Guaranteeing Pfair Supertasks by Reweighting*

Philip Holman and James H. Anderson

Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599-3175
Phone: (919) 962-1757
Fax: (919) 962-1799
E-mail: {holman,anderson}@cs.unc.edu

May 2001

Abstract

We consider the "supertask" approach, proposed by Moir and Ramamurthy at RTSS '99 as means for supporting non-migratory tasks in Pfair-scheduled systems. In this approach, tasks bound to the same processor are combined into a single supertask, which is scheduled as an ordinary Pfair task; when a supertask is scheduled, one of its component tasks is selected for execution. Unfortunately, while Moir and Ramamurthy's paper suggests that supertasking is a promising approach, counterexamples presented by them show that non-migratory tasks can actually miss their deadlines when supertasking is used in conjunction with all known Pfair scheduling algorithms.

In this paper, we show that such deadline misses can be prevented by inflating each supertask's utilization. We present experimental evidence that shows that the required inflation factors should be small in practice. We also show that these inflation factors usually can be reduced if, instead of being scheduled in a Pfair manner, non-migratory tasks are scheduled (perhaps "unfairly") using earliest-deadline-first priorities. In soft-real-time and rate-based systems, it may be permissible for a task to miss a deadline by a small amount. We show that our inflation factors also can be reduced if this is the case.

*Work supported by NSF grants CCR 9732916, CCR 9972211, CCR 9988327, and ITR 0082866. The first author was also supported by an IBM fellowship.

1 Introduction

Pfair scheduling, first proposed by Baruah *et al.* [4, 5], is the only known optimal method for scheduling periodic tasks on multiple processors. Pfair scheduling differs from more conventional scheduling disciplines in that tasks are explicitly required to make progress at steady rates, while respecting a fixed allocation quantum. Specifically, Pfairness requires the allocation error for each task to be always less than one quantum, where "error" is determined by comparing to an ideal fluid system. Due to this requirement, each task is effectively subdivided into quantum-length *subtasks* that must execute within *windows* of approximately equal lengths: if a subtask executes outside of its window, then the corresponding task's allocation error becomes either too large or too small.

Three optimal Pfair scheduling algorithms have been proposed: PF [4], PD [5], and PD^2 [3]. Each of these algorithms uses an earliest "pseudo-deadline" rule to prioritize subtasks — a subtask's *pseudo-deadline* is simply the last time slot of its window. If two subtasks have the same pseudo-deadline, then some tie-breaking strategy must be employed. PF, PD, and PD^2 use different tie-breaking schemes, with the PF scheme being the least efficient to calculate, and the PD^2 scheme being the most efficient. In each of these algorithms, it is assumed that tasks are fully migratory.

In recent work, Moir and Ramamurthy questioned this task-migration assumption [8]. Their work was motivated by the fact that many applications have tasks that cannot migrate because they interact with sensors and actuators at fixed locations. By using a flow-graph argument, Moir and Ramamurthy showed that a Pfair schedule exists for any feasible task set with non-migratory tasks. In the worst case, the flow graph is of exponential size, and thus cannot be practically applied to produce schedules on-line. As an alternative, Moir and Ramamurthy proposed the *supertask* approach. In this approach, the non-migratory tasks bound to a specific processor are combined into a single "supertask," which is then scheduled as an ordinary Pfair task; when a supertask is scheduled, one of its component tasks is selected for execution. Unfortunately, while Moir and Ramamurthy's paper suggests that supertasking is a promising approach, counterexamples presented by them show that non-migratory tasks can actually miss their deadlines when supertasking is used in conjunction with PF, PD, or PD^2.

Example. To see why supertasking can fail, consider the two-processor Pfair schedule shown in Fig. 1(a). In the Pfair scheduling literature, the ratio of a task's execution cost to its period is referred to as its *weight*; a task's weight determines the length and alignment of its Pfair windows. The task set in Fig. 1(a) consists of four normal tasks **T2**–**T5** with weights $\frac{2}{9}$, $\frac{1}{3}$, $\frac{1}{3}$, and $\frac{1}{2}$, respectively, along with one supertask, **T1**, which represents two component tasks **C1**–**C2** with weights $\frac{1}{5}$ and $\frac{1}{45}$, respectively (shown in the lower region). **T1** competes with a weight of $\frac{1}{5} + \frac{1}{45} = \frac{2}{9}$. In the figure, each subtask window is indicated by a marked interval; for example, the first window of **T1** spans the interval $[0, 4]$. An "X" is used to denote the quantum allocated to a subtask; for example, the first subtask of **T1** is scheduled in time slot 1. All scheduling decisions in the upper region are consistent with the PD^2 Pfair algorithm. In the lower region, allocations within **T1** are shown.

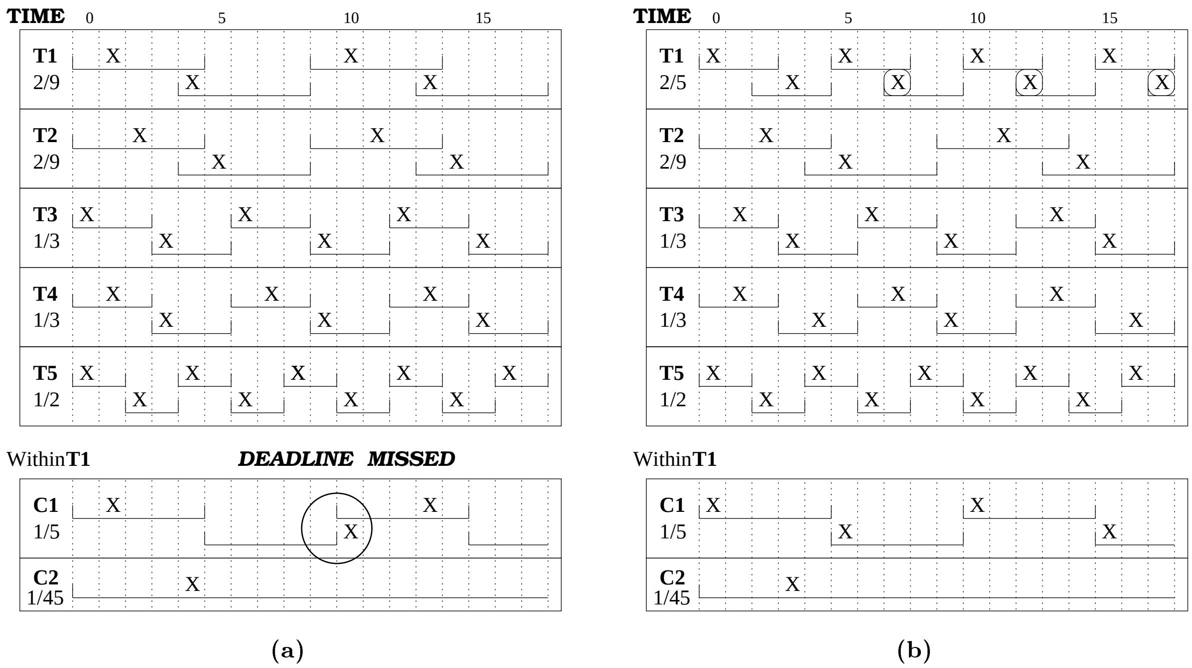


Figure 1: PD^2/EPDF schedule with a **(a)** normal and **(b)** reweighted supertask **T1** on two processors.

These allocations are based on *earliest-pseudo-deadline-first* (EPDF) priorities. Under EPDF scheduling, an earliest-pseudo-deadline rule is used to prioritize subtasks, with ties broken arbitrarily. (EPDF is optimal on one or two processors [3]. On more than two processors, a non-trivial tie-breaking scheme is required.)

As the schedule shows, **C1** misses a pseudo-deadline at time slot 9. This is because no quantum is allocated to **T1** in the interval $[5, 9]$. In general, component tasks may be misallocated if there exists an interval that properly contains more component-task windows than supertask windows. Observe that $[5, 9]$ is such an interval since it contains one component-task window and no supertask windows.

Contributions of this paper. In this paper, we show that component-task misallocations within a supertask can be prevented by inflating the supertask's weight. We present reweighting rules both for systems in which component tasks are EPDF-scheduled, and also systems in which they are scheduled using earliest-deadline-first (EDF) priorities. Under EDF scheduling, all subtasks within the same job (*i.e.*, task instance) have the same priority, which is determined by the job's deadline. By considering EDF-scheduled component tasks, this paper breaks new ground, as Moir and Ramamurthy only considered EPDF-scheduled component tasks. The main virtue of using EDF for component-task scheduling is that it often results in lower inflation factors. This is because, under EDF scheduling, pseudo-deadlines strictly *within* a job can be missed.

Our reweighting rules are expressed in a general form that also allows them to be applied in systems in which tasks can overshoot their deadlines (either subtask pseudo-deadlines or job deadlines) by some constant factor. Such flexibility might be useful in both soft-real-time and rate-based systems. Our results show that, as the overshoot parameter increases, the required inflation factor decreases. As a result, much lower inflation factors are typically needed in systems in which bounded deadline overshoots can be tolerated.

Our proofs make no assumptions concerning the Pfair algorithm used to allocate processor time to supertasks, other than the Pfair allocation-error constraint. Indeed, as our analysis shows, the problem of computing inflation factors for reweighting is really a "uniprocessor" problem. In essence, a supertask can be viewed as a uniprocessor that executes at a steady rate, where "steady" is defined with respect to Pfair allocation rates.

Supertasking is beneficial in contexts other than Moir and Ramamurthy's non-migratory task problem. (This is why we call the tasks represented by a supertask "component tasks" rather than "non-migratory tasks.") For example, since a supertask simulates a quantum-based uniprocessor, object sharing by component tasks of the same supertask can be implemented using highly optimized uniprocessor wait-free algorithms [1] rather than more costly multiprocessor schemes. (To apply a multiprocessor scheme such as the multiprocessor priority ceiling protocol [9] in a Pfair-scheduled system, significant modifications to the protocol would be required.) In addition, supertasks can be applied to service aperiodic job requests, with the advantage that response times are more predictable than is possible with background scheduling. (In this paper, we only consider *periodic* component tasks; aperiodic jobs will be considered in a forthcoming paper [7].) These examples suggest that supertasking is a useful mechanism — *when it works*.

Example, revisited. Fig. 1(b) shows the result of applying our EPDF reweighting rules to supertask **T1** in Fig. 1(a). As seen in Fig. 1(b), deadline misses are avoided by using an inflation factor of $\frac{2}{5} - \frac{2}{9} = \frac{8}{45}$. Because of the inflation, some quanta allocated to **T1** are unused (indicated by circled X's). In practice, inflation factors should be much lower. In particular, our reweighting rules never reweight by more than $\frac{1}{mcw}$, where mcw is the minimum window length of any component task (this follows from Rule 3B in Sec. 3.1). In Fig. 1, mcw is 5. In practice, real tasks are expected to be lighter than $\frac{1}{5}$ and hence have much longer windows. (Since our example schedules must fit within a page, we are constrained to use somewhat heavy tasks.)

The remainder of this paper is organized as follows. In Sec. 2, we explain the supertasking problem in detail and provide definitions that will be used in the remainder of the paper. Our reweighting rules are then presented and proven correct in Secs. 3 and 4, for systems with EPDF- and EDF-scheduled component tasks, respectively. In Sec. 5, we present experimental results that show the practicality of reweighting. We conclude in Sec. 6.

2 Preliminaries

In this section, we present definitions and notational conventions that will be used in the rest of the paper.

Quantum scheduling. In Pfair-scheduled systems, processor time is allocated in discrete time units, or *quanta*. The time interval $[t, t+1)$, where t is a non-negative integer, is called *time slot* t. The sequence of allocation decisions over time defines a "schedule." Formally, a schedule $\mathcal{S}$ is a mapping $\mathcal{S}\colon \mathbf{T} \times \mathcal{N} \mapsto \{0, 1\}$, where $\mathbf{T}$ is a task set and $\mathcal{N}$ is the set of non-negative integers, with the interpretation that $\mathcal{S}(T,\, t)$ quanta are allocated to task $T \in \mathbf{T}$ in time slot t.

Supertasks. As mentioned in the introduction, the problem of determining inflation factors is really a "uniprocessor" problem that can be solved by considering one supertask in isolation. We let $\mathcal{T}$ denote this supertask and we let τ denote the set of N periodic component tasks represented by $\mathcal{T}$. Each component task T has an execution requirement $T.e$ and a period $T.p$. Every $T.p$ time units, T releases a new *job* with a cost of $T.e$ time units. T's k^{th} job, where $k \geq 1$, is released at time slot $(k-1) \cdot T.p$, has a deadline at time slot $k \cdot T.p - 1$, and must be allocated $T.e$ time units within $[(k-1) \cdot T.p, k \cdot T.p)$. (The k^{th} job deadline occurs at $k \cdot T.p - 1$ because time slots are numbered from 0.) The ratio $\frac{T.e}{T.p}$ is called the *weight* of T and is denoted $T.w$.

Because supertasks are scheduled using reweighted parameters, we need to distinguish a supertask's "actual" parameters from its "scheduling" parameters. $\mathcal{T}$'s *actual* parameters include its original execution requirement $\mathcal{T}.e$, period $\mathcal{T}.p$, and weight $\mathcal{T}.w = \frac{\mathcal{T}.e}{\mathcal{T}.p}$. These parameters represent the requirements of the component tasks. $\mathcal{T}$'s *scheduling* parameters, which result from inflating, include an execution requirement $\mathcal{T}.\hat{e}$, period $\mathcal{T}.\hat{p}$, and weight $\mathcal{T}.\hat{w} = \frac{\mathcal{T}.\hat{e}}{\mathcal{T}.\hat{p}}$. These parameters are used in scheduling $\mathcal{T}$. For example, in Fig. 1(b), **T1** has actual parameters $\mathbf{T1}.e = 2$, $\mathbf{T1}.p = 9$, and $\mathbf{T1}.w = \frac{2}{9}$. Its scheduling parameters are $\mathbf{T1}.\hat{e} = 2$, $\mathbf{T1}.\hat{p} = 5$, and $\mathbf{T1}.\hat{w} = \frac{2}{5}$. For consistency, we will also use this notation for component tasks with the understanding that $T.\hat{e}$, $T.\hat{p}$, and $T.\hat{w}$ are equal to $T.e$, $T.p$, and $T.w$, respectively. Therefore, we define the actual weight of a supertask $\mathcal{T}$ to be $\mathcal{T}.w = \sum_{T \in \tau} T.\hat{w}$. The difference $\mathcal{T}.\hat{w} - \mathcal{T}.w$ is called $\mathcal{T}$'s *inflation factor*.

Subtasks and windows. As in [3], we let V_i denote the i^{th} subtask of V, where V is either a supertask or a component task; V_i is the subtask associated with the i^{th} quantum allocated to V. We let $\omega(V_i)$ denote the *window* of V_i. For example, in Fig. 1, $\omega(\mathbf{T2}_2) = [4, 9]$. We let $r(V_i)$ denote the *pseudo-release* of V_i (*i.e.*, the first time slot of V_i's window) and $d(V_i)$ denote the *pseudo-deadline* of V_i (*i.e.*, the last time slot of V_i's window). For example, in Fig. 1, $r(\mathbf{T2}_2) = 4$ and $d(\mathbf{T2}_2) = 9$. For brevity, we often refer to pseudo-deadlines and pseudo-releases as simply deadlines and releases, respectively. We let *windows*(V, I) (respectively, *jobs*(V, I)) denote the number of windows (jobs) of V properly contained in the interval I.

Scheduling algorithms. In our analysis, we assume that the supertask $\mathcal{T}$ uses either EPDF or EDF priorities to schedule its component tasks. Under EPDF (respectively, EDF) scheduling, higher priority is given to subtasks (jobs) with earlier deadlines (job deadlines). Ties can be broken arbitrarily. In addition, we require that all schedules $\mathcal{S}$ containing $\mathcal{T}$ satisfy the Pfairness constraint (P1) given below. Since all properties of Pfair schedulers that we require can be derived from this condition, we will not describe Pfair scheduling in any further detail. Instead, we refer the interested reader to [4, 5, 3].

P1: $\mathcal{T}$ receives either $\lfloor \mathcal{T}.\hat{w} \cdot t \rfloor$ or $\lceil \mathcal{T}.\hat{w} \cdot t \rceil$ quanta over any interval $[0,t)$ in $\mathcal{S}$.

We let *allocation*$(V, I, \mathcal{S}) = \sum_{t=a}^{b} \mathcal{S}(V, t)$ denote the number of quanta allocated to V over the interval $I = [a, b]$ in schedule $\mathcal{S}$, where V is either a supertask or a component task.

Minimal windows and periods. By [3], the shortest window of a Pfair-scheduled task V is of length $\left\lceil \frac{1}{V.\hat{w}} \right\rceil$. We let $\mathcal{T}.mcw$ denote the shortest window of any component task of $\mathcal{T}$. Formally,

$$\mathcal{T}.mcw = \mathsf{min}\left\{ \left\lceil \frac{1}{T.\hat{w}} \right\rceil \;\middle|\; T \in \tau \right\}. \tag{1}$$

We also define $\mathcal{T}.mcp$ to be the shortest period of any component task of $\mathcal{T}$, as follows.

$$\mathcal{T}.mcp = \mathsf{min}\{T.\hat{p} \mid T \in \tau\} \tag{2}$$

In addition, we let $\mathcal{T}.msw$ denote the shortest *supertask* window when $\mathcal{T}.\hat{w} = \mathcal{T}.w$. Formally,

$$\mathcal{T}.msw = \left\lceil \frac{1}{\mathcal{T}.w} \right\rceil. \tag{3}$$

Notice that these definitions imply

$$\mathcal{T}.msw \leq \mathcal{T}.mcw \leq \mathcal{T}.mcp. \tag{4}$$

Other properties. We hereafter assume the following.

P2: $\mathcal{T}.w \leq 1$

P3: $\mathcal{T}.\hat{w} \geq \mathcal{T}.w$

P4: $N > 1$

(P2) and (P3) are necessary to ensure the feasibility of τ. (P4) eliminates the trivial case of a single component task. (Such a component task can be scheduled as an ordinary Pfair task.)

Other notational conventions. We use I and I' to denote arbitrary intervals and $|I|$ to denote the number of time slots in I. In addition, we use x, y, and z to denote real values, and c, i, k, k', n, L, L', and ℓ to denote integer values. Unless otherwise stated, these terms are assumed to be universally quantified.

3 Reweighting with EPDF-Scheduled Component Tasks

In this section, we present our rules for reweighting a supertask to ensure EPDF component-task schedulability. We present two approaches for selecting a scheduling weight. The first results in a smaller inflation factor but has exponential time complexity. Specifically, this approach produces a scheduling weight by iterating over a set of at most $\frac{\mathcal{T}.e}{\mathsf{gcd}(\mathcal{T}.e,\mathcal{T}.p)} + 1$ values. Since reweighting generally will be an off-line process, time complexity is not expected to be an issue. Moreover, since $\frac{\mathcal{T}.e}{\mathsf{gcd}(\mathcal{T}.e,\mathcal{T}.p)} \leq \frac{\mathcal{T}.p}{\mathsf{gcd}(\mathcal{T}.e,\mathcal{T}.p)}$, time complexity is pseudo-polynomial whenever the hyperperiod of the component task set τ (which defines the period of $\mathcal{T}$) is pseudo-polynomial. For cases in which exponential time complexity is a problem, we provide a second approach that usually results in a larger inflation factor, but has time complexity that is linear in the number of component tasks.

3.1 EPDF Reweighting Rules

Our reweighting rules are generalized to allow deadline overshoots. The allowable overshoot is specified by a non-negative integer parameter c. Our first reweighting rule, stated below, prevents supertasks with unit weight from being reweighted. In this case, reweighting is unnecessary because EPDF scheduling is optimal on uniprocessors [3].

Rule 1: If $\mathcal{T}.w = 1$, then let $\mathcal{T}.\hat{w} = 1$.

As shown later, if a supertask's weight is not inflated, then no component task can miss a deadline by $\mathcal{T}.msw$ or more. From this fact, we have the following rule.

Rule 2: If $c \geq \mathcal{T}.msw$, then let $\mathcal{T}.\hat{w} = \mathcal{T}.w$.

The remaining rules, Rule 3A and Rule 3B, are applicable if $\mathcal{T}.w < 1$ and $c < \mathcal{T}.msw$. Rule 3A calculates a scheduling weight in exponential time. On the other hand, Rule 3B takes linear time, but may produce a larger scheduling weight. Before stating these rules, we first introduce the following functions.

$$\Delta(c, w, L) = \frac{1 + \lfloor w \cdot L \rfloor}{L + c} \tag{5}$$

$$\phi(c, w, L) = \frac{1 + w \cdot L}{L + c} \tag{6}$$

$$\mathcal{L}(V, L) = \left\lceil L \cdot \frac{\mathsf{gcd}(V.e, V.p)}{V.p} \right\rceil \cdot \frac{V.p}{\mathsf{gcd}\,(V.e, V.p)} \tag{7}$$

Rule 3A calculates a scheduling weight by considering intervals of varying lengths. The set of interval lengths to consider is called the *testing set*. The function $\Delta(c, w, L)$ gives the candidate scheduling weight that results when considering an interval of length L. The function $\phi(c, w, L)$ upper bounds $\Delta(c, w, L)$, due to the floor expression in the numerator of $\Delta(c, w, L)$. $\phi(c, w, L)$ is used both for reducing the size of the testing set in Rule 3A and for reducing the time complexity of reweighting in Rule 3B. Finally, $\mathcal{L}(V, L)$ calculates the smallest $L' \geq L$ that satisfies $\Delta(c, V.w, L') = \phi(c, V.w, L')$. These functions will be described in more detail later.

Rule 3A (EPDF-Exponential): If no other rules apply, then let

$$\mathcal{T}.\hat{w} = \mathsf{max}\left\{\{\Delta(c, \mathcal{T}.w, \mathcal{T}.mcw)\} \cup \left\{ \Delta\left(c, \mathcal{T}.w, \left\lceil \frac{k}{\mathcal{T}.w} \right\rceil\right) \;\middle|\; \mathcal{T}.w \cdot \mathcal{L}(\mathcal{T}, \mathcal{T}.mcw) \geq k > \lfloor \mathcal{T}.w \cdot \mathcal{T}.mcw \rfloor \right\}\right\}.$$

Rule 3B (EPDF-Linear): If no other rules apply, then let

$$\mathcal{T}.\hat{w} = \mathsf{min}\left\{\phi(c, \mathcal{T}.w, \mathcal{T}.mcw), \frac{2}{\mathcal{T}.msw}\right\}.$$

Example, revisited (again). To demonstrate these rules, we reconsider the task set in Fig. 1. Suppose $c = 0$, *i.e.*, no deadline misses are allowed. Notice that neither Rule 1 nor Rule 2 applies to **T1**. We will calculate the inflation factor resulting from both Rule 3A and Rule 3B. To apply these rules, we need to know $\mathbf{T1}.mcw$ and $\mathbf{T1}.msw$. By (1), $\mathbf{T1}.mcw = \mathsf{min}\left\{\lceil\frac{5}{1}\rceil, \lceil\frac{45}{1}\rceil\right\} = 5$, and by (3), $\mathbf{T1}.msw = \lceil\frac{9}{2}\rceil = 5$.

Consider Rule 3A. For the first subset in the max expression, we have $\Delta(c, \mathbf{T1}.w, \mathbf{T1}.mcw) = \Delta(0, \frac{2}{9}, 5) = \frac{1+\lfloor\frac{2}{9}\cdot 5\rfloor}{5+0} = \frac{1+\lfloor\frac{10}{9}\rfloor}{5} = \frac{2}{5}$. As for the remaining subset, we first evaluate the bounds for k. For the lower bound, $\lfloor\mathbf{T1}.w \cdot \mathbf{T1}.mcw\rfloor = \lfloor\frac{2}{9}\cdot 5\rfloor = 1$. Since $\mathsf{gcd}(\mathbf{T1}.e, \mathbf{T1}.p) = 1$, the upper bound yields $\mathbf{T1}.w \cdot \mathcal{L}(\mathbf{T1}, \mathbf{T1}.mcw) = \frac{2}{9}\cdot\mathcal{L}(\mathbf{T1}, 5) = \frac{2}{9}\cdot\left(\lceil 5\cdot\frac{1}{9}\rceil\cdot\frac{9}{1}\right) = \frac{2}{9}\cdot 9 = 2$. (As explained later, there is no need to actually evaluate this upper bound because there is a simpler method for detecting when the largest value of k has been reached.) Therefore, the range of k in the second subset is $2 \geq k > 1$, which implies that only $k = 2$ must be considered. For this case, we have $\Delta\left(c, \mathbf{T1}.w, \left\lceil\frac{k}{\mathbf{T1}.w}\right\rceil\right) = \Delta\left(0, \frac{2}{9}, \lceil 2\cdot\frac{9}{2}\rceil\right) = \Delta\left(0, \frac{2}{9}, 9\right) = \frac{1+\lfloor\frac{2}{9}\cdot 9\rfloor}{9+0} = \frac{1+2}{9} = \frac{1}{3}$. Therefore, Rule 3A selects $\mathbf{T1}.\hat{w} = \mathsf{max}\left\{\frac{2}{5}, \frac{1}{3}\right\} = \frac{2}{5}$. Thus, the inflation factor is $\frac{2}{5} - \frac{2}{9} = \frac{8}{45}$.

Now, consider Rule 3B. For the first subexpression in this rule, we have $\phi(c, \mathbf{T1}.w, \mathbf{T1}.mcw) = \phi(0, \frac{2}{9}, 5) = \frac{1+\frac{2}{9}\cdot 5}{5+0} = \frac{\frac{19}{9}}{5} = \frac{19}{45}$. Since $\mathbf{T1}.msw = 5$, the second subexpression evaluates to $\frac{2}{5}$. Therefore, Rule 3B selects $\mathbf{T1}.\hat{w} = \mathsf{min}\left\{\frac{19}{45}, \frac{2}{5}\right\} = \frac{2}{5}$ as the new weight, which is the same as Rule 3A. As shown later, this happens because this example represents a worst-case scenario for a supertask with an actual weight of $\frac{2}{9}$. In general, these rules produce different inflations (as shown in the experimental results presented later).

3.2 Correctness of Reweighting

We prove that our rules are correct in four steps. First, we derive necessary conditions for a component subtask deadline to be missed in a schedule that satisfies (P1). Second, we show that with a scheduling weight of at least $\mathsf{max}\{\,\Delta(c, \mathcal{T}.w, L) \mid L \geq \mathcal{T}.mcw\}$, no component task deadline is missed in any schedule that satisfies (P1). Third, we bound the testing set of $\mathsf{max}\{\,\Delta(c, \mathcal{T}.w, L) \mid L \geq \mathcal{T}.mcw\}$ to produce Rule 3A. Fourth, we show that both subexpressions of Rule 3B upper bound the scheduling weight calculated in Rule 3A.

To simplify the statement of the theorems that follow, we define the supertask $\mathcal{T}$ to be ($\mathcal{S}$, c)*-EPDF-safe* iff EPDF schedules τ, the set of component tasks of $\mathcal{T}$, within $\mathcal{T}$'s allocation in $\mathcal{S}$ such that no pseudo-deadlines of tasks in τ are missed by more than c time slots. The schedule $\mathcal{S}$ is omitted if *every* schedule satisfying (P1) satisfies this condition.

3.2.1 Step 1: Necessary Condition for Component Subtask Deadline Misses

We begin by stating two simple lemmas, which are proved in an appendix.

Lemma 1 (Number of subtask windows in an interval): For any Pfair-scheduled task V and any interval $I = [t_0, t_0 + L)$, $\lfloor V.\hat{w}\cdot L\rfloor - 1 \leq \mathit{windows}(V, I) \leq \lfloor V.\hat{w}\cdot L\rfloor$.

Lemma 2 (Pfair task allocation over an interval): For any Pfair schedule $\mathcal{S}$ containing task V and any interval I, $\mathit{allocation}(V, I, \mathcal{S}) \geq \mathit{windows}(V, I)$.

In Theorem 1 below, we show that a component subtask's deadline can be missed only if there exists an interval I such that the number of component-task windows contained in I exceeds the number of quanta allocated to the supertask in I. (Recall that the interval [5,9] in Fig. 1(a) is such an interval.) Theorem 1 also shows that only one additional quantum is needed to satisfy the aggregate demand of all subtasks in I. The theorem is actually stated in a more general way in which the overshoot parameter c is taken into account. From Theorem 1, we derive a sufficient schedulability condition for a given schedule $\mathcal{S}$ in Corollary 1. We then use this condition in Corollary 2 to show that Rule 2 is correct.

Theorem 1 (EPDF underallocated interval): For any EPDF-scheduled component task T of a supertask $\mathcal{T}$, integer c satisfying $c \geq 0$, and schedule $\mathcal{S}$ satisfying (P1), a subtask T_i misses its deadline by more than c time units only if there exists intervals $I = [d(T_i) - L + 1, d(T_i)]$ and $I' = [d(T_i) - L + 1, d(T_i) + c]$, where $L \geq |\omega(T_i)|$, satisfying the following conditions.

(i) $\mathit{allocation}(\mathcal{T}, I', \mathcal{S}) = \mathit{windows}(\mathcal{T}, I') = \lfloor \mathcal{T}.w \cdot L \rfloor - 1$

(ii) $\sum_{U \in \mathcal{T}} \mathit{windows}(U, I) = \lfloor \mathcal{T}.w \cdot L \rfloor$

Proof: Assume that some subtask T_i misses its deadline by more than c time units. We prove the theorem by applying a uniprocessor *idle instant* argument. A (*subtask*) *t-idle instant* of supertask $\mathcal{T}$ occurs at $t_0 \leq t$ in $\mathcal{S}$ iff all subtasks of tasks in τ that execute in the interval $[t_0, t]$ have releases at or after t_0. Note that at least one t-idle instant exists since $t_0 = 0$ is a t-idle instant for any t.

Let t_0 be the latest $(d(T_i) + c)$-idle instant of $\mathcal{T}$ at or before $r(T_i)$. For the remainder of the proof, let $L = d(T_i) - t_0 + 1$, $I = [t_0, d(T_i)]$, and $I' = [t_0, d(T_i) + c]$. (Notice that $L \geq |\omega(T_i)|$.) All quanta allocated to $\mathcal{T}$ in I' are used by some component task since an idle quantum at $t \in [r(T_i), d(T_i) + c]$ would be allocated to T_i and an idle quantum at $t \in [t_0, r(T_i))$ would imply $t + 1$ is a later idle instant than t_0. Similarly, no subtasks with deadlines after $d(T_i)$ execute in I' since T_i has higher priority for $t \in [r(T_i), d(T_i) + c]$ and the execution of such a subtask at $t \in [t_0, r(T_i))$ would again contradict the choice of t_0. Therefore, all quanta allocated to $\mathcal{T}$ in I' are consumed by subtasks with windows that are properly contained in I. Since T_i misses its deadline by more than c time units, the following relationship exists.

$$\mathit{allocation}(\mathcal{T}, I', \mathcal{S}) < \sum_{U \in \mathcal{T}} \mathit{windows}(U, I) \tag{8}$$

Applying Lemma 1 yields the following upper bound on the right-hand side of (8).

Claim 1: $\sum_{U \in \mathcal{T}} \mathit{windows}(U, I) \leq \lfloor \mathcal{T}.w \cdot L \rfloor$

Proof of Claim:

$$\begin{aligned}
\sum_{U \in \mathcal{T}} \mathit{windows}(U, I) &\leq \sum_{U \in \mathcal{T}} \lfloor U.\hat{w} \cdot L \rfloor && \text{, by Lemma 1} \\
&\leq \left\lfloor \Big(\sum_{U \in \mathcal{T}} U.\hat{w} \Big) \cdot L \right\rfloor && \text{, } \lfloor x + y \rfloor \geq \lfloor x \rfloor + \lfloor y \rfloor^{\dagger} \\
&= \lfloor \mathcal{T}.w \cdot L \rfloor && \text{, definition of } \mathcal{T}.w
\end{aligned}$$

□

†In these justifications, we will use x, y, and z to denote real values and k and n to denote integer values.

Combining (8) and Claim 1, we get

$$allocation(\mathcal{T}, I', \mathcal{S}) < \lfloor \mathcal{T}.w \cdot L \rfloor. \tag{9}$$

Furthermore, the following expressions hold, by Lemma 2, Lemma 1, (P3), and $c \geq 0$, respectively.

$$allocation(\mathcal{T}, I', \mathcal{S}) \geq windows(\mathcal{T}, I') \tag{10}$$

$$windows(\mathcal{T}, I') \geq \lfloor \mathcal{T}.\hat{w} \cdot (L + c) \rfloor - 1 \tag{11}$$

$$\lfloor \mathcal{T}.\hat{w} \cdot (L + c) \rfloor - 1 \geq \lfloor \mathcal{T}.w \cdot (L + c) \rfloor - 1 \tag{12}$$

$$\lfloor \mathcal{T}.w \cdot (L + c) \rfloor - 1 \geq \lfloor \mathcal{T}.w \cdot L \rfloor - 1 \tag{13}$$

Expressions (9)–(13) yield Condition (i) of the theorem. Furthermore, by (8) and Condition (i), we have $\sum_{U \in \mathcal{T}} windows(U, I) > \lfloor \mathcal{T}.w \cdot L \rfloor - 1$, which implies that Condition (ii) holds, by Claim 1. □

Corollary 1 (Sufficient schedulability condition for EPDF component tasks): For any supertask $\mathcal{T}$ with EPDF-scheduled component tasks, schedule $\mathcal{S}$ satisfying (P1), and integer c satisfying $c \geq 0$, if $allocation(\mathcal{T}, I, \mathcal{S}) \geq \lfloor \mathcal{T}.w \cdot L \rfloor$ for all I and L where $|I| = L + c$ and $L \geq \mathcal{T}.mcw$, then $\mathcal{T}$ is ($\mathcal{S}$, c)-EPDF-safe.

Proof: By (1), $\mathcal{T}.mcw$ lower bounds all window lengths of tasks in $\mathcal{T}$. Therefore, the stated condition prevents Condition (i) of Theorem 1 from holding. □

Corollary 2 (Correctness of Rule 2): For any supertask $\mathcal{T}$ with EPDF-scheduled component tasks, if $\mathcal{T}.\hat{w} = \mathcal{T}.w$ then $\mathcal{T}$ is c-EPDF-safe for all integers $c \geq \mathcal{T}.msw$.

Proof: Consider any interval I where $|I| = L + c$ for some $L \geq 0$ and $c \geq \mathcal{T}.msw$. By Lemmas 1 and 2, $allocation(\mathcal{T}, I, \mathcal{S}) \geq \lfloor \mathcal{T}.w \cdot (L + c) \rfloor - 1$ holds for any schedule $\mathcal{S}$ satisfying (P1). Since $c \geq \mathcal{T}.msw$, it follows from (3) that $\mathcal{T}.w \cdot c \geq \mathcal{T}.w \cdot \left\lceil \frac{1}{\mathcal{T}.w} \right\rceil \geq 1$, which implies that $allocation(\mathcal{T}, I, \mathcal{S}) \geq \lfloor \mathcal{T}.w \cdot L + 1 \rfloor - 1 = \lfloor \mathcal{T}.w \cdot L \rfloor$. Therefore, the stated condition prevents Condition (i) of Theorem 1 from holding in *any* $\mathcal{S}$ satisfying (P1). □

3.2.2 Step 2: Selecting the New Weight

We now calculate the minimum scheduling weight necessary to ensure that the condition of Corollary 1 is satisfied in *any* schedule satisfying (P1). According to this condition,

$$allocation(\mathcal{T}, I, \mathcal{S}) \geq \lfloor \mathcal{T}.w \cdot L \rfloor$$

must hold for all I and L where $|I| = L + c$ and $L \geq \mathcal{T}.mcw$. We refer to $\mathcal{T}.mcw$ as the *critical interval length*. To select a scheduling weight, we first calculate the minimum scheduling weight needed to ensure that the above condition holds for a specific L (Lemma 3). The scheduling weight can then be determined by considering all intervals of length at least the critical interval length (Theorem 2).

Lemma 3 (Reweighting for a specific interval length): For any supertask $\mathcal{T}$, schedule $\mathcal{S}$ satisfying (P1), integer c satisfying $\mathcal{T}.msw > c \geq 0$, and integer L satisfying $L > 0$, if $\mathcal{T}.\hat{w} \geq \Delta(c, \mathcal{T}.w, L)$ then $allocation(\mathcal{T}, I, \mathcal{S}) \geq \lfloor \mathcal{T}.w \cdot L \rfloor$ for all I where $|I| = L + c$.

Proof: Assume to the contrary that $\mathcal{T}.\hat{w} \geq \Delta(c, \mathcal{T}.w, L)$, but for some I where $|I| = L + c$, $allocation(\mathcal{T}, I, \mathcal{S}) < \lfloor \mathcal{T}.w \cdot L \rfloor$ holds. By Lemmas 1 and 2, we get $allocation(\mathcal{T}, I, \mathcal{S}) \geq \lfloor \mathcal{T}.\hat{w} \cdot (L + c) \rfloor - 1$. Thus,

$$\lfloor \mathcal{T}.\hat{w} \cdot (L + c) \rfloor - 1 < \lfloor \mathcal{T}.w \cdot L \rfloor,$$

which leads to the following contradiction.

$$\begin{aligned}
&\lfloor \mathcal{T}.\hat{w} \cdot (L + c) \rfloor - 1 < \lfloor \mathcal{T}.w \cdot L \rfloor \\
&\quad \Rightarrow \mathcal{T}.\hat{w} \cdot (L + c) < 1 + \lfloor \mathcal{T}.w \cdot L \rfloor && , \lfloor x \rfloor < n \Rightarrow x < n \\
&\quad \Rightarrow \mathcal{T}.\hat{w} < \tfrac{1 + \lfloor \mathcal{T}.w \cdot L \rfloor}{L + c} && \text{, division by } (L + c) \\
&\quad \Rightarrow \mathcal{T}.\hat{w} < \Delta(c, \mathcal{T}.w, L) && \text{, by (5)}
\end{aligned}$$

□

Theorem 2 (Reweighting for schedulability): For any supertask $\mathcal{T}$ and integer c satisfying $\mathcal{T}.msw > c \geq 0$, if $\mathcal{T}.\hat{w} \geq \mathsf{max}\{\, \Delta(c, \mathcal{T}.w, L) \mid L \geq \mathcal{T}.mcw \}$ then $\mathcal{T}$ is c-EPDF-safe.

Proof: Consider the following derivation.

$$\begin{aligned}
&\mathcal{T}.\hat{w} \geq \mathsf{max}\{\, \Delta(c, \mathcal{T}.w, L) \mid L \geq \mathcal{T}.mcw \} \\
&\quad \Rightarrow (\forall L : L \geq \mathcal{T}.mcw : \mathcal{T}.\hat{w} \geq \Delta(c, \mathcal{T}.w, L)) \qquad \text{, transitivity of } \geq \\
&\quad \Rightarrow (\forall I, L, \mathcal{S} : L \geq \mathcal{T}.mcw \wedge |I| = L + c \wedge (\mathcal{S} \text{ satisfies (P1)}) : allocation(\mathcal{T}, I, \mathcal{S}) \geq \lfloor \mathcal{T}.w \cdot L \rfloor) \\
&\qquad\qquad \text{, by Lemma 3}
\end{aligned}$$

From this derivation and Corollary 1, it follows that $\mathcal{T}$ is c-EPDF-safe. □

3.2.3 Step 3: Bounding the Testing Set

We now provide some insights on the functions defined earlier in (5)–(7). Recall that $\Delta(c, \mathcal{T}.w, L)$ is the reweighting function that first appeared in Lemma 3. Since $L > 0$ and all other parameters are non-negative,

$$\phi(c, \mathcal{T}.w, L) = \frac{1 + \mathcal{T}.w \cdot L}{L + c} \geq \frac{1 + \lfloor \mathcal{T}.w \cdot L \rfloor}{L + c} = \Delta(c, \mathcal{T}.w, L). \qquad (14)$$

Thus, $\phi(c, \mathcal{T}.w, L)$ is a *bounding function* of $\Delta(c, \mathcal{T}.w, L)$. Furthermore, these functions are equal iff $\mathcal{T}.w \cdot L$ is an integer. In this case, we say that the interval length L is *proper*. (Thus, $\Delta(c, \mathcal{T}.w, L) = \phi(c, \mathcal{T}.w, L)$ iff L is proper.) Notice also that $\mathcal{T}.w \cdot L$ is an integer iff L is a multiple of $\frac{\mathcal{T}.p}{\mathsf{gcd}(\mathcal{T}.e, \mathcal{T}.p)}$. By definition, $\mathcal{L}(\mathcal{T}, L)$ produces the minimum such multiple that is at least L.

We now use ϕ to restrict the testing set size in $\max\{\Delta(c, \mathcal{T}.w, L) \mid L \geq \mathcal{T}.mcw\}$. We first show that ϕ monotonically decreases as L increases (Lemma 4). Next, we show that proper interval lengths maximize Δ for values of L over a specific range (Corollary 3). We then use this fact to remove all interval lengths longer than $\mathcal{L}(\mathcal{T}, \mathcal{T}.mcw)$ from the testing set (Corollary 4). Finally, we reduce the size of the testing set further by observing that, when $\lfloor \mathcal{T}.w \cdot L \rfloor$ remains constant, Δ monotonically decreases as L increases (Theorem 3). After presenting these results, we derive the worst-case size of the testing set for Rule 3A (which determines its time complexity).

Lemma 4 (Monotonicity of ϕ): For any supertask $\mathcal{T}$, integer c satisfying $\mathcal{T}.msw > c \geq 0$, and positive integers L and ℓ, $\phi(c, \mathcal{T}.w, L + \ell) \leq \phi(c, \mathcal{T}.w, L)$.

Proof: Assume to the contrary that there exists some integer $L + \ell$ where $\ell > 0$ such that $\phi(c, \mathcal{T}.w, L + \ell) > \phi(c, \mathcal{T}.w, L)$. This assumption leads to a contradiction as follows.

$$
\begin{array}{ll}
\phi(c, \mathcal{T}.w, L + \ell) > \phi(c, \mathcal{T}.w, L) & \\
\quad \Rightarrow \frac{1+\mathcal{T}.w\cdot(L+\ell)}{L+\ell+c} > \frac{1+\mathcal{T}.w\cdot L}{L+c} & \text{, by (6)} \\
\quad \Rightarrow \frac{(1+\mathcal{T}.w\cdot L)+\mathcal{T}.w\cdot\ell}{(L+c)+\ell} > \frac{1+\mathcal{T}.w\cdot L}{L+c} & \text{, rearranging} \\
\quad \Rightarrow (L+c)\cdot[(1+\mathcal{T}.w\cdot L)+\mathcal{T}.w\cdot\ell] > [(L+c)+\ell]\cdot(1+\mathcal{T}.w\cdot L) & \text{, cross multiplication} \\
\quad \Rightarrow (L+c)\cdot\mathcal{T}.w\cdot\ell > \ell\cdot(1+\mathcal{T}.w\cdot L) & \text{, simplification} \\
\quad \Rightarrow c\cdot\mathcal{T}.w\cdot\ell > \ell & \text{, simplification} \\
\quad \Rightarrow c > \frac{1}{\mathcal{T}.w} & \text{, division by } \mathcal{T}.w\cdot\ell \\
\quad \Rightarrow c \geq \left\lfloor \frac{1}{\mathcal{T}.w} \right\rfloor + 1 & , n > x \Rightarrow n \geq \lfloor x \rfloor + 1 \\
\quad \Rightarrow c \geq \left\lceil \frac{1}{\mathcal{T}.w} \right\rceil & , \lfloor x \rfloor + 1 \geq \lceil x \rceil \\
\quad \Rightarrow c \geq \mathcal{T}.msw & \text{, by (3)}
\end{array}
$$

□

Corollary 3 (Maximality of proper interval lengths): For any supertask $\mathcal{T}$, integer c satisfying $\mathcal{T}.msw > c \geq 0$, and integer $k > 0$, $\Delta\left(c, \mathcal{T}.w, k \cdot \frac{\mathcal{T}.p}{\mathsf{gcd}(\mathcal{T}.e, \mathcal{T}.p)}\right) = \max\left\{\Delta(c, \mathcal{T}.w, L) \mid L \geq k \cdot \frac{\mathcal{T}.p}{\mathsf{gcd}(\mathcal{T}.e, \mathcal{T}.p)}\right\}$.

Proof: Let $L' = k \cdot \frac{\mathcal{T}.p}{\mathsf{gcd}(\mathcal{T}.e, \mathcal{T}.p)}$. By definition, L' is a proper interval length, which implies that $\Delta(c, \mathcal{T}.w, L') = \phi(c, \mathcal{T}.w, L')$. Therefore, by Lemma 4, $\Delta(c, \mathcal{T}.w, L') = \max\{\phi(c, \mathcal{T}.w, L) \mid L \geq L'\}$. Furthermore, (14) implies that $\Delta(c, \mathcal{T}.w, L') = \max\{\Delta(c, \mathcal{T}.w, L) \mid L \geq L'\}$. □

Corollary 4 (Making the testing set finite): For any supertask $\mathcal{T}$ and integer c satisfying $\mathcal{T}.msw > c \geq 0$, if $\mathcal{T}.\hat{w} \geq \max\{\Delta(c, \mathcal{T}.w, L) \mid \mathcal{L}(\mathcal{T}, \mathcal{T}.mcw) \geq L \geq \mathcal{T}.mcw\}$ then $\mathcal{T}$ is c-EPDF-safe.

Proof: This is a direct application of Corollary 3 to the result of Theorem 2. Recall that $\mathcal{L}(\mathcal{T}, \mathcal{T}.mcw)$ is the next multiple of $\frac{\mathcal{T}.p}{\mathsf{gcd}(\mathcal{T}.e, \mathcal{T}.p)}$ at or after $\mathcal{T}.mcw$. In addition, by Corollary 3, $\Delta(c, \mathcal{T}.w, \mathcal{L}(\mathcal{T}, \mathcal{T}.mcw)) = \max\{\Delta(c, \mathcal{T}.w, L) \mid L \geq \mathcal{L}(\mathcal{T}, \mathcal{T}.mcw)\}$. Therefore, we can remove all $L > \mathcal{L}(\mathcal{T}, \mathcal{T}.mcw)$ from the testing set. □

Theorem 3 (Correctness of Rule 3A): For any supertask $\mathcal{T}$ and integer c satisfying $\mathcal{T}.msw > c \geq 0$, if $\mathcal{T}.\hat{w} \geq \max\left\{\{\Delta(c, \mathcal{T}.w, \mathcal{T}.mcw)\} \cup \left\{\Delta\left(c, \mathcal{T}.w, \left\lceil \frac{k}{\mathcal{T}.w} \right\rceil\right) \;\middle|\; \mathcal{T}.w \cdot \mathcal{L}(\mathcal{T}, \mathcal{T}.mcw) \geq k > \lfloor \mathcal{T}.w \cdot \mathcal{T}.mcw \rfloor\right\}\right\}$ then $\mathcal{T}$ is c-EPDF-safe.

Proof: Observe that, when $\lfloor \mathcal{T}.w \cdot L \rfloor$ remains constant, $\Delta(c, \mathcal{T}.w, L) = \frac{1+\lfloor \mathcal{T}.w \cdot L \rfloor}{L+c}$ monotonically decreases with increasing L. Therefore, when evaluating the testing set in Corollary 4, only the first value of L associated with each distinct value of $\lfloor \mathcal{T}.w \cdot L \rfloor$ must be evaluated. We now show that the first value of L satisfying $\lfloor \mathcal{T}.w \cdot L \rfloor = k$ is $\left\lceil \frac{k}{\mathcal{T}.w} \right\rceil$.

$$
\begin{aligned}
&(\lfloor \mathcal{T}.w \cdot L \rfloor = k) \\
&\quad = (k+1 > \mathcal{T}.w \cdot L \geq k) && \text{, definition of } \lfloor x \rfloor \\
&\quad = \left(\frac{k+1}{\mathcal{T}.w} > L \geq \frac{k}{\mathcal{T}.w}\right) && \text{, division by } \mathcal{T}.w \\
&\quad \Rightarrow \left(\left\lceil \frac{k+1}{\mathcal{T}.w} \right\rceil > L \geq \left\lceil \frac{k}{\mathcal{T}.w} \right\rceil\right) && \text{, } x > n \geq y \Rightarrow \lceil x \rceil > n \geq \lceil y \rceil
\end{aligned}
$$

Furthermore, since $k = \lfloor \mathcal{T}.w \cdot L \rfloor$, Corollary 4 implies that the largest value of k that must be checked is $\lfloor \mathcal{T}.w \cdot \mathcal{L}(\mathcal{T}, \mathcal{T}.mcw) \rfloor = \mathcal{T}.w \cdot \mathcal{L}(\mathcal{T}, \mathcal{T}.mcw)$. In addition, Corollary 4 also implies that $L \geq \mathcal{T}.mcw$. Therefore, $k \geq \lfloor \mathcal{T}.w \cdot \mathcal{T}.mcw \rfloor$. However, this first value of k should be handled separately, which is the reason for the $\Delta(c, \mathcal{T}.w, \mathcal{T}.mcw)$ term in Rule 3A. Note that if we substitute $k = \lfloor \mathcal{T}.w \cdot \mathcal{T}.mcw \rfloor$ within $\Delta\left(c, \mathcal{T}.w, \left\lceil \frac{k}{\mathcal{T}.w} \right\rceil\right)$, then the third parameter becomes $\left\lceil \frac{\lfloor \mathcal{T}.w \cdot \mathcal{T}.mcw \rfloor}{\mathcal{T}.w} \right\rceil$, which could be *smaller* than $\mathcal{T}.mcw$. Therefore, only values of k *larger* than $\lfloor \mathcal{T}.w \cdot \mathcal{T}.mcw \rfloor$ should be considered when evaluating $\Delta\left(c, \mathcal{T}.w, \left\lceil \frac{k}{\mathcal{T}.w} \right\rceil\right)$.

Combining these arguments, we only need to evaluate $\Delta\left(c, \mathcal{T}.w, \left\lceil \frac{k}{\mathcal{T}.w} \right\rceil\right)$ for $\mathcal{T}.w \cdot \mathcal{L}(\mathcal{T}, \mathcal{T}.mcw) \geq k > \lfloor \mathcal{T}.w \cdot \mathcal{T}.mcw \rfloor$ along with $\Delta(c, \mathcal{T}.w, \mathcal{T}.mcw)$ to determine the value of the reweighting expression shown in Corollary 4. $\square$

Observe that $\mathcal{L}(\mathcal{T}, \mathcal{T}.mcw)$ is the only proper interval length within the testing set of Rule 3A. This implies that the largest value of k to consider can be determined implicitly by checking whether or not its associated interval length, $\left\lceil \frac{k}{\mathcal{T}.w} \right\rceil$, is proper. If it is proper, then $\left\lceil \frac{k}{\mathcal{T}.w} \right\rceil = \mathcal{L}(\mathcal{T}, \mathcal{T}.mcw)$ and no further evaluation is needed. Due to this, Rule 3A can be applied without ever having to evaluate $\mathcal{L}(\mathcal{T}, \mathcal{T}.mcw)$.

Deriving the worst-case testing set size. We now show that the testing set of the expression in Theorem 3 contains at most $\frac{\mathcal{T}.e}{\mathsf{gcd}(\mathcal{T}.e, \mathcal{T}.p)} + 1$ interval lengths. First, consider the size of the testing set in Corollary 4, which tests all L in the range $\mathcal{L}(\mathcal{T}, \mathcal{T}.mcw) \geq L \geq \mathcal{T}.mcw$. As explained above, this range contains only one proper interval length, $\mathcal{L}(\mathcal{T}, \mathcal{T}.mcw)$. This length must be a multiple of $\frac{\mathcal{T}.p}{\mathsf{gcd}(\mathcal{T}.e, \mathcal{T}.p)}$, so it can be written as $\mathcal{L}(\mathcal{T}, \mathcal{T}.mcw) = k \cdot \frac{\mathcal{T}.p}{\mathsf{gcd}(\mathcal{T}.e, \mathcal{T}.p)}$ for some $k > 0$. Because $\mathcal{L}(\mathcal{T}, \mathcal{T}.mcw)$ is the only proper interval length in the testing set, $\mathcal{T}.mcw > (k-1) \cdot \frac{\mathcal{T}.p}{\mathsf{gcd}(\mathcal{T}.e, \mathcal{T}.p)}$. Note that the testing set includes one value for each unique value of $\lfloor \mathcal{T}.w \cdot L \rfloor$, where L ranges over $\mathcal{L}(\mathcal{T}, \mathcal{T}.mcw) \geq L \geq \mathcal{T}.mcw$. Since $\lfloor \mathcal{T}.w \cdot L \rfloor$ monotonically increases with increasing L, the number of distinct values in the testing set is no more than $\left\lfloor \mathcal{T}.w \cdot k \cdot \frac{\mathcal{T}.p}{\mathsf{gcd}(\mathcal{T}.e, \mathcal{T}.p)} \right\rfloor -$

$$\left\lfloor \mathcal{T}.w \cdot (k-1) \cdot \frac{\mathcal{T}.p}{\mathsf{gcd}(\mathcal{T}.e, \mathcal{T}.p)} \right\rfloor + 1 = \frac{\mathcal{T}.e}{\mathsf{gcd}(\mathcal{T}.e, \mathcal{T}.p)} + 1.$$

3.2.4 Step 4: Safety of Quick Reweighting

We now show that both of Rule 3B's subexpressions, $\phi(c, \mathcal{T}.w, \mathcal{T}.mcw)$ and $\frac{2}{\mathcal{T}.msw}$, produce supertask weights that are sufficient to ensure EPDF component-task schedulability. In Lemma 5, we consider $\phi(c, \mathcal{T}.w, \mathcal{T}.mcw)$, followed by $\frac{2}{\mathcal{T}.msw}$ in Lemma 6 and Theorem 4. Notice that $\frac{2}{\mathcal{T}.msw}$ is not a function of c. In fact, $\frac{2}{\mathcal{T}.msw}$ is the worst-case reweighting for $c \geq 0$. Although this suggests that $\phi(c, \mathcal{T}.w, \mathcal{T}.mcw)$ will usually provide a smaller scheduling weight, $\frac{2}{\mathcal{T}.msw}$ guarantees that $\mathcal{T}.\hat{w} \leq 1$ since $\mathcal{T}.w < 1 \Rightarrow \mathcal{T}.msw \geq 2$. (It *is* possible for $\phi(c, \mathcal{T}.w, \mathcal{T}.mcw)$ to exceed 1 when $c = 0$.)

Lemma 5 (Safety of Rule 3B's first subexpression): For any supertask $\mathcal{T}$ with EPDF-scheduled component tasks and integer c satisfying $\mathcal{T}.msw > c \geq 0$, if $\mathcal{T}.\hat{w} \geq \phi(c, \mathcal{T}.w, \mathcal{T}.mcw)$ then $\mathcal{T}$ is c-EPDF-safe.

Proof: This lemma follows directly from previous results. By Lemma 4, $\mathcal{T}.\hat{w} \geq \phi(c, \mathcal{T}.w, \mathcal{T}.mcw)$ implies that $\mathcal{T}.\hat{w} \geq \mathsf{max}\{\, \phi(c, \mathcal{T}.w, L) \mid L \geq \mathcal{T}.mcw\}$, which implies that $\mathcal{T}.\hat{w} \geq \mathsf{max}\{\, \Delta(c, \mathcal{T}.w, L) \mid L \geq \mathcal{T}.mcw\}$ by (14). Therefore, by Theorem 2, $\mathcal{T}$ is c-EPDF-safe. □

Lemma 6 (Worst-case reweighting): For any supertask $\mathcal{T}$ and integer c satisfying $c \geq 0$, if $\frac{1}{i} \leq \mathcal{T}.w < \frac{1}{i-1}$ for some integer $i \geq 2$, then, for any $L \geq i$, $\Delta(c, \mathcal{T}.w, L) \leq \frac{2}{i}$.

Proof: Assume that $\frac{1}{i} \leq \mathcal{T}.w < \frac{1}{i-1}$ holds for some integer $i \geq 2$. First, observe that $c \geq 0$ implies that $\Delta(0, w, L) = \frac{1+\lfloor w \cdot L \rfloor}{L} \geq \frac{1+\lfloor w \cdot L \rfloor}{L+c} = \Delta(c, w, L)$. Therefore, if the lemma holds for $c = 0$, then it holds for all $c \geq 0$. Assume that $L = i + \ell$ for some $\ell \geq 0$.

Claim 2: $\left(\Delta(0, \mathcal{T}.w, L) \leq \frac{2}{i}\right) = \left(\lfloor \mathcal{T}.w \cdot L \rfloor \leq 1 + \frac{2 \cdot \ell}{i}\right)$

Proof of Claim:

$$\begin{aligned}
&\left(\Delta(0, \mathcal{T}.w, L) \leq \tfrac{2}{i}\right) \\
&= \left(\frac{1+\lfloor \mathcal{T}.w \cdot L \rfloor}{L} \leq \frac{2}{i}\right) && \text{, by (5)} \\
&= \left(\lfloor \mathcal{T}.w \cdot L \rfloor \leq \frac{2 \cdot L}{i} - 1\right) && \text{, rearranging} \\
&= \left(\lfloor \mathcal{T}.w \cdot L \rfloor \leq \frac{2 \cdot (i+\ell)}{i} - 1\right) && \text{, definition of } L \\
&= \left(\lfloor \mathcal{T}.w \cdot L \rfloor \leq 1 + \frac{2 \cdot \ell}{i}\right) && \text{, simplification}
\end{aligned}$$

□

Using $\frac{1}{i} \leq \mathcal{T}.w < \frac{1}{i-1}$, we derive the following upper bound on $\lfloor \mathcal{T}.w \cdot L \rfloor$.

Claim 3: $\left(\mathcal{T}.w < \frac{1}{i-1}\right) \Rightarrow \left(\lfloor \mathcal{T}.w \cdot L \rfloor \leq \left\lceil \frac{\ell+1}{i-1} \right\rceil\right)$

Proof of Claim:

$$
\begin{array}{ll}
\mathcal{T}.w < \frac{1}{i-1} & \\
\quad \Rightarrow \mathcal{T}.w \cdot L < \frac{L}{i-1} & \text{, multiplication by } L \\
\quad \Rightarrow \mathcal{T}.w \cdot L < \frac{i+\ell}{i-1} & \text{, definition of } L \\
\quad \Rightarrow \mathcal{T}.w \cdot L < 1 + \frac{\ell+1}{i-1} & \text{, } i+\ell = (i-1)+(\ell+1) \\
\quad \Rightarrow \lfloor \mathcal{T}.w \cdot L \rfloor \leq \left\lceil \frac{\ell+1}{i-1} \right\rceil & \text{, } (x < y) \Rightarrow (\lfloor x \rfloor \leq \lceil y \rceil - 1)
\end{array}
$$

□

Now, let $\lfloor \mathcal{T}.w \cdot L \rfloor = k$. It follows from Claim 3 that $\left\lceil \frac{\ell+1}{i-1} \right\rceil \geq k$, which implies that $\frac{\ell+1}{i-1} > (k-1)$. Rearranging yields $\ell > (k-1)\cdot(i-1)-1$. Since both sides are integers, it follows that $\ell \geq (k-1)\cdot(i-1)$, which implies that $k \leq 1 + \frac{\ell}{i-1}$. Therefore, by Claim 2, the theorem holds if $1 + \frac{\ell}{i-1} \leq 1 + \frac{2\cdot\ell}{i}$ holds. Assume to the contrary that $\frac{\ell}{i-1} > \frac{2\cdot\ell}{i}$ holds for some $i \geq 2$. This leads to a contradiction as follows.

$$
\begin{array}{ll}
\frac{\ell}{i-1} > \frac{2\cdot\ell}{i} & \\
\quad \Rightarrow i \cdot \ell > (i-1) \cdot 2 \cdot \ell & \text{, cross multiplication} \\
\quad \Rightarrow i \cdot \ell < 2 \cdot \ell & \text{, simplification} \\
\quad \Rightarrow i < 2 & \text{, division by } \ell \text{ (note that } \frac{\ell}{i-1} > \frac{2\cdot\ell}{i} \Rightarrow \ell > 0)
\end{array}
$$

□

Theorem 4 (Simple EPDF reweighting): For any supertask $\mathcal{T}$ and integer c satisfying $c \geq 0$, if $0 < \mathcal{T}.w < 1$ and $\mathcal{T}.\hat{w} \geq \frac{2}{\mathcal{T}.msw}$, then $\mathcal{T}$ is c-EPDF-safe.

Proof: This theorem follows from previous results and a simple observation. Assume that $\frac{1}{i} \leq \mathcal{T}.w < \frac{1}{i-1}$ holds for some integer $i \geq 2$. (Such an i exists for any weight satisfying $0 < \mathcal{T}.w < 1$.) We now show that $\frac{1}{i} \leq \mathcal{T}.w < \frac{1}{i-1}$ implies $i = \mathcal{T}.msw$.

$$
\begin{array}{ll}
\frac{1}{i} \leq \mathcal{T}.w < \frac{1}{i-1} & \\
\quad \Rightarrow i \geq \frac{1}{\mathcal{T}.w} > i-1 & \text{, } (x \leq y < z) \Rightarrow (\frac{1}{x} \geq \frac{1}{y} > \frac{1}{z}) \\
\quad \Rightarrow \frac{1}{\mathcal{T}.w} \leq i < \frac{1}{\mathcal{T}.w} + 1 & \text{, rearranging} \\
\quad \Rightarrow \left\lceil \frac{1}{\mathcal{T}.w} \right\rceil \leq i \leq \left\lceil \frac{1}{\mathcal{T}.w} \right\rceil & \text{, } (x \leq n < y) \Rightarrow (\lceil x \rceil \leq n \leq \lceil y \rceil - 1) \\
\quad \Rightarrow i = \mathcal{T}.msw & \text{, by (3)}
\end{array}
$$

By Lemma 6,

$$\left(\forall L : L \geq \mathcal{T}.msw : \Delta(c, \mathcal{T}.w, L) \leq \frac{2}{\mathcal{T}.msw}\right),$$

which can also be expressed as

$$\max\{\, \Delta(c, \mathcal{T}.w, L) \mid L \geq \mathcal{T}.msw \} \leq \frac{2}{\mathcal{T}.msw}.$$

By (4), this implies that

$$\max\{\,\Delta(c, \mathcal{T}.w, L) \mid L \geq \mathcal{T}.mcw\} \leq \frac{2}{\mathcal{T}.msw}.$$

Therefore, by Theorem 2, if $\mathcal{T}.\hat{w} \geq \frac{2}{\mathcal{T}.msw}$, then $\mathcal{T}$ is c-EPDF-safe. □

4 Reweighting with EDF-Scheduled Component Tasks

In this section, we present reweighting rules for supertasks with EDF-scheduled component tasks. These results closely parallel those in Sec. 3.

4.1 EDF Reweighting Rules

By ignoring pseudo-deadlines that are properly-contained within a job, EDF scheduling increases the critical interval length from the smallest component task window ($\mathcal{T}.mcw$) to the smallest component task period ($\mathcal{T}.mcp$). In fact, this is the *only* change from the rules in Sec. 3. Rule 1 can be applied as previously defined since EDF scheduling is also optimal for uniprocessors. In addition, we will show that a variant of Theorem 1 from Sec. 3 applies to EDF-scheduled component tasks, which establishes the correctness of Rule 2 with either EPDF or EDF scheduling. Finally, EDF variants of Rules 3A and 3B can be obtained by simply replacing $\mathcal{T}.mcw$ with $\mathcal{T}.mcp$ as shown below.

Rule 3A (EDF-Exponential): If no other rules apply, then let

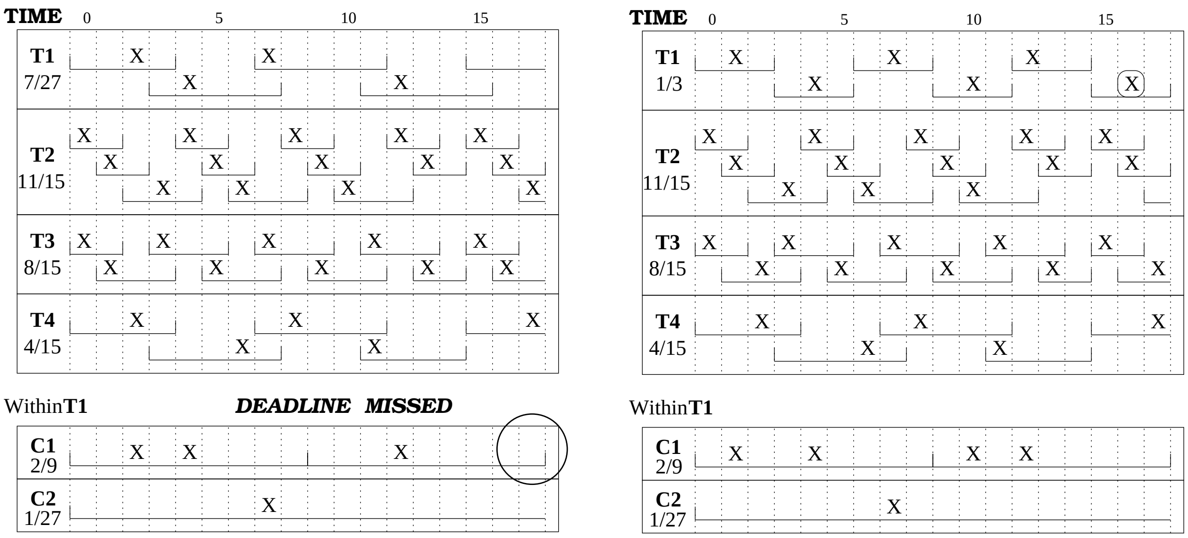
$$\mathcal{T}.\hat{w} = \max\left\{\{\Delta(c, \mathcal{T}.w, \mathcal{T}.mcp)\} \cup \left\{\Delta\left(c, \mathcal{T}.w, \left\lceil\frac{k}{\mathcal{T}.w}\right\rceil\right) \;\middle|\; \mathcal{T}.w \cdot \mathcal{L}(\mathcal{T}, \mathcal{T}.mcp) \geq k > \lfloor \mathcal{T}.w \cdot \mathcal{T}.mcp \rfloor\right\}\right\}.$$

Rule 3B (EDF-Linear): If no other rules apply, then let

$$\mathcal{T}.\hat{w} = \min\left\{\phi(c, \mathcal{T}.w, \mathcal{T}.mcp), \frac{2}{\mathcal{T}.msw}\right\}.$$

Another example. To demonstrate these rules, we consider the task set in Fig. 2. This task set is similar to the one presented in Fig. 1 and has only one supertask **T1**. Again, suppose that $c = 0$, *i.e.*, no deadline misses are allowed. Notice that neither Rule 1 nor Rule 2 applies to **T1**. We will now calculate the inflation factor resulting from the EDF versions of Rules 3A and 3B. First, we need to know $\mathbf{T1}.mcp$ and $\mathbf{T1}.msw$. By (2), $\mathbf{T1}.mcp = \min\{9, 27\} = 9$ (the period of component task **C1**), and by (3), $\mathbf{T1}.msw = \lceil \frac{27}{7} \rceil = 4$.

Consider Rule 3A. For the first term in the max expression, we have $\Delta(c, \mathbf{T1}.w, \mathbf{T1}.mcp) = \Delta(0, \frac{7}{27}, 9) = \frac{1 + \lfloor \frac{7}{27} \cdot 9 \rfloor}{9 + 0} = \frac{1 + \lfloor \frac{63}{27} \rfloor}{9} = \frac{3}{9} = \frac{1}{3}$. For the other term, we again evaluate the bounds for k first. For the lower bound, $\lfloor \mathbf{T1}.w \cdot \mathbf{T1}.mcp \rfloor = \lfloor \frac{7}{27} \cdot 9 \rfloor = 2$. Because $\gcd(\mathbf{T1}.e, \mathbf{T1}.p) = 1$, the upper bound yields $\mathbf{T1}.w \cdot \mathcal{L}(\mathbf{T1}, \mathbf{T1}.mcp) = \frac{7}{27} \cdot \mathcal{L}(\mathbf{T1}, 9) = \frac{7}{27} \cdot \left(\lceil 9 \cdot \frac{1}{27} \rceil \cdot \frac{27}{1}\right) = \frac{7}{27} \cdot 27 = 7$. Therefore, the range of k in the second term is $7 \geq k > 2$. It can be shown that Rule 3A selects a new weight of $\frac{1}{3}$ for **T1**, as shown in Fig. 2(b). We will omit the details



(a) (b)

Figure 2: PD^2/EDF schedule with a **(a)** normal and **(b)** reweighted supertask **T1** on two processors.

of this calculation because the testing set is rather large and because a similar calculation was given earlier in Sec. 3.

Now, consider Rule 3B. For the first subexpression in this rule, we have $\phi(c, \mathbf{T1}.w, \mathbf{T1}.mcp) = \phi(0, \frac{7}{27}, 9) = \frac{1+\frac{7}{27}\cdot 9}{9+0} = \frac{\frac{90}{27}}{9} = \frac{10}{27}$. Since $\mathbf{T1}.msw = 4$, the second subexpression evaluates to $\frac{1}{2}$. Therefore, Rule 3B selects $\mathbf{T1}.\hat{w} = \min\left\{\frac{10}{27}, \frac{1}{2}\right\} = \frac{10}{27}$ as the new weight, which is higher than the weight selected by Rule 3A.

Notice that the inflation factor is $\frac{1}{3} - \frac{7}{27} = \frac{2}{27}$ by Rule 3A and $\frac{10}{27} - \frac{7}{27} = \frac{1}{9}$ by Rule 3B. If **C1** and **C2** were EPDF-scheduled instead, then the inflation factor would increase to at least $\frac{19}{135}$, which is significantly higher.

4.2 Correctness of Reweighting

The correctness of the EDF rules follows from almost identical reasoning to that presented earlier in Sec. 3. For this reason, we present only the first part of the proof of Theorem 5 below, which parallels Theorem 1 in both form and proof, to demonstrate the correspondence. Remaining EDF-related results are proved in the appendix. The following lemma is an EDF counterpart of Lemma 1.

Lemma 7 (Number of jobs in an interval): For any periodic task T and any interval $I = [t_0, t_0 + L)$, $\left\lfloor \frac{L}{T.\hat{p}} \right\rfloor - 1 \leq jobs(T, I) \leq \left\lfloor \frac{L}{T.\hat{p}} \right\rfloor$.

Theorem 5 (EDF underallocated interval): For any EDF-scheduled component task T of a supertask $\mathcal{T}$, integer c satisfying $c \geq 0$, and schedule $\mathcal{S}$ satisfying (P1), the i^{th} job of T misses its deadline by more than c time units only if there exists intervals $I = [i \cdot T.\hat{p} - L, i \cdot T.\hat{p})$ and $I' = [i \cdot T.\hat{p} - L, i \cdot T.\hat{p} + c)$, where $L \geq T.\hat{p}$, satisfying the following conditions.

(i) $allocation(\mathcal{T}, I', \mathcal{S}) = windows(\mathcal{T}, I') = \lfloor \mathcal{T}.w \cdot L \rfloor - 1$

(ii) $\sum_{U \in \mathcal{T}} (jobs(U, I) \cdot U.\hat{e}) = \lfloor \mathcal{T}.w \cdot L \rfloor$

Proof: As before, we use an idle instant argument. However, due to the change from subtasks to jobs, a different idle instant definition is required. A *(job) t-idle instant* of supertask $\mathcal{T}$ occurs at $t_0 \leq t$ in $\mathcal{S}$ iff all jobs of tasks in τ that execute in the interval $[t_0, t]$ have job releases at or after t_0. Again, note that at least one t-idle instant exists since $t_0 = 0$ is a t-idle instant for any t.

Assume that the i^{th} job of T misses its deadline by more than c time units. Let t_0 be the latest $(i \cdot T.\hat{p} - 1)$-idle instant of $\mathcal{T}$ at or before $(i-1) \cdot T.\hat{p}$. (Recall that i^{th} job of T has a deadline at $i \cdot T.\hat{p} - 1$ rather than $i \cdot T.\hat{p}$ because time slots are numbered starting from 0.) For the remainder of the proof, let $L = i \cdot T.\hat{p} - t_0$, $I = [t_0, i \cdot T.\hat{p})$, and $I' = [t_0, i \cdot T.\hat{p} + c)$. (Notice that $L \geq T.\hat{p}$.) All quanta allocated to $\mathcal{T}$ in I' are used by some component task since an idle quantum at $t \in [(i-1) \cdot T.\hat{p}, i \cdot T.\hat{p})$ would be allocated to T and an idle quantum at $t \in [t_0, (i-1) \cdot T.\hat{p})$ would imply that $t+1$ is a later idle instant than t_0. Similarly, no jobs with deadlines at or after $i \cdot T.\hat{p}$ execute in I' since T has higher priority for $t \in [(i-1) \cdot T.\hat{p}, i \cdot T.\hat{p} + c)$ and the execution of such a subtask at $t \in [t_0, (i-1) \cdot T.\hat{p})$ would again contradict the choice of t_0. Therefore, all quanta allocated to $\mathcal{T}$ in I' are consumed by jobs that are properly contained in I. Hence, the following relationship exists.

$$allocation(\mathcal{T}, I', \mathcal{S}) < \sum_{U \in \mathcal{T}} (jobs(U, I) \cdot U.\hat{e}) \tag{15}$$

Applying Lemma 7 yields the following upper bound on the right-hand side of (15).

Claim 4: $\sum_{U \in \mathcal{T}} (jobs(U, I) \cdot U.\hat{e}) \leq \lfloor \mathcal{T}.w \cdot L \rfloor$

Proof of Claim:

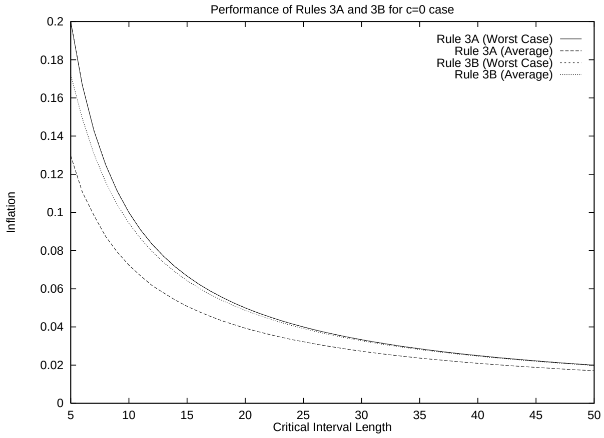
$$\begin{aligned}
\sum_{U \in \mathcal{T}} (jobs(U, I) \cdot U.\hat{e}) &\leq \sum_{U \in \mathcal{T}} \left(\left\lfloor \frac{L}{U.\hat{p}} \right\rfloor \cdot U.\hat{e} \right) && \text{, Lemma 7} \\
&\leq \sum_{U \in \mathcal{T}} \lfloor U.\hat{w} \cdot L \rfloor && , n \cdot \lfloor x \rfloor \leq \lfloor n \cdot x \rfloor \\
&\leq \left\lfloor \left(\sum_{U \in \mathcal{T}} U.\hat{w} \right) \cdot L \right\rfloor && , \lfloor x + y \rfloor \geq \lfloor x \rfloor + \lfloor y \rfloor \\
&= \lfloor \mathcal{T}.w \cdot L \rfloor && \text{, definition of } \mathcal{T}.w
\end{aligned}$$

□

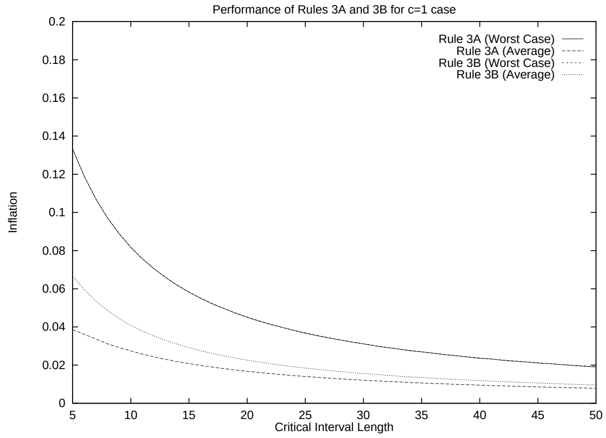
Notice that the upper bound in Claim 4 matches that in Claim 1. The rest of the proof is nearly identical to that given for Theorem 1. It is deferred to the appendix, along with other EDF-related results. □

5 Experimental Results

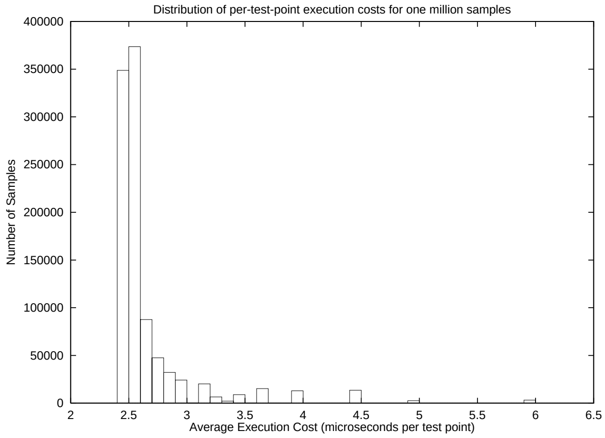
We now present the results of an experimental evaluation of our reweighting rules. Since the EPDF and EDF versions of Rules 3A and 3B differ only in the critical interval length (CIL) assumed, these rules can be analyzed



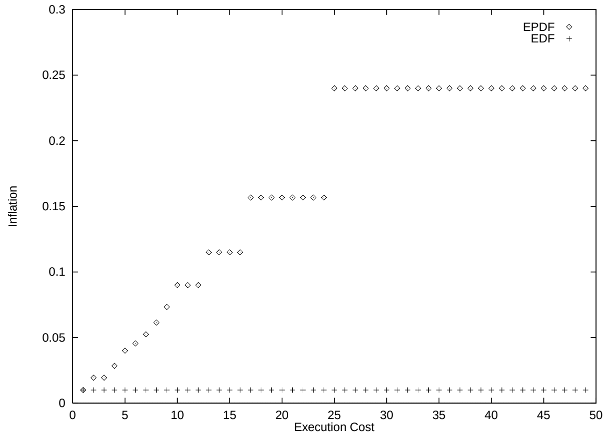
(a)



(b)



(c)



(d)

Figure 3: **(a)** and **(b)** Inflation factors calculated by Rules 3A and 3B as a function of CIL assuming $c = 0$ and $c = 1$, respectively. **(c)** Distribution of per-test-point execution costs for Rule 3A. **(d)** Comparison of EPDF and EDF inflation factors.

as a function of this length, without distinguishing between versions. This is done in most of our experiments, although we do include one experiment conducted specifically to compare the EPDF and EDF versions.

Inflation factors produced by Rules 3A and 3B. To compare Rules 3A and 3B, we used both rules to reweight a supertask with weight $\frac{x}{5001}$ for every integer x in the range $1 \leq x \leq 5000$ and every CIL in the range $[5, 50]$. Fig. 3(a) depicts average- and worst-case inflations for each CIL assuming the deadline overshoot parameter c equals 0. Fig. 3(b) shows corresponding results for $c = 1$. (Although there appears to be no worst-case curve for Rule 3B in these graphs, this curve actually coincides with that of Rule 3A, because ϕ is defined by Δ's worst-case behavior.) As the two graphs show, a lower inflation factor is required if overshoots can be tolerated (as expected). Also, observe how steeply the inflation factor drops as the CIL increases. For a CIL of 50, the worst-case inflation has already dropped to 0.02 in Fig. 3(a). Since relatively long CILs are expected in practice, these graphs suggest that a supertask's inflation will be small in practice. Also, notice that Rule 3A provides much lower inflation factors for smaller CILs, and that the two rules converge as the CIL increases. This suggests that, in practice, the difference between Rules 3A and 3B may be negligible.

Execution cost of exponential reweighting. The next experimental evaluation was conducted to assess the execution cost of Rule 3A. This experiment was conducted on a Linux box using a 32-bit 200 MHz processor. One million randomly generated samples were considered, with each "sample" being defined by a supertask period $\mathcal{T}.p$ and execution cost $\mathcal{T}.e$ and a CIL L, satisfying $\mathcal{T}.p \in [100, 2000]$, $\mathcal{T}.e \in [1, \mathcal{T}.p - 1]$, and $L \in [\mathcal{T}.msw, 2000]$. Rule 3A was applied to each sample and timed. The total elapsed time was then divided by the size of the testing set for that sample to produce a per-test-point execution cost. The distribution of these per-test-point execution costs (in microseconds) is shown in Fig. 3(c). As the figure shows, approximately 2.7 microseconds per test point was required on average. This suggests that a supertask with an execution cost of one million, which is unlikely to occur in practice, could still be reweighted within just a few seconds. Based on this, the cost of applying Rule 3A is expected to be low in practice, despite its exponential time complexity.

EPDF vs. EDF. The final experimental evaluation was conducted to compare EPDF and EDF. In this experiment, we considered a scenario in which the CIL is defined by a component task with parameters $T.\hat{e} = x$ and $T.\hat{p} = 100$, which is part of a supertask with parameters $\mathcal{T}.e = 51$ and $\mathcal{T}.p = 100$. We calculated the inflation factor produced by both the EPDF and EDF versions of Rule 3A for each $x \in [1, 50]$. (This scenario is admittedly contrived, because varying x would likely affect both the supertask parameters and the CIL. Nonetheless, the results do demonstrate the price to be paid for using EPDF instead of EDF.) Fig. 3(d) shows the inflation factors produced by the two versions as a function of x. As the figure shows, using EDF results in much lower inflation factors for most values of x. At $x = 1$, the two version produce the same inflation factor, since the minimum window size and period are the same in this case. As x increases, the minimum window size decreases, causing the EPDF version of Rule 3A to produce higher inflation factors. These results suggest that, in practice, EDF will often result in much lower inflation factors than EPDF.

6 Concluding Remarks

We have shown that component-task misallocations can be prevented in Pfair supertasks by reweighting. We have presented reweighting rules for both EPDF- and EDF-scheduled component tasks. Indeed, this is the first paper to consider EDF-scheduled component tasks. Our reweighting rules are given in a general form in which a deadline overshoot parameter may be specified. Inflation factors can be reduced if overshoots can be tolerated. We have presented experimental evidence that shows that, in practice, inflation factors should be low and should be efficiently computable. We have also shown that EDF-scheduled component tasks are likely to require lower inflations than EPDF-scheduled component tasks.

Anderson and Srinivasan have proposed variants of Pfair scheduling in which subtasks can be released either "early" or "late" (*i.e.*, they may be jittered) [2, 3]. Our results are applicable in systems using these variants *provided subtasks of supertasks are not released early or late*, so that (P1) remains intact. It is even permissible for *component tasks* to release their subtasks early or late. (Note that sporadic job releases are a special case of this.) In particular, late releases can only *decrease* processor demand and thus have no adverse impact. Early releases also are not a problem because algorithms designed to handle them do so by postponing pseudo-deadlines so that they are the same as in an ordinary Pfair-scheduled system. That is, the deadlines used in such algorithms are the same as those assumed in our calculations.

One potential problem with using EDF scheduling for component tasks is that "fairness" is sacrificed. As a compromise, one could apply our EDF reweighting rules in conjunction with EPDF scheduling and early-release subtasks. In this case, all job deadlines of component tasks would be met, but pseudo-deadlines strictly *within* a job might be missed. Nonetheless, this results in a more fair distribution of processor time than EDF.

Our reweighting method bears some resemblance to a technique proposed by Baruah and Lin in work on pinwheel scheduling [6]. In their work, a pinwheel task is implemented as a Pfair-scheduled task, which is reweighted so that the pinwheel requirements will be met. Because we are reweighting for different reasons, our reweighting rules, though similar in spirit, are nonetheless quite different from theirs.

References

[1] J. Anderson, R. Jain, and K. Jeffay. Efficient object sharing in quantum-based real-time systems. In *Proceedings of the 19th IEEE Real-Time Systems Symposium*, pp. 346–355. IEEE, Dec. 1998.

[2] J. Anderson and A. Srinivasan. Early-release fair scheduling. In *Proceedings of the 12th Euromicro Conference on Real-Time Systems*, pp. 35–43, June 2000.

[3] J. Anderson and A. Srinivasan. Pfair scheduling: Beyond periodic task systems. In *Proceedings of the 7th International Conference on Real-Time Computing Systems and Applications*, pp. 297–306, Dec. 2000.

[4] S. Baruah, N. Cohen, C.G. Plaxton, and D. Varvel. Proportionate progress: A notion of fairness in resource allocation. *Algorithmica*, 15:600–625, 1996.

[5] S. Baruah, J. Gehrke, and C. G. Plaxton. Fast scheduling of periodic tasks on multiple resources. In *Proceedings of the 9th International Parallel Processing Symposium*, pp. 280–288, April 1995.

[6] S. Baruah and S.-S. Lin. Pfair scheduling of generalized pinwheel task systems. *IEEE Transactions on Computers*, 47(7):812–816, July 1998.

[7] P. Holman, A. Srinivasan, and J. Anderson. Using pfair supertasks to service aperiodic jobs. Forthcoming.

[8] M. Moir and S. Ramamurthy. Pfair scheduling of fixed and migrating periodic tasks on multiple resources. In *Proceedings of the Twentieth IEEE Real-Time Systems Symposium*, pp. 294–303, Dec. 1999.

[9] R. Rajkumar. *Synchronization In Real-Time Systems – A Priority Inheritance Approach.* Kluwer Academic Publishers, Boston, 1991.

Appendix

In this appendix, proofs omitted previously are given.

EPDF results. Lemmas 1 and 2, stated in Sec. 3, are proved below.

Lemma 1 (Number of subtask windows in an interval): For any Pfair-scheduled task V and any interval $I = [t_0, t_0 + L)$, $\lfloor V.\hat{w} \cdot L \rfloor - 1 \leq \mathit{windows}(V, I) \leq \lfloor V.\hat{w} \cdot L \rfloor$.

Proof: By [4], a Pfair-scheduled task V receives between $\lfloor V.\hat{w} \cdot t \rfloor$ and $\lceil V.\hat{w} \cdot t \rceil$ quantum in any interval $[0, t)$. Thus, V has $\lfloor V.\hat{w} \cdot t \rfloor$ deadlines and $\lceil V.\hat{w} \cdot t \rceil$ releases before t. Therefore, $\mathit{windows}(V, I)$ can be expressed as

$$\mathit{windows}(V, I) = \lfloor V.\hat{w} \cdot (t_0 + L) \rfloor - \lceil V.\hat{w} \cdot t_0 \rceil, \tag{16}$$

which allows the desired bounds to be derived as follows.

$$\begin{aligned}
\mathit{windows}(V, I) &= \lfloor V.\hat{w} \cdot (t_0 + L) \rfloor - \lceil V.\hat{w} \cdot t_0 \rceil \\
&\geq \lfloor V.\hat{w} \cdot t_0 \rfloor + \lfloor V.\hat{w} \cdot L \rfloor - \lceil V.\hat{w} \cdot t_0 \rceil && , \lfloor x + y \rfloor \geq \lfloor x \rfloor + \lfloor y \rfloor \\
&\geq \lceil V.\hat{w} \cdot t_0 \rceil - 1 + \lfloor V.\hat{w} \cdot L \rfloor - \lceil V.\hat{w} \cdot t_0 \rceil && , \lfloor x \rfloor \geq \lceil x \rceil - 1 \\
&= \lfloor V.\hat{w} \cdot L \rfloor - 1 && , \text{simplification}
\end{aligned}$$

$$\begin{aligned}
\mathit{windows}(V, I) &= \lfloor V.\hat{w} \cdot (t_0 + L) \rfloor - \lceil V.\hat{w} \cdot t_0 \rceil \\
&\leq \lceil V.\hat{w} \cdot t_0 \rceil + \lfloor V.\hat{w} \cdot L \rfloor - \lceil V.\hat{w} \cdot t_0 \rceil && , \lfloor x + y \rfloor \leq \lceil x \rceil + \lfloor y \rfloor \\
&= \lfloor V.\hat{w} \cdot L \rfloor && , \text{simplification}
\end{aligned}$$

□

Lemma 2 (Pfair task allocation over an interval): For any Pfair schedule $\mathcal{S}$ containing task V and any interval I, $\mathit{allocation}(V, I, \mathcal{S}) \geq \mathit{windows}(V, I)$.

Proof: Since V misses no deadlines in $\mathcal{S}$, a quantum is allocated to each window properly contained in I. □

EDF results. In the rest of this appendix, a full proof of the EDF reweighting rules is given. This proof was omitted previously due to its strong resemblance to the proof already presented for the EPDF rules. To simplify the statements of the results that follow, we define the supertask $\mathcal{T}$ to be $(\mathcal{S}, c)$*-EDF-safe* iff EDF schedules τ, the set of component tasks of $\mathcal{T}$, within $\mathcal{T}$'s allocation in $\mathcal{S}$ such that no job deadlines of tasks in τ are missed by more than c time slots. The schedule $\mathcal{S}$ is omitted if *every* schedule satisfying (P1) satisfies this condition.

We begin by proving Lemma 7, which was stated in Sec. 4 without proof. We then give a full proof of Theorem 5. Finally, we state and prove results that parallel those for EPDF scheduling given after Theorem 1.

Lemma 7 (Number of jobs in an interval): For any periodic task T and any interval $I = [t_0, t_0 + L)$,

$\left\lfloor \frac{L}{T.\hat{p}} \right\rfloor - 1 \leq jobs(T,I) \leq \left\lfloor \frac{L}{T.\hat{p}} \right\rfloor.$

Proof: Since T is a periodic task, it has a job release every $T.\hat{p}$ time units, with each job's deadline one slot before the next job release. Assume that the first job release occurs at $t = t'$. It follows that T has $\left\lfloor \frac{t-t'}{T.\hat{p}} \right\rfloor$ job deadlines and $\left\lceil \frac{t-t'}{T.\hat{p}} \right\rceil$ job releases before t. Therefore, $jobs(T,I)$ can be expressed as

$$jobs(T,I) = \left\lfloor \frac{t_0 + L - t'}{T.\hat{p}} \right\rfloor - \left\lceil \frac{t_0 - t'}{T.\hat{p}} \right\rceil,$$

which allows the desired bounds to be derived as follows.

$$\begin{aligned}
jobs(T,I) &= \left\lfloor \frac{t_0 + L - t'}{T.\hat{p}} \right\rfloor - \left\lceil \frac{t_0 - t'}{T.\hat{p}} \right\rceil \\
&\geq \left\lfloor \frac{t_0 - t'}{T.\hat{p}} \right\rfloor + \left\lfloor \frac{L}{T.\hat{p}} \right\rfloor - \left\lceil \frac{t_0 - t'}{T.\hat{p}} \right\rceil && ,\lfloor x+y \rfloor \geq \lfloor x \rfloor + \lfloor y \rfloor \\
&\geq \left\lceil \frac{t_0 - t'}{T.\hat{p}} \right\rceil - 1 + \left\lfloor \frac{L}{T.\hat{p}} \right\rfloor - \left\lceil \frac{t_0 - t'}{T.\hat{p}} \right\rceil && ,\lfloor x \rfloor \geq \lceil x \rceil - 1 \\
&= \left\lfloor \frac{L}{T.\hat{p}} \right\rfloor - 1 && ,\text{simplification}
\end{aligned}$$

$$\begin{aligned}
jobs(T,I) &= \left\lfloor \frac{t_0 + L - t'}{T.\hat{p}} \right\rfloor - \left\lceil \frac{t_0 - t'}{T.\hat{p}} \right\rceil \\
&\leq \left\lceil \frac{t_0 - t'}{T.\hat{p}} \right\rceil + \left\lfloor \frac{L}{T.\hat{p}} \right\rfloor - \left\lceil \frac{t_0 - t'}{T.\hat{p}} \right\rceil && ,\lfloor x+y \rfloor \leq \lceil x \rceil + \lfloor y \rfloor \\
&= \left\lfloor \frac{L}{T.\hat{p}} \right\rfloor && ,\text{simplification}
\end{aligned}$$

□

Theorem 5 (EDF underallocated interval): For any EDF-scheduled component task T of a supertask $\mathcal{T}$, integer c satisfying $c \geq 0$, and schedule $\mathcal{S}$ satisfying (P1), the i^{th} job of T misses its deadline by more than c time units only if there exists intervals $I = [i \cdot T.\hat{p} - L, i \cdot T.\hat{p})$ and $I' = [i \cdot T.\hat{p} - L, i \cdot T.\hat{p} + c)$, where $L \geq T.\hat{p}$, satisfying the following conditions.

(i) $allocation(\mathcal{T}, I', \mathcal{S}) = windows(\mathcal{T}, I') = \lfloor \mathcal{T}.w \cdot L \rfloor - 1$

(ii) $\sum_{U \in \mathcal{T}} (jobs(U,I) \cdot U.\hat{e}) = \lfloor \mathcal{T}.w \cdot L \rfloor$

Proof: The first part of the proof was given in Sec. 4. The rest is given here.

Combining (15) and Claim 4, we get

$$allocation(\mathcal{T}, I', \mathcal{S}) < \lfloor \mathcal{T}.w \cdot L \rfloor. \tag{17}$$

Furthermore, the following expressions hold, by Lemma 2, Lemma 1, (P3), and $c \geq 0$, respectively.

$$allocation(\mathcal{T}, I', \mathcal{S}) \geq windows(\mathcal{T}, I') \tag{18}$$

$$windows(\mathcal{T}, I') \geq \lfloor \mathcal{T}.\hat{w} \cdot (L + c) \rfloor - 1 \tag{19}$$

$$\lfloor \mathcal{T}.\hat{w} \cdot (L+c) \rfloor - 1 \geq \lfloor \mathcal{T}.w \cdot (L+c) \rfloor - 1 \tag{20}$$

$$\lfloor \mathcal{T}.w \cdot (L+c) \rfloor - 1 \geq \lfloor \mathcal{T}.w \cdot L \rfloor - 1 \tag{21}$$

Expressions (17)–(21) yield Condition (i) of the theorem. Furthermore, by (15) and Condition (i), we have $\sum_{U \in \tau} (jobs(U, I) \cdot U.\hat{e}) > \lfloor \mathcal{T}.w \cdot L \rfloor - 1$, which implies that Condition (ii) holds, by Claim 4. □

Corollary 5 (Sufficient schedulability condition for EDF component tasks): For any supertask $\mathcal{T}$ with EDF-scheduled component tasks, schedule $\mathcal{S}$ satisfying (P1), and integer c satisfying $c \geq 0$, if $allocation(\mathcal{T}, I, \mathcal{S}) \geq \lfloor \mathcal{T}.w \cdot L \rfloor$ for all I and L where $|I| = L + c$ and $L \geq \mathcal{T}.mcp$, then $\mathcal{T}$ is ($\mathcal{S}$, c)-EDF-safe.

Proof: By (2), $\mathcal{T}.mcp$ lower bounds all periods of component tasks in τ. Therefore, the stated condition prevents Condition (i) of Theorem 5 from holding. □

Corollary 6 (Correctness of Rule 2 for EDF): For any supertask $\mathcal{T}$ with EDF-scheduled component tasks, if $\mathcal{T}.\hat{w} = \mathcal{T}.w$ then $\mathcal{T}$ is c-EDF-safe for all integers $c \geq \mathcal{T}.msw$.

Proof: Identical to the proof of Corollary 2. □

Theorem 6 (Reweighting for EDF schedulability): For any supertask $\mathcal{T}$ and integer c satisfying $\mathcal{T}.msw > c \geq 0$, if $\mathcal{T}.\hat{w} \geq \max\{\, \Delta(c, \mathcal{T}.w, L) \mid L \geq \mathcal{T}.mcp\}$ then $\mathcal{T}$ is c-EDF-safe.

Proof: Consider the following derivation.

$$\begin{aligned}
&\mathcal{T}.\hat{w} \geq \max\{\, \Delta(c, \mathcal{T}.w, L) \mid L \geq \mathcal{T}.mcp\} \\
&\quad \Rightarrow (\forall L : L \geq \mathcal{T}.mcp : \mathcal{T}.\hat{w} \geq \Delta(c, \mathcal{T}.w, L)) \qquad \text{, transitivity of } \geq \\
&\quad \Rightarrow (\forall I, L, \mathcal{S} : L \geq \mathcal{T}.mcp \wedge |I| = L + c \wedge (\mathcal{S} \text{ satisfies (P1)}) : allocation(\mathcal{T}, I, \mathcal{S}) \geq \lfloor \mathcal{T}.w \cdot L \rfloor) \\
&\qquad \text{, by Lemma 3}
\end{aligned}$$

From this derivation and Corollary 5, it follows that $\mathcal{T}$ is c-EDF-safe. □

Corollary 7 (Making the testing set finite for EDF): For any supertask $\mathcal{T}$ and integer c satisfying $\mathcal{T}.msw > c \geq 0$, if $\mathcal{T}.\hat{w} \geq \max\{\, \Delta(c, \mathcal{T}.w, L) \mid \mathcal{L}(\mathcal{T}, \mathcal{T}.mcp) \geq L \geq \mathcal{T}.mcp\}$ then $\mathcal{T}$ is c-EDF-safe.

Proof: This is a direct application of Corollary 3 to the result of Theorem 6. Recall that $\mathcal{L}(\mathcal{T}, \mathcal{T}.mcp)$ is the next multiple of $\frac{\mathcal{T}.p}{\mathsf{gcd}(\mathcal{T}.e, \mathcal{T}.p)}$ at or after $\mathcal{T}.mcp$. In addition, by Corollary 3, $\Delta(c, \mathcal{T}.w, \mathcal{L}(\mathcal{T}, \mathcal{T}.mcp)) = \max\{\, \Delta(c, \mathcal{T}.w, L) \mid L \geq \mathcal{L}(\mathcal{T}, \mathcal{T}.mcp)\}$. Therefore, we can remove all $L > \mathcal{L}(\mathcal{T}, \mathcal{T}.mcp)$ from the testing set. □

Theorem 7 (Correctness of EDF Rule 3A): For any supertask $\mathcal{T}$ and integer c satisfying $\mathcal{T}.msw > c \geq 0$, if $\mathcal{T}.\hat{w} \geq \max\left\{\{\Delta(c, \mathcal{T}.w, \mathcal{T}.mcp)\} \cup \left\{\, \Delta\left(c, \mathcal{T}.w, \left\lceil \frac{k}{\mathcal{T}.w} \right\rceil\right) \,\middle|\, \mathcal{T}.w \cdot \mathcal{L}(\mathcal{T}, \mathcal{T}.mcp) \geq k > \lfloor \mathcal{T}.w \cdot \mathcal{T}.mcp \rfloor \right\}\right\}$ then $\mathcal{T}$ is c-EDF-safe.

Proof: Observe that, when $\lfloor \mathcal{T}.w \cdot L \rfloor$ remains constant, $\Delta(c, \mathcal{T}.w, L) = \frac{1 + \lfloor \mathcal{T}.w \cdot L \rfloor}{L + c}$ monotonically decreases with increasing L. Therefore, when evaluating the testing set in Corollary 7, only the first value of L associated with each distinct value of $\lfloor \mathcal{T}.w \cdot L \rfloor$ must be evaluated. As shown in the proof of Theorem 3, the first value of L satisfying $\lfloor \mathcal{T}.w \cdot L \rfloor = k$ is $\left\lceil \frac{k}{\mathcal{T}.w} \right\rceil$. Furthermore, since $k = \lfloor \mathcal{T}.w \cdot L \rfloor$, Corollary 7 implies that the largest value of k that must be checked is $\lfloor \mathcal{T}.w \cdot \mathcal{L}(\mathcal{T}, \mathcal{T}.mcp) \rfloor = \mathcal{T}.w \cdot \mathcal{L}(\mathcal{T}, \mathcal{T}.mcp)$. In addition, Corollary 7 also implies that $L \geq \mathcal{T}.mcp$. Therefore, $k \geq \lfloor \mathcal{T}.w \cdot \mathcal{T}.mcp \rfloor$. However, this first value of k should be handled separately, which is the reason for the $\Delta(c, \mathcal{T}.w, \mathcal{T}.mcp)$ term in Rule 3A. Note that if we substitute $k = \lfloor \mathcal{T}.w \cdot \mathcal{T}.mcp \rfloor$ within $\Delta\left(c, \mathcal{T}.w, \left\lceil \frac{k}{\mathcal{T}.w} \right\rceil\right)$, then the third parameter becomes $\left\lceil \frac{\lfloor \mathcal{T}.w \cdot \mathcal{T}.mcp \rfloor}{\mathcal{T}.w} \right\rceil$, which could be *smaller* than $\mathcal{T}.mcp$. Therefore, only values of k *larger* than $\lfloor \mathcal{T}.w \cdot \mathcal{T}.mcp \rfloor$ should be considered when evaluating $\Delta\left(c, \mathcal{T}.w, \left\lceil \frac{k}{\mathcal{T}.w} \right\rceil\right)$.

Combining these arguments, we only need to evaluate $\Delta\left(c, \mathcal{T}.w, \left\lceil \frac{k}{\mathcal{T}.w} \right\rceil\right)$ for $\mathcal{T}.w \cdot \mathcal{L}(\mathcal{T}, \mathcal{T}.mcp) \geq k > \lfloor \mathcal{T}.w \cdot \mathcal{T}.mcp \rfloor$ along with $\Delta(c, \mathcal{T}.w, \mathcal{T}.mcp)$ to determine the value of the reweighting expression shown in Corollary 7. □

Lemma 8 (Safety of EDF Rule 3B's first subexpression): For any supertask $\mathcal{T}$ with EDF-scheduled component tasks and integer c satisfying $\mathcal{T}.msw > c \geq 0$, if $\mathcal{T}.\hat{w} \geq \phi(c, \mathcal{T}.w, \mathcal{T}.mcp)$ then $\mathcal{T}$ is c-EDF-safe.

Proof: This lemma follows directly from previous results. By Lemma 4, $\mathcal{T}.\hat{w} \geq \phi(c, \mathcal{T}.w, \mathcal{T}.mcp)$ implies that $\mathcal{T}.\hat{w} \geq \max\{\phi(c, \mathcal{T}.w, L) \mid L \geq \mathcal{T}.mcp\}$, which implies that $\mathcal{T}.\hat{w} \geq \max\{\Delta(c, \mathcal{T}.w, L) \mid L \geq \mathcal{T}.mcp\}$ by (14). Therefore, by Theorem 6, $\mathcal{T}$ is c-EDF-safe. □

Theorem 8 (Simple EDF reweighting): For any supertask $\mathcal{T}$ and integer c satisfying $c \geq 0$, if $0 < \mathcal{T}.w < 1$ and $\mathcal{T}.\hat{w} \geq \frac{2}{\mathcal{T}.msw}$, then $\mathcal{T}$ is c-EDF-safe.

Proof: Identical to the proof of Theorem 4, but uses Theorem 6 rather than Theorem 2. □