The Journal of Systems and Software 137 (2018) 36–49

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Multi-level contention-free policy for real-time multiprocessor scheduling

Hyeongboo Baek[a], Jinkyu Lee[a,*], Insik Shin[b]

[a] Sungkyunkwan University (SKKU), Suwon, South Korea
[b] Korea Advanced Institute of Science and Technology (KAIST), Deajeon, South Korea

ARTICLE INFO

Article history:
Received 14 July 2017
Revised 8 November 2017
Accepted 11 November 2017
Available online 13 November 2017

Keywords:
Real-time systems
Real-time multiprocessor scheduling
Schedulability analysis
Multi-level contention-free policy

ABSTRACT

The *contention-free* policy has received attention in real-time multiprocessor scheduling owing to its wide applicability and significant improvement in offline schedulability guarantees. Utilizing the notion of *contention-free* slots in which the number of active jobs is smaller than or equal to the number of processors, the policy improves the schedulability by offloading executions in contending time slots to contention-free ones. In this paper, we propose the multi-level contention-free policy by exploiting a new, generalized notion of multi-level contention-free slots. In a case study, we present how the multi-level contention-free policy is applied to EDF (Earliest Deadline First) scheduling and develop a schedulability test for EDF that adopts the new policy. Our evaluation results demonstrate that the multi-level contention-free policy significantly improves the schedulability by up to 4188% and 127%, compared to vanilla EDF and EDF adopting the existing contention-free policy, respectively.

1. Introduction

Timing constraints are critical to embedded computing systems, leading to extensive studies on real-time scheduling of a set of tasks each of which invokes a series of jobs (Brandenburg and Gül, 2016; Li et al., 2016; Guasque et al., 2016). Starting from Liu and Layland's seminal work (Liu and Layland, 1973), uniprocessor scheduling theories have been fully matured for both scheduling algorithms (which determine the order of job execution) and schedulability tests (which guarantee no job deadline miss under a target scheduling algorithm) with respect to both implicit- and constrained-deadline task models. That is, it has been proved that EDF (Earliest Deadline First) (Liu and Layland, 1973) is optimal, and many exact (i.e., necessary and sufficient) schedulability tests have been developed, e.g., for EDF (Liu and Layland, 1973) and RM (Rate Monotonic) (Audsley et al., 1991).

When it comes to multiprocessor platforms, however, such maturity has been limited to the implicit-deadline task model. A bunch of optimal scheduling algorithms such as P-Fair, ER-Fair, LL-REF, EKG, DP-Wrap, RUN, U-EDF and QPS have been developed (Anderson and Srinivasan, 2000; Cho et al., 2006; Andersson and Tovar, 2006; Levin et al., 2010; Nelissen et al., 2012; Massa et al., 2016; Regnier et al., 2011), which aim at much effectively reducing preemptions/migrations costs or accommodating new environments (e.g., supporting sporadic releases). Also, a semi-partitioned scheduling with the C=D heuristic (Burns et al., 2012) and its extension (Brandenburg and Gül, 2016) demonstrated their near-optimal schedulability performance (i.e., about 98% and exceeding 99% schedulable processor utilizations, respectively). However, above algorithms lose their (near-)optimality when a more general task model is considered (e.g., the sporadic constrained-deadline task model). Considering a well-known fact that optimal online multiprocessor scheduling of the sporadic constrained-deadline task model is impossible (Fisher et al., 2010), developing advanced heuristic scheduling algorithms that are applicable to a more general task model is a promising approach to obtain higher schedulability performance.

While a number of heuristic algorithms for a more general task model have been proposed, there has been another direction to develop prioritization policies that can be incorporated into and applied to most (if not all) existing scheduling algorithms (called base algorithms) to improve schedulability on multiprocessors effectively. Among such policies, the zero-laxity (ZL) policy (Baker et al., 2008; Lee et al., 2011b) and the contention-free (CF) policy (Lee et al., 2011a; 2014) have received considerable attentions due to their wide applicability (even to scheduling algorithms to be developed in the future) and significant schedulability improvement. Between these two policies, this paper focuses on the CF policy, owing to the potential for further schedulability improvement.

* Corresponding author.
E-mail addresses: hbbaek@skku.edu (H. Baek), jinkyu.lee@skku.edu (J. Lee), insik.shin@cs.kaist.ac.kr (I. Shin).

https://doi.org/10.1016/j.jss.2017.11.027
0164-1212/

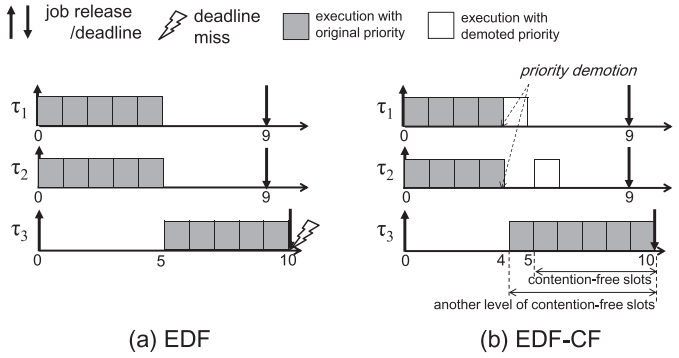


Fig. 1. Schedules under EDF and EDF-CF on a two-processor platform for a task set τ consisting of $\tau_1(T_1 = 15, C_1 = 5, D_1 = 9)$, $\tau_2(15, 5, 9)$, and $\tau_3(15, 6, 10)$.

The CF policy exploits the notion of a contention-free slot in which the number of active jobs (i.e., jobs with remaining execution time) is smaller than or equal to the number of processors; the most important property is that all active jobs in a contention-free slot are schedulable without any contention between them. The key technique to the CF policy is its offline calculation of the minimum number of contention-free slots that exists between each job's release and deadline; such a minimum number obtained by the offline calculation is independent of a base algorithm, thereby resulting in wide applicability of the CF policy. By utilizing the calculated minimum number of contention-free slots of a newly-released job, we trace the job's remaining minimum number of contention-free slots and remaining execution time at every time instant. If the former is larger than or equal to the latter, we can demote the job's priority without compromising schedulability since the remaining execution will be performed within its deadline regardless of its priority.

Fig. 1 depicts a scheduling example in which there are three jobs on two processors, i.e., three jobs of τ_1, τ_2, and τ_3, with execution times of 5, 5, and 6, release times of 0, 0, and 0, and absolute deadlines of 9, 9, and 10. When it comes to an algorithm adopting the CF policy, we can obtain the minimum number of contention-free slots of 1, 1, and 2, for jobs of τ_1, τ_2, and τ_3 respectively by offline calculation. We will detail how to calculate it for a general case in Section 4.4, but here is a high-level explanation. Between the release and absolute deadline of the job of τ_1 (i.e., within interval [0,9)), at most 5+5+6=16 executions can be performed regardless of its scheduling, and thus at least one contention-free slot exist since we assume a two-processor platform (i.e., $9 - \lfloor \frac{16}{2} \rfloor = 1$). The same holds for the job of τ_2, and the similar reasoning results in $10 - \lfloor \frac{16}{2} \rfloor = 2$ for the job of τ_3.

Whereas the job of τ_3 misses its deadline under EDF, as shown in Fig. 1(a), there is no job deadline miss under EDF-CF (i.e., EDF adopting the CF policy), as shown in Fig. 1(b). Since [0,4) is not contention-free, at $t = 4$, the minimum number of remaining contention-free slots of the jobs of τ_1 and τ_2 is still 1, and it is same as their remaining execution time (i.e., 1). This triggers a priority demotion of the jobs of τ_1 and τ_2 at $t = 4$, yielding no deadline miss for the job of τ_3 as well as jobs of τ_1 and τ_2 under EDF-CF.

Although the CF policy is successful in the previous example, what if the execution time of τ_3 is increased to 7 from 6? It is straightforward for EDF-CF to incur a deadline miss of τ_3's job, as shown in Fig. 3(a), and, therefore, it is necessary to make the priority demotion earlier than $t = 4$. For example, if the priority demotion occurs at $t = 2$, as shown in Fig. 3(b), there is no deadline miss, which will be explained in Section 4. An interesting phenomenon observed in Fig. 3(a) (likewise in Fig. 1(b)) is that jobs of τ_1 and τ_2 exploit only one contention-free slot although there are four contention-free slots up to their deadlines (i.e., in [5,9)]), which indicates the limitation of the existing CF policy. To advance the priority demotion without compromising the schedulability of jobs with a demoted priority after the demotion, we need to further exploit the undisclosed contention-free slots that the existing CF policy failed to utilize; this entails investigation of how jobs behave after the CF policy is applied.

In this paper, we propose the multi-level CF policy that generalizes the existing CF policy by effectively exploiting different levels of contention-free slots; the new policy can further improve the schedulability by demoting the priority of jobs earlier than what the existing CF policy does. To this end, we first reinterpret the existing CF policy, by developing new notions of contention-free/contending slots and normal/demoted jobs and their relations. We then develop the two-level CF policy using the notions and their relations, and generalize it to the N-level CF policy. Such generalization is quite sophisticated since the N-level CF policy handles N+1 different priority levels (unlike the existing CF policy simply handling two priority levels for demoted and undemoted jobs) and demotes the priority of a job N times (if necessary) during its execution from the highest-priority level to the lowest-priority one sequentially, not directly, by comparing its remaining execution time and remaining minimum numbers of N different levels of contention-free slots. Such sequential priority-demotion is the key technique for the N-level CF policy to work correctly (i.e., without compromising the schedulability of the base algorithm) since each level of contention-free slots allows only jobs with an equal- or higher-level priority to execute without any contention. A simple priority-demotion to the lowest priority does not guarantee the schedulability of a job, which indicates that generalizing the existing CF policy to the N-level one is not straightforward, which will be detailed in Section 4.

We also show how to perform an offline calculation of the number of multi-level contention-free slots that a newly-released job will experience until its deadline, and derive the important properties regarding the multi-level CF policy, both of which are essential parts for runtime operation and offline timing guarantees for the multi-level CF policy. In a case study, we apply the multi-level CF policy to EDF scheduling (as a popular global scheduling algorithm), and develop its schedulability test in order to show how much schedulability of base scheduling algorithms can be potentially improved by the multi-level CF policy. Our simulation results demonstrate that the multi-level CF policy significantly improves the schedulability by up to 4,188% and 127%, compared to vanilla EDF and EDF adopting the existing level CF policy, respectively.

In summary, this study provides the following four contributions:

- Reinterpretation of the existing CF policy by developing new notions of contention-free/contending slots and normal/demoted jobs and their relations (Section 3),
- Development of a multi-level CF policy using the notions of multi-level contention-free/contending slots and normal/demoted jobs, and their relations (Section 4),
- Application of the multi-level CF policy to EDF scheduling and development of a schedulability test thereof (Section 5.1), and
- Demonstration of the effectiveness of the multi-level CF policy in terms of schedulability performance and preemption overhead via simulations (Sections 5.2 and 5.3).

2. System model

In this paper, we consider a set τ of η sporadic real-time tasks (Mok, 1983) to be scheduled on m identical processors using a global preemptive work-conserving scheduling algorithm. A task

$\tau_i = (T_i, C_i, D_i)$ in a task set τ is specified by the minimum separation (or period) T_i, the worst-case execution time C_i, and the relative deadline D_i. Recognizing that an implicit-deadline task system (i.e., $D_i = T_i$) already has been studied extensively, we focus on a constrained-deadline task system in which $C_i \le D_i \le T_i$ holds for every task $\tau_i \in \tau$. Each task τ_i invokes a series of jobs, each of which is separated from its predecessor by at least T_i time units, and is supposed to finish its execution within D_i time units from its release. We assume quantum based time, and without loss of generality, a time unit describes a quantum length of 1; all task parameters are specified by the multiples of the quantum length. A single job is assumed to be unable to execute in parallel. A job is called active at t if it has a remaining execution time at t. As the CF policy can be incorporated into most (if not all) existing preemptive work-conserving algorithms, we let A-CF denote the algorithm A that adopts the CF policy. For ease of presentation, we refer to the existing CF policy (Lee et al., 2011a; 2014) as the one-level CF policy.

3. One-level CF policy: reinterpretation

Building on our own reinterpretation, this section recapitulates the existing one-level CF policy (Lee et al., 2011a; 2014). Our reinterpretation entails new notions and their relations, which will be used for developing the multi-level CF policy in Section 4.

The principle of the one-level CF policy is to demote the priority of a job if we can guarantee the job to complete its remaining execution before its deadline; such a guarantee is supported by the notion of a contention-free slot in which the number of active jobs is no larger than the number of processors, yielding no contention among the active jobs in the slot. That is, a newly released job calculates how many contention-free slots exist until its deadline (which is presented in Lee et al., 2011a; 2014 and in Section 4.4 of this paper), and the job traces its remaining contention-free slots and execution time at each time slot. If the remaining execution time is not larger than the remaining contention-free slots, the job is reassigned to the lowest priority since the remaining execution even with the lowest priority can still be successfully performed in the remaining contention-free slots. This allows the jobs with original priorities to avoid contending with the jobs with demoted priorities, yielding a schedulability improvement.

To systematically analyze the one-level CF policy, we introduce the notions of time slots and jobs, derive their relations, and then explain the one-level CF policy by using the notions and their relations.

The notions and their relations newly developed in this section will be the basis for developing the multi-level CF policy.

Notions. Under the principle of the one-level CF policy, a time slot is categorized into two types, depending on the possibility of contention in the slot, as follows:

Definition 1. (From Lee et al., 2011a; 2014) A time slot is called *one-level contention-free* if the number of active jobs in the slot is less than or equal to the number of processors ($\mathtt{m}$).

Otherwise, the time slot is called *one-level contending*.

The one-level CF policy demotes the priority of an active job if the job's remaining execution time is not larger than the number of remaining one-level contention-free slots. Therefore, active jobs scheduled by A-CF (i.e., the base algorithm A adapting the one-level CF policy) are separated into two types: active jobs with their initial priority assigned by a base algorithm A, and those with a demoted priority assigned by the one-level CF policy, which are formally defined as follows:

Table 1
$\mathtt{N}$-level CF relations ($\mathtt{N} \ge 1$).

Expression	Description
$S = S_c^{\mathtt{N}} \uplus S_f^{\mathtt{N}}$	$S_c^{\mathtt{N}}$ and $S_f^{\mathtt{N}}$ are disjoint, and S is a union of $S_c^{\mathtt{N}}$ and $S_f^{\mathtt{N}}$.
$J_n^{\mathtt{N}-1}(t) = J_n^{\mathtt{N}}(t) \uplus J_d^{\mathtt{N}}(t)$	$J_n^{\mathtt{N}}(t)$ and $J_d^{\mathtt{N}}(t)$ are disjoint, and $J_n^{\mathtt{N}-1}(t)$ is a union of $J_n^{\mathtt{N}}(t)$ and $J_d^{\mathtt{N}}(t)$.
$J_n^{\mathtt{N}}(t) \overset{p}{\succ} J_d^{\mathtt{N}}(t)$	$J_n^{\mathtt{N}}(t)$ is a set of active jobs having priorities higher than any job in $J_d^{\mathtt{N}}(t)$
$\left(J_n^{\mathtt{N}-1}(t) = J_n^{\mathtt{N}}(t) \uplus J_d^{\mathtt{N}}(t)\right) \overrightarrow{\mathrm{F}}\, S_f^{\mathtt{N}}$	Active jobs in $(J_n^{\mathtt{N}}(t) \uplus J_d^{\mathtt{N}}(t))$ are contention-free in time slots in $S_f^{\mathtt{N}}$

Definition 2. An active job is called a *one-level demoted job* if its priority is demoted to the lowest priority by the one-level CF policy. Otherwise, the active job is called *one-level normal job*.

Relations. Since such notions of time slots and active jobs for the one-level CF policy are closely related to each other, we can derive the relations between them. We let S denote a set of all time slots during the scheduling of A-CF, and we denote by $S_c^1 \subseteq S$ a set of one-level contending slots and by $S_f^1 \subseteq S$ a set of one-level contention-free slots. By definition, S_c^1 and S_f^1 are disjoint (i.e., $S_c^1 \bigcap S_f^1 = \varnothing$), and S is a union of S_c^1 and S_f^1 (i.e., $S = S_c^1 \bigcup S_f^1$), expressed as follows:

- $S = S_c^1 \uplus S_f^1$.

Then, we let $J(t)$ denote a set of active jobs at time instant t, and we denote by $J_n^1(t)$ a set of one-level normal jobs and by $J_d^1(t)$ a set of one-level demoted jobs at time instant t, respectively. For ease of presentation (which is actually needed for the multi-level CF policy), we let $J_n^0(t)$ denote $J(t)$, meaning that we express all active jobs at t as 0-level normal jobs at t. By definition, $J_n^1(t)$ and $J_d^1(t)$ are disjoint (i.e., $J_n^1(t) \bigcap J_d^1(t) = \varnothing$), and $J_n^0(t)$ is a union of $J_n^1(t)$ and $J_d^1(t)$ (i.e., $J_n^0(t) = J_n^1(t) \bigcup J_d^1(t)$), expressed as follows:

- $J_n^0(t) = J_n^1(t) \uplus J_d^1(t)$.

Since $J_d^1(t)$ represents a set of active jobs with a demoted priority, active jobs in $J_n^1(t)$ have a higher priority than that of any active job in $J_d^1(t)$, expressed as follows:

- $J_n^1(t) \overset{p}{\succ} J_d^1(t)$.

By the definition of the one-level contention-free slot, all active jobs at time instant t (i.e., jobs in $J_n^0(t) = J_n^1(t) \uplus J_d^1(t)$) can execute without any contention if [t, t+1) belongs to S_f^1, and we express this relation as follows:

- $\left(J_n^0(t) = J_n^1(t) \uplus J_d^1(t)\right) \overrightarrow{\mathrm{F}}\, S_f^1$.

We can see the expression and description of the four one-level CF relations in Table 1 by applying $\mathtt{N} = 1$, which are visualized in the upper part of Fig. 2.

One-level CF Policy. We formally present how the one-level CF policy operates using the notions and their relations explained above. The one-level CF policy manages two different priority queues, namely, Q^1 and Q^0, that will contain the jobs in $J_n^1(t)$ and $J_d^1(t)$, respectively. Let $C_i(t)$ denote the remaining execution time of a job $\boldsymbol{j}_i$ of a task τ_i at time instant t. Moreover, let Φ_i^1 and $\Phi_i^1(t)$ denote the number of time slots belonging to S_f^1 that exist between the release time and the deadline of $\boldsymbol{j}_i$, and that between t and the deadline of $\boldsymbol{j}_i$ (where t is between the release time and the deadline), respectively.

A-CF runs the following six rules sequentially at each time instant t.

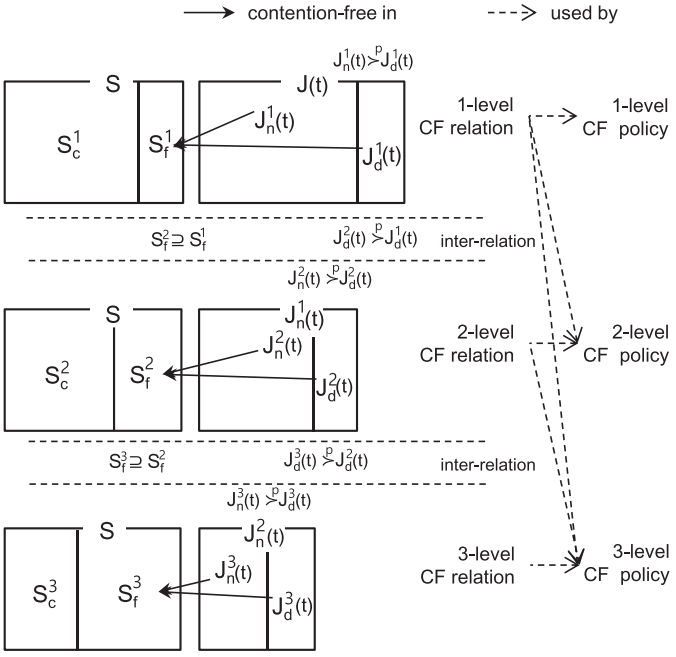


Fig. 2. Visualization of the N-level CF relations (N=1, 2, and 3).

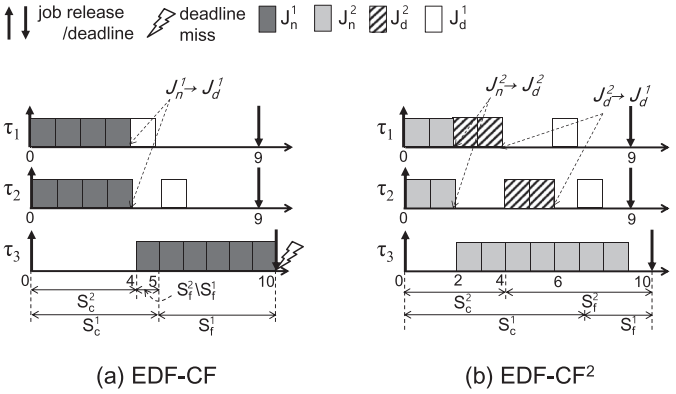


Fig. 3. Schedules under EDF-CF and EDF-CF² on a two processor platform for a task set τ consisting of $\tau_1(T_1 = 15,\ C_1 = 5,\ D_1 = 9)$, $\tau_2(15, 5, 9)$, and $\tau_3(15, 7, 10)$; here, $\Phi_1^2 = \Phi_2^2 = 3$, $\Phi_3^2 = 4$, $\Phi_1^1 = \Phi_2^1 = 1$, and $\Phi_3^1 = 2$, which will be calculated by Lemma 2 in Section 4.4.

- If a job $\boldsymbol{j}_i$ of a task τ_i is released,

  *R*1. Put $\boldsymbol{j}_i$ into Q^1, with its own priority given by the base algorithm A.
  *R*2. Calculate Φ_i^1 of $\boldsymbol{j}_i$ and set $\Phi_i^1(t) \leftarrow \Phi_i^1$ and $C_i(t) \leftarrow C_i$.

- For every job $\boldsymbol{j}_i$ in Q^1,

  *R*3. If $C_i(t) \le \Phi_i^1(t)$ holds, move the corresponding job from Q^1 to Q^0.
  *R*4. If the current time slot belongs to S_f^1, reduce $\Phi_i^1(t)$ by 1.

- Then,

  *R*5. Prioritize the jobs in Q^1 and Q^0 separately according to the base algorithm A and execute the (at most) m highest-priority jobs, considering that every job in Q^1 has a higher priority than that of every job in Q^0.
  *R*6. Update $C_i(t) \leftarrow C_i(t) - 1$ for the (at most) m selected jobs, and remove each job from its queue if the job has no remaining execution time.

Note that once a job is moved from Q^1 to Q^0 by *R*3, we can guarantee the job's schedulability by the relation $\left(J_n^0(t) = J_n^1(t) \uplus J_d^1(t)\right) \overrightarrow{F}\, S_f^1$ (i.e., no contention for the remaining execution time). By moving some jobs from Q^1 to Q^0, the algorithm ensures that the other jobs in Q^1 execute with relatively higher priorities in the time slots belonging to S_c^1, thereby improving the schedulability.

4. Multi-level CF policy

Generalizing the existing one-level CF policy, this section proposes the multi-level CF policy. To this end, we first develop the two-level CF policy by utilizing the new notions and their relations introduced in Section 3. We then generalize it to the multi-level CF policy on the basis of the ideas used for developing the two-level CF policy. Finally, we show how to calculate a lower bound of the number of contention-free slots that a job can have under different levels of CF policies and derive useful properties for the multi-level CF policy.

4.1. Two-level CF policy

Let us focus on a schedule under EDF-CF in Fig. 3(a). Let $\boldsymbol{j}_1$, $\boldsymbol{j}_2$, and $\boldsymbol{j}_3$ denote the jobs of three tasks τ_1, τ_2 and τ_3, respectively. The interval [0,5) and [5,10) belong to S_c^1 and S_f^1, respectively, because whether a time slot is the one-level contention-free or contending is determined by the number of all active jobs in the slot (i.e., $J_n^0(t)$). On the basis of tracing the number of remaining contention-free slots, the priorities of $\boldsymbol{j}_1$ and $\boldsymbol{j}_2$ are demoted at $t = 4$, which cannot avoid $\boldsymbol{j}_3$'s deadline miss at $t = 10$. To make the task set schedulable, we need to demote the priorities of $\boldsymbol{j}_1$ and $\boldsymbol{j}_2$ earlier than $t = 4$ (see, e.g., Fig. 3(b)). To this end, we pay attention to the jobs in $J_n^1(t)$ instead of those in $J_n^0(t)$ (i.e., one-level normal jobs instead of all jobs); once we define the contention-free/contending slots based on the basis of number of jobs in $J_n^1(t)$, we can advance the time for the priority demotion, which is a core idea of the two-level CF policy. In this subsection, we present the two-level CF policy, including how to guarantee that the remaining execution time of jobs having demoted priorities is finished within their deadline. We first present the notions of two-level contention-free/contending slots and normal/demoted jobs and their relations, as follows.

Notions. To derive a new type of contention-free slots, let us assume that our background policy is the one-level CF policy, which assigns the lowest priority to the demoted jobs (i.e., $J_d^1(t)$). Under the priority assignment policy, we focus on the jobs in $J_n^1(t)$ instead of those in $J_n^0(t)$. Since all jobs in $J_n^1(t)$ have a higher priority than that of all jobs in $J_d^1(t)$, all jobs in $J_n^1(t)$ will execute at t as long as the number of jobs in $J_n^1(t)$ is no larger than the number of processors (m), which yields another level of contention-free slots.

Definition 3. A time slot is called *two-level contention-free* if the number of one-level normal jobs in the slot is less than or equal to the number of processors (m). Otherwise, the time slot is called *two-level contending*.

Note that the definition does not care about the one-level demoted jobs (i.e., $J_n^0(t) \setminus J_n^1(t) = J_d^1(t)$) because we apply the demotion strategy of *the one-level CF policy*, which guarantees the schedulability of jobs in $J_d^1(t)$. This implies that the two-level CF policy should employ the demotion strategy of the one-level CF policy, to utilize the notion of two-level contention-free slots; that is, it moves a job to *the lowest-priority queue* if the number of remaining one-level contention-free slots until its deadline is no smaller than its remaining execution time. In addition to the its demotion strategy, the two-level CF policy also moves a

job to *the lower-priority queue* if the number of remaining two-level contention-free slots until its deadline is no smaller than its remaining execution time. In other words, whereas the one-level CF policy maintains two queue, i.e., Q^1 and Q^0, to distinguish jobs with undemoted and demoted priorities, the two-level policy maintains three queues, i.e., Q^2, Q^1, and Q^0, to distinguish jobs with undemoted priorities, with priorities demoted by tracing two-level contention-free slots (the lower-priority queue), and with priorities demoted by tracing one-level contention-free slots (the lowest-priority queue). Then, we can formally express jobs with priorities demoted by tracing two-level contention-free slots, as follows:

Definition 4. A one-level normal job is called a *two-level demoted job* if its priority is demoted by the two-level CF policy to the lowest priority among the one-level normal jobs, but is higher than that of any one-level demoted job. Otherwise, the one-level normal job is called a *two-level normal job*.

Note that the two-level CF policy utilizes not only Definitions 3 and 4, but also Definitions 1 and 2 derived for the one-level CF policy.

Relations. Like those of the one-level CF policy, the new notions of the two-level CF policy are also closely related to each other, which we will discuss now. Let S_c^2 and S_f^2 denote a set of two-level contending slots and contention-free slots, respectively, which exist during the scheduling of A-CF^2 (i.e., a base algorithm A adopting the two-level CF policy), and let $J_n^2(t)$ and $J_d^2(t)$ denote a set of two-level normal jobs and demoted jobs at time instant t, respectively. By Definitions 3 and 4, we can derive the relations of the two-level CF relation, as follows:

- $S = S_c^2 \uplus S_f^2$.
- $J_n^1(t) = J_n^2(t) \uplus J_d^2(t)$.
- $J_n^2(t) \overset{p}{\succ} J_d^2(t)$.
- $\left(J_n^1(t) = J_n^2(t) \uplus J_d^2(t)\right) \overrightarrow{\mathrm{F}} S_f^2$.

The meaning of the relations above is the same as that of the corresponding relations for the one-level CF policy (i.e., simply replacing 1 with 2), which are described in Table 1 with $\mathtt{N} = 2$, and are visualized in the middle part of Fig. 2.

Inter-relations. We now develop the relations between the one- and the two-level notions, which are illustrated in Fig. 2.

As illustrated in the upper and middle parts of Fig. 2, jobs in $J(t) = J_n^1(t) \uplus J_d^1(t)$ are contention-free in S_f^1 if [t, t+1) belongs to S_f^1 (i.e., $\left(J_n^0(t) = J_n^1(t) \uplus J_d^1(t)\right) \overrightarrow{\mathrm{F}} S_f^1$) whereas only jobs in $J_n^1(t)$ are contention-free in S_f^2 if [t, t+1) belongs to S_f^2 (i.e., $J_n^1(t) \overrightarrow{\mathrm{F}} S_f^2$). This indicates that S_f^2 includes not only the time slots in S_f^1, but also the time slots in which the jobs in $J_n^1(t)$ are contention-free but those in $J_d^1(t)$ are contending. Thus, we derive the following relation for the sets of one-level and two-level contention-free time slots.

- $S_f^2 \supseteq S_f^1$.

As for the jobs, $J(t)$ is partitioned into $J_n^1(t)$ and $J_d^1(t)$, and $J_n^1(t) \overset{p}{\succ} J_d^1(t)$ holds (shown in the upper part of Fig. 2). Likewise, $J_n^1(t)$ is partitioned into $J_n^2(t)$ and $J_d^2(t)$, and $J_n^2(t) \overset{p}{\succ} J_d^2(t)$ holds (shown in the middle part of Fig. 2), expressed as follows:

- $J_d^2(t) \overset{p}{\succ} J_d^1(t)$.

Two-level CF policy. We present how the two-level CF policy is conducted using both the one-level and the two-level CF relations, and the inter-relation between the one-level and the two-level notions. The two-level CF policy manages three queues, i.e., Q^2, Q^1, and Q^0, which will contain the jobs in $J_n^2(t)$, $J_d^2(t)$ and $J_d^1(t)$, respectively. In addition to Φ_i^1 and $\Phi_i^1(t)$ defined in Section 3, let Φ_i^2 and $\Phi_i^2(t)$ denote the number of time slots belonging to S_f^2 that exists between the release time and the deadline of a job $\boldsymbol{j}_i$ of a task τ_i, and that between t and its deadline (where t is between the release time and the deadline), respectively.

A-CF^2 runs the following rules sequentially at each time t, where a rule $\hat{R}x$ for the two-level CF policy corresponds to a rule Rx for the one-level CF policy in Section 3.

- If a job $\boldsymbol{j}_i$ of a task τ_i is released,

  $\hat{R}1$. Put $\boldsymbol{j}_i$ into Q^2, with its own priority given by the base algorithm A.
  $\hat{R}2$. Calculate Φ_i^1 and Φ_i^2 of $\boldsymbol{j}_i$, and set $\Phi_i^1(t) \leftarrow \Phi_i^1$, $\Phi_i^2(t) \leftarrow \Phi_i^2$, and $C_i(t) \leftarrow C_i$.

- For every job j_i in $Q^{\mathtt{x}}$, for $\mathtt{x}$ = from 2 to 1.

  $\hat{R}3$. If $C_i(t) \le \Phi_i^{\mathtt{x}}(t)$ holds, move the corresponding job from $Q^{\mathtt{x}}$ to $Q^{\mathtt{x}-1}$.

- For every job j_i in $Q^{\mathtt{x}}$, for $\mathtt{x}$ = from 2 to 1.

  $\hat{R}4$. If the current time slot belongs to $S_f^{\mathtt{y}}$, reduce $\Phi_i^{\mathtt{y}}(t)$ by 1, for $\mathtt{y}$ = from $\mathtt{x}$ to 1.

- Then,

  $\hat{R}5$. Prioritize the jobs in Q^2, Q^1, and Q^0 separately according to the base algorithm A and execute the (at most) $\mathtt{m}$ highest-priority jobs, considering that every job in Q^2 has a higher priority than that of every job in Q^1 and that every job in Q^1 has a higher priority than every job in Q^0.
  $\hat{R}6$. Update $C_i(t) \leftarrow C_i(t) - 1$ for the (at most) $\mathtt{m}$ selected jobs, and remove each job from its queue if the job has no remaining execution time.

Let us investigate how a job $\boldsymbol{j}_i$ of τ_i moves between queues without compromising the schedulability guarantee. Suppose that $\boldsymbol{j}_i$ in Q^2 satisfies $C_i(t') \le \Phi_i^2(t')$ at t'. This implies that $\boldsymbol{j}_i$ will never miss its deadline despite its demoted priority (but higher than that of jobs in $J_d^1(t')$) since the remaining execution of $\boldsymbol{j}_i$ will be successfully performed owing to the relation $\left(J_n^1(t) = J_n^2(t) \uplus J_d^2(t)\right) \overrightarrow{\mathrm{F}} S_f^2$ of the two-level CF relation. Therefore, we can move the job to Q^1 by $\hat{R}3$ without compromising the schedulability guarantee. Thereafter, suppose that $\boldsymbol{j}_i$ in Q^1 satisfies $C_i(t'') \le \Phi_i^1(t'')$ at t''. This also implies that $\boldsymbol{j}_i$ will never miss its deadline either despite the demoted priority (i.e., the lowest priority) owing to the relation $\left(J_n^0(t) = J_n^1(t) \uplus J_d^1(t)\right) \overrightarrow{\mathrm{F}} S_f^1$ of the one-level CF relation. Therefore, we can guarantee the schedulability of $\boldsymbol{j}_i$ despite moving to Q^0 by $\hat{R}3$.

Fig. 3 illustrates how the two-level CF policy is conducted to further improve the schedulability with an example task set. Let us consider a task set τ containing three tasks, $\tau_1 = \tau_2 = (15, 5, 9)$ and $\tau_3 = (15, 7, 10)$. We assume that $\Phi_1^2 = \Phi_2^2 = 3$, $\Phi_3^2 = 4$, $\Phi_1^1 = \Phi_2^1 = 1$ and $\Phi_3^1 = 2$, which will be calculated by Lemma 2 in Section 4.4. The high-level principle behind the offline calculation is as follows. Between the release and the deadline of $\boldsymbol{j}_1$ (likewise of $\boldsymbol{j}_2$), at most 5+5+7 = 17 executions can be performed, and thus at least one one-level contention-free slot exist since we consider a two-processor platform (i.e., $\Phi_1^1 = \Phi_2^1 = 9 - \lfloor \frac{17}{2} \rfloor = 1$). From the similar reasoning, we obtain $\Phi_3^1 = 10 - \lfloor \frac{17}{2} \rfloor = 2$. For calculating Φ_i^2, we can ignore Φ_i^1 amount of executions since jobs in $J_n^1(t) = J_n^2(t) \uplus J_d^2(t)$ are not interfered by jobs in $J_d^1(t)$ due to the

relation $J_n^1(t) \overset{p}{\succ} J_d^1(t)$. Thus, we obtain $\Phi_1^2 = \Phi_2^2 = 9 - \lfloor \frac{4+4+5}{2} \rfloor = 3$ and $\Phi_3^2 = 10 - \lfloor \frac{4+4+5}{2} \rfloor = 4$.

We also assume that τ is scheduled by EDF-CF and EDF-CF^2 on two processors, shown in Fig. 3(a) and (b), respectively. As seen in Fig. 3(a), $\boldsymbol{j}_3$ misses its deadline at time instant 10, although the priorities of $\boldsymbol{j}_1$ and $\boldsymbol{j}_2$ are demoted at time instant 4, where $\Phi_1^1(4) = \Phi_2^1(4) = 1$. In the case of Fig. 3(b), the priorities of $\boldsymbol{j}_1$ and $\boldsymbol{j}_2$ are demoted (i.e., moved from Q^2 to Q^1) at time instant 2, where $\Phi_1^2(2) = \Phi_2^2(2) = 3$, and $\boldsymbol{j}_3$ meets its deadline at time instant 9. Then, the priorities of $\boldsymbol{j}_1$ and $\boldsymbol{j}_2$ are further demoted (i.e., moved from Q^1 to Q^0) at time instants 4 and 6, respectively since $C_1(4) = \Phi_1^1(4)$ and $C_2(6) = \Phi_2^1(6)$ hold. Such a two-level priority demotion makes τ schedulable.

4.2. N-level CF policy

In this subsection, we generalize the two-level CF policy to the N-level CF policy using the notions and their relations derived for the one-level and the two-level CF policies.

In the previous subsection, we showed that the two-level CF policy is developed with new notions and their relations, which are derived by focusing on the jobs in $J_n^1(t)$. This idea is also applicable to the jobs in $J_n^2(t)$, meaning that we can develop a higher-level CF policy recursively, by focusing on higher-level normal jobs, which yields a further improvement in schedulability. For example, let us focus on an interval [0,4) belonging to S_c^2 in Fig. 3(b). We can partition it into two types of time slots: time slots in which active jobs in $J_n^2(t)$ can execute without any contention, i.e., [2,4), and time slots in which the jobs in $J_n^2(t)$ should contend for their execution, i.e., [0,2). Thus, we can define another level of contention-free slots and demoted jobs for the jobs in $J_n^2(t)$, which yields the three-level CF policy.

On the basis of such chaining reasoning, the N-level CF policy exploits N different levels of notions and their relations all together. We now present the notions of N-level ($\mathtt{N} \geq \mathtt{x} \geq 1$) time slots and jobs, and relations thereof.

Notions. By focusing on x-level normal jobs in $J_n^{\mathtt{x}}(t)$ (for $\mathtt{N} \geq \mathtt{x} \geq 1$) under the N-level CF policy, we can define x-level contention-free and contending slots as follows:

Definition 5. A time slot is called x-*level contention-free* if the number of $(\mathtt{x}-1)$-level normal jobs is less than or equal to the number of processors (m). Otherwise, the time slot is called x-*level contending*.

Note that Definition 5 is a generalization of Definitions 1 and 3. Moreover, we can define x-level demoted and normal jobs as follows:

Definition 6. An $(\mathtt{x}-1)$-level normal job is called an x-level demoted job if its priority is demoted by the x-level CF policy to the lowest priority among the $(\mathtt{x}-1)$-level normal jobs, but is higher than that of any $(\mathtt{x}-1)$-level demoted job. Otherwise, the $(\mathtt{x}-1)$-level normal job is called an x-level normal job.

Likewise, Definition 6 is also a generalization of Definitions 2 and 4.

Relations. Then, we present the relations between the notions of x-level time slots and jobs, referred to as x-level CF relations. We explain the relations by applying x to N in Table 1, and the three-level CF relation is visualized in the lower part of Fig. 2 as an example of x-level CF relation. Here, we let $S_c^{\mathtt{x}}$ and $S_f^{\mathtt{N}}$ denote a set of x-level contending and contention-free slots, respectively which exist during the scheduling of A-CF$^{\mathtt{N}}$ (i.e., a base algorithm A adopting the N-level CF policy), and $J_n^{\mathtt{x}}(t)$ and $J_d^{\mathtt{x}}(t)$ denote a set of x-level normal and demoted jobs at time instant t, respectively, where $1 \leq \mathtt{x} \leq \mathtt{N}$.

Inter-relations. Similar to the inter-relations between the one-level and the two-level notions, the following are the inter-relations between $(\mathtt{x}-1)$-level and x-level notions:

- $S_f^{\mathtt{x}} \supseteq S_f^{\mathtt{x}-1}$.
- $J_d^{\mathtt{x}}(t) \overset{p}{\succ} J_d^{\mathtt{x}-1}(t)$.

N-level CF policy. We then present how the N-level CF policy is conducted with N different levels of CF relations and inter-relations between N different levels of notions. The N-level CF policy manages (N+1) queues such that $Q^{\mathtt{N}}$ contains jobs in $J_n^{\mathtt{N}}(t)$ and the other queues $Q^{\mathtt{x}}$ contain jobs in $J_d^{\mathtt{x}+1}(t)$ for $0 \leq \mathtt{x} \leq \mathtt{N}-1$; the N-level CF policy also uses N different levels of contention-free slots. Let $\Phi_i^{\mathtt{x}}$ and $\Phi_i^{\mathtt{x}}(t)$ denote the number of time slots belonging to $S_f^{\mathtt{x}}$ that exist between the release time and the deadline of a job $\boldsymbol{j}_i$ of a task τ_i, and that between t and the deadline (where t is between the release time and the deadline), respectively.

Algorithm 1 describes the N-level CF policy with a base algorithm A. For each time slot, the N-level CF policy puts a job $\boldsymbol{j}_i$ of τ_i into $Q^{\mathtt{N}}$ when the job is released; then it sets N different levels of remaining contention-free slots $\Phi_i^{\mathtt{x}}(t)$ of $\boldsymbol{j}_i$ to $\Phi_i^{\mathtt{x}}$, respectively for $\mathtt{N} \geq \mathtt{x} \geq 1$; and it sets its remaining execution time $C_i(t)$ to C_i (Lines 1–3). For each queue $Q^{\mathtt{x}}$ ($\mathtt{N} \geq \mathtt{x} \geq 1$), the N-level CF policy moves a job in $Q^{\mathtt{x}}$ to $Q^{\mathtt{x}-1}$, and assigns the job a priority lower than that of any job in $Q^{\mathtt{x}}$ but higher than that of any job in $Q^{\mathtt{x}-2}$ (if any) if the number of remaining x-level contention-free slots $\Phi_i^{\mathtt{x}}(t)$ is greater than or equal to $C_i(t)$ (Lines 4–10). Then, if the current time slot belongs to x-level contention-free slots ($\mathtt{N} \geq \mathtt{x} \geq 1$), we decrease the number of remaining x-level contention-free slots of each job by 1 (Lines 11–15). Then, we prioritize the jobs in every $Q^{\mathtt{x}}$ for $\mathtt{N} \geq \mathtt{x} \geq 0$ separately, by the base algorithm A (Line 16), and choose (at most) m jobs to be executed, considering that all the jobs in $Q^{\mathtt{x}}$ have a higher priority than that of all jobs in $Q^{\mathtt{x}-1}$ for $\mathtt{N} \geq \mathtt{x} \geq 1$.

Algorithm 1 The N-level CF policy with the base algorithm A.

For each time slot,

1: **if** a job of τ_i is released **then**
2: Put the job into $Q^{\mathtt{N}}$ and set $\Phi_i^{\mathtt{x}}(t) \leftarrow \Phi_i^{\mathtt{x}}$ for all $\mathtt{N} \geq \mathtt{x} \geq 1$ and $C_i(t) \leftarrow C_i$.
3: **end if**
4: **for** x = from N to 1 decreasing by 1 **do**
5: **for** all jobs in $Q^{\mathtt{x}}$ **do**
6: **if** the job of τ_i satisfies $\Phi_i^{\mathtt{x}}(t) \geq C_i(t)$ **then**
7: Move the job to $Q^{\mathtt{x}-1}$.
8: **end if**
9: **end for**
10: **end for**
11: **for** x = from N to 1 decreasing by 1 **do**
12: **if** $\sum_{\mathtt{y}=\mathtt{x}-1}^{\mathtt{N}} |Q^{\mathtt{y}}| \leq m$ **then**
13: Update $\Phi_i^{\mathtt{x}}(t+1) \leftarrow \max\left(0, \Phi_i^{\mathtt{x}}(t) - 1\right)$ for all jobs inevery $Q^{\mathtt{y}}$ for all $\mathtt{N}+1 \geq \mathtt{y} \geq \mathtt{x}$.
14: **end if**
15: **end for**
16: Prioritize jobs in every $Q^{\mathtt{x}}$ for $\mathtt{N} \geq \mathtt{x} \geq 0$ separately, according to the base algorithm A.
17: Update $C_i(t) \leftarrow C_i(t) - 1$ for the (at most) m highest-priority jobs considering that all jobs in$Q^{\mathtt{x}}$ have a higher priority than that of all jobs in $Q^{\mathtt{x}-1}$, for $\mathtt{N} \geq \mathtt{x} \geq 1$, and remove each job from its queue if the job has no remaining execution time.

4.3. Theoretical computational overhead

We now discuss time complexity of EDF-CF$^{\mathtt{N}}$. Since the multi-level CF policy is the per time instant operation (as seen in

Algorithm 1), we investigate additional computations that should be conducted at every time instant when the multi-level CF policy is applied. Let f be the average job release/completion frequency (e.g., $f = 1/2$ means that a job release/completion occurs once every two time instants in average). Since EDF schedules jobs only when a job is released or finishes its execution, the time complexity of EDF is $\mathcal{O}(1/f)$. The N-level CF policy keeps track of the remaining execution time of jobs, $C_i(t)$, executing on the processors ($\mathcal{O}(\mathtt{m})$) and the remaining N different levels of contention-free slots, $\Phi_i^{\mathtt{N}}(t)$, for all active jobs ($\mathcal{O}(\mathtt{N}\cdot\eta)$), and compares $C_i(t)$ with $\Phi_i^{\mathtt{N}}(t)$ for every active job ($\mathcal{O}(\mathtt{N}\cdot\eta)$). If $\mathtt{m}\geq\eta$, then the system is trivially schedulable; thus we assume $\mathtt{m}<\eta$. Therefore, the time complexity of EDF-CF$^{\mathtt{N}}$ is $\mathcal{O}(1/f)+\mathcal{O}(\mathtt{m})+\mathcal{O}(\mathtt{N}\cdot\eta)+\mathcal{O}(\mathtt{N}\cdot\eta) = \mathcal{O}(1/f)+\mathcal{O}(\eta)+2\cdot\mathcal{O}(\mathtt{N}\cdot\eta) = \mathcal{O}(1/f)+\mathcal{O}(\mathtt{N}\cdot\eta)+2\cdot\mathcal{O}(\mathtt{N}\cdot\eta) = \mathcal{O}(1/f)+3\cdot\mathcal{O}(\mathtt{N}\cdot\eta) = \mathcal{O}(1/f)+\mathcal{O}(\mathtt{N}\cdot\eta)$. Considering the time complexity of EDF-CF is $\mathcal{O}(1/f)+\mathcal{O}(\eta)$, the time complexity of EDF-CF$^{\mathtt{N}}$ indicates that increasing level of the CF policy increases its computational overhead linearly, not exponentially.

We now address the number of additional migrations that can occur in EDF-CFN compared to EDF and EDF-CF. The migration of an active job conditionally happens when a job is preempted on one processor by a higher priority job and resumes its execution on another processor, which is conducted by the given scheduler. This implies that the number of migrations is upper-bounded by the number of preemptions, and thus, we need to consider how frequently each job is preempted by each scheduling algorithm. In the case of EDF, the number of preemptions is trivially upper-bounded by the number of jobs released during the system operation since the priority of each job is changed only when a job release/completion occurs, and a job completion cannot preempt an executing job. When it comes to EDF-CF and EDF-CF$^{\mathtt{N}}$, such a preemption can occur at most one and N times more, respectively, since EDF-CF and EDF-CF$^{\mathtt{N}}$ demote the priority of a job whose remaining execution is equal to the corresponding level of remaining contention-free slots. Therefore, for a given number of released jobs denoted by γ, the number of migrations (i.e., upper-bounded by the number of preemptions) is upper-bounded by $\mathcal{O}(\gamma)$, $\mathcal{O}(2\cdot\gamma)$ and $\mathcal{O}((\mathtt{N}+1)\cdot\gamma)$ for EDF, EDF-CF and EDF-CF$^{\mathtt{N}}$. Naturally, the cost of overall migration of the system with each scheduling algorithm is proportional to each upper-bound. Note that such derived maximum numbers of migrations are theoretical upper-bounds; we will demonstrate that the increasing average number of migrations under EDF-CF$^{\mathtt{N}}$ is marginal compared to EDF and EDF-CF (e.g., less than 2% increase) via empirical evaluation results in Section 5.3.

4.4. Lower-bound of the number of contention-free slots

We now explain how to calculate the lower-bound of the number of N different levels of contention-free slots, which a job experiences between its release time and its deadline under the N-level CF policy. Between the two mechanisms for calculating the lower-bound of the number of one-level contention-free slots proposed in Lee et al. (2011a, 2014), respectively, we extend the latter owing to its higher average schedulability performance. As $S = S_c^{\mathtt{x}} \uplus S_f^{\mathtt{x}}$ holds for $1\leq\mathtt{x}\leq\mathtt{N}$, we now calculate the lower-bound of the number of x-level contention-free slots that exist between the release time and the deadline of a job of a task τ_i by calculating the upper-bound of the number of x-level contending slots that exist within the same interval. To calculate the latter, we need to calculate how many executions of $(\mathtt{x}-1)$-level normal jobs are performed with contention according to Definition 5. Thus, we first investigate how many executions of $(\mathtt{x}-1)$-level normal jobs of a task τ_i can be performed in $(\mathtt{x}-1)$-level contending slots and then derive the upper-bound of the x-level contending slots on the basis of investigation result.

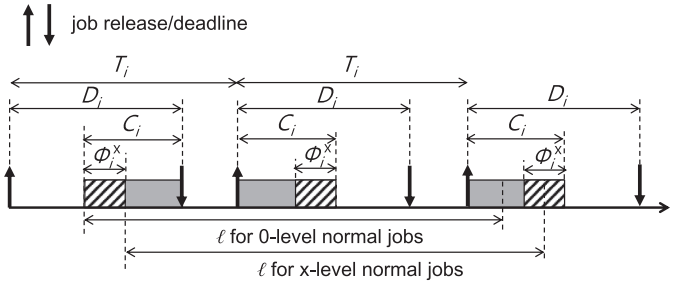


Fig. 4. Scenario maximizing the amount of executions performed by the x-level normal jobs of a task τ_i in x-level contending slots during the interval of interest of length ℓ.

Let $W_i^{\mathtt{x}}(\ell)$ denote the maximum amount of executions performed by the x-level normal jobs of τ_i in x-level contending slots for $1\leq\mathtt{x}\leq\mathtt{N}$, during an interval of length ℓ, and let $W_i^0(\ell)$ denote that performed by all jobs of τ_i during an interval of length ℓ. Fig. 4 depicts a scenario in which $W_i^0(\ell)$ is produced for a task τ_i. The first job of τ_i in Fig. 4 starts its execution at the left end of the interval of length ℓ and completes the execution at its deadline, which uses C_i time slots in the interval of length ℓ while it fully executes for its C_i. Thereafter, the following jobs are released and scheduled as soon as possible (Bertogna et al., 2009). As for $W_i^{\mathtt{x}}(\ell)$ for $1\leq\mathtt{x}\leq\mathtt{N}$, we can ignore the amount of $\Phi_i^{\mathtt{x}}$ executions out of C_i by using the following lemma.

Lemma 1. *An x-level normal job $\boldsymbol{j}_i$ executes in x-level contending slots during at most $C_i-\Phi_i^{\mathtt{x}}$ time instants.*

Proof. We consider the following two cases for $\boldsymbol{j}_i$:

Case 1. $\boldsymbol{j}_i$ does not migrate to $Q^{\mathtt{x}-1}$ (i.e., does not become a x-level demoted job) until its deadline: In this case, $\boldsymbol{j}_i$ satisfies $C_i(t) > \Phi_i^{\mathtt{x}}(t)$ until its deadline and faces at least $\Phi_i^{\mathtt{x}}$ x-level contention-free slots since there are at least $\Phi_i^{\mathtt{x}}$ x-level contention-free slots up to its deadline. Thus, at most $C_i-\Phi_i^{\mathtt{x}}$ executions are performed in x-level contending slots.

Case 2. $\boldsymbol{j}_i$ migrates from $Q^{\mathtt{x}}$ to $Q^{\mathtt{x}-1}$ (i.e., becomes an x-level demoted job) at time instant t' before its deadline: Before it migrates to $Q^{\mathtt{x}-1}$, $\boldsymbol{j}_i$ faces exactly $\Phi_i^{\mathtt{x}}-\Phi_i^{\mathtt{x}}(t')$ x-level contention-free slots. Thus, the number of x-level contending slots in which $\boldsymbol{j}_i$ executes is exactly $C_i-(\Phi_i^{\mathtt{x}}-\Phi_i^{\mathtt{x}}(t'))-\Phi_i^{\mathtt{x}}(t') = C_i-\Phi_i^{\mathtt{x}}$. □

As seen in Fig. 4, in the worst-case scenario for $W_i^{\mathtt{x}}(\ell)$, the interval of interest of length ℓ moves to the right by the amount of $\Phi_i^{\mathtt{x}}$, compared to $W_i^0(\ell)$.

We then let $n_i^{\mathtt{x}}(\ell)$ denote the number of x-level normal jobs of τ_i that can execute completely within the interval of interest (e.g., the first two jobs from the left end of the interval of length ℓ in Fig. 4) upper-bounded by

$$n_i^{\mathtt{x}}(\ell) = \left\lfloor \frac{\ell + D_i - C_i^{\mathtt{x}}}{T_i} \right\rfloor, \tag{1}$$

where $C_i^{\mathtt{x}} = \max\left(0, C_i-\Phi_i^{\mathtt{x}}\right)$ for $\mathtt{N}\geq\mathtt{x}\geq 1$, and $C_i^{\mathtt{x}} = C_i$ for $\mathtt{x}=0$. Then, the execution time of the last job of τ_i (e.g., the right-most jobs in Fig. 4) within the interval of length ℓ can be upper-bounded by $\min(C_i^{\mathtt{x}}, \ell+D_i-C_i^{\mathtt{x}}-n_i^{\mathtt{x}}(\ell)\cdot T_i)$. Therefore, $W_i^{\mathtt{x}}(\ell)$ is calculated by

$$W_i^{\mathtt{x}}(\ell) = \min\left(\ell, n_i^{\mathtt{x}}(\ell)\cdot C_i^{\mathtt{x}} + \min\left(C_i^{\mathtt{x}}, \ell+D_i-C_i^{\mathtt{x}}-n_i^{\mathtt{x}}(\ell)\cdot T_i\right)\right). \tag{2}$$

If a time slot is x-level contending, there are at least $\mathtt{m}-(\mathtt{x}-1)$-level normal jobs in the slot by Definition 5. Thus, we can calculate the upper-bounded number of x-level contending slots in an interval of length ℓ by $\left\lfloor \frac{\sum_{\tau_i\in\tau} W_i^{\mathtt{x}-1}(\ell)}{m} \right\rfloor$. Using the upper-bound of the

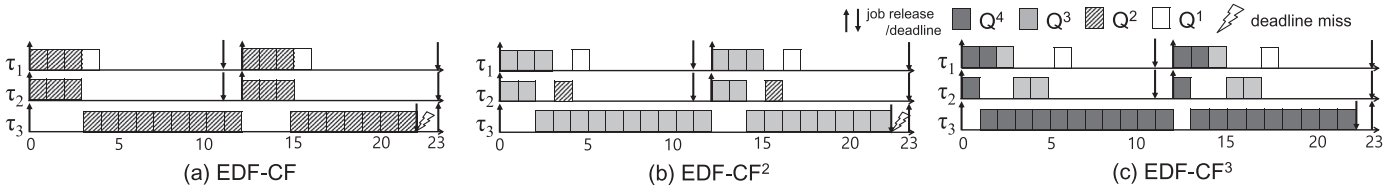


Fig. 5. Schedules of different levels of contention-free policy with an example task set.

number of x-level contending slots in an interval of length ℓ, we can calculate the lower-bound of the number of x-level contention-free slots as follows:

Lemma 2. *For a job of a task τ_k scheduled by A-CF$^{\mathtt{N}}$, there are at least $\Phi_k^{\mathtt{x}}$ x-level contention-free slots ($\mathtt{N} \geq \mathtt{x} \geq 1$) between the job's release time and its deadline, which are computed as follows:*

$$\Phi_k^{\mathtt{x}} = \max\left(0, D_k - \left\lfloor \frac{C_k^{\mathtt{x}-1} + \sum_{\tau_i \in \tau \setminus \{\tau_k\}} W_i^{\mathtt{x}-1}(D_k)}{m} \right\rfloor\right). \quad (3)$$

Proof. When we limit our attention to the interval between the release and the deadline of a job of a task τ_k, the lower-bound of the contention-free slots in the interval is derived by $D_k - \min\left(D_k, \left\lfloor \frac{\sum_{\tau_i \in \tau} W_i^{\mathtt{x}-1}(D_k)}{m} \right\rfloor\right)$. Here, we can reduce $W_k^{\mathtt{x}-1}(D_k)$ by replacing it with $C_k^{\mathtt{x}-1}$ with the observation that it is guaranteed that only a single job of a task τ_k is invoked (unlike the other jobs invoked by $\tau_i \in (\tau \setminus \tau_k)$) between the release time and the deadline of a job of a task τ_k. Thus, the lemma holds. □

4.5. Example

This section shows how the higher-level CF policy improves schedulability with an example. We consider EDF-CF, EDF-CF^2 and EDF-CF^3. Suppose that a task set $\tau = \{\tau_1 = (12, 4, 11),\ \tau_2 = (12, 3, 11),\ \tau_3 = (23, 20, 22)\}$ is scheduled by each algorithm on a two-processor platform, and all the tasks invoke their jobs periodically beginning at $t = 0$. We check whether there exists a deadline miss in the interval [0, 22). We denote the first and second released jobs of τ_1 (*likewise* τ_2) by $\boldsymbol{j}_1^1$ and $\boldsymbol{j}_1^2$ (*likewise* $\boldsymbol{j}_2^1$ and $\boldsymbol{j}_2^2$), respectively, and denote the first released job of τ_3 by $\boldsymbol{j}_3^1$.

- EDF-CF (Fig. 5(a)): At $t = 0$, Φ_i^1 is calculated by Eq. (3) in Section 4.4; $\Phi_1^1 = 11 - \lfloor \frac{4+6+11}{2} \rfloor = 1$, $\Phi_2^1 = 11 - \lfloor \frac{3+8+11}{2} \rfloor = 0$, and $\Phi_3^1 = 22 - \lfloor \frac{20+12+9}{2} \rfloor = 2$ hold. $\boldsymbol{j}_1^1$ and $\boldsymbol{j}_1^2$ migrate from Q^1 to Q^0 at $t = 3$ and $t = 14$ respectively. Unfortunately, such migrations do not reduce the interference on $\boldsymbol{j}_3^1$, yielding a deadline miss of $\boldsymbol{j}_3^1$ at $t = 22$.
- EDF-CF^2 (Fig. 5(b)): In addition to Φ_i^1, Φ_i^2 is calculated at $t = 0$; $\Phi_1^2 = \Phi_2^2$ and Φ_3^2 are calculated as $11 - \lfloor \frac{3+6+11}{2} \rfloor = 1$ and $22 - \lfloor \frac{18+9+9}{2} \rfloor = 4$, respectively. At $t = 2$ and $t = 13$, $\boldsymbol{j}_2^1$ and $\boldsymbol{j}_2^2$ migrate from Q^2 to Q^1, which reduces the interferences on $\boldsymbol{j}_3^1$ by 2, but it is not sufficient to make $\boldsymbol{j}_3^1$ schedulable.
- EDF-CF^3 (Fig. 5(c)): In addition to Φ_i^1 and Φ_i^2, Φ_i^3 is calculated at $t = 0$; Φ_1^3, Φ_2^3 and Φ_3^3 are calculated as $11 - \lfloor \frac{3+4+11}{2} \rfloor = 2$, $11 - \lfloor \frac{2+6+11}{2} \rfloor = 2$ and $22 - \lfloor \frac{16+9+6}{2} \rfloor = 7$, respectively. $\boldsymbol{j}_2^1$ and $\boldsymbol{j}_2^2$ move from Q^3 to Q^2 at $t = 1$ and $t = 12$, respectively, and $\boldsymbol{j}_1^1$ and $\boldsymbol{j}_1^2$ migrate from Q^3 to Q^2 at $t = 2$ and $t = 13$, respectively. $\boldsymbol{j}_3^1$ is now schedulable at $t = 22$.

Table 2 describes how each active job of the above example of EDF-CF^3 (Fig. 5(c)) behaves in each time slot in [0, 5) with changing values of $C_i(t)$ and $\Phi_i^{\mathtt{x}}$. The 1st row of Table 2 indicates the index of the time instant, and the 2nd to 5th rows indicate which jobs each queue contains. The 6th to 9th rows indicate the number of remaining multi-level contention-free slots up to the deadline of each job, and the 10th to 12th rows indicate the remaining execution time of each job.

With the values of $C_i(t)$ and $\Phi_i^{\mathtt{x}}$, three jobs, i.e., $\boldsymbol{j}_1^1$, $\boldsymbol{j}_2^1$, and $\boldsymbol{j}_3^1$ behave in each time slot as follows.

- At $t = 0$, $\boldsymbol{j}_1^1$, $\boldsymbol{j}_2^1$, and $\boldsymbol{j}_3^1$ are released and put into Q^3, and Φ_i^3, Φ_i^2, and Φ_i^1 are calculated. Then, $\boldsymbol{j}_1^1$ and $\boldsymbol{j}_2^1$ execute as they have deadlines earlier than that of $\boldsymbol{j}_3^1$.
- At $t = 1$, $\boldsymbol{j}_2^1$ moves to Q^2 as $C_2(1) = \Phi_2^3(1) = 2$, followed by $\boldsymbol{j}_1^1$ and $\boldsymbol{j}_3^1$. We mark the value corresponding to Φ_2^3 by "-" as no job uses the value after $t = 1$.
- At $t = 2$, $\boldsymbol{j}_1^1$ moves to Q^2 as $C_1(2) = \Phi_1^3(2) = 2$. $\boldsymbol{j}_3^1$ and $\boldsymbol{j}_1^1$ execute; for tie-breaking, we assume that an active job of lower task index has a higher priority when they have the same absolute deadlines and are in the same queue.
- At $t = 3$, $\boldsymbol{j}_1^1$ moves to Q^0 as $C_1(3) = \Phi_1^2(3) = \Phi_1^1(3) = 1$. $\boldsymbol{j}_3^1$ and $\boldsymbol{j}_2^1$ execute. Φ_3^3, Φ_3^2 and Φ_2^2 are reduced by 1 as the corresponding time slot belongs not only to the three-level contention-free slots, but also to the two-level contention-free slots. Then, $\boldsymbol{j}_2^1$ and $\boldsymbol{j}_3^1$ execute.
- At $t = 4$, Φ_3^3 and Φ_3^2 are reduced by 1, and $\boldsymbol{j}_2^1$ and $\boldsymbol{j}_3^1$ execute. $\boldsymbol{j}_2^1$ finishes its execution here.

4.6. Derivation of properties of the multi-level CF policy

As illustrated so far, as the level of the CF policy increases, the schedulability improves by demoting the priorities of jobs earlier than what the lower-level CF policy does. We show that such early priority demotion entails an important property—the interference reduction on a job that is scheduled by *A*-CF$^{\mathtt{N}}$. This property indicates that the amount of interference on a job can be reduced by the multi-level CF policy.

To prove the property, we derive the following lemma:

Lemma 3. *Once a job moves from $Q^{\mathtt{N}}$ to $Q^{\mathtt{N}-1}$ under A-CF$^{\mathtt{N}}$, the remaining execution time of the job will be finished before its deadline.*

Proof. A job $\boldsymbol{j}_i$ of a task τ_i scheduled by *A*-CF$^{\mathtt{N}}$ migrates from $Q^{\mathtt{N}}$ to $Q^{\mathtt{N}-1}$ at time instant t' only when remaining execution of $\boldsymbol{j}_i$ is less than or equal to the remaining N-level contention-free slots (i.e., $C_i(t') \leq \Phi_i^{\mathtt{N}}(t')$). Since jobs in $Q^{\mathtt{N}}$ and $Q^{\mathtt{N}-1}$ can execute without any contention in the N-level contention-free slots, $\boldsymbol{j}_i$ in $Q^{\mathtt{N}-1}$ has at least $C_i(t)$ time slots in which $\boldsymbol{j}_i$ can execute without any contention as long as it stays in $Q^{\mathtt{N}-1}$. Then, $\boldsymbol{j}_i$ migrates to $Q^{\mathtt{N}-2}$ at time instant t'' when $C_i(t'') \leq \Phi_i^{\mathtt{N}-1}(t'')$ or finishes its execution in $Q^{\mathtt{N}-1}$. Even if $\boldsymbol{j}_i$ migrates to $Q^{\mathtt{N}-2}$, there are at least $C_i(t'')$ time slots in which $\boldsymbol{j}_i$ can execute without any contention as long as it stays in $Q^{\mathtt{N}-2}$. On the basis of the chaining reasoning, $\boldsymbol{j}_i$ can finish its execution in $Q^{\mathtt{x}}$ before its deadline ($\mathtt{N}\text{-}1 \geq \mathtt{x} \geq 0$). Thus, this lemma holds. □

From the above lemma, we derive the following important theorem.

Table 2
Values for EDF-CF^3; empty cells indicate the corresponding queues contain no jobs at the corresponding time instant t, and "–" represents that the corresponding value will not be used from the corresponding time instant t.

t	0	1	2	3	4
Q^3	$\boldsymbol{j}_1^1$, $\boldsymbol{j}_2^1$, $\boldsymbol{j}_3^1$	$\boldsymbol{j}_1^1$, $\boldsymbol{j}_3^1$	$\boldsymbol{j}_3^1$	$\boldsymbol{j}_3^1$	$\boldsymbol{j}_3^1$
Q^2		$\boldsymbol{j}_2^1$	$\boldsymbol{j}_1^1$, $\boldsymbol{j}_2^1$	$\boldsymbol{j}_2^1$	$\boldsymbol{j}_2^1$
Q^1					
Q^0				$\boldsymbol{j}_1^1$	$\boldsymbol{j}_1^1$
	Φ^3, Φ^2, Φ^1	Φ^3, Φ^2, Φ^1	Φ^3, Φ^2, Φ^1	Φ^3, Φ^2, Φ^1	Φ^3, Φ^2, Φ^1
$\boldsymbol{j}_1^1$	2, 1, 1	2, 1, 1	–, 1, 1	–, –, –	–, –, –
$\boldsymbol{j}_2^1$	2, 1, 0	–, 1, 0	–, 1, 0	–, 0, 0	–, 0, 0
$\boldsymbol{j}_3^1$	7, 4, 2	7, 4, 2	7, 4, 2	6, 3, 2	5, 2, 2
$C_1(t)$	4 → 3	3 → 2	2 → 1	1	1
$C_2(t)$	3 → 2	2	2	2 → 1	1 → 0
$C_3(t)$	20	20 → 19	19 → 18	18 → 17	17 → 16

Theorem 1. *A job $\boldsymbol{j}_i$ of a task τ_i under A-CF^N can interfere with a single job $\boldsymbol{j}_k$ whose schedulability is not guaranteed, during at most $C_i - \Phi_i^N$ time slots.*

Proof. Lemma 3 indicates that a job $\boldsymbol{j}_k$ of τ_k is not guaranteed to be schedulable by the N-level CF policy only when $\boldsymbol{j}_k$ is in Q^N. By Lemma 1, a job $\boldsymbol{j}_i$ in Q^N can interfere with another job $\boldsymbol{j}_k$ in Q^N at most $C_i - \Phi_i^N$ since a job can interfere only in contending slots. Also, a job $\boldsymbol{j}_i$ in Q^x (N-1 ≥ x ≥ 0) cannot interfere with $\boldsymbol{j}_k$ in Q^N owing to its lower priority. Thus, the theorem holds.■ □

As A-CF does not make any task set to be schedulable by the base algorithm A unschedulable as presented in Lee et al. (2011a; 2014), such a dominance property also holds between A-CF^N and A-CF$^{(N-1)}$.

We prove the property by the following theorem.

Theorem 2. *If a task set τ is schedulable by a scheduling algorithm A-CF^{N-1}, then it is also schedulable by A-CF^N.*

Proof. As A-CF^N and A-CF^{N-1} schedule the same task set, $W_i^{N-1}(\ell) \le W_i^{N-2}(\ell)$ and $C_k^{N-1} \le C_k^{N-2}$ hold by the Lemma 1 and the definition of C_i^x. This makes $\Phi_k^N \ge \Phi_k^{N-1}$ according to Eq. (3). Since $S_f^x \supseteq S_f^{x-1}$ holds for every x-level contention-free slots, and active jobs in Q^N under A-CF^N and those in Q^{N-1} under A-CF^{N-1} are scheduled by the same algorithm A, when Φ_k^{N-1} is reduced by 1, Φ_k^N is also reduced by 1 by Algorithm 1. Thus, the priority of an active job $\boldsymbol{j}_i$ under A-CF^N is demoted no later than what the same job $\boldsymbol{j}_i$ under A-CF^{N-1} does.

Then, active jobs with a demoted priority under A-CF^{N-1} and A-CF^N are guaranteed to be schedulable respectively by Lemma 3 and cannot interfere with the active jobs in Q^N and Q^{N-1} respectively by the priority ordering policies of A-CF^N and A-CF^{N-1}.

Thus, the theorem holds. □

5. Case study: EDF-CF^N and its schedulability test

In this section, we apply the N-level CF policy to the global preemptive EDF and develop a new schedulability test for EDF-CF^N. Then, we evaluate the schedulability performance of EDF-CF^N with different value of N via simulation.

5.1. Schedulability tests for EDF-CF^N

Before we develop the schedulability test for EDF-CF^N, we first re-visit the EDF schedulability test presented in Bertogna et al. (2009). To test the schedulability of a given job of a task τ_k, this test checks whether the job has enough interference from other jobs to miss its deadline. Since calculating the exact interference is difficult, it uses the upper-bound of the interference

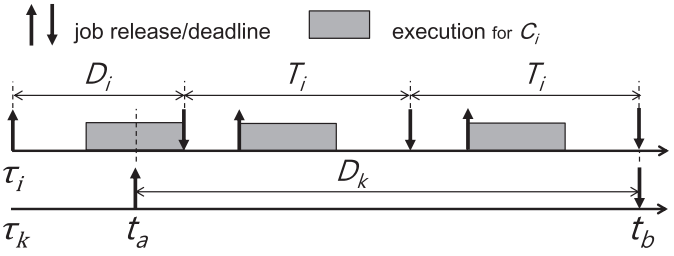


Fig. 6. Worst-case scenario wherein the interference of the jobs of τ_i on a job of τ_k under EDF is maximized.

on the basis of the worst-case release pattern illustrated in Fig. 6. As shown in Fig. 6, the interference from the jobs of τ_i to a job of τ_i is maximized when their deadlines are aligned since a job having a later absolute deadline cannot interfere with a job having an earlier absolute deadline. With this reasoning, the upper-bound of the amount of interference from the jobs of τ_i to a job of τ_k is calculated by $E(D_k, C_i, T_i)$ as follows:

$$E(D_k, C_i, T_i) = \left\lfloor \frac{D_k}{T_i} \right\rfloor C_i + \min\left(C_i, D_k - \left\lfloor \frac{D_k}{T_i} \right\rfloor \cdot T_i\right). \quad (4)$$

With $E(D_k, C_i, T_i)$, we can evaluate the schedulability of a given job of τ_k by using the EDF schedulability test as follows:

Lemma 4 (Theorem 7 in Bertogna et al., 2009)**.** *A task set τ is schedulable by EDF on an m-processor platform if the following inequality holds for every task $\tau_k \in \tau$.*

$$\sum_{\tau_i \in \tau - \{\tau_k\}} \min\left(E(D_k, C_i, T_i), D_k - C_k + 1\right) < m \cdot (D_k - C_k + 1). \quad (5)$$

Proof. We briefly summarize the proof of Bertogna et al. (2009) for their Theorem 7. To miss a deadline for a given job of τ_k scheduled on m processors, the job executes in at most $C_k - 1$ time slots. At each time slot, at least m other jobs are required to block the execution of a job of τ_k. Hence, at least $D_k - (C_k - 1)$ amount of interference is required to miss the job of τ_k. □

Through Theorem 1, we have shown that a job of a task τ_i can interfere with other jobs in at most $C_i - \Phi_i^N$ time slots as long as a base algorithm A follows the N-level CF policy. Thus, the EDF schedulability test presented in Lemma 4 can be modified for EDF-CF^N using a reduced execution by Φ_i^N as follows:

Theorem 3. *A task set τ is schedulable by EDF-CF^N on an m-processor platform, if the following inequality holds for every task $\tau_k \in \tau$.*

$$\sum_{\tau_i \in \tau - \{\tau_k\}} \min\left(E(D_k, C_i^N, T_i), D_k - C_k + 1\right) < m \cdot (D_k - C_k + 1), \quad (6)$$

Table 3
Schedulable ratio of EDF, EDF-CF and EDF-CF$^{\mathtt{N}}$ with $\mathtt{N} = 2, 3, 4$ and 5 for constrained-deadline task sets.

$\mathtt{m}$	EDF	EDF − CF	EDF − CF^2	EDF − CF^3	EDF − CF^4	EDF − CF^5	Ratio of EDF-CF^5 to EDF	Ratio of EDF-CF^5 to EDF-CF
2	9.7%	27.6%	36.7%	42.2%	45.7%	48.1%	495.8%	174.2%
4	4.6%	20.2%	28.8%	33.9%	37.4%	39.8%	865.2%	197.0%
8	2.1%	16.8%	25.1%	30.4%	33.7%	36.2%	1723.8%	215.4%
16	0.8%	15.1%	23.3%	28.4%	31.9%	34.3%	4287.5%	227.1%

where $C_i^{\mathtt{N}} = \max\left(0, C_i - \Phi_i^{\mathtt{N}}\right)$.

Proof. By Theorem 1 and Lemma 4, this theorem holds. □

5.2. Schedulability evaluation

We then evaluate the performance of the multi-level CF policy for constrained-deadline task sets. For our evaluation, we randomly generate 100,000 task sets using a popular task set generation technique proposed and used in Baker (2005), Bertogna et al. (2009) and Andersson et al. (2008). We consider the following two input parameters: the number of processors $m \in \{2, 4, 8, 16\}$, and the individual task utilization (C_i/T_i) distribution (bimodal or exponential with its input parameter p selected in {0.1, 0.3, 0.5, 0.7, 0.9} (Lee et al., 2014)). For a given bimodal parameter p, a value for C_i/T_i is uniformly selected in [0, 0.5) and [0.5, 1) with probability p and $1 - p$, respectively, and for a given exponential parameter $1/\lambda$, the value is selected according to the exponential distribution whose probability density function is $\lambda \cdot exp(-\lambda \cdot x)$. Each task has three parameters: T_i uniformly chosen from the interval $[1, T_{\max} = 1000]$, C_i chosen with the bimodal or exponential parameter, and D_i uniformly chosen from the interval $[C_i, T_i]$.

We use the schedulability ratio as the metric of performance, which is defined as the ratio of the number of task sets deemed schedulable by a schedulability test to the total number of generated task sets.

We consider the following schedulability tests:

- EDF: for the EDF scheduling algorithm (in Bertogna et al., 2009),
- EDF − CF: for the EDF-CF scheduling algorithm (in Lee et al., 2011a; 2014), and
- EDF − CF$^{\mathtt{N}}$: for the EDF-CF$^{\mathtt{N}}$ scheduling algorithms, where $\mathtt{N} \in \{2, 3, 4, 5\}$ (i.e., Theorem 3).

We first observe schedulable ratio of each schedulability test and compare existing techniques with the multi-level CF policy on different numbers of processors in with Table 3. We then investigate how schedulability of each schedulability analysis varies according to different task utilization distributions, in each of which a task set has different average number of tasks (denoted by $\overline{n}$) and a task has different average task utilization (denoted by $\overline{C_i/T_i}$) with Fig. 7. Among ten utilization models (bimodal and exponential utilization distributions with five input parameters), we consider bimodal distribution with 0.9 and exponential distributions with 0.1 and 0.9 since they present distinct $\overline{n}$ (the smallest, the largest and medium respectively) and $\overline{C_i/T_i}$ (the largest, the smallest and medium respectively). Fig. 7 shows schedulability test results over varying task utilization distributions for $\mathtt{m} = 2$ and 16. Each figure plots the number of tasks sets deemed schedulable by each schedulability test over varying task set utilization ($U_{sys} \triangleq \sum_{\tau_i \in \tau} C_i/T_i$). TOT represents the number of generated tasks with each task set utilization.

We have the following five main observations from Table 3 and Fig. 7.

*O*1. EDF performs very poorly on a multi-processor system, but EDF employing the multi-level CF policies significantly improves the schedulable ratio. Less than 10% and 1% of the task sets are deemed schedulable by EDF for $\mathtt{m} = 2$, and $\mathtt{m} = 16$, respectively.

*O*2. EDF − CF^5 improves the schedulable ratio of EDF by up to 4.96 times and even 42.88 times for $\mathtt{m} = 2$ and $\mathtt{m} = 16$ respectively.

*O*3. With increasing number of processors ($\mathtt{m}$) from 2 to 16, the schedulable ratio of EDF drops sharply (from 9.7% to 0.8%) whereas EDF with the (any-level) CF policies decrease at a much lower rate than that of EDF.

*O*4. EDF − CF^5 improves the schedulable ratio of EDF − CF with increasing rate as $\mathtt{m}$ increases from 2 (1.74 times) to 16 (2.27 times).

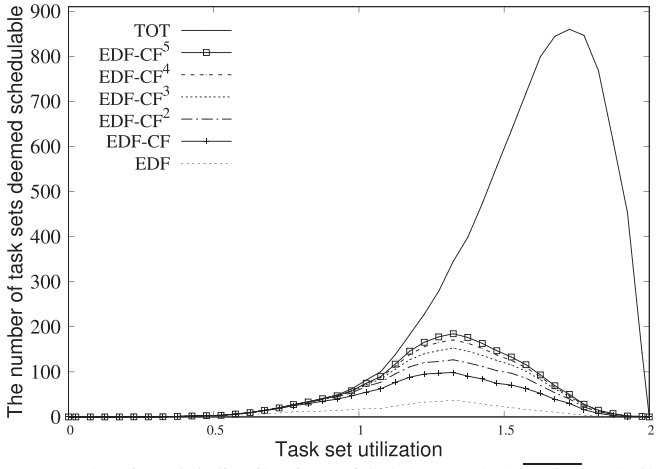
*O*5. For set of less CPU demanding tasks whose value of $\overline{C_i/T_i}$ is small (e.g., exponential distributions with 0.1 in Fig. 7(b) and (e)), EDF − CF^5 improves the schedulability of EDF even with a larger value of $\overline{n}$.

The first three observations, *O*1, *O*2 and *O*3, can be interpreted by the same reasoning. EDF performs very poorly on a multi-processor system owing to the well-known property of the EDF scheduling algorithm in that it only considers which jobs are urgent according to their absolute deadlines, thereby failing to utilize the sufficient computing power of multi-processors to make the task set schedulable (Lee et al., 2011b).

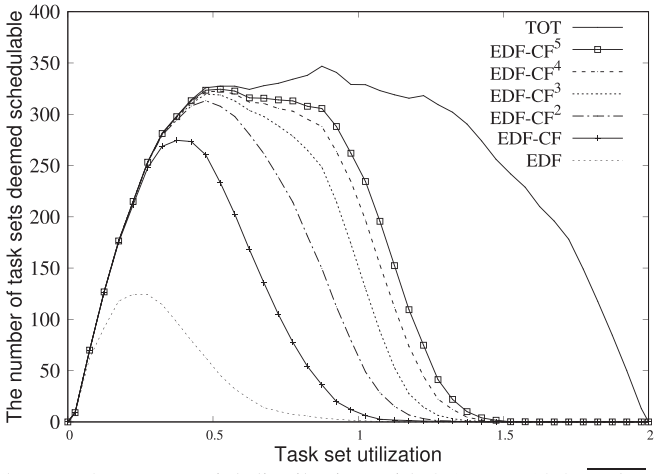
On the other hand, the higher-level CF policy allows the EDF scheduling algorithm to exploit a larger number of contention-free slots that were not used initially, while jobs, each of whose schedulability is guaranteed, yield their priorities in contending slots at runtime. Hence, the multi-level CF policy efficiently utilizes the computing power of a multi-processor system, which also results in less performance degradation on a large number of processors.

From observation *O*4, we infer that there is still a big room for improvement of the schedulability using higher-level contention-free slots that are not used in the one-level CF policy. As the number of processors increases, exploiting such unused time slots becomes much more crucial for improving the schedulability since the amount of computing power that a scheduling algorithm should utilize also increases. We believe that the multi-level CF policy effectively captures such a property, yielding a larger performance gap between EDF − CF and EDF − CF^5 with a larger number of processors.

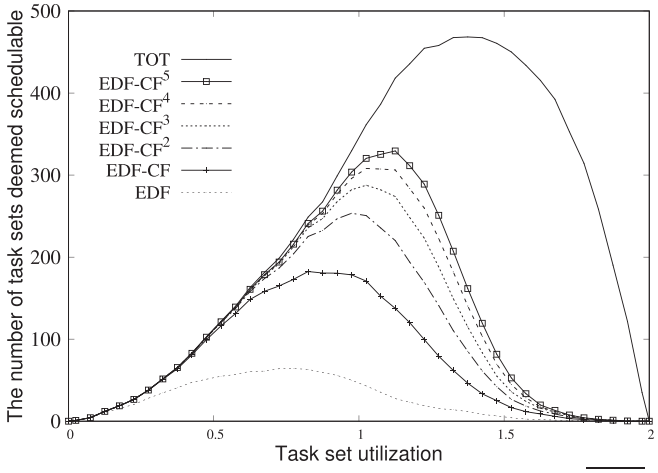
*O*5 is due to the virtue of the CF policy that it effectively mitigates pessimism of schedulability analysis test of EDF by demoting a large number of jobs' priorities so that such jobs impose less interference on the job of interest. Since EDF utilizes an upper-bound of interference from individual tasks (i.e., Eq. (2)), more tasks in a task set increases the pessimism in the interference bound, which more likely produces lower schedulability. In particular, such pessimism becomes apparent for a larger number of processors (e.g., $\mathtt{m} = 16$) since more tasks execute on the system as seen in Fig. 7(d), (e) and (f). The high level of CF policy well overcomes EDF's inherent limitation when it comes to task sets whose average task set utilizations are large since as task set utilization gets smaller, the number of contention-free slots of a job



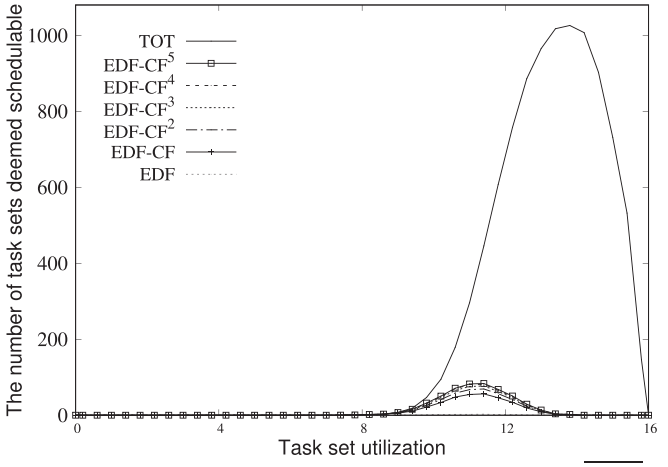
(a) $\mathtt{m} = 2$, Bimodal distribution with 0.9 ($\overline{n} = 3.1$, $\overline{C_i/T_i} = 0.52$)



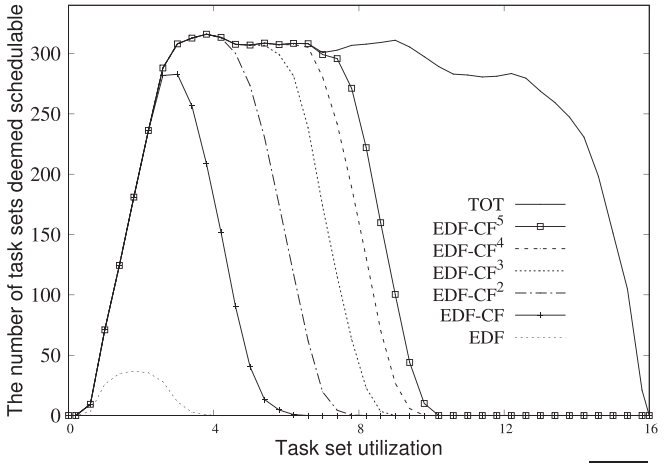
(b) $\mathtt{m} = 2$, Exponential distribution with 0.1 ($\overline{n} = 10.3$, $\overline{C_i/T_i} = 0.09$)



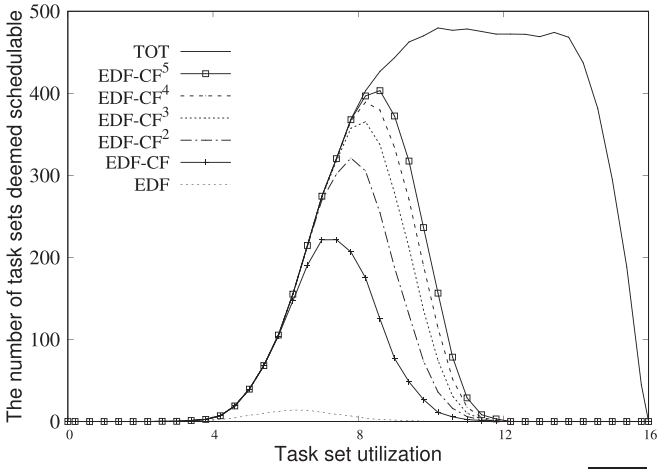
(c) $\mathtt{m} = 2$, Exponential distribution with 0.9 ($\overline{n} = 4.1$, $\overline{C_i/T_i} = 0.33$)



(d) $\mathtt{m} = 16$, Bimodal distribution with 0.9 ($\overline{n} = 19.4$, $\overline{C_i/T_i} = 0.69$)



(e) $\mathtt{m} = 16$, Exponential distribution with 0.1 ($\overline{n} = 80.0$, $\overline{C_i/T_i} = 0.1$)



(f) $\mathtt{m} = 16$, Exponential distribution with 0.9 ($\overline{n} = 27.3$, $\overline{C_i/T_i} = 0.4$)

Fig. 7. Schedulability tests for constrained deadline task sets.

Table 4
Average actual number of preemptions incurred by each constrained deadline task set during 100,000 time units.

m	EDF	EDF-CF	EDF-CF^5	EDF-CF^5 /EDF	EDF-CF^5 /EDF-CF
2	778.4	779.3	784.0	100.72%	101.60%
4	1070.2	1071.5	1074.8	100.43%	100.30%
8	1380.1	1380.9	1382.9	100.21%	100.14%
16	1560.6	1561.9	1563.8	100.20%	100.12%

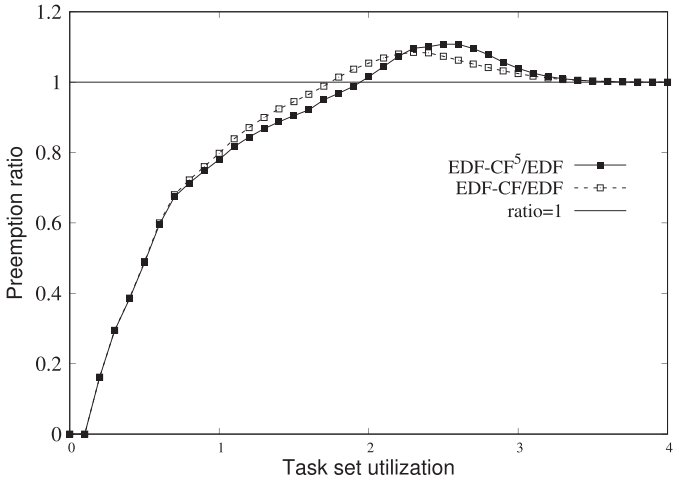


Fig. 8. Preemption ratio over varying task set utilization for $m = 4$.

gets larger. Hence, most tasks more likely become the lowest priority ones by the CF policy, and only few tasks' interference are considered to test the schedulability of a task of interest. For example, as seen in Fig. 7(e) and (f), EDF − CF^5 significantly improves performance of EDF even for the close to three times average number of tasks (80.0 vs. 27.3) owing to the lower average task utilization.

5.3. Preemption evaluation

One may wonder that a large number of additional preemption can occur with the multi-level CF policy due to the increased number of priority re-ordering at runtime. To address such an issue, we investigate how many preemptions occurs under EDF-CF^5 compared to EDF and EDF-CF in the following evaluation. Table 4 presents the average actual number of preemptions incurred by each constrained deadline task set during 100,000 time units under the three scheduling algorithms EDF, EDF-CF and EDF-CF^5. Comparing EDF-CF^5 with EDF and EDF-CF, we can see in the table that the number of preemptions does not increase significantly despite the five opportunities for each active job to change its priority during its execution; for $m = 2$ and $m = 16$, the percentage of increases are only about 0.72% and 0.2% respectively compared to EDF, and 1.6% and 0.12% respectively compared to EDF-CF.

Fig. 8 plots the ratio of the number of preemptions incurred by EDF-CF^5 (and EDF-CF), to that by EDF (referred to as preemption ratio) according to varying task set utilization. As seen in Fig. 8, preemption ratio of EDF-CF^5 (and EDF-CF) is much lower with the low utilization task, and it becomes higher than 1.0 only between task set utilizations around 2.0 and 3.0. As discussed by Lee et al. (2014), such trend of EDF-CF is a result of an active job having a number of contention-free slots that is higher than its execution time (i.e., $\Phi_i^1 \geq C_i$); when such jobs, which can be a majority, move to Q^0 upon their release, no preemption can incur. When it comes to the high utilization task set, active jobs incur similar numbers of preemptions to those under EDF since most active jobs remain in Q^1 owing to the insufficient number of contention-free slots to move to Q^0. Only active jobs in middle utilization task set incur visibly higher number of preemptions than EDF, but its increased number of preemptions is limited to 20% of that under EDF, thereby resulting in similar numbers of preemptions to those under EDF in total. A similar phenomenon happens in EDF that employs the multi-level CF policy; in the low utilization task set, most active jobs move from Q^N to Q^0 directly because of the large number of contention-free slots, whereas in high utilization task set, they remain in Q^N and show a similar schedule to that of EDF.

6. Conclusion

In this paper, we presented the multi-level CF policy as a generalization of the existing one-level CF policy, based on new notions of multi-level contention-free/contending slots and multi-level normal/demoted jobs. The policy significantly improves the schedulability performance, without compromising its applicability to existing scheduling algorithms and schedulability tests. As an example, we applied the multi-level CF policy to global EDF scheduling, demonstrating a schedulability improvement by up to 4188% and 127% over vanilla EDF and EDF adopting the existing one-level CF policy (Lee et al., 2014), respectively.

While we extended the level of the existing CF policy to multiple one, further extending the capability of the CF policy into the following two directions would be promising future works. The first direction is to accommodate parallel task models entailed by thread programmings such as fork-join, which captures intra-task thread-level parallelism (Chwa et al., 2016). The second direction is to develop a composition technique for schedulability analysis on multi-core platforms (Lee et al., 2016).

Acknowledgments

This work was supported in part by BSRP (NRF-2015R1D1A1A01058713; NRF-2016R1A6A3A11930688; NRF-2017R1A2B2002458), NRF(2017H1D8A2031628), NRF(2015M3A9A7067220), IITP(2015-0-00914, the National Program for Excellence in SW), IITP(2014-0-00065, Resilient Cyber-Physical Systems Research; 2017M3C4A7065925), and DAPA/ADD (High-Speed Vehicle Research Center of KAIST) funded by the Korea Government (MEST/MSIT/MOTIE).

References

Anderson, J.H., Srinivasan, A., 2000. Early-release fair scheduling. In: Proceedings of Euromicro Conference on Real-Time Systems, pp. 35–43.
Andersson, B., Bletsas, K., Baruah, S., 2008. Scheduling arbitrary-deadline sporadic task systems on multiprocessor. In: Proceedings of IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, pp. 197–206.
Andersson, B., Tovar, E., 2006. Multiprocessor scheduling with few preemptions. In: Proceedings of IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, pp. 322–334.
Audsley, N., Burns, A., Richardson, M., Wellings, A., 1991. Hard real-time scheduling: the deadline-monotonic approach. In: Proceedings of the IEEE Workshop on Real-Time Operating Systems and Software, pp. 133–137.
Baker, T.P., 2005. Comparison of Empirical Success Rates of Global vs. Partitioned Fixed-Priority EDF Scheduling for Hard Real-Time. Technical Report, TR–050601. Department of Computer Science, Florida State University, Tallahassee.
Baker, T.P., Cirinei, M., Bertogna, M., 2008. EDZL scheduling analysis. Real-Time Syst. 40 (3), 264–289.
Bertogna, M., Cirinei, M., Lipari, G., 2009. Schedulability analysis of global scheduling algorithms on multiprocessor platforms. IEEE Trans. Parallel Distrib. Syst. 20 (4), 553–566.
Brandenburg, B., Gül, M., 2016. Global scheduling not required: simple, near-optimal multiprocessor real-time scheduling with semi-partitioned reservations. In: Proceedings of IEEE Real-Time Systems Symposium, pp. 99–110.
Burns, A., Davis, R.I., Wang, P., Zhang, F., 2012. Partitioned edf scheduling for multiprocessors using a c=d task splitting scheme. Real-Time Syst. 48 (1), 3–33.
Cho, H., Ravindran, B., Jensen, E.D., 2006. An optimal real-time scheduling algorithm for multiprocessors. In: Proceedings of IEEE Real-Time Systems Symposium, pp. 101–110.

Chwa, H.S., Lee, J., Lee, J., Phan, K.-M., Easwaran, A., Shin, I., 2016. Global edf schedulability analysis for parallel tasks on multi-core platforms. IEEE Trans. Parallel Distrib. Syst. 28 (5), 1331–1345.
Fisher, N., Goossens, J., Baruah, S., 2010. Optimal online multiprocessor scheduling of sporadic real-time tasks is impossible. Real-Time Syst. 45, 26–71.
Guasque, A., Ballastre, P., Crespo, A., 2016. Real-time hierarchical systems with arbitrary scheduling at global level. J. Syst. Softw. 119 (5), 70–86.
Lee, J., Easwaran, A., Shin, I., 2011a. Maximizing contention-free executions in multiprocessor scheduling. In: Proceedings of IEEE Real-Time Technology and Applications Symposium, pp. 235–244.
Lee, J., Easwaran, A., Shin, I., Lee, I., 2011b. Zero-laxity based real-time multiprocessor scheduling. J. Syst. Softw. 84 (12), 2324–2333.
Lee, J., Easwaran, A., Shin, I., 2014. Contention-free executions for real-time multiprocessor scheduling. ACM Trans. Embedded Comput. Syst. 13 (69), 1–69.
Lee, J., Shin, K.G., Shin, I., Easwaran, A., 2016. Composition of schedulability analyses for real-time multiprocessor systems. IEEE Trans. Comput. 64 (4), 941–954.
Levin, G., Funk, S., Sadowski, C., Pye, I., Brandt, S., 2010. DP-FAIR: as simple model for understanding optimal multiprocessor scheduling. In: Proceedings of Euromicro Conference on Real-Time Systems, pp. 3–13.
Li, Z., Guo, C., Hua, X., Ren, S., 2016. Reliability guaranteed energy minimization on mixed-criticality systems. Journal of Systems and Software 112 (1), 1–10.
Liu, C., Layland, J., 1973. Scheduling algorithms for multi-programming in a hard-real-time environment. J. ACM 20 (1), 46–61.
Massa, E., Lima, G., Regnier, P., Levin, G., Brandt, S., 2016. Quasi-partitioned scheduling: optimality and adaptation in multiprocessor real-time systems. Real-Time Syst. 52 (5), 566–597.
Mok, A., 1983. Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment. Massachusetts Institute of Technology Ph.D. thesis.
Nelissen, G., Berten, V., Nelis, V., Goossens, J., Milojevic, D., 2012. U-EDF: An unfair but optimal multiprocessor scheduling algorithm for sporadic tasks. In: Proceedings of Euromicro Conference on Real-Time Systems, pp. 13–23.
Regnier, P., Lima, G., Massa, E., Levin, G., Brandt, S., 2011. RUN: Optimal multiprocessor real-time scheduling via reduction to uniprocessor. In: Proceedings of IEEE Real-Time Systems Symposium, pp. 104–115.

Hyeongboo Baek is a postdoctoral research fellow in Sungkyunkwan University (SKKU), South Korea. He received the BS degree in Computer Science and Engineering from Konkuk University, South Korea in 2010 and the MS and PhD 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.

Jinkyu Lee is an assistant professor in Department of Computer Science and Engineering, Sungkyunkwan University (SKKU), South Korea, where he joined in 2014. He received the BS, MS, and PhD degrees in computer science from the Korea Advanced Institute of Science and Technology (KAIST), South Korea, in 2004, 2006, and 2011, respectively. He has been a research fellow/visiting scholar in the Department of Electrical Engineering and Computer Science, University of Michigan until 2014. His research interests include system design and analysis with timing guarantees, QoS support, and resource management in real-time embedded 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.

Insik Shin received the BS degree from Korea University, the MS degree from Stanford University, and the PhD degree from the University of Pennsylvania all in computer science in 1994, 1998, and 2006, respectively. He is currently an associate professor in the Department of Computer Science at KAIST, South Korea, where he joined in 2008. He has been a postdoctoral research fellow at Malardalen University, Sweden, and a visiting scholar at the University of Illinois, Urbana-Champaign until 2008. His research interests include cyber-physical systems and real-time embedded systems. He is currently a member of the Editorial Board of Journal of Computing Science and Engineering. He has been cochair of various workshops including satellite workshops of RTSS, CPSWeek, and RTCSA and has served various program committees in real-time embedded systems, including RTSS, RTAS, ECRTS, and EMSOFT. He received best paper awards, including Best Paper Awards from RTSS in 2003 and 2012, Best Student Paper Award from RTAS in 2011, and Best Paper runner-ups at ECRTS and RTSS in 2008. He is a member of the IEEE.