Schedulability Analysis for a Mode Transition in Real-Time Multi-Core Systems

Jinkyu Lee and Kang G. Shin
Real-Time Computing Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan – Ann Arbor, U.S.A.
{jinkyul, kgshin}@eecs.umich.edu

***Abstract*—To enable real-time systems to adapt to dynamically changing environments, update functionalities and/or accommodate those tasks migrated from other failed sub-systems, there have been a number of studies on making timing guarantees while accounting for change of parameters and addition/deletion of tasks. While most of them have dealt with "transition" protocols that delay next task releases or discard the unfinished tasks released before the transition, such protocols are not suitable for many control systems in which missing/delaying control updates (by completing periodic tasks) even during a transition or mode-change may cause system instability or incur a significant incremental operational cost. In this paper, we focus on a transition protocol that does not miss/delay control updates during a system transition, and develop a new schedulability analysis for the transition in a real-time multi-core system, which provides sufficient timing guarantees without requiring any online information, such as the release and execution patterns of tasks and the start time of a transition. To achieve this, we extend an existing popular schedulability analysis framework for non-transitional tasks, and identify the scenarios that maximize the duration of a task's interference to another task in the case of a transition. Since the analysis works for any arbitrary transition order of tasks, we can improve the schedulability performance by enforcing a specific order. We formulate the problem of assigning an optimal transition order, and develop a solution by deriving some properties of optimality. Our evaluation results demonstrate that the proposed solution finds more schedulable task sets, which are not covered by naive approaches.**

I. Introduction

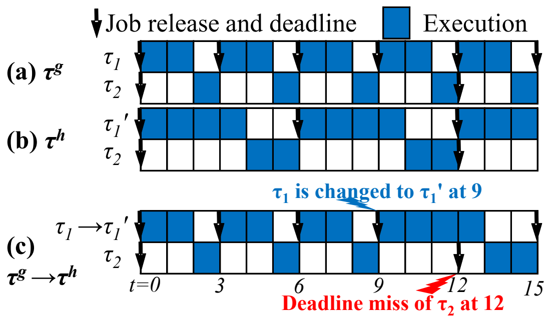
Most studies of real-time systems have focused on meeting deadlines of a given set of static periodic tasks whose parameters do not change over time, and many scheduling algorithms such as EDF (Earliest Deadline First) and FP (Fixed Priority) [1] have been studied extensively to guarantee the deadlines of such tasks. However, today's real-time systems are more dynamic, requiring dynamic change of task parameters, in order to cope with changing environments and physical system's state, improve system performance with new protocols/functionalities, and/or accommodate the tasks migrated from other failed sub-systems. Furthermore, such *transition* of tasks should be done without missing/delaying systems' periodic control updates in many systems, such as real-time control systems in power and automation domains [2–4]. In these systems, an additional transition delay that skips/suspends control updates may cause system instability or incur high control cost. Therefore, it is increasingly important to support real-time systems in the case of a transition in which parameter changes and/or addition/deletion of tasks occur during the system's operation, and both unchanged and changed tasks coexist and do not skip/suspend their control updates.

To make timing guarantees in the presence of changes of task parameters, a number of studies focused on mode-change protocols. Starting from [5], timing guarantees of periodic tasks with mode changes have been developed for uniprocessor platforms [6–11] (see a survey [7]). Some of them have then been extended to multi-core systems [12–14]. On the other hand, some studies have analyzed mode changes of real-time systems with streaming data using real-time calculus [15–17]. If we confine our interest to periodic tasks, then there have been only a few studies (e.g., [6, 9]) that support the transition without skipping/suspending control updates, while other mode-change protocols require an additional transition delay. These studies, however, have been limited to *uniprocessor* platforms.

In this paper, we focus on multi-core systems that have become popular as real-time system platforms due to their potential for high performance at low cost, and support timing guarantees with a transition that yields continuous task execution. One may argue that timing guarantees with a transition can be simply achieved by making the timing guarantees of both the task sets before and after the transition, which does not hold as explained below. As shown in Figs. 1(a) and (b), the traditional response-time analysis [18] guarantees the schedulability of each of task sets τ^g and τ^h on a uniprocessor, when each task set is scheduled by FP (assuming τ_1 and τ_1' have a higher priority than τ_2). However, if τ^g makes a transition to τ^h at $t = 9$, τ_2 misses a deadline at $t = 12$ as shown in Fig. 1(c).[1] Therefore, we need to develop a new schedulability analysis for transitions with continuous task execution, which has not been studied before for real-time *multi-core* systems.

We consider a series of transitions, each of which is performed after completing its predecessor transition. Within a single transition, multiple tasks can change their parameters, including addition/deletion of tasks and change of tasks' periods and execution times. During each transition, no task misses

[1]For simplicity, we show an example on a uniprocessor system, but the same phenomenon occurs on a multi-core system. Another example can be found in [6].



Job release and deadline
Execution
(a) τ^g
τ_1
τ_2
(b) τ^h
τ_1'
τ_2
τ_1 is changed to τ_1' at 9
(c) $\tau^g \rightarrow \tau^h$
$\tau_1 \rightarrow \tau_1'$
τ_2
t=0
3
6
9
12
15
Deadline miss of τ_2 at 12

Fig. 1. $\tau^g = \{\tau_1(\text{period and relative deadline=3, execution time=2}), \tau_2(12,4)\}$ as well as $\tau^h = \{\tau_1'(6,4), \tau_2(12,4)\}$ is schedulable by FP (assuming τ_1 and τ_1' having a higher priority than τ_2) on a uniprocessor in the absence of transition. However, the task set is not schedulable in the presence of transition from τ^g to τ^h at $t = 9$.

or delays its control updates due to the transition. Section III will detail our transition protocol.

Our goal is to develop a sufficient, offline schedulability analysis that guarantees all task deadlines in the presence of a single transition on multi-core platforms. We make this analysis independent of online information, such as the time instant when the transition starts, and task release and execution patterns; otherwise, the system must monitor/predict the information, and hence the analysis cannot provide any offline guarantees. We also make the analysis independent of the history of previous transitions. This relieves the system from keeping track of transition history, and the analysis from considering all of the transition history. Then, the timing guarantees of a task set with a series of transitions can be decomposed into those with individual transitions, each of which can be made with our proposed schedulability analysis.

To achieve the above goal, we focus on an existing popular schedulability analysis framework for non-transitional tasks on multi-core platforms, called *deadline-based schedulability analysis* [19, 20]. To extend this analysis, it is essential to calculate the duration a task can interfere with other tasks in the presence of a transition. Since this duration depends on the underlying scheduling algorithms, we identify and calculate the maximum duration under two popular scheduling algorithms, EDF and FP. By extending the analysis framework to a transition, and incorporating the maximum duration into the framework, we develop a sufficient schedulability analysis for a transition, which is, to the best of our knowledge, the first attempt in real-time multi-core systems.

While the thus-developed analysis provides safe guarantees on timing requirements, it can be pessimistic due to its applicability to any arbitrary transition order of tasks. Once we fix the transition order of tasks, we find that a task with one of both modes (before and after the transition) cannot interfere with another task with a given mode. This entails two questions: (i) how to guarantee timing requirements with a specific transition order, and (ii) how to find a transition order that guarantees the timing requirements of a given task set by (i). We achieve (i) by adapting the proposed schedulability analysis to a given transition order. For (ii), we derive some properties toward an optimal order, and then develop an optimal assignment of transition order that divides tasks into three groups and determines the group-level optimal transition order. For the relative transition order within each group, we prove that any arbitrary order of tasks in the first and third groups preserves the optimality, and apply a heuristic algorithm for the second group. Our simulation results demonstrate that the optimal transition-order assignment covers a large number of schedulable task sets, which are otherwise not proven schedulable.

In summary, this paper makes the following contributions.

- Development of a new (first) schedulability analysis for transitions (without any additional transition delay) in real-time multi-core systems;
- Identification of the problem of the sequential transition-order assignment, and development of a grouping framework for an optimal transition-order assignment using the derived properties regarding optimality; and
- Demonstration of the effectiveness of the framework via simulation.

The rest of this paper is organized as follows. Section II presents our system model, assumptions and notations, and recapitulates an existing schedulability analysis framework for non-transitional tasks. Section III describes our transition protocol, and addresses the requirements of the proposed schedulability analysis. Section IV develops a schedulability analysis for a transition. Section V develops an optimal transition-order assignment framework. Section VI evaluates the effectiveness of this framework, and Section VII concludes the paper.

II. Background

In this section, we first describe the system model, assumptions, and notations to be used throughout this paper. Then, we summarize an existing schedulability analysis framework for non-transitional tasks, which will be used as a basis for our schedulability analysis for a system transition.

A. System model, assumptions and notations

We consider a periodic task model [1, 21] associated with different operation modes. A task τ_i associated with a mode M^g (denoted by τ_i^g) is specified by (p_i^g, e_i^g, d_i^g) where p_i^g is the time separation between two successive invocations (called period),[2] e_i^g the worst-case execution time, and d_i^g the relative deadline of τ_i^g. Our focus is confined to constrained deadline tasks, each of which satisfies the inequality $d_i^g \leq p_i^g$. Different modes imply not only the change of task parameters (e.g., $p_i^g \neq p_i^h$, $e_i^g \neq e_i^h$ and/or $d_i^g \neq d_i^h$), but also addition/deletion of tasks. For convenience of presentation, we let τ denote a set of all tasks existing in at least one mode; if a task τ_i does not exist in a mode M^g, τ_i^g represents a *dummy* task ($p_i^g = 1, e_i^g = 0, d_i^g = 1$), which does not affect the actual execution of other tasks. Then, let τ^g denote a task set τ associated with a mode M^g, i.e., $\tau^g = \{\tau_1^g, \cdots, \tau_{|\tau|}^g\}$, where $|\tau|$ denotes the number of tasks in τ. Whenever g is irrelevant, we will omit it, i.e., using τ_i, p_i, e_i and d_i instead of τ_i^g, p_i^g, e_i^g and d_i^g.

[2]Note that all the analytic results in this paper are also applicable to sporadic tasks in which p_i^g represents the *minimum* separation, not the exact separation.

A task τ_i^g invokes a series of jobs, each separated from its predecessor by p_i^g time units, supposed to finish its execution within d_i^g time units, and taking at most e_i^g time units for its execution. We call the interval between the release time and deadline of a job J, the *scheduling window* of J. We assume a quantum-based time, and let the length of a quantum be one time unit, without loss of generality. All task parameters are specified in multiples of this quantum.

The computing platform is a multi-core chip containing m identical cores. We consider global, preemptive and work-conserving scheduling algorithms under which the execution of a job can migrate from one core to another; a job can be preempted at any time; and there should be no idle core as long as there is an unfinished, ready job. We also assume that a job cannot be executed in parallel.

B. Existing schedulability analysis

To ensure no deadline miss in a set of non-transitional tasks, numerous schedulability analyses have been developed. Of them, a deadline-based analysis has been popular in real-time multi-core scheduling, due to its wide applicability (EDF, FP, EDZL, LLF and potentially more) and low (polynomial) time-complexity [22–24].

For completeness, we summarize the deadline-based analysis for non-transitional tasks in real-time multi-core systems, which was originally introduced in [19, 20]. This technique employs the notion of *interference* [20]. The interference to τ_k in $[a, b)$ (denoted by $I(\tau_k, a, b)$) is defined as the cumulative length of all sub-intervals in $[a, b)$ such that a job of τ_k is ready to execute, but it cannot execute due to other higher-priority jobs' execution. Also, the interference of τ_i with τ_k in $[a, b)$ (denoted by $I(\tau_k \leftarrow \tau_i, a, b)$) is defined as the cumulative length of all sub-intervals in $[a, b)$ such that a job of τ_k cannot execute although it is ready to execute, but a job of τ_i executes instead. Since a job of τ_k cannot execute only when m other jobs execute, the following equation holds under any global work-conserving algorithm [20]:

$$I(\tau_k, a, b) = \frac{\sum_{\tau_i \in \tau \setminus \{\tau_k\}} I(\tau_k \leftarrow \tau_i, a, b)}{m}. \quad (1)$$

A relationship between $I(\tau_k, a, b)$ and $I(\tau_k \leftarrow \tau_i, a, b)$ was derived in [20] as:

$$I(\tau_k, a, b) < x \iff \sum_{\tau_i \in \tau \setminus \{\tau_k\}} \min\Big(I(\tau_k \leftarrow \tau_i, a, b), x\Big) < m \cdot x. \quad (2)$$

Note that the inequality holds for any arbitrary positive x.

We compute the maximum interference of $\tau_i \in \tau \setminus \{\tau_k\}$ with τ_k in an interval of length d_k between the release and completion times of any job of τ_k (denoted by $\mathbf{I}(\tau_k \leftarrow \tau_i)$) as:

$$\mathbf{I}(\tau_k \leftarrow \tau_i) \triangleq \max_{t \,|\, \text{the release time of any job of } \tau_k} I(\tau_k \leftarrow \tau_i, t, t + d_k). \quad (3)$$

If the maximum interference to τ_k in an interval of length d_k starting from the release time of any job of τ_k is less than or equal to $d_k - e_k$ (i.e., strictly less than $d_k - e_k + 1$), any job of τ_k successfully completes its execution (that amounts to e_k) before its deadline. Incorporating Eqs. (1) and (2), this leads to the following deadline-based analysis framework:

Lemma 1 (Theorem 5 in [22])**.** *Suppose that a set τ of non-transitional tasks is scheduled by a global, preemptive, and work-conserving algorithm. Then, τ is schedulable if the following inequality holds for all $\tau_k \in \tau$:*

$$\sum_{\tau_i \in \tau \setminus \{\tau_k\}} \min\Big(\mathbf{I}(\tau_k \leftarrow \tau_i), d_k - e_k + 1\Big) < m \cdot (d_k - e_k + 1). \quad (4)$$

Since the interference $\mathbf{I}(\tau_k \leftarrow \tau_i)$ varies with the underlying scheduling algorithms, upper-bounds on the interference under EDF and FP have been presented in [22]. We will develop their generalizations for a transition in Section IV.

III. Transition Protocol and Schedulability Analysis Requirements

In this section, we first describe our transition protocol and then state the requirements of the schedulability analysis for a transition to be developed in Section IV.

We consider a series of transitions $M^{g_1}, M^{g_2}, \cdots M^{g_n}$ of a task set τ, and each transition is separated from its predecessor and successor transitions, meaning that the scheduling window of any job with mode M^{g_z} cannot overlap with that of any job with other modes than its previous, current and next modes (i.e., $M^{g_{z-1}}$, M^{g_z} and $M^{g_{z+1}}$). Within a single transition from M^g to M^h, multiple tasks can change their task parameters; recall that addition or deletion of tasks is also expressed as change of task parameters by using dummy tasks as we discussed in Section II-A.

To support a transition that does not result in missing/delaying any task's control update, we follow the protocol in [6], as explained next. Suppose that a Mode Transition Request (MTR) from M^g to M^h is released at t_1, as shown in Fig. 2. We consider two types of tasks: (i) tasks whose parameters are not affected by the transition (i.e., $p_i^g = p_i^h$, $e_i^g = e_i^h$ and $d_i^g = d_i^h$) and (ii) the other tasks, which satisfy at least one of $p_i^g \neq p_i^h$, $e_i^g \neq e_i^h$ and $d_i^g \neq d_i^h$. Then, the protocol does not affect release patterns of each task in (i) at all, e.g., τ_1 in the figure. For each task τ_i in (ii), the next release time (i.e., the earliest release time of jobs of τ_i after t_1) is not different from the time without the transition, but the difference is that at the next release time, a job of τ_i^h (associated with a new mode M^h) is released instead of that of τ_i^g (associated with an old mode M^g), e.g., at t_2, a job of τ_2^h is released in the figure. After the release of the job with M^h, jobs of τ_i^h are periodically released until another MTR is released. Note that if τ_i^g is a dummy task, a job of τ_i^h is released when an MTR is released; for example, τ_3^g in Fig. 2 is a dummy task, so a job of τ_3^h is released as soon as the MTR is released at t_1. Therefore, this transition protocol supports not only each task's transition without missing/delaying its control update, but also immediate task migration from other systems due to failure.

While many existing mode transition protocols require to discard unfinished jobs of the old-mode tasks, or need to wait until some time instant for synchronous releases of jobs of

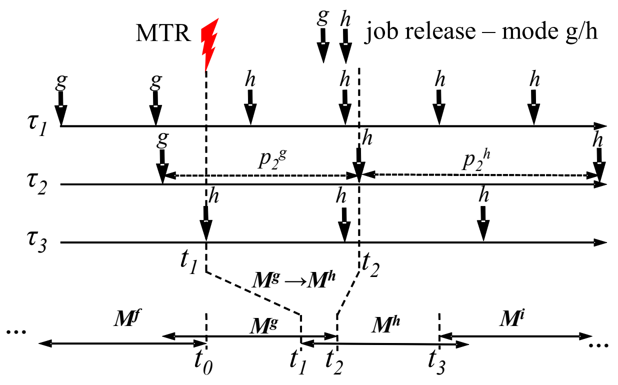


Fig. 2. An MTR from M^g to M^h is released at t_1: while the mode transition does not change any task parameter of τ_1, it extends the period of τ_2 and introduces a new task τ_3 (meaning τ_3^g is a dummy task).

the new-mode tasks (see the survey in [7]), this protocol does not perform such functions. Therefore, the protocol is suitable for real-time control systems, which require timely control updates, even in the presence of transitions.

Then, we want to determine whether or not a task set τ is schedulable in the presence of a single transition from M^g to M^h. The actual execution behavior during a transition depends on online information, such as the time instant when the current MTR is released, and the release and execution patterns, which are also affected by the previous transitions. If we develop a schedulability analysis that requires such online information, the system should be able to monitor/predict the information, and should keep track of the information. Since an MTR is triggered due to many situations (e.g., triggered by users, systems and failures), we may not predict the release time.

Therefore, our goal is to develop a sufficient offline schedulability analysis of a task set in the presence of a single transition, which does not require any online information, including the history of previous transitions. We also generalize the target schedulability analysis for non-transitional tasks. The (non)requirements for our analysis are summarized as follows.

R1. The schedulability analysis does not require any online information including release and execution patterns of jobs and the release time of an MTR.

R2. It focuses on a single transition, and should be independent of the history of previous transitions; by a single transition, we mean a period between the previous transition's completion and the next transition's beginning, e.g., $[t_0, t_3)$ for a single transition from M^g to M^h in Fig. 2. Then, we can guarantee the timing requirements of a task set with mode M^g (old mode without transition), with a transition from M^g to M^h, and with mode M^h (new mode without transition).

R3. It should be a generalization of the existing schedulability analysis described in Lemma 1.

Once we develop a schedulability analysis that satisfies R1–R3, we can determine the schedulability of a task set with a series of transitions, by sequentially applying the analysis with each single transition.

We will develop a schedulability analysis with R1–R3 in Section IV, and improve the analysis by adding some constraints to the transition protocol in Section V.

IV. Schedulability Analysis for a Transition

In this section, we develop a schedulability analysis framework for a single transition. First, we extend the existing deadline-based schedulability analysis framework presented in Lemma 1. Second, we calculate upper-bounds of the interference of a task in the presence of a transition under EDF and FP. Finally, we propose the deadline-based schedulability analysis by incorporating the upper-bounds into the framework, and demonstrate that the proposed analysis satisfies our design requirements R1–R3.

A. *Extension of the existing framework*

Since the schedulability analysis framework in Section II-B assumes only a single mode for each task, we need to extend the framework with two modes (hence a transition from M^g to M^h). To express the situation where two modes of a task can interfere with another task in a given mode, let $\tau_i^{g\Rightarrow h}$ denote τ_i in the presence of a transition from M^g to M^h. Then, we define interference of $\tau_i^{g\Rightarrow h}$ to τ_k^u (u is either g or h) in $[a, b)$, as the cumulative length of all intervals in $[a, b)$ such that a job of τ_k^u cannot execute although it is ready for execution, but a job of τ_i^g or τ_i^h executes; let $I(\tau_k^u \leftarrow \tau_i^{g\Rightarrow h}, a, b)$ denote the interference. Similarly to Eq. (3), we define $\mathbf{I}(\tau_k^u \leftarrow \tau_i^{g\Rightarrow h})$ as follows.

$$\mathbf{I}(\tau_k^u \leftarrow \tau_i^{g\Rightarrow h}) \triangleq \max_{t \mid \text{the release time of any job of } \tau_k^u} I(\tau_k^u \leftarrow \tau_i^{g\Rightarrow h}, t, t + d_k^u). \tag{5}$$

On the other hand, if we focus on a task τ_k which is interfered by other tasks, then we also consider two modes of τ_k because a task in one mode has different task parameters from the same task in another mode. This means that we should calculate the schedulability of both τ_k^g and τ_k^h independently. Finally, the deadline-based schedulability analysis framework in Lemma 1 is generalized by the following lemma.

Lemma 2. *Suppose that a task set τ makes a transition from M^g to M^h and is scheduled by a global, preemptive, and work-conserving algorithm. Then, the task set τ with the transition is schedulable if the following inequality holds for all $\tau_k \in \tau$ and $u \in \{g, h\}$:*

$$\sum_{\tau_i \in \tau \setminus \{\tau_k\}} \min\left(\mathbf{I}(\tau_k^u \leftarrow \tau_i^{g\Rightarrow h}), d_k^u - e_k^u + 1\right) < m \cdot (d_k^u - e_k^u + 1). \tag{6}$$

Proof: The lemma holds due to Lemma 1 and the definitions of $\mathbf{I}(\tau_k^u \leftarrow \tau_i^{g\Rightarrow h})$. ■

Then, how to upper-bound the interference $\mathbf{I}(\tau_k^u \leftarrow \tau_i^{g\Rightarrow h})$ is the most critical part of the schedulability analysis for a transition, which we will address next.

B. Calculation of the amount of interference

Since the interference depends on the underlying scheduling algorithm, we now calculate the interference under two popular scheduling algorithms, FP and EDF. To satisfy R1 and R2, we first compute the maximum amount of the interference of a task on another task under FP, which upper-bounds the amount of interference with any release and execution patterns and any release time of an MTR. Now, let us consider the case of FP.

1) Upper-bound of interference under FP: FP schedules jobs according to the pre-determined task-level priorities. That is, if τ_i has a lower priority than τ_k, a job of τ_i cannot interfere with any job of τ_k. Otherwise, a job of τ_i can potentially interfere with any job of τ_k. Therefore, we need to calculate the amount of execution of jobs of a given task in an interval.

For non-transitional tasks, the release and execution patterns that maximize the amount of execution of a single-mode task τ_i^g's jobs in an interval of length ℓ have been identified. As shown in Fig. 3(a), the interval of interest (i) starts when the first job of τ_i starts execution (at a), or (ii) ends when the last job of τ_i finishes execution (at b). In both cases, the first job of τ_i executes as late as possible, and the last job of τ_i executes as early as possible. Both cases result in the same amount of execution of τ_i^g's jobs, and the amount of execution in the case of (i) (denoted by $W_i^g(\ell)$) is calculated by [20, 22] as follows.

$$W_i^g(\ell) \triangleq F_i^g(\ell + d_i^g - e_i^g), \tag{7}$$

where $F_i^g(\ell)$ is defined as:

$$F_i^g(\ell) \triangleq \begin{cases} \left\lfloor \frac{\ell}{p_i^g} \right\rfloor \cdot e_i^g + \min\left(e_i^g, \ell - \left\lfloor \frac{\ell}{p_i^g} \right\rfloor \cdot p_i^g\right), & \text{if } l > 0, \\ 0, & \text{otherwise,} \end{cases} \tag{8}$$

and the physical meaning of $F_i^g(\ell)$ is the amount of execution of jobs of τ_i^g in an interval of length ℓ, when the first job is released at the beginning of the interval and all jobs of τ_i^g in the interval execute as early as possible.

In case of transitional tasks, it is more difficult to find the maximum amount of execution of jobs of τ_i because the amount depends not only on release and execution patterns, but also on the time of the MTR. However, we discover a property regarding the maximum amount, as stated in the following observation.

Observation 1. *Suppose that τ makes a transition from M^g to M^h in an interval of length ℓ, in which the scheduling window of at least one job of τ_i^g and at least one job of τ_i^h (either partially or entirely) overlap with the interval as shown in Figs. 3(b) and (c). Then, the amount of execution of jobs of both τ_i^g and τ_i^h in the interval is maximized with one of the following release and execution patterns: (i) when the first job of τ_i^g is executed as late as possible and starts its execution at the beginning of the interval (at a), and the last job of τ_i^h is executed as early as possible as shown in Fig. 3(b), and (ii) when the last job of τ_i^h executes as early as possible and finishes its execution at the end of the interval (at b), and the first job of τ_i^g is executed as late as possible as shown in Fig. 3(c).*

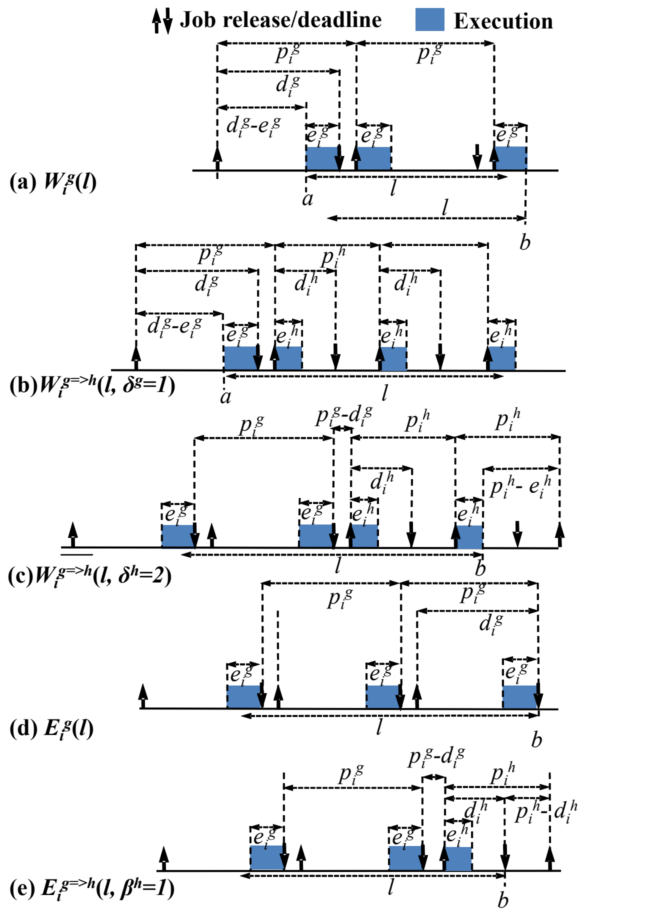


Fig. 3. Release and execution patterns that derive $W_i^g(\ell)$, $W_i^{g\Rightarrow h}(\ell, \delta^g = 1)$, $W_i^{g\Rightarrow h}(\ell, \delta^h = 2)$, $E_i^g(\ell)$ and $E_i^{g\Rightarrow h}(\ell, \delta^h = 1)$, where an interval of interest of length ℓ either starts at a or ends at b.

If a release pattern belongs to neither (i) nor (ii), shifting the release pattern towards (i) or (ii) yields the larger (or at least an equal) amount of execution of jobs of both τ_i^g and τ_i^h. With a fixed release pattern, it is straightforward that the execution pattern of (i) and (ii) maximizes the amount of execution. Therefore, the observation holds.

A native approach requires an exhaustive search for the release time of an MTR since we develop a schedulability analysis that satisfies R1, meaning that an MTR can be released at any time in the interval of interest. Using Observation 1, however, we are allowed to look at only the following two types of finite patterns.

P1. Release and execution patterns of (i) with a situation where the scheduling window of a given number of jobs of τ_i^g (denoted by δ^g) overlaps with the interval of interest.

P2. Release and execution patterns of (ii) with a situation where the scheduling window of a given number of jobs of τ_i^h (denoted by δ^h) overlaps with the interval of interest.

Fig. 3(b) depicts P1 with $\delta^g = 1$. Then, the amount of execution of jobs of τ_i^g in the interval of interest (starting at a) is equal to $\delta^g \cdot e_i^g$, and that of τ_i^h is calculated by $F_i^h(\ell + d_i^g - e_i^g - \delta^g \cdot p_i^g)$ as shown in Fig. 3(b). Therefore, the amount of execution of jobs of τ_i^g and τ_i^h in the interval of length ℓ with P1 and given δ^g (denoted by $W_i^{g\Rightarrow h}(\ell, \delta^g)$) is calculated by

$$W_i^{g\Rightarrow h}(\ell, \delta^g) \triangleq \delta \cdot e_i^g + F_i^h(\ell + d_i^g - e_i^g - \delta^g \cdot p_i^g), \tag{9}$$

where $1 \le \delta^g \le \lfloor (\ell + d_i^g - e_i^g)/p_i^g \rfloor$.

Similarly, Fig. 3(c) depicts P2 with $\delta^h = 2$. Then, the amount of execution of jobs of τ_i^h in the interval of interest (ending at b) is equal to $\delta^h \cdot e_i^h$, and that of τ_i^g is calculated by $F_i^g(\ell + p_i^h - e_i^h - (p_i^g - d_i^g) - \delta^h \cdot p_i^h)$ as shown in Fig. 3(c). Therefore, the amount of execution of jobs of τ_i^g and τ_i^h in the interval of length ℓ with P2 and given δ^h (denoted by $W_i^{\overline{g\Rightarrow h}}(\ell, \delta^h)$) is calculated by

$$W_i^{\overline{g\Rightarrow h}}(\ell, \delta^h) \triangleq \delta^h \cdot e_i^h + F_i^g(\ell + p_i^h - e_i^h - (p_i^g - d_i^g) - \delta^h \cdot p_i^h), \tag{10}$$

where $1 \le \delta^h \le \lfloor (\ell + p_i^h - e_i^h)/p_i^h \rfloor$.

Then, an upper-bound of the amount of execution of jobs of τ_i^g and τ_i^h in an interval of length ℓ is the maximum among the cases where only jobs of τ_i^g are executed in the interval (when the MTR occurs after the interval) and only jobs of τ_i^h are executed in the interval (when the MTR occurs before the interval), and the cases of P1 and P2 (in which the MTR occurs within the interval). The upper-bound (denoted by $\mathbf{W}_i^{g\Rightarrow h}(\ell)$) is calculated as:

$$\mathbf{W}_i^{g\Rightarrow h}(\ell) \triangleq \max\Big\{ W_i^g(\ell), \quad W_i^h(\ell), \max_{1\le\delta^g\le\lfloor(\ell+d_i^g-e_i^g)/p_i^g\rfloor} W_i^{g\Rightarrow h}(\ell, \delta^g), \max_{1\le\delta^h\le\lfloor(\ell+p_i^h-e_i^h)/p_i^h\rfloor} W_i^{\overline{g\Rightarrow h}}(\ell, \delta^h)\Big\}. \tag{11}$$

Under FP, a job of τ_i can interfere with another job τ_k only when the job of τ_i is executed and τ_i has a higher task-level priority than τ_k. Therefore, $\mathbf{I}(\tau_k^u \leftarrow \tau_i^{g\Rightarrow h})$ (u is either g or h) in Lemma 2 under FP is upper-bounded by $\mathbf{W}_i^{g\Rightarrow h}(d_k^u)$ if τ_i has a higher priority than τ_k; otherwise, $\mathbf{I}(\tau_k^u \leftarrow \tau_i^{g\Rightarrow h}) = 0$.

2) Upper-bound of amount of interference under EDF: EDF determines jobs' priorities based on their deadlines; a job with an earlier deadline has a higher priority than another job with a later deadline. Therefore, a job J_A can interfere with another job J_B only when the deadline of J_A is no later than that of J_B. Then, we derive a property that can be used to derive an upper-bound of the interference under EDF, as stated in the following observation.

Observation 2. *Suppose that τ makes a transition from M^g to M^h in an interval of length ℓ, in which at least one job of τ_i^g and at least one job of τ_i^h (either partially or entirely) overlap with the interval of interest as shown in Fig. 3(e). Then, the amount of execution of jobs of τ_i^g and τ_i^h in the interval whose deadlines are no later than the end of the interval (at b in Fig. 3(e)), is maximized when the deadline of the last job of τ_i^h is at the end of the interval, and the first job of τ_i^g executes as late as possible as shown in Fig. 3(e).*

If we shift the job releases a little bit later, the last job's deadline is later than the end of the interval. Shifting the job releases to the other way also does not increase the amount of execution. Therefore, the observation holds.

Let β^h denote the number of jobs of τ_i^h whose scheduling windows overlap with the interval of interest when the release and execution patterns accord with Observation 2; for example, β^h is equal to 1 in Fig. 3(e). Then, the amount of execution of jobs of τ_i^h in the interval is equal to $\beta^h \cdot e_i^h$, and that of τ_i^g is calculated by $F_i^g(\ell + p_i^h - d_i^h - (p_i^g - d_i^g) - \beta^h \cdot p_i^h)$. Therefore, the amount of execution of jobs of τ_i^g and τ_i^h in an interval of length ℓ whose deadlines are no later than the end of the interval (denoted by $E_i^{g\Rightarrow h}(\ell, \beta^h)$), is calculated by

$$E_i^{g\Rightarrow h}(\ell, \beta^h) \triangleq \beta^h \cdot e_i^h + F_i^g(\ell + p_i^h - d_i^h - (p_i^g - d_i^g) - \beta^h \cdot p_i^h)), \tag{12}$$

where $1 \le \beta^h \le \lfloor (\ell + p_i^h - d_i^h)/p_i^h \rfloor$.

Similar to $\mathbf{W}_i^{g\Rightarrow h}(\ell)$, an upper-bound of the amount of execution of jobs of τ_i^g and τ_i^h in an interval of length ℓ whose deadlines are no later than the end of the interval, is the maximum of the cases where the MTR occurs outside the interval (either only jobs of τ_i^g or those of τ_i^h execute within the interval, which is calculated by $E_i^g(\ell) \triangleq F_i^g(\ell)$ or $E_i^h(\ell) \triangleq F_i^h(\ell)$ [20, 22] as shown in Fig. 3(d)), and the cases where MTR occurs in the interval ($E_i^{g\Rightarrow h}(\ell, \beta^h)$ with different β^h). Then, the upper-bound (denoted by $\mathbf{E}_i^{g\Rightarrow h}(\ell)$) is calculated by

$$\mathbf{E}_i^{g\Rightarrow h}(\ell) \triangleq \max\Big(E_i^g(\ell), E_i^h(\ell), \max_{1\le\beta^h\le\lfloor(\ell+p_i^h-d_i^h)/p_i^h\rfloor} E_i^{g\Rightarrow h}(\ell, \beta^h)\Big). \tag{13}$$

Since a job with a later deadline cannot interfere with another job with an earlier deadline under EDF, $\mathbf{I}(\tau_k^u \leftarrow \tau_i^{g\Rightarrow h})$ (u is either g or h) in Lemma 2 is upper-bounded by $\mathbf{E}_i^{g\Rightarrow h}(d_k^u)$ under EDF.

C. Deadline-based schedulability analysis and its property

Using Lemma 2 and the derived upper-bounds on the amount of interference, we derive the deadline-based schedulability analysis for a transition under EDF and FP as follows:

Theorem 1. *Suppose that a task set τ makes a transition from M^g to M^h. Then, a task set τ with the transition is schedulable under FP (likewise EDF), if Eq.* (14) *(likewise, Eq.* (15)*) holds for all $\tau_k \in \tau$ and $u \in \{g, h\}$.*

$$\sum_{\tau_i\in\tau\setminus\{\tau_k\}} \min\Big(\mathbf{W}_i^{g\Rightarrow h}(d_k^u), d_k^u - e_k^u + 1\Big) < m \cdot (d_k^u - e_k^u + 1). \tag{14}$$

$$\sum_{\tau_i \in \tau \setminus \{\tau_k\}} \min\Big(\mathbf{E}_i^{g \Rightarrow h}(d_k^u), d_k^u - e_k^u + 1\Big) < m \cdot (d_k^u - e_k^u + 1). \tag{15}$$

Note that Eq. (14) *holds only when τ_i has a higher priority than τ_k; otherwise, $\mathbf{W}_i^{g \Rightarrow h}(d_k^u)$ should be replaced by* 0.

Proof: The theorem holds by Lemma 2 and the derivation of $\mathbf{W}_i^{g \Rightarrow h}(\ell)$ and $\mathbf{E}_i^{g \Rightarrow h}(\ell)$. ■

Note that the time-complexity of the theorem is $O(|\tau|^2 \cdot D_{max})$, where $D_{max} = \max_{\tau_i \neq \tau_k \in \tau} \frac{\max(d_k^g, d_k^h)}{\min(p_i^g, p_i^h)}$. Then, the proposed schedulability analysis has the following property.

Lemma 3. *The proposed schedulability analysis in Theorem 1 meets R1–R3 in Section III.*

Proof: Due to space limitation, we refer readers to Appendix I in the supplementary file [25] for a full proof. ■

V. Optimal Transition Order Assignment

As proved in Lemma 3, the proposed schedulability analysis in Theorem 1 satisfies the design requirements R1–R3. In particular, the analysis is independent of job release and execution patterns, and the release time of an MTR, in meeting R1. However, this requirement is met at the expense of over-estimating the interference, as discussed below.

In Fig. 2, after the release of an MTR from M^g to M^h at t_1, the first jobs of τ_1^h and τ_3^h are released earlier than t_2, while that of τ_2^h is released at t_2. Therefore, the scheduling window of any job of τ_2^h does not overlap with that of any job of tasks in M^g. If we enforce such a transition order of tasks within a single transition (e.g., M^g to M^h), we can rule out other tasks in mode M^g in the calculation of the amount of interference to τ_2^h during the transition from M^g to M^h. This reduces the interference and increases the possibility of τ_2^h's schedulability.

Therefore, we consider a task-level transition protocol that allows only one task's transition at a time, as opposed to the task-set-level transition in Section III that allows multiple tasks to transit from one mode to another concurrently. We call a series of single task-level transitions *a sequential transition*, and the task-set-level transition (described in Section III), *any-order transition*. A sequential transition can improve the schedulability guarantees over the corresponding any-order transition, and this improvement can be achieved at the expense of increasing the system's transition time.

There are two challenges in making timing guarantees with the sequential transition: (i) how to guarantee the timing requirements with the sequential transition, and (ii) how to find the best transition order of tasks, whose schedulability is guaranteed by (i). Now, we first address (i) by modifying the schedulability analysis proposed in Theorem 1.

Suppose that a task set τ makes a sequential transition from M^g to M^h with a given order. To express the relative transition order of tasks, let $\tau_k^{g \Rightarrow h} \prec \tau_i^{g \Rightarrow h}$ denote a situation where τ_k's transition from M^g to M^h is performed before τ_i's transition, meaning that the scheduling window of any job of τ_k^g cannot overlap with that of any job of τ_i^h, e.g., $\tau_1^{g \Rightarrow h} \prec \tau_2^{g \Rightarrow h}$ holds in Fig. 2. Therefore, if $\tau_k^{g \Rightarrow h} \prec \tau_i^{g \Rightarrow h}$ holds, any job of τ_i^h cannot interfere with any job of τ_k^g. This implies that the interference of $\tau_i^{g \Rightarrow h}$ on τ_k^g is reduced by that of τ_i^g on τ_k^g. Hence, if $\tau_k^{g \Rightarrow h} \prec \tau_i^{g \Rightarrow h}$ holds, the upper-bound of $\mathbf{I}(\tau_k^g \leftarrow \tau_i^{g \Rightarrow h})$ under FP (when τ_i has a higher priority than τ_k) in Theorem 1 is changed from $\mathbf{W}_i^{g \Rightarrow h}(d_k^g)$ to ${}^{\mathsf{S}}\mathbf{W}_i^{g \Rightarrow h}(d_k^g)$, where

$${}^{\mathsf{S}}\mathbf{W}_i^{g \Rightarrow h}(d_k^g) \triangleq \begin{cases} W_i^g(d_k^g), & \text{if } \tau_k^{g \Rightarrow h} \prec \tau_i^{g \Rightarrow h} \text{ (reduced)}, \\ \mathbf{W}_i^{g \Rightarrow h}(d_k^g), & \text{if } \tau_k^{g \Rightarrow h} \succ \tau_i^{g \Rightarrow h} \text{ (no change)}. \end{cases} \tag{16}$$

On the other hand, any job of τ_i^g cannot interfere with any job of τ_k^h, if $\tau_k^{g \Rightarrow h} \succ \tau_i^{g \Rightarrow h}$ holds. Therefore, an upper-bound of $\mathbf{I}(\tau_k^h \leftarrow \tau_i^{g \Rightarrow h})$ by FP (when τ_i has a higher priority than τ_k) in Theorem 1 is changed from $\mathbf{W}_i^{g \Rightarrow h}(d_k^h)$ to ${}^{\mathsf{S}}\mathbf{W}_i^{g \Rightarrow h}(d_k^h)$, where

$${}^{\mathsf{S}}\mathbf{W}_i^{g \Rightarrow h}(d_k^h) \triangleq \begin{cases} \mathbf{W}_i^{g \Rightarrow h}(d_k^h), & \text{if } \tau_k^{g \Rightarrow h} \prec \tau_i^{g \Rightarrow h} \text{ (no change)}, \\ W_i^h(d_k^h), & \text{if } \tau_k^{g \Rightarrow h} \succ \tau_i^{g \Rightarrow h} \text{ (reduced)}. \end{cases} \tag{17}$$

Note that all theories developed in this section can be applied to EDF, by applying the same modification for $\mathbf{E}_i^{g \Rightarrow h}(d_k^u)$. Due to the space limit, our description is confined to FP.

We can now derive a schedulability analysis for a sequential transition, as stated in the following theorem.

Theorem 2. *Suppose that a task set τ makes a sequential transition from M^g to M^h with a given order. Then, a task set τ with the transition is schedulable under FP, if the following inequality holds for all $\tau_k \in \tau$ and $u \in \{g, h\}$:*

$$\sum_{\tau_i \in \tau \setminus \{\tau_k\}} \min\Big({}^{\mathsf{S}}\mathbf{W}_i^{g \Rightarrow h}(d_k^u), d_k^u - e_k^u + 1\Big) < m \cdot (d_k^u - e_k^u + 1). \tag{18}$$

Proof: The theorem holds by Lemma 2 and the derivation of ${}^{\mathsf{S}}\mathbf{W}_i^{g \Rightarrow h}(\ell)$. ■

By the definition of $\mathbf{W}_i^{g \Rightarrow h}(d_k^u)$, the upper-bound of the amount of interference with sequential transition (i.e., ${}^{\mathsf{S}}\mathbf{W}_i^{g \Rightarrow h}(d_k^u)$) is always smaller than or equal to that with any-order transition (i.e., $\mathbf{W}_i^{g \Rightarrow h}(d_k^u)$). Therefore, the following observation holds.

Observation 3. *Suppose that τ makes a transition from M^g to M^h. If τ_k^g and τ_k^h are deemed schedulable with any-order transition (proven by Theorem 1), then they are also deemed schedulable with any sequential transition order (proven by Theorem 2).*

As shown in Eqs. (16) and (17), enforcing the task-level sequential order will reduce (or at least stay) the interference on each task, yielding a possibility of finding additional schedulable task sets, which are not deemed schedulable with any-order transition. Then, here comes an important question: "How can we find an optimal transition order of tasks?" By "optimal order," we mean that a task set in the presence of a sequential transition with the order is guaranteed to be schedulable by the proposed schedulability analysis, as long as there is at least one transition order to be schedulable. Since an exhaustive search requires to investigate $O(|\tau|!)$ transition orders, we need to develop an efficient way to find an optimal order. To achieve this, we present how the transition order of a given task affects the interference of other tasks on the task itself, as stated in the following observation.

Observation 4. *Suppose that τ makes a sequential transition from M^g to M^h. If τ_k's transition order is placed in the first (last), the left-hand side of Eq.* (18) *for τ_k^g is minimized (maximized) while that for τ_k^h is maximized (minimized).*

As shown in Eq. (16), if $\tau_k^{g\Rightarrow h} \prec \tau_i^{g\Rightarrow h}$ holds, the upper-bound on $\mathbf{I}(\tau_k^g \leftarrow \tau_i^{g\Rightarrow h})$ is reduced from $\mathbf{W}_i^{g\Rightarrow h}(d_k^g)$ to $W_i^g(d_k^g)$. Therefore, the left-hand side of Eq. (18) for τ_k^g is minimized, if the order of τ_k's transition is the earliest. Likewise, the other case for τ_k^g and both cases for τ_k^h hold.

Using the above observations, we now derive some properties of optimal transition order assignment. Let us focus only on the schedulability of τ_k (i.e., both τ_k^g and τ_k^h), not on that of other tasks. If τ_k^h is schedulable with any-order transition (proven by Theorem 1), placing τ_k's transition order in the earliest position not only maximizes the possibility of the schedulability of τ_k^g as shown in Observation 4, but also guarantees the schedulability of τ_k^h by Observation 3 (regardless of the transition order). However, such an assignment favorable for τ_k's schedulability may increase the interference of τ_k on other tasks. To address this, we introduce two notations.

First, we use $\tau_k^g \overset{I(\tau')}{>} \tau_k^h$ (likewise, $\tau_k^g \overset{I(\tau')}{<} \tau_k^h$), if $\min(\mathbf{W}_k^{g\Rightarrow h}(d_i^u), d_i^u - e_i^u + 1) = \min(W_k^g(d_i^u), d_i^u - e_i^u + 1)$ (likewise, $\min(\mathbf{W}_k^{g\Rightarrow h}(d_i^u), d_i^u - e_i^u + 1) = \min(W_k^h(d_i^u), d_i^u - e_i^u + 1)$) holds for all $\tau_i \in \tau' \setminus \{\tau_k\}$ and $u \in \{g, h\}$. Its physical meaning is that the interference of $\tau_k^{g\Rightarrow h}$ on other tasks is dominated by that of τ_k^g. Thus, $\min({}^{\mathsf{S}}\mathbf{W}_k^{g\Rightarrow h}(d_i^u), d_i^u - e_i^u + 1)$ in Eq. (18) is fixed at $\min(W_k^g(d_i^u), d_i^u - e_i^u + 1)$ regardless of transition order.

Second, let τ^* denote a set of tasks in τ, which are schedulable with any-order transition (proven by Theorem 1); then, any task $\tau_k \in \tau^*$ with M^g and that with M^h are schedulable with any sequential transition order according to Observation 3.

Let us focus on a task τ_k which satisfies that (i) τ_k^h is schedulable with any-order transition (proven by Theorem 1), and (ii) $\tau_k^g \overset{I(\tau\setminus\tau^*)}{>} \tau_k^h$ holds. Then, we determine the order of τ_k by considering two aspects: (a) τ_k is schedulable or not; and (b) τ_k makes other tasks schedulable or not. For (a), we should place τ_k's transition in the earliest position, because this placement maximizes the chance of the schedulability of τ_k^g by Observation 4 and τ_k^h is schedulable with any sequential transition order by Observation 3. For (b), placing τ_k's transition in the first yields smaller $\min({}^{\mathsf{S}}\mathbf{W}_k^{g\Rightarrow h}(d_i^h), d_i^h - e_i^h + 1)$ ($= \min(W_k^h(d_i^h), d_i^h - e_i^h + 1)$) for a given τ_i^h while $\min({}^{\mathsf{S}}\mathbf{W}_k^{g\Rightarrow h}(d_i^g), d_i^g - e_i^g + 1)$ for a given τ_i^g is independent of transition order. Therefore, in terms of (i) and (ii), τ_k's transition should be performed earliest. Note that we do not consider tasks in τ^* for (b) because they are schedulable with

Algorithm 1 Optimal transition order assignment framework

```
1: for $\tau_k \in \tau$ do
2:     Check whether $\tau_k^g \overset{I(\tau\setminus\tau^*)}{>} \tau_k^h$ and $\tau_k^h$ is schedulable with any-order transition (i.e., Theorem 1). If so, add $\tau_i$ to $\tau(1)$.
3:     Check whether $\tau_k^g \overset{I(\tau\setminus\tau^*)}{<} \tau_k^h$ and $\tau_k^g$ is schedulable with any-order transition (i.e., Theorem 1). If so, add $\tau_i$ to $\tau(3)$.
4: end for
5: Determine the first $|\tau(1)|$ orders by tasks in $\tau(1)$.
6: Determine the next $|\tau(2)|$ orders by tasks in $\tau(2) \triangleq \tau \setminus (\tau(1) \cup \tau(3))$.
7: Determine the last $|\tau(3)|$ orders by tasks in $\tau(3)$.
```

any sequential order by Observation 3.

With this reasoning, we develop an optimal transition-order assignment framework that identifies three groups in Algorithm 1. In Steps 2 and 3, the algorithm identifies two groups of tasks, whose transition orders should be placed the earliest ($\tau(1)$) and latest ($\tau(3)$) as shown in Steps 5 and 7, respectively; the remaining tasks belong to the second group ($\tau(2)$) as shown in Step 6. Then, Lemma 4 proves its optimality.

Lemma 4. *Suppose that τ makes a sequential transition from M^g to M^h. Then, Algorithm 1 yields an optimal transition order.*

Proof: See Appendix I in the supplementary file [25] for a full proof. ■

The remaining step is then to determine a transition order of tasks within individual groups $\tau(1)$, $\tau(2)$ and $\tau(3)$ in Steps 5–7 in Algorithm 1. The following lemma finds an optimal transition order for tasks in $\tau(1)$ and $\tau(3)$.

Lemma 5. *Suppose that τ makes a sequential transition from M^g to M^h with a given order compliant with Algorithm 1. The relative transition order of tasks within $\tau(1)$ (likewise, $\tau(3)$) in Algorithm 1 does not change the schedulability of any task in τ.*

Proof: See Appendix I in the supplementary file [25] for a full proof. ■

Therefore, by applying Algorithm 1 with any arbitrary order for tasks in $\tau(1)$ and $\tau(3)$, we can derive an optimal transition order except for the relative transition order of tasks in $\tau(2)$. To determine a transition order for tasks in $\tau(2)$, we may apply an exhaustive search (if $|\tau(2)|$ is small) or some heuristic transition order assignment algorithms.

In Algorithm 2, we introduce a heuristic transition order assignment algorithm. The algorithm uses a property derived for Algorithm 1: if $\min(\mathbf{W}_k^{g\Rightarrow h}(d_i^u), d_i^u - e_i^u + 1)$ is equal to $\min(W_k^g(d_i^u), d_i^u - e_i^u + 1)$ for all $\tau_i \in \tau \setminus \{\tau_k\}$ and $u = \{g, h\}$, placing τ_k's transition order in the first minimizes its interference on other tasks. To approximate the property, each task τ_k calculates $\overline{W_k}$, which sums the ratio between $\min(\mathbf{W}_k^{g\Rightarrow h}(d_i^u), d_i^u - e_i^u + 1)$ and $\min(W_k^g(d_i^u), d_i^u - e_i^u + 1)$ for all $\tau_i \in \tau \setminus \{\tau_k\}$ and $u \in \{g, h\}$, as shown in Steps 4–7.

Algorithm 2 Heuristic transition order assignment algorithm

```
1:  $\mathcal{T} \leftarrow []$ (empty list).
2:  $\overline{W}_k \leftarrow 0$ for all $\tau_k \in \tau$ (new variables).
3:  for $\tau_k \in \tau$ do
4:      for $\tau_i \in \tau \setminus \{\tau_k\}$ do
5:          $\overline{W}_k \leftarrow \overline{W}_k + \frac{\min(\mathbf{W}_k^{g \Rightarrow h}(d_i^g), d_i^g - e_i^g + 1)}{\min(W_k^g(d_i^g), d_i^g - e_i^g + 1)}$.
6:          $\overline{W}_k \leftarrow \overline{W}_k + \frac{\min(\mathbf{W}_k^{g \Rightarrow h}(d_i^h), d_i^h - e_i^h + 1)}{\min(W_i^g(d_k^h), d_i^h - e_i^h + 1)}$.
7:      end for
8:  end for
9:  while TRUE do
10:     for $\tau_i \in \tau$ in ascending order of $\overline{W}_i$ do
11:         if $\tau_i^g$ and $\tau_i^h$ are schedulable with a $\tau_i$'s transition
            order such that $\tau_i \prec \tau_j$ for all $\tau_j \in \tau \setminus \{\tau_i\}$ then
12:             $\tau \leftarrow \tau \setminus \{\tau_i\}$.
13:             $\mathcal{T} \leftarrow \mathcal{T} \mid [\tau_i]$.
14:             Go to Step 18.
15:         end if
16:     end for
17:     Return INFEASIBLE.
18:     if $\tau = \emptyset$ then
19:         Return $\mathcal{T}$.
20:     end if
21: end while
```

The algorithm assigns transition order of tasks from the earliest to the latest, and each assignment examines unassigned tasks in ascending order of $\overline{W_k}$, such that both modes of a task are schedulable if the task is assigned to the current order. Such an examination is possible, because the relative order of other unassigned tasks does not affect the task's schedulability as shown in Observation 5 in Appendix I of the supplementary file [25]. Steps 10–16 describe each assignment.

The time-complexity of Algorithm 1 without determining the relative order of tasks in $\tau(2)$ is $O(|\tau|^2 \cdot D_{max})$, which is the same as that of Theorem 1 for a single task set. If we apply Algorithm 2 for transition order assignment of tasks in $\tau(2)$, the total time-complexity of transition order assignment with its schedulability test is polynomial in $|\tau|$, i.e., $O(|\tau|^3 \cdot D_{max})$, while the exhaustive search with its schedulability test requires an exponential time-complexity in $|\tau|$, i.e., $O(|\tau|! \cdot |\tau^2| \cdot D_{max})$.

In Section VI, we will evaluate the optimal transition order assignment framework in Algorithm 1 with our heuristic assignment in Algorithm 2, in terms of the number of schedulable task sets.

VI. Evaluation

In this section, we demonstrate the effectiveness of the optimal transition order assignment framework and the heuristic assignment algorithm, in terms of the number of schedulable task sets.

We generate task sets, each of which has two mode M^g and M^h (in order to express a transition from M^g to M^h). We tailor a popular method [26], and Appendix II in the supplementary file [25] details our task set generation. For generated 10,000 task sets for each $m = 2, 4, 8$ and 16, we compare the number of schedulable task sets with different transitions using their corresponding schedulability analyses:

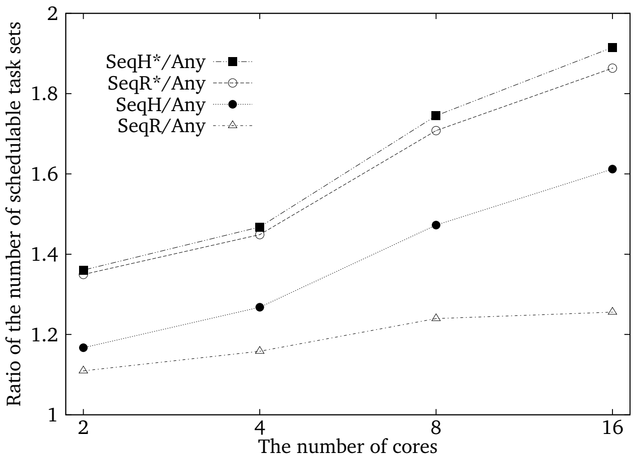


Fig. 4. The ratio between the number of schedulable task sets by SeqR, SeqH, SeqR* or SeqH*, and that by Any

- Any-order transition (denoted by Any);
- A sequential transition—the entire order is determined randomly (denoted by SeqR);
- A sequential transition—the entire order is determined by Algorithm 2 (denoted by SeqH);
- A sequential transition—the entire order is grouped by Algorithm 1, and then the relative order of tasks in $\tau(2)$ is determined randomly[3] (denoted by SeqR*); and
- A sequential transition—the entire order is grouped by Algorithm 1, and then the relative order of tasks in $\tau(2)$ is determined by Algorithm 2 (denoted by SeqH*).

Here, the schedulability of Any is judged by Theorem 1, while that of others is proven by Theorem 2. We only present the simulation results for FP, since those for EDF exhibit a similar trend.

Fig. 4 shows the ratio between the number of schedulable task sets by SeqR, SeqH, SeqR* or SeqH*, and that by Any, with different m. The most important observation is the effectiveness of the optimal transition order assignment framework in Algorithm 1. That is, if we compare SeqR with SeqR*, and SeqH with SeqH*, the framework finds up to 44.7% and 17.3% additional schedulable task sets, respectively. Also, the improvement becomes significant as the number of cores is increasing. For example, compared to SeqR, SeqR* covers additional 20.8%, 23.8%, 37.1% and 44.7% task sets when $m = 2, 4, 8$ and 16, respectively. This implies that the framework is scalable to the number of cores.

The heuristic transition order assignment in Algorithm 2 increases the number of schedulable task sets, compared to a random ordering. If we compare SeqR with SeqH, its improvement on the number of schedulable task sets is up to 26.8%. However, the improvement is marginal if the optimal

[3]For SeqR* and SeqH*, the relative order of tasks in $\tau(1)$ and $\tau(3)$ is determined randomly because the optimality holds with any sequential transition order within the groups according to Lemma 5.

transition order assignment framework is applied; SeqH* finds only up to 3.3% additional schedulable task sets which are not deemed schedulable by SeqR*. This is because the optimal framework is so effective that there is only limited room for improvement.

In summary, the optimal transition order assignment framework in Algorithm 1 outperforms the other approaches, and our heuristic transition order assignment in Algorithm 2 is a good candidate to assign the transition order of the remaining group (i.e., $\tau(2)$) in the optimal framework. Then, SeqH*, our best approach in which Algorithms 1 and 2 are applied together, finds up to 91.5% additional schedulable task sets, which are not covered by Any.

VII. Conclusion

In this paper, we addressed the problem of guaranteeing the timing requirements of task sets with mode transitions without disrupting task execution in real-time multi-core systems. By generalizing a popular schedulability analysis for non-transitional tasks, we developed an offline, sufficient schedulability analysis, which does not require any online information. To improve the analysis, we enforced the transition order of tasks, and solved the optimal transition sequence assignment problem by deriving the useful properties of an optimal transition order.

Although we focused on a deadline-based schedulability analysis due to its simplicity, low time-complexity and independence of the history of previous operational modes, it would be interesting to extend other analyses to handle a mode-transition, e.g., a response-time analysis, which is known to have better schedulability performance at the expense of higher time-complexity. Since the response time is affected by previous modes, there will be many challenging issues to be addressed. For example, it is necessary to address how to manage the history of response times of each task with previous modes and how to handle the case where such information is not available, for example, due to system overhead.

Acknowledgement

We are grateful to Dr. Shige Wang at GM R&D Center for helpful comments. The work reported in this paper was supported in part by the NSF under grants CNS-0930813, CNS-1138200, and CNS-1329702.

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. Wahler, S. Richter, and M. Oriol, “Dynamic software updates for real-time systems,” in *Proceedings of the 2nd International Workshop on Hot Topics in Software Upgrades*, 2009.

[3] M. Wahler, S. Richter, S. Kumar, and M. Oriol, “Non-disruptive large-scale component updates for real-time controller,” in *Proceedings of the 27th International Conferences on Data Engineering Workshops*, 2011, pp. 174–178.

[4] J. C. Romero and M. Garcia-Valls, “Scheduling component replacement for timely execution in dynamic systems,” *To appear in Software - Practice and Experience*, vol. -, no. -, pp. –, -.

[5] L. Sha, R. Rajkumar, J. Lehoczky, and K. Ramamritham, “Mode change protocols for priority-driven preemptive scheduling,” *Real-Time Systems*, vol. 1, no. 3, pp. 243–264, 1989.

[6] K. W. Tindell, A. Burns, and A. J. Wellings, “Mode changes in priority pre-emptively scheduled systems,” in *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, 1992, pp. 100–109.

[7] J. Real and A. Crespo, “Mode change protocols for real-time systems: A survey and a new proposal,” *Real-Time Systems*, vol. 26, no. 2, pp. 161–197, 2004.

[8] M. Ahmed, N. Fisher, and D. Grosu, “A parallel algorithm for EDF-schedulability analysis of multi-modal real-time systems,” in *Proceedings of IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2012, pp. 154–163.

[9] J. Kim, K. Lakshmanan, and R. Rajkumar, “Rhythmic tasks: A new task model with continually varying periods for cyber-physical systems,” in *Proceedings of IEEE/ACM International Conference on Cyber-Physical Systems (ICCPS)*, 2012, pp. 55–64.

[10] G. C. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni, “Elastic scheduling for flexible workload management,” *IEEE Transactions on Computers*, vol. 51, no. 3, pp. 289–302, 2002.

[11] Q. Guangming, “An earlier time for inserting and/or accelerating tasks,” *Real-Time Systems*, vol. 41, no. 3, pp. 181–194, 2009.

[12] V. Nelis, J. Goossens, and B. Andersson, “Two protocols for scheduling multi-mode real-time systems upon identical multiprocessor platforms,” in *Proceedings of Euromicro Conference on Real-Time Systems (ECRTS)*, 2009, pp. 151–160.

[13] V. Nelis, B. Andersson, J. Marinho, and S. M. Petters, “Global-EDF scheduling of multimode real-time systems considering mode independent tasks,” in *Proceedings of Euromicro Conference on Real-Time Systems (ECRTS)*, 2011, pp. 205–214.

[14] P. Rattanatamrong and J. A. B. Fortes, “Mode transition for online scheduling of adaptive real-time systems on multiprocessors,” in *Proceedings of IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2011, pp. 25–32.

[15] L. T. X. Phan, S. Chakraborty, and P. S. Thiagarajan, “A multi-mode real-time calculus,” in *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, 2008, pp. 59–69.

[16] N. Stoimenov, S. Perathoner, and L. Thiele, “Reliable mode changes in real-time systems with fixed priority or EDF scheduling,” in *Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2009, pp. 99–104.

[17] Y. Hang and H. Hansson, “Timing analysis for mode switch in component-based multi-mode systems,” in *Proceedings of Euromicro Conference on Real-Time Systems (ECRTS)*, 2012, pp. 255–264.

[18] 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.

[19] T. P. Baker, “Multiprocessor EDF and deadline monotonic schedulability analysis,” in *RTSS*, 2003.

[20] M. Bertogna, M. Cirinei, and G. Lipari, “Improved schedulability analysis of EDF on multiprocessor platforms,” in *Proceedings of Euromicro Conference on Real-Time Systems (ECRTS)*, 2005, pp. 209–218.

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

[22] 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.

[23] T. P. Baker, M. Cirinei, and M. Bertogna, “EDZL scheduling analysis,” *Real-Time Systems*, vol. 40, pp. 264–289, 2008.

[24] J. Lee, A. Easwaran, and I. Shin, “LLF schedulability analysis on multiprocessor platforms,” in *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, 2010, pp. 25–36.

[25] J. Lee and K. G. Shin, “Supplement: schedulability analysis for continuous task execution during a mode transition in real-time multi-core systems,” http://kabru.eecs.umich.edu/papers/publications/2013/supplement.pdf.

[26] T. P. Baker, “Comparison of empirical success rates of global vs. paritioned fixed-priority and EDF scheduling for hand real time,” Dept. of Computer Science, Florida State University, Tallahasee, Tech. Rep. TR-050601, 2005.