Contestion-Free Scheduling for Mixed-Criticality Multiprocessor Real-Time System

Hyeongboo Baek 1, and Kilho Lee 2,*

1 Department of Computer Science and Engineering, Incheon National University (INU), Incheon 22012, Korea; hbbaek@inu.ac.kr
2 Department of Smart Systems Software, Soongsil University (SSU), Seoul 06978, Korea
* Correspondence: khlee.cs@ssu.ac.kr; Tel.: +82-2-820-0954

Received: 31 July 2020; Accepted: 9 September 2020; Published: 14 September 2020

Abstract: Zero-laxity (ZL) and contention-free (CF) policies have received considerable attention owing to their simplicity and applicability to real-time systems equipped with symmetry multiprocessors. Recently, the ZL policy for mixed-criticality (MC) systems has been proposed and studied, but the applicability to and performance of the CF policy for MC systems have not been investigated yet. In this paper, we propose the CF policy (as a scheduling policy) for MC symmetry multiprocessor systems, referred to as the MC systems tailored CF policy (MC-CF), and a schedulability analysis in support thereof. We define the notion of contention-free slots for two different criticalities (of MC systems) of tasks, propose a technique to limit the amount to be utilized for each task by defining an upper bound, and subsequently explain the way in which the contention-free slots are systematically utilized to improve the schedulability of MC symmetry multiprocessor systems. Following this, we develop a deadline analysis (DA) for MC-CF. Using our experimental results under various environmental settings, we demonstrate that MC-CF can significantly improve the schedulability of fixed-priority scheduling.

Keywords: hard real-time systems; schedulability; zero-laxity policy; symmetry multiprocessor platform; contention-free

1. Introduction

Real-time embedded systems differ from general-purpose systems in that designated multiple real-time tasks (e.g., engine control and sensing) are repeatedly executed on limited computing resources (e.g., CPU and memory). At the same time, real-time tasks need to be conducted within a predefined time, known as a deadline, as a strict design requirement. To achieve this goal and considering the characteristics of embedded systems, the following two problems have been extensively studied by the real-time systems community: (i) ways to effectively allocate the limited computing resources to real-time tasks and (ii) approaches to judge whether the system satisfies the design requirement. Methodologies to address (i) and (ii) are referred to as the real-time scheduling algorithm and schedulability analysis, respectively. In the case of uniprocessor systems, these two methodologies have been fully matured as optimal scheduling algorithms, namely, the earliest deadline first (EDF) and the rate monotonic (RM) algorithms, have been developed and extensively studied, and the exact schedulability analyses to support them were proposed [1]. With regard to multiprocessor systems, optimal techniques such as ER-Fair, LLREF, RUN, and EKG have been suggested [2–5], and have evolved to reduce their scheduling costs [6].

However, these research achievements targeted a system in which real-time tasks have an equal impact on the safety of the system, known as the criticality, which does not support an increasingly important trend in the design of real-time systems considering different criticalities.
of tasks. For example, modern automotive systems conduct safety-critical tasks (e.g., wheel steering and breaking systems) as well as mission-critical ones (e.g., the camera and various sensing systems), the various criticalities of which are explicitly categorized in the ISO 26262 standard [7]. Although a mixed-criticality (MC) system should support different criticalities with high CPU utilization, relatively few approaches have been proposed [8–10] compared to those for single-criticality (SC) systems.

For example, the zero-laxity (ZL) [11] and contention-free (CF) [12,13] policies have received considerable attention owing to their simplicity and applicability to SC systems equipped with multiprocessors. ZL and CF policies are incorporated into most (if not all) existing scheduling algorithms and improve the real-time scheduling performance of SC systems in that they have opposite objectives. The ZL policy promotes the priority of a task to the highest level at a time instant when it should be scheduled to avoid missing the deadline, whereas the CF policy relegates the priority to the lowest level at the time instant at which its schedulability is guaranteed. Recently, the ZL policy for MC systems was proposed and studied [14,15], but the applicability and performance of the CF policy for MC systems have not been investigated.

In this paper, we propose the CF policy (as a scheduling policy) for MC multiprocessor systems, referred to as the MC systems tailored CF policy (MC-CF), in addition to the schedulability analysis to support the policy. We consider the fixed-priority (FP) and earliest-deadline first (EDF) scheduling as the base scheduling algorithms where MC-CF is applied. The key idea of the CF policy is to exploit the notion of a contention-free slot, which is defined as the time slot in which the number of active (i.e., released but unfinished) jobs is less than or equal to the number of processors [12,13]. In other words, in contention-free slots, active jobs can be executed without any contention to obtain processors (or cores). We define the notion of contention-free slots for two different criticalities (of MC systems) of tasks, propose a technique to restrict the amount to be utilized for each task by defining an upper bound, and then present the way in which the contention-free slots are systematically utilized to improve the schedulability of MC multiprocessor systems. Then, we develop the deadline analysis (DA) [16] for MC-CF. Finally, we conducted experiments with settings in various environments to demonstrate that MC-CF can significantly improve the schedulability of fixed-priority scheduling.

The following summarizes the contributions of our study:

- We propose a method to calculate an upper bound of the number of contention-free slots for each task;
- We then propose MC-CF;
- We develop the DA test for MC-CF.

The remainder of this paper is organized as follows. Section 3 presents our considered system model. Section 4 first defines the notion of contention-free slot for MC systems and explains how to calculate the upper bounded number of contention-free slots that exit for each task. Then, MC-CF is proposed. Section 5 presents DA for MC-CF proposed in this paper. In Section 6, the performance of MC-CF is discussed. Section 2 summarizes the related work, and Section 7 concludes this paper.

2. Related Work

For the last few decades, a number of studies on real-time systems have focused on developing scheduling algorithms for multiprocessor systems to improve schedulability of the target systems. ER-Fair, LLREF, EKG, and RUN were proposed as optimal scheduling algorithms for multiprocessor systems, and have evolved to reduce their scheduling costs [6]. For example, RUN transforms the problem of task scheduling for multiprocessors into that of a uniprocessor, and reduced the number of preemptions significantly compared to the other techniques that exploits rated-based approaches.

On the other hand, another approach has tried to develop scheduling policies that can be applied to most (if not all) existing scheduling algorithms to improve schedulability analysis. The ZL [11] and CF policies [12,13] are representative examples the such techniques. The ZL policy promotes the priority of a task to the highest level at a time instant when it should be scheduled to avoid missing the
deadline, whereas the CF policy relegates the priority to the lowest level at the time instant at which its schedulability is guaranteed. The CF policy were further studied to better exploit contention-free slots [17], and the response-time analysis (RTA) for the CF policy was proposed [18].

While the studies above are for single-criticality systems that for mixed-criticality ones was initiated by the seminal work of Vestal [19], and a number of studies have been conducted for uniprocessor platforms. Baruah proposed RTA [20] for static mixed-criticality scheduling, and proposed a new MC-scheduling algorithm called the earliest-deadline first virtual deadline (EDF-VD) [21,22]. Li proposed new criticality based priority (OCBP) scheduling algorithm [23,24], and OCBP was extended to PLRS by Guan [24]. Baruah also proposed a more general model for MC scheduling, wherein the task parameter can be varied according to its criticality [25]. The task model in [25] assumes that WCET estimates are non-decreasing, period estimates are non-increasing, and relative deadline estimates are non-decreasing with increasing criticality because a higher criticality requires more conservative estimates.

With respect to the multiprocessor platform, Pathan proposed an OPA-compatible schedulability analysis for FP scheduling, and Li extended EDF-VD to multiprocessor MC systems [26]. Lee applied fluid scheduling to MC systems in which tasks have different execution rates according to their criticality [27]. In [14], Jung first applied the ZL policy to EDF for MC systems, and proposed a new RTA to support it. Mollison proposed a rather practical technique to utilize a concept of “slack” defined as the gap between the worst-case execution time prediction and actual execution times [24]. In [24], high-criticality tasks provide slack, and low-criticality tasks utilize it to improve schedulability.

Our study is motivated by that the CF policy has been investigated a lot for single-criticality systems, but it is not for mixed-criticality ones while the ZL policy for MC systems was proposed and studied [14,15]. The DA test is also proposed in [12,13] to support the CF policy, but it only supports single-criticality systems. That is, the worst-case scenario to estimate the worst-case interference from higher-priority jobs for a task \( \tau_k \) only consider single-criticality systems. To the best of our knowledge, we first proposed the CF policy and its schedulability analysis to support mixed-criticality systems.

### 3. System Model

We consider an \( m \) processor system operating real-time tasks of which the criticalities are categorized by two levels, \( HI \) and \( LO \). A task set \( \tau \) in which a task \( \tau_i \) is specified by a five-tuple of parameters \((X_i, C_{i}^{LO}, C_{i}^{HI}, D_i, T_i)\) each of which parameter is specified as follows:

- \( X_i \in \{HI, LO\} \) is the criticality of a task, and a task \( \tau_i \) with \( X_i = HI \) or \( X_i = LO \) is either a HI-criticality or LO-criticality task, respectively.
- \( C_{i}^{LO} \) and \( C_{i}^{HI} \) denote the worst-case execution times (WCETs) for the low and high criticalities, respectively. We assume that \( C_{i}^{LO} \leq C_{i}^{HI} \) because a task with higher criticality requires a more conservative analysis for WCET.
- \( D_i \) and \( T_i \) represent the relative deadline and the period of the task, respectively, and \( D_i = T_i \) is satisfied.

\( \tau^{LO} \) and \( \tau^{HI} \) represent a set of tasks each whose criticality \( X_i \) is \( HI \) and \( LO \), respectively; by definition, \( \tau^{LO} \cup \tau^{HI} = \tau \) and \( \tau^{LO} \cap \tau^{HI} = \emptyset \) hold.

A task \( \tau_i \) invokes a series of jobs, and the \( n \)-th job \( J_n^i \) is released at \( r_n^i \) and finished at \( f_n^i \). We use \( f_i \) (likewise \( r_i \) and \( f_i \)) if it indicates an arbitrary job released by \( \tau_i \). We assume that the release times of two jobs that are consecutively released by the same task \( \tau_i \) are at least \( T_i \) apart from each other. We call a released job before its completion an active job. The absolute deadline \( d_n^i \) of a job \( J_n^i \) is defined as \( d_n^i = r_n^i + D_i \). We use \( f_i \), when it denotes an arbitrary job of \( \tau_i \). \( J_n^i \) is schedulable if it completes its execution before \( d_n^i \), and \( \tau_i \) is schedulable if every job released by \( \tau_i \) is schedulable. In addition, \( \tau \) is schedulable if all tasks in \( \tau \) are schedulable.

We consider global, preemptive, and work-conserving FP and EDF scheduling. Scheduling with these characteristics enables an active job to migrate from one processor to another at every
scheduling decision (e.g., a job release, completion, and priority demotion), assigns a processor to its higher-priority job, and processors are always utilized by active jobs if any. Fixed-priority scheduling assigns a priority to a task rather than a job, and thus all jobs released by the same task have the same priority. On the other hand, EDF assigns a priority to a job rather than a task. Table 1 lists up notations used throughout this paper.

Table 1. Notations and their description.

| Notation | Description | Notation | Description |
|----------|-------------|----------|-------------|
| $m$ | the number of processors | $\tau$ | a task set |
| $\tau_i$ | a task in $\tau$ | $X_i$ | the criticality of $\tau_i$ |
| $C_{LO}^i$ | WCET for the low criticality | $C_{HI}^i$ | WCET for high criticality |
| $D_i$ | the relative deadline of $\tau_i$ | $T_i$ | the period of $\tau_i$ |
| $\tau^{LO}$ | a set of tasks of which the criticality is low | $\tau^{HI}$ | a set of tasks of which the criticality is high |
| $C_{LO}$ | WCET for the low criticality | $C_{HI}$ | WCET for high criticality |
| $D_i$ | the relative deadline of $\tau_i$ | $T_i$ | the period of $\tau_i$ |
| $\tau^{LO}$ | a set of tasks of which the criticality is low | $\tau^{HI}$ | a set of tasks of which the criticality is high |
| $J_{i}^{n}$ | the $n$-th job invoked by $\tau_i$ | $r_i$ | the release time of $J_i$ |
| $f_i$ | the finishing time of $f_i$ | $d_i$ | the absolute deadline of $J_i$ |
| $hep(\tau_i)$ | a set of tasks of which the priority of each is higher than $\tau_i$ | $\Phi_{i}^{LO}$ | the lower bound of the contention-free slots for $\tau_i \in \tau^{LO}$ |
| $\Phi_{i}^{HI}$ | the lower bound of the contention-free slots for $\tau_i \in \tau^{HI}$ | $C_{i}^{(t)}$ | the remaining execution time of $J_i$ at $t$ |
| $\Phi_{i}^{(t)}$ | the remaining contention-free slots of $J_i$ at $t$ | $Q^{H}$ | the queue in which tasks have their original priorities |
| $Q^{L}$ | the queue in which tasks have demoted priorities | $\Phi_{k}^{HI}(\ell')$ | the lower bound of the contention-free slots for $\tau_i \in \tau^{HI}$, assuming that a mode transition occurs at $t_i + \ell'$ |
| $I_{LO,k}^{LO}(\ell)$ | the interference of $\tau_i$ on $\tau_k \in \tau$ in the LO-mode | $I_{HI,k}^{LO}(\ell, \ell')$ | the interference $I_{HI,k}^{LO}(\ell, \ell')$ of $\tau_i$ on $\tau_k \in \tau^{HI}$, assuming that a mode transition occurs at $t_i + \ell'$ |

The considered MC systems support two operating modes, LO- and HI-modes, respectively. The system starts in the LO-mode, and the system transition to HI-mode occurs at a time instant when any job $J_i$ executes for more than its $C_{LO}^i$. After a mode transition, we assume that jobs of tasks $\tau_i \in \tau^{HI}$ execute for up to $C_{HI}^i$, and jobs of tasks $\tau_i \in \tau^{LO}$ are aborted (i.e., dropped). A task set $\tau$ scheduled on the MC systems is said to be schedulable if and only if the following is satisfied:

- In the LO-mode, every task $\tau_i \in \tau$ is schedulable; and
- In the HI-mode, every task $\tau_i \in \tau^{HI}$ is schedulable.
4. MC-CF

In this section, we first describe existing MC scheduling as background. We then present the way in which MC-CF effectively schedules tasks by utilizing the notion of contention-free slots.

4.1. MC-CF Scheme

MC-CF scheme utilizes the notion of contention-free slots that is formally defined as follows.

**Definition 1.** (contention-free slot (from [13])): A time slot is called contention-free if the number of active jobs is less than or equal to the number of processors $m$ at the slot; it is called a contending slot, otherwise.

By definition, if the remaining execution (to be finished) of a job is less than or equal to the remaining contention-free slots that will be encountered before its deadline, the schedulability of a job is guaranteed because the remaining execution can proceed without any contention. The underlying idea of the CF policy is to improve the schedulability by transferring the execution of certain jobs in contending slots to contention-free slots to improve the chances of lower-priority tasks being scheduled in contending slots. This migration of executions is conducted by relegation of the priority (at a certain time $t$) of jobs whose schedulabilities are guaranteed by comparing their remaining executions and contention-free slots. Let $\Phi^L_i$ and $\Phi^H_i$ be the lower bound of the contention-free slots that exist in an interval $[r_i, d_i)$ for $\tau_i \in \tau^L$ and $\tau_i \in \tau^H$, respectively; the calculation of thereof is explained in detail in the next subsection. In addition, let $C_i(t)$ and $\Phi_i(t)$ be the amount of remaining execution time of $J_i$ at $t$ to be finished and the remaining contention-free slots (also, lower bounded) that exist before $d_i$.

MC-CF exploits two separated queues $Q^H$ and $Q^L$ where any job in $Q^H$ has a higher priority than any job in $Q^L$. The following are the detailed steps of the CF policy.

Algorithm 1 presents how MC-CF operates. Before the system starts, MC-CF calculates the lower bound of the contention-free slots $\Phi^L_i$ and $\Phi^H_i$ that exists in the interval $[r_i, d_i)$ for $\tau_i \in \tau^L$ and $\tau_i \in \tau^H$, respectively (Line 1). Then, MC-CF operates the following at each time slot $t$ (Line 2); note that the system starts in the LO-mode. If a job $J_i$ is released, then it sets the remaining contention-free slots $\Phi_i(t)$ to $\Phi^L_i$ (for $\tau_i \in \tau^L$) or $\Phi^H_i$ (for $\tau_i \in \tau^H$) and the remaining execution time $C_i(t)$ to $C^L_i$ (in the LO-mode) or $C^H_i$ (in the HI-mode), and puts $J_i$ to $Q^H$ (Lines 3–7). If a job $J_i$ in $Q^H$ holds $\Phi_i(t) \geq C_i(t)$, then it moves to $Q^L$, which implies that the remaining execution $C_i(t)$ can be performed without deadline miss even if its priority is demoted to the lowest. In addition, $\Phi_i(t)$ of each active job $J_i$ in $Q^H$ is deducted by 1 if the current slot is contention-free (i.e., $|Q^H| + |Q^L| \leq m$ holds) (Lines 13–17). Then, the remaining execution time $C_i(t)$ of $m$-highest priority jobs is deducted by 1 (Lines 18–22). If a mode transition occurs, all LO-criticality jobs are dropped and the remaining execution of HI-criticality jobs increases by $C^H_i - C^L_i$ (Lines 23–26). The difference between Algorithm 1 in [13] and Algorithm 1 in our paper is as follows. In our paper, high- and low-criticality tasks have their own contention-free slots (i.e., $\Phi^L_i$ and $\Phi^H_i$), respectively (in Line 4). In addition, they have their own worst-case execution times (i.e., $C^L_i$ and $C^H_i$), respectively (in Line 5). Finally, Algorithm 1 in our paper support a mode transition conducted by mixed-criticality systems (in Lines 23–26).
Algorithm 1 CF policy for MC multiprocessor systems.

1: Calculate $\Phi_i^{LO}$ for every $\tau_i \in \tau^{LO}$, and $\Phi_i^{HI}$ for every $\tau_i \in \tau^{HI}$.
2: for Every time slot $t$ do
3: \hspace{1em} if $J_i$ is released by $\tau_i$ then
4: \hspace{2em} Set $\Phi_i^{(t)}$ ← $\Phi_i^{LO}$ (for $\tau_i \in \tau^{LO}$) or $\Phi_i^{HI}$ (for $\tau_i \in \tau^{HI}$).
5: \hspace{2em} Set $C_i^{(t)} ← C_i^{LO}$ (in the LO-mode) or $C_i^{HI}$ (in the HI-mode).
6: \hspace{2em} Put the job in $Q^{H}$.
7: \hspace{1em} end if
8: \hspace{1em} for Each job $J_i$ in $Q^{H}$ do
9: \hspace{2em} \hspace{1em} if $\Phi_i^{(t)} \geq C_i^{(t)}$ then
10: \hspace{3em} \hspace{1em} Move $J_i$ to $Q^{L}$.
11: \hspace{2em} \hspace{1em} end if
12: \hspace{1em} end for
13: \hspace{1em} if $|Q^{H}| + |Q^{L}| \leq m$ then
14: \hspace{2em} \hspace{1em} for Each job $J_i$ in $Q^{H}$ do
15: \hspace{3em} \hspace{1em} Update $\Phi_i^{(t+1)} ← \max(0, \Phi_i^{(t)} - 1)$.
16: \hspace{2em} \hspace{1em} end for
17: \hspace{1em} end if
18: \hspace{1em} Prioritize jobs in $Q^{H}$ according to the considered base algorithm.
19: \hspace{1em} for Each job $J_i$ selected among the (up to) $m$ highest-priority jobs do
20: \hspace{2em} \hspace{1em} Update $C_i^{(t)} ← C_i^{(t)} - 1$.
21: \hspace{2em} \hspace{1em} If the job is finished, remove the job from its queue.
22: \hspace{2em} end for
23: \hspace{1em} if A mode transition occurs then
24: \hspace{2em} \hspace{1em} Drop all tasks $\tau_i$ in $\tau^{H}$, they are not scheduled any more.
25: \hspace{2em} \hspace{1em} Update $C_i^{(t)} ← C_i^{(t)} + C_i^{HI} - C_i^{LO}$ for all active jobs of tasks $\tau_i$ in $\tau^{HI}$.
26: \hspace{2em} end if
27: \hspace{1em} end for

4.2. Lower Bound of Contention-Free Slots

The key technique of MC-CF is to calculate the lower bound of the contention-free slots $\Phi_i^{LO}$ and $\Phi_i^{HI}$ that exists in an interval $[r_i, d_i]$ for each task $\tau_i \in \tau^{LO}$ and $\tau_i \in \tau^{HI}$, respectively (Line 1 of Algorithm 1). The underlying idea of the technique is to calculate the upper bound of the maximum workload conducted in an interval $[r_i, d_i]$, and then derive the lower bound of contending slots in the interval. Thereafter, the lower bound of contention-free slots $\Phi_i^{LO}$ and $\Phi_i^{HI}$ is obtained by Definition 1.

We first derive $\Phi_i^{LO}$ (for $\tau_i \in \tau^{LO}$) with the following strategy. Let $\ell_i^{LO}$ be the maximum workload of a task $\tau_i$ (whose $X_i$ can be either $LO$ or $HI$) in an interval of length $\ell$ in the LO-mode. Figure 1a shows the worst-case scenario where $\ell_i^{LO}$ occurs [14]. As shown in Figure 1a, the last (i.e., the right-most job in Figure 1a) job starts its execution at $r^T$, and finishes it at the end of the interval of length $\ell$. The remaining jobs (i.e., the left-most and middle jobs) start their executions as late as possible and end at their absolute deadlines. Considering a job $J_i$ fully executing for $C_i^{LO}$ (e.g., the rightmost and middle jobs in Figure 1a) and jobs executing the portion of $C_i^{LO}$, $\ell_i^{LO}(\ell)$ is calculated as follows:

$$\ell_i^{LO}(\ell) = N_i^{LO}(\ell) \cdot C_i^{LO} + \min(C_i^{LO}, \ell + D_i - C_i^{LO} - N_i^{LO}(\ell) \cdot T_i), \tag{1}$$

where $N_i^{LO}(\ell)$ is the number of jobs executing for $C_i^{LO}$ given by

$$N_i^{LO}(\ell) = \frac{\ell + D_i - C_i^{LO}}{T_i}. \tag{2}$$

Then, the following derives the lower-bounded number of contention-free slots $\Phi_i^{LO}$ of $\tau_k \in \tau^{LO}$ in an interval of length $D_k$ (i.e., for $(r_k, d_k)$) in the LO-mode.
Lemma 1. For a job $J_k$ of a task $\tau_k \in \tau^{LO}$ scheduled by MC-CF in the LO-mode, there are at least $\Phi_k^{LO}$ contention-free slots in $[r_k,d_k)$, which is calculated as follows:

$$\Phi_k^{LO} = \max \left( 0, D_k - \left\lfloor \frac{\sum_{i \in r_k \setminus n_k} W_i^{LO}(D_k)}{m} \right\rfloor \right).$$

Proof. By definition, the workload (or execution) of a task $\tau_i$ is upper bounded by $W_i^{LO}(\ell)$ in the LO-mode, and at least $m$ executions are needed for a time slot to be contending. Thus, there are at least $D_k - \min \left( D_k, \left\lfloor \frac{\sum_{i \in r_k \setminus n_k} W_i^{LO}(D_k)}{m} \right\rfloor \right)$ contention-free slots in the interval $[r_k,d_k)$ for $\tau_k \in \tau^{LO}$. Then, $W_k^{LO}(D_k)$ is reduced to $C_k^{LO}$ because only one job is released in $[r_k,d_k)$. Thus, the lemma holds.

Note that $\tau_k \in \tau^{LO}$ is immediately dropped at the mode transition, and we do not care about the schedulability of the task. That is, the number of contention-free slots that exist in $[r_k,d_k)$ does not need to be considered to guarantee the schedulability for $\tau_k \in \tau^{LO}$ in the presence of a mode transition. However, MC-CF sets $\Phi_k(t) \leftarrow \Phi_k^{LO}$ for $\tau_k \in \tau^{LO}$ (Line 4 in Algorithm 1) even though it is unable to foresee the timing of the mode transition. Then, the schedulability analysis proposed in Section 5 takes this behavior of MC-CF into account to judge the schedulability of $\tau_k \in \tau^{HI}$.

With regard to $\Phi_i^{HI}$ for $\tau_i \in \tau^{HI}$, we need to consider that a mode transition can occur in $[r_i,d_i)$ because such a task can execute even after the mode transition. Let $W_i^{HI}(\ell, \ell')$ be the maximum workload of a task $\tau_i \in \tau^{HI}$ in an interval of length $\ell$ assuming that a mode transition occurs at the end of $\ell'$, $\ell'$ is assumed to start at the beginning of $\ell$. Figure 1b presents the scenario where $W_i^{HI}(\ell, \ell')$ occurs. As shown in Figure 1b, the last job $J_i^n$ released after the mode transition starts its execution for $C_i^{HI}$ at $r_i^n$ and finishes it at the end of the interval of length $\ell$. When it comes to the middle job $J_i^{n-1}$ that a mode transition occurs in $[r_i^{n-1}, d_i^{n-1})$, it executes for $C_i^{HI}$. The left-most job $J_i^{n-2}$ that a mode transition occurs after $d_i^{n-2}$ starts its execution as late as possible and finishes it at $d_i^{n-2}$. Considering a job $J_i$ fully executing for $C_i^{HI}$ (e.g., the rightmost and middle jobs in Figure 1b) and jobs executing the portion of $C_i^{LO}$, $W_i^{HI}(\ell, \ell')$ is calculated as follows:

$$W_i^{HI}(\ell, \ell') = N_i^{HI}(\ell - \ell') \cdot C_i^{HI} + W_i^{LO}(\chi) + \min(\chi - \ell', C_i^{HI} - C_i^{LO})$$

Figure 1. Scenarios where the workload of a task $\tau_i$ is maximized in an interval of length $\ell$. 

(a) for $W_k^{HI}(\ell)$

(b) for $W_i^{HI}(\ell, \ell')$
where \(N^H_i(\ell, \ell')\) is the number of jobs executing for \(C^H_i\) given by

\[
N^H_i(\ell, \ell') = \left\lfloor \frac{\ell - \ell' + D_i - C^H_i}{T_i} \right\rfloor, 
\]

and \(\chi\) is given by

\[
\chi = \ell + D_i - C^H_i - N^H_i(\ell, \ell) \cdot T_i. 
\]

Then, considering tasks \(\tau_i \in \tau^LO\) that are dropped at the end of \(\ell'\), the following derives the lower-bounded number of contention-free slots \(\Phi^H_k(\ell')\) of \(\tau_k \in \tau^HI\) in an interval of length \(D_k\) (i.e., for \([r_k, d_k]\)) for a given \(\ell'\).

**Lemma 2.** For a job \(J_k\) of a task \(\tau_k \in \tau^HI\) scheduled by MC-CF when FP is applied, there are at least \(\Phi^H_k(\ell')\) contention-free slots in \([r_k, d_k]\) for a given \(\ell'\), which is calculated as follows:

\[
\Phi^H_k(\ell') = \max \left(0, D_k - \left[\chi^H + \sum_{\tau \in \tau^HI} W^H_i(\ell') + \sum_{\tau \in \tau^HI \setminus \tau_k} W^H_i(D_k, \ell')\right] \right). 
\]

**Proof.** With given \(\ell'\), the workload (or execution) of \(\tau_i \in \tau^LO\) is upper bounded by \(W^L_i(\ell')\) and that of \(\tau_i \in \tau^HI\) is upper bounded by \(W^H_i(\ell, \ell')\). In addition, at least \(m\) executions are needed for a time slot to be contending. Thus, there are at least \(D_k - \min \left(D_k, \frac{\chi^H + \sum_{\tau \in \tau^HI} W^L_i(\ell') + \sum_{\tau \in \tau^HI \setminus \tau_k} W^H_i(D_k, \ell')}{m}\right)\) contention-free slots in the interval \([r_k, d_k]\) for \(\tau_k \in \tau^HI\). Then, \(W^H_k(D_k)\) is reduced to \(C^H_k\) because only one job is released in \([r_k, d_k]\). Thus, the lemma holds. \(\square\)

Because MC-CF is unable to predict the timing of the mode transition, we need to consider all \(\ell'\) for \(0 \leq \ell' \leq D_k\) to derive the lower bound of contention-free slots in \([r_k, d_k]\) for \(\tau_i \in \tau^HI\). Thus, \(\Phi^H_k\) of \(\tau_k \in \tau^HI\) in an interval of length \(D_k\) (i.e., for \([r_k, d_k]\)) with an arbitrary \(\ell'\) for \(0 \leq \ell' \leq D_k\) is derived as follows.

**Lemma 3.** For a job \(J_k\) of a task \(\tau_k \in \tau^HI\) scheduled by MC-CF, there are at least \(\Phi^H_k\) contention-free slots in \([r_k, d_k]\) with an arbitrary \(\ell'\) for \(0 \leq \ell' \leq D_k\), which is calculated as follows:

\[
\Phi^H_k = \min \left\{\Phi^H_k(\ell') \mid 0 \leq \ell' \leq D_k\right\} 
\]

5. Schedulability Analysis for MC-CF

In this section, we propose a new schedulability analysis for MC-CF. We first present a schedulability analysis for the LO-mode by extending the existing DA for MC-CF. We then present that, in the HI-mode, higher-priority tasks are observed to interfere with a task. Note that a task set \(\tau\) scheduled on the MC system is schedulable if it is schedulable for both LO- and HI-modes, as presented in Section 3. According to Lemmas 1 and 2, \(\Phi^L_k\) and \(\Phi^H_k\) can be greater than \(C^L_k\) and \(C^H_k\), respectively, but we upper-bound it to \(C^L_k\) and \(C^H_k\), respectively, for the correctness of the schedulability analysis that will be presented in this section.

5.1. Schedulability Analysis for LO-Mode

The underlying idea of DA is to judge whether every job \(J_k\) of \(\tau_k\) can complete its execution in \([r_k, d_k]\) by investigating whether the upper bound of the interference is smaller than or equal to \(D_k - C^L_k\) for the LO-mode and \(D_k - C^L_k\) for the HI-mode. Before we develop DA for MC-CF in the LO-mode, we define the following notion for the LO-mode.
Definition 2. Interference $I_{k+1}^{LO}(\ell)$ of $t_k$ on $t_k \in \tau$ is defined as the cumulative length of all intervals in an interval of length $\ell$ where $j_k$ is ready to execute, but it cannot execute on any processor while jobs of $t_k$ do in the LO-mode.

The DA for MC-CF reduces $I_{k+1}^{LO}(\ell)$ by exploiting the policy of MC-CF that it denotes the priority of higher-priority jobs each of which priority is guaranteed by comparing its remaining execution $C_i^{LO}$ and contention-free slots $\Phi_i^{LO}$ (Lines 9–11 in Algorithm 1), and thus lower-priority jobs can execute with relatively higher priority in contending slots. The following lemma presents an important property of MC-CF in the LO-mode.

Lemma 4. A job $J_i$ of $t_i \in \tau^{LO}$ (or $t_i \in \tau^{HI}$) can interfere with any job during at most $C_i^{LO} - \Phi_i^{LO}$ (or $C_i^{LO} - \Phi_i^{HI}$) time slots in the LO-mode.

Proof. When $J_i$ is released, it enters $Q_H$, and $\Phi_i^{LO}$ is set to $\Phi_i^{LO}$ (for $t_i \in \tau^{LO}$) or $\Phi_i^{HI}$ (for $t_i \in \tau^{HI}$). We consider the following two cases:

(Case 1: $J_i$ stays in $Q_H$ until $d_j$.) In this case, $\Phi_i^{LO}$ becomes zero before or at the time slot when $C_i^{HI}$ becomes one. Otherwise (e.g., $\Phi_i^{LO} = C_i^{LO} = 1$), $J_i$ moves to $Q_L$ according to Lines 9–11 in Algorithm 1. This implies that $J_i$ of $t_i \in \tau^{LO}$ (or $t_i \in \tau^{HI}$) encounters at least $\Phi_i^{LO}$ (or $\Phi_i^{HI}$) contention-free slots before $d_j$. By Definition 1, all jobs in contention-free slots can execute without any contention, and it also means that no job can interfere with the other jobs in the time slot. Thus, $J_i$ of $t_i \in \tau^{LO}$ (or $t_i \in \tau^{HI}$) can interfere with other jobs in at most $C_i^{LO} - \Phi_i^{LO}$ (or $C_i^{LO} - \Phi_i^{HI}$) time slots in the LO-mode.

(Case 2: $J_i$ moves to $Q_L$ at a time slot $t'$ (i.e., satisfying $\Phi_i^{(t')} = C_i^{(t')}))$. In this case, $J_i$ of $t_i \in \tau^{LO}$ (or $t_i \in \tau^{HI}$) encounters exactly $\Phi_i^{LO} - \Phi_i^{(t')}$ (or $\Phi_i^{HI} - \Phi_i^{(t')}$) contention-free slots before it moves to $Q_L$ at $t'$. After $t'$, the job cannot interfere with other jobs in $Q_H$ because any job in $Q_L$ has a lower priority than that of $Q_H$. Thus, $C_i^{LO} - C_i^{(t')} - (\Phi_i^{LO} - \Phi_i^{(t')}) = C_i^{LO} - \Phi_i^{LO}$ (or $C_i^{HI} - C_i^{(t')} - (\Phi_i^{HI} - \Phi_i^{(t')}) = C_i^{LO} - \Phi_i^{HI}$) executions of $J_i$ of $t_i \in \tau^{LO}$ (or $t_i \in \tau^{HI}$) can interfere with other jobs. □

Lemma 4 implies that the amount of execution for $J_i$ of $t_i \in \tau^{LO}$ (or $t_i \in \tau^{HI}$) to interfere with other jobs is limited to $C_i^{LO} - \Phi_i^{LO}$ (or $C_i^{LO} - \Phi_i^{HI}$) in the LO-mode when MC-CF is applied. Thus, instead of counting $C_i^{LO}$ to derive $W_i(\ell)$ in Equation (1), we use $\max(0, C_i^{LO} - \Phi_i^{LO})$ (or $\max(0, C_i^{LO} - \Phi_i^{HI})$). Let $W_i^{LO}(\ell, \Phi_i^{LO})$ (or $W_i^{LO}(\ell, \Phi_i^{HI})$) be the maximum workload of $t_i$ performed in contending slots in an interval of length $\ell$ when $t_i \in \tau^{LO}$ (or $t_i \in \tau^{HI}$) is scheduled by MC-CF in the LO-mode. Figure 2 presents the worst-case scenario where $W_i^{LO}(\ell, \Phi_i^{LO})$ (or $W_i^{LO}(\ell, \Phi_i^{HI})$) occurs.

![Figure 2](image-url)  

Figure 2. Scenario where $W_i^{LO}(\ell, \Phi_i^{LO})$ or $W_i^{LO}(\ell, \Phi_i^{HI})$ occurs.

Because at least $\Phi_i^{LO}$ (or $\Phi_i^{HI}$) amount of execution is performed in contention-free slots, a job $J_i$ of $t_i \in \tau^{LO}$ (or $t_i \in \tau^{HI}$) is performed at most $C_i^{LO} - \Phi_i^{LO}$ (or $C_i^{LO} - \Phi_i^{HI}$) in contending slots. Thus, when $W_i^{LO}(\ell, \Phi_i^{LO})$ (or $W_i^{LO}(\ell, \Phi_i^{HI})$) occurs, only $C_i^{LO} - \Phi_i^{LO}$ (or $C_i^{LO} - \Phi_i^{HI}$) amount of execution of the last job (e.g., the right-most job in Figure 2) is included in an interval of length $\ell$. Then, the previous jobs are as late as possible without a deadline miss; their first $C_i^{LO} - \Phi_i^{LO}$
(or $C_i^{HI} - \Phi_i^{HI}$) executions are performed in contending slots, and the others are done in contention-free slots. With the reasoning, $W_i^{LO}(\ell, \Phi_i)$ for $\Phi_i = (\Phi_i^{LO}$ or $\Phi_i^{HI})$ is calculated as follows:

$$W_i^{LO}(\ell, \Phi_i) = N_i^{LO}(\ell, \Phi_i) \cdot (C_i^{LO} - \Phi_i) + \min \left( (C_i^{LO} - \Phi_i), \ell + D_i - (C_i^{LO} - \Phi_i) - N_i^{LO}(\ell, \Phi_i) \cdot T_i \right),$$

where $N_i^{LO}(\ell, \Phi_i)$ is the number of jobs executing for $C_i^{LO} - \Phi_i$ given by

$$N_i^{LO}(\ell, \Phi_i) = \left\lfloor \frac{\ell + D_i - (C_i^{LO} - \Phi_i)}{T_i} \right\rfloor.$$

By the definition of $W_i^{LO}(\ell, \Phi_i)$ and $I_{k+1}^{LO}(\ell)$, $I_{k+1}^{LO}(D_k)$ is upper bounded by $W_i^{LO}(D_k, \Phi_i)$ where $\Phi_i = \Phi_i^{LO}$ for $\tau_i \in \tau^{LO}$ and $\Phi_i^{HI}$ for $\tau_i \in \tau^{HI}$. Then, the following provides DA for LO-mode when MC-CF is applied to FP scheduling.

**Theorem 1.** A task set is schedulable under MC-CF for FP scheduling in the LO-mode if the following holds for each task $\tau_k \in \tau$:

$$\sum_{\tau \in \text{E} \in \tau_k} \min(W_i^{LO}(D_k, \Phi_i), D_k - C_i^{LO} + 1) < m \cdot (D_k - C_i^{LO} + 1)$$

**Proof.** For $J_k$ to miss its deadline in the LO-mode, $J_k$ is executed at most $C_i^{LO} - 1$ time units in $[r_k, r_k]$, and, at each time slot, interference from at least $m$ other jobs is needed to block $\tau_k$’s execution. Thus, the job cannot miss its deadline if the total interference of other jobs on a job of $\tau_k$ is less than $m \cdot (D_k - C_i^{LO} - 1)$. □

We then derive schedulability analysis for EDF scheduling. Let $E_i^{LO}(\ell, \Phi_i^{LO})$ (or $E_i^{LO}(\ell, \Phi_i^{HI})$) be the maximum execution higher priority jobs performed in contending slots in an interval of length $\ell = D_k$ when $\tau_i \in \tau^{LO}$ (or $\tau_i \in \tau^{HI}$) is scheduled by MC-CF with EDF scheduling in the LO-mode. Figure 3 presents the worst-case scenario where $E_i^{LO}(\ell, \Phi_i^{LO})$ (or $E_i^{LO}(\ell, \Phi_i^{HI})$) occurs.

![Figure 3. Scenario where $E_i^{LO}(\ell, \Phi_i^{LO})$ or $E_i^{LO}(\ell, \Phi_i^{HI})$ occurs.](image)

When $E_i^{LO}(\ell, \Phi_i^{LO})$ (or $E_i^{LO}(\ell, \Phi_i^{HI})$) occurs, only $C_i^{LO} - \Phi_i^{LO}$ (or $C_i^{LO} - \Phi_i^{HI}$) amount of execution of the last job is included in an interval of length $\ell$. Then, the previous jobs are as late as possible without a deadline miss; their first $C_i^{LO} - \Phi_i^{LO}$ (or $C_i^{HI} - \Phi_i^{HI}$) executions are performed in contending slots, and the others are done in contention-free slots. With the reasoning, $E_i^{LO}(\ell, \Phi_i)$ for $\Phi_i = (\Phi_i^{LO}$ or $\Phi_i^{HI})$ is calculated as follows:

$$E_i^{LO}(\ell, \Phi_i) = EN_i^{LO}(\ell, \Phi_i) \cdot (C_i^{LO} - \Phi_i) + \min \left( (C_i^{LO} - \Phi_i), \ell - EN_i^{LO}(\ell, \Phi_i) \cdot T_i \right),$$

where $EN_i^{LO}(\ell, \Phi_i)$ is the number of jobs executing for $C_i^{LO} - \Phi_i$ given by
Theorem 2. A task set is schedulable under MC-CF with EDF scheduling in the LO-mode if the following holds for each task $\tau_k \in \tau$:

$$\sum_{\tau_i \in \tau \setminus \tau_k} \min(E^{LO}_i(D_k, \Phi_i), D_k - C^{LO}_i + 1) < m \cdot (D_k - C^{LO}_k + 1)$$

(14)

Proof. The proof is the same as Theorem 1. □

5.2. Schedulability Analysis for HI-Mode

According to the policy of MC-CF, tasks in $\tau^{LO}$ are dropped at the mode change, and thus we consider the schedulability of tasks in $\tau^{HI}$ in the HI-mode. Before we develop DA for MC-CF in the HI-mode, we define the following notion for HI-mode as follows.

Definition 3. Interference $I_{k+1}^{HI}(\ell, \ell')$ of $\tau_k$ on $\tau_{k+1} \in \tau^{HI}$ is defined as the cumulative length of all intervals in $[r_k, d_k)$ where $j_k$ is ready to execute, but it cannot execute on any processor while jobs of $\tau_k$ do, which assumes that a mode transition occurs at $r_k + \ell'$ for $\ell' < D_k$.

Similar the LO-mode case, DA for MC-CF reduces $I_{k+1}^{HI}(\ell, \ell')$ by exploiting the priority-demotion policy of MC-CF in the HI-mode (Lines 9–11 in Algorithm 1). The following lemma presents an important property of MC-CF in the HI-mode.

Lemma 5. A Job $J_i$ of $\tau_i \in \tau^{HI}$ can interfere with any job during at most $C_i^{HI} - \Phi_i^{HI}$ time slots in the LO-mode.

Proof. The proof is derived straightforwardly from Lemma 4 by substituting $C_i^{LO}$ and $\Phi_i^{LO}$ in Lemma 4 to $C_i^{HI}$ and $\Phi_i^{HI}$, respectively. □

Let $W_i^{HI}(\ell, \ell', \Phi_i^{HI})$ denote the maximum workload of $\tau_i \in \tau^{HI}$ performed in contending slots in an interval of length $\ell$ when $\tau_i$ is scheduled by MC-CF with FP scheduling, which assumes that a mode transition occurs when $\ell'$ the amount of time elapses from the beginning of the interval of length $\ell$. Figure 4 presents the worst-case scenario where $W_i^{HI}(\ell, \ell', \Phi_i^{HI})$ occurs. As shown in Figure 4, only $C_i^{HI} - \Phi_i^{HI}$ amount of execution of the last job after a mode transition (e.g., the right-most job in Figure 2) is included in an interval of length $\ell$. Then, the previous jobs are executed as late as possible without a deadline miss; the first job before the mode transition executes for $C_i^{LO} - \Phi_i^{HI}$, and the second job experiencing the mode transition executes for $C_i^{HI} - \Phi_i^{HI}$ in contending slots. With the reasoning, $W_i^{HI}(\ell, \ell', \Phi_i^{HI})$ is calculated as follows:

$$W_i^{HI}(\ell, \ell', \Phi_i^{HI}) = N_i^{HI}(\ell - \ell', \Phi_i^{HI}) \cdot (C_i^{HI} - \Phi_i^{HI}) + W_i^{LO}(\chi^{\Phi}, \Phi_i^{HI}) + \min(\chi^{\Phi} - \ell', C_i^{HI} - C_i^{LO})$$

(15)

where $N_i^{HI}(\ell - \ell', \Phi_i^{HI})$ is the number of jobs executing for $C_i^{HI} - \Phi_i^{HI}$ given by

$$N_i^{HI}(\ell - \ell', \Phi_i^{HI}) = \left\lceil \frac{\ell - \ell' + D_i - (C_i^{HI} - \Phi_i^{HI})}{T_i} \right\rceil,$$

(16)
and \( \chi^\Phi \) is given by
\[
\chi^\Phi = \ell + D_i - (C^H_i - \Phi^H_i) - N^H_i (\ell - \ell') \cdot T_i. \tag{17}
\]

By the definition of \( W^H_i(\ell, \ell', \Phi^H_i) \) and \( I^H_{k, i}(\ell, \ell'), \) \( I^H_{k, i}(D_k, \ell') \) is upper bounded by \( W^H_i(D_k, \ell', \Phi^H_i) \). Then, the following provides DA for HI-mode when MC-CF is applied.

![Figure 4. Scenario where \( W^H_{k+1}(\ell, \ell', \Phi^H_{k+1}) \) for \( \tau_i \in \tau^H_i \) occurs.](image)

**Lemma 6.** A task set is schedulable under MC-CF in the HI-mode if the following holds with given \( \ell' \) for each task \( \tau_k \in \tau^H_i \):
\[
\sum_{\tau_i \in (hep(\tau_j) \cap \tau^{HI})} \min (I^H_{k-1}(\ell'), D_k - C^H_k + 1) + \\
\sum_{\tau_i \in (hep(\tau_j) \cap \tau^{HI})} \min (I^H_{k-1}(D_k, \ell'), D_k - C^H_k + 1) < m \cdot (D_k - C^H_k + 1) \tag{18}
\]

**Proof.** For \( j_k \) to miss its deadline in the HI-mode, \( j_k \) is executed at most \( C^H_k - 1 \) time units in \([r_k, r_d)\), and, at each time slot, interference from at least \( m \) other jobs is needed to block \( \tau_k \)'s execution. Thus, the job cannot miss its deadline if the total interference of other jobs on a job of \( \tau_k \) is less than \( m \cdot (D_k - (C^H_k - 1)) \).

Since MC-CF cannot predict the timing of the mode transition, we need to consider all \( \ell' \) for \( 0 \leq \ell' \leq D_k \) to guarantee the schedulability of \( \tau_k \in \tau^H_i \) in the HI-mode. Thus, the following theorem completes DA for MC-CF for HI-mode.

**Theorem 3.** A task set is schedulable under MC-CF with FP scheduling in the HI-mode Equation (18) holds for every \( 0 \leq \ell' \leq D_k \) and each task \( \tau_k \in \tau^H_i \)

We then derive schedulability analysis for EDF scheduling. Let \( E^H_i(\ell, \ell', \Phi^H_i) \) denote the maximum execution of higher-priority jobs performed in contending slots in an interval of length \( \ell = D_k \) when \( \tau_i \) is scheduled by MC-CF with EDF scheduling, which assumes that a mode transition occurs when \( \ell' \) the amount of time elapses from the beginning of the interval of length \( \ell \). Figure 5 presents the worst-case scenario where \( E^H_i(\ell, \ell', \Phi^H_i) \) occurs. The difference with Figure 4 is that the last job’s deadline cannot be later than the end of \( \ell \). With the reasoning, \( E^H_i(\ell, \ell', \Phi^H_i) \) is calculated as follows:
Figure 5. Scenario where $E_{k-1}^{HI}(\ell, \ell', \Phi_{i}^{HI})$ for $\tau_i \in \tau^{HI}$ occurs.

\[
E_{k-1}^{HI}(\ell, \ell', \Phi_{i}^{HI}) = E_{i}^{HI}(\ell - \ell', \Phi_{i}^{HI}) \cdot (C_{i}^{HI} - \Phi_{i}^{HI}) + E_{i}^{LO}(\chi^{\Phi} - \ell', C_{i}^{HI} - C_{i}^{LO}) + \min(\chi^{\Phi} - \ell', D_{k} - C_{k}^{HI} + 1).
\]

where $E_{i}^{HI}(\ell - \ell', \Phi_{i}^{HI})$ is the number of jobs executing for $C_{i}^{HI} - \Phi_{i}^{HI}$ given by

\[
E_{i}^{HI}(\ell - \ell', \Phi_{i}^{HI}) = \left\lceil \frac{\ell - \ell'}{T_{i}} \right\rceil,
\]

and $\chi^{\Phi}$ is given by

\[
\chi^{\Phi} = \ell - N_{i}^{HI}(\ell - \ell') \cdot T_{i}.
\]

By the definition of $E_{i}^{HI}(\ell, \ell', \Phi_{i}^{HI})$ and $I_{k-1}^{HI}(\ell, \ell')$, $I_{k-1}^{HI}(D_{k}, \ell')$ is upper bounded by $E_{k-1}^{HI}(D_{k}, \ell', \Phi_{i}^{HI})$ when EDF is applied. Then, the following provides DA for HI-mode when MC-CF with EDF scheduling is applied.

**Lemma 7.** A task set is schedulable under MC-CF with EDF scheduling in the HI-mode if the following holds with given $\ell'$ for each task $\tau_k \in \tau^{HI}$:

\[
\sum_{\tau_i \in (\tau \setminus \tau_k) \cap \tau^{LO}} \min(E_{i}^{LO}(\ell'), D_{k} - C_{k}^{HI} + 1) + \sum_{\tau_i \in (\tau \setminus \tau_k) \cap \tau^{HI}} \min(E_{i}^{HI}(D_{k}, \ell'), D_{k} - C_{k}^{HI} + 1) < m \cdot (D_{k} - C_{k}^{HI} + 1)
\]

**Proof.** The proof is the same as Lemma 6.  \(\square\)

The following theorem completes DA for MC-CF with EDF scheduling for HI-mode.

**Theorem 4.** A task set is schedulable under MC-CF with EDF scheduling in the HI-mode Equation (22) holds for every $0 \leq \ell' \leq D_{k}$ and each task $\tau_k \in \tau^{HI}$

6. Evaluation

In this section, we evaluate the performance of the proposed MC-CF under various environmental settings. We modified the simulation program used in [13] to accommodate the features of mixed-criticality. The simulation code is implemented by JAVA and runs on a desktop machine (e.g., equipped with Intel i5-9400F CPU and 32 GB RAM). It provides a number of functions regarding task set generation, scheduling simulation, and schedulability analysis. We consider RM scheduling EDF scheduling as base scheduling algorithms in which the proposed MC-CF is applied. We randomly generates task sets with various input factors; then, we measure the number of task sets deemed...
The workload derived in this way is the upper-bounded workload rather than the exact one, and thus the pessimism becomes prominent as \( n \) increases. As seen in Figure 8a, \( n = 6 \) and \( n = 20 \) yield the highest and lowest performance, respectively, for \( m = 4, CP = 0.1 \), and \( CF = 2 \). Compared to the combination of \( m = 4, CP = 0.1 \), and \( CF = 2 \) (Figure 8a), that of \( m = 4, CP = 0.9 \), and \( CF = 4 \) (Figure 8b) deliver lower performance because the task sets in the setting contain a higher number of \( HI \)-criticality tasks.
Figure 6. Experimental comparison of MC-RM and MC-CF-RM.
Figure 7. Experimental comparison of MC-EDF and MC-CF-EDF.

Figure 8. Experimental results of MC-CF-RM in a varying number of task sets.
7. Conclusions

We propose MC-CF for MC multiprocessor systems and MC-CF tailored schedulability analysis to support it. We defined the notion of contention-free slots for two different criticalities (of MC systems) of tasks, proposed a technique to specify an upper bound for the amount to be utilized for each task, and finally explained the systematic utilization of contention-free slots to improve the schedulability of MC multiprocessor systems. Next, we developed DA for MC-CF. Using the experimental results with various environmental settings, we demonstrated that MC-CF can significantly improve the schedulability of fixed-priority scheduling. As a limitation of the proposed MC-CF and DA, it derives the lower bound of contention-free slots and judges the schedulability of a HI-criticality task in a rather time-consuming way. In the future, we plan to develop a less time-consuming method to overcome the aforementioned limitation. In addition, considering heterogeneous platforms appears to be promising for future work.

Author Contributions: Conceptualization, H.B. and K.L.; Software, H.B.; data curation, K.L.; writing—original draft preparation, H.B. and K.L.; writing—review and editing, K.L.; Supervision, H.B.; project administration, H.B.; funding acquisition, H.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Incheon National University (International Cooperative) Research Grant in 2019.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Baruah, S.; Mok, A.; Rosier, L. Preemptively scheduling hard-real-time sporadic tasks on one processor. In Proceedings of the IEEE Real-Time Systems Symposium (RTSS), Lake Buena Vista, FL, USA, 5–7 December 1990; pp. 182–190.

2. Anderson, J.H.; Srinivasan, A. Early-release fair scheduling. In Proceedings of the 12th Euromicro Conference on Real-Time Systems, Euromicro, RTSECRTS, Stockholm, Sweden, 19–21 June 2000; pp. 35–43.

3. Cho, H.; Ravindran, B.; Jensen, E.D. An Optimal Real-Time Scheduling Algorithm for Multiprocessors. In Proceedings of the 27th IEEE International Real-Time Systems Symposium (RTSS’06), Rio de Janeiro, Brazil, 5–8 December 2006; pp. 101–110.

4. Regnier, P.; Lima, G.; Massa, E.; Levin, G.; Brandt, S. RUN: Optimal Multiprocessor Real-Time Scheduling via Reduction to Uniprocessor. In Proceedings of the IEEE Real-Time Systems Symposium (RTSS), Vienna, Austria, 29 November–2 December 2011; pp. 104–115.

5. Andersson, B.; Tovar, E. Multiprocessor Scheduling with Few Preemptions. In Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA’06), Sydney, Australia, 16–18 August 2006; pp. 322–334.

6. Levin, G.; Funk, S.; Sadowski, C.; Pye, I.; Brandt, S. DP-FAIR: A Simple Model for Understanding Optimal Multiprocessor Scheduling. In Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS), Brussels, Belgium, 6–9 July 2010; pp. 3–13.

7. Ernst, R.; Natalie, M.D. Mixed Criticality Systems—A History of Misconceptions? IEEE Des. Test 2016, 33, 65–74. [CrossRef]

8. Anderson, J.H.; Baruah, S.; Brandenburg, B.B. Multicore operating-system support for mixed criticality. In Proceedings of the Workshop on Mixed-Criticality: Roadmap to Evolving UAV Certification, San Francisco, CA, USA, 16 April 2009.

9. Pathan, R. Schedulability Analysis of Mixed-Criticality Systems on Multiprocessors. In Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS), Pisa, Italy, 11–13 July 2012; pp. 309–320.

10. Baek, H.; Jung, N.; Chwa, H.S.; Shin, I.; Lee, J. Non-Preemptive Scheduling for Mixed-Criticality Real-Time Multiprocessor Systems. IEEE Trans. Parallel Distrib. Syst. 2018, 29, 1766–1779. [CrossRef]

11. Baker, T.P.; Cirinei, M.; Bertogna, M. EDZL Scheduling Analysis. Real Time Syst. 2008, 40, 264–289. [CrossRef]

12. Lee, J.; Easwaran, A.; Shin, I. Maximizing Contention-Free Executions in Multiprocessor Scheduling. In Proceedings of the IEEE Real-Time Technology and Applications Symposium (RTAS), Chicago, IL, USA, 11–14 April 2011; pp. 235–244.
13. Lee, J.; Easwaran, A.; Shin, I. Contention-Free Executions for Real-Time Multiprocessor Scheduling. *ACM Trans. Embed. Comput. Syst.* 2014, 13, 1–69. [CrossRef]

14. Jung, N.; Baek, H.; Lim, D.; Lee, J. Incorporating Zero-Laxity Policy into Mixed-Criticality Multiprocessor Real-Time Systems. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* 2018, 101, 1888–1899. [CrossRef]

15. Burns, A.; Davis, R.I. *Mixed Criticality Systems—A Review*; University of York: York, UK, 2015.

16. Bertogna, M.; Cirinei, M. Response-Time Analysis for Globally Scheduled Symmetric Multiprocessor Platforms. In Proceedings of the 28th IEEE International Real-Time Systems Symposium (RTSS 2007), Tucson, AZ, USA, 3–6 December 2008; pp. 149–160.

17. Baek, H.; Lee, J.; Shin, I. Multi-Level Contention-Free Policy for Real-Time Multiprocessor Scheduling. *J. Syst. Softw.* 2018, 137, 36–49. [CrossRef]

18. Baek, H.; Lee, J. Improved Schedulability Analysis of the Contention-Free Policy for Real-Time Systems. *J. Syst. Softw.* 2019, 154, 112–124. [CrossRef]

19. Vestal, S. Real-Time Sampled Signal Flows through Asynchronous Distributed Systems. In Proceedings of the IEEE Real-Time Technology and Applications Symposium (RTAS), San Francisco, CA, USA, 7–10 March 2005; pp. 170–179.

20. Baruah, S.; Bonifaci, V.; D’Angelo, G.; MarchettiSpaccamela, A.; van der Ster, S.; Stougie, L. Mixed-criticality scheduling of sporadic task systems. In Proceedings of the 19th Annual European Symposium on Algorithms, Saarbrücken, Germany, 5–9 September 2011; pp. 555–566.

21. Baruah, S.; Bonifaci, V.; D’Angelo, G.; Marchetti-Spaccamela, H.L.A.; van der Ster, S.; Stougie, L. The Preemptive Uniprocessor Scheduling of Mixed-Criticality Implicit-Deadline Sporadic Task Systems. In Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS), Pisa, Italy, 11–13 July 2012; pp. 145–154.

22. Baruah, S.; Burns, A.; Davis, R.I. Response-Time Analysis for Mixed Criticality Systems. In Proceedings of the IEEE Real-Time Systems Symposium (RTSS), Vienna, Austria, 29 November–2 December 2011; pp. 34–43.

23. Baruah, S.; Li, H.; Stougie, L. Toward the design of certifiable mixed-criticality systems. In Proceedings of the IEEE Real-Time Technology and Applications Symposium (RTAS), Stockholm, Sweden, 12–15 April 2010; pp. 13–22.

24. Li, H.; Baruah, S. An algorithm for scheduling certifiable mixed-criticality sporadic task systems. In Proceedings of the IEEE Real-Time Systems Symposium (RTSS), San Diego, CA, USA, 30 November–3 December 2010; pp. 183–192.

25. Baruah, S. Schedulability Analysis for a General Model of Mixed-Criticality Recurrent Real-Time Tasks. In Proceedings of the IEEE Real-Time Systems Symposium (RTSS), Porto, Portugal, 29 November–2 December 2016.

26. Li, H.; Baruah, S. Global mixed-criticality scheduling on multiprocessors. In Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS), Pisa, Italy, 10–13 July 2012; pp. 166–175.

27. Lee, J.; Phan, K.; Gu, X.; Lee, J.; Easwaran, A.; Shin, I.; Lee, I. MC-Fluid: Fluid Model-Based Mixed-Criticality Scheduling on Multiprocessors. In Proceedings of the IEEE Real-Time Systems Symposium (RTSS), Rome, Italy, 2–5 December 2014; pp. 41–52.

28. Davis, R.; Burns, A. Priority Assignment for Global Fixed Priority Pre-emptive Scheduling in Multiprocessor Real-Time Systems. In Proceedings of the IEEE Real-Time Systems Symposium (RTSS), Washington, DC, USA, 1–4 December 2009; pp. 398–409.

29. Bini, E.; Buttazzo, G. Measuring the performance of schedulability tests. *Real Time Syst.* 2005, 30, 129–154. [CrossRef]

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).