Fixed-Priority Scheduling of Mixed Soft and Hard Real-Time Tasks on Multiprocessors

Jian-Jia Chen[1], Wen-Hung Huang[1]
[1]TU Dortmund University, Germany

Zheng Dong[2], Cong Liu[2]
[2]The University of Texas at Dallas, USA

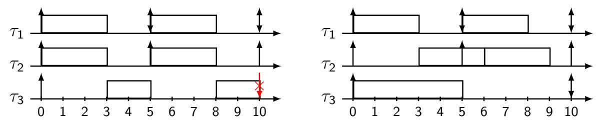
Abstract—[1] **This paper answers several open questions of practical concerns to schedule soft real-time (SRT) tasks, to guarantee their bounded tardiness, under fixed-priority scheduling in homogeneous multiprocessor systems. We consider both cases with only SRT tasks and with mixed sets of SRT and hard real-time (HRT) tasks. For the case in which the system has only SRT tasks, we show that any fixed priority assignment policy yields a capacity augmentation factor of $2 - \frac{1}{M}$ where M is the number of processors. We prove the optimality of the utilization-monotonic (UM) priority assignment (i.e., assigning higher priorities to high-utilization tasks) under our sufficient test for guaranteeing bounded tardiness. We show that UM priority assignment can yield a utilization bound of $\frac{M+1}{2M}$, which is shown asymptotically the best possible bound.**

For the case in which the system has mixed SRT and HRT tasks, we present two new fixed-priority assignment algorithms and their associated schedulability tests. One is a clustering-based greedy priority assignment policy and another is based on Audsley's optimal priority assignment (OPA) approach. We show that the utilization bounds, augmentation factors, and speedup factors are still maintained by the hard real-time cases. Therefore, introducing soft real-time tasks does not create additional problems (at least in those metrics) for scheduling if the priority assignments are properly done. As demonstrated by extensive experiments, these two policies yield reasonably good performance overall and much better performance than the deadline-monotonic priority assignment.

1 Introduction

Motivation. The driving problem that motivates this work is to correctly support a mixed set of soft and/or hard real-time workloads under fixed-priority scheduling. Some applications need hard real-time (HRT) guarantees. Some need only soft real-time (SRT) guarantees, where deadlines can be occasionally missed but any such misses must be provably bounded. We consider a set of sporadic real-time tasks. Each task τ_i has its worst-case execution time C_i and minimal inter-arrival time T_i. An HRT task τ_i is also associated with a relative deadline D_i. A soft real-time task does not have any specified relative deadline, but its response time (or tardiness) should be bounded. We assume global scheduling, in which each job of a task can be executed in any of the available processors. The utilization of task τ_i is C_i/T_i.

An extensive amount of work has been focusing on scheduling HRT tasks and analyzing the corresponding schedulability (see [12]), where tasks must meet their deadlines. More recently, an extensive set of works has been done on supporting soft real-time task systems on a multiprocessor, where tasks' response times must be provably bounded. One seminal work on SRT task scheduling is done by Devi and



(a) Priority assignment under RM (b) A feasible priority assignment

Fig. 1: A motivational example illustrating that neither RM (nor DM) yields feasible priority assignments when both HRT and SRT tasks are present for two processors.

Anderson [15], where it is proved that global earliest-deadline-first (GEDF) scheduling is optimal for scheduling SRT tasks on a multiprocessor, i.e., SRT tasks can be correctly scheduled under GEDF on a multiprocessor without any utilization loss. Unfortunately, there does not exist any efficient schedulability test for SRT task systems under fixed-priority scheduling. Under fixed-priority scheduling, each task is assigned to a priority level, and the priority does not change over time.

Motivated by these, we answer two open questions of practical concerns in this paper: (*i*) how to appropriately assign priority levels to SRT sporadic task systems and what are the corresponding fixed-priority schedulability tests? (*ii*) how to resolve the first problem if there exists a mixed set of HRT and SRT tasks? To answer these questions, an intuitive idea is to apply well-known fixed-priority schemes such as rate-monotonic (RM) or deadline-monotonic (DM) and derive the corresponding schedulability tests. However, we find out that current priority assignment policies may not effectively support a mixed task set with both HRT and SRT constraints. Consider an example task set with three tasks scheduled in a two-processor system, where both tasks τ_1 and τ_2 have an execution cost of 3ms and a period of 5ms, and task τ_3 has an execution cost of 5ms and a period of 10ms. τ_1 and τ_3 are HRT tasks while τ_2 is an SRT task. As seen in Figure 1(a), under RM, the HRT task τ_3 misses its deadline; while Figure 1(b) shows that there exists a feasible priority assignment (where τ_1 and τ_3 are given higher priority) for all tasks to meet their timing constraints (τ_2 has a tardiness bound of 1ms)[2]. This simple example motivates the needs of designing new priority assignment polices as well as deriving the corresponding schedulability tests. (We will provide more general examples in Sec. 3 to further motivate the importance of priority assignments in resolving this problem.)

Related work. Davis and Burns [12] have a very comprehensive survey for scheduling HRT sporadic tasks in multiprocessor platforms. The problem of globally scheduling SRT sporadic task systems on multiprocessors with bounded tardiness has received much attention (e.g., several recent

[1]This paper is supported by DFG, as part of the Collaborative Research Center SFB876 (http://sfb876.tu-dortmund.de/) and NSF grant CNS 1527727.

[2]The example is carefully designed such that the worst-case tardiness under the fixed-priority scheduling is exactly like Figure 1(b) (by testing all combinations with an exhaustive search).

978-1-5386-1898-1/17/$31.00 © 2017 IEEE

dissertations focus on this topic [14], [19]). Existing results with respect to SRT sporadic task systems are only based on global dynamic-priority scheduling, e.g., GEDF. They only focus on SRT task systems by ignoring the coexistence of HRT and SRT tasks in a task set.

However, how to efficiently support an SRT sporadic task system on a multiprocessor under fixed-priority scheduling remains open. Fortunately, in 2015, Huang and Chen [18] provided an efficient response-time analysis (RTA) to calculate WCRTs of sporadic tasks scheduled under fixed-priority on a multiprocessor platform. The analysis in [18] can be used to verify whether the response time (hence, tardiness) is bounded. Although their RTA [18] can be applied to determine the schedulability of a given SRT task system, we still have to answer the two critical questions: (*i*) How to identify a proper priority assignment policy for the SRT case? (*ii*) For different priority assignment policies, what are the corresponding analytical properties?

Moreover, although there exists a set of methods that seek to handle a mixed set of HRT and "best-effort" tasks on either a uniprocessor or a multiprocessor (see Chapter 7 of [22] for an overview on such methods), the problem of supporting a mixed set of HRT tasks and SRT tasks with bounded tardiness remains an open problem.

Contribution. We make the following contributions.

1) We investigate the performance under various priority assignment policies for scheduling SRT sporadic task systems on a multiprocessor and identify a proper priority assignment policy that yields a reasonable utilization bound. Specifically, we prove that the utilization-monotonic priority assignment (i.e., assigning higher priorities to high-utilization tasks) is optimal to ensure bounded tardiness among all priority assignment policies *under the adopted sufficient schedulability tests*. Finally, we show that the utilization-monotonic (UM) priority assignment can yield a utilization bound of $\frac{M+1}{2M}$ for guaranteeing bounded tardiness. We also present an example SRT task set with a utilization bound of $\frac{M+1}{2M} + \epsilon$ that cannot be correctly scheduled under any fixed-priority assignment policy, where ϵ can be arbitrarily small.
2) We present two new fixed-priority assignment algorithms and their associated schedulability tests to schedule a mixed set of HRT and SRT sporadic tasks on a multiprocessor: a clustering-based greedy priority assignment policy and an optimal priority assignment-based (OPA-based) priority assignment policy. As demonstrated by extensive experiments, these two policies yield reasonably good performance overall and much better performance than DM. Also, as demonstrated by experiments, these two policies are in general incomparable but may be preferred under different circumstances where the number of SRT tasks varies.

2 System Models

This section presents the system model used in this paper. We will first present the task and scheduling models, and, then, formally define the *schedulability* of a task set, consisting of mixed SRT and HRT tasks. At the end of the section, we will define the capacity augmentation factors and speedup factors to be used for quantifying the imperfectness of the schedulability tests and the scheduling policies.

2.1 Task and Scheduling Model

We consider n independent sporadic real-time tasks $\mathbf{T} = \{\tau_1, \tau_2, \ldots, \tau_n\}$ upon a system comprised of M identical processors, where $M \geq 2$ is an integer. Each sporadic task τ_i can release an infinite number of jobs (also called task instances) under the given minimum inter-arrival time (temporal) constraint T_i (also called period). That is, if a job of task τ_i arrives at time θ_a, the next instance of the task must arrive no earlier than θ_a plus the minimum inter-arrival time, i.e., $\theta_a + T_i$. For each sporadic task τ_i, the *worst-case execution time* (WCET) C_i of task τ_i is also specified. The WCET can be obtained by using static timing analysis. We assume that all task parameters are positive integers. Since all the M processors are identical, the execution time of a job of a task does not depend on the processors that execute the job.

We assume that the system is fully *preemptive*, and allows *global* inter-processor migration. The cost of preemption and migration has been subsumed into the worst-case execution time of each task. At any given time, a job is executed on at most one processor. Moreover, we do not allow intra-task parallelism. That is, a job of task τ_i cannot start its execution until all the jobs of task τ_i that arrive earlier are completed. Therefore, at any time point, the jobs of the (at most) M-highest priority tasks in the ready queue are executed on the processors. Due to the disallowance of intra-task parallelism, at some time point, we may have more than M jobs available to be executed, but less than M jobs are executed.

2.2 Definitions and Schedulability Guarantees

Throughout this paper, we refer to the *utilization* of task τ_i as $U_i = C_i/T_i$. We further assume $\sum_{i=1}^{n} U_i \leq M$ and $\max_i U_i \leq 1$. We denote $hp(\tau_i)$ the set of the tasks with higher priority than task τ_i. For the simplicity of presentation, we define the following terms:

- Worst-case response time R_k of an HRT or SRT task τ_k: An upper bound of the response times of all the jobs of task $\tau_k \in \mathbf{T}$ for any *legal sequence* of the jobs of $\mathbf{T}$. We call a sequence of jobs of the task system $\mathbf{T}$ a legal sequence if any two consecutive jobs of task $\tau_i \in \mathbf{T}$ are separated by *at least* T_i.
- Tardiness bound Δ_ℓ of an SRT task τ_ℓ: the worst-case response time of task τ_ℓ minus the period of task τ_ℓ if R_ℓ is larger than T_i, i.e., $\Delta_\ell = \max\{0, R_\ell - T_\ell\}$.

We consider systems in which both SRT and HRT tasks co-exist in the system. For the rest of the paper, we define $\mathbf{T}_{soft}$ and $\mathbf{T}_{hard}$ as the subsets of the SRT and HRT tasks in $\mathbf{T}$, respectively. The cardinality of a set $\mathbf{X}$ is $|\mathbf{X}|$. For notational brevity, n_{soft} and n_{hard} are the cardinality of $\mathbf{T}_{soft}$ and $\mathbf{T}_{hard}$, respectively.

For an HRT task $\tau_k \in \mathbf{T}_{hard}$, it is also specified with a relative deadline constraint D_k. Therefore, an HRT task τ_k imposes a hard constraint: $R_k \leq D_k$. We assume that $C_k \leq D_k$ for an HRT task $\tau_k \in \mathbf{T}_{hard}$; otherwise, the task cannot be feasibly scheduled anyway. If the relative deadline D_k of task τ_k in $\mathbf{T}_{hard}$ is always equal to the period T_k, such a task set $\mathbf{T}_{hard}$ is called *implicit-deadline* HRT task set. If the relative deadline D_k of task τ_k in $\mathbf{T}_{hard}$ is always no more than the period T_k, such a task set $\mathbf{T}_{hard}$ is called *constrained-deadline* HRT task set. Otherwise, such a task set is called *arbitrary-deadline* HRT task set.

If a task is an SRT task in $\mathbf{T}_{soft}$, there is no specific relative deadline constraint[3], but the tardiness has to be bounded. That is, we have to ensure that the maximum tardiness (or equivalently the worst-case response time) of soft real-time task $\tau_\ell \in \mathbf{T}_{soft}$ is not infinity. We will implicitly look at the bounded response time, which implies the bounded tardiness, of task τ_ℓ for the rest of the paper.

We consider fixed-priority multiprocessor scheduling. Each task is assigned to a *static* priority level. At run-time, the jobs of the M-highest priority tasks in the ready queue are executed on the M processors. Note that a job of task τ_i cannot start until all the jobs of task τ_i that arrive earlier are completed. There are several simple priority assignment algorithms. For example, the global rate-monotonic (RM) priority assignment gives a task higher priority if its period is shorter. Moreover, the global deadline-monotonic (DM) priority assignment gives a task higher priority if its relative deadline is shorter. (Ties can be arbitrarily broken in both cases.)

We define the schedulability of the task system as follows:

Definition 1 (Feasibility)**.** *Given two sets $\mathbf{T}_{soft}$ and $\mathbf{T}_{hard}$ of sporadic real-time tasks, a schedule is* ***feasible*** *if $R_k \leq D_k$ for each task $\tau_k \in \mathbf{T}_{hard}$ and $\Delta_\ell < \infty$ for each task $\tau_\ell \in \mathbf{T}_{soft}$.*

Definition 2. *Given two sets $\mathbf{T}_{soft}$ and $\mathbf{T}_{hard}$ of sporadic real-time tasks, a schedulability test of a scheduling algorithm is a test to verify whether its resulting schedule is always feasible for the tasks in $\mathbf{T}_{soft}$ and $\mathbf{T}_{hard}$.*

Our objective is to answer the open question: how to feasibly schedule HRT and SRT tasks under fixed-priority multiprocessor scheduling, to ensure the hard real-time guarantees for $\mathbf{T}_{hard}$ and bounded tardiness for $\mathbf{T}_{soft}$. A necessary condition for the feasibility of any scheduling policy is

- $U_i \leq 1$ for every task $\tau_i \in \mathbf{T}_{soft} \cup \mathbf{T}_{hard}$,
- $C_i \leq D_i$ for every task $\tau_i \in \mathbf{T}_{hard}$, and
- $\sum_{i=1}^{n} U_i \leq M$.

2.3 Different Bounds and Factors

It has been shown that fixed priority scheduling is non-optimal for scheduling hard real time systems in uniprocessor systems [21] and multiprocessor systems [16]. Throughout this paper, we will quantify the error of the schedulability tests and the scheduling policies by providing their utilization bound, resource augmentation by using speed-up factors [23] and the capacity augmentation factors [20].

The utilization bound U^* of a scheduling algorithm provides a simple schedulability test. That is, if $\sum_{i=1}^{n} U_i \leq MU^*$, then the task set can be feasibly scheduled by the scheduling algorithm. For implicit-deadline HRT task systems on uniprocessor (i.e., $M = 1$), Liu and Layland [21] provide a utilization upper bound $\ln 2 \approx 0.693$ under rate-monotonic (RM) scheduling. Multiprocessor global RM for HRT tasks cannot admit a utilization bound better than $U^* = \frac{1}{M} + \epsilon$, for arbitrarily small $\epsilon > 0$, due to "Dhall's effect" [16]. The best utilization bound so far for global fixed-priority scheduling is at most $U^* = 38\%$ [1]. However, the utilization bound to guarantee bounded tardiness for SRT tasks under global fixed-priority multiprocessor scheduling remains open.

If we constrain the total utilization $\sum_{\tau_i} \frac{C_i}{MT_i} \leq \frac{1}{b}$ and the maximum utilization $\max_{\tau_i} U_i \leq \frac{1}{b}$, it is possible to provide the schedulability guarantee of global RM by setting b to $3 - \frac{1}{M}$ [2], [4], [8]. Such a factor b has been recently named as a *capacity augmentation factor* [20]. A capacity augmentation factor provides a balanced factor between the maximum utilization among the given tasks and the total utilization of the tasks.

An algorithm $\mathcal{A}$ is with speed-up factor b (e.g., [23]): *If there exists a feasible schedule for the task system, it is schedulable by algorithm $\mathcal{A}$ by speeding up (each processor) to b times as fast as in the original platform (speed).* A sufficient schedulability test for scheduling algorithm $\mathcal{A}$ is with speed-up factor b: *If the task system cannot pass the sufficient schedulability test, the task set is not schedulable by any scheduling algorithm if (each processor) is slowed down to $\frac{1}{b}$ times of the original platform speed.*

In the above three definitions, a utilization bound U^* implies a capacity augmentation factor $\frac{1}{U^*}$, and a capacity augmentation factor b implies a speedup factor b as well. Therefore, it is obvious that the utilization bound is weaker than the capacity augmentation factor, and the capacity augmentation factor is weaker than the speed-up factor.

3 Motivational Examples

This section presents the motivational examples to explain why proper priority assignments are important. With an improper priority assignment, like global rate-monotonic scheduling, the utilization bound can be very low. One example is as follows:

Example 1. *We are given $n = M + 1$ tasks on $M \geq 2$ processors: M tasks are* light *with execution time ϵ and period 1 and* one *task is* heavy *with execution time $1 + \delta$ and period $1 + \delta$, where $0 < \delta < \epsilon \ll 1$.*[4] *If the heavy task is the lowest priority task, its tardiness goes to ∞ in the worst cases, as illustrated in Figure 2.*

This shows that global rate-monotonic scheduling is not a very good priority assignment strategy. However, this does not rule out the possibility to use global fixed-priority scheduling. The above example was indeed the example provided by Dhall and Liu [16] to demonstrate that global EDF (also for global RM) scheduling has very low utilization bound for HRT tasks. Here, such an argument still holds for soft real-time tasks under fixed-priority scheduling. That is, an improper priority assignment may result in a low utilization bound. (Note that the above example does not have unbounded tardiness under global EDF.)

In many systems, HRT tasks are placed greedily with higher priority levels than any SRT tasks. However, the following example shows that such a design is also problematic.

Example 2. *Consider the same example as used in Example 1. The M light tasks are HRT tasks and the heavy task is an SRT task. Therefore, assigning the SRT task as the lowest priority task leads to unbounded tardiness for the SRT task, as also explained in Example 1.*

In the above two examples, if the heavy task is not the lowest-priority task, we can use Theorem 1 (introduced later) to verify the feasibility of the resulting schedules.

[3]This can be also relaxed to allow different settings of the relative deadlines, but this does not change our solutions and conclusions in this paper.

[4]We use fractional numbers here for being aligned with the well-known Dhall's effect. The same effect can be created by using only integer parameters.

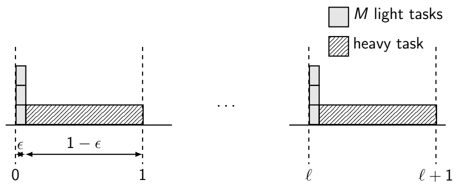


Fig. 2: The unbounded tardiness of the heavy real-time task assigned to the lowest priority. By releasing all the jobs periodically, at time ℓ, the workload left from the first ℓ jobs of the heavy task is $\epsilon \cdot \ell$.

To get rid of the Dhall's effect (in the case of purely HRT task systems) for the troublesome case in Example 1, Andersson et al. [1], [2] propose to use RM-US[ς] and SM-US[ς] algorithms, which give the highest priority levels to the tasks τ_i with $U_i > \varsigma$. The priority assignment of the remaining tasks with $U_i \leq \varsigma$ are then based on rate-monotonic scheduling for RM-US[ς] and slack-monotonic scheduling for SM-US[ς]. For RM-US, ς is $\frac{M}{3M-2} \approx 0.33$ when M is sufficiently large, and for SM-US, ς is $\frac{2}{3+\sqrt{5}} \approx 0.38$. Andersson et al. [1], [2] show that the utilization bound U^* is ς in these two algorithms.

Using RM-US or SM-US can improve the utilization bound to 33% and 38% for the input instances in Examples 1 and 2. However, that also implies that the worst-case response time of the SRT tasks are bounded by its period, which may be not necessary. Moreover, these approaches cannot be applied for constrained-deadline and arbitrary-deadline task systems in $\mathbf{T}_{hard}$. Our solutions are more general and applicable for task systems in which certain tasks only need bounded tardiness and the tasks in $\mathbf{T}_{hard}$ can be arbitrary-deadline tasks.

4 Response-Time Analysis

This section summarizes the state of the art in the worst-case response time analysis for global fixed-priority scheduling. The following two theorems are mainly from Huang and Chen [18]. For the simplicity of presentation, we will mainly use the polynomial-time worst-case response time analysis in Theorem 1, but most of the arguments in this paper can still hold with the schedulability test in Theorem 2.

Theorem 1 (revised Theorem 2 in [18])**.** *Suppose that the priority assignment is given. The worst-case response time R_k of task τ_k in the global fixed-priority scheduling is at most*

$$R_k \leq \begin{cases} C_k & \text{if } |hp(\tau_k)| < M, \\ R_k^\dagger & \text{else if } MU_k + \sum_{\tau_i \in hp(\tau_k)} U_i < M, \\ \infty & \text{otherwise,} \end{cases} \tag{1}$$

where $R_k^\dagger = \frac{MC_k + Z_\Sigma + \sum_{\tau_i \in hp(\tau_k)} C_i(1-U_i)}{M - \sum_{\tau_i \in hp(\tau_k)} U_i}$ and Z_Σ denotes the sum of the $(M-1)$ largest $R_i \cdot U_i$'s among the tasks in $hp(\tau_k)$.[5]

A simple corollary of Theorem 1 to test whether an SRT task has bounded tardiness is as follows:

Corollary 1. *Task τ_k has bounded tardiness (response time) if (1) $|hp(\tau_k)| < M$, or (2) $MU_k + \sum_{\tau_i \in hp(\tau_k)} U_i < M$ and all the higher-priority tasks in $hp(\tau_k)$ are with bounded response times.*

[5]The definition of $R_k^\dagger$ is slightly different here, since we need to use the $(M-1)$ largest $R_i \cdot U_i$ instead of the $(M-1)$ largest $D_i \cdot U_i$ in [18].

Therefore, for a given priority assignment, we can use Theorem 1 and Corollary 1 to test whether an HRT task τ_k meets its deadline and an SRT task τ_k has bounded tardiness or not, respectively.

Let $\Omega_k(t)$ be the maximum interference from the higher priority tasks in $hp(\tau_k)$ with an interval length t. How to calculate $\Omega_k(t)$ can be found in [18].[6] Here, we only give the final results. For any interval length t, the workload function $W_i(t)$ of a sporadic task τ_i bounds the maximum cumulative execution requirement by jobs of τ_i that are released and may execute within any interval of length t.

$$W_i(t) = \left\lfloor \frac{t}{T_i} \right\rfloor C_i + min(t \bmod T_i, C_i) \tag{2}$$

For a given positive integer h and task τ_k, let $R_{k,h}$ be the smallest t satisfying the following inequality:

$$\Omega_k(t) \leq m \times (t - hC_k). \tag{3}$$

The concept of the analysis in [18] is to classify the higher-priority tasks with two different functions $I_i^1(t)$ and $I_i^2(t)$ for an interval length t, in which

$$I_i^1(t) = min\left(W_i(t), max(0, t - hC_k + 1)\right) \tag{4}$$
$$I_i^2(t) = min\left(W_i(R_i + t), max(0, t - hC_k + 1)\right). \tag{5}$$

It is proved in [18] that we only need to consider at most $M-1$ tasks with $I_i^2(t)$, for any given $t > 0$. Let $I_i^{DIFF}(t) = I_i^2(t) - I_i^1(t)$. Therefore, by [18],

$$\Omega_k(t) = \sum_{\tau_i \in hp(k)} I_i^1(t) + \sum_{\substack{\text{the } (M-1) \text{ largest} \\ \tau_i \in hp(k)}} I_i^{DIFF}(t) \tag{6}$$

Distinct from the uniprocessor response time analysis, $\Omega_k(t)$ is a function of the worst-case response time R_i of a higher-priority task $\tau_i \in hp(k)$. The worst-case response time R_k of task τ_k with $|hp(\tau_k)| \geq M$ is at most:

$$R_k \leq R_k^* = max_{h \in \{1,\ldots,H\}}\{R_{k,h} - (h-1)T_k\}, \tag{7}$$

where $H = min\{h \geq 1 | \frac{\Omega_k(hT_i)}{m} + hC_k \leq hT_k\}$.[7] The following theorem is the schedulability test from [18].

Theorem 2 (**revised** Theorem 1 in [18])**.** *Suppose that the priority assignment is given. The worst-case response time R_k of task τ_k in the global fixed-priority scheduling is at most*

$$R_k \leq \begin{cases} C_k & \text{if } |hp(\tau_k)| < M, \\ R_k^* & \text{else if } MU_k + \sum_{\tau_i \in hp(\tau_k)} U_i < M, \\ \infty & \text{otherwise,} \end{cases} \tag{8}$$

where R_k^* is derived from Eq. (7).

[6]To formally summarize the equations precisely, it would take roughly two pages that serve only little information related to our methods. We decide to only shortly present the equations to derive $\Omega_k(t)$ here for the simplicity of the presentation flow.

[7]The least common multiple of the periods of the tasks in $hp(\tau_k)$ and T_k can be very large, and H can become infinite. More discussions can be found in Sections 4.2 and 4.3 in [18].

5 Soft Real-Time Guarantees

Throughout this section, we consider that the system only has soft real-time tasks, i.e., $\mathbf{T}_{hard}$ is empty. One essential question is to define the *priority levels* of the tasks. Although Example 1 provides some bad news, if we further constrain ourselves to disallow high-utilization tasks, in fact, we can obtain a capacity augmentation factor (asymptotically) $2 - \frac{1}{M}$ for any arbitrary fixed-priority assignment for soft real-time tardiness guarantees.

Theorem 3. *Any arbitrary priority assignment has a capacity augmentation factor $2 - \frac{1}{M} + \epsilon$ for ensuring the bounded tardiness of soft real-time tasks in $\mathbf{T}_{soft}$, in which ϵ is an extremely small positive number.*

Proof: By the argument, we suppose that $\sum_{i=1}^{n} \frac{C_i}{MT_i} \leq \frac{1}{2-\frac{1}{M}+\epsilon}$ and $U_k \leq \frac{1}{2-\frac{1}{M}+\epsilon}$ for $k = 1, 2, \ldots, n$. Therefore, for a task τ_k, we have

$$MU_k + \sum_{\tau_i \in hp(\tau_k)} U_i = (M-1)U_k + \sum_{\tau_i \in hp(\tau_k)\cup\{\tau_k\}} U_i < \frac{M-1}{2-\frac{1}{M}} + \frac{M}{2-\frac{1}{M}} = M.$$

As a result, by Theorem 1, we know that task τ_k has bounded tardiness for $k = 1, 2, \ldots, n$. ■

The above theorem shows that capacity augmentation (as well as speedup) factors are <u>**not**</u> meaningful for quantifying the sub-optimality of global fixed-priority scheduling for SRT tasks. Instead, here, we will analyze the utilization bounds.

5.1 Utilization Monotonic (UM)

Corollary 1 provides a sufficient condition for ensuring the bounded tardiness. We know that the M highest priority tasks will automatically have bounded delay equal to the worst-case execution time due to Corollary 1 under the assumption that $U_i \leq 1$ for every task τ_i in $\mathbf{T}_{soft}$. Moreover, it can be observed that putting a high-utilization task τ_k to be a lower-priority task implies that the condition $MU_k + \sum_{\tau_i \in hp(\tau_k)} U_i < M$ in Corollary 1 is more difficult to achieve.

The above discussions actually lead to the following simple priority assignment: *utilization monotonic* (UM) by assigning the higher-utilization task as a higher-priority task, in which ties are broken arbitrarily. (One can also break the ties by using the rate-monotonic priority assignment if two tasks are with the same utilization.) The utilization monotonic assignment actually solves the issues in Example 1.

For the rest of this section, for notational brevity, we index the $n = n_{soft}$ tasks in $\mathbf{T}_{soft}$ from 1 to n, in which $U_1 \geq U_2 \geq \cdots \geq U_n$. With the above algorithm, we have the following theorems:

Theorem 4. *Utilization monotonic priority assignment is optimal to ensure bounded tardiness by using the SRT schedulability test in Corollary 1.*

Proof: This can be proved by swapping. Suppose that there are two tasks, (1) one is assigned to the priority level k and one to the priority level $k+1$, and (2) the task in priority level k is with smaller utilization. For notational brevity, let these two tasks be indexed as τ_k and τ_{k+1}. That is, $U_k < U_{k+1}$ and both tasks τ_k and τ_{k+1} are with bounded tardiness by using the schedulability test in Corollary 1. We can easily prove that swapping their priority levels does not affect the *schedulability*. It is clear that swapping the priority level of τ_k and τ_{k+1} has no impact on the tasks whose priority levels are higher than τ_k.

If $k < M$, the above swapping remains a feasible priority assignment trivially. We focus on the other case with $k \geq M$. Task τ_{k+1} is still with bounded tardiness *after the swapping of the priority levels*, i.e., Δ_{k+1} is bounded and R_{k+1} is bounded. Moreover, since the schedulability test in Corollary 1 is adopted, the bounded tardiness assumption *before swapping* implies that $MU_{k+1} + \sum_{\tau_i \in hp(\tau_{k+1})} U_i < M$ and $R_j \neq \infty$ for every task $\tau_j \in hp(\tau_{k+1})$. By the assumption $U_{k+1} > U_k$, we have $MU_k + \sum_{\tau_i \in hp(\tau_{k+1})} U_i < M$, which implies that the worst-case response time of task τ_k after swapping is bounded since the higher-priority tasks are with bounded tardiness.

According to the condition in Corollary 1, this swapping has no impact regarding to the bounded tardiness of any other lower-priority tasks. Therefore, we can keep swapping two consecutive priority levels. This means that the utilization-monotonic priority assignment is optimal when we use the SRT schedulability test in Corollary 1. ■

Theorem 5. *All the tasks in the task set $\mathbf{T}_{soft}$ are with bounded tardiness guarantees (soft real-time schedulable) under utilization-monotonic (UM) fixed-priority multiprocessor scheduling if $U_i \leq 1$ for every task $\tau_i \in \mathbf{T}_{soft}$ and*

$$\sum_{\tau_i \in \mathbf{T}_{soft}} U_i < \frac{M+1}{2}. \tag{9}$$

Proof: We index the $n = n_{soft}$ tasks in $\mathbf{T}_{soft}$ from 1 to n, in which $U_1 \geq U_2 \geq \cdots \geq U_n$ According to Corollary 1, task τ_k has bounded response time if $k \leq M$. We focus on the case when $|\mathbf{T}_{soft}| \geq k \geq M+1$. Therefore, we have

$$MU_k + \sum_{i=1}^{k-1} U_i = (M-1)U_k + \sum_{i=1}^{k} U_i$$
$$\leq (M-1)\frac{\sum_{i=1}^{k} U_i}{k} + \sum_{i=1}^{k} U_i \leq \left(\frac{M-1}{M+1} + 1\right)\sum_{i=1}^{k} U_i$$
$$\leq \left(\frac{2M}{M+1}\right) \sum_{\tau_i \in \mathbf{T}_{soft}} U_i < M,$$

where the first inequality comes from ordering of the algorithm with $U_1 \geq U_2 \geq \cdots \geq U_k$, the second inequality comes from the case that $k \geq M+1$, the third inequality comes from the fact that $k \leq |\mathbf{T}_{soft}|$ and the last inequality is due to Eq. (9). By Corollary 1, we reach the conclusion. ■

Note that the above theorems only provide the *schedulability test* for SRT tasks under a global multiprocessor fixed-priority scheduling. Once the bounded tardiness is ensured, if the worst-case tardiness is required, we can either use Theorem 1 or Theorem 2 to calculate the worst-case response time (or worst-case tardiness) accordingly. Moreover, the optimality of UM in Theorem 4 is only based on the adopted schedulability test in Corollary 1. To the best of our knowledge, the test in Corollary 1 is the only test known for bounded tardiness in fixed-priority scheduling.

5.2 Tightness

We conclude this section by showing that the utilization bound in Theorem 5 is asymptotically tight.

Theorem 6. *There exists an input instance with $n = M + 1$ tasks in $\mathbf{T}_{soft}$, in which $\sum_{\tau_i \in \mathbf{T}_{soft}} U_i = \frac{M+1}{2} + \epsilon$, and the task set does not admit any fixed-priority assignment with bounded tardiness, for $M \geq 2$ and arbitrarily small ϵ.*

Proof: We prove this theorem by providing a concrete example. For notational brevity, we use fractional parameters instead of integer parameters. Each of the $M+1$ tasks has the same period 1, execution time $\frac{1}{2} + \frac{\epsilon}{M+1}$. Therefore, this input instance has $\sum_{\tau_i \in \mathbf{T}_{soft}} U_i = \frac{M+1}{2} + \epsilon$. For notational brevity, let σ be $\frac{\epsilon}{M+1}$. Since all the $M+1$ tasks are identical, there is only one fixed-priority assignment.

Let all the $M+1$ tasks release their first jobs at time 0 and the subsequent jobs periodically. At time ℓ, the $(\ell+1)$-th job of the lowest-priority task arrives. According to global fixed-priority scheduling, the *residual workload* (or backlog) of the lowest-priority task at time ℓ is $(\frac{1}{2}+\sigma)\cdot\ell - (1-\frac{1}{2}-\sigma)\cdot\ell = 2\sigma\ell$. (That is, in the time interval $(0, \ell]$, the M higher priority tasks will use $(\frac{1}{2}+\sigma)\cdot\ell$ amount of time and the lowest priority task only runs for $(1-\frac{1}{2}-\sigma)\cdot\ell$ amount of time.)

Therefore, the $(\ell+1)$-th job of the lowest-priority task has to wait until the residual workload has been finished. As a result, the response time of the $(\ell+1)$-th job of the lowest-priority task *is at least* $\frac{1}{2}+\sigma+2\sigma\ell$, which is unbounded when ℓ is sufficiently large.[8] ∎

6 Mixture of HRT and SRT Tasks

In this section, we consider that $\mathbf{T}_{soft}$ and $\mathbf{T}_{hard}$ are not empty. We will present how to assign the priority levels of each task in these two sets. We will present how to assign the priority levels as follows:

- Section 6.1 will design a simple greedy strategy to assign the priority levels with three clusters. We will assume that the HRT tasks in $\mathbf{T}_{hard}$ are implicit-deadline tasks.
- Section 6.2 will explain how to extend the above greedy strategy to handle arbitrary-deadline and constrained-deadline task systems in $\mathbf{T}_{hard}$.
- Section 6.3 will explain how to adopt Audsley's priority assignment strategy [3] to *potentially* improve the greedy approach. However, to make the schedulability test compatible with Audsley's priority assignment strategy, some pessimism is introduced.

6.1 Greedy Assignment with Three Clusters

In this subsection, we will consider a simpler case: the task set $\mathbf{T}_{hard}$ is an *implicit-deadline* task set. We propose to use the following simple priority assignment motivated by the above discussions:

- $\mathbf{T}_{high}$: If the utilization U_i of task $\tau_i \in \mathbf{T}_{soft} \cup \mathbf{T}_{hard}$ is larger than or equal to 0.5, we greedily put this task to $\mathbf{T}_{high}$. These tasks in $\mathbf{T}_{high}$ are prioritized by using the *utilization-monotonic* priority assignment.
- $\mathbf{T}_{middle}$: This task set consists of the residual tasks of the HRT tasks that are not in $\mathbf{T}_{high}$, i.e., $\mathbf{T}_{middle} = \mathbf{T}_{hard} \setminus \mathbf{T}_{high}$. These tasks in $\mathbf{T}_{middle}$ are prioritized by using any HRT task priority ordering algorithm, that has been proposed in the literature, e.g., RM-US [2], [5], [8], OPA [11], SM-US [1].
- $\mathbf{T}_{low}$: This task set consists of the residual tasks of the SRT tasks that are not in $\mathbf{T}_{high}$, i.e., $\mathbf{T}_{low} = \mathbf{T}_{soft} \setminus \mathbf{T}_{high}$. These tasks in $\mathbf{T}_{low}$ are prioritized by the utilization-monotonic priority assignment.

The tasks in $\mathbf{T}_{high}$ are assigned to higher priority levels than the tasks in $\mathbf{T}_{middle}$, and the tasks in $\mathbf{T}_{middle}$ are assigned to higher priority levels than $\mathbf{T}_{low}$.

The above priority assignment is referred to as the greedy *three-cluster priority assignment* for the rest of the paper. We can now examine the feasibility of the resulting fixed-priority multiprocessor schedules. We provide the following three lemmas individually for verifying the feasibility of the tasks in each of the three priority clusters.

Lemma 1. *If a task τ_k in $\mathbf{T}_{high}$ cannot be feasibly scheduled (to meet the hard deadline if τ_k is from $\mathbf{T}_{hard}$ or to have bounded tardiness if τ_k is from $\mathbf{T}_{soft}$) in the greedy three-cluster priority assignment, then $|\mathbf{T}_{high}| \geq M+1$, i.e., $\sum_{\tau_i \in \mathbf{T}_{high}} U_i \geq \frac{M+1}{2}$.*

Proof: This is by the definition of the algorithm. ∎

Lemma 2. *Suppose that all the tasks in $\mathbf{T}_{high}$ are feasibly scheduled. Moreover, suppose that the priority assignment algorithm for assigning the priority levels of $\mathbf{T}_{middle}$ has a utilization bound ς, in which $\varsigma \leq 0.5$. If a task τ_k in $\mathbf{T}_{middle}$ cannot be feasibly scheduled to meet the hard deadline in the greedy three-cluster priority assignment, then $\sum_{\tau_i \in \mathbf{T}_{high} \cup \mathbf{T}_{middle}} U_i > \varsigma \cdot M$.*

Proof: We only sketch the proof since the concept is very similar to the original proofs of RM-US[ς] [2], [5], [8]. The impact of the soft real-time tasks in $\mathbf{T}_{high}$ can be imagined as if they are promoted to ensure the hard real-time deadline set to $T_i \geq C_i$ for each task $\tau_i \in \mathbf{T}_{high}$. Due to the tasks in $\mathbf{T}_{high}$, we can imagine that there are only $M - |\mathbf{T}_{high}|$ processors available for the tasks in $\mathbf{T}_{middle}$. By the clustering algorithm, we have $\sum_{\tau_i \in \mathbf{T}_{high}} U_i \geq \frac{1}{2}|\mathbf{T}_{high}|$. By the utilization bound of the priority assignment algorithm and the failure of task τ_k, we have $\sum_{\tau_i \in \mathbf{T}_{middle}} U_i > \varsigma \cdot (M - |\mathbf{T}_{high}|)$. Since $\varsigma \leq 0.5$ and $\{\tau_k\} \subseteq \mathbf{T}_{middle} \neq \emptyset$, we conclude that $\sum_{\tau_i \in \mathbf{T}_{high} \cup \mathbf{T}_{middle}} U_i > \frac{1}{2}|\mathbf{T}_{high}| + \varsigma \cdot (M - |\mathbf{T}_{high}|) \geq \varsigma \cdot M$. ∎

Lemma 3. *Suppose that all the tasks in $\mathbf{T}_{high} \cup \mathbf{T}_{middle}$ are feasibly scheduled. If a task τ_k in $\mathbf{T}_{low}$ cannot be feasibly scheduled to have bounded tardiness in the greedy three-cluster priority assignment (by using the response time analysis in Theorem 1 or Theorem 2 or Corollary 1), then $\sum_{\tau_i \in \mathbf{T}_{soft} \cup \mathbf{T}_{hard}} U_i \geq \frac{M+1}{2}$.*

Proof: By the assumption that τ_k is in $\mathbf{T}_{low}$, we know $U_k < 0.5$. By Corollary 1, task τ_k has unbounded tardiness only when $MU_k + \sum_{\tau_i \in hp(\tau_k)} U_i \geq M$. Therefore, $\sum_{\tau_i \in \mathbf{T}_{soft} \cup \mathbf{T}_{hard}} U_i \geq U_k + \sum_{\tau_i \in hp(\tau_k)} U_i \geq \frac{M+1}{2}$. ∎

Now, we can conclude the theorem.

Theorem 7. *If*

$$\sum_{\tau_i \in \mathbf{T}_{high}} U_i < \frac{M+1}{2}, \textit{ and } \sum_{\tau_i \in \mathbf{T}_{high} \cup \mathbf{T}_{middle}} U_i \leq \varsigma \cdot M,$$

$$\textit{and } \sum_{\tau_i \in \mathbf{T}_{soft} \cup \mathbf{T}_{hard}} U_i < \frac{M+1}{2},$$

[8]We do not calculate the actual response time here, since such a value is not necessary to explain the consequence of unbounded tardiness.

then the greedy three-cluster priority assignment provides a feasible schedule for $\mathbf{T}_{hard}$ and $\mathbf{T}_{soft}$, where $\varsigma \leq 0.5$ is defined according to the scheduling policy (e.g., RM-US[ς] or SM-US[ς]) in $\mathbf{T}_{middle}$.

Proof: Due to Lemmas 1, 3, and 2 by contrapositive. ■

Note that the condition in Theorem 7 can be further generalized by testing the schedulability by using Theorems 1 and 2 and Corollary 1. We can conclude the discussion of this algorithm section by the following corollary.

Corollary 2. *The speedup factor and the capacity augmentation factor of the greedy three-cluster priority assignment algorithm are both $\frac{1}{\varsigma}$, which is defined according to the scheduling policy in $\mathbf{T}_{middle}$, when $\varsigma \leq 0.5$.*

6.2 Constrained and Arbitrary Deadlines

This subsection further presents a generalization of the priority assignment in Section 6.1. For constrained- and arbitrary-deadline tasks in $\mathbf{T}_{middle}$, a simple strategy is to use global deadline monotonic (DM) priority assignment and use Theorems 1 or 2 to analyze the feasibility. Note that, there have been several results in the literature to verify the feasibility of deadline-monotonic scheduling, including [5], [6], [10], [18] for arbitrary-deadline cases, and [7], [8], [10], [17], [18] for constrained-deadline cases. All of them can be adopted based on a similar argument in the proof of Lemma 2.

Since we assume that $U_i \leq 1$ and $C_i \leq D_i$ for every task τ_i in $\mathbf{T}$, the tasks in $\mathbf{T}_{high}$ can meet their deadlines if $|\mathbf{T}_{high}| \leq M$. If $|\mathbf{T}_{high}| > M$, we know that $\sum_{\tau_i \in \mathbf{T}} U_i \geq (M+1)/2$. Therefore, the speedup factor of the greedy three-cluster priority assignment algorithm is defined by the speedup factor (if ≥ 2) of the corresponding scheduling policies and schedulability tests for the tasks in $\mathbf{T}_{middle}$. As a result, we can have a speedup factor of $3 - \frac{1}{M}$ for constrained-deadline task systems inherited from [7], [10] or a speedup factor of 3.73 for arbitrary-deadline task systems inherited from [6].

6.3 Assignment from Lowest Priority

In our greedy priority assignment in Section 6.1, soft real-time tasks in $\mathbf{T}_{low}$ may have unbounded tardiness due to unfavorably high utilizations from tasks in $\mathbf{T}_{middle}$. This may be tackled by promoting the priorities of same soft real-time tasks in $\mathbf{T}_{low}$ to higher levels. Unfortunately, permuting all the priority assignments for checking the feasibility is computationally intractable. Instead, we here consider *Optimal Priority Assignment* (OPA) proposed by Audsley [3]. The OPA algorithm assigns each priority level to one of the unassigned tasks that has no deadline miss along with the other unassigned tasks, assumed to have higher priorities. The iterative priority assignmen terminates as soon as either no unassigned task can be assigned at the current priority level or all priority levels are assigned. OPA was designed originally for uniprocessor fixed-priority scheduling for hard real-time tasks with arbitrary deadlines. It has been extended to handle also global multiprocessor fixed-priority scheduling by Davis and Burns [11] for constrained-deadline hard real-time tasks.

As pointed out by Davis and Burns [11], if a schedulability test is compatible with the OPA algorithm, we can obtain an *optimal* priority assignment (with respect to the schedulability test). Nevertheless, for being OPA-compatible, three conditions must be complied. These conditions are as follows [11]:

- **Condition 1.** The schedulability of a task τ_k may, according to schedulability test $\mathcal{S}$, depend on any independent properties of tasks with priorities higher than τ_k, but not on any properties of those tasks that relate to their relative priority ordering.
- **Condition 2.** The schedulability of a task τ_k may, according to schedulability test $\mathcal{S}$, depend on any independent properties of tasks with priorities lower than τ_k, but not on any properties of those tasks that relate to their relative priority ordering.
- **Condition 3.** When the priorities of any two tasks of adjacent priority are swapped, the task being assigned the higher priority cannot become unschedulable according to schedulability test $\mathcal{S}$, if it was previously schedulable at the lower priority.

The tests in Theorems 1 and 2 are not compatible with OPA, since they depend on the response time of the higher-priority tasks and the response time of a higher priority task depends on the priority orderings. Therefore, Condition 1 is violated. Suppose that we are now about to assign the priority level q and there are q tasks left in $\mathbf{T}_{soft} \cup \mathbf{T}_{hard}$. To verify whether task $\tau_k \in \mathbf{T}_{soft} \cup \mathbf{T}_{hard}$ can be feasibly assigned in the priority level q, we need to check two cases:

- If task τ_k is an SRT task, we need to check whether $(M-1)U_k + \sum_{\tau_i \in \mathbf{T}_{soft} \cup \mathbf{T}_{hard}} U_i \leq M$ holds.
- If task τ_k is an HRT task, we need to check whether $R_k^{\dagger} \leq D_k$.

In fact, the first test for SRT task τ_k is OPA-compatible, but the second test for HRT task τ_k is not. For higher-priority hard real-time tasks, we can greedily assume that their worst-case response time will be met and the examination of the assumption will be taken place in the upcoming iterations. However, for a higher-priority soft real-time task, its worst-case response time (or tardiness) may be very long, which disables the possibility to directly use Theorem 1 for verifying whether an HRT task τ_k can be placed in the priority level q if $\mathbf{T}_{soft}$ is not empty.

We can observe one trick to make the test in Theorem 1 OPA-compatible by investigating Eq. (5) to quantify the interference of a higher-priority task $\tau_i \in hp(\tau_k)$. If τ_i is a soft real-time task, the maximum interference resulting from $I_i^2(t)$ is at most $t - h \cdot C_k + 1$, regardless of its worst-case response time. Therefore, if there are s SRT tasks in $hp(\tau_k)$, then the worst-case response time of task τ_k can be considered as if there is only interference from the HRT tasks in $hp(\tau_k)$ and $M - s$ processors. As a result, we have the following corollary of Theorem 1 to safely handle such situations.

Corollary 3. *Suppose that there are s SRT tasks in $hp(\tau_k)$ with $s < M$. Let M' be $M - s$. The worst-case response time R_k of task τ_k is at most*

$$R_k \leq \begin{cases} C_k & \text{if } |hp(\tau_k)| < M, \\ R_k^{\sharp} & \text{else if } M'U_k + \sum_{\tau_i \in (hp(\tau_k) \cap \mathbf{T}_{hard})} U_i < M', \\ \infty & \text{otherwise,} \end{cases} \tag{10}$$

where

$$R_k^{\sharp} = \frac{M'C_k + Z_{\Sigma} + \sum_{\tau_i \in (hp(\tau_k) \cap \mathbf{T}_{hard})} C_i(1-U_i)}{M' - \sum_{\tau_i \in (hp(\tau_k) \cap \mathbf{T}_{hard})} U_i}$$

and Z_{Σ} denotes the sum of the $(M'-1)$ largest $D_i \cdot U_i$'s among the tasks in $hp(\tau_k) \cap \mathbf{T}_{hard}$.

Algorithm 1 OPA-based Priority Assignment

```
Input: $\mathbf{T}_{soft}$, $\mathbf{T}_{hard}$, $M$ processors;
Output: priority assignment for the tasks in $\mathbf{T}_{soft}$ and $\mathbf{T}_{hard}$;
    $q \leftarrow |\mathbf{T}_{soft}| + |\mathbf{T}_{hard}|$;
 1: while $q \geq 1$ do
 2:     if $|\mathbf{T}_{soft}| > 0$ then
 3:         if $\exists\tau_k$ in $|\mathbf{T}_{soft}|$ such that $(M-1)U_k + \sum_{\tau_i \in \mathbf{T}_{soft} \cup \mathbf{T}_{hard}} U_i < M$ then
 4:             assign such a $\tau_k$ to priority level $q$ (ties are broken arbitrarily);
 5:             $\mathbf{T}_{soft} \leftarrow \mathbf{T}_{soft} \setminus \{\tau_k\}$; $q \leftarrow q - 1$;
 6:             continue;
 7:         end if
 8:     end if
 9:     if $|\mathbf{T}_{hard}| > 0$ then
10:         if $\exists\tau_k$ in $|\mathbf{T}_{hard}|$ such that $\tau_k$ can be feasibly scheduled by Corollary 3 then
11:             assign such a $\tau_k$ to priority level $q$ (ties are broken arbitrarily);
12:             $\mathbf{T}_{hard} \leftarrow \mathbf{T}_{hard} \setminus \{\tau_k\}$; $q \leftarrow q - 1$;
13:             continue;
14:         end if
15:     end if
16:     return "no feasible priority assignment is found";
17: end while
18: return "the priority assignment as a feasible one";
```

Proof: This corollary can be proved formally by following the same analysis flow as in [18]. The reason why it holds is due to the over-approximation here by greedily setting the worst-case response time R_i of a higher-priority SRT task $\tau_i \in hp(\tau_k)$ to any arbitrarily large number to enforce $I_i^2(t)$ to $t - h \cdot C_k + 1$. In the response time analysis in Eq. (3), this logically implies that one processor is removed from the available processors as if this processor is always occupied by this SRT task τ_i. ■

We now prove that testing an HRT task τ_k by using Corollary 3 is OPA-compatible since all the three conditions listed above can be easily verified.

Lemma 4. *The sufficient schedulability test by Corollary 3 is OPA-compatible.*

Proof: Eq. (10) for the schedulability of task τ_k depends only on the set of higher-priority tasks (both hard and soft real-time tasks) but not on their relative priority ordering. Hence, Condition 1 holds. Similarly, they are independent on the set of lower-priority tasks, and hence Condition 2 holds.

Inspections of Eq. (10) show that the more the number of available processors, the less the response time by Eq. (10). As a result, the set of soft real-time high-priority tasks assumed to fully occupy one processor for each task according to Eq. (10) adversely affects the upper-bound response time by Eq. (10) of the task being analyzed. Similarly, the set of hard real-time higher-priority tasks adversely affects the upper-bound response time by Eq. (10) of the task being analyzed: the more the hard real-time higher-priority tasks, the larger the response time by Eq. (10).

Now, consider two tasks τ_a and τ_b initially at priorities k and $k+1$, respectively. We separately consider two cases as follows:

- If task τ_b is a soft real-time task and the condition that $(M-1)U_k + \sum_{\tau_i \in \mathbf{T}_{soft} \cup \mathbf{T}_{hard}} U_i < M$ in Algorithm 1 is satisfied, this condition is still satisfied when task τ_b shifted one priority level up to priority k, since there will be a decrease in the utilizations from higher-priority tasks.
- If task τ_b is a hard real-time task and schedulable according to Corollary 3, it is still schedulable when it is shifted one priority level up to priority level k, since the only change of higher-priority task demand is the removal of task τ_a (soft or hard real-time tasks) from the tasks that are assigned higher priority than task τ_b .

Hence, in either of the two cases, Condition 3 holds. ■

Similarly, it is not difficult to prove that the overall test (for testing an SRT task τ_k or an HRT task τ'_k) is OPA-compatible. Therefore, we can adopt OPA for the priority assignment. The algorithm is listed in Algorithm 1. According to the above analysis, we also notice that the schedulability test for HRT tasks is more pessimistic, as we have to remove one processor greedily when there is one higher-priority SRT tasks. However, the test for soft real-time tasks is only to verify whether $(M-1)U_k + \sum_{\tau_i \in \mathbf{T}_{soft} \cup \mathbf{T}_{hard}} U_i \leq M$, with less pessimism. Therefore, if we can either assign priority level q to an HRT task or an SRT task (they are both schedulable at this priority level), it is in general better to assign this priority level to an SRT task. As a result, in Algorithm 1, we adopt such a policy to start from soft real-time tasks before HRT tasks when considering the priority assignment.

The priority assignment based on OPA can be good if the number of soft real-time tasks is not too large. The reason is due to the over-approximation in Corollary 3 by greedily reducing the availability of s processors when there are s higher-priority SRT tasks. However, such a treatment may be pessimistic as the worst-case response time of a soft real-time task may still be short if it is a higher-priority task.

Theorem 8. *If Algorithm 1 derives a priority assignment, this assignment provides a feasible schedule for $\mathbf{T}_{hard}$ and $\mathbf{T}_{soft}$.*

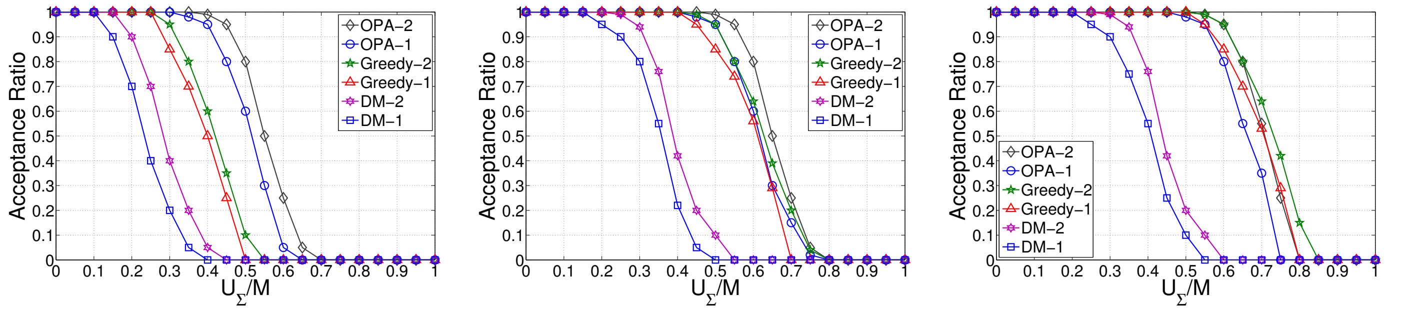
Proof: This comes from the above discussions. ■

7 Experiments

In this section, we conduct experiments using synthesized task sets with a range of parameters that are wide enough to cover SRT workloads in practice (e.g., the periods for SRT video decoding applications typically range from 17ms (i.e., 4K video with 60 FPS) to 67ms (e.g., AVI video with 15 FPS)). The metric to compare the results is to measure the acceptance ratio of the above tests with respect to a given goal of task set utilization level U_Σ/M. We generate 100 task sets for each utilization level (i.e., total utilization of the generated task set divided by M), from 0.01 to 0.99, in steps of 0.01. The acceptance ratio of a level is the number of task sets that are schedulable divided by the number of task sets, i.e., 100.

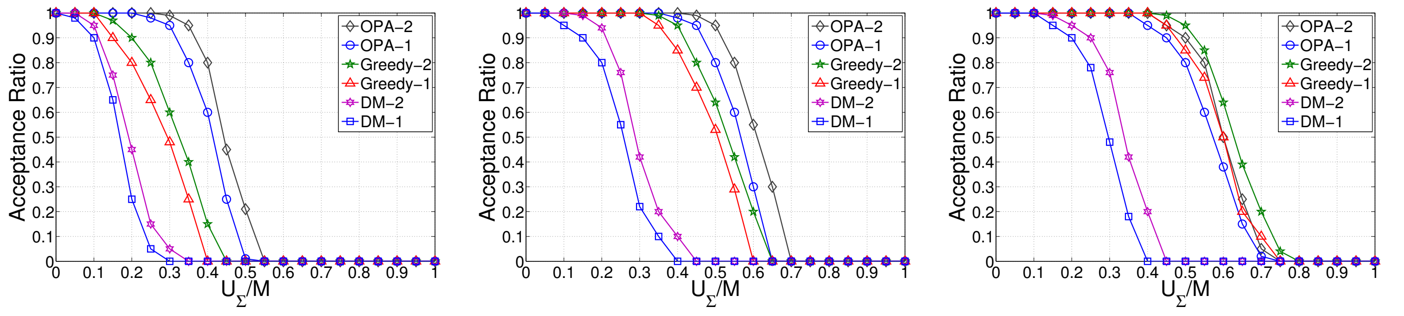
7.1 Simulation Environment

We first generated a set of sporadic tasks. The cardinality of the task set was 10 times the number of processors. The UUniFast-Discard method [9] was adopted to generate a set of utilization values with the given goal. We here used the approach suggested by Davis and Burns [13] to generate the task periods according to the exponential distribution. The order of magnitude p to control the period values between largest and smallest periods is parameterized in evaluations. (For example, $1ms$–$10ms$ for $p = 1$, $1ms$–$100ms$ for $p = 2$). The execution time was set accordingly, i.e., $C_i = T_i U_i$. The value of $\frac{|\mathbf{T}_{hard}|}{|\mathbf{T}_{soft}|}$ to control the ratio between the number of hard real-time tasks and the number of soft real-time tasks is also parameterized in evaluations. (For example, $\frac{|\mathbf{T}_{hard}|}{|\mathbf{T}_{soft}|} = 2, 1, \frac{1}{2}$ and $\frac{1}{9}$.) We also



(a) $\frac{|\mathbf{T}_{hard}|}{|\mathbf{T}_{soft}|} = 2$ (b) $\frac{|\mathbf{T}_{hard}|}{|\mathbf{T}_{soft}|} = \frac{1}{2}$ (c) $\frac{|\mathbf{T}_{hard}|}{|\mathbf{T}_{soft}|} = \frac{1}{9}$

Fig. 3: Acceptance ratios on 8 processors, where $\frac{D_i}{T_i} \in [0.8, 1]$ for $\tau_i \in \mathbf{T}_{hard}$ and $p = 2$.



(a) $\frac{|\mathbf{T}_{hard}|}{|\mathbf{T}_{soft}|} = 2$ (b) $\frac{|\mathbf{T}_{hard}|}{|\mathbf{T}_{soft}|} = \frac{1}{2}$ (c) $\frac{|\mathbf{T}_{hard}|}{|\mathbf{T}_{soft}|} = \frac{1}{9}$

Fig. 4: Acceptance ratios on 8 processors, where $\frac{D_i}{T_i} \in [0.6, 0.8]$ for $\tau_i \in \mathbf{T}_{hard}$ and $p = 1$.

report the evaluation results of the proposed tests with pure soft real time task systems.

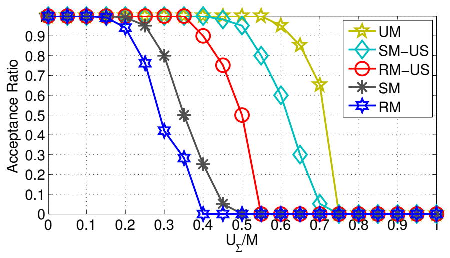
7.2 Results for Mixed HRT and SRT Systems

We have proposed two priority assignment algorithms: greedy three-cluster priority assignment (Greedy) in Sections 6.1 and 6.2 and OPA-compatible priority assignment (OPA) in Section 6.3. The tests evaluated are shown as follows:

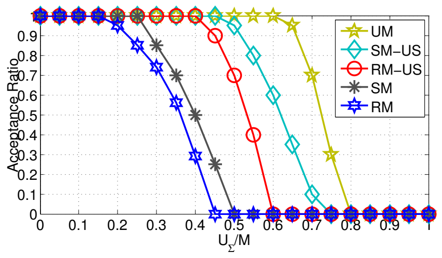
- DM-1: The priority levels are assigned according to deadline-monotonic (DM). Since a soft real-time task does not have any specified relative deadline in our problem definition, we artificially use its period while using DM priority assignment. However, the schedulability test applies Theorem 1, from the highest-priority task to the lowest-priority task, to calculate the corresponding worst-case response time and to verify whether the worst-case response time and tardiness can satisfy the requirement.
- DM-2: The same as DM-1, and the response time calculation is according to Theorem 2.
- Greedy-1: The priority assignment is the three-cluster priority assignment in Section 6.2 (based on DM in $\mathbf{T}_{middle}$) and the response time calculation is according to Theorem 1.
- Greedy-2: The same as Greedy-1 and the response time calculation is according to Theorem 2.
- OPA-1: The priority assignment is OPA in Section 6.3 and the response time calculation of a hard real-time task is according to the extension, i.e., Corollary 3, of Theorem 1.
- OPA-2: The priority assignment is OPA in Section 6.3 and the response time calculation of a hard real-time task is according to the extension of Theorem 2.

The obtained schedulability results are shown in Figure 3 (the organization of which is explained in the caption). Task relative deadlines for HRT tasks were uniformly drawn from the interval $[0.8T_i, T_i]$. We first notice that the response time analysis with OPA and Greedy can admit a larger number of task sets than the response time analysis with DM in all cases. For example, as seen in Figure 3a, when $\frac{|\mathbf{T}_{hard}|}{|\mathbf{T}_{soft}|} = 2$ and $M = 8$, OPA and Greedy can achieve 100% schedulability when $\frac{U_\Sigma}{M}$ equals 0.25 and 0.3 respectively, while DM fail to do so when $\frac{U_\Sigma}{M}$ merely exceeds 0.1 respectively. Note that when $\frac{|\mathbf{T}_{hard}|}{|\mathbf{T}_{soft}|}$ is smaller, the improvement margin by OPA and Greedy over DM increases. This is because OPA and Greedy prioritize the SRT tasks with reasonable strategies, but for DM, the priority of task is assigned only according to deadline-monotonic scheduling. Another observation is Greedy outperforms OPA when $\frac{|\mathbf{T}_{hard}|}{|\mathbf{T}_{soft}|}$ becomes small enough. This is because when more SRT tasks are involved in the task system, the performance of OPA suffers from its pessimistic strategy when there are higher-priority SRT tasks.

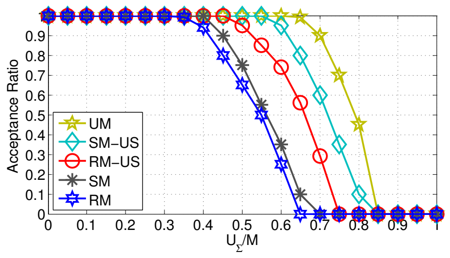
To show the impact of the relative deadline on the acceptance ratio for all tests, we also performed simulations with a smaller $\frac{D_i}{T_i}$ and the results are shown in Figure 4. Task relative deadlines were uniformly drawn from the interval $[0.6T_i, 0.8T_i]$. As seen in Figure 4, OPA and Greedy still achieve better performance than DM. Another observation is that for all scheduling algorithms, the acceptance ratio decreases compared to the corresponding results in Figure 3. This is because more HRT tasks miss their deadlines when their relative deadlines decrease.



(a) $p = 1$



(b) $p = 2$



(c) $p = 3$

Fig. 5: Acceptance ratios with only SRT tasks on 8 processors.

7.3 Results for Pure SRT Systems

In addition to the mixed task systems' schedulability test, the pure soft real-time task systems' behaviour is also important. We here report the experimental results for SRT task systems. The tests evaluated are shown as follows:

- UM: The priority levels of the tasks are assigned according to utilization-monotonic (UM) scheduling.
- RM: The priority levels of the tasks are assigned according to RM scheduling.
- RM-US: The priority levels of the task are assigned according to RM-US [2] scheduling.
- SM: The priority levels of the tasks are assigned according to SM scheduling.
- SM-US: The priority levels of the task are assigned according to SM-US [1] scheduling.

In Figure 5, we show the obtained schedulability results for pure SRT task systems. Figure 5 depicts the average of the computed acceptance ratio for each of the priority assignments when $M = 8$. In all the tests, we use Corollary 1 to verify whether the tasks have bounded tardiness. As shown in Theorem 4, UM is an optimal priority assignment in such cases. Therefore, UM analytically dominates the others. Such analytical dominance can also be seen in the reported results.

8 Conclusion and Extensions

To the best of our knowledge, this is the first result tackling bounded tardiness in multiprocessor fixed-priority scheduling. For the pure SRT case, we show that the utilization-monotonic priority assignment yields a utilization bound of $\frac{M+1}{2M}$. For a mixed set of HRT and SRT tasks, we present two fixed-priority assignment algorithms and their associated schedulability tests.

With our new findings in this paper, we show that introducing soft real-time tasks does not create additional problems (with respect to utilization bounds, speedup factors, or augmentation factors) for scheduling if the priority assignments are properly done. Note that our proposed greedy three-cluster priority assignment in Section 6.1 is in general independent from the adopted schedulability tests. Although we present the paper based on the recent result by Huang and Chen [18], any new results should be directly applicable. This paper is also an initial thread for further exploring the applicability of mixed SRT and HRT sporadic tasks in the system. A very interesting future work is to extend these results to provide *fair* or *weighted* tardiness guarantees.

References

[1] B. Andersson. Global static-priority preemptive multiprocessor scheduling with utilization bound 38%. In *Principles of Distributed Systems, 12th International Conference, OPODIS*, pages 73–88, 2008.

[2] B. Andersson, S. K. Baruah, and J. Jonsson. Static-priority scheduling on multiprocessors. In *Real-Time Systems Symposium (RTSS)*, pages 193–202, 2001.

[3] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5):284–292, Sep 1993.

[4] T. P. Baker. Multiprocessor EDF and deadline monotonic schedulability analysis. In *IEEE Real-Time Systems Symposium*, pages 120–129, 2003.

[5] T. P. Baker. An analysis of fixed-priority schedulability on a multiprocessor. *Real-Time Systems*, 32(1-2):49–71, 2006.

[6] S. Baruah and N. Fisher. Global fixed-priority scheduling of arbitrary-deadline sporadic task systems. In *Proceedings of the 11th International Conference on Principles of Distributed Systems*, pages 215–226, 2008.

[7] S. K. Baruah, V. Bonifaci, A. Marchetti-Spaccamela, and S. Stiller. Improved multiprocessor global schedulability analysis. *Real-Time Systems*, 46(1):3–24, 2010.

[8] M. Bertogna, M. Cirinei, and G. Lipari. New schedulability tests for real-time task sets scheduled by deadline monotonic on multiprocessors. In *Principles of Distributed Systems, 9th International Conference, OPODIS*, pages 306–321, 2005.

[9] E. Bini and G. C. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Systems*, 30(1-2):129–154, 2005.

[10] J.-J. Chen, W. Huang, and C. Liu. k2Q: A quadratic-form response time and schedulability analysis framework for utilization-based analysis. In *2016 IEEE Real-Time Systems Symposium, RTSS*, pages 351–362, 2016.

[11] 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.

[12] R. I. Davis and A. Burns. A survey of hard real-time scheduling for multiprocessor systems. *ACM Computing Surveys (CSUR)*, 43(4):35, 2011.

[13] R. I. Davis, A. Zabos, and A. Burns. Efficient exact schedulability tests for fixed priority real-time systems. *Computers, IEEE Transactions on*, 57(9):1261–1276, 2008.

[14] U. Devi. *Soft Real-Time Scheduling on Multiprocessors*. PhD thesis, University of North Carolina at Chapel Hill, 2006.

[15] U. Devi and J. Anderson. Tardiness bounds under global EDF scheduling on a multiprocessor. In *Proceedings of the 26th IEEE Real-Time Systems Symposium*, pages 330-341, 2005.

[16] S. K. Dhall and C. L. Liu. On a real-time scheduling problem. *Operations Research*, 26(1):127–140, 1978.

[17] N. Guan, M. Stigge, W. Yi, and G. Yu. New response time bounds for fixed priority multiprocessor scheduling. In *IEEE Real-Time Systems Symposium*, pages 387–397, 2009.

[18] W.-H. Huang and J.-J. Chen. Response time bounds for sporadic arbitrary-deadline tasks under global fixed-priority scheduling on multiprocessors. In *Proceedings of the 23rd International Conference on Real Time Networks and Systems, RTNS*, pages 215–224, 2015.

[19] H. Leontyev. *Compositional Analysis Techniques For Multiprocessor Soft Real-Time Scheduling*. PhD thesis, University of North Carolina at Chapel Hill, 2010.

[20] J. Li, J.-J. Chen, K. Agrawal, C. Lu, C. Gill, and A. Saifullah. Analysis of federated and global scheduling for parallel real-time tasks. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 85–96, 2014.

[21] 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.

[22] J. Liu. *Real-Time Systems*. Prentice Hall, 2000.

[23] C. Phillips, C. Stein, E. Torng, and J. Wein. Optimal time-critical scheduling via resource augmentation. In *Proc. of the 29th ACM Symposium on Theory of Computing*, pages 140–149, 1997.