Improvement of Real-Time Multi-Core Schedulability with Forced Non-Preemption

Jinkyu Lee, *Member, IEEE* and Kang G. Shin, *Life Fellow, IEEE*

Abstract—While tasks may be preemptive or non-preemptive (due to their transactional operations), deadline guarantees in multi-core systems have been made only for those task sets in each of which all tasks are preemptive or non-preemptive, not a mixture thereof, i.e., fully preemptive or fully non-preemptive. In this paper, we first develop a schedulability analysis framework that guarantees the timing requirements of a given task set in which a task can be either preemptive or non-preemptive in multi-core systems. We then apply this framework to the prioritization polices of EDF (earliest deadline first) and FP (fixed priority), yielding schedulability tests of mpn-EDF (Mixed Preemptive/Non-preemptive EDF) and mpn-FP, which are generalizations of corresponding fully-preemptive and non-preemptive algorithms, i.e., fp-EDF and np-EDF, and fp-FP and np-FP. In addition to their timing guarantees for any task set that consists of a mixture of preemptive and non-preemptive tasks, the tests outperform the existing schedulability tests of np-EDF and np-FP (i.e., special cases of mpn-EDF and mpn-FP). Using these tests, we also improve schedulability by developing an algorithm that optimally disallows preemption of a preemptive task under a certain assumption. We demonstrate via simulation that the algorithm finds up to 47.6 percent additional task sets that are schedulable with mpn-FP (likewise mpn-EDF), but not with fp-FP and np-FP (likewise fp-EDF and np-EDF).

Index Terms—Forced non-preemption, preemptive and non-preemptive tasks, multi-core systems, real-time systems, EDF (earliest deadline first), FP (fixed priority)

1 Introduction

As multi-core chips are increasingly used for embedded real-time applications due to their potential for high performance at low cost, there have been a number of real-time scheduling algorithms proposed for multi-core systems, which can be characterized by their prioritization and preemption policies. While the prioritization policy determines each task's priority, such as EDF (earliest deadline first) and FP (fixed priority) [2], the preemption policy decides on the degree of restriction to preemption, such as the non-preemptive policy that prohibits the preemption of a currently-executing task, and the fully-preemptive policy that always allows a higher-priority task to preempt a lower-priority executing task. In spite of the significant advance in scheduling theory to date, there still exists much room to improve real-time multi-core scheduling. For example, no scheduling algorithm can dominate existing ones for general periodic/sporadic tasks under both fully-preemptive and non-preemptive policies. Besides, most scheduling theories have focused on fully-preemptive scheduling.

Depending on the nature of tasks, some of them can be preempted at any time during their execution, while others should not be preempted due to their transactional operations, e.g., interrupts. Under the fully-preemptive (non-preemptive) policy, all tasks in a set are preemptive (non-preemptive), i.e., all or nothing. In contrast, we treat the preemption requirement of each task as a *variable*; it is safe[1] to execute a given preemptive task as if it were non-preemptive, but the converse is not. In order to support task sets in each of which tasks have different preemption requirements, or in order to improve schedulability by using the preemption requirement of each task as a *control knob*, more general preemption policies have been developed for real-time uniprocessor scheduling (see [3] for a survey). However, such general preemption policies have not yet been studied for real-time *multi-core* scheduling.

In this paper, we consider such general preemption policies, and focus on the following important questions in real-time multi-core scheduling.

Q1. How can we guarantee the timing requirements of a given set of preemptive and non-preemptive tasks?
Q2. Can we improve schedulability by executing some preemptive tasks non-preemptively, but not conversely? If so, how can we find the optimal "assignment" of non-preemptiveness to each preemptive task?

To address these questions, we first define the MPN (mixed preemptive/non-preemptive) policy, under which each task can be either preemptive or non-preemptive. This policy is a generalization of both fully-preemptive and non-preemptive policies. Then, we give a formal description of a generic mpn-$*$ scheduling algorithm that

• J. Lee is with Department of Computer Science and Engineering, Sungkyunkwan University, Suwon, Gyeonggi-Do, South Korea. E-mail: jinkyu.lee@skku.edu.
• K.G. Shin is with Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109-2121. E-mail: kgshin@eecs.umich.edu.

*Manuscript received 31 Jan. 2013; revised 14 Nov. 2013; accepted 12 Dec. 2013; date of publication 15 Jan. 2014; date of current version 21 Mar. 2014. Recommended for acceptance by M.E. Acacio.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.2297098*

1. The term "safe" means that it satisfies the task specification. For schedulability guarantee, we need to check the timely-correctness of the task.

employs the MPN preemption policy and the prioritization policy of $*$, which is also a generalization of both fp-$*$ (that adopts the fully-preemptive policy and $*$ as its preemption and prioritization policies) and np-$*$ (that adopts the non-preemptive policy and $*$).

To address Q1, we choose a popular schedulability analysis[2] [4] designed for scheduling algorithms under the fully-preemptive policy; the analysis computes how long it takes for a given preemptive task to finish its execution (called the *task response time*), by analyzing how the execution of a given preemptive task affects that of the other preemptive tasks under a given prioritization policy (e.g., EDF, FP). However, it is challenging to extend this analysis to mpn-$*$ algorithms that employ the MPN policy because we need to (i) develop a response time analysis framework for both preemptive and non-preemptive tasks, and (ii) analyze the effect of the execution of a preemptive/non-preemptive task on that of other preemptive/non-preemptive tasks (a total of four cases).

While the calculation of the response time of a preemptive task requires the information on the time when the last unit of its execution is finished, that of a non-preemptive task only needs to know when the *first unit* of its execution begins; once it begins, the remaining execution will be completed without interruption. Using these properties, we address (i), developing a schedulability analysis framework for *any* mpn-$*$ scheduling algorithm. To address (ii), we first identify some properties of the effect between the executions of two tasks, which also hold under *any* mpn-$*$ scheduling algorithm.

Then, we can develop the schedulability analysis of any mpn-$*$ scheduling algorithm using the response time analysis framework and the derived properties; in this paper, we demonstrate their applicability to mpn-EDF and mpn-FP, which adopt MPN as their preemption policy, and EDF and FP as their prioritization policies, respectively. By carefully analyzing how the prioritization policies EDF and FP affect the four cases of (ii), and incorporating them into the generic framework and properties, we finally develop schedulability tests of mpn-EDF and mpn-FP (simple and improved tests for both algorithms), which have the following three characteristics. First, they can guarantee the timing requirements of a given task set that consists of preemptive and non-preemptive tasks. Second, they are generalizations of existing response-time based schedulability tests of fp-EDF and fp-FP [4]. Finally, they outperform the existing schedulability tests of np-EDF [5], [6], [7] and np-FP [6], [8] when they deal with np-EDF and np-FP, special cases of mpn-EDF and mpn-FP, respectively.

As to Q2, we first investigate how (ii) varies if a given preemptive task is disallowed to be preempted. Based on the results of this investigation, we develop an algorithm that disallows preemption of each preemptive task under any mpn-$*$ scheduling, without investigating all possible preemption assignments. To demonstrate the effectiveness of the algorithm for better schedulability, we also choose mpn-EDF and mpn-FP scheduling algorithms with their schedulability tests we derived. We prove that the algorithm is "optimal" under our simple schedulability tests of mpn-EDF and mpn-FP. We then demonstrate via simulation that disallowing preemption of preemptive tasks is also effective even under the improved schedulability test of mpn-FP (likewise mpn-EDF) in that it finds up to 47.6 percent additional task sets which are schedulable with neither np-FP nor fp-FP (likewise neither np-EDF and fp-EDF).

In summary, this paper makes the following contributions:

- Introduction of a new preemption policy, MPN, which accommodates tasks with different preemption requirements, which is, to the best of our knowledge, the first attempt in the area of real-time multicore scheduling;
- Development of a schedulability analysis framework for any mpn-$*$ scheduling algorithm, and derivation of schedulability analyses of mpn-EDF and mpn-FP;
- Demonstration of the superior average schedulability of our analyses of np-EDF and np-FP (special cases of mpn-EDF and mpn-FP) over existing analysis techniques; and
- Development of an algorithm by using the schedulability analyses of mpn-EDF and mpn-FP to disallow preemption of preemptive tasks, and demonstration of its schedulability improvement over np-EDF and fp-EDF, and np-FP and fp-FP.

Compared to our earlier conference version [1], this paper has the following new technical contributions in addition to re-organizing and re-writing the entire paper for better and/or concise presentation.

- While the conference version is confined to application of the MPN policy to EDF only, this paper generalizes the scheduling algorithm, the schedulability analysis and the preemption-assignment algorithm for the generic mpn-$*$.
- As examples of the application of mpn-$*$, we demonstrate mpn-FP in addition to mpn-EDF.
- We include more evaluation results, including a comparison with another schedulability analysis of np-EDF [7] that was missing in [1].

The rest of the paper is organized as follows. Section 2 presents our system model, and recapitulates a schedulability analysis for fully-preemptive algorithms in [4]. Section 3 develops the MPN policy, and gives a formal description of mpn-$*$ algorithms. Section 4 develops a new schedulability analysis framework for mpn-$*$ algorithms, and performs schedulability analyses of mpn-EDF and mpn-FP. Section 5 presents an algorithm of disallowing preemption of preemptive tasks for better schedulability. Section 6 evaluates our schedulability analyses of mpn-EDF and mpn-FP, and the algorithm of disallowing preemption, via simulation. Section 7 summarizes the related work, and finally Section 8 concludes the paper.

2 Background

In this section, we first introduce the system model, assumptions and notations to be used throughout the paper.

2. A schedulability analysis in real-time systems determines whether all tasks meet their deadlines, by considering the worst case of all situations (e.g., release/execution patterns) to guarantee no deadline miss.

TABLE 1
Notations

Symbol	Description
m	the number of cores
Φ	task set
τ_i	task i
T_i	the minimum separation between two consecutive jobs of τ_i
C_i	the worst-case execution time of τ_i
D_i	the relative deadline of τ_i
Y_i	the preemption capability of τ_i
R_i	the response time of τ_i
$\mathsf{HP}(\tau_i)$	a set of higher-priority tasks than τ_i
$\mathsf{LP}(\tau_i)$	a set of lower-priority tasks than τ_i
$I_k(a,b)$	the interference to τ_k in $[a,b)$
$I_{k\leftarrow i}(a,b)$	τ_i's interference to τ_k in $[a,b)$
$I^*_{k\leftarrow i}(l)$	τ_i's maximum interference to τ_k in any interval of length l
$W_i(l)$	the maximum amount of execution of jobs of τ_i in any interval of length l
$E_{k\leftarrow i}$	the maximum amount of execution of jobs of τ_i with a no-later deadline than a job of τ_k in an interval between the release and the deadline of the τ_k's job

Then, we scrutinize an existing schedulability analysis technique for fully-preemptive scheduling algorithms in [4], which will be a basis for our schedulability analysis (Section 4) of the scheduling algorithms that employ the MPN preemption policy.

2.1 System Model, Assumptions and Notations

Here we focus on a sporadic task model [9] in which a task $\tau_i \in \Phi$ is represented as (T_i, C_i, D_i), where T_i is the minimum separation between two successive invocations, C_i the worst-case execution time, and D_i the relative deadline of τ_i. Our discussion is confined to implicit ($C_i \le D_i = T_i$) and constrained ($C_i \le D_i \le T_i$) deadline task sets.[3] A task τ_i invokes a series of jobs, each separated from its predecessor/successor by at least T_i time units. Without loss of generality, we assume a quantum-based time and let the length of a quantum be one time unit. All task parameters are specified in multiples of the quantum or time unit.

For each $\tau_i \in \mathcal{T}$, we introduce a new additional parameter Y_i, indicating whether τ_i is *preemptive* ($Y_i = 1$) or *non-preemptive* ($Y_i = 0$). That is, if $Y_i = 1$ ($Y_i = 0$), jobs of τ_i can (cannot) be preempted by any other higher-priority job at any time.

In this paper, we deal with two existing prioritization policies: EDF (earliest deadline first) scheduling that gives a higher priority to a job with an earlier deadline, and FP (fixed priority) scheduling in which the priority of a job is determined by the priority of the task that invokes the job. To express task-level fixed-priorities for FP, let $\mathsf{HP}(\tau_k)$ and $\mathsf{LP}(\tau_k)$ denote a set of tasks in Φ whose task-priority is higher and lower than τ_k, respectively. Table 1 summarizes the notations used throughout this paper.

The system is assumed to have been built with multi-core chips, each of which consists of m identical cores. We also focus on global work-conserving algorithms, i.e., a job can be executed on any core, and a core cannot be left idle if there is an unfinished ready job. For convenience, we will henceforth use the term "scheduling algorithm" to mean "global work-conserving scheduling algorithm." We also assume that a job cannot be executed in parallel.

We will use the terms *carry-in*, *body*, and *carry-out* jobs in a time interval of interest, defined as follows.

- A *carry-in* job is released before the interval, but its deadline is within the interval;
- A *body* job has its release time and deadline within the interval; and
- A *carry-out* job is released within the interval, but its deadline is after the interval.

2.2 An Existing Schedulability Analysis for Fully-Preemptive Scheduling Algorithms

As the basis for a schedulability analysis of algorithms that employ the MPN policy, we choose a response-time-based schedulability analysis technique for fully-preemptive scheduling algorithms [4] due to its applicability to various prioritization policies (e.g., fp-EDF, fp-FP and potentially more) and schedulability performance (e.g., the authors of [10] showed the test of fp-EDF [4] to be one of the best with respect to average schedulability). In the supplement, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.2297098, we summarize the existing schedulability analysis in [4].

3 MPN POLICY AND MPN-$*$ ALGORITHMS

In this section, we first define the MPN preemption policy, in which each task can be either (artificially) preemptive or non-preemptive. Then, we give a formal description of mpn-$*$ algorithms, in which MPN is employed with a prioritization policy of $*$ (e.g., EDF, FP); we present mpn-EDF and mpn-FP as typical examples. Finally, we present the generalization property of the MPN policy and mpn-$*$ algorithms.

We consider a preemption policy under which preemption decisions are made based on a task parameter Y_i of τ_i. That is, if $Y_i = 1$ ($Y_i = 0$), a job of τ_i can (cannot) be preempted by any other higher-priority job during its execution. We call this the *mixed preemptive/non-preemptive* (MPN) policy.

Let mpn-$*$ denote a scheduling algorithm that adopts MPN and $*$ as its preemption and prioritization policies, respectively. Algorithm 1 provides a formal description of an mpn-$*$ scheduling algorithm on a multi-core platform. Since there may be some non-preemptive tasks under mpn-$*$, Job Release Steps 4 and 5 exempt currently-executing non-preemptive jobs from being preempted by a newly-released job, which is a main difference between fp-$*$ (algorithms with the fully-preemptive policy) and mpn-$*$. The implementation overhead of mpn-$*$ is not significant in that most steps in Algorithm 1 are also required by the corresponding fully-preemptive algorithm fp-$*$. That is, mpn-$*$ additionally requires to check the condition of $Y_k = 1$ in Step 4 and the second "if" condition in Step 5, and to store one additional bit per task for Y_k. On the other hand, while all tasks under fp-$*$ can be preempted, only some tasks under mpn-$*$ can, thus

3. While the scheduling *algorithms* to be discussed in this paper can handle arbitrary deadline task sets (no restriction between D_i and T_i), the schedulability *analyses* can deal with implicit and constrained deadline task sets only.

incurring less preemptions and migrations, which is an advantage of mpn-$*$.

Algorithm 1 mpn-$*$ scheduling algorithm

Job release: The following steps are taken whenever a job J_{new} of τ_i is released at t:

```
1: if there is an idle core then
2:     Start execution of $J_{new}$.
3: else
4:     Let $J_{curr}$ denote a currently executing job of a preemptive task $\tau_k$ (i.e., $Y_k = 1$), which has the lowest priority according to the prioritization policy of $*$.
5:     if $J_{new}$'s priority is lower than $J_{curr}$'s priority (according to $*$), or all currently executing jobs are invoked by non-preemptive tasks (i.e., $Y_k = 0$) then
6:         Put $J_{new}$ into the wait queue.
7:     else
8:         Stop executing $J_{curr}$, put $J_{curr}$ into the wait queue, and start to execute $J_{new}$.
9:     end if
10: end if
```

Job completion: The following step is taken whenever a currently executing job J_{curr} finishes its execution,

```
1: Start execution of a job with the highest priority (according to $*$) in the wait queue.
```

We now show two representative examples of mpn-$*$ algorithms. When we employ EDF as a prioritization policy, mpn-EDF determines the priority of jobs according to their deadlines. Therefore, the job deadline is used for comparison of job priorities in Job Release Steps 4 and 5 and Job Completion Step 1 of Algorithm 1. On the other hand, mpn-FP determines job priorities based on fixed task priorities, and therefore, the task priority is used for the job priority comparison. Note that the description of Algorithm 1, for now, targets prioritization policies in which each job's priority does not change over time, e.g., EDF, FP, EQDF (earliest quasi-deadline first) [11] and SPDF (smallest pseudo-deadline first) [12]; however, we can easily extend the algorithm for job-level dynamic prioritization policies, e.g., LLF (least laxity first) [13], in which a job's priority depends on its remaining time to deadline and remaining execution time, and thus, the priority varies with time.

Then, the MPN policy and mpn-$*$ are generalizations of non-preemptive and preemptive policies, and np-$*$ and fp-$*$, respectively, as stated in the following lemma.

Lemma 1. *The MPN policy subsumes non-preemptive and preemptive policies, while the mpn-EDF and mpn-FP scheduling algorithms subsume np-EDF and fp-EDF, and np-FP and fp-FP, respectively.*

Proof. The proof is straightforward. The non-preemptive (preemptive) policy is equivalent to the MPN policy with $Y_i = 0$ ($Y_i = 1$) for all $\tau_i \in \Phi$, and np-EDF (fp-EDF) is equivalent to mpn-EDF with $Y_i = 0$ ($Y_i = 1$) for all $\tau_i \in \Phi$. The same holds for mpn-FP over np-FP and fp-FP. □

4 Schedulability Analysis of mpn-$*$ Algorithms

In this section, we first develop a schedulability analysis framework for scheduling algorithms that employ the MPN preemption policy, and derive some properties that are commonly applied to any prioritization policy with the MPN policy. Then, we apply the analysis framework to two specific mpn-$*$ algorithms, yielding schedulability analyses of mpn-EDF and mpn-FP.

4.1 Schedulability Analysis Framework for mpn-$*$ Algorithms

In Section 2.2, schedulability analyses of fp-EDF and fp-FP have been developed by addressing the following framework/upper-bound.

F1. A response time analysis framework for a *preemptive* task (i.e., Lemma 9); and

U1. An upper-bound of $I^*_{k \leftarrow i}(l)$ if both τ_k and τ_i are preemptive (i.e., $W_i(l)$ and $E_{k \leftarrow i}$ in Eqs. (14) and (16)).

However, under any scheduling algorithm that employs the MPN preemption policy for a mixture of preemptive and non-preemptive tasks, we need to address the following framework/upper-bounds in addition to F1 and U1.

F2. A response time analysis framework for a *non-preemptive* task;

U2. An upper-bound of $I^*_{k \leftarrow i}(l)$ if τ_k is non-preemptive but τ_i is preemptive;

U3. An upper-bound of $I^*_{k \leftarrow i}(l)$ if both τ_k and τ_i are non-preemptive; and

U4. An upper-bound of $I^*_{k \leftarrow i}(l)$ if τ_k is preemptive but τ_i is non-preemptive.

In this section, we will develop F2, and derive some properties for U2, U3 and U4, which are commonly applied to any scheduling algorithm with the MPN policy. To address F2, we first introduce a non-preemptive task's property. By definition, any job of a non-preemptive task cannot be interrupted by any other job, and thus, the following property holds.

Observation 1. Once a job of a non-preemptive task starts its first time unit of execution, the execution should not be interrupted by any other job. Therefore, if a job of τ_k finishes its first time unit of execution at t, it finishes its entire execution no later than $t + C_k - 1$.

Based on the above observation, we can derive an upper-bound of the response time of a given non-preemptive task by calculating when the first time unit of execution of any job of the task is finished. Then, the following lemma provides a condition for an upper-bound of the response time.

Lemma 2. *The response time of a non-preemptive task τ_k is upper-bounded by $l + C_k - 1$ if the following inequality holds:*

$$1 + \left\lfloor \frac{1}{m} \sum_{\tau_i \in \Phi - \{\tau_k\}} \min\left(I^*_{k \leftarrow i}(l), l\right) \right\rfloor \leq l. \tag{1}$$

Proof. We prove the lemma by contradiction. Suppose that Eq. (1) holds for a given l, but the response time of τ_k is strictly greater than $l + C_k - 1$.

In this case, there exists t such that $I_k(t, t+l) \geq l$; otherwise, at least one unit of execution of any job of τ_k is performed within $[t, t+l)$, and then the response time is upper-bounded by $l + C_k - 1$. By Lemma 8 and the definition of $I^*_{k \leftarrow i}(l)$, the following inequality holds:

$$\begin{aligned} & I_k(t, t+l) \geq l \\ & \Leftrightarrow \sum_{\tau_i \in \Phi - \{\tau_k\}} \min(I_{k \leftarrow i}(t, t+l), l) \geq m \cdot l \\ & \Longrightarrow \sum_{\tau_i \in \Phi - \{\tau_k\}} \min\big(I^*_{k \leftarrow i}(l), l\big) \geq m \cdot l. \end{aligned} \quad (2)$$

Applying the final result of Eqs. (2) to (1), we show the contradiction, i.e., $1 + l \leq l$. □

Then, we can develop F2 using Lemma 2, and by merging F1 and F2, we develop a schedulability analysis framework for any global work-conserving scheduling algorithm that employs the MPN policy, as stated in the following theorem.

Theorem 1. *When a task set Φ is scheduled by an mpn-$*$ scheduling algorithm, an upper-bound of the response time of a preemptive task $\tau_k | Y_k = 1 \in \Phi$ is $R_k = R_k^x$ such that $R_k^{x+1} \leq R_k^x$ holds in the following expression, starting from $R_k^0 = C_k$:*

$$R_k^{x+1} \leftarrow C_k + \left\lfloor \frac{1}{m} \sum_{\tau_i \in \Phi - \{\tau_k\}} \min\big(I^*_{k \leftarrow i}(R_k^x), R_k^x - C_k + 1\big) \right\rfloor, \quad (3)$$

and an upper-bound of the response time of a non-preemptive task $\tau_k | Y_k = 0 \in \Phi$ is $R_k = F_k^x + C_k - 1$ such that $F_k^{x+1} \leq F_k^x$ holds in the following expression, starting from $F_k^0 = 1$:

$$F_k^{x+1} \leftarrow 1 + \left\lfloor \frac{1}{m} \sum_{\tau_i \in \Phi - \{\tau_k\}} \min\big(I^*_{k \leftarrow i}(F_k^x), F_k^x\big) \right\rfloor. \quad (4)$$

Then, if $R_k \leq D_k$ holds for all $\tau_k \in \Phi$, Φ is schedulable by the algorithm.

Note that the iteration of Eq. (4) (Eq. (3)) for a non-preemptive (preemptive) task τ_k halts if $F_k^x + C_k - 1 > D_k$ ($R_k^x > D_k$), meaning that τ_k is deemed unschedulable.

Proof. By Lemma 9, R_k derived by Eq. (3) is an upper-bound of the response time of a preemptive task τ_k. Also, by Lemma 2, the response time of a non-preemptive task τ_k is guaranteed to be upper-bounded by $R_k = F_k^x + C_k - 1$ if $F_k^{x+1} \leq F_k^x$. Thus, if $R_k \leq D_k$ holds for all $\tau_k \in \Phi$, Φ is schedulable. □

Using Theorem 1, we can find whether or not a task set is schedulable under a specific mpn-$*$ scheduling algorithm, as long as the upper-bounds of $I^*_{k \leftarrow i}(l)$ of U1–U4 under the algorithm can be derived. While the upper-bounds depend on the underlying scheduling algorithms, here we study some interesting properties of the upper-bounds, which are applicable to any mpn-$*$ scheduling algorithm. Then, these properties will be the key in calculating $I^*_{k \leftarrow i}(l)$ under specific mpn-$*$ algorithms, e.g., mpn-EDF and mpn-FP in Sections 4.2 and 4.3.

For U1–U4 under any mpn-$*$ scheduling algorithm, a higher-priority job can interfere with a lower-priority job. However, it depends on the preemptiveness of lower- and higher-priority jobs, whether a lower-priority job can interfere with a higher-priority job.

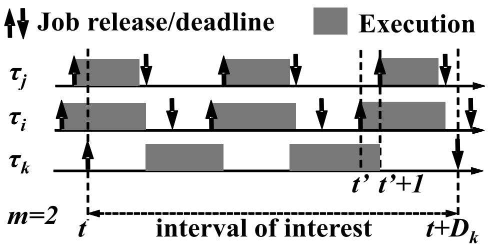

Fig. 1. A case where a lower-priority job of a non-preemptive task τ_i can interfere with a high-priority job of a preemptive τ_k when the highest-priority job of τ_j is released.

Now, we identify some properties regarding interference, starting from the cases of U1 and U2.

Observation 2 (The Cases of U1 and U2). Suppose that a job J_A is preemptive. Then J_A can interfere with another job J_B only when the priority of J_A is higher than that of J_B.

Since a preemptive lower-priority job can be blocked at any time by a higher-priority job, Observation 2 holds. However, a non-preemptive lower-priority job may interfere with a higher-priority via its non-preemptive behavior. The following observation deals with the case of U3.

Observation 3 (The Case of U3). Suppose that both jobs J_A and J_B are non-preemptive. Then, J_A can interfere with J_B even if the priority of J_A is lower than that of J_B. This occurs only when the execution of J_A starts before the release of J_B. Therefore, there are at most m lower-priority non-preemptive jobs that can interfere with a higher-priority non-preemptive job.

Suppose that a non-preemptive lower-priority job J_A starts its execution before t_B, at which a non-preemptive higher-priority job J_B is released. Then, regardless of priorities of J_A and J_B, J_A does not pause its execution due to non-preemptiveness, and therefore, J_A can interfere with J_B. However, since J_B is non-preemptive, such priority inversion cannot occur if J_A does not start its execution before t_B; in this case, J_A cannot start its execution before J_B's execution starts, and once J_B starts its execution, it also does not stop its execution. Therefore, the only case for J_A to interfere with J_B is that J_A executes in $[t_B - 1, t_B)$ and then continues the execution after t_B due to its non-preemptiveness. Since the number of jobs executed in any given time slot is upper-bounded by m, the number of lower-priority non-preemptive jobs that interfere with a given higher-priority non-preemptive job is upper-bounded by m.

One may think that Observation 3 holds for the case where J_A is non-preemptive and J_B is preemptive, i.e., the case of U4. However, this is not true because priority inversion occurs more often for this case. Consider an interval $[t, t + D_k)$ where t is the release time of a job of a preemptive task τ_k. Also, a job of a non-preemptive task τ_i is released after t and has a lower priority than the job of τ_k. In this case, the job of τ_i can interfere with that of τ_k in $[t, t + D_k)$, although the job of τ_i does not start its execution before t. Fig. 1 represents the case with $m = 2$ and three tasks τ_i, τ_j and τ_k. In $[t', t' + 1)$, both jobs of a non-preemptive task τ_i and a preemptive task τ_k execute. When the highest-priority job of τ_j is released at $t' + 1$, the job of τ_k has to pause its execution due to the highest-priority job of τ_j, but the job of τ_i

does not pause due to its non-preemptiveness. In this case the job of τ_i interferes with the job of τ_k after $t'+1$ regardless of its priority, and this can happen for any job of τ_i.

We summarize the above observation as follows.

Observation 4 (The Case of U4). Suppose that a job J_A is non-preemptive, and another job J_B is preemptive. Then, J_A can interfere with J_B even if the priority of J_A is lower than that of J_B. This may occur regardless of the release time of J_A.

Note that the case in Fig. 1 cannot occur when both τ_k and τ_i are non-preemptive (i.e., the case of U3). This is because the job of τ_k in the figure cannot be preempted once it starts execution (since the job of τ_k is non-preemptive). Therefore, the job of τ_k cannot be preempted at $t'+1$, meaning that a job of τ_i released at t' cannot interfere with the job of τ_k.

While the schedulability analysis framework in Theorem 1, and Observations 2, 3 and 4 can be applied to any mpn-$*$ algorithm with existing prioritization policies (e.g., EDF, FP, EQDF, and SPDF), we will present schedulability analyses of mpn-EDF and mpn-FP as examples, in the following sections.

4.2 Schedulability Analysis of mpn-EDF

In this section, we develop the schedulability analysis of mpn-EDF by calculating upper-bounds of $I^*_{k\leftarrow i}(l)$ under mpn-EDF, based on the observations in Section 4.1 and properties of the prioritization policy of EDF.

As presented in the supplement, available online, $E_{k\leftarrow i}$ in Eq. (16) is the maximum amount of execution of higher-priority jobs of a task τ_i than a job of another task τ_k in an interval between the release and the deadline of the τ_k's job under EDF. Therefore, $E_{k\leftarrow i}$ can be used for an upper-bound of U1–U4 when only higher-priority jobs of τ_i can interfere with a lower-priority job of τ_k.

By Observation 2, a lower-priority job of a preemptive task τ_i cannot interfere with a higher-priority job of a task τ_k for the cases of U1 and U2; the remaining step is to calculate U3 and U4 when at least one lower-priority job of τ_i interfere with the job of τ_k. As stated in Observation 3, a lower-priority job of a non-preemptive task τ_i can interfere with a higher-priority job of a non-preemptive task τ_k only if the job of τ_i starts its execution before the release of the job of τ_k. Therefore, $I^*_{k\leftarrow i}(l)$ is upper-bounded by C_i-1 in this case, and there are at most m jobs of such τ_i. Under the prioritization policy of EDF, $D_i > D_k$ is a necessary condition for a job of τ_i to start its execution before a job of τ_k's release and to have a lower priority than the job of τ_k.

When it comes to U4, Observation 4 represents that any job of a non-preemptive task τ_i can interfere with a job of a preemptive task τ_k. Therefore, we use $W_i(l)$ in Eq. (14) (an upper-bound applicable to all cases) as an upper-bound of $I^*_{k\leftarrow i}(l)$.

We summarize upper-bounds of $I^*_{k\leftarrow i}(l)$ under mpn-EDF for different cases in Table 2.

Considering the fact that $W_i(l)$ in Eq. (14) is an upper-bound for all cases, $I^*_{k\leftarrow i}(l)$ for U1 and U2 is upper-bounded by $\min(W_i(l), E_{k\leftarrow i})$. For U3 and U4, we do not know when lower-priority jobs of τ_i interfere with a job of τ_k. Therefore, for U4, we use $W_i(l)$, which is the maximum of $\min(W_i(l), E_{k\leftarrow i})$ and $W_i(l)$. For U3, we choose the maximum of $\min(W_i(l), E_{k\leftarrow i})$ and $\min(W_i(l), C_i-1)$; however, as stated in Observation 3, the number of tasks τ_i that belong to the case of interference from the lower-priority job is at most m.

TABLE 2
Upper-Bounds of $I^*_{k\leftarrow i}(l)$ under mpn-EDF

	When only higher-priority jobs of τ_i interfere	When at least one lower-priority job of τ_i interfere
U1	$E_{k\leftarrow i}$	Not applicable
U2	$E_{k\leftarrow i}$	Not applicable
U3	$E_{k\leftarrow i}$	C_i-1 only if $D_i>D_k$, (The number of tasks τ_i that belong to U3 is at most m.)
U4	$E_{k\leftarrow i}$	$W_i(l)$

Then, the following lemma presents upper-bounds on $\sum_{\tau_i\in\Phi-\{\tau_k\}} \min(I^*_{k\leftarrow i}(l), l-C_k+1)$ in Eq. (3) and $\sum_{\tau_i\in\Phi-\{\tau_k\}} \min(I^*_{k\leftarrow i}(l), l)$ in Eq. (4) under mpn-EDF.

Lemma 3. *Under mpn-EDF, the following inequalities hold for all $\tau_k\in\Phi$ and $0\le l\le D_k$:*

If τ_k is preemptive, i.e., $Y_k=1$ (the cases of U1 and U4),

$$\begin{aligned}&\sum_{\tau_i\in\Phi-\{\tau_k\}} \min\big(I^*_{k\leftarrow i}(l), l-C_k+1\big) \text{ in Eq. (3)}\\ &\le \sum_{\tau_i|Y_i=1\in\Phi-\{\tau_k\}} \min(W_i(l), E_{k\leftarrow i}, l-C_k+1)\\ &\quad+\sum_{\tau_i|Y_i=0\in\Phi-\{\tau_k\}} \min\big(W_i(l), l-C_k+1\big),\end{aligned} \tag{5}$$

and if τ_k is non-preemptive, i.e., $Y_k=0$ (the cases of U2 and U3),

$$\begin{aligned}&\sum_{\tau_i\in\Phi-\{\tau_k\}} \min\Big(I^*_{k\leftarrow i}(l), l\Big) \text{ in Eq. (4)}\\ &\le \sum_{\tau_i\in\Phi-\{\tau_k\}} \min\big(W_i(l), E_{k\leftarrow i}, l\big)\\ &\quad+\sum_{\text{mlargest}\tau_i|Y_i=0\&D_i>D_k\in\Phi-\{\tau_k\}}\\ &\max\big(0, \min\big(W_i(l), C_i-1, l\big)-\min\big(W_i(l), E_{k\leftarrow i}, l\big)\big).\end{aligned} \tag{6}$$

Proof. By U1 and U4 in Table 2, the RHS of Eq. (5) is a safe upper-bound on the LHS. To safely upper-bound the LHS of Eq. (6), we initially add the upper-bound of U2 (which is equivalent to U3 for the case of no priority inversion) for every task. Then, we choose m non-preemptive tasks τ_i which have the largest values of the upper-bound of U3 for the case of priority inversion (i.e., $\min(W_i(l), C_i-1, l)$) minus that for the case of no priority inversion (i.e., $\min(W_i(l), E_{k\leftarrow i}, l)$), and add each difference when it is positive. Then, the RHS of Eq. (6) is a safe upper-bound on the LHS even if any combination of at most m non-preemptive tasks τ_i belongs to the case of priority inversion to a non-preemptive task τ_k. □

Then, to check the schedulability of a given task Φ under mpn-EDF, we use Theorem 1 with the upper-bounds in Lemma 3. Then, the way of using the slack values $\{S_i\}_{\tau_i\in\Phi}$ is the same as that of fp-EDF. We can obtain a *simple* schedulability test of mpn-EDF by setting all S_i to zero, and also an *improved* test of mpn-EDF by iterations for slack reclamation.

TABLE 3
Upper-Bounds of $I^*_{k\leftarrow i}(l)$ under mpn-FP

	$\tau_i \in \mathsf{HP}(\tau_k)$	$\tau_i \in \mathsf{LP}(\tau_k)$
U1	$W_i(l)$	0
U2	$W_i(l)$	0
U3	$W_i(l)$	$C_i - 1$ (The number of tasks τ_i that belong to U3 is at most m.)
U4	$W_i(l)$	$W_i(l)$

4.3 Schedulability Analysis of mpn-FP

Unlike mpn-EDF, mpn-FP employs task-level priorities. That is, while the priority of a job of τ_i can be higher or lower than that of a job of τ_k, depending on their deadlines under mpn-EDF, mpn-FP clearly separates jobs by tasks whether a job of τ_i can have a higher priority than a job of τ_k, regardless of job parameters. Therefore, we calculate U1–U4 for two cases: $\tau_i \in \mathsf{HP}(\tau_k)$ and $\tau_i \in \mathsf{LP}(\tau_k)$.

Under mpn-FP, when $\tau_i \in \mathsf{HP}(\tau_k)$, we use $W_i(l)$ as an upper-bound for U1–U4, as presented in Section 2. Now, we calculate U1–U4 in case of $\tau_i \in \mathsf{LP}(\tau_k)$. By Observation 2, a lower-priority preemptive job cannot interfere with a higher-priority job, and therefore, U1 and U2 are zero. Similar to mpn-EDF, U3 is upper-bounded by $C_i - 1$, and the number of lower-priority tasks τ_i that interfere with a job of τ_k is at most m. However, unlike mpn-EDF, the condition of $D_i > D_k$ is not necessary since the priority of jobs of τ_i is always lower than any job of τ_k if $\tau_i \in \mathsf{LP}(\tau_k)$. For U4, we also use $W_i(l)$ as an upper-bound according to Observation 4.

We summarize U1–U4 under mpn-FP in Table 3. Considering $I^*_{k\leftarrow i}(l)$ is upper-bounded by $W_i(l)$ in Eq. (14) in any case and the upper-bounds in Table 3 are clearly separated by task priorities, we present upper-bounds on $\sum_{\tau_i\in\Phi-\{\tau_k\}}\min(I^*_{k\leftarrow i}(l), l - C_k + 1)$ in Eq. (3) and $\sum_{\tau_i\in\Phi-\{\tau_k\}}\min(I^*_{k\leftarrow i}(l), l)$ in Eq. (4) under mpn-FP, as stated in the following lemma.

Lemma 4. *Under mpn-FP, the following inequalities hold for all $\tau_k \in \Phi$ and $0 \le l \le D_k$.*

If τ_k is preemptive, i.e., $Y_k = 1$ (the cases of U1 and U4),

$$\begin{aligned}&\sum_{\tau_i\in\Phi-\{\tau_k\}}\min\big(I^*_{k\leftarrow i}(l), l - C_k + 1\big) \text{ in Eq. (3)}\\ &\le \sum_{\tau_i\in\mathsf{HP}(\tau_k)\text{ or }\tau_i|Y_i=0\in\mathsf{LP}(\tau_k)}\min\big(W_i(l), l - C_k + 1\big),\end{aligned} \tag{7}$$

and if τ_k is non-preemptive, i.e., $Y_k = 0$ (the cases of U2 and U3),

$$\begin{aligned}&\sum_{\tau_i\in\Phi-\{\tau_k\}}\min\Big(I^*_{k\leftarrow i}(l), l\Big) \text{ in Eq. (4)}\\ &\le \sum_{\tau_i\in\mathsf{HP}(\tau_k)}\min\big(W_i(l), l\big)\\ &\quad+ \sum_{\text{m largest}\tau_i|Y_i=0\in\mathsf{LP}(\tau_k)}\min\big(W_i(l), C_i - 1, l\big).\end{aligned} \tag{8}$$

Proof. We first investigate the case of a preemptive τ_k. As shown in Table 3, $I^*_{k\leftarrow i}(l)$ is upper-bounded by $W_i(l)$ if $\tau_i \in \mathsf{HP}(\tau_k)$ (i.e., U1 and U4), or $\tau_i \in \mathsf{LP}(\tau_k)$ and $Y_i = 0$ (i.e., U4). Therefore, Eq. (7) holds.

Then, consider a non-preemptive τ_k. Table 3 shows that $I^*_{k\leftarrow i}(l)$ is upper-bounded by $W_i(l)$ if $\tau_i \in \mathsf{HP}(\tau_k)$ (i.e., U2 and U3), and it is upper-bounded by $C_i - 1$ if $\tau_i \in \mathsf{LP}(\tau_k)$ and $Y_i = 0$ (i.e., U3). Similar to mpn-EDF, we choose m non-preemptive tasks τ_i which have the largest $\min(W_i(l), C_i - 1, l)$ among tasks τ_i in $\mathsf{LP}(\tau_k)$ with $Y_i = 0$. Therefore, the RHS of Eq. (8) is a safe upper-bound of the LHS. □

Then, it is the same as that under mpn-EDF, how to check the schedulability of a given task Φ under mpn-FP, i.e., applying Theorem 1 with/without the slack values.

The schedulability tests of mpn-EDF in Section 4.2 and mpn-FP in this section have the following property.

Lemma 5. *The schedulability tests of mpn-EDF and mpn-FP in Theorem 1 with the upper-bounds in Lemmas 3 and 4 with/without slack reclamation (simple/improved tests) generalize the corresponding schedulability tests for fp-EDF and fp-FP in Lemma 9 with the upper-bounds in Eqs. (17) and (18) with/without slack reclamation (i.e., Theorem 6 with Eqs. (4) and (5), and Theorem 6 with Eqs. (8) and (9) in [4]).*

Proof. The proof is straightforward; the schedulability analysis of fp-EDF is equivalent to that for mpn-EDF with $Y_i = 1$ for all $\tau_i \in \Phi$. This also holds for mpn-FP over fp-FP. □

Since mpn-EDF and mpn-FP also generalize np-EDF and np-FP, we highlight that the schedulability tests of mpn-EDF and mpn-FP in Theorem 1 with the upper-bounds in Lemmas 3 and 4 are actually new schedulability tests of np-EDF and np-FP by setting all Y_i to zero. We will demonstrate in Section 6 that our np-EDF and np-FP schedulability tests outperform existing schedulability tests of np-EDF and np-FP.

Then, the time-complexity of the response time analysis for mpn-EDF and mpn-FP (i.e., Theorem 1 with the upper-bounds in Lemmas 3 and 4) is the same as that for fully-preemptive scheduling [4] described in the supplement, available online: $O(n^2 \cdot \max_{\tau_i\in\Phi} D_i)$ and $O(n^3 \cdot \max_{\tau_i\in\Phi} D_i^2)$ for the simple and improved tests, respectively [4].

5 Optimal Assignment of $\{Y_i\}$ for mpn-$*$

While allowance/disallowance of preemption is a specification of each task, it does not violate the specification to execute preemptive tasks as if it were non-preemptive. To utilize this for improving schedulability, we first study how the response times of other tasks vary when we execute a preemptive task as a non-preemptive one (i.e., $Y_i = 1 \rightarrow 0$). Based on this, we develop an algorithm that finds an assignment of $\{Y_i\}$. Then, we prove its optimality when the underlying schedulability tests of mpn-EDF or mpn-FP are the simple ones.

The following lemma presents the effect of making a preemptive task non-preemptive on the response time of other tasks.

Lemma 6. *Suppose we apply the simple schedulability tests of mpn-EDF or mpn-FP without slack reclamation, i.e., S_j is set to 0 and does not change for all $\tau_j \in \Phi$. Also, suppose a single preemptive task $\tau_i \in \Phi$ is made non-preemptive (i.e., $Y_i = 1 \rightarrow 0$). Then, $R_k \le R'_k$ holds, where R_k and R'_k denote the* *upper-bounds of the response time of τ_k ($\neq \tau_i$) when $Y_i = 1$ and $Y_i = 0$, respectively.*

Proof. We consider two cases, i.e., τ_k is preemptive or non-preemptive. If τ_k is preemptive, making a preemptive task τ_i ($\neq \tau_k$) non-preemptive does not decrease the upper-bound on $I^*_{k \leftarrow i}(l)$. (i.e., U1 $\leq$ U4 under mpn-EDF or mpn-FP). If τ_k is non-preemptive, the same holds (i.e., U2 $\leq$ U3 under mpn-EDF or mpn-FP). Since the upper-bound on the interference gets larger, $R_k \leq R'_k$ holds. □

While our control knob is to make some preemptive tasks non-preemptive, Lemma 6 states the fact that any unschedulable task (whose upper-bound of the response time is larger than it relative deadline) cannot be schedulable by making other preemptive tasks non-preemptive when the simple schedulability test is applied. Therefore, the only way to improve schedulability is to make unschedulable preemptive tasks themselves non-preemptive. Note that if we make a preemptive task τ_k non-preemptive, an upper-bound of $I^*_{k \leftarrow i}(l)$ remains or decreases (i.e., U1 = U2, and U4 $\geq$ U3 under mpn-EDF or mpn-FP). Algorithm 2 disallows the preemption of each preemptive task (assignment of $\{Y_i\}_{\tau_i \in \Phi}$) when the response time is greater than the relative deadline (i.e., $R_i > D_i$). Note that in Algorithm 2, we do not change tasks with $Y_i = 0$ to $Y_i = 1$ since it violates their specification. The following lemma proves the optimality of Algorithm 2.

Algorithm 2 ASSIGNMENT OF $\{Y_i\}_{\tau_i \in \Phi}$ FOR MPN-$*$

```
1: while true do
2:   Calculate $R_i, \forall \tau_i \in \Phi$ using Theorem 1 with the upper-
     bounds in Lemma 3. If $R_i \leq D_i, \forall \tau_i \in \Phi$, return
     SCHEDULABLE with $\{Y_i\}_{\tau_i \in \Phi}$.
3:   if $Y_i = 0$ holds for all $\tau_i \in \Phi$ then
4:     Return UNSCHEDULABLE.
5:   end if
6:   for $\tau_i \in \Phi$ do
7:     if $R_i > D_i$ then
8:       if $Y_i = 1$ then
9:         $Y_i \leftarrow 0$
10:      end if
11:    end if
12:  end for
13:  if There is no update of $Y_i$ for all $\tau_i \in \Phi$ in Steps 6-12
     then
14:    Choose a task $\tau_k \in \Phi$ with $Y_k = 1$, and $Y_k \leftarrow 0$.
15:  end if
16: end while
```

Lemma 7 (Optimality of Algorithm 2). *Suppose the simple schedulability test of mpn-EDF (or mpn-FP) is applied, i.e., Theorem 1 with the upper-bounds in Lemma 3 (or Lemma 4) without slack reclamation, meaning that $\forall \tau_j \in \Phi, S_j$ is set to 0 and does not change. If Algorithm 2 deems Φ unschedulable, any $\Phi' \triangleq \{\tau'_i\}$ such that τ'_i is the same as τ_i except Y'_i ($\leq Y_i$) cannot be deemed schedulable by the simple schedulability test of mpn-EDF (or mpn-FP).*

Proof. Suppose that there exists a task set Φ' which is schedulable by the simple schedulability test of mpn-EDF (or mpn-FP), but Algorithm 2 returns unschedulable. We divide Φ into two disjoint task sets: $\Phi_A = \{\tau_i \in \Phi \,|\, Y_i = Y'_i\}$ and $\Phi_B = \{\tau_i \in \Phi \,|\, Y_i = 1 \text{ and } Y'_i = 0\}$. Since Φ' is schedulable, at the first iteration of Steps 2-15, tasks in Φ_A satisfy $R_i \leq D_i$ by Lemma 6, and tasks in Φ_B may violate $R_i \leq D_i$. Therefore, at the first iteration, only some tasks in Φ_B can change its Y_i as 0. Similarly, at each iteration, tasks in Φ_A always satisfy $R_i \leq D_i$ by Lemma 6, and each task in Φ_B either satisfies $R_i \leq D_i$ or eventually becomes non-preemptive (i.e., $Y_i = 0$). This means that the algorithm finds another schedulable assignment or tests Φ' eventually. In both cases, the algorithm returns schedulable, which is a contradiction. □

Then, we can find the optimal disallowance of preemptions with low time-complexity; while a naive approach needs to consider $O(2^n)$ assignments, Algorithm 2 finds the optimal assignment by considering only $O(n)$ assignments, where n is the number of preemptive tasks in a task set Φ. This is because, for each iteration of Steps 2-15 in the algorithm, we set at least one task with $Y_i = 1$ to $Y_i = 0$, and once Y_i is set to 0, it is not re-set to 1; therefore, the number iterations is $O(n)$. Since each iteration requires a response time analysis (i.e., Theorem 1 with the upper-bounds in Lemmas 3 and 4), the total time-complexity of Algorithm 2 with the simple and improved tests is $O(n^3 \cdot \max_{\tau_i \in \Phi} D_i)$ and $O(n^4 \cdot \max_{\tau_i \in \Phi} D_i^2)$, respectively.

However, such optimality does not necessarily hold when we apply the improved schedulability test of mpn-EDF or mpn-FP, which uses slack reclamation. This is because if we make a preemptive task τ_i non-preemptive, its response time may get decreased, meaning that the slack value may also get increased. This can help reduce the response time of another task τ_k. Despite its non-optimality, Algorithm 2 with the improved test effectively finds a large number of additional schedulable task sets which are not schedulable by the corresponding improved tests of np-EDF and fp-EDF, and np-FP and fp-FP, which will be demonstrated in Section 6.

6 EVALUATION

In this section, we evaluate the schedulability analysis of mpn-EDF and mpn-FP, and the algorithm that disallows preemption based on this analysis. We first describe how task sets are generated. Then, we present the average schedulability performance of EDF with different preemption policies, and then FP with the policies; we demonstrate the schedulability improvement by mpn-EDF and mpn-FP with disallowance of preemption, and the improvement of the np-EDF and np-FP schedulability analysis.

6.1 Generation of Task Sets

We generate task sets based on the technique in [14], which has also been widely used elsewhere [15], [16]. See the supplement, available online, for the details of the task-set generation.

6.2 Average Schedulability of EDF with Different Preemption Policies

We show the number of task sets schedulable by the following schedulability tests of EDF with different preemption policies:

- The only existing np-EDF tests [5], [6], [7] (denoted by np-EDF[GBL]), i.e., schedulable by at least one of the three tests.[4]
- Our schedulability test for np-EDF, i.e., Theorem 1 with the upper-bounds in Lemma 3 when $Y_i = 0$ for all $\tau_i \in \Phi$ (denoted by np-EDF).
- The existing fp-EDF test [4], which is equivalent to our schedulability test for fp-EDF, i.e., Theorem 1 with the upper-bounds in Lemma 3 when $Y_i = 1$ for all $\tau_i \in \Phi$ (denoted by fp-EDF).
- np-EDF and fp-EDF(denoted by np-EDF+fp-EDF), i.e., schedulable by at least one of np-EDF and fp-EDF.
- Our schedulability test of mpn-EDF with disallowance preemption by Algorithm 2 when the initial preemption requirement is $Y_i = 1$ for all $\tau_i \in \Phi$ (denoted by mpn-EDF).

Note that we present our schedulability tests with the best performance; in other words, we present the improved test with slack reclamation for np-EDF, fp-EDF, and mpn-EDF.

Fig. 5 (in the supplement, available online) shows average schedulability results for different task sets and m. As shown in the figure, np-EDF improves np-EDF[GBL], and this improvement is pronounced when m is large. That is, np-EDF finds 3.0, 7.0, 13.6, and 33.3 percent (likewise 3.5, 11.7, 16.2, and 21.9 percent) additional schedulable constrained deadline task sets (likewise implicit deadline task sets), which are deemed unschedulable by np-EDF[GBL], respectively for $m = 2, 4, 8$ and 16.

There are two reasons why np-EDF outperforms np-EDF[GBL]. First, while np-EDF[GBL] finds the time of completion of the last execution, np-EDF uses the framework of Eq. (4), which identifies the time instant of the first execution of a given job J. Once we find this time instant, any other job executed after that instant cannot block the execution of J, reducing the pessimism in calculating the amount of interference. Second, we have shown in Observation 3 that a non-preemptive lower-priority job J_1 can interfere with a non-preemptive higher-priority job J_2 only when J_1 starts its execution before the release time of J_1.[5] These two cases, incorporated into the fact that the number of jobs of (ii) is upper-bounded by m, yield a tighter (i.e., less pessimistic) interference bound than those used in np-EDF[GBL].

We now compare mpn-EDF with np-EDF+fp-EDF. As shown in the figures, mpn-EDF can find additional schedulable task sets which are schedulable by neither np-EDF nor fp-EDF. The fraction of such additional task sets also gets increased, as the number of cores becomes larger; mpn-EDF can find 10.2, 20.9, 30.9 and 39.2 percent (likewise 5.0, 12.5, 21.3 and 28.7 percent) additional schedulable constrained deadline task sets (likewise implicit deadline task sets) which are deemed schedulable by neither np-EDF nor fp-EDF, respectively, for $m = 2, 4, 8$ and 16.

4. The study [7] was missing in our preliminary conference version in [1].

5. The technique in [7] addresses the second issue while other techniques in [5], [6] do not.

More evaluation results are given in the supplement, available online.

6.3 Average Schedulability of FP with Different Preemption Policies

In the supplement, available online, we demonstrate the average schedulability of FP with different preemption policies, which exhibits similar behaviors as in the EDF case.

In summary, the proposed schedulability analyses of mpn-EDF and mpn-FP significantly improve the schedulability analyses of np-EDF and np-FP, which are special cases of mpn-EDF and mpn-FP. The disallowance of preemption enables discovery of a large number of additional schedulable task sets by exploiting non-preemptiveness as a control knob.

7 Related Work

For real-time *uniprocessor* systems, researchers proposed more general preemption policies than the fully-preemptive and non-preemptive policies to meet different goals (e.g., see a survey [3]). Starting from the preemption threshold scheduling [17], some researchers [18], [19], [20] attempted to improve the schedulability of FP by using the preemption requirement as a control knob, because the fully-preemptive policy is not an optimal preemption policy for FP on a uniprocessor platform. Several studies [21], [22], [23], [24] also focused on expressing broader preemption requirements, such as non-preemptive execution parts and the limit on the number of preemptions, and performed their schedulability analyses for uniprocessor systems. However, little has been done on schedulability analysis for more general preemption policies than fully-preemptive and non-preemptive policies for increasingly popular real-time *multi-core* platforms, which is the subject of this paper.

While numerous multi-core schedulability tests have been developed for fully-preemptive algorithms, there have been only a few studies [5], [6], [7], [8] for non-preemptive algorithms. Baruah has developed the first multi-core schedulability analysis of np-EDF [5] by extending the existing study for fp-EDF [25]. Also, two different schedulability tests of np-EDF [6], [7] have been published in the same conference, which adapt the carry-in bound technique for fp-EDF [26] to np-EDF. When it comes to np-FP, Guan et al. briefly presented the schedulability test of np-FP when they proposed that of np-EDF [6]. Later, the same authors developed another np-FP schedulability test in [8], focusing on time-complexity reduction.

8 Conclusion

We have proposed the MPN preemption policy, a generalization of preemptive and non-preemptive policies for real-time multicore platforms, and then developed a schedulability analysis framework for mpn-$*$ algorithms. We have chosen mpn-EDF and mpn-FP as examples and carried out their schedulability analyses, showing that they not only generalize existing schedulability analyses of fp-EDF and fp-FP, but also outperform existing schedulability analyses of np-EDF and np-FP, respectively. We have also presented a (sub)optimal assignment algorithm of disallowing preemptions, and demonstrated that the algorithm efficiently finds additional schedulable task sets which are schedulable by neither corresponding fp-∗ nor np-∗ algorithms.

In future, we would like to develop a schedulability analysis framework for more general preemption policies than the MPN policy. That is, instead of accommodating only preemptive and non-preemptive tasks, we may also consider a task with limited preemptions in terms of the number and/or the duration of preemptions, which have already been studied for *uniprocessor* scheduling, but not for multi-core scheduling. For example, we may apply the preemption threshold scheduling (PTS) [17] to real-time multi-core systems. PTS is more expressive than the MPN policy; a job of task A cannot preempt any job of task B if task B's preemption threshold is higher than that of task A. To apply PTS, we need to develop a new schedulability analysis; this is challenging and requires investigation of more general cases for frameworks/upper-bounds corresponding to F1–F2/U1–U4 in this paper, which is part of our future work.

Acknowledgments

An earlier (shorter) version [1] of this paper was presented at the IEEE RTSS 2012. The work reported in this paper was supported in part by the NSF under grants CNS-0930813 and CNS-1138200. The work reported in this paper was also supported in part by National Research Foundation of Korea Grant funded by the Korean Government (Ministry of Education, Science and Technology) [NRF-2011-357-D00186]. Jinkyu Lee is the corresponding author.

References

[1] J. Lee and K.G. Shin, "Controlling Preemption for Better Schedulability in Multi-Core Systems," *Proc. IEEE Real-Time Systems Symp. (RTSS)*, pp. 29-38, 2012.

[2] C. Liu and J. Layland, "Scheduling Algorithms for Multi-Programming in a Hard-Real-Time Environment," *J. ACM*, vol. 20, no. 1, pp. 46-61, 1973.

[3] G. Buttazzo, M. Bertogna, and G. Yao, "Limited Preemptive Scheduling for Real-Time Systems: A Survey," *IEEE Trans. Industrial Informatics*, vol. 9, no. 1, pp. 3-15, Feb. 2013.

[4] M. Bertogna and M. Cirinei, "Response-Time Analysis for Globally Scheduled Symmetric Multiprocessor Platforms," *Proc. IEEE Real-Time Systems Symp. (RTSS)*, pp. 149-160, 2007.

[5] S. Baruah, "The Non-Preemptive Scheduling of Periodic Tasks Upon Multiprocessors," *Real-Time Systems*, vol. 32, no. 1, pp. 9-20, 2006.

[6] N. Guan, W. Yi, Z. Gu, Q. Deng, and G. Yu, "New Schedulability Test Conditions for Non-Preemptive Scheduling on Multiprocessor Platforms," *Proc. IEEE Real-Time Systems Symp. (RTSS)*, pp. 137-146, 2008.

[7] H. Leontyev and J.H. Anderson, "A Unified Hard/Soft Real-Time Schedulability Test for Global EDF Multiprocessor Scheduling," *Proc. IEEE Real-Time Systems Symp. (RTSS)*, pp. 375-384, 2008.

[8] N. Guan, W. Yi, Q. Deng, Z. Gu, and G. Yu, "Schedulability Analysis for Non-Preemptive Fixed-Priority Multiprocessor Scheduling," *J. Systems Architecture*, vol. 57, no. 5, pp. 536-546, 2011.

[9] A. Mok, "Fundamental Design Problems of Distributed Systems for The Hard-Real-Time Environment," PhD dissertation Massachusetts Inst. of Technology, 1983.

[10] M. Bertogna and S. Baruah, "Tests for Global EDF Schedulability Analysis," *J. Systems Architecture*, vol. 57, pp. 487-497, 2011.

[11] H. Back, H.S. Chwa, and I. Shin, "Schedulability Analysis and Priority Assignment for Global Job-Level Fixed-Priority Multiprocessro Scheduling," *Proc. IEEE Real-Time Technology and Applications Symp. (RTAS)*, pp. 297-306, 2011.

[12] H.S. Chwa, H. Back, S. Chen, J. Lee, A. Easwaran, I. Shin, and I. Lee, "Extending Task-Level to Job-Level Fixed Priority Assignment and Schedulability Analysis Using Pseudo-Deadlines," *Proc. IEEE Real-Time Systems Symp. (RTSS)*, pp. 51-62, 2012.

[13] J.Y.-T. Leung, "A New Algorithm for Scheduling Periodic, Real-Time Tasks," *Algorithmica*, vol. 4, pp. 209-219, 1989.

[14] T.P. Baker, "Comparison of Empirical Success Rates of Global vs. Partitioned Fixed-Priority and EDF Scheduling for Hand Real Time," Technical Report TR-050601, Dept. of Computer Science, Florida State Univ., 2005.

[15] M. Bertogna, M. Cirinei, and G. Lipari, "Schedulability Analysis of Global Scheduling Algorithms on Multiprocessor Platforms," *IEEE Trans. Parallel and Distributed Systems*, vol. 20, no. 4, pp. 553-566, Apr. 2009.

[16] J. Lee, A. Easwaran, and I. Shin, "Maximizing Contention-Free Executions in Multiprocessor Scheduling," *Proc. IEEE Real-Time Technology and Applications Symp. (RTAS)*, pp. 235-244, 2011.

[17] Y. Wang and M. Saksena, "Scheduling Fixed-Priority Tasks with Preemption Threshold," *Proc. IEEE Int'l Conf. Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pp. 318-335, 1999.

[18] M. Bertogna, G. Buttazzo, and G. Yao, "Improving Feasibility of Fixed Priority Tasks Using Non-Preemptive Regions," *Proc. IEEE Real-Time Systems Symp. (RTSS)*, pp. 251-260, 2011.

[19] R.J. Bril, M.M. van den Heuvel, U. Keskin, and J.J. Lukkien, "Generalized Fixed-Priority Scheduling with Limited Preemptions," *Proc. Euromicro Conf. Real-Time Systems (ECRTS)*, pp. 209-220, 2012.

[20] R.I. Davis and M. Bertogna, "Optimal Fixed Priority Scheduling with Deferred Pre-Emption," *Proc. IEEE Real-Time Systems Symp. (RTSS)*, pp. 39-50, 2012.

[21] S. Baruah, "The Limited-Preemption Uniprocessor Scheduling of Sporadic Task Systems," *Proc. Euromicro Conf. Real-Time Systems (ECRTS)*, pp. 137-144, 2005.

[22] G. Yao, G. Buttazzo, and M. Bertogna, "Bounding The Maximum Length of Non-Preemptive Regions under Fixed Priority Scheduling," *Proc. IEEE Int'l Conf. Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pp. 351-360, 2009.

[23] M. Bertogna and S. Baruah, "Limited Preemption EDF Scheduling of Sporadic Task Systems," *IEEE Trans. Industrial Informatics*, vol. 6, no. 4, pp. 579-591, 2010.

[24] G. Yao, G. Buttazzo, and M. Bertogna, "Feasibility Analysis under Fixed Priority Scheduling with Limited Preemptions," *Real-Time Systems*, vol. 47, no. 3, pp. 198-223, 2011.

[25] J. Goossens, S. Funk, and S. Baruah, "Priority-Driven Scheduling of Periodic Task Systems on Multiprocessors," *Real-Time Systems*, vol. 25, no. 2-3, pp. 187-205, 2003.

[26] S. Baruah, "Techniques for Multiprocessor Global Schedulability Analysis," *Proc. IEEE Real-Time Systems Symp. (RTSS)*, pp. 119-128, 2007.

[27] M. Bertogna, M. Cirinei, and G. Lipari, "Improved Schedulability Analysis of EDF on Multiprocessor Platforms," *Proc. Euromicro Conf. Real-Time Systems (ECRTS)*, pp. 209-218, 2005.

[28] T.P. Baker and M. Cirinei, "A Necessary and Sometimes Sufficient Condition for the Feasibility of Sets of Sporadic Hard-Deadline Tasks," *Proc. IEEE Real-Time Systems Symp. (RTSS)*, pp. 178-190, 2006.

[29] J. Leung and J. Whitehead, "On the Complexity of Fixed-Priority Scheduling of Periodic Real-Time Tasks," *Performance Evaluation*, vol. 2, pp. 237-250, 1982.

Jinkyu Lee is an assistant professor in Department of Computer Science and Engineering, Sungkyunkwan University, 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.

Kang G. Shin is the Kevin & Nancy O'Connor professor of Computer Science in the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor. His current research focuses on QoS-sensitive computing and networking as well as on embedded real-time and cyber-physical systems. He has supervised the completion of 74 PhDs, and authored/coauthored about 800 technical articles (about 300 of these are in archival journals), a textbook and more than 20 patents or invention disclosures, and received numerous best paper awards, including the Best Paper Awards from the 2011 ACM International Conference on Mobile Computing and Networking (MobiCom'11), the 2011 IEEE International Conference on Autonomic Computing, the 2010 and 2000 USENIX Annual Technical Conferences, as well as the 2003 IEEE Communications Society William R. Bennett Prize Paper Award and the 1987 Outstanding IEEE Transactions of Automatic Control Paper Award. He has also received several institutional awards, including the Research Excellence Award in 1989, Outstanding Achievement Award in 1999, Distinguished Faculty Achievement Award in 2001, and Stephen Attwood Award in 2004 from The University of Michigan (the highest honor bestowed to Michigan Engineering faculty); a Distinguished Alumni Award of the College of Engineering, Seoul National University in 2002; 2003 IEEE RTC Technical Achievement Award; and 2006 Ho-Am Prize in Engineering (the highest honor bestowed to Korean-origin engineers). He is a life fellow of the IEEE.