Computing (2017) 99:1257–1270
DOI 10.1007/s00607-017-0562-9

Real-time uniprocessor scheduling with fewer preemptions

Jinkyu Lee[1]

Received: 12 September 2016 / Accepted: 10 June 2017 / Published online: 28 June 2017

Abstract In this paper, we propose a simple, but effective scheduling framework for EDF and RM, which reduces the number of preemptions by simply introducing a dummy task. We first observe useful preemption behavior under EDF and RM, leading to an interesting finding: an effective way to reduce the number of preemptions is to prevent jobs of a task with the *smallest task period* from preempting other jobs upon their *release*. To achieve this, we add a dummy task that invokes its job only when a newly-released job of the task with the smallest task period has a higher priority than the currently-executing job. Then, the currently-executing job can continue its execution without getting preempted by inheriting the priority of the dummy job. Since adding the dummy task can make a schedulable task set unschedulable, we propose how to set the dummy task's parameters without compromising schedulability. In addition to the negligible overhead of this framework due to its simplicity, it holds an important property that does not increase the number of preemptions of any task set, compared to the original scheduling algorithm, which has not been achieved by existing studies. We also demonstrate via simulation that the proposed framework effectively reduces the number of preemptions under EDF and RM.

Keywords Fewer preemptions · EDF (Earliest Deadline First) · RM (Rate Monotonic) · Real-time scheduling · Uniprocessor systems

Mathematics Subject Classification 68M20

✉ Jinkyu Lee
jinkyu.lee@skku.edu

1 Department of Computer Science and Engineering, Sungkyunkwan University (SKKU), Suwon, Republic of Korea

1 Introduction

To satisfy the timing requirements of critical tasks, the subject of real-time scheduling has been studied extensively [1,2]. As a result, EDF (Earliest Deadline First) [3] and RM (Rate Monotonic) [3] have been proved as optimal dynamic and static scheduling algorithms for uniprocessor platforms, respectively, if a higher-priority job is always allowed to preempt a lower-priority one. However, a preemption incurs additional delay and power consumption for a context switch. For a uniprocessor system equipped with a cache, this overhead becomes greater because the preemption may cause loss of the cached contents.

There have been numerous efforts to reduce the number of preemptions on a uniprocessor platform [4], which can be classified into three categories. First, some studies have controlled preemptions in order to support various preemption requirements or improve schedulability. Starting from [5] that enforces a dual prioritization mechanism for job selection in the wait queue and job preemption (later extended in [6–8]), the studies in [9–13] accommodated non-preemptive regions. Since techniques in these studies have been originally designed to accommodate various preemption requirements or improve schedulability, there is no guarantee that the number of preemptions with the techniques is always smaller than that without the techniques for any task set (albeit the former is smaller, on average). Second, there have been some proposals for reducing preemptions. The proposal in [14] exchanges the order of execution based on the original schedule, but it requires knowledge of future job release patterns offline and complex run-time mechanisms for safe execution exchanges without missing any deadline. Third, several studies rely on dynamic voltage/frequency scaling [15–18], all of which require hardware support. Most studies in the third category incorporate non-zero preemption delays into schedulability analyses [19–21], which focus on a system model different from the one in this paper.

To overcome the limitations of the existing studies, the goal of this paper is to develop a scheduling framework that reduces the number of preemptions with the following salient features.

(i) Simplicity: it incurs little run-time/system overhead;
(ii) Wide applicability—it can be applied to not only existing algorithms including EDF and RM, but also existing schedulability analyses;
(iii) Independence from hardware/information: it does not rely on hardware support of dynamic voltages/frequency scaling and information of future job release patterns; and
(iv) Satisfaction of the important property: for *every* task set, the number of preemptions with our approach is smaller than (or at least equal to) that with the original scheduling.

To this end, we propose a new scheduling framework based on a dummy task, which provides the above four features. To achieve this, we first observe that under EDF and RM, (a) all preemptions take place when a job is released, and (b) a task with the smallest task period is a dominant source of preemption. Based on this preemption behavior, we artificially add a dummy task, which invokes its job only when a newly-released job of the task with the smallest task period has a higher priority than the currently-executing job. Then, the currently-executing job can continue execution without getting preempted by inheriting the priority of the dummy job. Since adding the dummy task may cause regular task deadline misses, we also present how to set the dummy task parameters without compromising the schedulability of a given task set.

Our proposed scheduling framework then provides the features (i)–(iv). Adding a dummy task makes the framework not only simple, but also applicable to most (if not all) scheduling algorithms as well as their schedulability analyses.[1] Also, the framework does not require any hardware support, nor any information on future job releases. More importantly, Sect. 3.3 proves that the framework meets the important property in (iv), which has not been achieved by existing studies. In addition to the four salient features, we demonstrate via simulation that EDF and RM associated with the framework based on the dummy task reduce the number of preemptions significantly, compared to vanilla EDF and RM.

The rest of the paper is organized as follows. Section 2 introduces the system model, assumptions and notations. Section 3 identifies preemption behavior under EDF and RM, proposes the dummy task based scheduling framework, and investigates its property. Section 4 describes how to set the dummy task parameters to preserve schedulability. Section 5 demonstrates the effectiveness of the framework using simulation. Finally, Sect. 6 concludes the paper.

2 System model, assumptions and notations

In this paper, we focus on a sporadic task model in [3], in which a task τ_i in a task set τ is specified by (T_i, C_i) where T_i is the the minimum separation between successive invocations (or the task period), and C_i is the worst-case execution time. Without loss of generality, we sort tasks in τ such that a task with a smaller T_i has a smaller task index, i.e., $T_1 \leq T_2 \leq \ldots T_{|\tau|}$, where $|\tau|$ is the number of tasks in the set τ. A task τ_i invokes a series of jobs; each job is separated from the predecessor/successor job by at least T_i time units, and supposed to finish its execution within T_i time units. We assume that a job can be preempted at any time.

We consider a uniprocessor system, on which at most one job can be executed in each time slot. We focus on two popular scheduling algorithms, EDF and RM [3]. In each time slot, while EDF executes a job with the earliest deadline, RM executes a job with the smallest task period (T_i).

3 Scheduling framework based on a dummy task for fewer preemptions

We now present a dummy-task-based scheduling framework to reduce preemptions. To achieve this, we first identify some preemption behavior under EDF and RM. Based on this observed behavior, we develop a dummy-task-based scheduling framework, which can significantly reduce the number of preemptions under EDF and RM. Finally, we derive an important property of the framework.

[1] We would like to stress that the framework can utilize any existing schedulability analysis in that it simply adds the dummy task to the original task set.

Table 1 Percentage of preemptions caused by a specific task under EDF and RM; 4500 task sets, each consisting of five tasks with $T_{max} = 1000$, are tested, and how to generate task sets is detailed in Sect. 5

Task	# of preemptions incurred by a specific task (%)	
	Under EDF (%)	Under RM (%)
τ_1	83.5	80.5
τ_2	12.4	13.3
τ_3	3.4	4.6
τ_4	0.8	1.5
τ_5	0.0	0.0

3.1 Preemption behavior under EDF and RM

While preemption depends on the underlying scheduling algorithms, we will show that under EDF and RM, the task with the shortest period is a dominant source of preemption. Specifically, we will present an analytic property and an empirical result of the dominant preemption source.

The following observation states the preemption behavior under EDF and RM.

Observation 1 *Under EDF and RM, a job J_x can preempt another job J_y, only upon release of J_x. Moreover, a preemption occurs only if the period of J_x is no larger than that of J_y.*

Proof Since the priority of jobs does not vary with time under EDF and RM, a job can preempt another job only when it is released. What remains is thus to prove the task period condition for preemptions.

Under EDF, if τ_i has a longer period than τ_j (i.e., $T_i > T_j$), a newly-released job of τ_i cannot have an earlier deadline than a currently-executing job of τ_j. Under RM, if $T_i > T_j$, all jobs of τ_i have lower priorities than all jobs of τ_j, meaning that no job of τ_i can preempt any job of τ_j. Thus, the lemma follows. □

By Observation 1, we know that a job of τ_i can preempt another job of τ_j only if $i < j$ (recall that tasks are indexed in ascending order according to their periods). Then, a task with a smaller index has more likely to cause preemptions. Thus, while τ_1 (i.e., the task with the smallest period) can preempt jobs of all other tasks, regardless of their deadlines under RM, and depending on their deadlines under EDF, $\tau_{|\tau|}$ (i.e., the task with the largest period) cannot preempt any job in any case under EDF and RM. To obtain the statistical results of this property, we simulate 4500 task sets of five tasks each, and measure the number of preemptions during the first 100,000 time units for each task set. As shown in Table 1, most preemptions are caused by τ_1: 83.5% under EDF, and 80.5% under RM. Note that τ_5 cannot cause any preemption by Observation 1.

Using Observation 1 and the simulation result, we will develop next a new scheduling framework that can reduce preemptions.

3.2 Dummy-task-based scheduling framework

Observation 1 and the simulation result in Table 1 indicate "which task" and "when" we should control. That is, to effectively reduce the number of preemptions under

Algorithm 1 Dummy-task-based scheduling framework

Timer Set: If a newly-released job of τ_1 has a higher priority than the currently-executing job at t, and the release time of the dummy task's previous job is no later than the current time $- T_x$,

1: Release a job of the dummy task τ_x with the execution time C_x.
2: Set the timer to $t + C_x$.
3: Let the currently-executing job inherit the priority of the job of the dummy task, i.e., keep the currently-executing job executing.
4: Put the newly-released job of τ_1 into the wait queue.

Timer expiration: If the timer expires at t and the currently-executing job is the one that inherits the priority of a job of the dummy task,

1: Let the currently-executing job stop inheriting the priority of the dummy task's job, i.e., let the currently-executing job stop execution.
2: Put the currently-executing job in the wait queue.
3: Start execution of the τ_1's job in the wait queue.

EDF or RM, we should control preemptions incurred by the *task with the smallest task period* (i.e., $\tau_1 \in \tau$), when it *releases* jobs. With this information, our goal is to reduce the number of preemptions without compromising task set schedulability. In other words, controlling preemptions should not make any schedulable task set unschedulable.

To achieve this goal, we add a dummy task $\tau_x(T_x, C_x)$ which invokes its jobs as follows:

(i) τ_x invokes a job only when a newly-released job of τ_1 has a higher priority than the currently-executing job.
(ii) The minimum separation between successive jobs (or the task period) of τ_x is T_1, i.e., $T_x = T_1$.

Since $T_x = T_1$, two jobs of τ_x and τ_1 always have the same priority under RM, and two jobs of τ_x and τ_1 released at the same time have the same priority under EDF. For the same-priority jobs, we enforce a tie-breaking rule; we give a higher priority to the dummy task's job if the priorities of multiple jobs are the same under EDF or RM. Then, between the two jobs of τ_1 and τ_x, released at the same time, the job of τ_x is always a given priority over that of τ_1. Therefore,

(iii) a job of τ_x has the highest priority when it is released, and this holds until its execution is completed. So, the job of τ_x executes for C_x time units without any preemption.

Under RM, a job of τ_x always has the highest priority. Recalling (i), under EDF, a job of τ_x has the highest priority when it is released. Since τ_x has the smallest period (T_x), no job released after τ_x releases a job, has a shorter deadline than the job of τ_x.

Using the property (iii), we can prevent a job of τ_1 from preempting the currently-executing job, by letting the currently-executing job inherit the priority of the job of τ_x. Then, by (iii), the currently-executing job can continue its execution without any preemption until the (virtual) execution of the dummy task's job is completed. Algorithm 1 details this dummy-task-based scheduling framework using (i), (ii) and (iii). It is important to note that the currently-executing job can have the highest priority during C_x time units because the virtual execution time of the job of the dummy task is C_x, and this is implemented using a timer in the algorithm.

As one can see in Algorithm 1, the dummy-task-based scheduling framework does not change the prioritization policy of the original scheduling algorithms. Instead, it adds jobs of the dummy task, and gives the currently-executing job chance to continue execution. Therefore, it need not (a) know job release patterns offline and (b) enforce complicated online mechanisms, which are required by some existing approaches. The only additional overhead is setting and expiration of a timer set for dummy jobs. Also, one more advantage of the framework in Algorithm 1 is its wider applicability to different scheduling algorithms. The framework can be applied not only to both EDF and RM, but also potentially to other existing algorithms; we can also re-use existing schedulability analyses by simply adding a dummy task when the task set is tested.

In the rest of the paper, let EDF-d (likewise RM-d) denote EDF (likewise RM) associated with Algorithm 1.

One may regard the proposed framework as a different expression of an existing technique called the *limited preemption* [9]. Under the limited preemption technique, a job always keeps its execution without any preemption during X time units, where X is the invoking task's length of non-preemptive region. Since the technique always disallows preemptions of a job of a task during X time units, it is impossible to selectively prevent a currently-executing job from being preempted by a job of τ_1. Therefore, the limited preemption technique with any parameter cannot yield the schedule generated by the proposed framework.

3.3 Property and example

Since Algorithm 1 is developed for fewer preemptions, it is important to determine whether the algorithm guarantees the reduction of the number of preemptions of *any task set* under a certain condition, compared to the corresponding original scheduling, which has not been done with any existing study. That is, we want to find the property that the number of preemptions of any task set under EDF-d is smaller than that under EDF. The following lemma addresses the property.

Lemma 1 *As long as there is no deadline miss, the number of preemptions of a task set under EDF-d (likewise RM-d) is no greater than that under EDF (likewise RM).*

Proof Let $n_{\mathsf{EDF}}(t)$ and $n_{\mathsf{EDF\text{-}d}}(t)$ denote the number of preemptions of a task set in $[0, t)$ under EDF and EDF-d, respectively. Suppose that there exists t such that $n_{\mathsf{EDF\text{-}d}}(t) > n_{\mathsf{EDF}}(t)$. We focus on the earliest t, called t_0. Then, at t_0, a preemption occurs under EDF-d, but not under EDF. We consider two cases: the job which incurs a preemption at t_0 belongs to (i) τ_1, and (ii) other task than τ_1.

Case (i). Since a job of the dummy task is released whenever a job of τ_1 is released, the job of τ_1 causes a preemption only when the timer is expired at t_0 (which was set to $t_0 - C_x$). Then, at $t_0 - C_x$ (when the timer is set), a preemption occurs under EDF, but not under EDF-d. Therefore, $n_{\mathsf{EDF\text{-}d}}(t_0 - C_x) \leq n_{\mathsf{EDF}}(t_0 - C_x) - 1$ holds; otherwise, $n_{\mathsf{EDF\text{-}d}}(t_0 - C_x - \epsilon) > n_{\mathsf{EDF}}(t_0 - C_x - \epsilon)$ holds for a small $\epsilon > 0$, which contradicts the definition of t_0. Using the inequality of $n_{\mathsf{EDF\text{-}d}}(t_0 - C_x) \leq n_{\mathsf{EDF}}(t_0 - C_x) - 1$ and the fact that there is no preemption under EDF-d in $(t_0 - C_x, t_0)$, we derive that $n_{\mathsf{EDF\text{-}d}}(t_0) \leq n_{\mathsf{EDF}}(t_0)$, which contradicts the supposition.

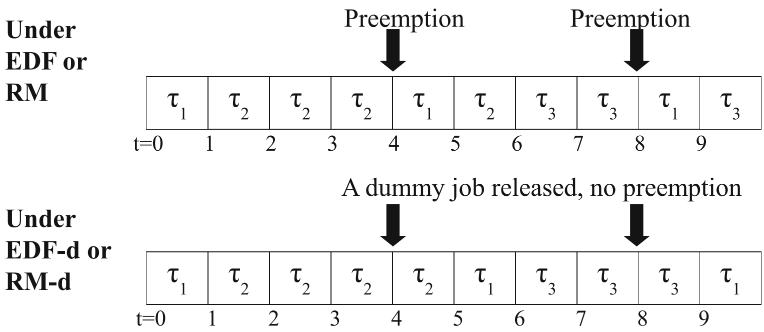


Fig. 1 Schedules with preemption behavior of $\tau = \{\tau_1(T_1 = 4, C_1 = 1), \tau_2(12, 4), \tau_3(20, 3)\}$ under EDF or RM, and EDF-d or RM-d

Case (ii). A job J of a task other than τ_1 can incur a preemption only when it is released and the currently-executing job has a lower priority than J. If we compare the schedule under EDF-d with that under EDF, the currently-executing job with the former has an equal- or higher-priority than that with the latter (since the former allows a job to inherit a dummy job' priority, which is the highest). Therefore, the supposition cannot hold.

By Cases (i) and (ii), the lemma holds for EDF-d; the proof for RM-d is the same as that for EDF-d. □

We would like to emphasize that such a guarantee on preemption reduction has not been achieved by existing studies (which can reduce the number of preemptions *on average*). While Lemma 1 guarantees fewer preemptions, we present an illustrative example, showing how Algorithm 1 actually reduces preemptions.

Example 1 Consider a task set τ with three tasks $\{\tau_1(T_1 = 4, C_1 = 1), \tau_2(12, 4), \tau_3(20, 3)\}$, and suppose that jobs of the tasks are periodically released starting from $t = 0$. Then, the execution order in [0, 10) under EDF or RM is shown in the upper figure of Fig. 1. Here, the total number of preemptions of τ in [0, 10) under EDF or RM is 2; the second job of τ_1 preempts the first job of τ_2 at $t = 4$, and the third job of τ_1 does the first job of τ_3 at $t = 8$. However, if we apply Algorithm 1 with a dummy task $\tau_x(T_x = 4, C_x = 1)$, the execution order in [0, 10) under EDF-d or RM-d is shown in the lower figure of Fig. 1, where no preemption occurs. That is, the currently-executing job of τ_2 at $t = 4$ (likewise τ_3 at $t = 8$) inherits the priority of a job of the dummy task, and finishes its execution without any preemption.

As shown in Lemma 1 and Example 1, we can effectively reduce preemptions by Algorithm 1, which delays the execution of τ_1's jobs through the virtual execution of dummy jobs. However, such a delay may make a schedulable task set unschedulable. In the next section, we will discuss how to set C_x (the virtual execution time of the dummy task) without compromising the schedulability of a given task set.

4 Dummy task parameter setting

This section describes how to set the execution time of the dummy task for EDF-d and RM-d, without compromising the schedulability of a given task set.

4.1 Generation of a dummy task for EDF-d

To guarantee the schedulability under EDF, we use the following existing exact schedulability condition.

Lemma 2 (Theorem 7 in [3]) *Under EDF with any arbitrary tie-breaking rule, a task set τ is schedulable if and only if the following conditions holds:*

$$\sum_{\tau_i \in \tau} \frac{C_i}{T_i} \leq 1. \tag{1}$$

Using Lemma 2, we determine the execution time of the dummy task as follows.

Theorem 1 *Suppose that* Lemma 2 *guarantees a task set τ to be schedulable by EDF. Then, τ is also schedulable by EDF-d, if the dummy task τ_x is set as follows:*

$$C_x \leq \left(1 - \sum_{\tau_i \in \tau} \frac{C_i}{T_i}\right) \cdot T_x. \tag{2}$$

Proof If we focus on $\tau \cup \{\tau_x\}$, the following conditions holds.

$$\frac{C_x}{T_x} + \sum_{\tau_i \in \tau} \frac{C_i}{T_i} \leq \frac{\left(1 - \sum_{\tau_i \in \tau} \frac{C_i}{T_i}\right) \cdot T_x}{T_x} + \sum_{\tau_i \in \tau} C_i/T_i = 1. \tag{3}$$

Therefore, by Lemma 2, $\tau \cup \{\tau_x\}$ is schedulable by EDF.

If we compare the schedule of τ under EDF-d, with that of $\tau \cup \{\tau_x\}$ under EDF, the finishing time of any job in the former is equal to, or earlier than that of the same job in the latter. Therefore, since $\tau \cup \{\tau_x\}$ is schedulable under EDF, τ is schedulable under EDF-d. Thus, the theorem follows. □

In Sect. 5, we will demonstrate the effectiveness of EDF-d in terms of the number of preemptions, by setting C_x to the RHS of Eq. (2), i.e., the largest possible C_x.

4.2 Generation of a dummy task for RM-d

For RM, we use the following existing exact schedulability condition.

Lemma 3 (Fig. 3 in [22]) *Under RM, a task set τ is schedulable if and only if every task τ_k satisfies $R_k^{s^*} \leq T_k$ such that $R_k^{s^*+1} \leq R_k^{s^*}$ for some s^*, starting from $R_k^0 = C_k$:*

$$R_k^{s+1} \leftarrow C_k + \sum_{\tau_i \in \mathsf{HP}(\tau_k, \tau)} \left\lceil \frac{R_k^s}{T_i} \right\rceil \cdot C_i, \tag{4}$$

where $\mathsf{HP}(\tau_k, \tau)$ *denotes a set of tasks in* τ *whose priority is higher than* τ_k*, i.e.,* $\mathsf{HP}(\tau_k, \tau) \triangleq \{\tau_i \in \tau | T_i \le T_k\}$.

Following Lemma 3, we set the execution time of the dummy task as follows.

Theorem 2 *Suppose that* Lemma 3 *guarantees a task set* τ *to be schedulable by RM. Then,* τ *is also schedulable by RM-d, if the dummy task* τ_x *is set as follows: every task* $\tau_k \in \tau$ *satisfies* $R_k^{s^*} \le T_k$ *such that* $R_k^{s^*+1} \le R_k^{s^*}$ *for some* s^**, starting from* $R_k^0 = C_k$*:*

$$R_k^{s+1} \leftarrow C_k + \left\lceil \frac{R_k^s}{T_x} \right\rceil \cdot C_x + \sum_{\tau_i \in \mathsf{HP}(\tau_k, \tau)} \left\lceil \frac{R_k^s}{T_i} \right\rceil \cdot C_i. \quad (5)$$

Proof The proof is similar to that of Theorem 1. If we compare the schedule of τ under RM-d, with that of $\tau \cup \{\tau_x\}$ under RM, the finishing time of any job in the former is equal to or earlier than that of the same job in the latter. Therefore, if $\tau \cup \{\tau_x\}$ is schedulable under RM, τ is schedulable under RM-d. By Lemma 3, the condition in Theorem 2 is the exact schedulability condition of a task set $\tau \cup \{\tau_x\}$ under RM. This proves the theorem. □

In Sect. 5, we will demonstrate the reduction of the number of preemptions by RM-d using the largest possible C_x, which is calculated by applying the binary search to Theorem 2.

5 Evaluation and discussion

This section compares the number of preemptions under EDF-d and RM-d, with that under EDF and RM.

Task set generation We generate task sets based on a widely used technique [23,24]. To make a variety of task sets, we consider two task parameters: task utilization (C_i/T_i) and the maximum task period (T_{max}). First, we consider 10 individual task utilization (C_i/T_i) distributions: bimodal with parameters 0.1, 0.3, 0.5, 0.7 and 0.9, and exponential with parameters 0.1, 0.3, 0.5, 0.7 and 0.9. For a given bimodal parameter p, a value for C_i/T_i is uniformly distributed in [0, 0.5) with probability p, and in [0.5, 1] with probability $1 - p$. For a given exponential parameter $1/\lambda$, a value for C_i/T_i is chosen according to an exponential distribution whose probability density function is $\lambda \cdot \exp(-\lambda \cdot x)$. Second, we consider two different maximum task periods: $T_{max} = 10$ and $T_{max} = 1000$. Then, for each task, T_i is uniformly chosen in $[1, T_{max}]$, and C_i is chosen based on the given bimodal or exponential parameter. Note that we set T_i and C_i to the closest positive integer values.

For a given bimodal or exponential parameter and a given T_{max}, we repeat the following procedure and generate 10,000 task sets, yielding 200,000 task sets in total.

1. We generate a set of two tasks since a task set consisting of a single task is trivially schedulable.
2. In order to exclude unschedulable task sets, we check the generated task set τ can pass the exact feasibility condition, i.e., $\sum_{\tau_i \in \tau} C_i/T_i \le 1$ [3].

3. If it fails to pass the feasibility test, we discard the generated task set and return to Step 1. Otherwise, we include this set for evaluation. Then, this set serves as a basis for the next new set; we create a new set by adding a new task into an already created and tested set, and return to Step 2.

Average performance comparison To compare the number of preemptions under EDF with EDF-d, and RM with RM-d, we simulate each task set with periodic job releases from $t = 0$ and on. Then, we measure the number of preemptions under each scheduling algorithm until 2520 time units when $T_{max} = 10$, which is a common multiple of task periods in any task set with $T_{max} = 10$. Then, the preemption behavior (as well as schedules) of each task set with $T_{max} = 10$ for the first 2520 time units is repeated forever. For task sets with $T_{max} = 1000$, it is intractable to simulate each task set up to the least common multiple of its task periods, and therefore, we measure the number of preemptions until 100,000 time units for task sets with $T_{max} = 1000$.

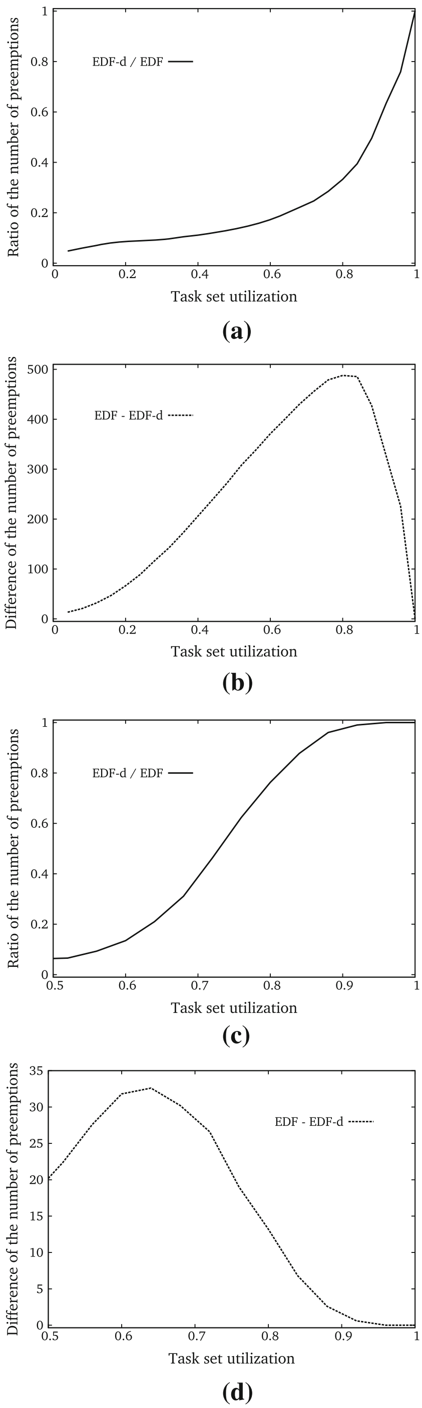
Then, Fig. 2 compares the number of preemptions of EDF-d with that of EDF, and RM-d with RM, in terms of their ratio and difference. In each figure, the x-axis represents task set utilization, i.e., $\sum_{\tau_i \in \tau} C_i / T_i$. Note that the smallest possible contribution of a task to task set utilization is 0.1 for $T_{max} = 10$ (when $C_i = 1$ and $T_i = 10$). Since the number of tasks in each task set is at least two, the number of generated task sets whose task set utilization is in $[0, 0.5]$ is small for $T_{max} = 10$. Therefore, we only show a partial range of task set utilization in the x-axis for $T_{max} = 10$, i.e., [0.5, 1.0], while we present the entire range for $T_{max} = 1000$, i.e., [0.0, 1.0].

Figure 2a, c show the ratio of the number of preemptions of EDF-d to that of EDF. As shown in the figures, when task set utilization is low, EDF-d significantly reduces the number of preemptions in terms of the ratio. This is because EDF-d can accommodate the dummy task with a large C_x, and then in most cases, a job of τ_1 does not resume its execution before the completion of the job that inherits the priority of the dummy job. As the task utilization gets higher, we have a smaller C_x, resulting in higher chance for a job of τ_1 to preempt the currently-executing job when the job of τ_1 resumes its execution. In the extreme case of task set utilization equal to 1.0, we cannot accommodate any positive value of C_x, and therefore, the number of preemptions under EDF-d is the same as that under EDF.

Figure 2b, d show the quantitative difference of the number of preemptions under EDF-d and EDF. For low task set utilization, EDF-d cannot reduce a larger number of preemptions, because the number of preemptions under EDF itself is not substantial in this environment. Therefore, although the ratio of the number of preemptions of EDF-d to EDF is increasing, the absolute value for EDF-d to reduce the number of preemptions is increasing up to a certain point; in the figures, the maximum point is around 0.8 when $T_{max} = 1000$ and 0.65 when $T_{max} = 10$. Beyond this point, it is difficult to reduce the number of preemptions due to a small (or even zero) C_x.

For the comparison of RM-d and RM, we only focus on RM-schedulable task sets by Lemma 3, resulting in 88,483 task sets out of 100,000 task sets with $T_{max} = 1000$ and 94,476 task sets out of 100,000 sets with $T_{max} = 10$. Then, the difference between the preemption behavior of RM-d and RM is similar to that between the preemption behavior of EDF-d and EDF. As shown in Fig. 3a, c, the ratio of the number of preemptions of RM-d to RM is increasing as the task utilization gets larger; if task set

Fig. 2 Comparison of the number of preemptions under EDF-d with EDF. **a** Ratio of the number of preemptions between EDF-d and EDF when $T_{max} = 1000$. **b** Difference of the number of preemptions between EDF and EDF-d when $T_{max} = 1000$. **c** Ratio of the number of preemptions between EDF-d and EDF when $T_{max} = 10$. **d** Difference of the number of preemptions between EDF and EDF-d when $T_{max} = 10$



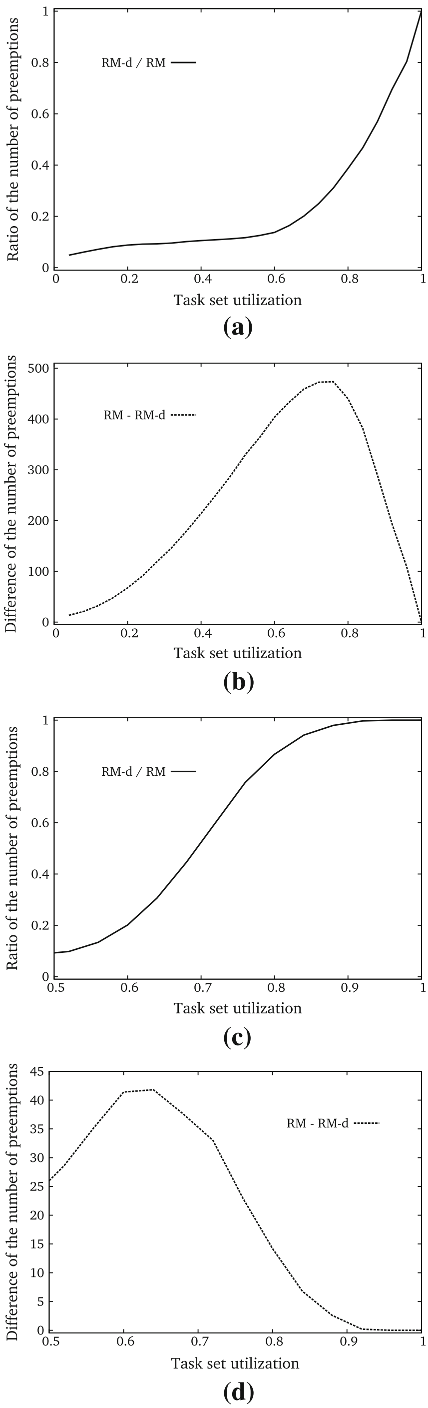


Fig. 3 Comparison of the number of preemptions under RM-d and RM. **a** Ratio of the number of preemptions between RM-d and RM when $T_{max} = 1000$. **b** Difference of the number of preemptions between RM and RM-d when $T_{max} = 1000$. **c** Ratio of the number of preemptions between RM-d and RM when $T_{max} = 10$. **d** Difference of the number of preemptions between RM and RM-d when $T_{max} = 10$

utilization equals to 1.0, the number of preemptions under RM-d is the same as that under RM. In Fig. 3b, d, the number of preemptions reduced by RM-d is maximized when task set utilization is around 0.7 when $T_{max} = 1000$ and 0.65 when $T_{max} = 10$.

In summary, EDF-d and RM-d effectively reduce the number of preemptions, compared to the corresponding scheduling algorithms EDF and RM. Also, we observe that the resource saved by EDF-d and RM-d is maximized with a certain task utilization, which varies with scheduling and task set specification. Then, the saved resource by reducing the number of preemptions can be utilized to enhance system performance, e.g., accommodation of more non-real-time tasks, and/or quick response of non-real-time tasks.

6 Conclusion

We proposed a simple but effective scheduling framework that incurs fewer preemptions, and applied the framework to two popular scheduling algorithms, EDF and RM, yielding EDF-d and RM-d. We not only proved that the framework does not increase the number of preemptions for any task set, but also demonstrated via simulation that EDF-d and RM-d effectively reduces the number of preemptions, compared to EDF and RM.

While the framework targets uniprocessor platforms, the dummy-task-based scheduling concept can be extended to multiprocessor platforms. It would be interesting to generalize the framework for multiprocessor platforms and global scheduling algorithms that are specialized for the platforms, e.g., EDZL [25], SPDF [26].

Acknowledgements This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016R1D1A1B03930580) and the Ministry of Science, ICT & Future Planning (NRF-2017R1A2B2002458, NRF-2014R1A1A103-5827).

References

1. Baskiyar S, Huang C-C, Tam T-Y (2016) Minimum energy consumption for rate monotonic scheduled tasks. Computing 98(6):661–684
2. Kim SC, Lee S (2015) Decentralized task scheduling for a fixed priority multicore embedded rtos. Computing 97(6):543–555
3. Liu C, Layland J (1973) Scheduling algorithms for multi-programming in a hard-real-time environment. J ACM 20(1):46–61
4. Buttazzo G, Bertogna M, Yao G (2013) Limited preemptive scheduling for real-time systems: a survey. IEEE Trans Ind Inf 9(1):3–13
5. Wang Y, Saksena M (1999) Scheduling fixed-priority tasks with preemption threshold, In: Proceedings of IEEE international conference on embedded and real-time computing systems and applications (RTCSA), pp 318–335
6. Kim S, Hong S, Kim T-H (2002) Integrating real-time synchronization schemes into preemption threshold scheduling, In: Proceedings of the 5th IEEE international symposium on object-oriented real-time distributed computing, pp 145–152
7. Kim S, Hong S, Kim T-H (2002) Perfecting preemption threshold scheduling for object-oriented real-time systems design: from the perspective of real-time synchronization, In: Proceedings of languages, compilers, and tools for embedded systems, pp 223–232

8. Bril RJ, van den Heuvel MM, Keskin U, Lukkien JJ (2012) Generalized fixed-priority scheduling with limited preemptions, In: Proceedings of Euromicro conference on real-time systems (ECRTS), pp 209–220
9. Baruah S (2005) The limited-preemption uniprocessor scheduling of sporadic task systems, In: Proceedings of Euromicro conference on real-time systems (ECRTS), pp 137–144
10. Bril RJ, Lukkien JJ, Verhaegh WFJ (2007) Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption revisited. In: Proceedings of the 19th Euromicro conference on real-time systems, pp 269–279
11. Yao G, Buttazzo G, Bertogna M (2009) Bounding the maximum length of non-preemptive regions under fixed priority scheduling, In: Proceedings of IEEE international conference on embedded and real-time computing systems and applications (RTCSA), pp 351–360
12. Bertogna M, Baruah S (2010) Limited preemption EDF scheduling of sporadic task systems. IEEE Trans Ind Inf 6(4):579–591
13. Davis RI, Bertogna M (2012) Optimal fixed priority scheduling with deferred pre-emption, In: Proceedings of IEEE real-time systems symposium (RTSS), pp 39–50
14. Dobrin R, Fohler G (2004) Reducing the number of preemptions in fixed priority scheduling. In: Proceedings of the 16th Euromicro conference on real-time systems, pp 144–152
15. Kim W, Kim J, Min SL (2004) Preemption-aware dynamic voltage scaling in hard real-time systems. In: Proceedings of the 2004 international symposium on low power electronics and design, pp 393–398
16. Yang C-Y, Chen J-J, Kuo T-W (2007) Preemption control for energy-efficient task scheduling in systems with a dvs processor and non-dvs devices. In: Proceedings of the 13th IEEE international conference on embedded and real-time computing systems and applications, pp 293–300
17. Yang L, Lin M, Yang LT (2009) Integrating preemption threshold to fixed priority dvs scheduling algorithms. In: Proceedings of the 15th IEEE international conference on embedded and real-time computing systems and applications, pp 165–171
18. Thekkilakattil A, Pillai AS, Dobrin R, Punnekkat S (2010) Reducing the number of preemptions in real-time systems scheduling by cpu frequency scaling. In: Proceedings of the 18th international conference on real-time and network systems, pp 144–152
19. Bertogna M, Buttazzo G, Marinoni M, Caccamo M (2010) Preemption points placement for sporadic task sets. In: Proceedings of Euromicro conference on real-time systems (ECRTS), pp 251–260
20. Bertogna M, Xhani O, Marinoni M, Esposito F, Buttazzo G (2011) Optimal selection of preemption points to minimize preemption overhead. In: Proceedings of Euromicro conference on real-time systems (ECRTS), pp 217–227
21. Lee J, Shin KG (2014) Preempt a job or not in EDF scheduling of uniprocessor systems. IEEE Trans Comput 63(5):1197–1206
22. 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
23. Baker TP (2005) Comparison of empirical success rates of global vs. paritioned fixed-priority and EDF scheduling for hard real time. Tech. Rep. TR-050601, Dept. of Computer Science, Florida State University, Tallahasee
24. Lee J, Easwaran A, Shin I (2011) Maximizing contention-free executions in multiprocessor scheduling. In: Proceedings of IEEE real-time technology and applications symposium (RTAS), pp 235–244
25. Lee SK (1994) On-line multiprocessor scheduling algorithms for real-time tasks. In: IEEE Region 10's ninth annual international conference, pp 607–611
26. Chwa HS, Back H, Chen S, Lee J, Easwaran A, Shin I, Lee I (2012) Extending task-level to job-level fixed priority assignment and schedulability analysis using pseudo-deadlines. In: Proceedings of IEEE real-time systems symposium (RTSS), pp 51–62