PAPER

Incorporating Zero-Laxity Policy into Mixed-Criticality Multiprocessor Real-Time Systems

Namyong JUNG[†], ***Nonmember***, **Hyeongboo BAEK**[†], ***Member***, **Donghyouk LIM**[††], ***and*** **Jinkyu LEE**[†a)], ***Nonmembers***

SUMMARY As real-time embedded systems are required to accommodate various tasks with different levels of criticality, scheduling algorithms for MC (Mixed-Criticality) systems have been widely studied in the real-time systems community. Most studies have focused on MC uniprocessor systems whereas there have been only a few studies to support MC multiprocessor systems. In particular, although the ZL (Zero-Laxity) policy has been known to an effective technique in improving the schedulability performance of base scheduling algorithms on SC (Single-Criticality) multiprocessor systems, the effectiveness of the ZL policy on MC multiprocessor systems has not been revealed to date. In this paper, we focus on realizing the potential of the ZL policy for MC multiprocessor systems, which is the first attempt. To this end, we design the ZL policy for MC multiprocessor systems, and apply the policy to EDF (Earliest Deadline First), yielding EDZL (Earliest Deadline first until Zero-Laxity) tailored for MC multiprocessor systems. Then, we develop a schedulability analysis for EDZL (as well as its base algorithm EDF) to support its timing guarantee. Our simulation results show a significant schedulability improvement of EDZL over EDF, demonstrating the effectiveness of the ZL policy for MC multiprocessor systems.

key words: *mixed-criticality, multiprocessor real-time systems, zero-laxity policy schedulability analysis, EDZL (Earliest Deadline first until Zero-Laxity), EDF (Earliest Deadline First)*

1. Introduction

Integrated systems have received considerable attention for their ability to reduce size, weight and power for real-time systems. In other words, instead of implementing each function in each distributed system, a single, integrated hardware controls all functions. Typical examples are ARINC 653 [1] for avionics and Autosar [2] used in the automotive industry. To support different criticality levels with high CPU utilization for such integrated systems, timing guarantees for MC (Mixed-Criticality) multiprocessor real-time systems have been studied in the real-time systems community [3]–[5]. However, the underlying theory has yet to mature, unlike in SC (Single-Criticality) multiprocessor systems.

For example, the ZL (Zero-Laxity) policy [6], [7] have received considerable attention due to its wide applicability and significant schedulability improvement, which can be incorporated into most (if not all) existing real-time scheduling algorithms (called base algorithms) on SC multiprocessor systems. The ZL policy assigns the highest priority to zero-laxity jobs and prioritizes other jobs according to the base algorithm. Here, a job's laxity at any time instant is defined as remaining time to the job's absolute deadline minus the amount of remaining execution time at that instant. By executing jobs that would otherwise miss their absolute deadlines (i.e., zero-laxity jobs), the ZL policy considerably improves the base algorithm in terms of schedulability in SC multiprocessor systems. However, such effectiveness of the ZL policy in enhancing schedulability has not been achieved in MC multiprocessor systems.

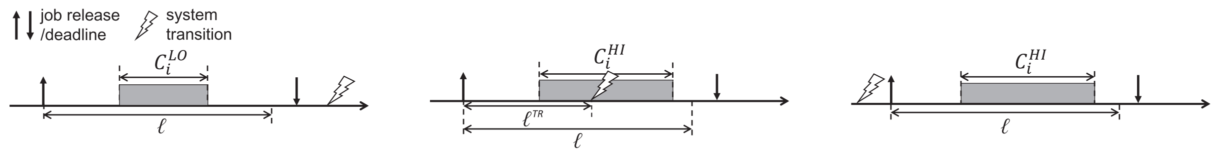
In this paper, we focus on demonstrating that the ZL policy is also effective in improving the schedulability of the base algorithm in MC multiprocessor systems, which is the first attempt. To this end, we first consider EDF (Earliest Deadline First) as the base algorithm and develop RTA (Response-Time Analysis) for EDF to support a timing guarantee in MC multiprocessor systems. Although RTA is one of the most popular schedulability analysis frameworks because of its higher schedulability performance, no RTA for EDF has been established in MC multiprocessor systems. Second, we design EDZL (Earliest Deadline first until Zero-Laxity) scheduling algorithm for MC multiprocessor systems by incorporating the ZL policy into EDF. Unlike for EDZL in SC multiprocessor systems, we must re-define the concept of a job's laxity because MC multiprocessor systems entail multiple types of execution times based on certification authorities. Finally, we develop RTA for EDZL in MC multiprocessor systems, which considers the new concept of a job's laxity. Our simulation results demonstrate that EDZL considerably improves the schedulability of EDF in MC multiprocessor systems.

We emphasize that we consider EDF as the base algorithm owing to its simplicity and popularity, and the applicability of the ZL policy for MC multiprocessor systems is not limited to EDF. Thus, the ZL policy can be used for most (if not all) existing scheduling algorithms such as EDF-VD (Earliest Deadline First with Virtual Deadlines) [3] and FP (Fixed Priority) [4]. This indicates the significance of our work as the first study that demonstrates the potential performance improvement achieved by the ZL policy when it is incorporated into the various base algorithms in MC multiprocessor systems.

In summary, this paper provides the following contributions to MC multiprocessor systems.

Manuscript received April 24, 2018.
Manuscript revised July 23, 2018.
[†]The authors are with Department of Computer Science and Engineering, Sungkyunkwan University (SKKU), Republic of Korea.
[††]The author is with RTST, Daejeon, Republic of Korea.
a) E-mail: Jinkyu.lee@skku.edu (Corresponding author)
DOI: 10.1587/transfun.E101.A.1888

Copyright © 2018 The Institute of Electronics, Information and Communication Engineers



(a) The system transition occurs after the interval of interest (b) The system transition occurs in the middle of the interval of interest (c) The system transition occurs before the interval of interest

Fig. 1 Three cases of the system transition occurrence.

- Design of the ZL policy and incoporate the policy to EDF, yielding EDZL tailored to MC multiprocessor systems,
- Development of RTA for EDF,
- Development of RTA for EDZL, and
- Demonstration of the potential of the ZL policy in improving schedulability performance via simulation.

The remainder of this paper is organized as follows. Section 2 presents our system model. Section 3 reviews our development of RTA for EDF in MC multiprocessor systems. Section 4 discusses our design of EDZL scheduling algorithm, which includes the concept of a job's laxity in MC multiprocessor systems, and our development of RTA for EDZL in MC multiprocessor systems. Section 5 evaluates the schedulability performance of the proposed RTAs for EDF and EDZL, and Sect. 6 discusses related work. Section 7 concludes this paper.

2. System Model

We consider a set of sporadic real-time tasks [8] having two-criticality levels [9]. A task $\tau_i \in \tau$ is characterized by five parameters $(T_i, C_i^{\mathsf{LO}}, C_i^{\mathsf{HI}}, D_i, L_i)$. T_i is the minimal interval between release times of consecutive jobs of τ_i. C_i^{LO} and C_i^{HI} are the worst-case execution times with low criticality (LO) and high criticality (HI), respectively. D_i is the relative deadline of τ_i. $L_i \in \{\mathsf{LO}, \mathsf{HI}\}$ is a criticality of τ_i. For τ_i satisfying $L_i = \mathsf{LO}$, $C_i^{\mathsf{LO}} = C_i^{\mathsf{HI}} \leq D_i$ holds. For τ_i satisfying $L_i = \mathsf{HI}$, $C_i^{\mathsf{LO}} \leq C_i^{\mathsf{HI}} \leq D_i$ holds. In the beginning, every job invoked by $\tau_i \in \tau$ performs its execution up to C_i^{LO}. If a time instant is observed at which the amount of execution of a job of τ_i is about to exceed C_i^{LO}, we say that a system transition occurs at this time instant, denoted as t^{TR}. After the system transition, we care only about tasks $\{\tau_i\}$ with $L_i = \mathsf{HI}$, assuming their execution time can be up to C_i^{HI}, and do not care about tasks $\{\tau_i\}$ with $L_i = \mathsf{LO}$. We say that the system exhibits LO- and HI-criticality behavior before and after the system transition, respectively.

We let R_k^{LO} and R_k^{HI} denote the response time of τ_k before and after the system transition, respectively. That is, before the system transition (*likewise* after the system transition), every job invoked by τ_k finishes its execution within R_k^{LO} time units (*likewise* R_k^{HI} time units) from its release. We then let S_k^{LO} and S_k^{HI} denote a slack of τ_k before and after the system transition, which is calculated by $S_k^{\mathsf{LO}} = D_k - R_k^{\mathsf{LO}}$ and $S_k^{\mathsf{HI}} = D_k - R_k^{\mathsf{HI}}$, respectively. In other words, these refer to every job of τ_k before the system transition finishes its execution at least S_k^{LO} ahead of its absolute deadline, and after the system transition finishes its execution at least S_k^{HI} ahead of its absolute deadline.

In this paper, we consider work-conserving, preemptive, and global scheduling algorithms. In other words, a ready job should be executed as long as at least one idle processor exists (*work-conserving*); a higher-priority job can preempt the execution of a lower-priority job at any time (*preemptive*); and a job is allowed to execute in any processor with migration (*global*). We assume that m identical processors are present in the system.

3. RTA for EDF in MC Multiprocessor Systems

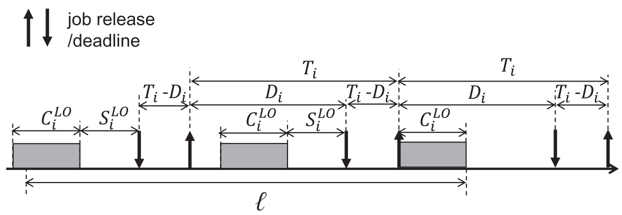
In this section, we discuss our development of RTA for EDF in MC multiprocessor systems. We first apply the existing RTA for EDF designed for SC multiprocessor systems, to that for MC multiprocessor systems under LO-criticality behavior. We then develop RTA for EDF in MC multiprocessor systems under HI-criticality behavior by deriving an upper-bounded interference for it.

3.1 RTA for EDF under LO-Criticality Behavior

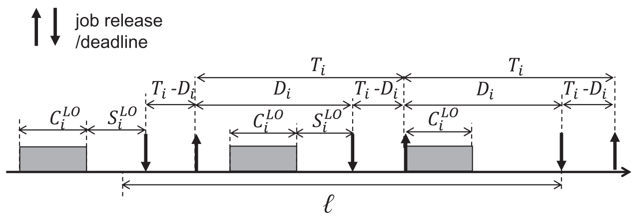
Before the system transition, underlying scheduling in MC multiprocessor systems is the same as that in SC multiprocessor systems. Therefore, we can apply existing RTA for EDF designed for SC multiprocessor systems [10] to MC multiprocessor systems. To calculate the response time of a job of τ_k, we must calculate the interference from jobs of τ_i to that of τ_k. Let $I_{k\leftarrow i}^{\mathsf{LO}}(\ell)$ denote the length of the cumulative intervals such that jobs of τ_i execute but the job of τ_k of interest cannot execute within an interval of length ℓ starting at the job's release time, when the system transition does not occur before the end of the interval of length ℓ, i.e., Fig. 1(a). Then, the job of τ_k of interest can finish executing (just as C_k^{LO}) within ℓ time units after its release time, if the sum of $I_{k\leftarrow i}^{\mathsf{LO}}(\ell)$ for every $\tau_i \in \tau \setminus \{\tau_k\}$ divided by m is not greater than $\ell - C_k^{\mathsf{LO}}$. In other words, the response time of a job of τ_k is no greater than ℓ if the following inequality holds [10]:

$$\ell \geq C_k^{\mathsf{LO}} + \left\lfloor \frac{1}{m} \cdot \sum_{\tau_i \in \tau \setminus \{\tau_k\}} \min\left(I_{k\leftarrow i}^{\mathsf{LO}}(\ell), \ell - C_k^{\mathsf{LO}} + 1\right) \right\rfloor. \quad (1)$$

Although $I_{k\leftarrow i}^{\mathsf{LO}}(\ell)$ depends on the target scheduling algorithm, the existing RTA for EDF calculates two upper-bounds for $I_{k\leftarrow i}^{\mathsf{LO}}(\ell)$ under EDF. First, with any scheduling



(a) $W(\ell, T_i, C_i^{\mathsf{LO}}, D_i, S_i^{\mathsf{LO}})$ in (2)



(b) $E(\ell, T_i, C_i^{\mathsf{LO}}, S_i^{\mathsf{LO}})$ in (3)

Fig. 2 Functions needed to upper-bound the interference.

algorithm (assuming no job deadline is missed), $I_{k\leftarrow i}^{\mathsf{LO}}(\ell)$ is upper-bounded by the maximum amount of execution of jobs of τ_i under LO-criticality behavior in an interval of length ℓ, as denoted by $W_i^{\mathsf{LO}}(\ell, S_i^{\mathsf{LO}}) \overset{\text{def.}}{=} W(\ell, T_i, C_i^{\mathsf{LO}}, D_i, S_i^{\mathsf{LO}})$ [10], where

$$W(\ell, T_i, C_i^{\mathsf{LO}}, D_i, S_i^{\mathsf{LO}}) = \left\lfloor \frac{\ell + D_i - C_i^{\mathsf{LO}} - S_i^{\mathsf{LO}}}{T_i} \right\rfloor \cdot C_i^{\mathsf{LO}} + min\left(C_i^{\mathsf{LO}}, \ell + D_i - C_i^{\mathsf{LO}} - S_i^{\mathsf{LO}} - \left\lfloor \frac{\ell + D_i - C_i^{\mathsf{LO}} - S_i^{\mathsf{LO}}}{T_i} \right\rfloor \cdot T_i\right). \tag{2}$$

The function $W(\ell, T_i, C_i^{\mathsf{LO}}, D_i, S_i^{\mathsf{LO}})$ shown in Fig. 2(a) calculates the maximum amount of execution of jobs of a task whose period, worst-case execution time, relative deadline, and slack are T_i, C_i^{LO}, D_i, and S_i^{LO}, respectively, within an interval of length ℓ. In Fig. 2(a), the last (i.e., right-most) job of the task in the interval of interest of length ℓ is executed as early as possible, whereas the other jobs finishes its execution S_i^{LO} time units ahead of its absolute deadline without any interference or delay. Then, $\lfloor \frac{\ell + D_i - C_i^{\mathsf{LO}} - S_i^{\mathsf{LO}}}{T} \rfloor$ is used to calculate the number of jobs whose executions are fully performed within the interval (the first job within the interval may not be counted in this number). For example, $\lfloor \frac{\ell + D_i - C_i^{\mathsf{LO}} - S_i^{\mathsf{LO}}}{T_i} \rfloor = 2$ in Fig. 2(a) refers to the second and third jobs. Next, the amount of execution of the first job (if the job is not counted in $\lfloor \frac{\ell + D_i - C_i^{\mathsf{LO}} - S_i^{\mathsf{LO}}}{T_i} \rfloor$) within the interval is calculated using the second term of Eq. (2). Therefore, in any scheduling algorithm, jobs of a task cannot execute more than $W(\ell, T_i, C_i^{\mathsf{LO}}, D_i, S_i^{\mathsf{LO}})$ within an interval of length ℓ.

Second, if we consider the prioritization policy of EDF, a job with a later absolute deadline cannot interfere with another job with an earlier absolute deadline. Therefore, under EDF, $I_{k\leftarrow i}(\ell)$ is upper-bounded by $E_{k\leftarrow i}^{\mathsf{LO}}(S_i^{\mathsf{LO}}) \overset{\text{def.}}{=} E(D_k, T_i, C_i^{\mathsf{LO}}, S_i^{\mathsf{LO}})$ (Theorem 5 in [10]), where

$$E(\ell, T_i, C_i, S_i^{\mathsf{LO}}) = \left\lfloor \frac{\ell}{T_i} \right\rfloor \cdot C_i^{\mathsf{LO}} + \max\left(0, \min\left(C_i^{\mathsf{LO}}, \ell - \left\lfloor \frac{\ell}{T_i} \right\rfloor \cdot T_i - S_i^{\mathsf{LO}}\right)\right). \tag{3}$$

$E(\ell, T_i, C_i^{\mathsf{LO}}, S_i^{\mathsf{LO}})$ shown in Fig. 2(b) computes the maximum amount of execution of jobs of a task whose period, worst-case execution time, and slack are T_i, C_i, and S_i^{LO}, respectively, within an interval of length ℓ such that each job's absolute deadline is no later than the end of the interval. Then, $\lfloor \frac{\ell}{T_i} \rfloor$ is used to count the number of jobs whose executions are fully performed within the interval (the first job in the interval may not be counted in this number). For example, $\lfloor \frac{\ell}{T_i} \rfloor = 2$ in Fig. 2(b) refers to the second and third jobs. Next, the amount of execution of the first job (if the job is not counted in $\lfloor \frac{\ell}{T_i} \rfloor$) can be calculated using the second term of Eq. (3).

Combining the two upper-bounds and applying that any interference in an interval of length ℓ cannot be larger than the length ℓ (i.e., $I_{k\leftarrow i}^{\mathsf{LO}}(\ell) \le \ell$), $I_{k\leftarrow i}^{\mathsf{LO}}(\ell)$ under EDF under LO-criticality behavior is upper-bounded by

$$I_{k\leftarrow i}^{\mathsf{LO}}(\ell) \le \min\left(\ell, W_i^{\mathsf{LO}}(\ell, S_i^{\mathsf{LO}}), E_{k\leftarrow i}^{\mathsf{LO}}(S_i^{\mathsf{LO}})\right). \tag{4}$$

Applying the upper-bound of $I_{k\leftarrow i}^{\mathsf{LO}}(\ell)$ to the existing RTA for SC multiprocessor systems [10], we can judge the schedulability of a task set as follows.

Lemma 3.1. *Let R_k^{LO} denote the smallest ℓ ($\le D_k$) that satisfies the following inequality; if this ℓ does not exist, R_k^{LO} is set to ∞.*

$$\ell \ge C_k^{\mathsf{LO}} + \left\lfloor \frac{1}{m} \cdot \sum_{\tau_i \in \tau \setminus \{\tau_k\}} \min\left(\textit{the RHS of Eq. (4)}, \ell - C_k^{\mathsf{LO}} + 1\right) \right\rfloor. \tag{5}$$

Then, τ is schedulable by EDF in MC multiprocessor systems under LO-criticality behavior, if every task $\tau_k \in \tau$ satisfies $R_k^{\mathsf{LO}} \le D_k$.

Proof. Here we summarize the proof in [10], which is by contradiction. Suppose that R_k^{LO} computed by the lemma is no greater than D_k but the actual response time of τ_k is greater than R_k^{LO}. We can derive the following equation from the fact that the iteration ends in Eq. (5).

$$R_k^{\mathsf{LO}} = C_k^{\mathsf{LO}} + \left\lfloor \frac{1}{m} \cdot \sum_{\tau_i \in \tau \setminus \{\tau_k\}} \min\left(W_i^{\mathsf{LO}}(R_k^{\mathsf{LO}}, S_i^{\mathsf{LO}}), E_{k\leftarrow i}^{\mathsf{LO}}(S_i^{\mathsf{LO}}), R_k^{\mathsf{LO}} - C_k^{\mathsf{LO}} + 1\right) \right\rfloor. \tag{6}$$

From Eqs. (4) and (6), we can derive the following inequality.

$$R_k^{\mathsf{LO}} \ge C_k^{\mathsf{LO}} + \left\lfloor \frac{1}{m} \cdot \sum_{\tau_i \in \tau \setminus \{\tau_k\}} \min\left(I_{k\leftarrow i}^{\mathsf{LO}}(R_k^{\mathsf{LO}}), R_k^{\mathsf{LO}} - C_k^{\mathsf{LO}} + 1\right) \right\rfloor. \tag{7}$$

It is a trivial fact that, for τ_k to be schedulable, the amount of time in which τ_k cannot be executed as a result of the execution of other jobs within an interval of length R_k^{LO} should be less than $(R_k^{\mathsf{LO}} - C_k^{\mathsf{LO}} + 1)$. Note that the interference by jobs of τ_i with that of τ_k of interest is limited to at most $(R_k^{\mathsf{LO}} - C_k^{\mathsf{LO}} + 1)$ [10]. Then, if the actual response time cannot be bounded by R_k^{LO}, the following inequality must hold.

$$\sum_{\tau_i \in \tau \setminus \{\tau_k\}} \min\left(I_{k\leftarrow i}^{\mathsf{LO}}(R_k^{\mathsf{LO}}), R_k^{\mathsf{LO}} - C_k^{\mathsf{LO}} + 1\right) \geq m \cdot \left(R_k^{\mathsf{LO}} - C_k^{\mathsf{LO}} + 1\right). \tag{8}$$

From Eqs. (7) and (8), we get

$$R_k^{\mathsf{LO}} \geq C_k^{\mathsf{LO}} + \left\lfloor \frac{1}{m} \cdot m \cdot \left(R_k^{\mathsf{LO}} - C_k^{\mathsf{LO}} + 1\right) \right\rfloor = R_k^{\mathsf{LO}} + 1, \tag{9}$$

which contradicts the supposition. □

In Sect. 4.2, we will explain how to find ℓ such that it satisfies Eq. (5). In addition, we will explain how to update $\{S_i^{\mathsf{LO}}\}_{\tau_i \in \tau}$.

3.2 RTA for EDF under HI-Criticality Behavior

The previous subsection describes the development of RTA for EDF under LO-criticality behavior in MC multiprocessor systems, which is accomplished by simply applying existing RTA for EDF in SC multiprocessor systems. Unlike RTA for EDF under LO-criticality behavior, developing that under HI-criticality behavior in MC multiprocessor systems requires a careful investigation how each job that experiences a system transition is interfered with by other jobs. Whereas a job of interest can be interfered with by jobs under LO-criticality behavior before the system transition, it can also be interfered with by those under HI-criticality behavior after the system transition. Addressing this concern, we next calculate the response times of tasks under EDF under HI-criticality behavior.

Extending $I_{k\leftarrow i}^{\mathsf{LO}}(\ell)$, we let $I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}})$ denote the length of the cumulative intervals such that jobs of τ_i execute but the job of τ_k of interest cannot execute within an interval of length ℓ starting at the job's release time when a transition occurs ℓ^{TR} after the job's release time for $0 < \ell^{\mathsf{TR}} < \ell$ as shown in Fig. 1(b). In addition, we express $I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}} = 0)$ as the length at which the system transition occurs before or at the job's release time as shown in Fig. 1(c). We do not consider the case of $\ell^{\mathsf{TR}} \geq \ell$, because it is a type of LO-criticality behavior as shown in Fig. 1(a).

Similar to Eq. (1), a job of τ_k finishes its execution within ℓ time units after its release when the system transition occurs after ℓ^{TR} time units from the job's release time (or before), if the following inequality holds.

$$\ell \geq C_k^{\mathsf{HI}} + \left\lfloor \frac{1}{m} \cdot \sum_{\tau_i \in \tau \setminus \{\tau_k\}} \min\left(I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}}), \ell - C_k^{\mathsf{HI}} + 1\right) \right\rfloor. \tag{10}$$

Next, we calculate the upper-bounds of $I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}})$ for $0 \leq \ell^{\mathsf{TR}} < \ell$. If a task τ_i satisfies $L_i = \mathsf{LO}$, $I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}})$ is the same as $I_{k\leftarrow i}^{\mathsf{LO}}(\ell^{\mathsf{TR}})$ because jobs of a task τ_i with $L_i = \mathsf{LO}$ cannot execute after the system transition. This calculation is recorded as follows from Eq. (4).

$$I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}}) \leq \min\left(\ell^{\mathsf{TR}}, W_i^{\mathsf{LO}}(\ell^{\mathsf{TR}}, S_i^{\mathsf{LO}}), E_{k\leftarrow i}^{\mathsf{LO}}(S_i^{\mathsf{LO}})\right). \tag{11}$$

However, calculating $I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}})$ for a task τ_i with $L_i = \mathsf{HI}$ requires that we carefully consider the system transition. Here, we consider two cases: $I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}})$ for $\ell^{\mathsf{TR}} = 0$ and $0 < \ell^{\mathsf{TR}} < \ell$.

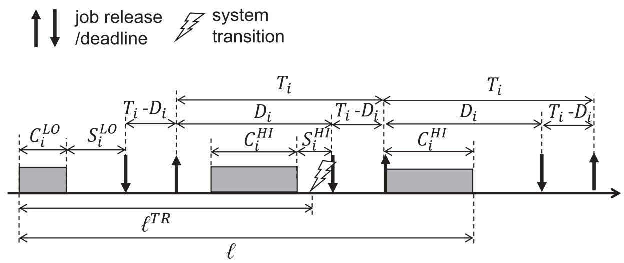
First, if the system transition occurs no later than the beginning of the interval of length ℓ (i.e., Fig. 1(c)), all jobs of τ_i within the interval can execute up to C_i^{HI}. Therefore, we can simply apply $E(\cdot)$ and $W(\cdot)$ functions for the parameter with C_i^{HI}. In other words, $I_{k\leftarrow i}^{\mathsf{HI}}(\ell, 0)$ is upper-bounded by $W_i^{\mathsf{HI}}(\ell, S_i^{\mathsf{HI}}) \overset{\text{def.}}{=} W(\ell, T_i, C_i^{\mathsf{HI}}, D_i, S_i^{\mathsf{HI}})$ under any work-conserving preemptive scheduling. In addition, $I_{k\leftarrow i}^{\mathsf{HI}}(\ell, 0)$ is upper-bounded by $E_{k\leftarrow i}^{\mathsf{HI}}(S_i^{\mathsf{HI}}) \overset{\text{def.}}{=} E(D_k, T_i, C_i^{\mathsf{HI}}, S_i^{\mathsf{HI}})$ under EDF. In summary, $I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}} = 0)$ is upper-bounded as follows:

$$I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}} = 0) \leq \min\left(W_i^{\mathsf{HI}}(\ell, S_i^{\mathsf{HI}}), E_{k\leftarrow i}^{\mathsf{HI}}(S_i^{\mathsf{HI}})\right). \tag{12}$$

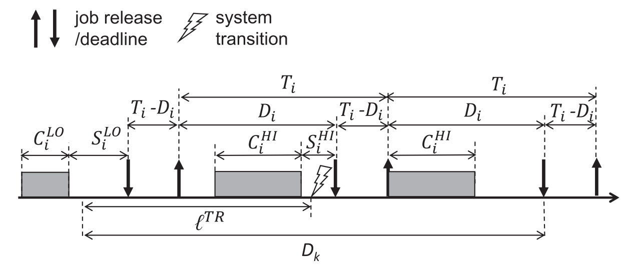
Second, if the system transition occurs in the middle of the interval of interest of length ℓ (i.e., Fig. 1(b)), we should consider that each job of τ_i is executed up to C_i^{LO} before the system transition, but up to C_i^{HI} after the system transition. Hereafter, we upper-bound $I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}})$ for $0 < \ell^{\mathsf{TR}} < \ell$.

We first develop an upper-bound of $I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}})$ for $0 < \ell^{\mathsf{TR}} < \ell$ under any work-conserving scheduling algorithm. Let us consider an imaginary situation as shown in Fig. 3(a). Similar to what is shown in Fig. 2(a), the interval of interest of length ℓ ends with the finishing time of the last job. The last job of τ_i is executed as early as possible, whereas the other jobs of τ_i finishes its execution S_i^{LO} (of the left-most job executing for C_i^{LO}) or S_i^{HI} (of the middle job executing for C_i^{HI}) ahead of its absolute deadline. Regarding execution time, a job of τ_i will execute up to C_i^{HI} if the system transition occurs before its absolute deadline (see the second and third jobs in Fig. 3(a)), and C_i^{LO} otherwise (see the first job in Fig. 3(a)). Note that this situation is imaginary. In reality, it is impossible for the second job in Fig. 3(a) to execute up to C_i^{HI}, because its execution completes before the system transition. Then, the number of jobs with C_i^{HI} is $N_i^W(\ell - \ell^{\mathsf{TR}})$, where $N_i^W(\ell) \overset{\text{def.}}{=} \left\lceil \frac{\ell + D_i - C_i^{\mathsf{HI}}}{T_i} \right\rceil$, and the total execution of jobs with C_i^{LO} is calculated by $\max\left(0, E(\ell - N_i^W(\ell - \ell^{\mathsf{TR}}) \cdot T_i, T_i, C_i^{\mathsf{LO}}, S_i^{\mathsf{LO}})\right)$. We denote the amount of execution of this situation by $W_i^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}}, S_i^{\mathsf{LO}})$, which is calculated as follows:

$$\begin{aligned} W_i^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}}, S_i^{\mathsf{LO}}) &= N_i^W(\ell - \ell^{\mathsf{TR}}) \cdot C_i^{\mathsf{HI}} \\ &\quad + \max\left(0, E(\ell - N_i^W(\ell - \ell^{\mathsf{TR}}) \cdot T_i, T_i, C_i^{\mathsf{LO}}, S_i^{\mathsf{LO}})\right). \end{aligned} \tag{13}$$



(a) $W_i^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}}, S_i^{\mathsf{LO}})$ in Eq. (13)



(b) $E_{k\leftarrow i}^{\mathsf{HI}}(\ell^{\mathsf{TR}}, S_i^{\mathsf{LO}})$ in Eq. (14)

Fig. 3 Functions required to upper-bound the interference when the system transition occurs in the middle of the interval of interest.

Then, $W_i^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}}, S_i^{\mathsf{LO}})$ is an upper-bound of $I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}})$ for $0 < \ell^{\mathsf{TR}} < \ell$ as follows.

Lemma 3.2. *$W_i^{HI}(\ell, \ell^{TR}, S_i^{LO})$ in Eq.* (13) *is an upper-bound of $I_{k\leftarrow i}^{HI}(\ell, \ell^{TR})$ for* $0 < \ell^{TR} < \ell$.

Proof. Suppose that the amount of execution of jobs of τ_i in an interval of length ℓ when the system transition occurs after ℓ^{TR} time units from the beginning of the interval of length ℓ exceeds $W_i^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}}, S_i^{\mathsf{LO}})$. We will show a contradiction.

Because we apply the ceiling function for $N_i^W(\ell - \ell^{\mathsf{TR}})$, it is straightforward that the amount of execution of jobs of τ_i after the system transition is at most $N_i^W(\ell - \ell^{\mathsf{TR}}) \cdot C_i^{\mathsf{HI}}$.

Therefore, $W_i^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}}, S_i^{\mathsf{LO}})$ upper-bounds $I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}})$ for $0 < \ell^{\mathsf{TR}} < \ell$ for the same reason that $W_i^{\mathsf{LO}}(\ell, S_i^{\mathsf{LO}})$ upper-bounds $I_{k\leftarrow i}^{\mathsf{LO}}(\ell)$. In other words, if we shift the release and execution pattern of $W_i^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}}, S_i^{\mathsf{LO}})$ in Eq. (13) slightly to the left, we cannot get any more interference from the end of the interval. However, if we shift slightly to the right, we lose some interference from the last job and may get some interference from the beginning of the interval. The last job loses up to C_i^{HI} of interference, but the first job gets up to C_i^{LO} of interference. This means that the change of interference cannot be positive, which contradicts the supposition. Therefore, the lemma holds. □

Using the prioritization policy of EDF, we can upper-bound $I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}})$ for $0 < \ell^{\mathsf{TR}} < \ell$. To this end, let us consider an imaginary situation as shown in Fig. 3(b). Similar to Fig. 2(b), the end of the interval of interest of length D_k corresponds to the end of the absolute deadline of the last job of τ_i; all jobs of τ_i are executed as late as possible. In addition, similar to what is shown in Fig. 3(a), a job of τ_i will execute up to C_i^{HI} if the system transition occurs before its absolute deadline (see the second and third jobs in Fig. 3(b)), and C_i^{LO} otherwise (see the first job in the figure). As a reminder, this situation is imaginary. In reality, the second job shown in Fig. 3(b) cannot execute up to C_i^{HI} because the system transition occurs after execution is finished. The number of jobs of τ_i with C_i^{HI} is calculated by $N_i^E(D_k - \ell^{\mathsf{TR}})$, where $N_i^E(\ell) \overset{\text{def.}}{=} \left\lceil \frac{\ell}{T_i} \right\rceil$ and the amount of execution jobs of τ_i with C_i^{LO} is calculated by $\max\left(0, E(D_k - N_i^E(D_k - \ell^{\mathsf{TR}}) \cdot T_i, T_i, C_i^{\mathsf{LO}}, S_i^{\mathsf{LO}})\right)$. We denote the amount of execution of this situation by $E_{k\leftarrow i}^{\mathsf{HI}}(\ell^{\mathsf{TR}}, S_i^{\mathsf{LO}})$, which is calculated as follows:

$$\begin{aligned} E_{k\leftarrow i}^{\mathsf{HI}}(\ell^{\mathsf{TR}}, S_i^{\mathsf{LO}}) &= N_i^E(D_k - \ell^{\mathsf{TR}}) \cdot C_i^{\mathsf{HI}} \\ &+ \max\left(0, E(D_k - N_i^E(D_k - \ell^{\mathsf{TR}}) \cdot T_i, T_i, C_i^{\mathsf{LO}}, S_i^{\mathsf{LO}})\right). \end{aligned} \tag{14}$$

Then, $E_{k\leftarrow i}^{\mathsf{HI}}(\ell^{\mathsf{TR}}, S_i^{\mathsf{LO}})$ is an upper-bound of $I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}})$ for $0 < \ell^{\mathsf{TR}} < \ell$ as follows.

Lemma 3.3. *$E_{k\leftarrow i}^{HI}(\ell^{TR}, S_i^{LO})$ in Eq.* (14) *is an upper-bound of $I_{k\leftarrow i}^{HI}(\ell, \ell^{TR})$ for* $0 < \ell^{TR} < \ell$.

Proof. The proof is similar to that of Lemma 3.2.

Consider an interval of length D_k. Here, we focus only on jobs of τ_k whose absolute deadline is no later than the end of the interval. Suppose that the amount of execution of jobs of τ_i in the interval when the system transition occurs after ℓ^{TR} time units from the beginning of the interval of length D_k, exceeds $E_{k\leftarrow i}^{\mathsf{HI}}(\ell^{\mathsf{TR}}, S_i^{\mathsf{LO}})$. We will show a contradiction.

In $N_i^E(D_k - \ell^{\mathsf{TR}})$, we apply the ceiling function. Therefore, the amount of execution of jobs of τ_i after the system transition is at most $N_i^E(D_k - \ell^{\mathsf{TR}}) \cdot C_i^{\mathsf{HI}}$. If we shift the release and execution pattern of $E_{k\leftarrow i}^{\mathsf{HI}}(\ell^{\mathsf{TR}}, S_i^{\mathsf{LO}})$ slightly to the left, no additional interference occurs, whereas the first job may lose some interference. On the other hand, if we shift slightly to the right, the last job's absolute deadline is later than the end of the interval of interest. Therefore, although we get some additional interference from the first job, we lose the entire interference of the last job. Therefore, any shift cannot increase interference, which contradicts the supposition. Therefore, the lemma holds. □

In addition, if the system transition occurs in the middle of the interval of interest of length ℓ, the amount of execution of jobs of τ_i with $L_i = \mathsf{HI}$ should be no greater than that when the system transition occurs before the interval in the worst case (i.e., the RHS of Eq. (12)). Therefore, the RHS of Eq. (12) is also an upper-bound of $I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}})$ for $0 < \ell^{\mathsf{TR}} < \ell$. In summary, $I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}})$ for $0 \le \ell^{\mathsf{TR}} < \ell$ can be calculated as follows:

$$\begin{aligned} I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}}) \le \min\Big(&W_i^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}}, S_i^{\mathsf{LO}}), W_i^{\mathsf{HI}}(\ell, S_i^{\mathsf{HI}}), \\ &E_{k\leftarrow i}^{\mathsf{HI}}(\ell^{\mathsf{TR}}, S_i^{\mathsf{LO}}), E_{k\leftarrow i}^{\mathsf{HI}}(S_i^{\mathsf{HI}})\Big). \end{aligned} \tag{15}$$

For a given ℓ^{TR}, we can judge the schedulability of a task set as follows.

Lemma 3.4. *Let $R_k^{HI}(\ell^{TR})$ denote the smallest ℓ ($\leq D_k$) that satisfies the following inequality for a given $0 \leq \ell^{TR} \leq \min(\ell, R_k^{LO})$; if this ℓ does not exist, $R_k^{HI}(\ell^{TR})$ is set to ∞.*

$$\ell \geq C_k^{HI} + \left\lfloor \frac{1}{m} \cdot \sum_{\tau_i \in \tau \setminus \{\tau_k\}} \min\Big(\text{the RHS of Eq. (11) or (15)}, \ell - C_k^{HI} + 1\Big) \right\rfloor. \tag{16}$$

Note that in Eq. (16)*, Eq.* (11) *is used for L_i = LO, and Eq.* (15) *is used for L_i = HI.*

Let R_k^{HI} denote $\max_{0 \leq \ell^{TR} \leq R_k^{LO}} R_k^{HI}(\ell^{TR})$. Here, τ is schedulable by EDF in MC multiprocessor systems under HI-criticality behavior, if all tasks $\tau_k \in \tau$ satisfy $R_k^{HI} \leq D_k$.

Proof. Because Eq. (15) holds, the lemma holds based on the same reasoning as Lemma 3.1. The difference is that multiple choices exist for ℓ^{TR}. The range of ℓ^{TR} is upper-bounded by R_k^{LO}; otherwise, the job of interest is already finished in LO-criticality behavior. □

Combining Lemmas 3.1 and 3.4, we can judge the schedulability of a task set in MC multiprocessor systems, recorded as follows.

Theorem 3.5. *τ is schedulable by EDF in MC multiprocessor systems, if τ is deemed schedulable by both Lemmas 3.1 and 3.4.*

Proof. The theorem immediately holds by Lemmas 3.1 and 3.4. □

4. EDZL Scheduling Algorithm and Its RTA in MC Multiprocessor Systems

In this section, we describe our design for EDZL scheduling algorithm in MC multiprocessor systems, and the subsequent development of its RTA.

4.1 EDZL Scheduling Algorithm in MC Multiprocessor Systems

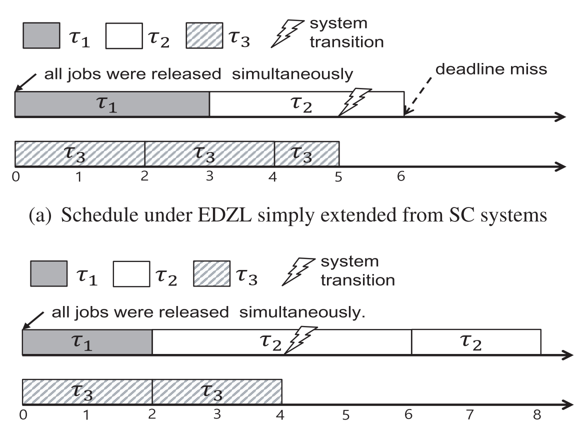
In SC systems, a job's laxity at any time instant is defined as the remaining time to its absolute deadline minus the amount of remaining execution time at that instant [6]. A zero-laxity job will miss its absolute deadline unless it starts its execution immediately. Therefore, zero-laxity-based algorithms that give the highest priority to zero-laxity jobs improve schedulability of their corresponding base algorithms (e.g., EDZL outperforms EDF).

Regarding MC multiprocessor systems, we must redefine a job's laxity under MC multiprocessor systems, because multiple types of the worst-case execution times depend on verification authorities (i.e., C_i^{LO} and C_i^{HI}) in this study. In other words, if we simply borrow the notion of a job's laxity from SC systems, the amount of remaining execution time of a job of τ_i at t under MC multiprocessor systems is calculated by, C_i^{LO} minus the amount of execution of the job performed until t under LO-criticality, and C_i^{HI} minus the amount of execution of the job performed until t under HI-criticality behavior. If we apply the notion, we cannot take full advantage of zero-laxity-based algorithms, as shown in the following example.

Example 1. *Consider a task set consisting of $\tau_1(T_1 = 5, C_1^{LO} = 3, C_1^{HI} = 3, D_1 = 5, L_1 = \text{LO})$, $\tau_2(6, 2, 4, 6, \text{HI})$ and $\tau_3(2, 2, 2, 2, \text{LO})$ scheduled on two processors. As shown in Fig. 4(a), we assume that three tasks periodically invoke their jobs from $t = 0$, and the execution time for a job of τ_2 exceeds $C_2^{LO} = 2$ at $t = 5$, meaning that the system transition occurs at $t = 5$. Note that we do not care tasks whose criticality is low (i.e., τ_1 and τ_3 with $L_1 = L_3 = \text{LO}$) after the system transition as we stated in Sect. 2; τ_1 in Fig. 4(b) after a system transition (i.e., after time instant 4) does not affect schedulability of the system even if execution of τ_1 is not completed until τ_1's absolute deadline. This model is a typical MC system model as we explained in Sect. 2.*

Suppose that we give the highest priority to zero-laxity jobs, in which the remaining execution for calculating a job's laxity at t is calculated by the execution time under the system criticality behavior at t (LO or HI) minus the amount of execution of the job performed until t. In addition, for jobs with positive laxity, we give a higher priority to a job with an earlier absolute deadline. Then, all jobs of τ_3 always have zero-laxity (i.e. at any time instant) and the highest priority. The first job of τ_1 has a higher priority than that of τ_2 because its absolute deadline is earlier. Therefore, the schedule until $t = 5$ is shown in Fig. 4(a). After the system transition occurs at $t = 5$ due to the job of τ_2, the job misses its absolute deadline at $t = 6$. This is because once the system transition occurs at $t = 5$, the job of τ_2's laxity is already negative, meaning that no schedule can meet its deadline.

As shown in the example, we should not define a laxity of a job using its remaining execution time that is based on system criticality behavior. Instead, although the system ex-



(a) Schedule under EDZL simply extended from SC systems

(b) Schedule under EDZL tailored for MC multiprocessor systems

Fig. 4 Two schedules under EDZL with different laxity definitions.

hibits LO-criticality behavior, the remaining execution time should be calculated based on HI-criticality behavior in order to reserve additional execution incurred after the system transition. To formalize EDZL scheduling algorithm in MC multiprocessor systems, let $C_i^{\mathsf{LO}}(t)$ denote the remaining execution time of a job of τ_i of interest at time instant t before the system transition. This is calculated by C_i^{LO} minus the amount of execution of the job of interest performed until t. Similarly, let $C_i^{\mathsf{HI}}(t)$ denote the remaining execution time of a job of τ_i of interest at time instant t after the system transition. This is calculated by C_i^{HI} minus the amount of execution of the job of interest performed until t. In addition, let $D_i(t)$ denote the remaining time to the absolute deadline of a job of τ_i of interest. We then define a laxity of a job of τ_i at t as follows.

Definition 1. *The laxity of a job of τ_i at t is defined as follows:*

- $D_i(t) - C_i^{\mathsf{LO}}(t) - (C_i^{\mathsf{HI}} - C_i^{\mathsf{LO}})$*, if t is before the system transition, and*
- $D_i(t) - C_i^{\mathsf{HI}}(t)$*, if t is after the system transition.*

EDZL scheduling algorithm for MC multiprocessor systems then functions as follows. If a job's laxity defined in Def. 1 is zero, the job's priority becomes the highest. Otherwise, each job's priority is determined by its absolute deadline; the earlier the absolute deadline, the higher the priority.

Once we apply EDZL for MC multiprocessor systems as previously explained, we can avoid a deadline miss for the situation shown in Fig. 4(a). In other words, as shown in Fig. 4(b), the first job of τ_2 has a zero laxity at $t = 2$, i.e., $D_2(2)-C_2^{\mathsf{LO}}(2)-(C_2^{\mathsf{HI}}-C_2^{\mathsf{LO}}) = 4-2-(4-2) = 0$. Therefore, the job has the highest priority after $t = 2$, which yields no deadline miss.

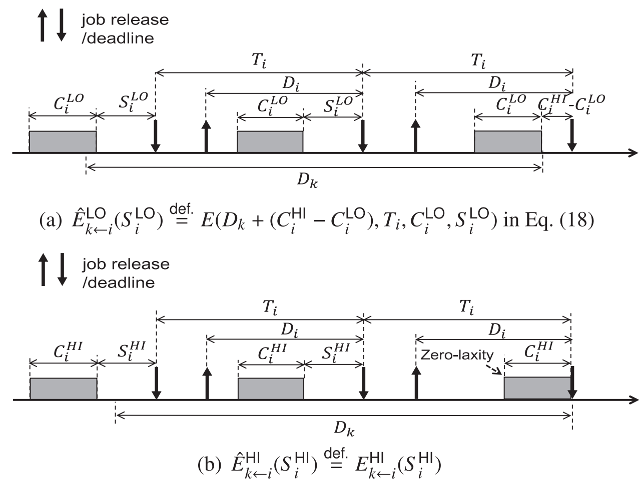
4.2 RTA for EDZL under LO-Criticality Behavior

We next develop RTA for EDZL under LO-criticality behavior. To this end, we calculate the interference upper-bound of jobs of τ_i to the job of interest of τ_k (i.e., $I_{k\leftarrow i}^{\mathsf{LO}}(\ell)$) under EDZL.

If a job of τ_i exhibits a positive laxity, the job cannot have a higher priority unless its absolute deadline is earlier than that of other jobs. Therefore, the interference upper-bound under EDZL is the same as that under EDF, recorded as follows:

$$\hat{E}_{k\leftarrow i}^{\mathsf{LO}}(S_i^{\mathsf{LO}}) = E_{k\leftarrow i}^{\mathsf{LO}}(S_i^{\mathsf{LO}}) = E(D_k, T_i, C_i^{\mathsf{LO}}, S_i^{\mathsf{LO}}). \quad (17)$$

However, if a job of τ_i exhibits a zero laxity, the job can have a higher priority even if its absolute deadline is later than that of other jobs. In addition, each job τ_i can have a zero-laxity even though $(C_i^{\mathsf{HI}} - C_i^{\mathsf{LO}})$ amount of slack exists before its absolute deadline. Therefore, the interference is maximized when the difference between the last job's absolute deadline and the end of the interval of interest of length D_k is $C_i^{\mathsf{HI}} - C_i^{\mathsf{LO}}$, as shown in Fig. 5(a). The interference upper-bound can be then calculated as follows:



(a) $\hat{E}_{k\leftarrow i}^{\mathsf{LO}}(S_i^{\mathsf{LO}}) \overset{\text{def.}}{=} E(D_k + (C_i^{\mathsf{HI}} - C_i^{\mathsf{LO}}), T_i, C_i^{\mathsf{LO}}, S_i^{\mathsf{LO}})$ in Eq. (18)

(b) $\hat{E}_{k\leftarrow i}^{\mathsf{HI}}(S_i^{\mathsf{HI}}) \overset{\text{def.}}{=} E_{k\leftarrow i}^{\mathsf{HI}}(S_i^{\mathsf{HI}})$

Fig. 5 Functions required to upper-bound the interference under EDZL.

$$\hat{E}_{k\leftarrow i}^{\mathsf{LO}}(S_i^{\mathsf{LO}}) = E(D_k + (C_i^{\mathsf{HI}} - C_i^{\mathsf{LO}}), T_i, C_i^{\mathsf{LO}}, S_i^{\mathsf{LO}}). \quad (18)$$

Considering that $W_i^{\mathsf{LO}}(\ell, S_i^{\mathsf{LO}})$ given in Sect. 3.1 can be an interference upper-bound in any work-conserving scheduling algorithm, $I_{k\leftarrow i}(\ell)$ under EDZL and LO-criticality behavior is upper-bounded as follows.

$$I_{k\leftarrow i}^{\mathsf{LO}}(\ell) \le \min\Big(W_i^{\mathsf{LO}}(\ell, S_i^{\mathsf{LO}}), \hat{E}_{k\leftarrow i}^{\mathsf{LO}}(S_i^{\mathsf{LO}})\Big). \quad (19)$$

Considering that at least $m + 1$ zero-laxity jobs should exist in order for a job to miss its absolute deadline [7], [11], we can calculate the response time of tasks under EDZL and LO-criticality behavior in the following lemma.

Lemma 4.1. *Let R_k^{LO} denote the smallest ℓ ($\le D_k$) that satisfies the following inequality; if this ℓ does not exist, R_k^{LO} is set to ∞.*

$$\ell \ge C_k^{\mathsf{LO}} + \left\lfloor \frac{1}{m} \cdot \sum_{\tau_i \in \tau \setminus \{\tau_k\}} \min\Big(\text{the RHS of Eq. (19)}, \ell - C_k^{\mathsf{LO}} + 1\Big) \right\rfloor. \quad (20)$$

τ is schedulable by EDZL in MC multiprocessor systems under LO-criticality behavior, if one of the following conditions holds:

C1. All tasks $\tau_k \in \tau$ satisfy $R_k^{\mathsf{LO}} \le D_k$, or

C2. $|\tau| - m$ tasks $\tau_k \in \tau$ satisfy $R_k^{\mathsf{LO}} < D_k$.

Proof. Because Eq. (19) holds, the lemma holds based on the same reasoning as Lemma 3.1. The difference is C2, which is derived from the prioritization policy of EDZL. In other words, if there are at most m tasks with zero laxity, then all zero-laxity tasks are always scheduled, yielding no deadline miss. Therefore, if C2 holds, τ is schedulable by EDZL.

One may argue that it does not make sense that C2 guarantees the schedulability of a task set when there exists a task τ_k with $R_k^{\mathsf{LO}} = \infty$. However, C2 is a basic principle of zero-laxity-based scheduling algorithm, and $R_k^{\mathsf{LO}} = \infty$ means that our schedulability test framework cannot guarantee τ_k's schedulability before the transition while the task in reality can be schedulable. If C2 holds, there are at most m tasks which may reach a zero-laxity state. Then, whenever the at most m tasks reach a zero-laxity state, they are scheduled according to the EDZL policy (giving the highest priority to the zero-laxity task). Therefore, the at most m tasks never miss their deadlines, although each of them is not deemed schedulable by RTA (i.e., $R_k^{\mathsf{LO}} = \infty$). Actually, this is why EDZL schedulability analysis is much better than EDF schedulability analysis. □

Two details about Lemma 4.1 may be perplexing: how to have ℓ satisfy Eq. (20), and how to update S_i^{LO}, which affects the RHS of Eq. (19). We can apply the technique in [10] for both details, which we can explain as follows. Initially, we assign S_k^{LO} to 0 for every $\tau_k \in \tau$. Then, for each τ_k, we set ℓ to C_k^{LO}, and calculate the RHS of Eq. (20). If the value is greater than ℓ, we reassign the value to ℓ, and repeat to calculate RHS. In addition, if we find ℓ such that it satisfies Eq. (20), then R_k^{LO} is set to ℓ. If ℓ is greater than D_k, we stop the iteration, meaning that τ_k is not schedulable under LO-criticality behavior.

Then, once we finish calculating R_k^{LO} for every $\tau_k \in \tau$, we update $S_k^{\mathsf{LO}} = D_k - R_k^{\mathsf{LO}}$ if $R_k^{\mathsf{LO}} < D_k$. We then repeat the entire process of calculating R_k^{LO} for every $\tau_k \in \tau$ with updated $\{S_k^{\mathsf{LO}}\}$ until no update remains for $\{S_k^{\mathsf{LO}}\}$.

4.3 RTA for EDZL under HI-Criticality Behavior

We next develop RTA for EDZL under HI-criticality behavior. To this end, we calculate the interference upper-bound of jobs of τ_i to the job of interest of τ_k (i.e., $I_{k\leftarrow i}^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}})$) under EDZL.

Similar to RTA for EDZL under LO-criticality behavior, we use the interference upper-bound under EDF, if a job of τ_i has a positive laxity. Therefore, we must check what occurs if a job of τ_i has a zero laxity. We first examine a case in which the system transition occurs before the beginning of the interval of interest of length D_k. Different from LO-criticality behavior, each job of τ_i can have a zero laxity only if its execution is performed until the absolute deadline (e.g., the third job in Fig. 5(b)). Although a zero-laxity job with a later absolute deadline can have a higher priority than a positive laxity job with an earlier absolute deadline, some execution from the zero-laxity job cannot interfere with the positive laxity job as shown in the third job in Fig. 5(b) [7]. Therefore, we can use the same interference upper-bound under EDF (i.e., $E_{k\leftarrow i}^{\mathsf{HI}}(S_i^{\mathsf{HI}})$) for EDZL when the system transition occurs before the beginning of the interval of interest of length D_k. Similarly, when the system transition occurs in the middle of the interval of interest of length D_k, we use the same interference upper-bound under EDF (i.e., $E_{k\leftarrow i}^{\mathsf{HI}}(\ell^{\mathsf{TR}}, S_i^{\mathsf{LO}})$) for EDZL.

Considering that the two interference upper-bounds under EDF when the system transition occurs before the beginning of the interval of interest (i.e., $W_i^{\mathsf{HI}}(\ell, S_i^{\mathsf{HI}})$) and in the middle of the interval of interest (i.e., $W_i^{\mathsf{HI}}(\ell, \ell^{\mathsf{TR}}, S_i^{\mathsf{LO}})$) hold for any work-conserving scheduling algorithm, we can use Eqs. (11) and (15) for the interference upper-bounds under EDZL. Using the upper-bounds, the following lemma records a schedulability test of EDZL for MC multiprocessor systems under HI-criticality behavior.

Lemma 4.2. *Let $R_k^{HI}(\ell^{TR})$ denote the smallest ℓ ($\le D_k$) that satisfies the following inequality for a given $0 \le \ell^{TR} \le \min(\ell, R_k^{\mathsf{LO}})$; if this ℓ does not exist, $R_k^{HI}(\ell^{TR})$ is set to ∞.*

$$\ell \ge C_k^{HI} + \left\lfloor \frac{1}{m} \cdot \sum_{\tau_i \in \tau \setminus \{\tau_k\}} \min\Big(\text{the RHS of Eq. (11) or (15)}, \ell - C_k^{HI} + 1\Big) \right\rfloor. \tag{21}$$

Note that in Eq. (21), Eq. (11) is used for L_i = LO, and Eq. (15) is used for L_i = HI.

Let R_k^{HI} denote $\max_{0 \le \ell^{TR} \le R_k^{\mathsf{LO}}} R_k^{HI}(\ell^{TR})$. τ is schedulable by EDZL in MC multiprocessor systems under HI-criticality behavior, if one of the following conditions holds:

C1. All tasks $\tau_k \in \tau$ satisfy $R_k^{HI} \le D_k$, or

C2. $|\tau| - m$ tasks $\tau_k \in \tau$ satisfy $R_k^{HI} < D_k$.

Proof. The lemma holds based on the same reasoning as Lemma 4.1. Note that the range of ℓ^{TR} is the same as that of Lemma 3.4. □

Using Lemmas 4.1 and 4.2, we finally develop a schedulability test of EDZL in MC multiprocessor systems, recorded in the following theorem.

Theorem 4.3. *τ is schedulable by EDZL in MC multiprocessor systems, if τ is deemed schedulable by both Lemmas 4.1 and 4.2.*

Proof. The theorem immediately holds by Lemmas 4.1 and 4.2. □

5. Evaluation

This section presents the evaluation results obtained by the experiments conducted under various simulation environments to demonstrate effectiveness of the ZL policy in MC multiprocessor systems, and then discusses the characteristics of the considered schedulability analysis by investigating simulation results. As we emphasize in Sect. 1, this paper aims at demonstrating performance improvement achieved by the ZL policy when it is incorporated into base algorithms in MC multiprocessor systems. Therefore, we focus on performance of our target base algorithm (i.e., EDF) and that with the ZL policy incorporated (i.e., EDZL). The performance gap between other (candidate) base algorithms (e.g., EDF-VD, FP, etc.) and those with the ZL policy incorporated (e.g., may be named as EDZL-VD, FPZL, etc.) can be discussed after one develops their schedulability analysis, which deserves another full paper.

For our simulations, we randomly generated task sets

Table 1 Schedulability under EDF and EDZL, and schedulable ratio between EDF and EDZL when $m = 2, 4, 8$, and 16 and $p = 0.1, 0.3, 0.5, 0.7$ and 0.9.

$m = 2$				$m = 4$			
p	EDF	EDZL	EDZL to EDF	p	EDF	EDZL	EDZL to EDF
0.1	45.7%	46.3%	101.2%	0.1	31.7%	33.0%	104.0%
0.3	36.9%	41.7%	112.9%	0.3	18.6%	26.1%	140.3%
0.5	28.3%	38.7%	136.7%	0.5	10.1%	25.0%	246.4%
0.7	22.7%	39.1%	172.7%	0.7	5.4%	25.5%	469.9%
0.9	14.4%	35.1%	243.4%	0.9	1.7%	22.5%	1313.5%
$m = 8$				$m = 16$			
p	EDF	EDZL	EDZL to EDF	p	EDF	EDZL	EDZL to EDF
0.1	21.0%	23.2%	110.3%	0.1	12.7%	15.9%	125.2%
0.3	8.1%	16.6%	205.9%	0.3	2.7%	11.5%	430.5%
0.5	2.3%	14.2%	627.4%	0.5	0.3%	7.6%	2606.9%
0.7	0.6%	16.0%	2703.4%	0.7	0.0%	8.1%	∞%
0.9	0.1%	14.7%	18387.5%	0.9	0.0%	7.9%	∞%

based on the task set generation method used in [12], [13]. We considered two parameters m and p of each task set τ, which are described as follows: m denotes the number of processors on which each task set is scheduled; we considered four choices of $m = 2, 4, 8$ or 16. p represents the probability that a task contained in a task set is a HI-criticality task, and was chosen from a set $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ (e.g, $p = 0.3$ for a task set τ means that a task τ_i in a task set τ can be a high-criticality task with 30% probability).

For each task $\tau_i \in \tau$, T_i was uniformly chosen from an interval $[1, 1000]$, and L_i was selected from a set $\{\mathsf{LO}, \mathsf{HI}\}$ with probability p. We uniformly selected two values from $[1, T_i]$. If $L_i = \mathsf{HI}$, C_i^{HI} and C_i^{LO} were set to the higher and lower values, respectively; otherwise (i.e., $L_i = \mathsf{HI}$), $C_i^{\mathsf{HI}} = C_i^{\mathsf{LO}}$ was set to the lower value.

With each value of m and p, we determined if the utilization was smaller than or equal to m for each task set†, and then we used the task set for our simulation if it satisfied this condition. More specifically, we repeated the following procedure until we get 10,000 task sets for each combination of m and p.

1. First, we generated a task set τ containing $m + 1$ tasks.
2. We then checked whether the utilization of the generated task set τ is smaller than or equal to m. If τ satisfied the condition, we used this set in our evaluation. Then, we inserted a new task into τ and returned to Step 2. If τ did not satisfy the condition, we abandoned this task set and returned to Step 1.

Note that this section focuses on implicit-deadline task sets in which $D_i = T_i$. The simulation results of constrained-deadline task sets showed similar trends to those of implicit-deadline task sets.

We evaluated the performance of the following schedulability analysis.

- EDF: for the proposed schedulability test in Theorem 3.5, and
- EDZL: for the proposed schedulability test in Theorem 4.3.

†The task set utilization U is defined as the larger summation of C_i/T_i of each task τ_i in the task set between HI and LO-criticality (i.e., $U = \max(\sum_{\tau_i \in \tau} C_i^{\mathsf{LO}}/T_i, \sum_{\tau_i \in \tau \wedge L_i = \mathsf{HI}} C_i^{\mathsf{HI}}/T_i))$.

In Table 1, we represent the ratio of schedulable task sets for each schedulability test (i.e., the number of task sets deemed schedulable by each schedulability test over the total number of task sets generated in a given condition), according to m and p. In addition, we depict the ratio according to varying task set utilization in Fig. 6. In the table and the figure, we yield the following five main observations that show the effectiveness of EDZL when compared to EDF.

$O1$. The number of tasks deemed schedulable by EDZL is much higher than EDF for all values of m.
$O2$. The schedulable ratio between EDF and EDZL becomes greater as m increases for all given values of p.
$O3$. The schedulable ratio between EDF and EDZL becomes greater as p increases for all given values of m.
$O4$. As p increases, the number of schedulable tasks deemed schedulable by EDF drops sharply whereas that by EDZL moderately decreases.
$O5$. Schedulability of EDF and EDZL according to varying task set utilization depends on both m and p.

Both $O1$ and $O2$ demonstrate the effectiveness of EDZL scheduling algorithm, as well as the high analytical capability of EDZL (i.e., schedulability analysis of EDZL scheduling algorithm) with respect to multiprocessor real-time systems under an MC domain. Based on the concept behind EDZL scheduling algorithm, at least m zero-laxity tasks are allowed to execute without any deadline miss whereas even a single zero-laxity task is not allowed to do so in an EDF scheduling algorithm, which generates a crucial difference in schedulability. Moreover, the necessary deadline miss condition of EDZL (e.g., at least $m + 1$ zero-laxity tasks are required to make a deadline miss as C2 in Lemma 4.2 indicates) well captures the advantage of EDZL scheduling algorithm. In addition, the superiority of EDZL is that it conserves this advantage as it safely captures the property of MC scheduling, e.g., the system transition (indicated by $O1$). Because the number of allowed zero-laxity tasks for guaranteeing no deadline miss in EDZL is proportional to the number of processors m, EDZL outperforms

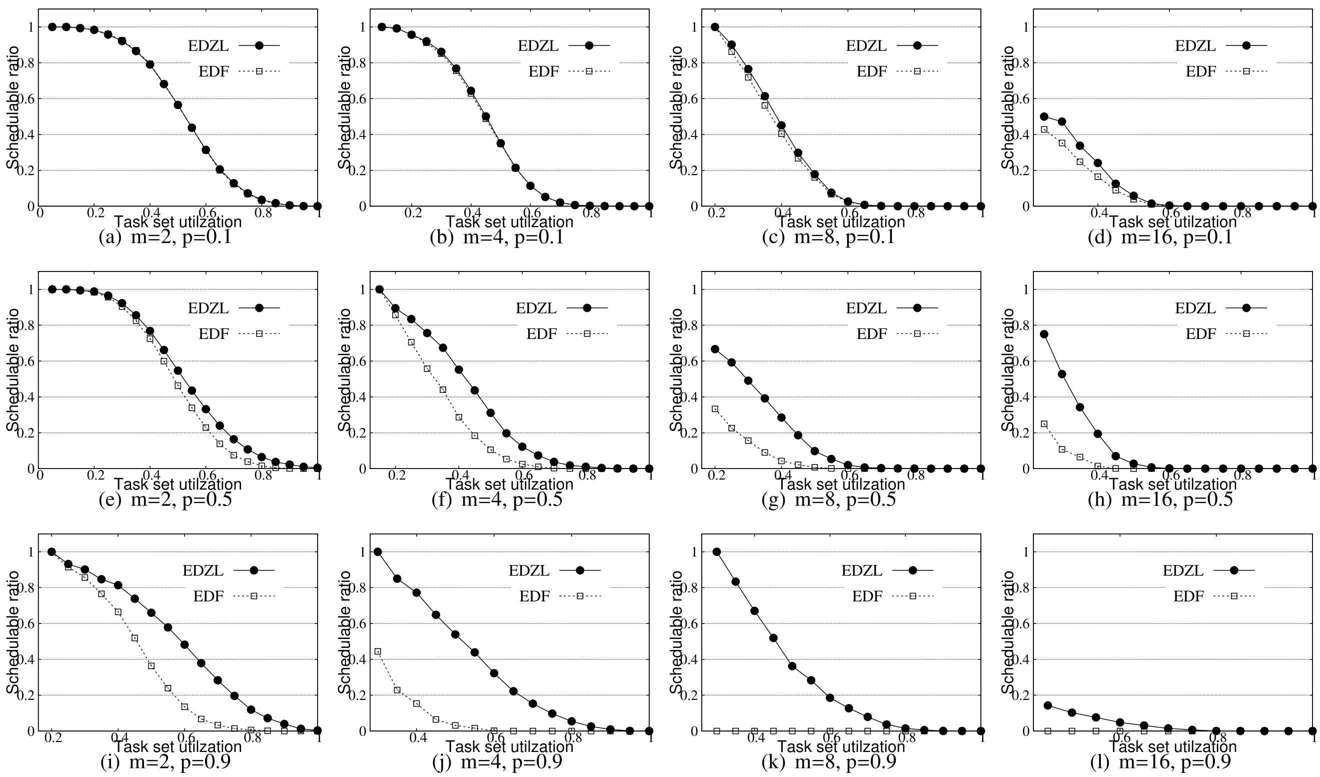


Fig. 6 Schedulability of EDF and EDZL under a pair of $m = 2, 4, 8$ or 16 and $p = 0.1$, 0.5 or 0.9, according to varying task set utilization.

EDF at an increasing rate as m increases (indicated by $O2$).

$O3$ and $O4$ show that EDZL can handle task sets with high utilization whereas EDF cannot. According to the set generation method, many tasks may exist with L_i = HI for a given higher value of p. Such tasks have relatively higher utilization than tasks with L_i = LO because they contains the higher worst-case execution time. This directly leads to a higher possibility that they are zero-laxity tasks because they are less able to accommodate the interference from higher priority tasks. Although EDF cannot effectively handle task sets with higher p because the vanilla EDF scheduling algorithm focuses solely on jobs with earliest deadlines, EDZL can handle such task sets because of a proper definition of the zero-laxity of EDZL scheduling algorithm; this virtue is well incorporated into its schedulability condition. Therefore, EDZL not only exhibits a better schedulability performance than EDF, but it also yields less schedulability degradation for higher p (indicated by $O3$ and $O4$).

$O5$ is simply observed by Fig. 6[†]. For example, schedulability of $m = 2$ and $p = 0.9$ in Fig. 6(i) according to varying task set utilization is very different from that of $m = 8$ and $p = 0.1$ in Fig. 6(c). In addition, we have the following observations from Fig. 6. First, if we focus on figures with different p and given m, the gap between schedulability of EDF and EDZL increases as p increases. For example, when it comes to $m = 4$, schedulability of EDF is similar to that of EDZL in Fig. 6(b). On the other hand, in Fig. 6(f), there is a gap between schedulability of EDF and EDZL; finally, the difference becomes significant in Fig. 6(j). Second, if we compare figures with different m and given p (e.g., Figs. 6(e), (f), (g), and (h)), the schedulability of EDF and EDZL decreases as m gets larger. Also, the difference between schedulability of EDF and EDZL increases as m gets larger except between m=8 (e.g., Fig. 6(k)) and m=16 (e.g., Fig. 6(l)). The exception (i.e., between Fig. 6(k) and Fig. 6(l)) occurs due to large values of m and p, which decrease overall schedulability of both EDF and EDZL. However, when it comes to schedulable ratio between EDZL and EDF (i.e., EDZL to EDF), it consistently increases as m gets larger as shown in Table 1.

6. Related Work

Beginning with the notion of MC scheduling [9], a large number of studies on MC scheduling for uniprocessor platforms have been made. Baruah et al. demonstrated a dominance relation between adaptive and static mixed-criticality in RTA [14], and proposed a new scheduling algorithm called EDF-VD and its schedulability analysis [15], [16]. Li et al. proposed OCBP (Own Criticality Based Priority) scheduling algorithm and expanded the schedulability

[†]For some sub-figures in Fig. 6, the range of x-axis starts larger than 0. In the sub-figures, task sets with very small task set utilization are not generated due to the task set generation procedure. For example, if m = 16, the number of tasks in a task set is at least 17; with p = 0.9, it is difficult for a task set to have less than 0.4 task set utilization because it is difficult for at least 17 (mostly HI-)tasks to have small task utilization.

analysis to general task sets [17], [18]. Employing OCBP scheduling algorithm, Guan et al. proposed a more efficient algorithm called PLRS (Priority List Reuse Scheduling) [19]. Regarding new models, Su et al. considered E-MC (Elastic Mixed-Criticality) model [20], and Baruah proposed a general recurrent real-time task model in which parameters of a task (the worst-case execution time, relative deadline and period) have different values according to each criticality level [21].

Based on achievement of real-time scheduling on MC uniprocessor platforms, several studies addressed MC global scheduling issues on a multiprocessor platform. Pathan analyzed FP and applied Ausley's approach [4]. Li et al. extended EDF-VD to multiprocessors [3]. Su et al. studied the E-MC model in multicore systems considering the systems with and without task migrations [22]. Lee et al. proposed MC-Fluid scheduling algorithm in which each task executes with a different criticality-dependent execution rate [23]. Liu et al. proposed a synchronous MC job model [24]. Although the interference-based schedulability tests known as RTA and DA (deadline analysis) [10], [13] are the most effective techniques for developing a tighter schedulability test for SC scheduling on a multiprocessor platform, only a few studies have extended the interference-based schedulability test to MC scheduling. Moreover, no interference-based schedulability test was known to exist for the most basic scheduling algorithm EDF (as well as EDZL), until we addressed it in this paper.

Suzuki et al. analyzed parallel scheduling with a directed acyclic graph [25]. Leng et al. analyzed a constant-time admission control algorithm under EDF [26]. Zhang et al. proposed an energy-aware scheduling for real-time tasks [27]. Yamaguchi et al. proposed an efficient EDF scheduling for out-of-order stream queues [28].

7. Conclusion

In this paper, we demonstrated that the ZL policy is also effective in improving the schedulability of the base algorithm in MC multiprocessor systems. To this end, we consider EDF as the base algorithm of the ZL policy and developed RTA for EDF in MC multiprocessor systems. We next designed EDZL scheduling algorithm by incorporating the ZL policy into EDF in MC multiprocessor systems, and then developed its RTA. Our simulation results demonstrated that the ZL policy considerably improves schedulability of the base algorithm (i.e., EDF). In the future, we would like to incorporate the ZL policy into other scheduling algorithms such as EDF-VD and FP, and develop schedulability analysis for them.

Acknowledgements

Earlier, naive ideas for EDZL scheduling algorithms and schedulability analysis for mixed-criticality multiprocessor real-time systems have been presented in 3-page-long (but about 2-page-long in this template) Korean conference papers [29], [30] (written in Korean).

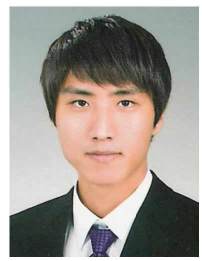
This research was also supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No.R0190-15-2071, Open PNP platform for diversity of autonomous vehicle based on cloud map). This research was also supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2017R1A2B2002458). Jinkyu Lee is the corresponding author.

References

[1] ARINC 653: Avionics application software standard interface: Arinc specification 653, required services, Aeronautical Radio, Inc.
[2] AUTOSAR classic platform 4.3, https://www.autosar.org/standards/classicplatform/release-43/
[3] H. Li and S. Baruah, "Global mixed-criticality scheduling on multiprocessors," Proc. Euromicro Conference on Real-Time Systems (ECRTS), pp.166–175, 2012.
[4] R. Pathan, "Schedulability analysis of mixed-criticality systems on multiprocessors," Proc. Euromicro Conference on Real-Time Systems (ECRTS), pp.309–320, 2012.
[5] H. Baek and J. Lee, "Incorporating security constraints into mixed-criticality real-time scheduling," IEICE Trans. Inf. & Syst., vol.E100-D, no.9, pp.2068–2080, Sept. 2017.
[6] S. Cho, S.K. Lee, S. Ahn, and K.J. Lin, "Efficient real-time scheduling algorithms for multiprocessor systems," IEICE Trans. Commun., vol.E85-B, no.12, pp.2859–2867, Dec. 2002.
[7] T.P. Baker, M. Cirinei, and M. Bertogna, "EDZL scheduling analysis," Real-Time Syst., vol.40, no.3, pp.264–289, 2008.
[8] A. Mok, Fundamental design problems of distributed systems for the hard-real-time environment, Ph.D. thesis, Massachusetts Institute of Technology, 1983.
[9] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," Proc. IEEE Real-Time Systems Symposium (RTSS), pp.239–243, 2007.
[10] M. Bertogna and M. Cirinei, "Response-time analysis for globally scheduled symmetric multiprocessor platforms," Proc. IEEE Real-Time Systems Symposium (RTSS), pp.149–160, 2007.
[11] J. Lee, A. Easwaran, I. Shin, and I. Lee, "Zero-laxity based real-time multiprocessor scheduling," J. Syst. Softw., vol.84, no.12, pp.2324–2333, 2011.
[12] T.P. Baker, "Comparison of empirical success rates of global vs partitioned fixed-priority and EDF scheduling for hard real-time," Technical Report, TR-050601, Dept. of Computer Science, Florida State University, Tallahasee, 2005.
[13] M. Bertogna, M. Cirinei, and G. Lipari, "Schedulability analysis of global scheduling algorithms on multiprocessor platforms," IEEE Trans. Parallel Distrib. Syst., vol.20, no.4, pp.533–566, 2009.
[14] S. Baruah, A. Burns, and R. Davis, "Response-time analysis for mixed criticality systems," Proc. Real-Time Systems Symposium (RTSS), pp.34–43, 2011.
[15] S. Baruah, V. Bonifaci, G. D'Angelo, H.L.A. Marchetti-Spaccamela, S. van der Ster, and L. Stougie, "The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems," Proc. Euromicro Conference on Real-Time Systems (ECRTS), pp.145–154, 2012.
[16] S. Baruah, V. Bonifaci, G. D'Angelo, A. Marchetti-Spaccamela, S. van der Ster, and L. Stougie, "Mixed-criticality scheduling of sporadic task systems," Proc. 19th Annual European Symposium on Algorithms, pp.555–566, 2011.
[17] S. Baruah, H. Li, and L. Stougie, "Toward the design of certifiable mixed criticality systems," Proc. IEEE Real-Time Technology and Applications Symposium (RTAS), pp.13–22, 2010.

[18] H. Li and S. Baruah, "An algorithm for scheduling certifiable mixed-criticality sporadic task systems," Proc. IEEE Real-Time Systems Symposium (RTSS), pp.183–192, 2010.
[19] N. Guan, P. Ekberg, M. Stigge, and W. Yi, "Effective and efficient scheduling of certifiable mixed-criticality sporadic task systems," Proc. IEEE Real-Time Systems Symposium (RTSS), pp.13–23, 2011.
[20] H. Su and D. Zhu, "An elastic mixed-criticality task model and its scheduling algorithm," Proc. Design Automation and Test in Europe (DATE), pp.147–152, 2013.
[21] S. Baruah, "Schedulability analysis for a general model of mixed-criticality recurrent real-time tasks," Proc. IEEE Real-Time Systems Symposium (RTSS), pp.25–34, 2016.
[22] H. Su, D. Zhu, and D. Moss'e, "Scheduling algorithms for elastic mixed-criticality tasks in multicore systems," Proc. IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), pp.352–357, 2013.
[23] J. Lee, K. Phan, X. Gu, J. Lee, A. Easwaran, I. Shin, and I. Lee, "MC-fluid: Fluid model-based mixed-criticality scheduling on multiprocessors," Proc. IEEE Real-Time Systems Symposium (RTSS), pp.41–52, 2014.
[24] G. Liu, Y. Lu, S. Wang, and Z. Gu, "Partitioned multiprocessor scheduling of mixed-criticality parallel jobs," Proc. Embedded and Real-Time Computing Systems and Applications (RTCSA), pp.1–10, 2014.
[25] Y. Suzuki, T. Azumi, N. Nishio, and S. Kato, "HLBS: Heterogeneous laxity-based scheduling algorithm for DAG-based real-time computing," Proc. Cyber-Physical Systems, Networks, and Applications (CPSNA), pp.83–88, 2016.
[26] C. Leng, Y. Qiao, H. Wang, J. Liu, and X. Zhang, "A new utilization based admission control algorithm for aperiodic tasks with constant time complexity under EDF scheduling," Proc. Embedded and Real-Time Computing Systems and Applications (RTCSA), pp.338–341, 2013.
[27] Z. Zhang, P. Liu, L. Ju, and Z. Jia, "Energy efficient real-time task scheduling for embedded systems with hybrid main memory," Embedded and Real-Time Computing Systems and Applications (RTCSA), pp.1–10, 2014.
[28] A. Yamaguchi, Y. Nakamoto, K. Sato, Y. Watanabe, and H. Takada, "EDF-PStream: Earliest deadline first scheduling of preemptable data streams – Issues related to automotive applications," Proc. Embedded and Real-Time Computing Systems and Applications (RTCSA), pp.257–267, 2015.
[29] N. Jung and J. Lee, "Development of mixed-criticality EDZL algorithm for multiprocessors," Korea Information Science Society, pp.1579–1581, 2015.
[30] N. Jung and J. Lee, "Mixed-criticality EDZL scheduling analysis on multiprocessors considering scenarios before criticality transition," Korea Information Science Society, pp.1573–1575, 2016.

Namyong Jung is a M.S. student at Sungkyunkwan University, where he receives the B.S. degree in 2016. His research interests are timing guarantees of real-time embedded systems.



Hyeongboo Baek is a postdoctoral research fellow in Sungkyunkwan University (SKKU), South Korea. He received the B.S. degree in Computer Science and Engineering from Konkuk University, South Korea in 2010 and the M.S. and Ph.D. degrees in Computer Science from KAIST, South Korea in 2012 and 2016, respectively. His research interests include cyber-physical systems, real-time embedded systems and system security. He won the best paper award from the 33rd IEEE Real-Time Systems Symposium (RTSS) in 2012.

Donghyouk Lim received the B.S. and M.S. degrees in Computer Science from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea in 2003 and 2005, respectively. He is currently the CTO of RTST, Korea. He has been a member of research staff in Electronics and Telecommunications Research Institute (ETRI), Korea, until 2014. His research interests are system software platforms for real-time embedded systems.

Jinkyu Lee is an assistant professor in Department of Computer Science and Engineering, Sungkyunkwan University (SKKU), Republic of Korea, where he joined in 2014. He received the B.S., M.S., and Ph.D. degrees in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Republic of Korea, in 2004, 2006, and 2011, respectively. He has been a visiting scholar/research fellow in the Department of Electrical Engineering and Computer Science, University of Michigan, U.S.A. in 2011–2014. His research interests include system design and analysis with timing guarantees, QoS support, and resource management in real-time embedded systems, mobile systems, and cyber-physical systems. He won the best student paper award from the 17th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS) in 2011, and the Best Paper Award from the 33rd IEEE Real-Time Systems Symposium (RTSS) in 2012.