

# Techniques for Schedulability Analysis in Mode Change Systems under Fixed-Priority Scheduling

Wen-Hung Huang and Jian-Jia Chen  
Department of Computer Science  
TU Dortmund University, Germany  
{wen-hung.huang, jia.chen}@tu-dortmund.de

**Abstract**—With the advent of cyber-physical systems, real-time tasks shall be run in different modes over time to react to the change of the physical environment. It is preferable to adopt high expressive models in real-time systems. In the light of simple implementation in kernels, fixed-priority scheduling has been widely adopted in commercial real-time systems. In this work we derive a technique for analyzing schedulability of the system where tasks can undergo mode change under fixed-priority scheduling. We study two types of fixed-priority scheduling in mode change systems: task-level and mode-level fixed-priority scheduling. The proposed tests run in polynomial time. We further show that a utilization of  $2 - \sqrt{2} \approx 0.5857$  can be guaranteed in implicit-deadline multi-mode systems if each mode is prioritized according to rate-monotonic policy. The effectiveness of the proposed tests is also shown via extensive simulation results.

## 1 Introduction

In the last decade, accessible networks and sensor devices have become ubiquitous. This gives rise to Cyber-Physical Systems (CPS) in which a system is designed as a network of interacting elements with physical input and output. Such an embedded real-time system continuously monitors and affects the physical environment which also interactively imposes the feedback to the embedded system.

In CPS, the characteristics of a real-time task may be able to change over time, e.g., the computational demand or the resource allocation. Such behavior is referred to as *mode changes*. The importance of mode changes for real-time systems has been pointed out in many perspectives, for instance, aircraft control systems, automotive Electronic Control Units (ECU) [10], [18], energy management [20], and server-based systems [7], [16], [19].

In automotive systems, the engine control has different computational demands according to different angular rotations. In each periodic interval, the engine control software calculates the engine speed and position to determine when to fire the next spark signal and evaluates the acceleration/deceleration commands from the driver to adjust the settings of fuel flow [10]. Such a control has the nature of computation mode changes according to the physical environment. Besides, the stringent timing requirement has to be met to inject and to deliver fuel to each cylinder at every revolution.

On the other hand, when the server-based system is used to achieve temporal isolation among applications, the reservation server parameters may need to change from one mode to

another [16], [19]. Hence, an additional guarantee is required to ensure feasibility not only in the steady state but also during mode changes.

**Related mode change models.** In the classical problem of mode changes in hard real-time systems [21], [24], on the request of mode change, all tasks have to switch to their new parameters. Before the new mode of the task system is fully established, no further mode changes are permitted. Several real-time task models have been proposed recently in the literature to analyze the schedulability of adaptive embedded systems [21], [24].

In this paper, the mode changes are associated to individual tasks. Such mode changes can be found in the generalized multiframe (GMF) model [3], digraph real-time model (DRT) [22], and variable rate-dependent behavior (VRB) task model [6], [8], [10], [13]. The generalized multiframe (GMF) model [3] allows a task to cycle through a static list of job types, each with potentially different WCET bounds and relative deadlines. Stigge et al. [22] propose a more expressive model, called digraph real-time model (DRT), in which the release structures of different types of jobs are represented by a directed graph. Some approaches on sufficient schedulability tests for DRT model for fixed priority scheduling have been reported [23]. In addition, the recent study by Guan et al. [12] presents two pseudo-polynomial-time approximations to improve the efficiency by losing some accuracy. In automotive applications, several tasks are linked to rotation (e.g., of the crankshaft, gears, or wheels). Thus their activation rate is proportional to the angular velocity of a specific device. In such a system a common practice is to design a rate-dependent task, called variable rate-dependent behavior (VRB) task model [6], [8], [10], [13]. Typically, at lower rotation some functions that minimize fuel consumption and emissions have to be executed, whereas they are shed at higher rotation speeds to reduce the processor utilization. Table I illustrates an example of a task with four levels of functionality, specified for different speed intervals.

**TABLE I: An example of variable-rate behavior task with four types of execution modes dependent on the rotation speed.**

rotation (rpm)	functions to be executed
[0, 2000]	$f1(); f2(); f3(); f4();$
(2000, 4000]	$f1(); f2(); f3();$
(4000, 6000]	$f1(); f2();$
(6000, 8000]	$f1();$

**This work and contribution.** In this work we study a real-time system comprised of  $n$  independent, preemptive uniprocessor *multi-mode* tasks  $\tau = \{\tau_1, \tau_2, \dots, \tau_n\}$  where each task has several execution *modes* to switch during the runtime. The cost of synchronization on mode change within a task is assumed to be subsumed into the worst-case execution time of each mode. A multi-mode task  $\tau_i$  is denoted by a set of triplets:

$$\begin{aligned} \tau_i = \{ & \tau_i^1 = (C_i^1, T_i^1, D_i^1), \\ & \tau_i^2 = (C_i^2, T_i^2, D_i^2), \dots, \\ & \tau_i^{M_i} = (C_i^{M_i}, T_i^{M_i}, D_i^{M_i}) \} \end{aligned}$$

$C_i^m$  denotes the *worst-case execution time* (WCET) of task  $\tau_i$  under mode  $m$ ,  $T_i^m$  denotes the *minimum inter-arrival time* of task  $\tau_i$  under mode  $m$ , and  $D_i^m$  denotes the *relative deadline*. That is, when a job of mode  $\tau_i^m$  is released at time  $t$ , the next release time of task  $\tau_i$  is no earlier than  $t + T_i^m$ , and when a job of mode  $\tau_i^m$  is released at time  $t$ , this job has to be finished no later than its *absolute deadline* at time  $t + D_i^m$ .

The studied mode change model is a generalization of the sporadic model [17]. The concept of mode change is distinct from that of system-wide operating modes [21], [24]. This model characterizes the system where different *tasks* may progress through their different execution modes independent of each other.

Even though the studied mode change model essentially differs from the VRB model where different angular sources drive the inter-arrival time, the mode change model is still applicable when considering only those thresholds that determine which level of functionality should be executed, and in fact the worst-case scenario occurs at the particular value of thresholds. Similar concepts have been presented in [10]. In general, the mode change model can be thought of as a relaxation model of variable rate-dependent behavior (VRB) task model [10], [13] and digraph real-time model (DRT) [22]. From the designer's perspective, the studied mode change model provides an easier way to specify and reason comparing to the more general DRT model.

This paper studies two scheduling algorithms upon multi-mode task systems: fixed-priority *task-level* (FPT) and *mode-level* (FPM). In this work, we derive a technique for analyzing the schedulability in multi-mode systems. Based on this technique, we propose tests that leverage the accuracy and the time complexity for both FPT and FPM scheduling in multi-mode systems.

We summarize our contributions as follows:

- 1) The proposed test can be efficiently run in polynomial time and is shown to be comparable to the existing state-of-the-art pseudo-polynomial tests for FPT scheduling.
- 2) Previous tests for determining whether multi-mode systems can be successfully scheduled are only applicable to the system under FPT scheduling. In this work we show that the FPT scheduling can perform rather poorly in the worst case.
- 3) We prove that a utilization bound of  $2 - \sqrt{2} \approx 0.5857$  can be guaranteed in implicit-deadline multi-mode systems under rate-monotonic (RM) scheduling, one example of FPM scheduling. Moreover, empirical results show that our proposed tests under RM scheduling are able to accept task sets that are deemed schedulable with utilizations of

up to 80%.

To the best of our knowledge, this is the first work studying the mode change systems under FPM scheduling.

## 2 System Properties and Notations

In this work we study a real-time system to execute a set of  $n$  independent, preemptive uniprocessor real-time tasks  $\tau = \{\tau_1, \tau_2, \dots, \tau_n\}$ . As presented and defined in Section 1, a multi-mode task  $\tau_i$  with  $M_i$  modes is denoted by a set of triplets:  $\tau_i = \{\tau_i^1 = (C_i^1, T_i^1, D_i^1), \tau_i^2 = (C_i^2, T_i^2, D_i^2), \dots, \tau_i^{M_i} = (C_i^{M_i}, T_i^{M_i}, D_i^{M_i})\}$ , to specify the worst-case execution time  $C_i^m$ , the minimum inter-arrival time  $T_i^m$ , and the relative deadline  $D_i^m$  of the corresponding task mode.

Throughout this paper, we restrict ourselves to either *constrained-deadline* ( $D_i^m \leq T_i^m$ ) or *implicit-deadline* ( $D_i^m = T_i^m$ ) multi-mode task systems. For any multi-mode task  $\tau_i = \{\tau_i^1 = (C_i^1, T_i^1, D_i^1), \tau_i^2 = (C_i^2, T_i^2, D_i^2), \dots, \tau_i^{M_i} = (C_i^{M_i}, T_i^{M_i}, D_i^{M_i})\}$ , the utilization  $U_i^m$  of mode  $\tau_i^m$  denotes the ratio  $C_i^m/T_i^m$  of its worst-case execution time to the minimum inter-arrival time. We only consider meaningful cases, in which  $\sum_{i=1}^n (\max_{m=1,2,\dots,M_i} U_i^m) \leq 1$ .

We do not impose any constraint on the mode changes between two modes except the temporal distance specified by the minimal inter-arrival time. When a job of task  $\tau_i$  under the execution mode  $m$  is released at time  $t$ , the next job of task  $\tau_i$  cannot be released earlier than  $t + T_i^m$ , independent of its next mode. Which mode to be selected as the next mode depends upon the required system properties for reacting to the physical environment or the configuration of the system, which is completely independent from the scheduler. As a result, the difficulty to analyze and schedule such mode change tasks is to precisely quantify and consider the worst-case execution patterns with mode changes.

In this paper we keep our focus on fixed-priority scheduling. We highlight the advantage of fixed-priority scheduling over dynamic scheduling, e.g., *earliest-deadline-first* (EDF), by its light overhead of the implementation. In commercial real-time kernel the explicit support for timing constraints, such as absolute deadlines, is not needed under fixed-priority scheduling.

There are two potential categories of associating the fixed priority level: fixed-priority *task-level* (FPT) and *mode-level* (FPM). In FPM scheduling algorithms, the priority of a mode does not change during runtime; however, different modes of the same task may have different priorities. In contrast, fixed-priority task-level (FPT) algorithms require that all modes of a task have the same priority. It is evident from these definitions that FPM scheduling is a generalization of FPT scheduling. The Rate Monotonic (RM) scheduling algorithm is an example of FPM scheduling algorithms. The RM scheduling algorithm prioritizes each mode according to its period generated by the task: the smaller the period, the higher the priority assigned. Despite that FPM allows different priority levels for different modes of a task, all the jobs generated by a task mode have the same priority level, whereas the jobs of a task mode in EDF have different priority levels, depending on the absolute deadlines.

The response time of a job in a mode is defined as the completion time of the job minus the release time of the job.

The worst-case response time  $R_k^h$  of task  $\tau_k$  for its mode  $h$  under a scheduling policy is defined as the longest response time among all released jobs by the mode.

**Definition 1** (Schedulability). *A task  $\tau_k$  is schedulable under a scheduling policy if for every execution mode  $h$  of the task,  $R_k^h \leq D_k^h$ , and a task set is schedulable under a scheduling policy if all of its tasks are schedulable under the scheduling policy.*

We will analyze the schedulability of a task mode  $\tau_k^h$  under the interference of higher-priority tasks (under FPT) or higher-priority task modes (under FPM). This requires proper definitions of the maximum execution time and the maximum utilization of task  $\tau_i$  among the modes that are assigned with higher priority than mode  $\tau_k^h$ . For FPT, as all the modes of a task  $\tau_i$  are either with higher priority or lower priority than task mode  $\tau_k^h$ , we define

$$C_i^{max} = \max_{\tau_i^m \in \tau_i} (C_i^m)$$

and

$$U_i^{max} = \max_{\tau_i^m \in \tau_i} (U_i^m)$$

With the above definition, we denote the total utilization as  $U^{sum} = \sum_{i=1}^n U_i^{max}$ , in which  $U^{sum} \leq 1$ . Note that the task modes resulting in  $C_i^{max}$  and  $U_i^{max}$  may be different. We further define the load factor of task  $\tau_i$  as follows:

$$\beta_i = \frac{C_i^{max}}{U_i^{max}}$$

For FPM, let  $hp(\tau_k^h)$  denote the *set* of the modes that are assigned with higher priority than mode  $\tau_k^h$ . Therefore, the intersect set  $hp(\tau_k^h) \cap \tau_i$  consists of the task modes of task  $\tau_i$  that are assigned with higher priority than  $hp(\tau_k^h)$ . Similarly, the maximum execution time and the maximum utilization of task  $\tau_i$  among the modes that are assigned with higher priority than mode  $\tau_k^h$  are denoted as follows:

$$C_i^{max}(\tau_k^h) = \max_{\tau_i^m \in (hp(\tau_k^h) \cap \tau_i)} (C_i^m)$$

and

$$U_i^{max}(\tau_k^h) = \max_{\tau_i^m \in (hp(\tau_k^h) \cap \tau_i)} (U_i^m)$$

Similarly, the load factor of task  $\tau_i$  with respect to mode  $\tau_k^h$  in FPM is

$$\beta_i(\tau_k^h) = \frac{C_i^{max}(\tau_k^h)}{U_i^{max}(\tau_k^h)}$$

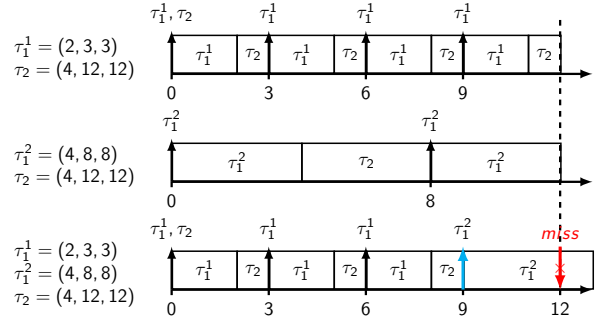
Note that if task  $\tau_i$  does not have any higher-priority task mode than mode  $\tau_k^h$ , we can simply remove such a task  $\tau_i$  in the schedulability analysis of task mode  $\tau_k^h$ .

### 3 Problem Statement and Existing Results

The objective of the mode change schedulability analysis is to guarantee that a system is feasible not only in the steady state but also during the mode transition. In mode change systems, during mode transitions, the phenomenon of demand strides may occur and lead to an unfeasible scheduling, even if there is no deadline miss in the steady state. The following example will further motivate the difficulty of mode change.

**Example** Consider a system with one multi-mode task and one sporadic task:  $\tau_1 = \{(2, 3, 3), (4, 8, 8)\}$  and  $\tau_2 = \{(4, 12, 12)\}$ . Both tasks are schedulable by RM without mode transition.

However, task  $\tau_2$  will miss its deadline at time-instant 12 when task  $\tau_1$  switches from mode  $\tau_1^1$  to mode  $\tau_1^2$  at time-instant 9, as illustrated in Figure 1.  $\square$



**Figure 1: The missed deadline during mode transition**

As a result, it is inevitable that the combinatorial releases have to be taken into account to identify the worst-case scenario. Towards this end, we shall first inspect multi-mode tasks' *critical instant*, which is defined as the instant at which the execution of a task will have the longest response time [14]. For sporadic tasks with constrained deadlines, it is proven that the critical instant for a task occurs whenever the task is released simultaneously with all higher priority tasks and all the following jobs are released as early as possible [14].

The critical instant for a multi-mode task under FPT scheduling has been presented in Theorem 1 in [10].<sup>1</sup> For completeness, we paraphrase this theorem in the following lemma:

**Lemma 1** (Davis et al. [10]). *There is a sequence  $Y$  of jobs of task  $\tau_i$ , that  $\tau_i$  releases the maximum interference  $I_i(w)$  in a window  $[0, w]$ , where*

- 1) *the offset, from the start of the window, of the first job of  $\tau_i$  is zero;*
- 2) *each of the jobs of  $\tau_i$  released in the window has the minimum period commensurate with its particular execution mode;*
- 3) *the last job has the largest WCET for any execution mode.*

We notice that unlike for the sporadic task, the critical instant provided in Lemma 1 is only necessary for creating the maximum interference. The *exact* worst-case interference from all tasks has to enumerate all possible release sequences that satisfy the above conditions for all the tasks, but is, however, computationally intractable since the time complexity is  $O(W_1 \times W_2 \times \dots \times W_{n-1})$  where  $W_i$  is the possible release sequences within the input period and is  $\geq M_i$ . As an amenable solution, instead of enumerating all possible sequences for all tasks, a method using integer linear programming (ILP) solvers for deriving the maximum demand has been proposed in [10]. Also, a dynamic programming method has been reported in [22] for the DRT model, which is a generalization of mode change model. However, both approaches may lead to an over-approximation of the exact response time, due to the non-concrete traces.

<sup>1</sup>Although the focus in [10] was for the VRB task model, the critical instant theorem in [10] works for the mode change task model in this paper as well. The proof to create the maximum interference is exactly the same.

## 4 Schedulability Analysis under FPT Scheduling

In this section, we first introduce the concept of the multi-mode demand bound function, and derive a sufficient schedulability test for multi-mode task systems under a given FPT assignment, in Theorem 1.

By Lemma 1, no matter how the sequence of task  $\tau_i$  releases prior to the arrival time of the last release, we can replace the last release mode in  $[0, w)$  by the mode with the largest WCET among all modes without decreasing the interference. Based on this observation, we can decompose the interference from a multi-mode task into two parts: (i) the execution time from the last release that has the largest WCET among modes and (ii) the total demand prior to the arrival time of the last release. We formally define the last release time and multi-mode demand bound function as follows:

**Definition 2** (Last Release Time). *For a given sequence of releases of task  $\tau_i$  in the interval  $[t_0, t_0 + t)$ , we define  $t_i$  as the time of the last release of task  $\tau_i$  upon the sequence.*

**Definition 3** (Multi-mode demand bound function). *For any interval length of  $t$ , the demand bound function  $DBF_i(t, \tau_k^h)$  of a multi-mode task  $\tau_i$  is defined as the maximum cumulative execution requirement by jobs of  $\tau_i$  that are assigned with higher priority than mode  $\tau_k^h$  and have both arrive time and next release time within an interval length of  $t$ .*

### 4.1 Multi-Mode Demand Bound Function

The concept of demand bound function (DBF) has been widely used in real-time schedulability analysis [4]. The conventional demand bound function (DBF) [4] bounds the maximum cumulative execution requirement by jobs of sporadic task  $\tau_i$  that both arrive in and have absolute deadlines within any interval of length  $t$ . As for the multi-mode demand bound function, the function involves the combinatorial releases.

In fact, by the definition of the multi-mode demand bound function, calculating the multi-mode demand bound function is equivalent to the well-known *unbounded knapsack problem* (UKP) [15]. The *unbounded knapsack problem* is to determine the number of each item to include in a collection of items so that total weight (execution time) of the selected items is less than or equal to a given limit (interval length) (called knapsack) and total value (cumulative executions) of the selected items is maximized.

As a result, the multi-mode demand bound function can be computed in pseudo-polynomial time using *dynamic programming* [15]. Furthermore, it has been shown in [15] that an upper bound  $B$  for UKP is

$$B = \left\lceil c \frac{p_1}{w_1} \right\rceil \quad (1)$$

where  $p_i$  denotes the *profit* of an item of type  $i$ ,  $w_i$  denotes the *weight* of an item of type  $i$ ,  $c$  is the limit of the knapsack, and the item types are ordered so that

$$\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \dots \geq \frac{p_n}{w_n} \quad (2)$$

In the above transformations,  $\frac{p_1}{w_1}$  is corresponding to the maximum utilization among the eligible modes. Lemma 2 follows immediately:

**Lemma 2.** *For any  $\tau_i$  and a higher-priority mode  $\tau_k^h$ , for  $t > 0$*

$$DBF_i(t, \tau_k^h) \leq t \cdot U_i^{max}(\tau_k^h)$$

*Proof:* Notice that  $\frac{p_1}{w_1} = U_i^{max}(\tau_k^h)$ .

$$\begin{aligned} \text{By Eq. (1)} \quad DBF_i(t, \tau_k^h) &\leq \lfloor t U_i^{max}(\tau_k^h) \rfloor \\ &\Rightarrow DBF_i(t, \tau_k^h) \leq t \cdot U_i^{max}(\tau_k^h) \end{aligned}$$

■

It is worth noting that by definition it is possible that the mode belonging to  $U_i^{max}(\tau_k^h)$  has a period  $> t$  and thus cannot both arrive in and have next release time within interval length of  $t$ . Nevertheless, the upper bound still holds. We will use the upper bound on the multi-mode demand function to derive a schedulability technique that requires the *continuity* of the upper bound with the interval length of  $t$ . The mode having period larger than  $t$  can be removed only if the interval of interest is known, e.g., under FPT  $0 < t \leq D_k^h$ . Due to space limitation, we do not explicitly present the detail.

### 4.2 Sufficient Test

In this section, we use the concept of the interference decomposition to derive a technique for analyzing schedulability of real-time systems represented using the mode change model.

Consider any legal sequence of jobs of task system  $\tau$ , on which a deadline miss occurs. Suppose that a mode  $m$  of the  $k$ th-highest priority task is the one to first miss a deadline and that the mode arrives at time-instant  $t_a$  and this deadline miss occurs at time-instant  $t_a + D_k^h$ .

Without loss of generality, by Lemma 1, we set  $t_a = 0$ . We first assume that the tasks assigned with higher priority than mode  $\tau_k^h$  are indexed according to the last release time ordering  $\pi$ , upon which the deadline miss occurs.

**Definition 4** (Last Release Time Ordering). *Let  $\pi$  be the assignment of a last release time ordering as a bijective function  $\pi : \tau \rightarrow \{1, 2, \dots, k-1\}$  to define the last release time ordering of task  $\tau_i \in hp(\tau_k^h)$ . The ordering of last releases is numbered from 1 to  $k-1$  where 1 is the earliest and  $k-1$  the latest.*

We now derive a necessary condition for a deadline miss to occur with a last release time ordering  $\pi$  in the following lemma:

**Lemma 3.** *If task mode  $\tau_k^h$  misses its deadline under FPT upon a last release assignment  $\pi$ , then there exists an assignment  $t_1, t_2, \dots, t_{k-1} \in [0, D_k^h)$  of the last release of task  $\tau_i$  such that  $\forall i \in \{1, \dots, k-1, k\}$*

$$\sum_{j=1}^{k-1} DBF_j(t_j, \tau_k^h) + \sum_{j=1}^{i-1} C_j^{max} + C_k^h > t_i \quad (3)$$

where  $t_k \equiv D_k^h$ .

*Proof:* By Lemma 1 and the assumption of the deadline miss of mode  $\tau_k^h$ , the released pattern described in Lemma 1 will result in a deadline miss for the job released by mode  $\tau_k^h$  at time  $t_a = 0$ . Let  $\pi$  be such a last release ordering, and define the last release time  $t_i$  of task  $\tau_i$  for  $i = 1, 2, \dots, k-1$  accordingly with  $t_i \leq t_{i+1}$ . The executed workload of the job

released by mode  $\tau_k^h$  must be strictly less than  $C_k^h$  amount of execution time over  $[0, D_k^h]$ . We observe the followings:

- At time point  $t_i$  for any  $i = 1, 2, \dots, k-1$ , the requested higher-priority execution time prior to  $t_i$  plus  $C_k^h$  is larger than  $t_i$ .
- Up to time  $t_i$ , a task  $\tau_j$  with  $j = 1, 2, \dots, i-1$  has requested  $DBF_j(t_j, \tau_k^h) + C_j^{max}$  amount of execution time to be executed.
- Up to time  $t_i$ , a task  $\tau_j$  with  $j = i, i+1, \dots, k-1$  has requested  $DBF_j(t_i, \tau_k^h)$  amount of execution time to be executed.

The above observation results in  $\forall i = 1, 2, \dots, k$ ,

$$C_k^h + \sum_{j=1}^{i-1} (DBF_j(t_j, \tau_k^h) + C_j^{max}) + \sum_{j=i}^{k-1} DBF_j(t_i, \tau_k^h) > t_i$$

By the fact that  $DBF_j(t, \tau_k^h)$  is monotonically non-decreasing with respect to the interval length  $t$ , we know that  $DBF_j(t_i, \tau_k^h) \leq DBF_j(t_j, \tau_k^h)$  when  $j \geq i$ . (Due to the last release ordering  $\pi$ , we have  $t_j \geq t_i$  for such cases.) As a result, we reach the conclusion in Eq. (3). ■

Without knowing the exact last release times  $t_i$ , all possible last release time-instants of  $t_i$  have to be enumerated by combinatorial releases as shown in [10]. In contrast, in this work we are aiming at obtaining tests that leverage the accuracy and the time complexity (in polynomial). In the following lemma we provide a necessary condition for a deadline miss by maximizing the higher-priority tasks' interference in Eq. (3), depending on the assignment of last release times:

**Lemma 4.** *If mode  $\tau_k^h$  misses its deadline upon a last release time assignment  $\pi$ , it must be either the case that*

$$C_k^h + \sum_{i=1}^{k-1} C_i^{max} > D_k^h \quad (4)$$

or

$$C_k^h > D_k^h - \sum_{i=1}^{k-1} \left( U_i^{max} \cdot \left( D_k^h - \sum_{j=i}^{k-1} C_j^{max} \right) \right) - \sum_{i=1}^{k-1} C_i^{max} \quad (5)$$

*Proof:* In the case of Eq. (4), it is clear that there will be a deadline miss.

We now consider the case where Eq. (4) does not hold. From Lemma 3, we know that it is necessary for a deadline miss to occur: there exists an assignment  $t_1, t_2, \dots, t_{k-1} \in [0, D_k^h)$  of the last release of task  $\tau_i$ ,  $\forall i \in \{1, \dots, k-1, k\}$

$$\begin{aligned} & \sum_{j=1}^{k-1} DBF_j(t_j, \tau_k^h) + \sum_{j=1}^{i-1} C_j^{max} + C_k^h > t_i \\ \text{(By Lemma 2)} \Rightarrow & \sum_{j=1}^{k-1} U_j^{max} t_j + \sum_{j=1}^{i-1} C_j^{max} + C_k^h > t_i \end{aligned}$$

where  $t_k \equiv D_k^h$ . Our objective is to find the infimum  $C_k^h$  such that the above constraints always hold. In fact, this is equivalent to the following linear programming (LP):

inf  $C^*$

$$\text{s.t. } \sum_{j=1}^{k-1} U_j^{max} t_j + \sum_{j=1}^{i-1} C_j^{max} + C^* > t_i, \quad \forall 1 \leq i \leq k \quad (6a)$$

$$t_i \leq t_{i+1}, \quad \forall 1 \leq i \leq k-1 \quad (6b)$$

$$t_i \geq 0, \quad \forall 1 \leq i \leq k-1 \quad (6c)$$

where Eq. (6c) and Eq. (6b) come from the definition of  $t_i$  and  $t_k \equiv D_k^h$  for notational brevity. We now replace  $>$  with  $\geq$  in Eq. (6a) as infimum and minimum are the same if  $\geq$  is used.

From Eq. (6a) when  $i = k$ , we get  $C^* \geq t_k - \sum_{j=1}^{k-1} U_j^{max} t_j + \sum_{j=1}^{k-1} C_j^{max}$ . Then, we can use this inequality by adding a slack variable  $s \geq 0$  into its RHS to replace  $C^*$  in our objective, and thus finding the minimum  $C_k^h$  is equivalent to finding the maximum  $\sum_{j=1}^{k-1} U_j^{max} t_j - s$  as  $t_k$  and  $\sum_{j=1}^{k-1} C_j^{max}$  are constant. Additionally, by replacing  $C^*$  in Eq. (6a), we get

$$\begin{aligned} & \sum_{j=1}^{k-1} U_j^{max} t_j + \sum_{j=1}^{i-1} C_j^{max} + t_k - \sum_{j=1}^{k-1} U_j^{max} t_j + \sum_{j=1}^{k-1} C_j^{max} + s \geq t_i \\ \equiv & t_k - \sum_{j=i}^{k-1} C_j^{max} + s \geq t_i, \quad \forall 1 \leq i \leq k-1 \end{aligned}$$

After reformulation, we have the following LP:

$$\max \sum_{j=1}^{k-1} U_j^{max} t_j - s$$

$$\text{s.t. } t_k - \sum_{j=i}^{k-1} C_j^{max} + s \geq t_i, \quad \forall 1 \leq i \leq k-1 \quad (7a)$$

$$t_i \leq t_{i+1}, \quad \forall 1 \leq i \leq k-1 \quad (7b)$$

$$s, t_i \geq 0, \quad \forall 1 \leq i \leq k-1 \quad (7c)$$

The objective function is maximized when  $t_i$  is maximized and  $s$  is minimized if all the constraints in Eq. (7a), Eq. (7b) and Eq. (7c) are feasible. From Eq. (7a), any increase  $\Delta$  on  $s$  will increase the feasible range of  $t_i$  by at most  $\Delta$ , but, at the same time, the objective function decreases due to the assumption that  $\sum_{i=1}^n U_i \leq 1$ , i.e.,  $\Delta \sum_{i=1}^n U_i - \Delta \leq 0$ . Thus, the objective function is maximized when  $s = 0$ . From Eq. (7a),  $t_i$  is upper bounded by  $t_k - \sum_{j=i}^{k-1} C_j^{max}$ , by assuming  $s = 0$ . In other words, if there are no constraints violated by setting  $t_i = t_k - \sum_{j=i}^{k-1} C_j^{max}$  and  $s = 0$ , then the objective function is maximized.

By setting  $t_i = t_k - \sum_{j=i}^{k-1} C_j^{max}$  and  $s = 0$ , the given condition that Eq. (4) does not hold implies  $t_i$ 's non-negativity by Eq. (7c), i.e.,  $t_k - \sum_{j=i}^{k-1} C_j^{max} \geq D_k^h - \sum_{j=1}^{k-1} C_j^{max} \geq C_k^h > 0$ . In addition, this setting assures that  $t_i \leq t_{i+1}$  by Eq. (7b). This immediately follows that the maximum  $\sum_{j=1}^{k-1} U_j^{max} t_j - s$  occurs when all the constraints in Eq. (7a) are active and  $s = 0$ , and thus we have that for  $1 \leq i \leq k-1$

$$t_i = t_k - \sum_{j=i}^{k-1} C_j^{max} \quad (8)$$

Replacing  $t_i$  in  $C^* = t_k - \sum_{j=1}^{k-1} U_j^{max} t_j + \sum_{j=1}^{k-1} C_j^{max} - s$  by the above equalities and  $s = 0$ , we obtain the minimum  $C^*$ , as represented in the RHS of Eq. (5). Thus, we conclude

that if mode  $\tau_k^h$  misses its deadline upon a last release time assignment  $\pi$  and Eq. (4) does not hold, it must be the case that  $C_k^h > C^*$ . Hence, this lemma is proven. ■

However, the necessary condition above for a deadline miss is established upon a given last release ordering  $\pi$ . Without knowing the exact ordering observed in the schedule, intuitively, one may relax the ordering assumption by examining all the possible permutations. However, this is computationally intractable as  $(k-1)!$  permutations have to be necessarily checked. Fortunately, the following lemma shows that  $D_k^h - \sum_{i=1}^{k-1} \left( U_i^{max} \cdot \left( D_k^h - \sum_{j=i}^{k-1} C_j^{max} \right) \right) - \sum_{i=1}^{k-1} C_i^{max}$  is minimized for a specific ordering of  $\pi$ .

**Lemma 5.**  $D_k^h - \sum_{i=1}^{k-1} \left( U_i^{max} \cdot \left( D_k^h - \sum_{j=i}^{k-1} C_j^{max} \right) \right) - \sum_{i=1}^{k-1} C_i^{max}$  is minimized when the last release time ordering  $\pi$  of the  $k-1$  higher priority tasks is with a non-increasing order of  $\beta_i$ .

*Proof:* Suppose that  $\pi^*$  does not follow a non-increasing order of  $\beta_i$ . That is, there exists  $\ell$  in the ordering  $\pi^*$  such that  $\beta_\ell < \beta_{\ell+1}$  for some  $\ell = 1, 2, \dots, k-2$ . We can now swap these two tasks, and this results in a new last release time ordering  $\pi'$ . By inspecting the term to be proved, we know that the last release ordering only changes  $\sum_{i=1}^{k-1} \left( U_i^{max} \sum_{j=i}^{k-1} C_j^{max} \right)$ . Therefore, to prove the lemma, we just have to show that the ordering  $\pi'$  has smaller  $\sum_{i=1}^{k-1} \left( U_i^{max} \sum_{j=i}^{k-1} C_j^{max} \right)$  than the ordering  $\pi^*$ . By comparing these two orderings  $\pi^*$  and  $\pi'$ , the term  $\left( U_i^{max} \sum_{j=i}^{k-1} C_j^{max} \right)$  remains the same in  $\pi$  and  $\pi'$  for any  $i \neq \ell$  and  $i \neq \ell+1$ . Therefore, the difference (the result by using  $\pi'$  minus the result by using  $\pi^*$ ) in the term  $\sum_{i=1}^{k-1} \left( U_i^{max} \sum_{j=i}^{k-1} C_j^{max} \right)$  is  $\left( U_{\ell+1}^{max} \sum_{j=\ell}^{k-1} C_j^{max} + U_\ell^{max} \sum_{j=\ell+1}^{k-1} C_j^{max} \right) - \left( U_\ell^{max} \sum_{j=\ell}^{k-1} C_j^{max} + U_{\ell+1}^{max} \sum_{j=\ell+1}^{k-1} C_j^{max} \right)$ , which results in

$$\begin{aligned} U_{\ell+1}^{max} C_\ell^{max} - U_\ell^{max} C_{\ell+1}^{max} &= U_{\ell+1}^{max} U_\ell^{max} \left( \frac{C_\ell^{max}}{U_\ell^{max}} - \frac{C_{\ell+1}^{max}}{U_{\ell+1}^{max}} \right) \\ &= U_{\ell+1}^{max} U_\ell^{max} (\beta_\ell - \beta_{\ell+1}) <^1 0, \end{aligned}$$

where the last inequality comes from the definition of  $\ell$  and  $<^1$  is due to the non-increasing order of  $\beta_i$ .

Therefore, we can keep swapping the last release ordering to reduce  $\sum_{i=1}^{k-1} \left( U_i^{max} \sum_{j=i}^{k-1} C_j^{max} \right)$ . By adopting the above swapping procedure repeatedly, we reach the conclusion. ■

Since Lemma 4 is the necessary condition for a deadline miss to occur, equivalently, the negation of Lemma 4 is the sufficient condition for a deadline to be met. We can now conclude the following schedulability test by using Lemma 4 and Lemma 5.

**Theorem 1 (QT-FPT).** A constrained-deadline multi-mode task  $\tau_k^h$  is schedulable if

$$D_k^h - \sum_{i=1}^{k-1} C_i^{max} - C_k^h \geq 0$$

and

$$C_k^h \leq D_k^h - \sum_{i=1}^{k-1} \left( U_i^{max} \cdot \left( D_k^h - \sum_{j=i}^{k-1} C_j^{max} \right) \right) - \sum_{i=1}^{k-1} C_i^{max}$$

where the higher priority tasks  $\tau_i$  are indexed by non-increasing  $\beta_i$ .

### 4.3 Computational Complexity

There are  $(M_1 + M_2 + \dots + M_n)$  task modes to be checked for the schedulability. The test for each mode runs in time of  $O(n^2)$  without any optimization while higher-priority tasks have to be sorted by non-increasing  $\beta_i$  beforehand, which requires  $O(n \cdot \log n)$ . Therefore, the proposed test can be computed in polynomial time of  $O(n^2 \cdot (M_1 + M_2 + \dots + M_n))$ .

## 5 Schedulability Analysis under FPM Scheduling

In this section, we derive the schedulability test for multi-mode systems under FPM scheduling by using the similar technique provided in Section 4. Specifically, we study RM scheduling, one example of FPM, and then provide utilization bounds for implicit-deadline multi-mode systems, in Theorem 4 and Theorem 5.

Several results have been reported on FPT scheduling [10], [23]. We here show that FPT scheduling may perform rather poorly comparing to FPM scheduling.

**Example.** Consider the set of tasks defined in Table II where task  $\tau_1$  is a sporadic task;  $\tau_2$  is a multi-mode task;  $0 < \epsilon \leq 0.5$ . If  $\tau_1$  has higher priority than  $\tau_2$ , mode  $\tau_2^1$  cannot meet its deadline. Similarly, if  $\tau_2$  has higher priority than  $\tau_1$ , task  $\tau_1$  cannot meet its deadline due to the preemption resulting from the mode  $\tau_2^2$ .

Task	Mode	$C_i^m$	$T_i^m$	$D_i^m$
$\tau_1$	1	1	$1/\epsilon$	$1/\epsilon$
$\tau_2$	1	$\epsilon$	1	1
	2	$1/\epsilon$	$(1/\epsilon)^2$	$(1/\epsilon)^2$

**TABLE II: An example of associating fixed priority with the task set**

According to the example, when FPT is adopted, a feasible scheduling that will always meet deadlines does not exist, no matter task  $\tau_1$  or task  $\tau_2$  is assigned to the higher priority for any  $\epsilon > 0$ . Therefore, the utilization bound for such a case is 0 when  $\epsilon$  is close to 0. That is, there exist input instances with a very small utilization such that the task set is not schedulable under any FPT assignments.

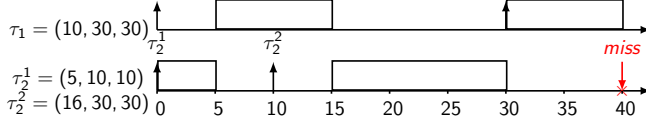
A naive way to check the schedulability under RM is to pessimistically consider each mode as an individual sporadic task. Subsequently, by inspecting the Liu & Layland bound [14], the schedulability of the above example can be guaranteed when  $\frac{C_1^1}{T_1^1} + \frac{C_2^1}{T_2^1} + \frac{C_2^2}{T_2^2} \leq 3(2^{\frac{1}{3}} - 1) = 0.779$  and accordingly  $\epsilon \leq \frac{0.779}{3}$ . Nevertheless, the schedulability bound is inversely proportional to the total number of modes in a system, and it thus cannot be put into practice.

To the best of our knowledge, there is no previous work studying on the mode change system under FPM.

## 5.1 Carry-in Effect under FPM Scheduling

Unfortunately, the critical instant provided by Lemma 1 is no longer satisfied under FPM scheduling due to the so-called *carry-in* effect. We show this with the following example.

**Example.** Consider a system with two tasks: a sporadic task  $\tau_1 = \{(10, 30, 30)\}$  and a multi-mode task  $\tau_2 = \{\tau_2^1 = (5, 10, 10), \tau_2^2 = (16, 30, 30)\}$ . We assign the highest, the middle, and the lowest priority to task  $\tau_2^1$ , task  $\tau_1$ , and task  $\tau_2^2$ , respectively. As shown in Figure 2, five additional time-units jobs from mode  $\tau_1$  are carried into the interval of task  $\tau_2^2$ 's releases due to the preemption from higher-priority mode  $\tau_2^1$  under FPM scheduling. Hence, task  $\tau_1^2$  misses its deadline at time 40, whereas it will meet its deadline if released as the *synchronous arrival sequence*, mentioned in Lemma 1.  $\square$



**Figure 2: An example of carry-in effects under FPM.**

## 5.2 Sufficient Test for FPM Scheduling

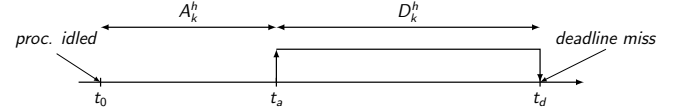
In this section, we first use the concept of problem window extension to quantify the carry-in job and then reuse the technique derived in the previous section for FPT scheduling to establish schedulability tests for FPM scheduling.

Consider any legal sequence of jobs of task system  $\tau$ , on which a deadline miss occurs. Without loss of generality, we assume mode  $\tau_k^h$  is the mode that there exist  $k-1$  tasks such that each task has at least one mode that is assigned higher priority than mode  $\tau_k^h$ . Suppose that mode  $\tau_k^h$  is the one to first miss a deadline and arrives at time-instant  $t_a$ , and this deadline miss occurs at time-instant  $t_a + D_k^h$ .

We now discard all those jobs that have priority lower than  $\tau_k^h$ 's priority from this sequence. Since those jobs with priority lower than  $\tau_k^h$ 's have no effect on the scheduling of the jobs with priority higher than or equals to task  $\tau_k^h$ , this schedule will see a deadline miss of mode  $\tau_k^h$  at time-instant  $t_a + D_k^h$ , and it will also be the first deadline miss in the schedule. Hence in this section, we consider only such a *reduced* sequence of jobs.

Let  $t_0$  denote the latest time-instant with  $t_0 \leq t_a$  at which the processor is idle (or executing some lower-priority jobs before discarding), c.f. Figure 3. Clearly,  $t_0$  exists and is well-defined. Let  $A_k^h = t_a - t_0$  and  $t_d = t_a + D_k^h$ . The technique to extend the interval of interest is needed for identifying the necessary condition for a deadline miss where the carry-in effect may occur. We have the following observation:

- Let  $\mathcal{X}$  denote the workload contributed from task  $\tau_k$ , that is to under analysis, over  $[t_0, t_d)$ . Due to mode  $\tau_k^h$ 's deadline miss, the amount of execution time, executed for mode  $\tau_k^h$ , is strictly less than  $C_k^h$  over  $[t_a, t_d)$ . By the definition of  $t_0$ , the jobs of task  $\tau_k$  that contribute to the interval  $[t_0, t_d)$  must arrive no earlier than  $t_0$ . The jobs of task  $\tau_k$  contributing to  $[t_0, t_a)$  is bounded by



**Figure 3: Illustration: a mode of task  $\tau_i$  arrives at  $t_a$  and misses its deadline at time-instant  $t_d$ . The latest time-instant prior to  $t_a$  when the processor is idle is denoted  $t_0$ .**

the multi-mode demand bound function of task  $\tau_i$  over  $[t_0, t_a)$ . Hence, we have

$$\mathcal{X} \leq C_k^h + DBF_i(t_a - t_0, \tau_k^h) \quad (9)$$

- The higher-priority jobs of any multi-mode task must arrive no earlier than time-instant  $t_0$ . Hence, the work executing prior to the last release time  $t_i$  is bounded from above by its multi-mode demand bound function with an interval length of  $t_i - t_0$ .

Consequently, we provide a necessary condition for a deadline to be missed under FPM scheduling in the following lemma:

**Lemma 6.** *If task mode  $\tau_k^h$  misses its deadline under FPM upon an assignment  $\pi$ , it must be the case that there exist an  $A_k^h \geq 0$  and an assignment  $t_1, t_2, \dots, t_{k-1}$  of the last release of task  $\tau_i$  such that  $\forall i \in \{1, \dots, k-1, k\}$ ,*

$$\mathcal{X} + \sum_{\tau_j \in hp(\tau_k^h)} DBF_j(t_j - t_0, \tau_k^h) + \sum_{j: t_j \leq t_i} C_j^{max}(\tau_k^h) > t_i - t_0$$

and for time  $t_d$

$$\mathcal{X} + \sum_{\tau_j \in hp(\tau_k^h)} DBF_j(t_j - t_0, \tau_k^h) + \sum_{j=1}^{k-1} C_j^{max}(\tau_k^h) > A_k^h + D_k^h$$

*Proof:* This is by the above discussions with a similar proof as in Lemma 3.  $\blacksquare$

Without loss of generality, we can simply set  $t_0$  to 0.

**Theorem 2 (QT-FPM).** *Task mode  $\tau_k^h$  is schedulable under an FPM scheduling if*

$$D_k^h - \sum_{i=1}^{k-1} C_i^{max}(\tau_k^h) - C_k^h \geq 0 \quad (10)$$

and

$$C_k^h \leq D_k^h - \sum_{i=1}^{k-1} (U_i^{max}(\tau_k^h) \cdot t_i^*) - \sum_{i=1}^{k-1} C_i^{max}(\tau_k^h) \quad (11)$$

where

$$t_i^* = D_k^h - \sum_{j=i}^{k-1} C_j^{max}(\tau_k^h)$$

in which higher-priority tasks  $\tau_i$  are indexed by non-increasing  $\beta_i(\tau_k^h)$ .

*Proof:* In case of Eq. (10), it is clear that there will be a deadline miss.



By Lemma 6 and Eq. (9) and using the same concept of Lemma 4 and Lemma 5, we then have the necessary condition for the deadline miss under FPM:

$$C_k^h > D_k^h - \sum_{i=1}^{k-1} (U_i^{max}(\tau_k^h) \cdot t_i^*) - \sum_{i=1}^{k-1} C_i^{max}(\tau_k^h) + A_k^h \left( 1 - \sum_{i=1}^k U_i^{max}(\tau_k^h) \right) \quad (12)$$

where

$$t_i^* = D_k^h - \sum_{j=i}^{k-1} C_j^{max}(\tau_k^h)$$

Notice that due to the assumption that  $U^{sum} \leq 1$ , the RHS of the above inequality is monotonically increasing with the length of interval  $A_k^h$ . By substituting  $A_k^h = 0$  and taking the negation of Eq. (12), this theorem is proven. ■

Roughly speaking, the work contributing over interval  $[t_0, t_d]$  where  $t_0 \neq t_a$  will be no more than that in the case  $t_0 = t_a$ . One may observe that in order to make the interval  $[t_0, t_a]$  busy, there will be a loss of some workloads over  $[t_a, t_d]$  due to  $U^{sum} \leq 1$  according to our analysis.

### 5.3 Utilization Bounds for Implicit-Deadline Tasks under RM

Here, we specifically study RM scheduling in implicit-deadline multi-mode systems where the relative deadline of each mode is equal to its period. Under RM scheduling, we first observe that the interference from the last release can be naturally bounded:

**Lemma 7.** *Under RM scheduling, for any task  $\tau_i$  with respect to mode  $\tau_k^h$*

$$C_i^{max}(\tau_k^h) \leq U_i^{max}(\tau_k^h) \cdot T_k^h$$

*Proof:* Under RM scheduling, only those modes with period  $T_a^b \leq T_k^h$  will be assigned higher priority than  $\tau_k^h$ . By the definition of  $C_i^{max}(\tau_k^h)$  we have that

$$C_i^{max}(\tau_k^h) = \max_{\tau_a^b \in hp(\tau_k^h) \cap \tau_i} \{U_a^b \cdot T_a^b\} \leq U_i^{max}(\tau_k^h) \cdot T_k^h \quad \blacksquare$$

By Lemma 7 we now pessimistically inflate each  $C_i^{max}(\tau_k^h)$  to  $U_i^{max}(\tau_k^h) \cdot T_k^h$ . Note that thereafter we can simply relax the ordering specified in Theorem 2 since according to Theorem 2 for all tasks  $\tau_i$

$$\beta_1 = \beta_2 = \dots = \beta_{k-1} = \frac{C_i^{max}(\tau_k^h)}{U_i^{max}(\tau_k^h)} = T_k^h$$

after the inflation. Theorem 3 immediately follows:

**Theorem 3.** *Implicit-deadline mode  $\tau_k^h$  is schedulable under RM if*

$$U_k^h \leq 1 - 2 \sum_{i=1}^{k-1} U_i^{max}(\tau_k^h) + \frac{1}{2} \left( \sum_{i=1}^{k-1} U_i^{max}(\tau_k^h) \right)^2 + \frac{1}{2} \sum_{i=1}^{k-1} (U_i^{max}(\tau_k^h))^2 \quad (13)$$

*Proof:* In case of Eq. (10), due to  $U^{sum} \leq 1$ , we know that  $T_k^h \cdot \left( 1 - \sum_{i=1}^{k-1} U_i^{max}(\tau_k^h) - U_k^h \right) \geq 0$ . For notational simplicity, we denote  $U_i$  as  $U_i^{max}(\tau_k^h)$  in this proof. In case of Eq. (11), substituting  $C_i^{max}$  in Eq. (11) by  $U_i^{max} \cdot T_k^h$  and  $D_k^h$  by  $T_k^h$ , we have

$$\begin{aligned} \frac{C_k^h}{T_k^h} &\leq 1 - \sum_{i=1}^{k-1} \left( U_i \cdot \left( 1 - \sum_{j=i}^{k-1} U_j \right) \right) - \sum_{i=1}^{k-1} U_i \\ &= 1 - 2 \sum_{i=1}^{k-1} U_j + \sum_{i=1}^{k-1} \left( U_i \cdot \left( \sum_{j=i}^{k-1} U_j \right) \right) \\ &= 1 - 2 \sum_{i=1}^{k-1} U_i + \frac{1}{2} \left( \sum_{i=1}^{k-1} U_i \right)^2 + \frac{1}{2} \sum_{i=1}^{k-1} U_i^2 \end{aligned}$$

Hence this theorem is proven. ■

The schedulability test proposed in the previous section must check each task mode  $\tau_k^h$  individually. If one is willing to sacrifice some precision, we can further provide two linear-time schedulability tests for a task set, called *quadratic bound* (QB) and *total utilization bound*, in Theorem 4 and 5, respectively.

**Theorem 4** (QB-RM). *A multi-mode task set  $\tau$  with implicit-deadline is schedulable under RM scheduling if*

$$U_a \leq 1 - 2 \sum_{\tau_i \in \tau \setminus \{\tau_a\}} U_i^{max} + \frac{1}{2} \left( \sum_{\tau_i \in \tau \setminus \{\tau_a\}} U_i^{max} \right)^2 + \frac{1}{2} \sum_{\tau_i \in \tau \setminus \{\tau_a\}} (U_i^{max})^2 \quad (14)$$

where  $U_a = \min_{\tau_i \in \tau} \{U_i^{max}\}$ .

*Proof:* The details are in the Appendix. ■

**Theorem 5.** *An implicit-deadline multi-mode tasks system  $\tau$  is schedulable under RM scheduling if*

$$U^{sum} \leq \begin{cases} \frac{2(n-1) - \sqrt{2(n-1)(n-2)}}{n} & \text{if } n \geq 3 \quad (15) \\ \frac{1}{2} + \frac{1}{2n} & \text{if } n = 2 \quad (16) \end{cases}$$

*Proof:* The details are in the appendix. ■

**Theorem 6.** *An implicit-deadline multi-mode tasks system  $\tau$  is schedulable under RM scheduling if*

$$U^{sum} \leq 2 - \sqrt{2} \approx 0.5857$$

*Proof:* By taking  $n \rightarrow \infty$  for  $n \geq 3$  in Theorem 5, Theorem 6 follows immediately. ■

It is not difficult to see that QB and the total utilization bound are tight in Theorem 4 and in Theorem 5, respectively. The idea is that increasing the maximum utilization of any higher-priority task mode  $\tau_i$  leads to mode  $\tau_n^h$ 's deadline miss. We illustrate this with the following example.

**Example.** For a given  $n$ , there are two modes in  $\tau_i$  for  $i = 1, 2, \dots, n-1$  with given  $U_i^{max} = \frac{2(n-1) - \sqrt{2(n-1)(n-2)}}{n}$ .  $\frac{1}{n-1}$ , in which  $T_i^2$  is  $T_n^h$ ,  $C_i^2$  is  $T_n^h \cdot U_i^{max}$ ,  $T_i^1$  is  $T_n^h$ .



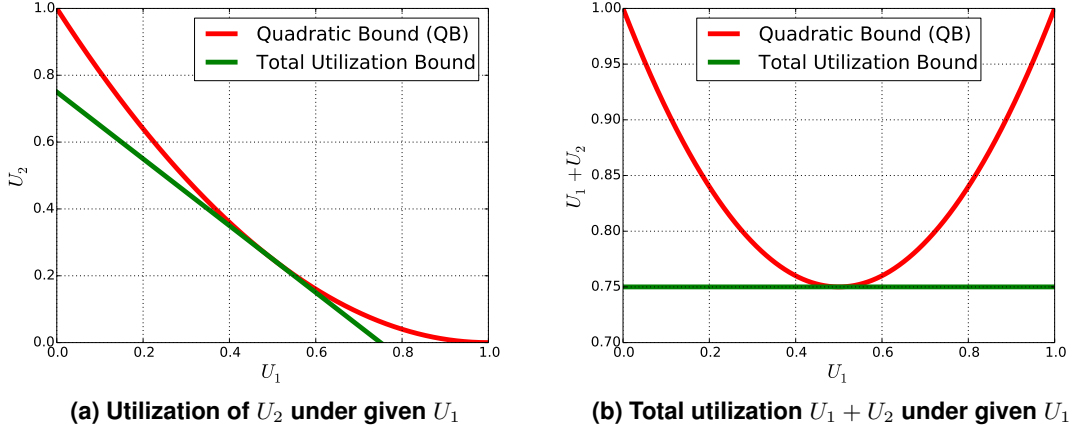


Figure 4: Utilization Bounds for 2 implicit-deadline multi-mode tasks under RM

$(1 - \sum_{j=i}^{n-1} U_j^{max})$ , and  $C_i^1$  is  $T_i^1 \cdot U_i^{max}$ . Thus, task mode  $\tau_n^h$  can only be schedulable when its utilization is 0.

**Comparing the total utilization bounds and quadratic bounds:** Figure 4 illustrates the difference between QB and the total utilization bound in Theorem 4 and in Theorem 5, respectively, for two tasks. We can clearly see that the feasibility region below QB is larger than that below the total utilization bound, in Figure 4a. Moreover, as shown in Figure 4b, the summation of utilizations that can be schedulable under QB is larger, especially when  $U_1$  is away from 0.5.

#### 5.4 Computational Complexity.

Considering all deadlines to be met, QT-RM runs in  $O(n^2 \cdot (M_1 + M_2 + \dots + M_n) + (M_1 + M_2 + \dots + M_n) \log(M_1 + M_2 + \dots + M_n))$  where  $n^2$  comes from the time complexity of the test for each mode and  $(M_1 + M_2 + \dots + M_n) \log(M_1 + M_2 + \dots + M_n)$  is the sorting time for prioritizing the modes at the beginning. Moreover, Theorem 3 can be computed in  $O(n \cdot (M_1 + M_2 + \dots + M_n) + (M_1 + M_2 + \dots + M_n) \cdot \log(M_1 + M_2 + \dots + M_n))$ , and QB-RM requires only  $O(n + (M_1 + M_2 + \dots + M_n))$  where  $(M_1 + M_2 + \dots + M_n)$  accounts for the time of deciding the maximum utilization among modes.

## 6 Evaluations

As can be seen, we have established several utilization-based tests for FPT and RM scheduling. In this section we evaluate the effectiveness of the existing tests and the proposed utilization-based tests in terms of the number of tasksets that are deemed schedulable. First, we recap these tests as follows:

- Demand-based Test under FPT (DT-FPT): the time-demand test approach under FPT presented in Section III.D in [10].
- Quadratic Test under FPT (QT-FPT): Theorem 1.
- Quadratic Bound under RM (QB-RM): Theorem 4.
- Quadratic Test under RM (QT-RM): RM is an example of QT-FPM test of Theorem 2.

The metric to compare the results is to measure the *success ratio* of these tests for a given goal of task set utilization. We generate 100 task sets for each utilization level. The success ratio of a level is said to be the number of task sets that are schedulable divided by the number of task sets for this level.

## 6.1 Task Set Generation

The task set was generated in a similar manner to the method in [10], for testing the variable rate-behavior (VRB) task models, as mentioned in Section 1. We first generated a set of sporadic tasks. The cardinality of the task set was 10. The UUniFast method [5] was adopted to generate a set of utilization values with the given goal. We here use the approach presented by Davis and Burns [11] to generate task periods according to an exponential distribution. Here the two orders of magnitude to control the period values between largest and smallest periods are explored, i.e.,  $1-100ms$ . Task relative deadlines were implicit. The worst-case execution time was computed accordingly, i.e.  $C_i = T_i U_i$ . We converted a proportion  $p$  of tasks to multi-mode tasks:

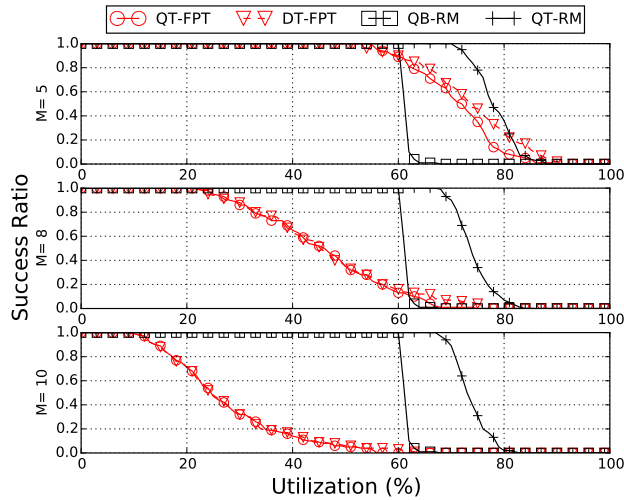
- A multi-mode task has  $M$  execution modes
- The generated sporadic task triplet  $(C_i, T_i, D_i)$  was assigned to the setting of task mode  $\tau_i^1$ .
- We use a scaling factor 1.5 to assign the parameters of the other modes<sup>2</sup>, i.e.,  $C_i^{m+1} = 1.5C_i^m$  and  $T_i^{m+1} = 1.5T_i^m$ .
- We randomly choose a mode to have the largest utilization. The worst-case execution times of the remaining modes were adjusted by multiplying them by uniform random values in the range  $[0.75, 1]$ .

Checking the FPT feasibility of a multi-mode task set was achieved by using the method for sporadic tasks, called *Audsley's Algorithm* [2], since the above tests comply with the required conditions for the compatibility provided in [9]. Due to the page limitation, we report only the results evaluating a variety of  $M$  and  $p$  in the following subsection.

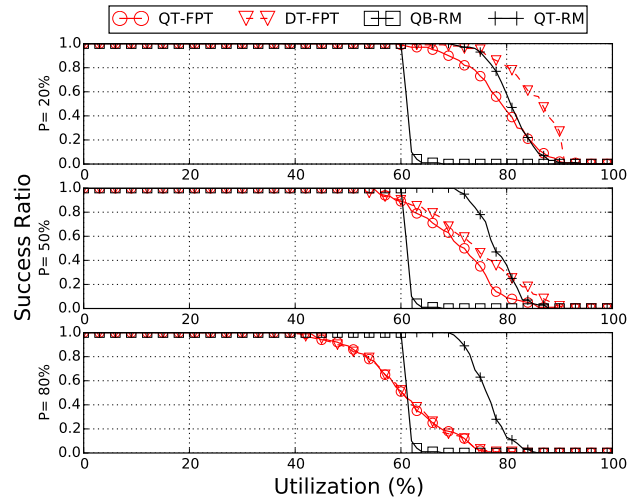
## 6.2 Experimental Results

We first evaluated the impact on different numbers of modes when  $p$  was set to 0.5, as illustrated in Figure 5a. With a less number of modes, the schedulability of multi-mode tasks under FPT scheduling can be provided with up to 90% of utilizations; however, it drops significantly when the number of modes increases. This also accords to the discussion

<sup>2</sup>We use the dynamic programming approach to implement DT-FPT instead of using the ILP solver for the sake of efficiency. To comply with it, we apply a correction factor  $\frac{\lceil T_i \rceil}{T_i}$  on both the generated period and execution time to ensure the discrete time model.



(a) Success ratio with different  $M$  when  $p = 50\%$ .



(b) Success ratio with different  $p$  when  $M = 5$ .

Figure 5: Comparison with different numbers of modes and proportions of multi-mode tasks.

in previous section: the FPT scheduling may perform rather poorly. In fact, under FPT, some jobs with sizable WCET released by different tasks may dominate each other, and this thus results in poor performance of the schedulability. On the other hand, RM scheduling is expected to be more sustainable due to the essential utilization bound guarantee. Moreover, QT-RM can accept the task set with total utilizations of up to 80% compared to only 65% by QB-RM. However, QB-RM takes the advantage on the runtime overhead and can be applied on-line efficiently when long execution times are prohibitive.

For FPT scheduling, the proposed QT-FPT is comparable to the pseudo-polynomial-time DT-FPT. With a number of modes, the workload we overly approximate before the last release may be very close to the actual workload. One can imagine that a task with a larger number of modes is more flexible to generate a worse release sequence for interfering the lower priority one.

We further evaluate different proportions of multi-mode tasks when  $M = 5$ . Similar results are observed in Figure 5b.

## 7 Conclusion

This paper addresses the scheduling problem of mode change real-time tasks under fixed-priority scheduling. Simulation results show that our proposed tests are comparable with the pseudo-polynomial-time demand-based test under FPT. To the best of our knowledge, this is the first work providing the utilization-based test for RM scheduling for multi-mode tasks. Empirical results show that our proposed tests for RM scheduling can accept test sets with utilizations of up to 80%.

Although we focus ourselves on the schedulability tests on multi-mode tasks, the utilization-based tests can also be used to analyze GMF tasks [3], digraph model [22], since the multi-mode task model has complete freedom to change modes, whereas the GMF task model and digraph model have some additional constraints. Our proposed tests in Theorems 5 and 6 for RM scheduling can also be adopted on aperiodic models, as proposed in [1], by relaxing the finite number of modes and are able to accept more tasks during on-line changes.

## References

- [1] T. F. Abdelzaher, V. Sharma, and C. Lu. A utilization bound for aperiodic tasks and priority driven scheduling. *Computers, IEEE Transactions on*, 53(3):334–350, 2004.
- [2] N. C. Audsley. *Optimal priority assignment and feasibility of static priority tasks with arbitrary start times*. Citeseer, 1991.
- [3] S. Baruah, D. Chen, S. Gorinsky, and A. Mok. Generalized multiframe tasks. *Real-Time Systems*, 17(1):5–22, 1999.
- [4] S. K. Baruah, A. K. Mok, and L. E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Real-Time Systems Symposium, 1990. Proceedings., 11th*, pages 182–190. IEEE, 1990.
- [5] E. Bini and G. C. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Systems*, 30(1-2):129–154, 2005.
- [6] A. Biondi, A. Melani, M. Marinoni, M. D. Natale, and G. Buttazzo. Exact interference of adaptive variable-rate tasks under fixed-priority scheduling. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 165–174, 2014.
- [7] G. Buttazzo. Research trends in real-time computing for embedded systems. *ACM SIGBED Review*, 3(3):1–10, 2006.
- [8] G. C. Buttazzo, E. Bini, and D. Buttle. Rate-adaptive tasks: Model, analysis, and design issues. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pages 1–6. IEEE, 2014.
- [9] R. I. Davis and A. Burns. Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. *Real-Time Systems*, 47(1):1–40, 2011.
- [10] R. I. Davis, T. Feld, V. Pollex, and F. Slomka. Schedulability tests for tasks with variable rate-dependent behaviour under fixed priority scheduling. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2014.
- [11] R. I. Davis, A. Zabus, and A. Burns. Efficient exact schedulability tests for fixed priority real-time systems. *Computers, IEEE Transactions on*, 57(9):1261–1276, 2008.
- [12] N. Guan, C. Gu, M. Stigge, Q. Deng, and W. Yi. Approximate response time analysis of real-time task graphs. In *Proceedings of the IEEE 35th IEEE Real-Time Systems Symposium, RTSS*, pages 304–313, 2014.
- [13] J. Kim, K. Lakshmanan, and R. R. Rajkumar. Rhythmic tasks: A new task model with continually varying periods for cyber-physical systems. In *International Conference on Cyber-Physical Systems (ICCPs)*, pages 55–64, 2012.
- [14] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61, 1973.
- [15] S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.
- [16] C. Mercer, R. Rajkumar, and J. Zelenka. Temporal protection in real-time operating systems. In *Real-Time Operating Systems and Software, Proceedings., 11th IEEE Workshop on*, pages 79–83, 1994.
- [17] A. K. Mok. Fundamental design problems of distributed systems for

the hard-real-time environment. 1983.

- [18] J. Real and A. Crespo. Mode change protocols for real-time systems: A survey and a new proposal. *Real-time systems*, 26(2):161–197, 2004.
- [19] L. Santinelli, G. Buttazzo, and E. Bini. Multi-moded resource reservations. In *RTAS*, pages 37–46, 2011.
- [20] A. Schranzhofer, J.-J. Chen, and L. Thiele. Dynamic power-aware mapping of applications onto heterogeneous mpsoe platforms. *IEEE Trans. Industrial Informatics*, 6(4):692–707, 2010.
- [21] L. Sha, R. Rajkumar, J. Lehoczky, and K. Ramamritham. Mode change protocols for priority-driven preemptive scheduling. *Real-Time Systems*, 1(3):243–264, 1989.
- [22] M. Stigge, P. Ekberg, N. Guan, and W. Yi. The digraph real-time task model. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 71–80, 2011.
- [23] M. Stigge and W. Yi. Combinatorial abstraction refinement for feasibility analysis. In *Real-Time Systems Symposium (RTSS)*, pages 340–349, 2013.
- [24] K. W. Tindell, A. Burns, and A. J. Wellings. Mode changes in priority preemptively scheduled systems. In *Real-Time Systems Symposium*, pages 100–109, 1992.

## Appendix

**Proof of Theorem 4.** It is clear that for any task  $\tau_k \in \tau$  if

$$U_k^{max} \leq 1 - 2 \sum_{\tau_i \in \tau \setminus \{\tau_k\}} U_i^{max} + \frac{1}{2} \left( \sum_{\tau_i \in \tau \setminus \{\tau_k\}} U_i^{max} \right)^2 + \frac{1}{2} \sum_{\tau_i \in \tau \setminus \{\tau_k\}} (U_i^{max})^2 \quad (17)$$

then Eq. (13) must hold for all modes  $\tau_k^h \in \tau_k$ . We then show that if Eq. (17) holds by the choice of  $U_k^{max} = \min_{\tau_i \in \tau} \{U_i^{max}\}$ , then it also holds for all the other cases.

Let  $\tau_a$  denote the task with minimum  $U_i^{max}$  among all tasks. Given that Eq. (17) holds for  $\tau_a$ , we have that

$$U_a^{max} \leq 1 - 2 \sum_{\tau_i \in \tau \setminus \{\tau_a\}} U_i^{max} + \frac{1}{2} \left( \sum_{\tau_i \in \tau \setminus \{\tau_a\}} U_i^{max} \right)^2 + \frac{1}{2} \sum_{\tau_i \in \tau \setminus \{\tau_a\}} (U_i^{max})^2$$

We prove this by contradiction: suppose for some  $U_b^{max} \geq U_a^{max}$  Eq. (14) does not hold:

$$U_b^{max} > 1 - 2 \sum_{\tau_i \in \tau \setminus \{\tau_b\}} U_i^{max} + \frac{1}{2} \left( \sum_{\tau_i \in \tau \setminus \{\tau_b\}} U_i^{max} \right)^2 + \frac{1}{2} \sum_{\tau_i \in \tau \setminus \{\tau_b\}} (U_i^{max})^2$$

For notational simplicity, let  $U_a^{max}$  and  $U_b^{max}$  denote  $U_a$  and  $U_b$ . Summing the above two inequities we have

$$\begin{aligned} U_a - U_b &< 2(U_a - U_b) + \frac{1}{2}(U_a^2 - U_b^2) \\ &+ \frac{1}{2} \left( U_a + U_b + 2 \sum_{\tau_i \in \tau \setminus \{\tau_a, \tau_b\}} U_i^{max} \right) (U_b - U_a) \\ &= (U_a - U_b) \left( 1 - \sum_{\tau_i \in \tau \setminus \{\tau_a, \tau_b\}} U_i^{max} \right) \\ &\equiv \sum_{\tau_i \in \tau \setminus \{\tau_a, \tau_b\}} U_i^{max} > 1 \end{aligned}$$

which contradicts the fact that  $U^{sum} \leq 1$  and  $U_a, U_b > 0$ . Hence this theorem is proven.

**Proof of Theorem 5.** Our objective in this proof is to find the minimum  $U^{sum}$  such that Eq. (14) always holds. For  $n = 2$  the minimization can be done directly by solving the differential equation in two variables. We discuss the case for  $n \geq 3$  by using Lagrange Multiplier Method.

$$\begin{aligned} \min \quad & \sum_{i=1}^n U_i \\ \text{s.t.} \quad & U_n = 1 - 2 \sum_{i=1}^{n-1} U_i + \frac{1}{2} \left( \sum_{i=1}^{n-1} U_i \right)^2 + \frac{1}{2} \sum_{i=1}^{n-1} U_i^2 \quad (18a) \\ & U_n \geq 0 \quad (18b) \end{aligned}$$

Let  $\lambda$  be the multiplier of the schedulability constraint by Eq. (18a) and  $\mu$  be the multiplier of the constraint  $U_n \geq 0$ . The Lagrange function is

$$\begin{aligned} L(U_1, U_2, \dots, U_n) &= \sum_{i=1}^n U_i + \mu(-U_n) \\ &+ \lambda \left( 1 - 2 \sum_{i=1}^{n-1} U_i + \frac{1}{2} \left( \sum_{i=1}^{n-1} U_i \right)^2 + \frac{1}{2} \sum_{i=1}^{n-1} U_i^2 - U_n \right) \end{aligned}$$

with derivatives

$$\frac{\partial L}{\partial U_i} = \begin{cases} 1 - \mu - \lambda, & \text{if } i = n \quad (19a) \\ 1 + \lambda \left( -2 + U_i + \sum_{i=1}^{n-1} U_i \right) & \text{otherwise} \quad (19b) \end{cases}$$

A necessary condition for the minimum is that the two derivatives of (19a) and (19b) are zero. Eq. (19b) implies that for all  $i \neq n$

$$U_1 = U_2 = \dots = U_{n-1} \quad (20)$$

To solve these equations, we look at several cases:

**Case 1:**  $\mu = 0$  **Case 2:**  $\lambda = 0$  **Case 3:**  $\mu, \lambda \neq 0$  When  $\mu = 0, \lambda = 1$  by Eq. (19a) according to the necessary condition for the minimum. Setting  $\lambda = 1$  in Eq. (19b) and using the above condition, we obtain optimum values  $U_i \forall 1 \leq i \leq n-1$ :

$$U_i = \frac{n-1}{n} \quad (21)$$

which leads to the violation of the nonnegative constraint  $U_n \geq 0$  after solving the condition of (18a).

**Case 2:**  $\mu \neq 0$

When  $\mu \neq 0$ , it must be the case  $U_n = 0$  due to the complementarity.

After solving Eq. (18a) with the condition (20) and  $U_n = 0$ , we get for all  $i \neq n$

$$U_i = \frac{2(n-1) - \sqrt{2(n-1)(n-2)}}{n} \cdot \frac{1}{n-1} \quad (22)$$

It follows that

$$\sum_{i=1}^n U_i = \frac{2(n-1) - \sqrt{2(n-1)(n-2)}}{n} \quad (23)$$