


FIGURE 4. A high-criticality server *MSC* executes under Pattern A during the time interval $[0, t]$, where a mode switch occurs at time instant *s*.

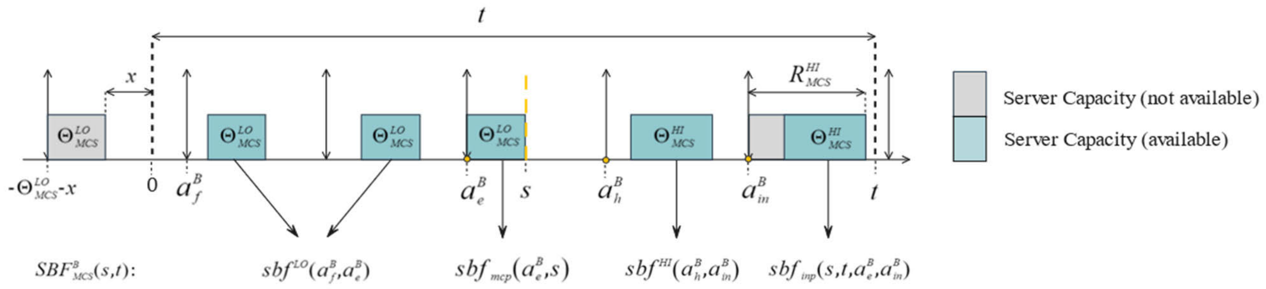


FIGURE 5. A high-criticality server *MSC* executes under Pattern B which is obtained by shifting the time interval $[0, t]$ in Pattern A to the right by *x* time units.

$\Pi_{MCS} - \Theta^{LO}_{MCS} - x$ time units to receive available capacity. Therefore, $a^B_f = \Pi_{MCS} - \Theta^{LO}_{MCS} - x$. And x is defined as $\lceil s/\Pi_{MCS} \rceil \cdot \Pi_{MCS} - s$ [8].

When $x > \Pi_{MCS} - \Theta^{LO}_{MCS}$, shifting the interval $[0, t]$ to the right by x time units under Pattern A is equivalent to shifting $[0, t]$ to the left by some time units. This results in an increase in the SBF. Therefore, we only consider the SBF under Pattern A when $x > \Pi_{MCS} - \Theta^{LO}_{MCS}$. We define the minimum execution time provided by *MCS* when the component executes under pattern A as $SBF^A_{MCS}(s, t)$. Similarly, the minimum execution time provided by *MCS* when the component executes under pattern B is denoted as $SBF^B_{MCS}(s, t)$. Thus, the SBF of *MCS* is defined as follows:

$$SBF_{MCS}(s,t) = \begin{cases} SBF^A_{MCS}(s,t) \text{ ,if } x > \Pi_{MCS} - \Theta^{LO}_{MCS} \\ \min\left(SBF^A_{MCS}(s,t), SBF^B_{MCS}(s,t)\right) \\ \text{,if } x \le \Pi_{MCS} - \Theta^{LO}_{MCS} \end{cases} \tag{14}$$

where $SBF^A_{MCS}(s, t)$ is derived in (22) and $SBF^B_{MCS}(s, t)$ is derived in (25)

Although the above analysis is for high-criticality servers, it can also be directly applied to low-criticality servers. Since low-criticality servers cease to operate after a mode switch, the SBF calculation under pattern A can be used directly. Based on the DBF and SBF calculations, we derive the schedulability test for mixed-criticality components, as stated in Theorem 3.

Theorem 3: A mixed-criticality component *CO* is scheduled by the EDF scheduler in the mixed-criticality deferrable server *MCS*. Then, MC tasks in *CO* are schedulable if they satisfy the following condition:

$$\forall 0 < t \le t_{MAX}, 0 \le s \le t : DBF_{CO}(s,t) \le SBF_{MCS}(s,t) \tag{15}$$

where $SBF_{MCS}(s, t)$ is performed in (14), and $DBF_{CO}(s, t)$ is derived in [15].

Next, we will discuss how to calculate $SBF^A_{MCS}(s, t)$ in section B and how to calculate $SBF^B_{MCS}(s, t)$ in section C.

B. SUPPLY BOUND FUNCTION UNDER PATTERN A

Initially, *MCS* operates in LO-Mode, where each server period (as shown in periods within time interval $[0, a^A_e]$ in FIGURE 4) contributes a processor budget of Θ^{LO}_{MCS} time units to the component. When a mode switch occurs, the execution budget contributed by the mode-switch server period (time interval $[a^A_e, a^A_h]$ as shown in FIGURE 4) needs further discussion. After the mode-switch server period ends, *MCS* operates in stable HI-Mode, where each server period (as shown in periods within time interval $[a^A_h, a^A_{in}]$ shown in FIGURE 4) contributes a processor budget of Θ^{HI}_{MCS}. Moreover, there is an incomplete server period (as shown in time interval $[a^A_{in}, t]$ in FIGURE 4) within $[0, t]$ whose contributed processor execution time also requires further discussion. *Therefore, the focus and challenge in calculating the supply bound function lie in determining the minimum processor* *execution time contributed by the mode-switch server period and the incomplete server period.*

In pattern A, MCS replenishes its first server period at time instant a_f^A during time interval $[0, t]$, where a_f^A is detailed in section A. Then, we derive the start time of the mode-switch server period a_e^A and the start time of the incomplete server period a_{in}^A. These values are calculated as follows:

$$a_e^A = \left\lfloor \frac{s + \Theta_{MCS}^{LO}}{\Pi_{MCS}} \right\rfloor \cdot \Pi_{MCS} - \Theta_{MCS}^{LO} \tag{16}$$

$$a_{in}^A = \left\lfloor \frac{t + \Theta_{MCS}^{LO}}{\Pi_{MCS}} \right\rfloor \cdot \Pi_{MCS} - \Theta_{MCS}^{LO} \tag{17}$$

Since the mixed-criticality system considers the mode switch at every time instant within time interval $[0, t]$, implying that $0 \leq s \leq t$ [15], the value of a_e^A must be less than or equal to a_{in}^A. According to the value of a_e^A and a_f^A, we divide the server periods within time interval $[0, t]$ into parts with different system modes.

We first consider the processor time guaranteed by server periods executing in stable LO-Mode and stable HI-Mode.

During the time interval $\left[0, a_e^A\right]$, server periods operate in LO-Mode with at least $\max\left(0, \left\lfloor \left(a_e^A - a_f^A\right) / \Pi_{MCS} \right\rfloor\right)$ server periods. Therefore, the execution time guaranteed by server periods in LO-Mode is calculated by $sbf^{LO}\left(a_f^A, a_e^A\right)$ as follows:

$$sbf^{LO}\left(a_f^A, a_e^A\right) = \max\left(0, \left\lfloor \frac{a_e^A - a_f^A}{\Pi_{MCS}} \right\rfloor\right) \cdot \Theta_{MCS}^{LO} \tag{18}$$

where $a_f^A = \Pi_{MCS} - \Theta_{MCS}^{LO}$, and a_e^A is calculated by (16).

After the mode-switch server period ends at time instant a_h^A, MCS operates in stable HI-Mode and replenishes capacity of Θ_{MCS}^{HI} time units for each server period. As shown in FIGURE 4, there are at least $\max\left(0, \left\lfloor \left(a_{in}^A - a_h^A\right) / \Pi_{MCS} \right\rfloor\right)$ complete server periods executing in stable HI-Mode. The minimum execution time guaranteed by these periods is calculated by $sbf^{HI}\left(a_h^A, a_{in}^A\right)$ as follows:

$$sbf^{HI}\left(a_h^A, a_{in}^A\right) = \max\left(0, \left\lfloor \frac{a_{in}^A - a_h^A}{\Pi_{MCS}} \right\rfloor\right) \cdot \Theta_{MCS}^{HI} \tag{19}$$

where $a_h^A = a_e^A + \Pi_{MCS}$ and a_{in}^A is calculated by (17).

Next, we consider the processor execution time contributed by the mode-switch server period (as shown in time interval $\left[a_e^A, a_h^A\right]$ in FIGURE 4).

When the mode switch occurs before a_f^A, the mode-switch server period guarantees 0 processor since the server capacity has been exhausted in the worst-case assumption. When the mode switch occurs after a_f^A, there are two types of the mode-switch server period. The first type is shown in FIGURE 4, where the end time of the mode-switch server period is no greater than t. In this case, the minimum execution time guaranteed by the mode-switch server period is calculated in Lemma 3. The second type is shown in

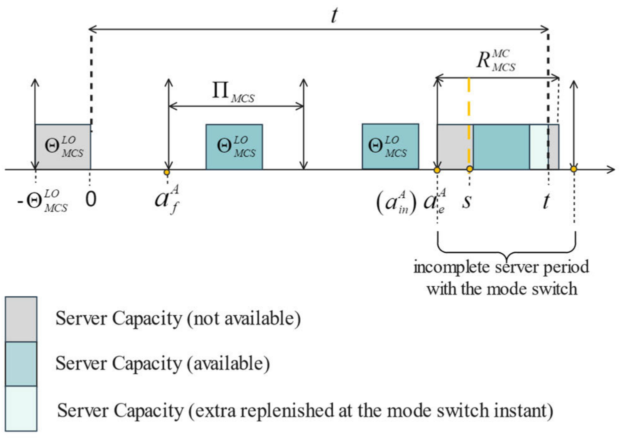

t
Π_{MCS}
R_{MCS}^{MC}
Θ_{MCS}^{LO}
Θ_{MCS}^{LO}
Θ_{MCS}^{LO}
$-\Theta_{MCS}^{LO}$
0
a_f^A
(a_{in}^A) a_e^A
s
t
incomplete server period with the mode switch
Server Capacity (not available)
Server Capacity (available)
Server Capacity (extra replenished at the mode switch instant)

FIGURE 6. **The mode-switch server period is an incomplete server period.**

FIGURE 6, where the end time of the mode-switch server period is greater than t. Then, the mode-switch server period is also an incomplete server period, and the processor execution time it contributes is obtained from Lemma 4.

Lemma 3: If the end time of the mode-switch server period is greater than a_f^A and less than t, the execution time guaranteed by this period is calculated as follows:

$$sbf_{mcp}\left(a_e^A, s\right) = \begin{cases} \Theta_{MCS}^{LO}, & \text{if } s - a_e^A \geq \Theta_{MCS}^{LO} \\ \Theta_{MCS}^{HI}, & \text{if } s - a_e^A < \Theta_{MCS}^{LO} \end{cases} \tag{20}$$

where a_e^A denotes the start time of the mode-switch server period, and is calculated by (16).

Proof: According to the rules of the MC-DS (as detailed in section III), the mode-switch server period does not replenish extra capacity when $s - a_e^A \geq \Theta_{MCS}^{LO}$. Therefore, it contributes Θ_{MCS}^{LO} time units of execution time. However, if $s - a_e^A < \Theta_{MCS}^{LO}$, the mode-switch server period replenishes server capacity to Θ_{MCS}^{HI}.

Finally, we discuss the minimum processor time contributed by the incomplete server period in Lemma 4.

Lemma 4: The incomplete server period during the time interval $[0, t]$ guarantees the minimum execution time as follows:

$$sbf_{inp}\left(s, t, a_e^A, a_{in}^A\right) = \begin{cases} 0, \text{ if } a_{in}^A < 0 \\ \max\left(0, \min\left(t - a_{in}^A, R_{MCS}^{HI}\right) - \left(R_{MCS}^{HI} - \Theta_{MCS}^{HI}\right)\right) \\ , \text{ if } a_{in}^A \neq a_e^A \\ \max\left(0, \min\left(t - a_{in}^A, R_{MCS}^{LO}\right) - \left(R_{MCS}^{LO} - \Theta_{MCS}^{LO}\right)\right) \\ , \text{ if } s = t \\ \max\left(0, \min\left(t - a_{in}^A, R_{MCS}^{MC}\right) - \left(R_{MCS}^{MC} - \Theta_{MCS}^{LO}\right)\right) \\ ,\text{if } a_{in}^A = a_e^A, s - a_e^A \geq \Theta_{MCS}^{LO} \text{and } s \neq t \\ \max\left(0, \min\left(t - a_{in}^A, R_{MCS}^{MC}\right) - \left(R_{MCS}^{MC} - \Theta_{MCS}^{HI}\right)\right) \\ ,\text{if } a_{in}^A = a_e^A, s - a_e^A < \Theta_{MCS}^{LO} \text{and } s \neq t \end{cases} \tag{21}$$

where a_{in}^A denotes the start time of the incomplete server period and is calculated by (17). a_e^A denotes the start time of the mode-switch server period and is calculated by (16). Notably, $a_e^A \leq a_{in}^A$.

Proof: When $s = t$, MC-DS executes in LO-Mode during the time interval $[0, t]$. Next, we consider the cases when $s \neq t$. **Case 1**($a_{in}^A < 0$): According to (17), $a_{in}^A < 0$ indicates that time instant t is smaller than $\Pi_{MCS} - \Theta_{MCS}^{LO}$ (which is equal to a_f^A). Since the server capacity is exhausted at 0 and will be replenished at $\Pi_{MCS} - \Theta_{MCS}^{LO}$, the execution time guaranteed by this period within time interval $[0, t]$ is 0. When $a_{in}^A \geq 0$, as shown in FIGURE 4, the incomplete server period starts at time instant $k * a_f^A$, where $k \geq 0$. We must consider if a mode switch occurs within the incomplete server period since the mode switch may result in additional replenishment. We analyze this in Case 2 and Case 3. **Case 2**($a_{in}^A \neq a_e^A$): Since $a_e^A \leq a_{in}^A$ and $a_{in}^A \neq a_e^A$ indicates that $a_e^A < a_{in}^A$. In this scenario, the mode-switch server period starts before the incomplete server period, implying the incomplete period starts with the stable HI-Mode. The worst-case response time of this period guaranteeing Θ_{MCS}^{HI} time units of capacity is R_{MCS}^{HI}. If $t - a_{in}^A \geq R_{MCS}^{HI}$, the server capacity has been fully consumed before t. Otherwise, if $t - a_{in}^A < R_{MCS}^{HI}$, we assume that the high-criticality server capacity Θ_{MCS}^{HI} is consumed at the very end of the interval R_{MCS}^{HI}. **Case 3**($a_{in}^A = a_e^A$): As depicted in FIGURE 6, the mode switch occurs within the incomplete server period. According to the executing rules of the MC-DS, if $s - a_e^A \geq \Theta_{MCS}^{LO}$, the incomplete server period does not supplement additional capacity. If $s - a_e^A < R_{MCS}^{LO}$, the MC-DS replenishes the server capacity to Θ_{MCS}^{HI}. The WCRT of the mode-switch server period is derived by (12).

In the analysis above, we categorized the server periods within $[0, t]$ into four types: ① periods running in stable low-criticality mode, ② periods running in stable high-criticality mode, ③ the mode-switch server period, and ④ the incomplete server period. Notably, when the mode-switch server period coincides with the incomplete server period (which means $a_e^A = a_{in}^A$), we only need to consider one of them. Therefore, the calculation of $SBF_{MCS}^A(s, t)$ is as follows:

$$SBF_{MCS}^A(s,t) = \begin{cases} sbf^{LO}\left(a_f^A, a_e^A\right) + sbf^{HI}\left(a_h^A, a_{in}^A\right) \\ +sbf_{mcp}\left(a_e^A, s\right) + sbf_{inp}\left(s, t, a_e^A, a_{in}^A\right), \text{if } a_e^A \neq a_{in}^A \\ \\ sbf^{LO}\left(a_f^A, a_e^A\right) + sbf_{inp}\left(s, t, a_e^A, a_{in}^A\right), \text{if } a_e^A = a_{in}^A \end{cases} \tag{22}$$

where $sbf^{LO}\left(a_f^A, a_e^A\right)$ represents the execution time contributed by server periods executing in stable LO-Mode, and its value is derived from (18). $sbf^{HI}\left(a_h^A, a_{in}^A\right)$ represents the execution time contributed by server periods executing in stable HI-Mode. Its value is derived from (19). $sbf_{mcp}\left(a_e^A, s\right)$ calculates the execution time guaranteed by the mode-switch server period, and its value is derived from (20). The execution time contributed by the incomplete server period is denoted as $sbf_{inp}\left(s, t, a_e^A, a_{in}^A\right)$, and its value is derived from (21).

This formulation accounts for the different contributions of each type of server period within the interval $[0, t]$, ensuring an accurate and comprehensive calculation of the processor execution time guaranteed by MC-DSs.

Then, we employ a similar method in deriving SBF under Pattern A to calculate the SBF under Pattern B.

C. SUPPLY BOUND FUNCTION UNDER PATTERN B

When x (see section A for details) is less than or equal to $\Pi_{MCS} - \Theta_{MCS}^{LO}$, we shift the interval $[0, t]$ of Pattern A to the right by x time units, resulting in the processor execution time contributed by the mode-switch server period decreasing, as shown in FIGURE 5. For pattern B, *MCS* first replenishes server capacity at time instant a_f^B during time interval $[0, t]$. The value of a_f^B is given by:

$$a_f^B = a_f^A - x \tag{23}$$

Similarly, for *MCS* running in Pattern B, the start time a_e^B of the mode-switch server period, the end time a_h^B of the mode-switch server period, and the start time a_{in}^B of the incomplete server are calculated as follows:

$$a_e^B = a_e^A - x, a_h^B = a_h^A - x, \ a_{in}^B = a_{in}^A - x \tag{24}$$

By applying the shift, we align the timing of critical server periods with the new arrival pattern B, thus providing an accurate basis for computing the SBF in this pattern. The detailed contributions of processor execution time in pattern B can now be computed using these adjusted timings. This ensures that our analysis accurately reflects the dynamics of server replenishments and mode transitions under pattern B.

$$SBF_{MCS}^B(s,t) = \begin{cases} sbf^{LO}\left(a_f^B, a_e^B\right) + sbf^{HI}\left(a_h^B, a_{in}^B\right) \\ +sbf_{mcp}\left(a_e^B, s\right) + sbf_{inp}\left(s, t, a_e^B, a_{in}^B\right), \text{if } a_e^B \neq a_{in}^B \\ \\ sbf^{LO}\left(a_f^B, a_e^B\right) + sbf_{inp}\left(s, t, a_e^B, a_{in}^B\right), \text{if } a_e^B = a_{in}^B \end{cases} \tag{25}$$

where $sbf^{LO}\left(a_f^B, a_e^B\right)$ can be derived from (18). $sbf^{HI}(a_h^B, a_{in}^B)$ represents the execution time contributed by server periods that start after the mode switch and complete before time. Its value can be derived from (19). $sbf_{mcp}\left(a_e^B, s\right)$ can be derived from (20). The execution time contributed by the incomplete server period is denoted as $sbf_{inp}\left(s, t, a_e^B, a_{in}^B\right)$, and its value can be derived from (21).

D. UPPER BOUND OF THE TEST TIME INTERVAL

In this section, we derive the upper bound for t_{MAX} in Theorem 3. For notational simplicity, we define utilization of component CO as follows:

$$U_{LO}^{LO} = \sum_{\chi_i=LO}^{\tau_i\in CO} C_i^{LO}/T_i,\ U_{HI}^{LO} = \sum_{\chi_i=HI}^{\tau_i\in CO} C_i^{LO}/T_i \quad (26)$$

$$U^{HI} = \sum_{\chi_i=HI}^{\tau_i\in CO} C_i^{HI}/T_i,\ U^{LO} = U_{LO}^{LO} + U_{HI}^{LO} \quad (27)$$

where U_{LO}^{LO} denotes the low-criticality utilization of LO-Tasks, U_{HI}^{LO} denotes the low-criticality utilization of HI-Tasks, U^{LO} denotes the low-criticality utilization of all tasks, and U^{HI} denotes the high-criticality utilization of HI-Tasks.

The utilization of the MC server S is defined as follows:

$$UB^{LO} = \Theta_S^{LO}/\Pi_S,\ UB^{HI} = \Theta_S^{HI}/\Pi_S \quad (28)$$

where UB^{LO} denotes the low-criticality utilization of MC-DSs and UB^{HI} denotes the high-criticality utilization of MC-DSs.

The upper bound of the time interval that needs to be tested in the DBF-based schedulability analysis is derived using a method similar to that in [40]. By assuming that the DBF within a time interval t_{MAX} is larger than the SBF within the same time interval (implying the system is not schedulable), we derive the upper bound of t_{MAX}. If the system is schedulable for each possible value of t_{MAX} that is less than its upper bound, the MC system is schedulable.

Then, the upper bound for t_{MAX} is derived in Lemma 4.

Lemma 5: Suppose the mixed-criticality component CO executing on the MC deferrable server S is not schedulable, $U^{LO} < UB^{LO}$ and $U^{HI} < UB^{HI}$. Then $DBF(s, t_{MAX}) > SBF(s, t_{MAX})$ implies that t_{MAX} must follows (29), as shown at the bottom of the page.

Proof: We assume that a mode switch occurs at time instant s where $0 \le s \le t$. During the interval $[0, t]$, $DBF(s, t)$ is upper bound as follows [8]:

$$\begin{aligned}
&\sum_{\chi_i=LO}^{\tau_i\in CO} \times (s/T_i + 1) \times C_i^{LO} \\
&\quad + \sum_{\chi_j=HI}^{\tau_j\in CO} \left(\frac{s}{T_j} \times C_j^{LO} + C_j^{HI} + \frac{t-s-D_j+T_j}{T_j} \times C_j^{HI}\right) \\
&= U_{LO}^{LO} \times s + \sum_{\chi_i=LO}^{\tau_i\in CO} C_i^{LO} \\
&\quad + \sum_{\chi_j=HI}^{\tau_j\in CO} \left(s \times C_j^{LO} + \left(t-s-D_j+2T_j\right) \times C_j^{HI}\right)/T_j \\
&\le \sum_{\chi_i=LO}^{\tau_i\in CO} C_i^{LO} + \left(U^{LO} - U^{HI}\right) \times s + U^{HI} \times t + U^{HI} \\
&\quad \times \max_{\substack{\chi_j = HI \\ \tau_j \in CO}} \left(2T_j - D_j\right)
\end{aligned} \quad (30)$$

And the lower bound of $SBF(s, t)$ is given by:

$$\begin{aligned}
&\frac{s-\Theta_S^{LO}}{\Pi_S} \times \Theta_S^{LO} + \left(\frac{t-\Theta_S^{LO}}{\Pi_S} - \frac{s-\Theta_S^{LO}}{\Pi_S} - 1\right) \times \Theta_S^{HI} \\
&= \left(UB^{LO} - UB^{HI}\right) \times s + UB^{HI} \times t - UB^{LO} \times \Theta_S^{LO} - \Theta_S^{HI}
\end{aligned} \quad (31)$$

Suppose there exists a time instant t_{MAX} such that $DBF(s, t_{MAX}) > SBF(s, t_{MAX})$, then (30) and (31) deriving that:

$$\sum_{\chi_i=LO}^{\tau_i\in CO} C_i^{LO} + U^{HI} \times \max_{\substack{\chi_j = HI \\ \tau_j \in CO}} \left(2T_j - D_j\right) + UB^{LO} \times \Theta_S^{LO} + \Theta_S^{HI}$$

$$t_{MAX} < \begin{cases} \dfrac{\sum_{\chi_i=LO}^{\tau_i\in CO} C_i^{LO} + U^{HI} \times \max_{\substack{\chi_j = HI \\ \tau_j \in CO}} \left(2T_j - D_j\right) + UB^{LO} \times \Theta_S^{LO} + \Theta_S^{HI}}{UB^{LO} - U^{LO}}, & if\ UB^{LO} - UB^{HI} \le U^{LO} - U^{HI} \\ \dfrac{\sum_{\chi_i=LO}^{\tau_i\in CO} C_i^{LO} + U^{HI} \times \max_{\substack{\chi_j = HI \\ \tau_j \in CO}} \left(2T_j - D_j\right) + UB^{LO} \times \Theta_S^{LO} + \Theta_S^{HI}}{UB^{HI} - U^{HI}}, & if\ UB^{LO} - UB^{HI} > U^{LO} - U^{HI} \end{cases} \quad (29)$$

$$> \left(UB^{HI} - U^{HI}\right) \times t_{MAX} + \left(UB^{LO} - UB^{HI} - \left(U^{LO} - U^{HI}\right)\right) \times s \qquad (32)$$

Since we are trying to derive the upper bound of t_{MAX} and $0 \leq s \leq t_{MAX}$, we eliminate s by considering the value of $UB^{LO} - UB^{HI} - (U^{LO} - U^{HI})$.

Case 1: if $UB^{LO} - UB^{HI} \leq U^{LO} - U^{HI}$, then from (32) we derive the following equation (33), as shown at the bottom of the page.

Case 2: if $B^{LO} - B^{HI} > U^{LO} - U^{HI}$, then from (32) we derive the following equation (34), as shown at the bottom of the page.

VII. EVALUATION

In this section, we evaluate the performance of our proposed approach through simulations using synthetic task sets. Our goal is to demonstrate the effectiveness of the proposed schedulability analysis for component tasks. Specifically, we evaluate the performance of our schedulability analysis based on Theorem 3 and compare the results with the schedulability analysis of the mixed-criticality periodic resource (MCPR) model [8]. Importantly, for consistency in the task model, we set $TL_H = 0$ in the MCPR method. This configuration implies that if any HI-Task within a component exhibits high-criticality behavior, all tasks switch to high-criticality mode. This particular setup has yielded the most effective results in experiments involving MCPR.

A. PARAMETER GENERATION

Since our focus is on the schedulability of component tasks, it is essential to define the resource allocation for each component and ensure that this allocation method makes the system globally schedulable before commencing the experiment. In our experiment, we focus on a system comprising three mixed-criticality components, with the priorities of the components decreasing from the first to the third. We specifically experiment with the component having the lowest priority, i.e., the third component C_3. Each component is assigned either a mixed-criticality deferrable server (using our method) or a mixed-criticality interface (using MCPR). For the *k-th* component in the system, its assigned server period is denoted by Π_k, with a low-criticality budget of Θ_k^{LO} and a high-criticality budget of Θ_k^{HI}. The server utilization in the low-criticality mode is UB_k^{LO}, and the utilization in the high-criticality mode is UB_k^{HI}. Based on the server period and utilization, the server capacity can be calculated as $\Theta_k^{LO} = UB_k^{LO} * \Pi_k$ and $\Theta_k^{HI} = UB_k^{HI} * \Pi_k$.

1) SERVER PARAMETERS

To better observe the experimental results, we set the experimental component C_3 as a high-criticality component. The high-criticality utilization UB_3^{HI} is set as 0.4. We give specific server period and utilization for the MC server associated with C_3 in section B. The specific server allocation for the other components is set as follows: The high-criticality component C_1 is assigned a high-criticality server with the server period $\Pi_1 = 50$, high-criticality utilization $UB_1^{HI} = 0.15$,

$$\sum_{\substack{\chi_i = LO}}^{\tau_i \in CO} C_i^{LO} + U^{HI} \times \max_{\substack{\chi_j = HI \\ \tau_j \in CO}} \left(2T_j - D_j\right) + UB^{LO} \times \Theta_S^{LO} + \Theta_S^{HI}$$
$$> \left(UB^{LO} - U^{LO}\right) \times t_{MAX}$$
$$\rightarrow t_{MAX} < \frac{\sum_{\substack{\chi_i = LO}}^{\tau_i \in CO} C_i^{LO} + U^{HI} \times \max_{\substack{\chi_j = HI \\ \tau_j \in CO}} \left(2T_j - D_j\right) + UB^{LO} \times \Theta_S^{LO} + \Theta_S^{HI}}{UB^{LO} - U^{LO}} \qquad (33)$$

$$\sum_{\substack{\chi_i = LO}}^{\tau_i \in CO} C_i^{LO} + U^{HI} \times \max_{\substack{\chi_j = HI \\ \tau_j \in CO}} \left(2T_j - D_j\right) + UB^{LO} \times \Theta_S^{LO} + \Theta_S^{HI}$$
$$> \left(UB^{HI} - U^{HI}\right) \times t_{MAX}$$
$$\rightarrow t_{MAX} < \frac{\sum_{\substack{\chi_i = LO}}^{\tau_i \in CO} C_i^{LO} + U^{HI} \times \max_{\substack{\chi_j = HI \\ \tau_j \in CO}} \left(2T_j - D_j\right) + UB^{LO} \times \Theta_S^{LO} + \Theta_S^{HI}}{UB^{HI} - U^{HI}} \qquad (34)$$

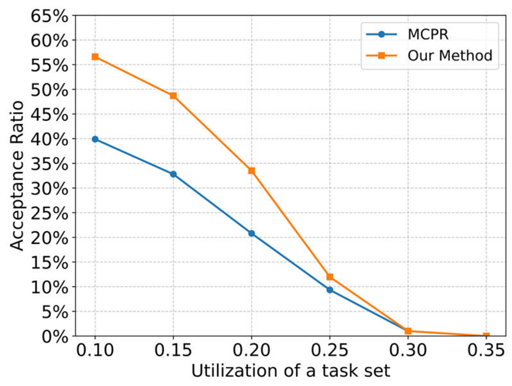


FIGURE 7. **Varying the low-criticality utilization within each component.**

and low-criticality utilization $UB_1^{LO} = 0.75 * UB_1^{HI}$. The low-criticality component C_2 is assigned a low-criticality server with the server period $\Pi_2 = 35$ and low-criticality utilization $UB_2^{LO} = 0.2$. Since all the parameters we use are integers, if the value of the calculated server budget is a decimal, we round it up to the nearest integer.

2) TASK PARAMETERS

Our experiments are based on randomly generated MC tasks. Each task τ_i is generated using the following procedure:

- T_i is generated according to the existing method [41] to ensure that the periods are uniformly distributed in a given range ($T_{\max}/T_{\min} = 10$). Here, $T_{\max}$ and $T_{\min}$ denote the maximum and minimum periods within a task set, respectively.
- *percentHI* denotes the percentage of HI-Tasks in each generated task set, with percentHI $\in [0, 1]$.
- The task utilization u_i is randomly generated by the UUniFast algorithm [42], which efficiently generates utilizations with uniform distributions. Let U^{LO} represents the low-criticality utilization of the component, where $U^{LO} = \sum u_i^{LO}$.
- The low-criticality WECT $C_i^{LO} = u_i \cdot T_i$.
- CF_{HI} denotes the factor C_i^{HI}/C_i^{LO} for HI-Tasks. Therefore, where $C_i^{HI} = CF_{HI} \cdot C_i^{LO}$.
- The relative deadline D_i of each task τ_i is randomly generated from $\left[C_i^{\max} + 0.1 \cdot \left(T_i - C_i^{\max}\right), T_i\right]$, where $C_i^{\max} = \max\limits_{k=i} \left\{C_k^{LO}, C_k^{HI}\right\}$.

B. SIMULATION RESULTS

In each experimental iteration, we generate 1500 task sets. The y-axis in all figures indicates the percentage of schedulable task sets out of the 1500 generated. To illustrate the superiority of our method more effectively, we present the experimental results under various parameter influences. Unless otherwise specified, the low-criticality utilization of the task sets in every experiment is set to $U^{LO} = 0.1$. Additionally, each task set contains 16 tasks, with a 0.2 probability of containing a high-criticality task, where the criticality factor for such tasks is 2.

1) THE IMPACT OF THE UTILIZATION OF THE COMPONENT

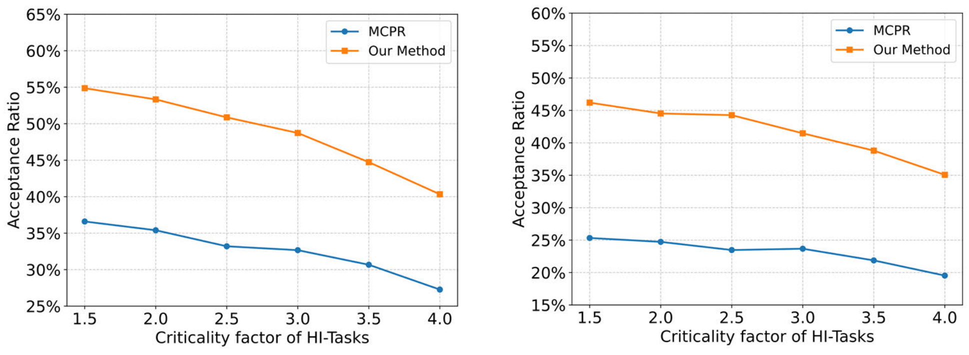
We set the ratio of the low-criticality server utilization UB_3^{LO} to the high-criticality server utilization UB_3^{HI} as 0.8. We configure Π_3 as 110. The experimental result as shown in FIGURE 7 indicates that with increasing utilization of the component in LO-Mode, the acceptance ratio of task sets significantly declines under both methods. Since the utilization of the mixed-criticality server in LO-Mode is slightly less than 0.35, the component is not schedulable when U^{LO} exceeds 0.35. It is evident that our approach demonstrates a more pronounced effect in enhancing the schedulability of task sets when the utilization is relatively low. As utilization increases, the execution time demand of tasks within each time interval rapidly increases. The improvement offered by our method over the existing method is not sufficient to overcome the increased execution demand. Therefore, the gap between the two methods decreases as utilization increases.

2) THE IMPACT OF THE CRITICALITY FACTOR CF_{HI}

The results of this experiment are shown in FIGURE 8. In FIGURE 8(a), Π_3 is set to 110, and in FIGURE 8(b), Π_3 is set to 130. We configure the ratio of the low-criticality server utilization UB_3^{LO} to the high-criticality server utilization UB_3^{HI} as 0.7. As the criticality factor increases, the execution time demand of tasks also increases, leading to a reduction in the number of schedulable task sets. The superiority of our method decreases with the increase of the criticality factor due to the rapid rise in execution demand. Comparing the results in FIGURE 8(a) and FIGURE 8(b), we observe a decline in the acceptance ratio of task sets as the server period increases. This is because the server period represents the frequency of capacity replenishment. When the server utilization is equal, a server with a larger period requires more time to replenish its capacity if the server capacity is exhausted, increasing the risk of task sets becoming unschedulable. However, the ratio of WCRT to the server period decreases as the period increases, making the superiority of our method more pronounced. Specifically, when $\Pi_3 = 110$, our method shows an average improvement of 14% in the acceptance ratio compared to the existing method. However, when $\Pi_3 = 130$, our method demonstrates an average improvement of 18% in the acceptance ratio.

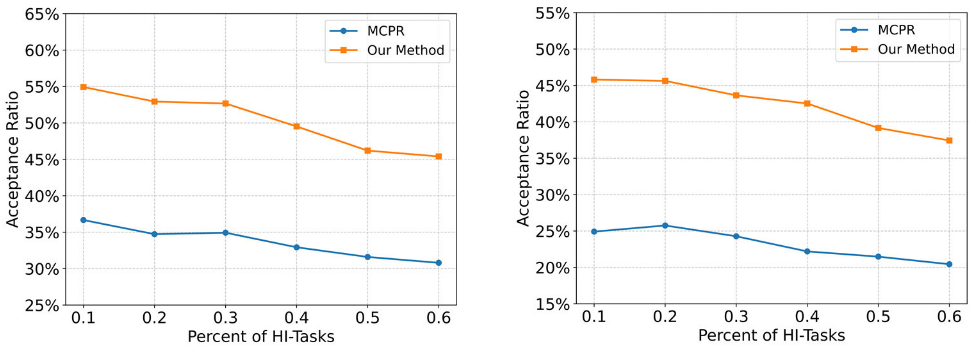
3) THE IMPACT OF PERCENTHI

In this experiment, we set the ratio of the low-criticality server utilization UB_3^{LO} to the high-criticality server utilization UB_3^{HI} as 0.7. In FIGURE.9(a), Π_3 is set to 110, and in FIGURE.9(b), Π_3 is set to 130. We set the x-axis values to range from 0.1 to 0.6. With the increase in the percentage of high-criticality tasks, the acceptance ratio of



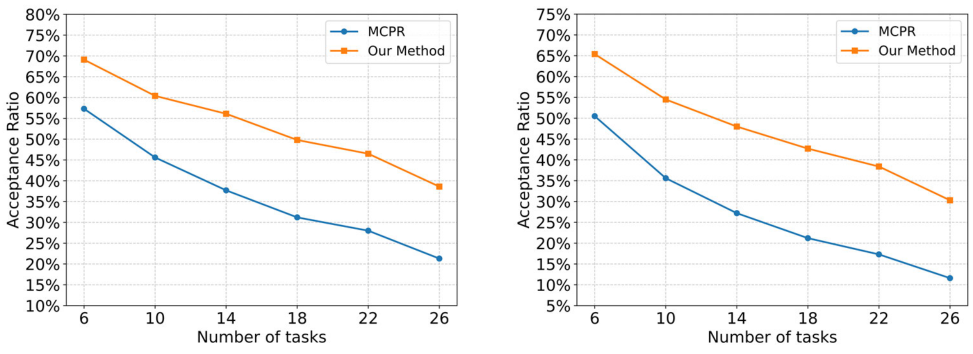
(a) $\Pi_3 = 110$ (b) $\Pi_3 = 130$

FIGURE 8. Varying the criticality factor of HI-Tasks within each component.



(a) $\Pi_3 = 110$ (b) $\Pi_3 = 130$

FIGURE 9. Varying the *percentHI* within each component.



(a) $\Pi_3 = 110$ (b) $\Pi_3 = 130$

FIGURE 10. Varying the number of tasks within each component.

task sets gradually decreases. This trend can be attributed to the augmented execution demand stemming from the greater proportion of high-criticality tasks within the component. Since the execution demand increases relatively slowly when the utilization is low, the decrease in acceptance ratio also drops slowly as *percentHI* increases. Throughout the entire experiment, our method consistently outperforms the existing method, exhibiting an average improvement in the acceptance ratio of approximately 17%.

4) THE IMPACT OF THE NUMBER OF TASKS

In this experiment, the x-axis values range from 6 to 26. As shown in FIGURE.10, we set $\Pi_3 = 110$ in FIGURE.10(a) and $\Pi_3 = 130$ in FIGURE.10(b). The ratio of the low-criticality server utilization UB_3^{LO} to the high-criticality server utilization UB_3^{HI} is 0.7. As the number of tasks increases, the competition for resources among tasks intensifies, making it more challenging to find a scheduling solution that meets all tasks' time constraints. As the number of tasks increases, the utilization of each task within a task set decreases. This condition allows our method to demonstrate superior performance in task scheduling compared to the existing method.

VIII. CONCLUSION

In this paper, we consider a component-based hierarchical scheduling framework where components exhibit different criticalities. Each component comprises a set of constrained-deadline sporadic tasks with dual criticalities. To efficiently schedule these component tasks, we propose the mixed-criticality deferrable server (MC-DS). This server defines its capacity in different modes and specifies how it replenishes capacity upon a mode switch. We employ a preemptive fixed-priority algorithm to schedule the mixed-criticality servers and develop the schedulability analysis for the servers based on worst-case response time (WCRT). Within each MC server, tasks are managed using the earliest deadline first (EDF) scheduler. We derive a schedulability analysis for components based on the demand bound function (DBF) and supply bound function (SBF). In computing the SBF, we integrate the WCRT calculation of the MC-DS to achieve a more precise assessment, thereby reducing the pessimism in the existing schedulability analysis. We conduct extensive experiments to validate our proposed method for analyzing the schedulability of the component tasks. The experimental results demonstrate the superiority of our approach over the existing method.

REFERENCES

[1] R. I. Davis and A. Burns, "Hierarchical fixed priority pre-emptive scheduling," in *Proc. 26th IEEE Int. Real-Time Syst. Symp.*, May 2005, pp. 389–398.

[2] M. Beckert and R. Ernst, "Response time analysis for sporadic server based budget scheduling in real time virtualization environments," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5s, pp. 1–19, Oct. 2017.

[3] J. Martinez, D. Dasari, A. Hamann, I. Sañudo, and M. Bertogna, "Exact response time analysis of fixed priority systems based on sporadic servers," *J. Syst. Archit.*, vol. 110, Nov. 2020, Art. no. 101836.

[4] I. Shin and I. Lee, "Periodic resource model for compositional real-time guarantees," in *Proc. Int. Symp. System Chip*, Jun. 2003, pp. 2–13.

[5] G. Bernat and A. Burns, "New results on fixed priority aperiodic servers," in *Proc. 20th IEEE Real-Time Syst. Symp.*, Apr. 1999, pp. 68–78.

[6] J. K. Strosnider, J. P. Lehoczky, and L. Sha, "The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments," *IEEE Trans. Comput.*, vol. 44, no. 1, pp. 73–91, Jun. 1995.

[7] B. Sprunt, L. Sha, and J. Lehoczky, "Aperiodic task scheduling for hard-real-time systems," *Real-time Syst.*, vol. 1, no. 1, pp. 27–60, Jun. 1989.

[8] X. Gu, A. Easwaran, K.-M. Phan, and I. Shin, "Resource efficient isolation mechanisms in mixed-criticality scheduling," in *Proc. 27th Euromicro Conf. Real-Time Syst.*, Jul. 2015, pp. 13–24.

[9] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *Proc. 28th IEEE Int. Real-Time Syst. Symp. (RTSS)*, Dec. 2007, pp. 239–243.

[10] K. Yang and Z. Dong, "Mixed-criticality scheduling in compositional real-time systems with multiple budget estimates," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2020, pp. 25–37.

[11] A. A. Arafat, S. Vaidhun, L. Liu, K. Yang, and Z. Guo, "Compositional mixed-criticality systems with multiple executions and resource-budgets model," in *Proc. IEEE 29th Real-Time Embedded Technol. Appl. Symp. (RTAS)*, May 2023, pp. 67–79.

[12] J. L. Lorente and J. C. Palencia, "An EDF hierarchical scheduling model for bandwidth servers," in *Proc. 12th IEEE Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA)*, Jun. 2006, pp. 261–266.

[13] S. Baruah, V. Bonifaci, G. Dangelo, H. Li, A. Marchetti-Spaccamela, S. van der Ster, and L. Stougie, "The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems," in *Proc. 24th Euromicro Conf. Real-Time Syst.*, Jul. 2012, pp. 145–154.

[14] P. Ekberg and W. Yi, "Outstanding paper award: Bounding and shaping the demand of mixed-criticality sporadic tasks," in *Proc. 24th Euromicro Conf. Real-Time Syst.*, Jul. 2012, pp. 135–144.

[15] A. Easwaran, "Demand-based scheduling of mixed-criticality sporadic tasks on one processor," in *Proc. IEEE 34th Real-Time Syst. Symp.*, Dec. 2013, pp. 78–87.

[16] D. Liu, N. Guan, J. Spasic, G. Chen, S. Liu, T. Stefanov, and W. Yi, "Scheduling analysis of imprecise mixed-criticality real-time tasks," *IEEE Trans. Comput.*, vol. 67, no. 7, pp. 975–991, Jul. 2018.

[17] V. K. Sundar and A. Easwaran, "A practical degradation model for mixed-criticality systems," in *Proc. IEEE 22nd Int. Symp. Real-Time Distrib. Comput. (ISORC)*, May 2019, pp. 1–26.

[18] Z. Jiang, X. Dai, and N. Audsley, "HIART-MCS: High resilience and approximated computing architecture for imprecise mixed-criticality systems," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2021, pp. 290–303.

[19] H. Su, N. Guan, and D. Zhu, "Service guarantee exploration for mixed-criticality systems," in *Proc. IEEE 20th Int. Conf. Embedded Real-Time Comput. Syst. Appl.*, Aug. 2014, pp. 1–10.

[20] C. Gill, J. Orr, and S. Harris, "Supporting graceful degradation through elasticity in mixed-criticality federated scheduling," in *Proc. Workshop Mixed Criticality Syst.*, 2018, pp. 19–24.

[21] M. Jan, L. Zaourar, and M. Pitel, "Maximizing the execution rate of low criticality tasks in mixed criticality system," in *Proc. WMC*, 2013, pp. 43–48.

[22] H. Su and D. Zhu, "An elastic mixed-criticality task model and its scheduling algorithm," in *Proc. Design, Autom. Test Eur. Conf. Exhibition*, Mar. 2013, pp. 147–152.

[23] S. Ramanathan, A. Easwaran, and H. Cho, "Multi-rate fluid scheduling of mixed-criticality systems on multiprocessors," *Real-Time Syst.*, vol. 54, no. 2, pp. 247–277, Apr. 2018.

[24] H. Su, P. Deng, D. Zhu, and Q. Zhu, "Fixed-priority dual-rate mixed-criticality systems: Schedulability analysis and performance optimization," in *Proc. IEEE 22nd Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA)*, Aug. 2016, pp. 59–68.

[25] J. Lee, H. S. Chwa, L. T. X. Phan, I. Shin, and I. Lee, "MC-ADAPT: Adaptive task dropping in mixed-criticality scheduling," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5s, pp. 1–21, Oct. 2017.

[26] X. Gu and A. Easwaran, "Dynamic budget management and budget reclamation for mixed-criticality systems," *Real-Time Syst.*, vol. 55, no. 3, pp. 552–597, Jul. 2019.

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

[28] Z. Guo, K. Yang, S. Vaidhun, S. Arefin, S. K. Das, and H. Xiong, "Uniprocessor mixed-criticality scheduling with graceful degradation by completion rate," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2018, pp. 373–383.

[29] K. Yang and Z. Guo, "EDF-based mixed-criticality scheduling with graceful degradation by bounded lateness," in *Proc. IEEE 25th Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA)*, Aug. 2019, pp. 1–6.

[30] A. Bhuiyan, S. Sruti, Z. Guo, and K. Yang, "Precise scheduling of mixed-criticality tasks by varying processor speed," in *Proc. 27th Int. Conf. Real-Time Netw. Syst.*, Nov. 2019, pp. 123–132.

[31] K. Yang, A. Bhuiyan, and Z. Guo, "F2VD: Fluid rates to virtual deadlines for precise mixed-criticality scheduling on a varying-speed processor," *Comput.-Aided Des.*, vol. 1, no. 1, pp. 1–9, 2020.
[32] S. Baruah and Z. Guo, "Mixed-criticality scheduling upon varying-speed processors," in *Proc. IEEE 34th Real-Time Syst. Symp.*, Dec. 2013, pp. 68–77.
[33] S. Baruah and Z. Guo, "Scheduling mixed-criticality implicit-deadline sporadic task systems upon a varying-speed processor," in *Proc. IEEE Real-Time Syst. Symp.*, Dec. 2014, pp. 31–40.
[34] A. Burns and R. I. Davis, *Mixed Criticality Systems—A Review*, 13th ed., Feb. 2022. [Online]. Available: https://eprints.whiterose.ac.uk/183619/
[35] R. R. L. Sha and J. P. Lehoczky, "Solutions for some practical problems in prioritized preemptive scheduling," in *Proc. IEEE Real-Time Syst. Symp.*, Jul. 1986, pp. 181–191.
[36] J. P. Lehoczky, L. Sha, and J. K. Strosnider, "Enhanced aperi-odic responsiveness in hard real-time environments," in *Proc. Real-Time Syst. Symp.*, 1987, pp. 261–270.
[37] A. Hamann, D. Dasari, J. Martinez, and D. Ziegenbein, "Response time analysis for fixed priority servers," in *Proc. 26th Int. Conf. Real-Time Netw. Syst.*, Oct. 2018, pp. 254–264.
[38] R. I. Davis and A. Burns, "Resource sharing in hierarchical fixed priority pre-emptive systems," in *Proc. 27th IEEE Int. Real-Time Syst. Symp. (RTSS)*, May 2006, pp. 257–270.
[39] P. Ekberg and W. Yi, "Bounding and shaping the demand of generalized mixed-criticality sporadic task systems," *Real-Time Syst.*, vol. 50, no. 1, pp. 48–86, Jan. 2014.
[40] S. K. Baruah, A. K. Mok, and L. E. Rosier, "Preemptively scheduling hard-real-time sporadic tasks on one processor," in *Proc. 11th Real-Time Syst. Symp.*, 1990, pp. 182–190.
[41] R. I. Davis, A. Zabos, and A. Burns, "Efficient exact schedulability tests for fixed priority real-time systems," *IEEE Trans. Comput.*, vol. 57, no. 9, pp. 1261–1276, Sep. 2008.
[42] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Syst.*, vol. 30, nos. 1–2, pp. 129–154, May 2005.

DANDI MA (Member, IEEE) was born in Sichuan, China, in 1999. She received the B.S. degree in the Internet of Things from Xihua University, in 2021. She is currently pursuing the M.S. degree with the College of Computer and Information Science, Southwest University, China. Her research interests include real-time scheduling, schedulability analysis, and embedded real-time systems.