Time-Reversibility of Schedulability Tests

Jinkyu Lee
Department of Computer Science and Engineering
Sungkyunkwan University (SKKU), Republic of Korea
jinkyu.lee@skku.edu

***Abstract*—For timing guarantees of a set of real-time tasks under a target scheduling algorithm, a number of schedulability tests have been studied. However, there still exist many task sets that are potentially schedulable by a target scheduling algorithm, but proven schedulable by none of existing schedulability tests, especially on a multiprocessor platform. In this paper, we propose a new notion of *time-reversibility* of schedulability tests, which yields tighter schedulability guarantees by viewing real-time scheduling *under a change in the sign of time*. To this end, we first define the notion of a *time-reversed scheduling algorithm* against a target scheduling algorithm; for example, the time-reversed scheduling algorithm against EDF (Earliest Deadline First) is LCFS (Last-Come, First-Served), and the converse also holds. Then, a schedulability test for a scheduling algorithm is said to be *time-reversible with respect to schedulability*, if all task sets deemed schedulable by the test are also schedulable by its time-reversed scheduling algorithm. To exploit the notion of time-reversibility for tighter schedulability guarantees, we not only prove time-reversibility of an existing schedulability test, but also develop a new time-reversible schedulability test, both of which cover additional schedulable task sets.**

Next, we generalize the time-reversibility theory towards *partial execution*. Utilizing the notion, we can assure the schedulability of a task under a target scheduling algorithm in a divide-and-conquer manner: (i) the first some units of execution guaranteed by a schedulability test for the scheduling algorithm, and (ii) the remaining execution guaranteed by a time-reversible (with respect to partial execution) schedulability test for its time-reversed scheduling algorithm. Such a divide-and-conquer approach has not been directly applied to existing schedulability tests in that they cannot address (ii) effectively. As a case study, this paper develops RTA (Response-Time Analysis) for LCFS, proves its time-reversibility, and applies the divide-and-conquer approach to the test along with an existing EDF schedulability test. Our simulation results show that the time-reversibility theory helps to find up to 13.1% additional EDF-schedulable task sets on a multiprocessor platform.

I. Introduction

In order to satisfy timing requirements of real-time systems, scheduling algorithms and their schedulability tests have been substantially studied. While a scheduling algorithm determines the order of execution of a series of jobs invoked by a set of real-time recurring tasks, its schedulability test judges whether all jobs satisfy their timing requirements under any permissible job release patterns. Since it is often challenging to develop an exact schedulability test, different sufficient schedulability tests have been developed for the same scheduling algorithm. That is, a newly-developed schedulability test covers additional schedulable task sets that are deemed schedulable by none of the existing schedulability tests. For example, preemptive EDF (Earliest Deadline First) [1], one of the most popular scheduling algorithms, has a number of schedulability tests on a multiprocessor platform (see a survey [2]). However, there is still room for tighter schedulability guarantees in that there exists no known exact schedulability test for many scheduling algorithms, e.g., preemptive EDF and RM (Rate Monotonic) [1] on a multiprocessor platform.

While existing studies focus on scheduling of a series of jobs in a time-ordered manner, we may investigate that *under a change in the sign of time*. To this end, we construct a job $\overline{J_i^{-q}}$ that corresponds a given job J_i^q as follows: (i) $\overline{J_i^{-q}}$'s deadline is set to J_i^q's release time under a change in the plus-minus sign, (ii) $\overline{J_i^{-q}}$'s release time is set to J_i^q's deadline under a change in the sign, and (iii) the priority of $\overline{J_i^{-q}}$ is set to that of J_i^q. Fig. 1 shows an example; since the release time and deadline of J_i^2 are 10 and 18, respectively, the release time and deadline of $\overline{J_i^{-2}}$ are -18 and -10, respectively. Then, for a given scheduling algorithm G that prioritizes J_i^q, a scheduling algorithm that prioritizes $\overline{J_i^{-q}}$ is said to be a time-reversed scheduling algorithm against G (denoted by $\overline{G}$). For example, since EDF gives the highest priority to a job with the *earliest deadline*, a time-reversed scheduling algorithm against EDF is LCFS (Last-Come, First-Served), which assigns the highest priority to a job with the *latest release time*; the converse also holds.

Then, we investigate a connection between a time-reversed scheduling algorithm against G (i.e., $\overline{G}$) and a schedulability test for G in terms of schedulability, and define the time-reversibility as follows: a schedulability test A_G for a scheduling algorithm G is said to be *time-reversible with respect to schedulability*, if all task sets deemed schedulable by A_G are also schedulable by $\overline{G}$. To utilize the notion of time-reversibility for finding additional task sets schedulable by $\overline{G}$, we identify the following issues to be addressed.

- I1. Can we find an existing time-reversible schedulability test for G?
- I2. Furthermore, can we develop a new time-reversible schedulability test for G?
- I3. If the answer of I1 or I2 is positive, can we demonstrate that a time-reversible schedulability test A_G for G covers additional schedulable task sets, which are not deemed schedulable by any existing schedulability test for $\overline{G}$?

To address I1–I3, we investigate a popular scheduling algorithm EDF and its time-reversed scheduling algorithm LCFS on a multiprocessor platform. We prove that a popular schedulability test, RTA (Response-Time Analysis) for EDF is not only time-reversible with respect to schedulability, but also

1052-8725/14 $31.00 © 2014 IEEE
DOI 10.1109/RTSS.2014.18

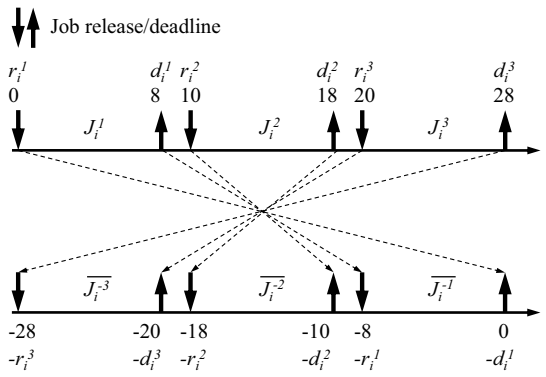

Fig. 1. Jobs under a scheduling algorithm G and the corresponding jobs under its time-reversed scheduling algorithm $\overline{G}$

capable of finding additional task sets schedulable by LCFS, which addresses I1 and I3. Also, we develop a new time-reversible schedulability test for LCFS, and demonstrate the test can cover additional EDF-schedulable task sets that are deemed schedulable by none of existing schedulability tests for EDF, which addresses I2 and I3.

While we successfully exploit the notion of time-reversibility for tighter schedulability guarantees, we can further benefit from the notion. To this end, we generalize the notion of time-reversibility towards *partial execution*. A schedulability test for a scheduling algorithm G is said to be *time-reversible with respect to partial execution*, if the following statement holds: if the test guarantees that every job of a task under G executes X time units between its release time and that after ℓ time units, it is guaranteed that every job of the task under $\overline{G}$ executes X time units between its deadline ahead of ℓ time units and the deadline (see Fig. 3). Then, each job's execution under $\overline{G}$ can be guaranteed by two schedulability tests; a schedulability test for $\overline{G}$ guarantees the first some units of execution, and a time-reversible schedulability test for G guarantees the remaining execution. As an example, we demonstrate that a collaboration between RTA for EDF and RTA for LCFS (that is time-reversible with respect to partial execution) results in covering additional EDF-schedulable task sets, which are not deemed schedulable by both schedulability tests.

While such a divide-and-conquer approach is effective in improving schedulability guarantees, it has not been achieved without the notion of time-reversibility with respect to partial execution. This is because, most schedulability tests cannot guarantee partial execution of a job in an interval between an arbitrary time instant and its deadline. Motivated by this, we further improve the schedulability test for EDF, which directly applies the divide-and-conquer approach without relying on the notion of time-reversibility.

To demonstrate quantitative schedulability improvement by the notion of time-reversibility, we generate a large number of task sets, and count the number of task sets proven schedulable by our schedulability tests motivated by the notion. The simulation results show that our schedulability tests can find up to 13.1% additional schedulable task sets that are not covered by the best existing EDF schedulability test on a multiprocessor platform. Also, a new schedulability test for LCFS covers up to 6.8% additional LCFS-schedulable task sets.

In summary, this papers makes the following contributions:

- Introduction of the notion of time-reversibility for real-time scheduling,
- Establishment of the theoretical foundation of time-reversibility towards schedulability guarantee improvement,
- Suggestion of a new direction of schedulability tests, called a divide-and-conquer approach, which is inspired by the notion of time-reversibility,
- Application of the time-reversibility theory to preemptive EDF, demonstrating the effectiveness of the notion in improving schedulability guarantees, and
- Demonstration of quantitative improvement on schedulability guarantees via simulation.

The rest of this paper is organized as follows. Section II explains our system model, assumptions, and notations. Section III introduces the notion of time-reversibility, and Section IV presents how the notion improves schedulability with a case study. Section V presents more general theory of time-reversibility, and points out a new direction of developing schedulability tests. Section VI evaluates schedulability guarantee improvement by the notion of time-reversibility via simulations. Finally, Section VII concludes the paper.

II. System Model

In this paper, we consider a sporadic real-time task model [3], in which a task $\tau_i \in \tau$ is specified by (T_i, C_i, D_i), where T_i is the minimum separation, C_i is the worst-case execution time, and D_i is the relative deadline. We focus on constrained deadline tasks, which satisfy $D_i \leq T_i$. We assume a quantum-based time; let the length of a quantum be one time unit, without loss of generality. All task parameters are multiples of the quantum.

A task τ_i invokes a series of jobs, each separated from its predecessor by at least T_i time units. Each job of τ_i, once released, should finish its execution within D_i time units. The q^{th} job of τ_i is denoted by J_i^q, and the release time and deadline of J_i^q are denoted by r_i^q and d_i^q, respectively (where $d_i^q = r_i^q + D_i$).

In this paper, we consider a computing platform consisting of m identical processors, where m is an integer value. For the ease of presentation, we will not specify the computing platform when no ambiguity arises in the rest of the paper.

When it comes to scheduling algorithms, this paper focuses on preemptive work-conserving scheduling algorithms, in which a higher-priority job can preempt a lower-priority job at any time, and any processor cannot be left idle as long as there is an unfinished job in the system.

III. Time-Reversibility

Time-reversibility is a widely-used concept in a stochastic/deterministic process, meaning that properties of interest hold under a change in the sign of time [4]. Since our

primary interest is schedulability guarantees of a set of real-time tasks, this section discusses time-reversibility with respect to schedulability. To this end, we first introduce the notion of a time-reversed scheduling algorithm. Then, we formally define time-reversibility of scheduling algorithms as well as that of schedulability tests. Finally, we bring up technical issues in order to exploit the notion of time-reversibility for schedulability guarantees.

A. Time-reversed scheduling algorithms

Suppose that a series of jobs invoked by τ (denoted by $\{J_i^q\}_{\tau_i \in \tau}$) is executed by a scheduling algorithm G. We now look at $\{J_i^q\}_{\tau_i \in \tau}$ *under a change in the sign of time*. To this end, we synthesize another series of jobs (denoted by $\{\overline{J_i^{-q}}\}_{\tau_i \in \tau}$), which is a one-to-one mapping of $\{J_i^q\}_{\tau_i \in \tau}$ as follows.

- R1. The release time of $\overline{J_i^{-q}}$ is set to $-d_i^q$; recall that d_i^q denotes the deadline of J_i^q.
- R2. The deadline of $\overline{J_i^{-q}}$ is set to $-r_i^q$; recall that r_i^q denotes the release time of J_i^q.
- R3. The worst-case execution time of $\overline{J_i^{-q}}$ is set to that of J_i^q.
- R4. The priority of $\overline{J_i^{-q}}$ is set to that of J_i^q.

For example, since the release time of J_i^2 in Fig. 1 is $r_i^2 = 10$, the deadline of $\overline{J_i^{-2}}$ (corresponding to J_i^2) is -10. Likewise, since the deadline of J_i^2 in the same figure is $d_i^2 = 18$, the release time of $\overline{J_i^{-2}}$ is -18.

Note that $\{\overline{J_i^{-q}}\}_{\tau_i \in \tau}$ is also an instance of a series of jobs invoked by τ in that it follows all the task parameters of τ. Then, execution of $\{\overline{J_i^{-q}}\}_{\tau_i \in \tau}$ corresponds to that of $\{J_i^q\}_{\tau_i \in \tau}$ reversely in time.[1] If we pay attention to two scheduling algorithms that prioritize $\{J_i^q\}_{\tau_i \in \tau}$ and $\{\overline{J_i^{-q}}\}_{\tau_i \in \tau}$, there is a relationship between the two, defined as follows.

Definition 1: Suppose that for a given $\{J_i^q\}_{\tau_i \in \tau}$ which is prioritized by a scheduling algorithm G, $\{\overline{J_i^{-q}}\}_{\tau_i \in \tau}$ is generated according to R1–R4. Then, we can derive a corresponding scheduling algorithm $\overline{G}$, such that $\overline{G}$ directly assigns job priorities to $\{\overline{J_i^{-q}}\}_{\tau_i \in \tau}$. A scheduling algorithm $\overline{G}$ is said to be a *time-reversed* scheduling algorithm against G.

Here we present two examples of $\overline{G}$ for a given G.

Observation 1: Since J_i^q's deadline matches $\overline{J_i^{-q}}$'s release time under a change in the plus-minus sign, scheduling of $\{J_i^q\}_{\tau_i \in \tau}$ by EDF (that gives the highest priority to a job with the *earliest deadline*) corresponds to that of $\{\overline{J_i^{-q}}\}_{\tau_i \in \tau}$ by a scheduling algorithm that gives the highest priority to a job with the *latest release time*, which is LCFS (Last-Come, First-Served). In other words, LCFS is a time-reversed scheduling algorithm against EDF (denoted by $\overline{\text{EDF}} = \text{LCFS}$). Similarly, $\overline{\text{LCFS}} = \text{EDF}$ holds.

[1] Here the meaning of the verb "correspond" is not "be equivalent," but "be similar or analogous."

Observation 2: Scheduling of $\{J_i^q\}_{\tau_i \in \tau}$ by RM (*likewise* DM) corresponds that of $\{\overline{J_i^{-q}}\}_{\tau_i \in \tau}$ by the same scheduling algorithm RM (*likewise* DM), because the priority of a job does not depend on its release time and deadline. In other words, $\overline{\text{RM}} = \text{RM}$ (*likewise* $\overline{\text{DM}} = \text{DM}$) holds.

B. Time-reversibility of scheduling algorithms

Since we are interested in schedulability guarantees, we need to establish a relationship between a scheduling algorithm G and its time-reversed one $\overline{G}$ in terms of schedulability, which is expressed as the notion of *time-reversibility* as follows.

Definition 2: A scheduling algorithm G is said to be *time-reversible with respect to schedulability*, if all task sets schedulable by G are also schedulable by $\overline{G}$.

Then, we can easily decide time-reversibility of existing scheduling algorithms, as shown in the following observations.

Observation 3: RM and DM are time-reversible with respect to schedulability. This is because, $\overline{\text{RM}} = \text{RM}$ and $\overline{\text{DM}} = \text{DM}$ hold as shown in Observation 2.

Observation 4: EDF and LCFS are not time-reversible with respect to schedulability. This is because, while $\overline{\text{EDF}} = \text{LCFS}$ and $\overline{\text{LCFS}} = \text{EDF}$ hold as shown in Observation 1, we can easily find a task set that is schedulable by LCFS but unschedulable by EDF (on a multiprocessor platform), and another task set that is schedulable by EDF but unschedulable by LCFS.

Once we find a time-reversible scheduling algorithm G satisfying $G \neq \overline{G}$, the notion of time-reversibility associated with G helps find task sets schedulable by $\overline{G}$. This comes from the definition of time-reversibility: a task set schedulable by G is also schedulable by $\overline{G}$. However, it is challenging (if not impossible) to find a time-reversible scheduling algorithm G which is different from $\overline{G}$. As a result, the notion of time-reversibility of *scheduling algorithms* may not be effective in improving schedulability guarantees. The next subsection discusses the notion of *time-reversibility of schedulability tests* (rather than that of scheduling algorithms), and then Section IV demonstrates how the notion can improve schedulability guarantees with a concrete example.

C. Time-reversibility of schedulability tests

A schedulability test judges whether a task set is schedulable by a scheduling algorithm on a platform. Due to the challenge of finding exact deadline-miss conditions, only a few existing schedulability tests are necessary and sufficient, e.g., the response time analysis for RM (and DM) [5] and the demand-based schedulability test for EDF [6] on a uniprocessor platform, and the schedulability condition for a class of optimal scheduling algorithms for implicit deadline task sets on a multiprocessor platform [7–9].

Therefore, there exist many task sets which are potentially schedulable by a scheduling algorithm, but not proven schedulable by any existing schedulability test for the scheduling algorithm; for example, a number of schedulability tests for EDF have been developed to cover such potentially schedulable task sets on a multiprocessor platform [2]. We hope to validate potentially schedulable task sets (but not proven schedulable

by any existing schedulability test) using time-reversibility of schedulability tests, defined as follows.

Definition 3: A schedulability test A_G for a scheduling algorithm G is said to be *time-reversible with respect to schedulability*, if all task sets deemed schedulable by A_G are also schedulable by $\overline{G}$.

D. Can time-reversibility improve schedulability guarantees?

While Definition 3 has potential in finding additional task sets schedulable by $\overline{G}$, the potential is achieved only after we address I1–I3 presented in the introduction. This is because, without addressing I1 and I2, no time-reversible schedulability test exists, which makes the notion of time-reversibility be in name only. I3 should be also resolved; otherwise, the notion is ineffective in covering additional schedulable task sets. Once we achieve I1 (or I2) and I3, time-reversibility helps achieve our goal of finding additional task sets schedulable by $\overline{G}$, which is demonstrated in the next section with a case study.

IV. Case Study: Response-Time Analysis for EDF and LCFS

In this section, we demonstrate how the notion of time-reversibility improves schedulability guarantees. To achieve this, we address I1–I3, using a case study of a popular schedulability test for a popular scheduling algorithm and its time-reversed one: RTA (Response-Time Analysis) for EDF and LCFS. First, we prove that RTA for EDF is not only time-reversible, but also able to find additional task sets schedulable by LCFS. Second, we develop a new schedulability test, RTA for LCFS, and show that the new time-reversible schedulability test can cover additional EDF-schedulable task sets.

A. Discovery of an existing time-reversible schedulability test

While there are many existing time-reversible schedulability tests, we now present a popular existing schedulability test framework, called RTA (Response Time Analysis) [5], which is known to have tight schedulability guarantees and wide applicability. That is, RTA yields an exact (i.e., sufficient and necessary) schedulability test for RM (DM) on a uniprocessor platform [5], and RTA for EDF and that for RM (DM) on a multiprocessor platform [10, 11] are (one of) the best among existing schedulability tests in terms of average schedulability guarantees [2, 11]. In this subsection, we explain RTA for EDF, and prove its time-reversibility.

The response time of J_i^q is defined as a duration between its release time and the completion time of its execution. Then, let J_i^* denote a job of τ_i whose response time is the longest among all jobs invoked by τ_i. RTA calculates the response time of J_i^* for every $\tau_i \in \tau$, and deems a task set schedulable if the response time of J_i^* is no larger than D_i for every $\tau_i \in \tau$.

To this end, RTA employs the concept of *interference*; the interference of τ_i on τ_k in $[a, b)$, denoted by $I_{k \leftarrow i}(a, b)$, means the amount of time a job of τ_k of interest cannot execute due to other higher-priority jobs, but jobs of τ_i execute in $[a, b)$ [10]. Then, RTA calculates the total interference of other tasks on τ_k in an interval between the release time of J_k^* (i.e., r_k^*) and a time instant later than r_k^* (i.e., $r_k^* + \ell$). Since a job cannot execute only when other m higher-priority jobs execute, J_k^* finishes its full execution in $[r_k^*, r_k^* + \ell)$ if the total interference of other tasks on J_k^* divided by m added to C_k is no larger than the interval of length ℓ, which means the response time of J_k^* is no larger than ℓ. Otherwise, we repeat the same process for a longer interval, which is expressed in Eq. (1) [10].

$$R_k^{x+1} \leftarrow C_k + \left\lfloor \frac{1}{m} \sum_{\tau_i \in \tau - \{\tau_k\}} \min\Big(I_{k \leftarrow i}(r_k^*, r_k^* + R_k^x), R_k^x - C_k + 1\Big) \right\rfloor. \tag{1}$$

Note that the initial value R_0 is set to C_k, and the repetition halts when $R_k^{x+1} > D_k$ (unschedulable) or $R_k^{x+1} = R_k^x$ (the response time no larger than R_k^x).

The remaining issue is to calculate $I_{k \leftarrow i}(r_k^*, r_k^* + \ell)$ under a target scheduling algorithm, but it is challenging to calculate the amount of interference exactly. Therefore, existing studies seek to find upper-bounds of the amount of interference, especially on a multiprocessor platform. There are in general two types of upper-bounds. Before explaining the two upper-bounds, we introduce S_i, which denotes the slack value of jobs of τ_i, meaning the minimum interval length between the finishing time and the deadline of any job of τ_i. In other words, any job of τ_i finishes its execution at least S_i time units ahead of its deadline, and therefore, a job of τ_i, J_i^q cannot be executed in $[d_i^q - S_i, d_i^q)$; see the first job in Fig. 2(a). We will explain how to calculate S_i later in this subsection.

Now we explain the first upper-bound that can be applied to any work-conserving scheduling algorithm. Since a job can interfere with another job only when the job executes, the amount of maximum execution of jobs of τ_i can be an upper-bound of interference of τ_i on τ_k. For a given interval of length ℓ, the execution of jobs of a task is maximized when the first job in the interval executes as late as possible and other jobs in the interval execute as early as possible; also, the beginning of the interval is a time instant when the first job starts its execution as shown in Fig. 2(a). Then, we count the number of jobs of τ_i executed in the interval except the last job (denoted by $N_i(\ell)$), which is calculated as follows [10].

$$N_i(\ell) = \left\lfloor \frac{\ell + D_i - C_i - S_i}{T_i} \right\rfloor. \tag{2}$$

For example, $N_i(\ell) = 2$ in Fig. 2(a). Here, $N_i(\ell)$ jobs perform its full execution in the interval of length ℓ, contributing to $N_i(\ell) \cdot C_i$. Along with computing the contribution of the last job, the amount of maximum execution of jobs of τ_i in an interval of length ℓ can be calculated by $W_i(\ell)$ as follows [10].

$$W_i(\ell) = N_i(\ell) \cdot C_i + \min\Big(C_i, \ell + D_i - C_i - S_i - N_i(\ell) \cdot T_i\Big). \tag{3}$$

For the second upper-bound, we use the property of the target scheduling algorithm. For example, under EDF, a job J_i^q can interfere with another job J_k^* only when the deadline of J_i^q is no later than that of J_k^*. Since we are interested in J_k^* whose interval of interest is $[r_k^*, d_k^*)$ of length D_k (from the release time to the deadline), the amount of time in $[r_k^*, d_k^*)$ jobs of τ_i can interfere with J_k^* is upper-bounded by the amount of execution of jobs of τ_i in an interval of length D_k in the following situation: the deadline of a job of τ_i is aligned to the

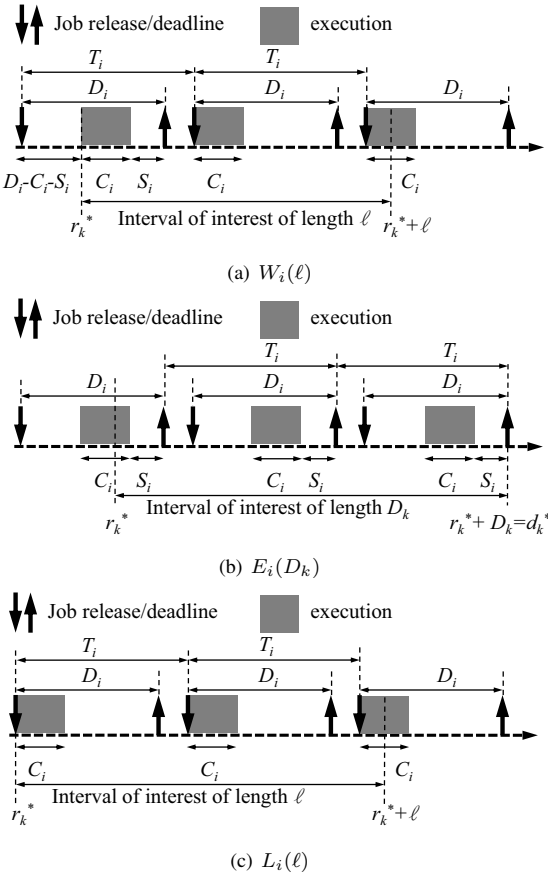

Fig. 2. Upper-bounds of interference: $W_i(\ell)$, $E_i(D_k)$ and $L_i(\ell)$

end of the interval, and all jobs of τ_i execute as late as possible as shown in Fig. 2(b). This is calculated by $E_i(D_k)$ [10], where

$$E_i(\ell) = \left\lfloor \frac{\ell}{T_i} \right\rfloor \cdot C_i + \max\left(0, \min\left(C_i, \ell - \left\lfloor \frac{\ell}{T_i} \right\rfloor \cdot T_i - S_i\right)\right). \tag{4}$$

In summary, we can upper-bound of $I^*_{k\leftarrow i}(r_k^*, r_k^* + \ell)$ in Eq. (1) under EDF as follows.

$$I_{k\leftarrow i}(r_k^*, r_k^* + \ell) \text{ in Eq. (1)} \leq \min\left(W_i(\ell), E_i(D_k)\right). \tag{5}$$

Then, we have two types of RTA for EDF: (a) without slack reclamation and (b) with slack reclamation [10]. For (a), S_i is statically set to 0 for all $\tau_i \in \tau$, and RTA for EDF calculates the response time of each task using Eq. (1) with an upper-bound of Eq. (5).

Suppose that the response time of τ_k is R_k by (a). Then, we know that any job of τ_k finishes its execution at least $D_k - R_k$ time units ahead of its deadline. Therefore, we update $S_k = D_k - R_k$ (if $D_k - R_k > 0$). For (b), RTA for EDF repeats (a) with updated slack values, until the response time of J_i^* is no larger than D_i for every $\tau_i \in \tau$ (schedulable) or there is no more update (unschedulable).

Then, we find out time-reversibility of RTA for EDF without slack reclamation, as stated in the following lemma.

Lemma 1: RTA for EDF without slack reclamation is time-reversible with respect to schedulability.

Proof: By Definition 3, we need to prove that a task set is schedulable by LCFS, as long as the task set is deemed schedulable by RTA for EDF without slack reclamation. To achieve this, we prove that $E_i(D_k)$ with $S_i = 0$ is no larger than the amount of time in $[r_k^*, d_k^*)$ jobs of τ_i can interfere with J_k^* when the scheduling algorithm is LCFS. Then, it holds that the actual response time under LCFS is upper-bounded by the response time calculated by RTA for EDF without slack reclamation.

Under LCFS, a job of τ_i can interfere with another job J_k^* only when the release time of the job of τ_i is no earlier than J_k^*. Therefore, the amount of interference of jobs of τ_i on J_k^* is maximized when the release time of the first job of τ_i is aligned with that of J_k^*. Then, the scenario that yields the maximum interference under LCFS shown in Fig. 2(c) is vertically symmetrical to the scenario of $E_i(D_k)$ with $S_i = 0$ in Fig. 2(b). This means, jobs of τ_i under LCFS interfere with J_k^* during at most $E_i(D_k)$ with $S_i = 0$, which proves the lemma. ■

As opposed to RTA for EDF without slack reclamation, that with slack reclamation is not time-reversible with respect to schedulability. An example is $\tau = \{\tau_1(T_1 = 4, C_1 = 3, D_1 = 4) = \tau_2, \tau_3(40, 3, 40)\}$ on two processors, which are deemed schedulable by RTA for EDF with slack reclamation, but not LCFS-schedulable.

Since RTA for EDF without slack reclamation is time-reversible, the schedulability test can potentially find additional schedulable task sets that are not deemed schedulable by any existing schedulability tests for its time-reversed scheduling algorithm, as discussed in the following lemma.

Lemma 2: RTA for EDF without slack reclamation covers additional task sets schedulable by LCFS, which are deemed schedulable by none of existing schedulability tests for LCFS.

Proof: To the best knowledge of the author, no schedulability test *specialized* for LCFS has been developed. Therefore, the best existing schedulability test to be applied to LCFS is the state-of-the-art schedulability test for any work-conserving (WC) scheduling algorithm. This is RTA for WC with slack reclamation, which employs $W_i(\ell)$ as an upper-bound of $I_{k\leftarrow i}(r_k^*, r_k^* + \ell)$ [10].

We can easily find task sets, which RTA for EDF without slack reclamation deems schedulable, but RTA for WC with slack reclamation does not. For example, while RTA for EDF without slack reclamation deems $\tau = \{\tau_1(T_1 = 2, C_1 = 1, D_1 = 2) = \tau_2 = \tau_3\}$ schedulable on a two-processor platform, RTA for WC with slack reclamation does not. ■

In summary, we address I1 and I3, which demonstrates that the notion of time-reversibility is effective in finding additional schedulable task sets. In the next subsection, we will address I2 and I3.

B. Development of a new time-reversible schedulability test

Section IV-A presented how the notion of time-reversibility can improve schedulability guarantees, focusing on an existing schedulability test *as it is*. That is, once we discover that

an existing schedulability test A_G for a scheduling algorithm G is time-reversible, the test can cover additional task sets schedulable by its time-reversed scheduling algorithm $\overline{G}$. This section utilizes the notion of time-reversibility on the other way around; we will develop a new time-reversible schedulability test for G to improve schedulability guarantees for a scheduling algorithm $\overline{G}$.

To this end, we develop a new schedulability test for LCFS (i.e., RTA for LCFS) and prove its time-reversibility. Then, we demonstrate that the test can find additional EDF-schedulable task sets which are not deemed schedulable by any existing EDF schedulability test including its corresponding EDF schedulability test (i.e., RTA for EDF).

We already discussed the generic framework of RTA in Section IV-A, and therefore, the remaining issue is to calculate an upper-bound of $I^*_{k\leftarrow i}(r^*_k, r^*_k+\ell)$ under LCFS. Under LCFS, a job J^q_i can interfere with another job J^*_k only when the release time of J^q_i is no earlier than that of J^*_k. Therefore, the amount of interference of jobs of τ_i on J^*_k in $[r^*_k, r^*_k+\ell)$ of length ℓ is maximized when the beginning of the interval coincides with the release time of the first job of τ_i in the interval and all jobs of τ_i in the interval execute as early as possible, as shown in Fig. 2(c). The amount of maximum interference of jobs of τ_i on J^*_k in $[r^*_k, r^*_k+\ell)$ of length ℓ is calculated by $L_i(\ell)$ as follows.

$$L_i(\ell) = \left\lfloor \frac{\ell}{T_i} \right\rfloor \cdot C_i + \min\left(C_i, \ell - \left\lfloor \frac{\ell}{T_i} \right\rfloor \cdot T_i\right). \tag{6}$$

Then, we can upper-bound of $I^*_{k\leftarrow i}(r^*_k, r^*_k+\ell)$ in Eq. (1) under LCFS as follows.

$$I_{k\leftarrow i}(r^*_k, r^*_k+\ell) \text{ in Eq. (1)} \le \min\left(W_i(\ell), L_i(\ell)\right) = L_i(\ell). \tag{7}$$

We make two observations for the upper-bounds, $W_i(\ell)$ and $L_i(\ell)$. First, $L_i(\ell)$ is always equal to or smaller than $W_i(\ell)$; in fact, they are equivalent when the slack value is the largest (i.e., $S_i = D_i - C_i$). Second, $L_i(\ell)$ is irrelevant to slack values. Therefore, $\min\left(W_i(\ell), L_i(\ell)\right) = L_i(\ell)$ holds, and there is the only unified RTA for LCFS in which slack reclamation is ineffective.

Similar to RTA for EDF without slack reclamation, RTA for LCFS is also time-reversible, as stated in the following lemma.

Lemma 3: RTA for LCFS is time-reversible with respect to schedulability. This means, if a task set is deemed schedulable by RTA for LCFS, the task set is actually schedulable by EDF.

Proof: We prove that $L_i(\ell)$ is no larger than the amount of time in $[d^*_k - \ell, d^*_k)$ jobs of τ_i can interfere with J^*_k when the scheduling algorithm is EDF; then, it holds that any job of τ_k under EDF does not miss its deadline as long as RTA for LCFS guarantees the schedulability of τ_k.

By definition, $L_i(\ell)$ in Eq. (6) is equal to $E_i(\ell)$ with $S_i = 0$ in Eq. (4). Since $E_i(\ell)$ with $S_i = 0$ is an upper-bound of the amount of interference of jobs of τ_i on J^*_k in $[d^*_k - \ell, d^*_k)$ under EDF, the lemma holds. ■

If we compared the RHS of Eq. (7) with that of Eq. (5), the difference is $E_i(D_k)$ against $L_i(\ell)$. That is, the upper-bound specialized for EDF focuses on the entire interval of J^*_k (i.e., $[r^*_k, d^*_k)$ of length D_k), because the alignment of the deadline of the last job of τ_i and the end of the interval is no longer valid if the interval of interest ends at an arbitrary time instant rather than the deadline of J^*_k (i.e., d^*_k). On the other hand, the upper-bound specialized for LCFS can handle an interval that ends at an arbitrary point because a job priority under LCFS depends solely on its release time, not on its deadline. Then, we can easily observe that $L_i(\ell)$ is always (*likewise* sometimes) smaller than or equal to $E_i(D_k)$ when the slack reclamation is not applied (*likewise* is applied). This means, RTA for LCFS is capable of covering additional task sets that are deemed schedulable by neither RTA for EDF without reclamation nor that with slack reclamation. The following lemma records this.

Lemma 4: RTA for LCFS can find additional EDF-schedulability task sets, which are not deemed schedulable by RTA for EDF with slack reclamation (as well as any other existing EDF schedulability tests).

Proof: Suppose that $\tau = \{\tau_1(T_1 = 3, C_1 = 1, D_1 = 3), \tau_2 = \tau_3 = \tau_4 = (2, 1, 2)\}$ is scheduled by EDF on a two-processor platform. Then, τ is deemed schedulable by RTA for LCFS, while it is not deemed schedulable by RTA for EDF with slack reclamation. Note that τ is not deemed schedulable by any single existing EDF schedulability test in a survey [2]. ■

This is a surprising result, in that RTA for EDF with slack reclamation is known to exhibit the best average performance of schedulability guarantees among all existing EDF schedulability tests on a multiprocessor platform [2]. Using the notion of time-reversibility, RTA for LCFS can find additional EDF-schedulable task sets, which are not covered by the best existing schedulability test for EDF, even if they share the same framework of RTA. In fact, RTA for LCFS finds additional EDF-schedulable task sets, not covered by any existing EDF schedulability test. This demonstrates the effectiveness of time-reversibility in finding additional schedulable task sets.

V. Advanced Time-Reversibility Theory

So far, Section III established theory of time-reversibility with respect to schedulability, and then Section IV presented how the theory can improve schedulability guarantees. This section presents more general theory of time-reversibility, called *time-reversibility with respect to partial execution*. Using the notion, we demonstrate how a time-reversible schedulability test A_G and a (not necessarily time-reversible) schedulability test $B_{\overline{G}}$ create a synergy effect in finding additional task sets schedulable by $\overline{G}$, which are deemed schedulable by neither A_G nor $B_{\overline{G}}$. Motivated by this, we also present a new direction of developing schedulability tests: a *divide-and-conquer* approach.

A. Time-reversibility with respect to partial execution

Section III investigated a series of jobs (denoted by $\{\overline{J^{-q}_i}\}_{\tau_i\in\tau}$) generated according to R1–R4, which corresponds to $\{J^q_i\}_{\tau_i\in\tau}$ prioritized by a scheduling algorithm G. Then, the notion of time-reversibility describes the *schedulability relationship* between a schedulability test for G and the time-reversed scheduling algorithm against G that prioritizes $\{\overline{J^{-q}_i}\}_{\tau_i\in\tau}$. Now, we establish a more general relationship

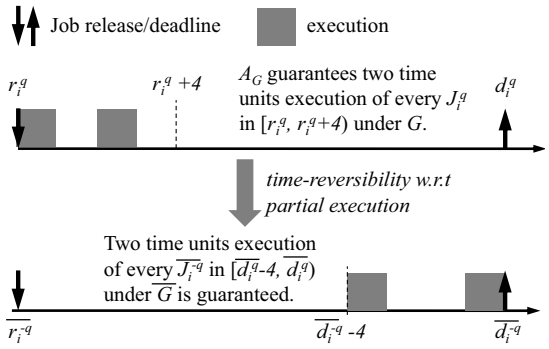

Fig. 3. Time-reversibility of a schedulability test A_G

between the two *in terms of partial execution*. That is, we make a connection between a part of execution of a job under G (guaranteed by a schedulability test for G) and that under the time-reversed scheduling algorithm against G (i.e., $\overline{G}$), which is defined as follows.

Definition 4: A schedulability test A_G for a scheduling algorithm G is said to be *time-reversible with respect to partial execution*, if the following condition holds for every $\tau_i \in \tau$, $C_i' \in [0, C_i]$, and $\ell \in [0, D_i]$:

- If A_G guarantees that the amount of execution of every job of τ_i under G (denoted by J_i^q) performed in $[r_i^q, r_i^q + \ell)$ is C_i', that of every job of τ_i under $\overline{G}$ (denoted by $\overline{J_i^{-q}}$) performed in $[\overline{d_i^{-q}} - \ell, \overline{d_i^{-q}})$ is equal to either (a) at least C_i' if the amount of the remaining execution of $\overline{J_i^{-q}}$ at $\overline{d_i^{-q}} - \ell$ is no smaller than C_i' or (b) the amount of the remaining execution of $\overline{J_i^{-q}}$ at $\overline{d_i^{-q}} - \ell$ otherwise.

Fig. 3 describes time-reversibility with respect to partial execution in Definition 4. Suppose that if A_G guarantees that every job of τ_i under G finishes $C_i' = 2$ time units execution between its release time and that after $\ell = 4$ time units, every job of τ_i under $\overline{G}$ finishes at least two time units execution between its deadline ahead of 4 time units and its deadline (or all the remaining execution if the amount of the remaining execution at its deadline ahead of 4 time units is less than two). If this relationship holds for every $\tau_i \in \tau$, $C_i' \in [0, C_i]$, and $\ell \in [0, D_i]$, A_G is said to be time-reversible with respect to partial execution. Note that the notion of time-reversibility with respect to partial execution (i.e., Definition 4) subsumes that with respect to schedulability (i.e., Definition 3). This is because, if we set C_i' to C_i and ℓ to D_i for every $\tau_i \in \tau$, the former is equivalent to the latter.

While some existing schedulability tests are time-reversible with respect to partial execution, we prove time-reversibility of the schedulability test developed in Section IV, as follows.

Lemma 5: RTA for LCFS is time-reversible with respect to partial execution.

Proof: By Definition 4, we need to prove that for every $\tau_k \in \tau$, $C_k' \in [0, C_k]$, and $\ell \in [0, D_k]$, if RTA for LCFS guarantees that the amount of execution of every job of τ_k under LCFS (denoted by J_k^q) in $[r_k^q, r_k^q + \ell)$ is C_k', the amount of execution of every job of τ_k under EDF (denoted by $\overline{J_k^{-q}}$) in $[\overline{d_k^{-q}} - \ell, \overline{d_k^{-q}})$ is either (a) no smaller than C_k' if the amount of the remaining execution at $\overline{d_k^{-q}} - \ell$ is no smaller than C_k' or (b) equal to the amount of the remaining execution otherwise. This is achieved by proving that $I_{k \leftarrow i}(\overline{d_k^*} - \ell, \overline{d_k^*})$ under EDF is no larger than the upper-bound of interference under LCFS (i.e., $L_i(\ell)$) for every $\ell \in [0, D_k]$.

For a job of τ_i to interfere with $\overline{J_k^*}$ in $[\overline{d_k^*} - \ell, \overline{d_k^*})$, the deadline of the job of τ_i is no later than that of $\overline{J_k^*}$. Therefore, the amount of execution of jobs of τ_i whose priority is higher than J_k^* is maximized when the deadline of the last job of τ_i in the interval is aligned to that of $\overline{J_k^*}$ as shown in Fig. 2(b). Then, $E_i(\ell)$ can be an upper-bound of the amount. Then, regardless of the slack value of τ_i (i.e., S_i), $E_i(\ell) \leq L_i(\ell)$ holds for every $\ell \in [0, D_k]$.

This implies that as long as RTA for LCFS guarantees C_i' amount of execution of every job of τ_i in $[r_k^q, r_k^q + \ell)$ under LCFS, we can also guarantee C_i' amount of execution of every job of τ_i in $[\overline{d_k^{-q}} - \ell, \overline{d_k^{-q}})$ under EDF. Therefore, the lemma holds. ■

Similar to time-reversibility with respect to schedulability, RTA for EDF without slack reclamation is also time-reversible with respect to partial execution, while RTA for EDF with slack reclamation is not.

B. Synergy of two schedulability tests beyond simple union of their individual schedulability

According to Definition 4, a time-reversible (with respect to partial execution) schedulability test A_G can guarantee the execution of a job under $\overline{G}$ in an interval *between an arbitrary time instant and its deadline*, which is not effectively addressed by existing schedulability tests. On the other hand, some existing schedulability tests can guarantee the execution of a job under their target scheduling algorithms in an interval *between its release time and an arbitrary time instant*. Therefore, if there exist a time-reversible (with respect to partial execution) schedulability test A_G and a schedulability test $B_{\overline{G}}$ (regardless of time-reversibility), they cooperate for the schedulability guarantee of a job under $\overline{G}$. That is, the latter directly guarantees C_i' amount of execution performed between a job's release time and an arbitrary instant t, while the former indirectly guarantees $C_i - C_i'$ amount of execution performed between t and the job's deadline. Fig. 4 shows an example, and the following theorem records this.

Theorem 1: Suppose there exist two schedulability tests, one for a scheduling algorithm G and the other for its time-reversed scheduling algorithm $\overline{G}$ (denoted by A_G and $B_{\overline{G}}$), and A_G is time-reversible with respect to partial execution. Then, a task set τ is schedulable by $\overline{G}$, if for every $\tau_i \in \tau$, there exist $C_i' \in [0, C_i]$ and $\ell \in [0, D_i]$ such that A_G guarantees that every job of τ_i under G (denoted by J_i^q) finishes its execution at least as much as C_i' in $[r_i^q, r_i^q + \ell)$ and $B_{\overline{G}}$ guarantees that every job of τ_i under $\overline{G}$ (denoted by $\overline{J_i^{-q}}$) finishes its execution at least as much as $C_i - C_i'$ in $[\overline{r_i^{-q}}, \overline{d_i^{-q}} - \ell)$.

Proof: By Definition 4, A_G guarantees that every job of τ_i under $\overline{G}$ (denoted by $\overline{J_i^{-q}}$) finishes its execution at least as

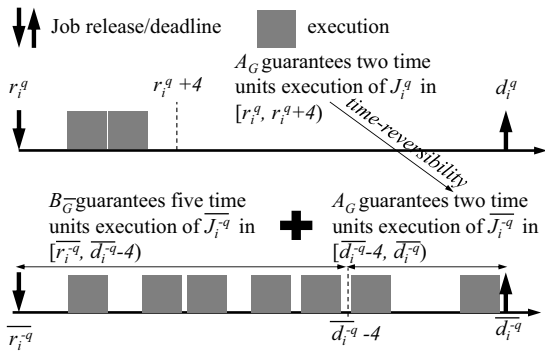

Fig. 4. Seven time units execution of $\overline{J_i^{-q}}$ under $\overline{G}$ is guaranteed by two parts: (a) the first five time units in $[\overline{r_i^{-q}}, \overline{d_i^{-q}}-4)$ by $B_{\overline{G}}$ directly and (b) the next two time units in $[\overline{d_i^{-q}}-4, \overline{d_i^{-q}})$ by A_G associated with time-reversibility with respect to partial execution.

much as C_i' in $[\overline{d_i^{-q}}-\ell, \overline{d_i^{-q}})$ (or the amount of the remaining execution at $\overline{d_i^{-q}}-\ell$ if it is less than C_i'). Since $B_{\overline{G}}$ guarantees $C_i - C_i'$ amount of execution of $\overline{J_i^{-q}}$ in $[\overline{r_i^{-q}}, \overline{d_i^{-q}}-\ell)$, the theorem holds. ■

Using Theorem 1, we immediately develop a new EDF schedulability test consisting of RTA for LCFS and that for EDF, as stated in the following lemma.

Lemma 6: A task set τ is schedulable by EDF, if for every $\tau_i \in \tau$, there exist $C_i' \in [0, C_i]$ and $\ell \in [0, D_i]$ such that RTA for LCFS guarantees that every job of τ_i under LCFS (denoted by J_i^q) finishes its execution at least as much as C_i' in $[r_i^q, r_i^q+\ell)$ and RTA for EDF with slack reclamation guarantees that every job of τ_i under EDF (denoted by $\overline{J_i^{-q}}$) finishes its execution at least as much as $C_i - C_i'$ in $[\overline{r_i^{-q}}, \overline{d_i^{-q}}-\ell)$.

Proof: By Lemma 5, RTA for LCFS is time-reversible with respect to partial execution. Then, the lemma immediately holds by Theorem 1. ■

While partial execution guarantees by RTA for EDF in Lemma 6 do not necessarily entail the calculation of the slack value, we may deliberately calculate the slack value for the slack reclamation, which improves the schedulability guarantees. For the selection of C_i', we may try some of choices, or all choices 0, 1, 2, ..., C_i by exploring a tradeoff between time-complexity and tightness of schedulability guarantees, which will be discussed in Section VI.

Theorem 1 yields significant improvement on schedulability guarantees consisting of two parts: (a) partial execution guarantees of a job in an interval between the release time and an arbitrary time instant, and (b) that in an interval between the time instant and the deadline. Such a divide-and-conquer approach has not been considered by existing approaches due to non-existence of schedulability tests that realize (b) effectively, while the notion of time-reversibility successfully resolves the issue. Motivated by this, the next subsection addresses (b) directly, yielding a tighter schedulability test than Lemma 6.

C. New direction of developing schedulability tests: a divide-and-conquer approach

Lemma 6 is successful in finding additional EDF-schedulable task sets, which are deemed schedulable by neither RTA for EDF nor RTA for LCFS. However, the lemma cannot exploit the slack values from the side of RTA for LCFS, since RTA for LCFS cannot address the slack value of a job under EDF despite its time-reversibility. Therefore, instead of making a detour to guarantee partial execution through the notion of time-reversibility, we directly apply the divide-and-conquer approach as follows: the first some time units execution is guaranteed by RTA for EDF with slack reclamation, while the remaining execution is guaranteed by another EDF schedulability test to be developed inspired by RTA for LCFS. To this end, we derive an upper-bound on the interference of jobs of τ_i on J_k^* under EDF in an interval between an arbitrary time instant and J_k^*'s deadline $[d_k^* - \ell, d_k^*)$, as follows.

Lemma 7: Under EDF, the following inequality holds:

$$I_{k \leftarrow i}(d_k^* - \ell, d_k^*) \leq \min\big(W_i(\ell), E_i(\ell)\big) = E_i(\ell). \tag{8}$$

Proof: Under EDF, jobs of τ_i can interfere with J_k^* only when their deadlines are no later than J_k^*'s deadline. Therefore, the interference of τ_i on τ_k in $[d_k^* - \ell, d_k^*)$ is maximized when the deadline of the last job of τ_i is d_k^* as shown in Fig. 2(b). Therefore, $I_{k \leftarrow i}(d_k^* - \ell, d_k^*)$ under EDF is upper-bounded by $E_i(\ell)$. Also, $E_i(\ell) \leq W_i(\ell)$ holds for every $\ell \in [0, D_k]$ and $S_i \in [0, D_i]$. ■

Then, instead of employing RTA for LCFS to guarantee partial execution of a job under EDF in an interval between an arbitrary time instant and its deadline, we directly assure the partial execution using the above upper-bound. If we compare the upper-bound of the interference under RTA for LCFS ($L_i(\ell)$) and the above upper-bound ($E_i(\ell)$), the inequality $E_i(\ell) \leq L_i(\ell)$ always holds, meaning such a direct assurance yields tighter schedulability guarantees than Lemma 6. The following lemma presents a tighter EDF schedulability test than Lemma 6.

Lemma 8: A task set τ is schedulable by EDF, if for every $\tau_k \in \tau$, there exist $C_k' \in [0, C_k]$ and $\ell \in [0, D_k]$ such that the following two inequalities hold:

$$\ell \leq C_k - C_k' + \left\lfloor \frac{1}{m} \sum_{\tau_i \in \tau - \{\tau_k\}} \min\Big(\min(W_i(\ell), E_i(D_k)), \ell - (C_k - C_k') + 1\Big)\right\rfloor, \tag{9}$$

$$D_k - \ell \leq C_k' + \left\lfloor \frac{1}{m} \sum_{\tau_i \in \tau - \{\tau_k\}} \min\Big(E_i(D_k - \ell), (D_k - \ell) - C_k' + 1\Big)\right\rfloor. \tag{10}$$

Proof: We divide the interval of interest $[r_k^*, d_k^*)$ of length D_k into two: $[r_k^*, r_k^* + \ell)$ and $[r_k^* + \ell, d_k^*)$. Then, we prove that (a) $C_k - C_k'$ amount of execution is performed in the former interval, and (b) C_k' amount of execution is performed in the latter interval.

Case (a): A job cannot execute only when there are other m jobs whose priorities are higher than the job of interest.

Therefore, from Eq. (1), we guarantee $C_k - C'_k$ amount of execution performed in $[r^*_k, r^*_k + \ell)$ of length ℓ, if the following inequality holds:

$$\ell \leq C_k - C'_k + \left\lfloor \frac{1}{m} \sum_{\tau_i \in \tau - \{\tau_k\}} I_{k \leftarrow i}(r^*_k, r^*_k + \ell), \ell - (C_k - C'_k) + 1 \right) \Big\rfloor.$$

Since $I_{k \leftarrow i}(r^*_k, r^*_k + \ell) \leq \min\left(W_i(\ell), E_i(D_k)\right)$ holds under EDF (from Eq. (5)), Eq. (9) implies that we can guarantee $C_k - C'_k$ amount of execution performed in $[r^*_k, r^*_k + \ell)$.

Case (b): Similar to Eq. (1), we also guarantee C'_k amount of execution performed in $[r^*_k + \ell, d^*_k)$ of length $D_k - \ell$, if the following inequality holds:

$$D_k - \ell \leq$$
$$C'_k + \left\lfloor \frac{1}{m} \sum_{\tau_i \in \tau - \{\tau_k\}} \min\left(I_{k \leftarrow i}(r^*_k + \ell, d^*_k), (D_k - \ell) - C'_k + 1\right) \right\rfloor.$$

Since $I_{k \leftarrow i}(r^*_k + \ell, d^*_k) \leq E_i(D_k - \ell)$ holds under EDF (from Eq. (8)), Eq. (10) implies that we can guarantee C'_k amount of execution performed in $[r^*_k + \ell, d^*_k)$.

The lemma holds by Cases (a) and (b). ■

Similar to Lemma 6, we intentionally calculate the slack values from RTA for EDF with slack reclamation, yielding tighter schedulability guarantees. Section VI will present the number of additional EDF-schedulable task sets proven by Lemma 8.

Inspired by the notion of time-reversibility, we open up a new direction of developing schedulability tests—*a divide-and-conquer* approach, and this entails the development of schedulability tests that assure a part of execution of a job between an arbitrary time instant and its deadline.

VI. Evaluation

In this section, we demonstrate via simulation that the notion of time-reversibility improves schedulability guarantees. First, we explain the task set generation procedure. Then, we present schedulability improvement of EDF as well as that of LCFS.

A. Task set generation

We generate real-time task sets based on a popular technique [13], used in a number of multiprocessor scheduling studies, e.g., [12, 14]. There are three input parameters: (a) the number of processors m (2, 4, 8 or 16), (b) the type of tasks in each task set (constrained deadline: $D_i \leq T_i$ or implicit deadline: $D_i = T_i$), and (c) utilization (C_i/T_i) distribution of individual tasks (bimodal with parameter: 0.1, 0.3, 0.5, 0.7 or 0.9, or exponential with parameter: 0.1, 0.3, 0.5, 0.7 or 0.9), detailed in [14]. For each task, T_i is uniformly chosen in $[1, 1000]$, C_i is chosen based on the bimodal or exponential parameter, and D_i is uniformly selected in $[C_i, T_i]$ for constrained deadline tasks or D_i is equal to T_i for implicit deadline tasks. In compliance with the quantum length, we set all task parameters to the closest integer values.

For each combination of (a), (b) and (c), we repeat the following steps, and generate 10,000 task sets. As a result, 100,000 task sets are generated, for given m (i.e., the number of processors) and the type of task sets.

1) We generate a set of $m + 1$ tasks, because a task set composed of m or less tasks is trivially schedulable.
2) We check whether the generated task set can pass a necessary feasibility condition in [15].
3) If it fails to pass the feasibility test, we discard the generated set and return to Step 1. Otherwise, we include this set for evaluation. This valid task set serves as a basis for the next new set; we add a new task into the valid task set, and return to Step 2 with this new set.

TABLE I. The number of constrained-deadline task sets proven schedulable by RTA-W_{EDF}, NEW-A_{EDF}, NEW-B_{EDF}, and NEW-C_{EDF}

	The number of schedulable task sets				Ratio
m	RTA-W_{EDF}	NEW-A_{EDF}	NEW-B_{EDF}	NEW-C_{EDF}	$\frac{\text{NEW-C}_{EDF}}{\text{RTA-W}_{EDF}}$
2	34450	34637	35287	35406	102.8 %
4	19674	19723	20561	20691	105.2 %
8	11948	11967	12922	13071	109.4 %
16	7426	7429	8271	8402	113.1 %

B. Evaluation results

Among all the generated task sets, we compare the number of task sets proven schedulable by different schedulability tests for EDF and LCFS on a multiprocessor platform, as follows.

- RTA for any work-conserving algorithm with slack reclamation that upper-bounds the interference by $W_i(\ell)$ [10] (denoted by RTA-W_{WC});
- RTA for EDF with slack reclamation [10] (denoted by RTA-W_{EDF});
- RTA for LCFS developed in this paper (denoted by RTA$_{LCFS}$).
- RTA-W_{EDF} or RTA$_{LCFS}$, i.e., each task is deemed schedulable if either the former or the latter deems the task schedulable (denoted by NEW-A_{EDF});
- Lemma 8 with $C'_i = 0, 0.1{\cdot}D_i, 0.2{\cdot}D_i, ..., C_i$ (denoted by NEW-B_{EDF}); and
- Lemma 8 with $C'_i = 0, 1, 2, ..., C_i$ (denoted by NEW-C_{EDF}).

Note that each of all the schedulability tests (including inherently non-EDF tests) can serve as an EDF schedulability test due to time-reversibility of RTA$_{LCFS}$ and the fact that WC subsumes EDF (RTA-W_{WC}). When it comes to LCFS, there are two schedulability tests: RTA-W_{WC} and RTA$_{LCFS}$. The former is the best existing schedulability test for LCFS due to non-existence of LCFS-specific tests, which will be compared to the latter.

Table I shows the number of constrained-deadline task sets deemed schedulable by individual EDF schedulability tests on 2, 4, 8 and 16 processor platforms. While RTA-W_{EDF} is known as the best existing schedulability test for EDF in terms of tightness of schedulability guarantees, NEW-A_{EDF} covers some additional task sets which RTA-W_{EDF} cannot cover. The additional schedulable task sets, although marginal, entirely

TABLE II. THE NUMBER OF CONSTRAINED-DEADLINE TASK SETS PROVEN SCHEDULABLE BY RTA-W_{WC} AND RTA_{LCFS}

	The number of schedulable task sets		Ratio
m	RTA-W_{WC}	RTA_{LCFS}	$\frac{RTA_{LCFS}}{RTA\text{-}W_{WC}}$
2	9088	9705	106.8 %
4	4406	4633	105.2 %
8	2060	2117	102.8 %
16	837	855	102.2 %

come from the notion of time-reversibility. If we directly apply the divide-and-conquer approach inspired by time-reversibility, NEW-B_{EDF} and NEW-C_{EDF} cover up to 11.4% and 13.1% additional EDF-schedulable task sets, compared to RTA-W_{EDF}. The difference between 11.4% and 13.1% arises from the number of attempts of C_i': 11 values for NEW-B_{EDF} and all possible integer values for NEW-C_{EDF}.

When it comes to LCFS schedulability improvement, we count the number of constrained-deadline task sets deemed schedulable by RTA-W_{WC} and RTA_{LCFS}. Table II shows that the new schedulability test for LCFS, RTA_{LCFS}, finds up to 6.8% additional LCFS-schedulable task sets that are not covered by RTA-W_{WC}, which is an additional contribution of this paper.

Note that for both EDF and LCFS schedulability tests, the schedulability trend for implicit-deadline task sets is the same as that for constrained-deadline task sets.

When it comes to time-complexity, it is known that RTA without and with slack reclamation requires $O\big(n^2 \cdot \max_{\tau_i \in \tau} D_i\big)$ and $O\big(n^3 \cdot (\max_{\tau_i \in \tau} D_i)^2\big)$ computations, respectively [10]. RTA_{LCFS} belongs to the former, while RTA-W_{WC}, RTA-W_{EDF}, and NEW-A_{EDF} belong to the latter. Due to the multiple attempts of C_i', NEW-B_{EDF} exhibits $11 \cdot O\big(n^3 \cdot (\max_{\tau_i \in \tau} D_i)^2\big)$, while NEW-$C_{EDF}$ exhibits $\max_{\tau_i \in \tau} C_i \cdot O\big(n^3 \cdot (\max_{\tau_i \in \tau} D_i)^2\big)$ time-complexity. Since we usually consider the offline schedulability guarantees, all the schedulability tests are practical in terms of time-complexity.

VII. CONCLUSION

In this paper, we introduced the notion of time-reversibility for real-time scheduling, and demonstrated how to utilize the notion for tighter schedulability guarantees. In addition to quantitative improvement of preemptive EDF schedulability, this paper pointed out a new direction of developing schedulability tests. That is, we can guarantee the schedulability of a job in a divide-and-conquer manner using two schedulability tests, and this calls for the development of schedulability tests that can assure partial execution of a job in an interval between an arbitrary time instant and its deadline, which has not been effectively realized by existing schedulability tests.

While we presented limited examples for the notion of time-reversibility to improve schedulability guarantees, we expect that the notion will make a more significant impact on real-time scheduling. In the future, we would like to exploit the notion to develop new schedulability tests for other scheduling algorithms than preemptive EDF. In particular, we have a plan to investigate how the notion can be effectively applied to non-preemptive scheduling algorithms. Also, it would be interesting to study how the notion is adapted for more complex task models, e.g., the mixed-criticality task model [16] and the synchronous parallel task model [17].

ACKNOWLEDGEMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2014R1A1A1035827).

REFERENCES

[1] C. Liu and J. Layland, "Scheduling algorithms for multi-programming in a hard-real-time environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[2] M. Bertogna and S. Baruah, "Tests for global EDF schedulability analysis," *Journal of systems architecture*, vol. 57, no. 5, pp. 487–497, 2011.

[3] A. Mok, "Fundamental design problems of distributed systems for the hard-real-time environment," Ph.D. dissertation, Massachusetts Institute of Technology, 1983.

[4] F. P. Kelly, "Networks of queues," *Advances in Applied Probability*, vol. 8, pp. 416–432, 1976.

[5] N. Audsley, A. Burns, M. Richardson, and A. Wellings, "Hard real-time scheduling: the deadline-monotonic approach," in *Proceedings of the IEEE Workshop on Real-Time Operating Systems and Software*, May 1991, pp. 133–137.

[6] S. Baruah, A. Mok, and L. Rosier, "Preemptively scheduling hard-real-time sporadic tasks on one processor," in *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, 1990, pp. 182–190.

[7] S. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel, "Proportionate progress: a notion of fairness in resource allocation," *Algorithmica*, vol. 15, no. 6, pp. 600–625, 1996.

[8] G. Levin, S. Funk, C. Sadowski, I. Pye, and S. Brandt, "DP-FAIR: A simple model for understanding optimal multiprocessor scheduling," in *Proceedings of Euromicro Conference on Real-Time Systems (ECRTS)*, 2010, pp. 3–13.

[9] P. Regnier, G. Lima, E. Massa, G. Levin, and S. Brandt, "RUN: Optimal multiprocessor real-time scheduling via reduction to uniprocessor," in *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, 2011, pp. 104–115.

[10] M. Bertogna and M. Cirinei, "Response-time analysis for globally scheduled symmetric multiprocessor platforms," in *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, 2007, pp. 149–160.

[11] N. Guan, W. Yi, Z. Gu, Q. Deng, and G. Yu, "New schedulability test conditions for non-preemptive scheduling on multiprocessor platforms," in *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, 2008, pp. 137–146.

[12] M. Bertogna, M. Cirinei, and G. Lipari, "Schedulability analysis of global scheduling algorithms on multiprocessor platforms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, pp. 553–566, 2009.

[13] T. P. Baker, "Comparison of empirical success rates of global vs. partitioned fixed-priority EDF scheduling for hard real-time," Department of Computer Science, Florida State University, Tallahassee, Tech. Rep. TR–050601, 2005.

[14] J. Lee, A. Easwaran, and I. Shin, "Laxity dynamics and LLF schedulability analysis on multiprocessor platforms," *Real-Time Systems*, vol. 48, no. 6, pp. 716–749, 2012.

[15] T. P. Baker and M. Cirinei, "A necessary and sometimes sufficient condition for the feasibility of sets of sporadic hard-deadline tasks," in *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, 2006, pp. 178–190.

[16] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, 2007, pp. 239–243.

[17] H. S. Chwa, J. Lee, K.-M. Phan, A. Easwaran, and I. Shin, "Global EDF schedulability analysis for synchronous parallel tasks on multicore platforms," in *Proceedings of Euromicro Conference on Real-Time Systems (ECRTS)*, 2013, pp. 25–34.