Improved Conditions for Bounded Tardiness under EPDF Fair Multiprocessor Scheduling *

UmaMaheswari C. Devi and James H. Anderson
Department of Computer Science, The University of North Carolina, Chapel Hill, NC

Abstract

The earliest-pseudo-deadline-first (EPDF) Pfair algorithm is more efficient than other known Pfair scheduling algorithms, but is not optimal on more than two processors. In earlier work, Srinivasan and Anderson established a sufficient per-task utilization restriction for ensuring a tardiness of at most one quantum under EPDF. They also conjectured that a tardiness bound of one quantum applies to systems that are not restricted in any way. In this paper, we present counterexamples that show that this conjecture is false. We also present sufficient utilization restrictions that are more liberal than theirs.

Keywords: *soft real-time systems, pfairness, real-time scheduling, multiprocessors, tardiness bounds.*

1 Introduction

Pfair scheduling, originally introduced by Baruah *et al.* [3], is the only known way of optimally scheduling recurrent real-time tasks on multiprocessors. Under Pfair scheduling, each task must execute at a uniform rate, while respecting a fixed-size allocation quantum. A task's execution rate is defined by its *weight* or *utilization*. Uniform rates are ensured by requiring 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 T is effectively subdivided into quantum-length *subtasks* that are subject to intermediate deadlines. To avoid deadline misses, ties among subtasks with the same deadline must be broken carefully. In fact, tie-breaking rules are of crucial importance when devising optimal Pfair scheduling algorithms.

As discussed by Srinivasan and Anderson [7], overheads associated with tie-breaking rules may be problematic in many soft real-time systems. Web-hosting systems, server farms, and high-speed routers are examples. In these systems, *fair* resource allocation is needed, so that quality-of-service guarantees can be provided. However, an extreme notion of fairness that precludes all deadline misses is not required. Moreover, in systems such as routers, the inclusion of tie-breaking information in subtask priorities may result in unacceptably high space overhead.

In dynamic systems that permit tasks to join or leave, tie-breaking rules may cause other problems. In such a system, spare processing capacity may become available that can be rellocated. In Pfair terminology, this amounts to *reweighting* tasks so that all processing capacity is utilized. As explained in [7], it is possible to reweight each task so that its next subtask deadline is preserved. If no tie-breaking information is maintained, such an approach entails very little cost. However, weight changes can cause tie-breaking information to change, so if tie-breaking rules are used, reweighting may necessitate a $\Theta(N \log N)$ cost for N tasks, due to the need to resort the scheduler's priority queue. This cost may be prohibitive if reallocations are frequent.

The observations above motivated Srinivasan and Anderson to consider, for soft real-time applications, the viability of the simplified *earliest-pseudo-deadline-first* (EPDF) algorithm, which uses no tie-breaking rules. They succeeded in showing that EPDF can guarantee a *tardiness* (amount by which a subtask misses its deadline) bound of one quantum for every subtask, provided a certain condition holds. This condition, which is described in detail later, can be ensured by limiting each task's weight to at most $1/2$, and can be gen-

*Work supported by NSF grants ITR 0082866 and CCR 0204312.

eralized to apply to tardiness bounds other than one. Unfortunately, Srinivasan and Anderson left open the question of whether such conditions are necessary to guarantee small constant tardiness.

In this paper, we provide counterexamples that show that, in general, restrictions on individual task utilizations are *necessary* to guarantee constant tardiness bounds. In addition, we show that for the general case, a more liberal per-task weight restriction of 2/3 (66.7%) is sufficient to ensure a tardiness of one quantum, and that for a somewhat special case, which is described in the next section, this restriction can be relaxed to 11/15 (73.3%). We also present generalizations of these conditions that can be applied to other tardiness bounds.

The rest of the paper is organized as follows. In Sec. 2, needed definitions are given. In Sec. 3, the results above are proved. We conclude in Sec. 4.

2 Pfair Scheduling

In defining notions relevant to Pfair scheduling, we limit attention (for now) to periodic tasks.[1] A periodic task T with an integer *period* $T.p$ and an integer *execution cost* $T.e$ has a *weight* $wt(T) = T.e/T.p$, where $0 < wt(T) < 1$. A task is *light* if its weight is less than 1/2, and *heavy* otherwise.

Pfair algorithms allocate processor time in descrete quanta; the time interval $[t, t+1)$, where t is a nonnegative integer, is called *slot* t. (Hence, time t refers to the beginning of slot t.) A task may be allocated time on different processors, but not in the same slot (*i.e.*, interprocessor migration is allowed but parallelism is not). The sequence of allocation decisions over time defines a *schedule* S. Formally, $S : \tau \times \mathcal{N} \mapsto \{0, 1\}$, where τ is a task set and $\mathcal{N}$ is the set of nonnegative integers. $S(T, t) = 1$ iff T is scheduled in slot t. On M processors, $\sum_{T \in \tau} S(T, t) \leq M$ holds for all t.

Lags and subtasks. The notion of a Pfair schedule is defined by comparing such a schedule to an ideal fluid schedule, which allocates $wt(T)$ processor time to task T in each slot. Deviation from the fluid schedule is formally captured by the concept of *lag*. Formally, the *lag of task T at time t* is[2] $lag(T, t) = wt(T) \cdot t - \sum_{u=0}^{t-1} S(T, u)$. A schedule is defined to be *Pfair* iff

$$(\forall T, t :: -1 < lag(T, t) < 1). \tag{1}$$

Informally, the allocation error associated with each task must always be less than one quantum.

These lag bounds have the effect of breaking each task T into an infinite sequence of *quantum-length subtasks*. We denote the i^{th} subtask of task T as T_i, where $i \geq 1$. As in [3], we associate a *pseudo-release* $r(T_i)$ and a *pseudo-deadline* $d(T_i)$ with each subtask T_i, as follows. (For brevity, we often drop the prefix "pseudo-.")

$$r(T_i) = \left\lfloor \frac{i-1}{wt(T)} \right\rfloor \qquad d(T_i) = \left\lceil \frac{i}{wt(T)} \right\rceil \tag{2}$$

To satisfy (1), T_i must be scheduled in the interval $w(T_i) = [r(T_i), d(T_i))$, termed its *window*. Note that $r(T_{i+1})$ is either $d(T_i) - 1$ or $d(T_i)$. Thus, consecutive windows either overlap by one slot, or are disjoint. The "b-bit," denoted by $b(T_i)$, distinguishes between these possibilities. Formally,

$$b(T_i) = \left\lceil \frac{i}{wt(T)} \right\rceil - \left\lfloor \frac{i}{wt(T)} \right\rfloor. \tag{3}$$

For example, in Fig. 1(a), $b(T_i) = 1$ for $1 \leq i \leq 7$ and $b(T_8) = 0$.

The *length* of T_i's window, denoted $|w(T_i)|$, is $d(T_i) - r(T_i)$. As an example, consider subtask T_1 in Fig. 1(a). Here, we have $r(T_1) = 0$, $d(T_1) = 2$, and $|w(T_1)| = 2$. Therefore, T_1 must be scheduled at either time 0 or time 1. The following lemma relates window lengths and weights.

Lemma 1 [**1**] *The length of each window of a task T is either $\lceil \frac{1}{wt(T)} \rceil$ or $\lceil \frac{1}{wt(T)} \rceil + 1$.*

The above lemma implies that the windows of heavy tasks are of length two or three (see Fig. 1(a)). For such tasks, the "group deadline" is used to mark the end of a sequence of windows of length two. Consider a sequence $T_i, \ldots, T_j$ of subtasks of a heavy task T such that $b(T_k) = 1$, $|w(T_{k+1})| = 2$ for all $i \leq k < j$. Then, scheduling T_i in its last slot forces the other subtasks in this sequence to be scheduled in their last slots. For example, in Fig. 1(a), scheduling T_3 in slot 4 forces T_4 and T_5 to be scheduled in slots 5 and 6, respectively. A group deadline corresponds to a time by which any such "cascade" of scheduling decisions must end. Formally, it is a time

[1] Unless specified otherwise, we assume that each periodic task begins execution at time 0.

[2] For conciseness, we leave the schedule implicit and use $lag(T, t)$ instead of $lag(T, t, S)$.

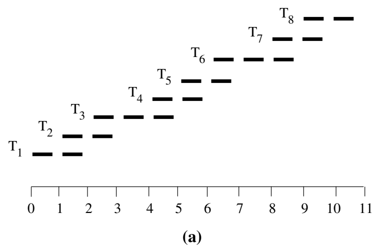

Figure 1. **(a)** Windows of the first job of a periodic task T with weight 8/11. This job consists of subtasks $T_1, \ldots, T_8$, each of which must be scheduled within its window, or else a lag-bound violation will result. (This pattern repeats for every job.) **(b)** The Pfair windows of an IS task. Subtask T_5 becomes eligible one time unit late. **(c)** The Pfair windows of a GIS task. Subtask T_4 is absent and T_6 is one time unit late.

t such that either $(t = d(T_i) \;\wedge\; b(T_i) = 0)$ or $(t + 1 = d(T_i) \wedge \; |w(T_i)| = 3)$ for some subtask T_i. For example, the task in Fig. 1(a) has group deadlines at times 4, 8, and 11.

We let $D(T_i)$ denote the group deadline of subtask T_i. If T is heavy, then $D(T_i) = (\min u : u \geq d(T_i) \wedge \; u$ is a group deadline of $T)$. For example, in Fig. 1(a), $D(T_1) = 4$ and $D(T_6) = 11$. If T is light, then $D(T_i) = 0$.

Pfair scheduling algorithms. The earliest-pseudo-deadline-first (EPDF) Pfair algorithm, considered in this paper, is optimal on one or two processors, but not on more than two processors [1]. As its name suggests, EPDF gives higher priority to subtasks with earlier deadlines. A tie between subtasks with equal deadlines is broken arbitrarily. As mentioned earlier, careful tie breaking is crucial for optimality on more than two processors. At present, three such optimal algorithms are known: PF[3], PD[4], and PD^2[6]. These algorithms prioritize subtasks on an EPDF basis, but differ in the choice of tie-breaking rules.

Task Models. In this paper, we consider the *intra-sporadic* (IS) and the *generalized-intra-sporadic* (GIS) task models [2, 6], which provide a general notion of recurrent execution that subsume that found in the well-studied periodic and sporadic task models. The *sporadic* model generalizes the periodic model by allowing jobs to be released "late"; the IS model generalizes the sporadic model by allowing subtasks to be released late, as illustrated in Fig. 1(b). More specifically, the separation between $r(T_i)$ and $r(T_{i+1})$ is allowed to be more than $\lfloor i/wt(T) \rfloor - \lfloor (i-1)/wt(T) \rfloor$, which would be the separation if T were periodic. Thus, an IS task is obtained by allowing a task's windows to be shifted right from where they would appear if the task were periodic.

Let $\theta(T_i)$ denote the offset of subtask T_i, *i.e.*, the amount by which $w(T_i)$ has been shifted right. Then, by (2), we have the following.

$$r(T_i) = \theta(T_i) + \left\lfloor \frac{i-1}{wt(T)} \right\rfloor \quad (4)$$

$$d(T_i) = \theta(T_i) + \left\lceil \frac{i}{wt(T)} \right\rceil \quad (5)$$

The offsets are constrained so that the separation between any pair of subtask releases is at least the separation between those releases if the task were periodic. Formally,

$$k > i \Rightarrow \theta(T_k) \geq \theta(T_i). \quad (6)$$

Each subtask T_i has an additional parameter $e(T_i)$ that specifies the first time slot in which it is eligible to be scheduled. It is assumed that $e(T_i) \leq r(T_i)$ and $e(T_i) \leq e(T_{i+1})$ for all $i \geq 1$. Additionally, no subtask can become eligible before its predecessor completes execution, *i.e.*,

$$h < i \;\wedge\; e(T_i) \leq r(T_i) \;\wedge\; S(T_h, u) = 1 \Rightarrow \; e(T_i) \geq \min(u+1, r(T_i)). \quad (7)$$

The interval $[r(T_i), d(T_i))$ is called the *PF-window* of T_i and the interval $[e(T_i), d(T_i))$ is called the *IS-window* of T_i. A schedule for an IS system is *valid* iff each subtask is scheduled in its IS-window. (Note that the notion of a job is not mentioned here. For systems in which subtasks are grouped into jobs that are released in sequence, the definition of e would preclude a subtask from becoming eligible before the beginning of its job.)

b-bits for IS tasks are defined in the same way as for periodic tasks. $r(T_i)$ is defined as follows.

$$r(T_i) = \begin{cases} e(T_i), & \text{if } i = 1 \\ max(e(T_i), d(T_{i-1}) - b(T_{i-1})), & \text{if } i \geq 2 \end{cases} \quad (8)$$

Thus, if T_i is eligible *during* T_{i-1}'s PF-window, then $r(T_i) = d(T_{i-1}) - b(T_{i-1})$, and hence, the spacing between $r(T_{i-1})$ and $r(T_i)$ is exactly as in a periodic task system. On the other hand, if T_i becomes eligible *after* T_{i-1}'s PF-window, then T_i's PF-window begins when T_i becomes eligible. Note that (8) implies that consecutive PF-windows of the same task are either disjoint, or overlap by one slot, as in a periodic system.

T_i's deadline $d(T_i)$ is defined to be $r(T_i) + |w(T_i)|$. PF-window lengths are defined as before. Thus, by (2), we have the following.

$$|w(T_i)| = \left\lceil \frac{i}{wt(T)} \right\rceil - \left\lfloor \frac{i-1}{wt(T)} \right\rfloor \quad (9)$$

$$d(T_i) = r(T_i) + \left\lceil \frac{i}{wt(T)} \right\rceil - \left\lfloor \frac{i-1}{wt(T)} \right\rfloor \quad (10)$$

The IS model is more suitable than the periodic model for the networking examples mentioned in Sec. 1. Due to network congestion and other factors, packets may arrive late or in bursts. The IS model treats these possibilities as first-class concepts and handles them more seamlessly. In particular, a late packet arrival corresponds to an IS delay. On the other hand, if a packet arrives early (as part of a bursty sequence), then its eligibility time will be less than its Pfair release time. Note that its Pfair release time determines its deadline. Thus, in effect, an early packet arrival is handled by postponing its deadline to where it would have been had the packet arrived on time.

Generalized intra-sporadic task systems. In our proof, we consider *generalized* intra-sporadic (GIS) task systems. Such a task system is obtained by removing subtasks from a corresponding IS task system. Specifically, in a GIS task system, a task T, after releasing subtask T_i, may release subtask T_k, where $k > i + 1$, instead of T_{i+1}, with the following restriction: $r(T_k) - r(T_i)$ is at least $\left\lfloor \frac{k-1}{wt(T)} \right\rfloor - \left\lfloor \frac{i-1}{wt(T)} \right\rfloor$. In other words, $r(T_k)$ is not smaller than what it would have been if T_{i+1}, T_{i+2}, ..., T_{k-1} were present and released as early as possible. For the special case where T_k is the first subtask released by T, $r(T_k)$ must be at least $\left\lfloor \frac{k-1}{wt(T)} \right\rfloor$. Fig. 1(c) shows an example. If T_i is the most recently released subtask of T, then T may release T_k, where $k > i$, as its next subtask at time t, if $r(T_i) + \left\lfloor \frac{k-1}{wt(T)} \right\rfloor - \left\lfloor \frac{i-1}{wt(T)} \right\rfloor \leq t$. If a task T, after executing subtask T_i, releases subtask T_k, then T_k is called the *successor* of T_i and T_i is called the *predecessor* of T_k.

As shown in [2], an IS or GIS task system τ is feasible on M processors iff

$$\sum_{T \in \tau} wt(T) \leq M. \quad (11)$$

Shares and lags in IS and GIS task systems. The lag of T at time t is defined in the same way as for periodic tasks. Let $ideal(T, t)$ denote the processor share that T receives in an ideal fluid (processor-sharing) schedule in $[0, t)$. Then,

$$lag(T, t) = ideal(T, t) - \sum_{u=0}^{t-1} S(T, u). \quad (12)$$

Before defining $ideal(T, t)$, we define $share(T, u)$, which is the share assigned to task T in slot u. $share(T, u)$ is defined in terms of a function f that indicates the share assigned to each subtask in each slot.

$$\begin{aligned} f(T_i, r(T_i)) &= \left(\left\lfloor \frac{i-1}{wt(T)} \right\rfloor + 1 \right) \cdot wt(T) - (i - 1) \\ f(T_i, d(T_i) - 1) &= i - \left(\left\lceil \frac{i}{wt(T)} \right\rceil - 1 \right) \cdot wt(T) \quad (13) \end{aligned}$$

$$f(T_i, u) = \begin{cases} wt(T), & \text{if } \; r(T_i) < u < d(T_i) - 1 \\ 0, & \text{if } \; u \notin [r(T_i), d(T_i) - 1] \end{cases}$$

Fig. 2 shows the values of f for different subtasks of a task of weight 5/16.

$share(T, u)$ is then simply defined in terms of f as

$$share(T, u) = \sum_i f(T_i, u). \quad (14)$$

Lemma 2 [**5**] *Let f be as defined by* (13). *Then, the following hold.*

(**a**) *In any time slot $u \geq 0$, at most two consecutive subtasks of a task may have positive values for f.*

(**b**) *If $b(T_{i-1}) = 1$ for subtask T_{i-1} of task T, and subtask T_i exists, then $f(T_{i-1}, d(T_{i-1})) + f(T_i, r(T_i)) = wt(T)$.*

(**c**) $(\forall T, t :: share(T, t) \leq wt(T))$.

(**d**) $(\forall T_i, t :: f(T_i, t) \leq wt(T))$.

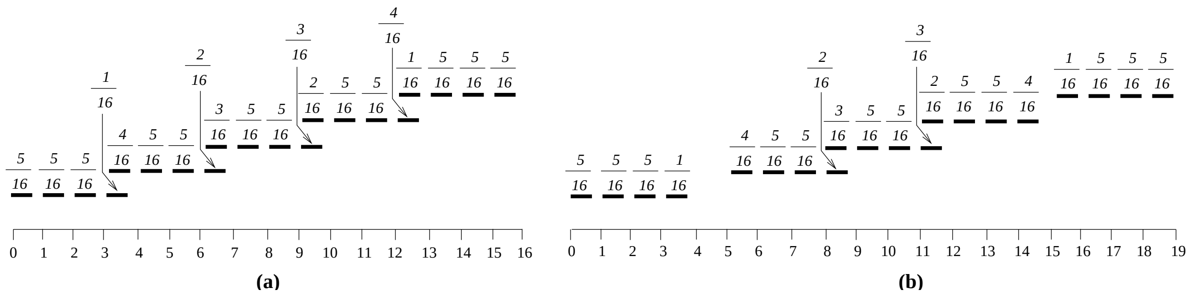

Figure 2. Fluid schedule for the first five subtasks ($T_1, \ldots, T_5$) of a task T of weight 5/16. The share of each subtask in each slot of its PF-window is shown. In **(a)**, no subtask is released late; in **(b)**, T_2 and T_5 are released late. Note that $share(T, 3)$ is either 5/16 or 1/16 depending on when subtask T_2 is released.

(e) $(\forall T_i :: \sum_{u=r(T_i)}^{d(T_i-1)} f(T_i, u) = 1)$.

(f) $(\forall T_i, t :: f(T_i, t) \geq \frac{1}{T.p})$. □

We can now define $ideal(T, t)$ as $\sum_{u=0}^{t-1} share(T, u)$. Hence, from (12),

$$
\begin{aligned}
lag(T, t+1) &= \sum_{u=0}^{t} (share(T, u) - S(T, u)) \\
&= lag(T, t) + share(T, t) - S(T, t). \qquad (15)
\end{aligned}
$$

Similarly, the total lag for a schedule S and task system τ at time $t+1$, denoted $LAG(\tau, t+1)$, is as follows. ($LAG(\tau, 0)$ is defined to be 0.)

$$
LAG(\tau, t+1) = LAG(\tau, t) + \sum_{T \in \tau} (share(T, t) - S(T, t)). \qquad (16)
$$

3 Tardiness Bounds for EPDF

In this section, we present results concerning tardiness bounds that can be guaranteed under EPDF. The term *tardiness* denotes the lateness of a task's subtasks. Formally, if subtask T_i completes execution at time t, then its tardiness is given by $\max(0, t - d(T_i))$. The *tardiness of a task system* is defined as the maximum tardiness among all of its subtasks in any schedule.

It is easy to show that subtask deadlines can be missed under EPDF. In [7], it was conjectured that EPDF always ensures a tardiness of at most one. We now show that this conjecture is false.

Theorem 1 *Tardiness under EPDF can exceed three quanta. In particular, if EPDF is used to schedule task system τ_i ($1 \leq i \leq 3$) in Table 1, then a tardiness of $i+1$ quanta is possible.*

Table 1. Counterexamples to show that tardiness under EPDF can exceed three.

	Task Set: # of tasks	Task Set: weight	Util. (M)	Tardiness (in quanta)
τ_1	4	1/2	10	2 at 50
	3	3/4		
	6	23/24		
τ_2	4	1/2	19	3 at 963
	3	3/4		
	5	23/24		
	10	239/240		
τ_3	4	1/2	80	4 at 43,204
	3	3/4		
	3	23/24		
	1	31/32		
	4	119/120		
	4	239/240		
	6	479/480		
	8	959/960		
	15	1199/1200		
	15	2399/2400		
	20	4799/4800		

Proof: Fig. 3 shows a schedule for τ_1, in which a subtask has a tardiness of two at time 50. The schedules for τ_2 and τ_3 are too lengthy to be depicted; we verified them using two independently-coded EPDF simulators. □

The sufficient condition for a tardiness of one quantum as given by Srinivasan and Anderson requires that the sum of the weights of the $M-1$ heaviest tasks be less than $\frac{M+1}{2}$. This can be

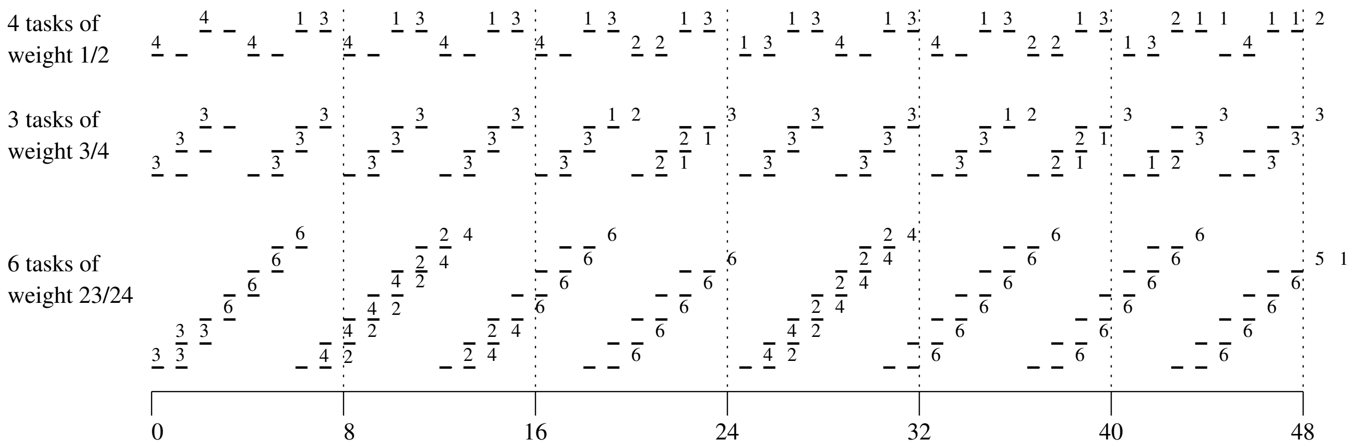

Figure 3. Counterexample to prove that tardiness under EPDF can exceed one quantum. 13 periodic tasks with total utilization 10 are scheduled on 10 processors. In the schedule, tasks of the same weight are shown together as a group. Each column corresponds to a time slot. The Pfair window of each subtask is shown as a sequence of dashes that are aligned. An integer value n in slot t means that n tasks in the corresponding group have a subtask scheduled at t. Subtasks that miss deadlines are shown scheduled after their windows. Ties are broken in favor of tasks with lower weights. In this schedule, 11 subtasks miss their deadlines at time 48. Hence, tardiness is 2 quanta for at least one subtask.

ensured if the weight of each task is restricted to be at most $1/2$. We next show that, in general, a weight restriction of $2/3$ (66.7%) per task is sufficient to guarantee a tardiness of one, and that for the special case where a subtask does not become eligible before its release time, the restriction can be improved to $11/15$ (73.3%). These restrictions are stated below.

(C) The weight of each task is at most $2/3$.
(D) The weight of each task is at most $11/15$, and for every subtask T_i, $e(T_i) = r(T_i)$.

In this paper, we prove the following theorem, which states that (C) or (D) is sufficient for EPDF to guarantee a tardiness of at most one.

Theorem 2 *EPDF ensures a tardiness of at most one quantum for feasible GIS task systems that satisfy* (C) *or* (D).

Before proving Theorem 2, we reproduce some helpful definitions and lemmas from [6] and [7].

In a schedule S, if k processors are idle at time slot t, then we say that there are k *holes* in S at slot t. The following lemma relates an increase in total lag to the presence of holes.

Lemma 3 [**6**] *If* $LAG(\tau, t) < LAG(\tau, t+1)$*, then there is a hole in slot* t *in* S.

We prove Theorem 2 in a manner similar to that used in [7]. If (C) or (D) is not sufficient, then t_d and τ defined as follows both exist.

Definition 1: t_d is the earliest deadline of a subtask with a tardiness of two under EPDF in any task system satisfying (C) or (D), *i.e.*, there exists some task system with a subtask with a deadline at t_d and a tardiness of two, and there does not exist any other task system with a subtask with a deadline prior to t_d and a tardiness of two.

Definition 2: τ is a feasible task system satisfying (C) or (D) with the following properties.

(T1) t_d is the earliest deadline of a subtask in τ with a tardiness of two under EPDF.

(T2) No feasible task system satisfying ((C) or (D)) and (T1) releases fewer subtasks in $[0, t_d)$ than τ.

(T3) No feasible task system satisfying ((C) or (D)), (T1), and (T2) has a larger rank than τ at t_d, where rank is defined as follows.

The *rank* of a system τ at t is the sum of the eligibility times of all subtasks with deadlines at most t. Formally, $rank(\tau, t) =$

$\sum_{T \in \tau} \sum_{\{T_i | d(T_i) \leq t\}} e(T_i)$. By (T1) and (T2), exactly one subtask in τ has a tardiness of two: if several such subtasks exist, then all but one can be removed and the remaining subtask will still have a tardiness of two, contradicting (T2). Additionally, the following assertions follow from the above properties and definitions.

(A1) $(\exists T_i \in \tau : d(T_i) = t_d \wedge tardiness(T_i) = 2)$
(A2) $(\forall T_i \in \tau : d(T_i) < t_d \Rightarrow tardiness(T_i) \leq 1)$

In the rest of this paper, we use S to denote an EPDF schedule for τ on M processors, in which subtask T_i with a deadline at t_d has a tardiness of two. We next prove some properties about τ and S.

Lemma 4 *The following properties hold for τ and S, the EPDF schedule for τ, where T_i is any subtask in τ.*

(**a**) *If T_i is scheduled at t, then, $e(T_i) \geq min(r(T_i), t)$.*

(**b**) *For all T_i, $d(T_i) \leq t_d$.*

(**c**) *At least $M + 1$ subtasks of τ miss their deadlines at t_d.*

(**d**) $LAG(\tau, t_d) \geq M + 1$.

(**e**) *There are no holes in slot $t_d - 1$.*

(**f**) $LAG(\tau, t_d - 1) \geq M + 1$.

(**g**) *There exists a time $u \in [0, t_d - 2]$ such that $LAG(\tau, u) < M + 1$ and $LAG(\tau, u+1) \geq M + 1$.*

Part (a) is proved in [6], and parts (c) and (d) are proved in [7]. The others are proved below.

Proof of (b): If $d(U_j) > t_d$, then U_j can be removed without affecting the scheduling of subtasks with higher priorities, and hence the tardiness of any remaining subtasks is unchanged. This contradicts (T2).

Proof of (e): If there is a hole in slot $t_d - 1$, then at most $M - 1$ subtasks are scheduled there. Thus, at least two of the $M + 1$ subtasks missing their deadlines at t_d are schedulable at $t_d - 1$ (because their predecessors are not scheduled there), and thus are scheduled there by EPDF.

Proof of (f): By (e), there are no holes in slot $t_d - 1$. Hence by Lemma 3, $LAG(\tau, t_d - 1) \geq LAG(\tau, t_d)$. Hence, by (d), $LAG(\tau, t_d - 1) \geq M + 1$.

Proof of (g): This follows from the fact that $LAG(\tau, 0) = 0$ and $LAG(\tau, t_d - 1) \geq M + 1$. $\square$

By Lemma 4(g), there exists a time slot $u < t_d - 1$ across which LAG increases to at least $M + 1$. By Lemma 3, there is at least one hole in u. Thus, there exists a time slot t_h with $h \geq 1$ holes satisfying the following.

(A3) $0 \leq t_h < t_d - 1 \wedge LAG(\tau, t_h + 1) \geq M + 1 \wedge (\forall u : u \in [0, t_h] :: LAG(\tau, u) < M + 1)$.

In other words, t_h is the earliest time slot across which LAG increases to $M + 1$. In what follows, we derive an upper bound on the lags of all tasks in τ at $t_h + 1$ and prove that if (C) or (D) is satisfied, then their sum is strictly less than $M + 1$, contradicting the existence of t_h.

3.1 Categorization of Subtasks

As can be seen from (12) and (13), the lag of a task T at t depends on the flows that subtasks of T receive in each time slot until t in the ideal system. Hence, a tight estimate of these flows is essential to bounding the lag of T reasonably accurately. If a subtask's index is not known, then (13), which can otherwise be used to compute the flow received by any subtask in any slot *exactly*, is not of much help. Hence, in this subsection, we define terms that will help in categorizing subtasks, and then derive upper bounds for the flows that these categories of subtasks receive in the ideal system.

***k*-dependent subtasks.** Subtasks of heavy tasks can be divided into "groups" based on their group deadlines in a straightforward manner: place all subtasks with identical group deadlines in the same *group* and identify the group using the smallest index of any subtask in that group. For example, in Fig. 1, $G_1 = \{T_1, T_2\}$, $G_3 = \{T_3, T_4, T_5\}$, and $G_6 = \{T_6, T_7, T_8\}$. If there are no IS or GIS separations among the subtasks of a group, then a deadline miss by one for a subtask T_i will necessarily result in a deadline miss by at least one for the remaining subtasks in T_i's group. Hence, a subtask T_j is dependent on all prior subtasks in its group for not missing its deadline. We say that T_j is *k-dependent*, where $k \geq 0$, if T is heavy and T_j is the $(k+1)^{\text{st}}$ subtask in its group (assuming all subtasks are present). If a task T is light, then we simply define all of its subtasks to be 0-dependent.

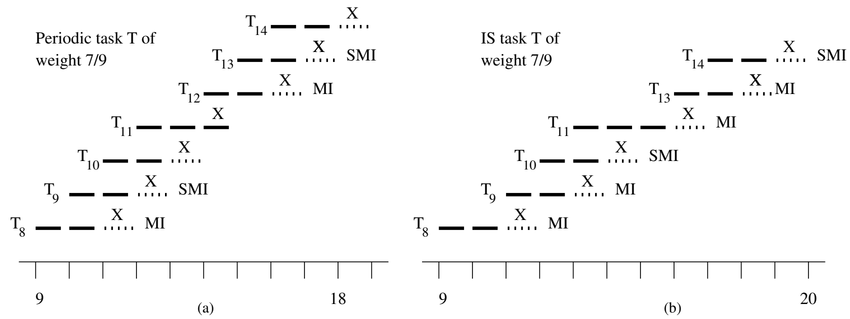

Figure 4. Possible schedules for the second job of **(a)** a periodic and **(b)** an IS task of weight 7/9 under EPDF. Subtasks are scheduled in the slots marked by an X. Solid (dotted) lines indicate slots that lie within (outside) the window of a subtask. A subtask scheduled in a dotted slot misses its deadline. In (a), T_8 and T_{12} are MIs, T_9 and T_{13} are SMIs, and the remaining subtasks fall within neither category. T_{10} and T_{14} have a tardiness of 1, and T_{11} has a tardiness of 0. In (b), T_8, T_9, T_{11}, and T_{13} are MIs, and T_{10} and T_{14} are SMIs. Note that T_8 and T_9 (T_{11} and T_{13}) belong to the same group G_8 (G_{11}). Thus, if there are IS separations, there may be more than one MI in a group.

Miss initiators. We call a subtask missing its deadline at t by *one* a *miss initiator* (MI) for its group if no subtask of the same task is scheduled at $t-1$. Thus, a subtask is an MI if it misses its deadline and is either the first subtask in its group to do so or is separated from its predecessor by an IS or GIS separation. Such a subtask is termed a miss initiator because in the absence of future separations, it causes all subsequent subtasks in its group to miss their deadlines as well. $T_k \in G_i$ is an MI if $tardiness(T_k) = 1 \ \wedge \ S(T_k, t) = 1$, and $S(T_j, t-1) = 0$, for all $j < k$. Several examples of MI's are shown in Fig. 4

Successors of miss initiators. The immediate successor T_{i+1} of a miss-initiator subtask T_i is called a *successor of a miss initiator* (SMI) if $tardiness(T_{i+1}) = tardiness(T_i) = 1$ and $S(T_{i+1}, t) = 1 \Rightarrow S(T_i, t-1) = 1$. Fig. 4 shows several examples. Note that for T_{i+1} to be an SMI, its predecessor in S must be T_i, rather than some lower-indexed subtask of T.

The next two lemmas follow from the definition of k-dependency.

Lemma 5 *If T_i is a k-dependent subtask of a **periodic** task T, where $i \geq 2$ and $k \geq 1$, then, $d(T_i) = d(T_{i-1}) + 1$ and $r(T_i) = d(T_{i-1}) - 1$.*

Lemma 6 *If T_i is a k-dependent subtask of T, where $k \geq 1$ and $wt(T) < 1$, then $|w(T_i)| = 2$ and $b(T_{i-1}) = 1$.*

The next lemma relates the weight of a task to the k-dependency of its subtasks.

Lemma 7 *If subtask T_i of task T is k-dependent, where $k \geq 0$, then $wt(T) > \frac{k}{k+1}$.*

Proof: If T_i is k-dependent, then $w(T_i)$ is part of a sequence of at least $k+1$ windows, at least k of which are of length two. Using this fact it is easy to show that $wt(T) > \frac{k}{k+1}$. □

The two lemmas that follow bound the flow received by a k-dependent subtask.

Lemma 8 *The flow $f(T_i, r(T_i))$ received by a k-dependent subtask T_i of a **periodic** task T in the first slot of its window in an ideal system, is at most $k \cdot \frac{T.e}{T.p} - (k-1) - \frac{1}{T.p}$, for all $k \geq 0$.*

Proof: The proof is by induction on k.

Induction Base: Let $k = 0$. From Lemma 2(d),

$$f(T_i, r(T_i)) \leq wt(T) = \frac{T.e}{T.p}.$$

Because $wt(T) < 1$, and $T.e$ and $T.p$ are integral, $T.e \leq T.p - 1$. Thus, $f(T_i, r(T_i)) \leq wt(T) \leq 1 - 1/T.p$, and the lemma holds for the base case.

Induction Step: Assuming that the lemma holds for $(k-1)$-dependent subtasks, we show that it holds for k-dependent subtasks, where $k \geq 1$. Because $k \geq 1$, T is heavy, by the definition of k-dependency. Hence, by Lemma 1, $|w(T_{i-1})|$ is either two or three. We consider two cases.

Case 1: $|w(T_{i-1})| = 2$. If $k \geq 1$, then T_{i-1} is

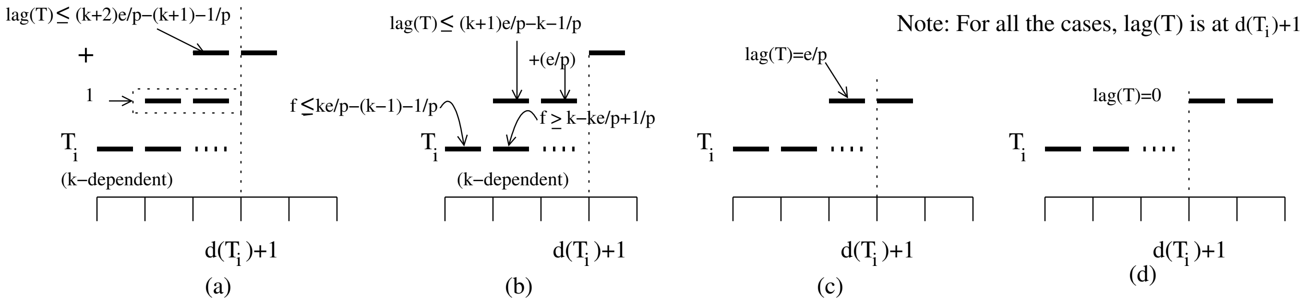

Figure 5. Lemma 10. **(a)** Case 1, **(b)** Case 2, **(c)** Case 3, and **(d)** Case 4.

$(k-1)$-dependent. Therefore, by the induction hypothesis,

$$f(T_{i-1}, r(T_{i-1})) \leq \\ (k-1)\cdot(T.e/T.p) - (k-2) - (1/T.p). \quad (17)$$

Because $|w(T_{i-1})| = 2$, by Lemma 2(e) $f(T_{i-1}, d(T_{i-1})) = 1 - f(T_{i-1}, r(T_{i-1}))$. Hence, from (17), $f(T_{i-1}, d(T_{i-1})) \geq (k-1) + (1/T.p) - (k-1)\cdot(T.e/T.p)$. Because T_i is k-dependent, where $k \geq 1$, from Lemma 6, $b(T_{i-1}) = 1$, and by Lemma 2(b) $f(T_i, r(T_i)) = (T.e/T.p) - f(T_{i-1}, d(T_{i-1})) \leq k\cdot(T.e/T.p) - (k-1) - (1/T.p)$.

Case 2: $|w(T_{i-1})| = 3$. By Lemma 6, T_{i-1} is 0-dependent; hence, T_i is 1-dependent, *i.e.*, $k = 1$. From Lemmas 6 and Lemma 2(b),

$$\begin{aligned} &f(T_i, r(T_i)) \\ &= \frac{T.e}{T.p} - f(T_i, d(T_{i-1})) \\ &\leq \frac{T.e}{T.p} - \frac{1}{T.p} \qquad \text{(from Lemma 2(f))}. \quad \square \end{aligned}$$

Lemma 9 *The flow $f(T_i, r(T_i))$ received by a k-dependent subtask T_i of a GIS task T in the first slot of its window in an ideal system is at most $k \cdot \frac{T.e}{T.p} - (k-1) - \frac{1}{T.p}$, for all $k \geq 0$.*

Proof: Follows from Lemma 8 and the definition of GIS tasks. (The flow that T_i receives in each slot of its window is identical to the flow that it would receive were T periodic.) □

Having determined a bound on the flow received by a subtask in the first slot of its window in the ideal system, we next bound the lag of a task at time t, based on the k-dependency of its last-scheduled subtask.

Lemma 10 *Let T_i be a k-dependent subtask of a GIS task T for $k \geq 0$, and let $d(T_i) < t_d$. Then $lag(T, d(T_i)+1) < (k+2)\cdot wt(T) - k$.*

Proof: Because $d(T_i) < t_d$, from (A2), we have $tardiness(T_i) \leq 1$. Therefore, T_i and all prior subtasks of T are scheduled in $[0, d(T_i)]$. Hence, $lag(T, d(T_i)+1)$ depends on the number of subtasks of T after T_i released prior to $d(T_i)+1$, the flows they receive in the ideal system, and whether they have been scheduled in S. It can be verified from (4) and (6) that at most two successors of T_i — T_{i+1} and T_{i+2} — are released before $d(T_i)+1$. Hence, the lag of T in S is maximized if T_{i+1} and T_{i+2} are present and are released without any IS separations and S has not scheduled either of them. We consider four cases.

Case 1: $r(T_{i+1}) = d(T_i) - 1$ and $r(T_{i+2}) = d(T_i)$. In this case, $d(T_{i+1}) = d(T_i)+1$ and T_{i+1} receives its full share of one quantum by $d(T_i)+1$ in the ideal system. Further, T_{i+2} receives a share of $f(T_{i+2}, r(T_{i+2}))$. This is illustrated in Fig. 5(a). Since T_i is k-dependent, T_{i+2} is $(k+2)$-dependent. Therefore, by Lemma 9, $f(T_{i+2}, r(T_{i+2})) \leq (k+2)\cdot(T.e/T.p) - (k+1) - (1/T.p)$. Hence, $lag(T, d(T_i)+1) \leq 1 + f(T_{i+2}, r(T_{i+2})) = (k+2)\cdot(T.e/T.p) - k - (1/T.p) < (k+2)\cdot wt(T) - k$.

Case 2: $r(T_{i+1}) = d(T_i) - 1$ and $r(T_{i+2}) \geq d(T_i)+1$. In this case, $d(T_{i+1}) \geq d(T_i)+1$ and the lag of T at $d(T_i)+1$ is given by the flow received by T_{i+1} in the first two slots of its window, as shown in Fig. 5(b). Reasoning as in Case 1, it can be shown that this flow is less than $(k+2)\cdot wt(T) - k$. We leave the details of this case to the reader.

Case 3: $r(T_{i+1}) = d(T_i)$. In this case, as shown in Fig. 5(c), the lag of T depends only on the flow received by T_{i+1} in the first slot of its window, *i.e.*, $lag(T, d(T_i)+1) = f(T_{i+1}, d(T_i)) = wt(T)$. By Lemma 7, $wt(T) > \frac{k}{k+1}$, which implies that $wt(T) < (k+2)\cdot wt(T) - k$.

Case 4: $r(T_{i+1}) > d(T_i)$. In this case, no successor of T_i is eligible before $d(T_i)+1$ in the ideal system. Hence, $lag(T, d(T_i)+1) = 0$. This is illustrated in Fig. 5(d). $\square$

Displacements. To prove Theorem 2, we consider task systems obtained by removing subtasks. Removing a subtask from a GIS system results in another GIS system, and may cause other subtasks to shift earlier in the schedule. Such a shift is called a displacement and is denoted by a 4-tuple $\langle X^{(1)}, t_1, X^{(2)}, t_2 \rangle$. This is equivalent to saying that $X^{(2)}$ originally scheduled at t_2 in S displaces $X^{(1)}$ scheduled at t_1 in S. A displacement $\langle X^{(1)}, t_1, X^{(2)}, t_2 \rangle$ is *valid* iff $e(X^{(2)}) \leq t_1$. Because there can be a cascade of shifts, we may have a chain of displacements. This chain is represented by a sequence of 4-tuples.

The next two lemmas concern displacements and are proved in [6]. Lemma 11 states that a subtask removal can only cause left shifts. Lemma 12 indicates when a left shift into a slot with a hole can occur.

Lemma 11 [**6**] *Let $X^{(1)}$ be a subtask that is removed from τ, and let the resulting chain of displacements in S be $C = \Delta_1, \Delta_2, \ldots, \Delta_k$, where $\Delta_i = \langle X^{(i)}, t_i, X^{(i+1)}, t_{i+1} \rangle$. Then $t_i + 1 > t_i$ for all $i \in [1, k]$.*

Lemma 12 [**6**] *Let $\Delta = \langle X^{(1)}, t_1, X^{(2)}, t_2 \rangle$ be a valid displacement in any EPDF schedule. If $t_1 < t_2$ and there is a hole in slot t_1 in that schedule, then $X^{(2)}$ is the successor of $X^{(1)}$.*

The share that a GIS task receives in the ideal system may be zero during certain time slots, if subtasks are absent or are released late. We distinguish between tasks with and without subtasks at time t using the following definition of an *active* task.

Definition 3: A task U is active at time t if it has a subtask U_j such that $e(U_j) \leq t < d(U_j)$.

As already mentioned, we prove Theorem 2 by showing that the total lag at $t_h + 1$ is strictly less than $M + 1$, thus contradicting (A3). To facilitate this lag calculation, following [6] and [7], we define A, B, and I, as follows.

A: Set of all tasks that are active and scheduled at t_h.

B: Set of all tasks that are active, but not scheduled at t_h.

I: Set of all tasks that are inactive at t_h.

A, B, and I form a partition of τ, *i.e.*,

$$A \cup B \cup I = \tau \text{ and } A \cap B = B \cap I = I \cap A = \phi. \quad (18)$$

We further classify tasks in A, based on the tardiness of their subtasks scheduled at t_h, as follows.

A_0: Includes T in A iff its subtask scheduled at t_h has zero tardiness.

A_1: Includes T in A iff its subtask scheduled at t_h has a tardiness of one.

A_1 is further partitioned into A_1^0, A_1^1, and A_1^2.

A_1^0: Includes T in A_1 iff its subtask scheduled at t_h is an MI.

A_1^1: Includes T in A_1 iff its subtask scheduled at t_h is an SMI.

A_1^2: Includes T in A_1 iff its subtask scheduled at t_h is neither an MI nor an SMI.

From the above, we have

$$A_0 \cup A_1 = A \text{ and } A_0^1 \cup A_1^1 \cup A_1^2 = A_1. \quad (19)$$

This classification of tasks is illustrated in Fig. 6.

The next lemma gives a necessary condition for LAG to increase.

Lemma 13 [**6**] *If $LAG(\tau, t) < LAG(\tau, t+1)$, then there exists a task that is active at t but not scheduled at t.*

The next definition identifies the last-released subtask at t of any task U.

Definition 4: [**6**] Subtask U_j is the *critical subtask of U at t* iff $e(U_j) \leq t < d(U_j)$ and no other subtask U_k of U, where $k > j$, satisfies $e(U_k) \leq t < d(U_k)$.

The lemma that follows concerns the scheduling of critical subtasks of tasks in B.

Lemma 14 [**5**] *The critical subtask at t_h of every task in B is scheduled before t_h.*

Lemma 13 implies that $|B| \geq 1$. The next definition identifies the latest time at which a critical subtask at t_h of any task in B is scheduled.

Definition 5: t_b denotes the latest time before t_h, at which a subtask of a task in B that is critical at t_h is scheduled.

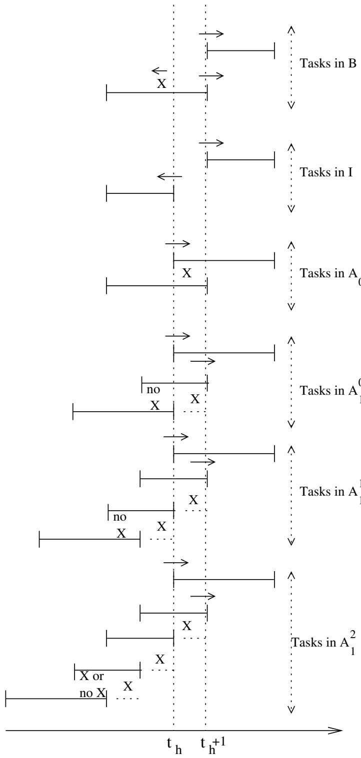

Figure 6. Task classification for Lemmas 18 – 23. The PF-windows of a sample task in each set are shown. An arrow over release (deadline) indicates that the release (deadline) could be anywhere in the direction of the arrow. An (no) X in a slot indicates that a subtask is (not) scheduled in that slot.

Definition 6: U henceforth denotes a task in B with a subtask U_j scheduled at t_b that is critical at t_h.

Lemmas 15 and 16 are concerned with the deadlines of a subtask scheduled at t_h or before.

Lemma 15 *For every subtask V_k scheduled at t_h, $d(V_k) \leq t_h + 1$.*

Proof: Assume not. Then, by (4) and (6), $r(V_l) \geq t_h+1$, where $l > k$ and V_l is V_k's successor. Because V_l is not scheduled at or before t_h, by Lemma 4(a), $e(V_l) \geq t_h + 1$. Hence, even if V_k is removed, V_l cannot be scheduled at t_h. By Lemma 12, because there is a hole in t_h, no other subtask other than V_l can shift left into t_h. Thus, V_k can be removed without causing any left shifts, which implies that $\tau' = \tau - \{V_k\}$, with one fewer subtask than τ, also has a subtask with a tardiness of two and a deadline at t_d, contradicting (T2). ☐

Lemma 16 *Let V_k be the critical subtask at t_h of a task V in B. Then, $d(V_k) = t_h + 1$.*

Proof: The proof is by contradiction. Assume to the contrary that $d(V_k) \neq t_h + 1$. Then, by Def. 4,

$$d(V_k) > t_h + 1. \tag{20}$$

We show that if (20) is true, then V_k can be removed without any impact on the tardiness of subtasks scheduled at or after t_h.

Let τ' be the system obtained by removing V_k, and let S' be the EPDF schedule for τ'. Let $\Delta_1, \Delta_2, \ldots, \Delta_n$ be the chain of displacements caused by removing V_k, where $\Delta_i = \langle X^{(i)}, t_i, X^{(i+1)}, t_{i+1}\rangle$, $X^{(1)} = V_k$, and $t_1 < t_h$. Then, by (20) and the priority definition of EPDF, $d(X^{(i)}) > t_h + 1$, for all $i \in [2, n]$. By Lemma 15, the deadline of every subtask scheduled at t_h is $t_h + 1$. Therefore, no subtask scheduled at t_h gets displaced. To see that no subtask scheduled later than t_h gets shifted left, assume that there exists a displacement $\langle X^{(k)}, t_k, X^{(k+1)}, t_{k+1}\rangle$ with $t_{k+1} > t_h$. Then, because $t_1 < t_h$ and no subtask scheduled at t_h gets displaced, $t_k < t_h$. If $\langle X^{(k)}, t_k, X^{(k+1)}, t_{k+1}\rangle$ is valid, then $e(X^{(k+1)}) \leq t_k < t_h$. But the hole in t_h implies that $X^{(k+1)}$ should be scheduled at t_h in S and not later. Hence, no subtask scheduled later than t_h can shift left in S'.

Thus, the tardiness of subtasks scheduled at or after t_h in S' is the same as their tardiness in S, which is a contradiction of (T2). Therefore, $d(V_k) = t_h + 1$. ☐

The following indicates that $|A_0| \geq 1$.

Lemma 17 *There exists a subtask W_l scheduled at t_h with $e(W_l) \leq t_b$, $d(W_l) = t_h + 1$, and $S(W, t) = 0$, for all $t \in [t_b, t_h - 1]$. Also, there are no holes in $[t_b, t_h - 1]$.*

Proof: We prove this lemma by showing that if a subtask with the stated properties does not exist, then U_j (Def. 6) can be removed without impacting the tardiness of any subtask.

Let τ' be the task system obtained by removing U_j from τ, and let S' be the EPDF schedule for τ'. Let $\Delta_1 = \langle X^{(1)}, t_1, X^{(2)}, t_2\rangle$ be the first displacement, if any, that results due to the removal of U_j. Then, $X^{(1)} = U_j$, $t_1 = t_b$, and by Lemma 11,

$$t_2 > t_b. \tag{21}$$

Let $X^{(2)} = W_l$. We first show that $t_2 \geq t_h$.

Assume to the contrary that $t_2 < t_h$. Then, by (21) and Def. 5, W is not in B. Therefore, W is in I or in A. In either case,

$$d(W_l) \leq t_h. \tag{22}$$

To see this, note that if $W \in I$, then because W is not active at t_h, by Def. 3, $d(W_l) \leq t_h$. On the other hand, if $W \in A$, then consider W's subtask, say W_i, scheduled at t_h. By Lemma 15, $d(W_i) \leq t_h + 1$. Because W_l is scheduled at $t_2 < t_h$, W_l is an earlier subtask of W, and hence, by (5) and (6), $d(W_l) \leq t_h$. Now, by Lemma 16,

$$d(U_j) = t_h + 1. \tag{23}$$

Thus, by (22) and (23), $d(U_j) > d(W_l)$. However, since EPDF selects U_j over W_l at time t_b, this is a contradiction. Thus, our assumption that $t_2 < t_h$ is false.

Having shown that $t_2 \geq t_h$, we next show $t_2 = t_h$. Assume, to the contrary, that $t_2 > t_h$. If $\langle U_j, t_b, W_l, t_2 \rangle$ were valid, then $e(W_l) \leq t_b$. This implies that W_l is eligible at t_h, and because there is a hole in t_h, it should have been scheduled there in S, and not later at t_2. We conclude that $t_2 = t_h$.

Because $\langle U_j, t_b, W_l, t_h \rangle$ is valid, no subtask of W prior to W_l is scheduled in $[t_b, t_h - 1]$. Also, there are no holes in $[t_b, t_h - 1]$ (otherwise, EPDF would have scheduled W_l there). Finally, because U_j is scheduled at t_b in preference to W_l, $d(W_l) \geq d(U_j) = t_h + 1$, which by Lemma 15 implies that $d(W_l) = t_h + 1$.

Thus, if the lemma is false, then removing U_j does not result in any displacements. Hence, the tardiness of every subtask in τ' is the same as it is in τ, which contradicts (T2). □

The next six lemmas give bounds on the lags of tasks in A, B, and I at $t_h + 1$.

Lemma 18 [7] *For $T \in I, lag(I, t_h + 1) = 0$.*

Lemma 19 [7] *For $T \in B, lag(B, t_h + 1) \leq 0$.*

Lemma 20 *For $T \in A_0, lag(T, t_h + 1) < wt(T)$.*

Proof: Let T_i be the subtask of T scheduled at t_h. As shown in Fig. 6, the ideal system can be ahead of the actual system in executing T only by the amount of flow in T_{i+1}'s first slot. By parts (b) and (f) of Lemma 2, this flow is less than $wt(T)$. □

Lemma 21 *For $T \in A_1^0$, $lag(T, t_h+1) < 2 \cdot wt(T)$.*

Proof: If $T \in A_1^0$, then the subtask T_i of T scheduled at t_h is an MI , and $d(T_i) = t_h$. If T_i is k-dependent, then by Lemma 10, $lag(T, t_h + 1)$ is less than $((k + 2) \cdot wt(T) - k)$, which is at most $2 \cdot wt(T)$, for all $k \geq 0$. □

The following two lemmas follow similarly.

Lemma 22 *For $T \in A_1^1$, $lag(T, t_h + 1) < 3 \cdot wt(T) - 1$.*

Lemma 23 *For $T \in A_1^2$, $lag(T, t_h + 1) < 4 \cdot wt(T) - 2$.*

Having classified the tasks at t_h and determined their lags at t_h, we are left with showing that if (C) or (D) hold, then (A3) is false. We do this by showing that $LAG(\tau, t_h + 1) < M + 1$ in each of the following cases.

Case A: $A_1 = \phi$.

Case B: $A_1^0 \neq \phi$.

Case C: $A_1^0 = \phi$ and $A_1^1 \neq \phi$.

Case D: $A_1^0 = A_1^1 = \phi$.

Cases A, B, and D do not impose the restriction that subtasks not become eligible before their release times, and allow weights up to 3/4, which is slightly higher than 11/15.

The following notation is used to denote subset cardinality.

$$a_0 = |A_0|;\ a_1^0 = |A_1^0|;\ a_1^1 = |A_1^1|;\ a_1^2 = |A_1^2|.$$

In each of the cases above, $LAG(\tau, t_h + 1)$ can be expressed as follows.

$$\begin{aligned}
LAG(\tau, t_h + 1) &= \sum_{T \in \tau} lag(T, t_h + 1) \\
&\leq \sum_{T \in A_0} lag(T, t_h + 1) + \sum_{T \in A_1^0} lag(T, t_h + 1) + \\
&\quad \sum_{T \in A_1^1} lag(T, t_h + 1) + \sum_{T \in A_1^2} lag(T, t_h + 1) \\
&\qquad \text{(from (18), (19), and Lemmas 18 and 19)} \\
&< \sum_{T \in A_0} wt(T) + \sum_{T \in A_1^0} 2 \cdot wt(T) + \\
&\quad \sum_{T \in A_1^1} 3 \cdot wt(T) - 1 + \sum_{T \in A_1^2} 4 \cdot wt(T) - 2 \\
&\qquad \text{(from Lemmas 20 – 23)}
\end{aligned}$$

Letting wt denote the weight of the heaviest task, $LAG(\tau, t_h + 1)$ can therefore be bounded as

$$\begin{aligned} & LAG(\tau, t_h+1) \\ & \quad < \quad a_0 \cdot wt + a_1^0 \cdot 2 \cdot wt + \\ & \qquad\quad a_1^1 \cdot (3 \cdot wt - 1) + a_1^2 \cdot (4 \cdot wt - 2). \end{aligned} \tag{24}$$

The total number of processors, M, expressed in terms of the number of subtasks in each subset of A scheduled at t_h, and the number of holes in t_h, is as follows.

$$M = a_0 + a_1^0 + a_1^1 + a_1^2 + h \tag{25}$$

3.2 Case A: $A_1 = \phi$

Case A is dealt with as follows.

Lemma 24 *If $A_1 = \phi$, then $LAG(\tau, t_h + 1) < M$.*

Proof: If $A_1 = \phi$, then

$$\begin{aligned} & LAG(\tau, t_h+1) \\ & \leq \quad a_0 \cdot wt \qquad\qquad \text{(by (24) and } a_1^0 = a_1^1 = a_1^2 = 0) \\ & < \quad M - h \qquad\qquad \text{(since } wt < 1 \text{ and } a_0 = M - h) \\ & < \quad M. \end{aligned}$$

□

3.3 Case B: $A_1^0 \neq \phi$

By Lemma 21, $lag(T, t_h + 1)$ could be as high as $2 \cdot wt(T)$, if the subtask T_i of T scheduled at t_h is an MI , *i.e.,* is in A_1^0. Therefore, if a_1^0 is large, then LAG could exceed $M + 1$. However, as we show below, if $a_1^0 \geq 2h - 2$, then $LAG(\tau, t_h + 1) \leq LAG(\tau, t_h)$, contradicting (A3).

We begin by giving a lemma concerning the sum of the weights of tasks in I.

Lemma 25 *If $LAG(\tau, t_h + 1) > LAG(\tau, t_h)$, then $\sum_{V \in I} wt(V) < h$.*

Proof: From (16),

$$\begin{aligned} & LAG(\tau, t_h+1) \\ & = \quad LAG(\tau, t_h) + \sum_{T \in \tau} (share(T, t_h) - S(T, t_h)) \\ & = \quad LAG(\tau, t_h) + \sum_{T \in A \cup B} (share(T, t_h)) - (M - h) \\ & \quad \text{(from (18) and } share(T, t_h) = 0 \text{ for } T \text{ in } I\text{, and (25))} \\ & \leq \quad LAG(\tau, t_h) + \sum_{T \in A \cup B} wt(T) - (M - h) \\ & \qquad\qquad\qquad\qquad\qquad \text{(by Lemma 2(c)).} \end{aligned}$$

If $LAG(\tau, t_h + 1) > LAG(\tau, t_h)$, then

$$\sum_{T \in A \cup B} wt(T) > M - h. \tag{26}$$

By (11) and (18), $\sum_{T \in I} wt(T) \leq M - \sum_{T \in A \cup B} wt(T)$, which by (26) implies that $\sum_{T \in I} wt(T) < h$. □

We next determine the largest number of MIs and SMIs that may be scheduled at t_h, for $\sum_{T \in I} wt(T) < h$ to hold. We begin with a lemma that gives the latest time that a subtask of a task in B may be scheduled, if $a_1^0 > 0$.

Lemma 26 *Subtask U_j defined by Def. 6 is scheduled no later than $t_h - 3$,* i.e., *$t_b \leq t_h - 3$.*

Proof: By Def. 5, $t_b < t_h$. Let T_i be an MI scheduled at t_h. Then, $d(T_i) = t_h$, and $S(T, t_h - 1) = 0$, from the definition of an MI. Hence, T_i is eligible at $t_h - 1$. Because T_i is not scheduled at $t_h - 1$, we can conclude that there are no holes in $t_h - 1$ and that the priority of every subtask V_k scheduled at $t_h - 1$ is at least that of T_i, *i.e.,*

$$(\forall V_k : S(V_k, t_h - 1) = 1 :: d(V_k) \leq t_h). \tag{27}$$

By Def. 4, the successor of U_j is not eligible before $t_h + 1$. Hence, the latest time before t_h that a subtask of U may be scheduled is given by the latest time that U_j may be scheduled. Also, from Lemma 16, $d(U_j) = t_h + 1$. Hence, by (27), U_j is not scheduled at $t_h - 1$, *i.e.,* $t_b < t_h - 1$. To complete the proof, we show that $t_b \neq t_h - 2$.

Assume to the contrary that $t_b = t_h - 2$. By (27), (4), and (5),

$$(\forall V_k : S(V_k, t_h - 1) = 1 :: r(V_k) \leq t_h - 2). \tag{28}$$

Because U is scheduled at $t_h - 2$ but not $t_h - 1$ and there are no holes in $t_h - 1$, there is at least one subtask W_l scheduled at $t_h - 1$ whose predecessor is not scheduled at $t_h - 2$. By (28), this implies that W_l is eligible at $t_h - 2$. By (27) and Lemma 16, $d(W_l) < d(U_j)$. Hence, EPDF should have scheduled W_l at $t_h - 2$ in preference to U_j, which is a contradiction. Hence, $t_b < t_h - 2$. □

From the above lemma, we have the following assertion.

(A4) $a_1^0 > 0 \;\Rightarrow\; t_b \leq t_h - 3$.

The lemma that follows will be used to identify tasks that are inactive at t_h.

Lemma 27 *Let T be a task that is not scheduled at t_h. If T is scheduled in any of the slots in $[t_b+1, t_h-1]$ then T is in I.*

Proof: T clearly is not in A. Because T is scheduled in $[t_b+1, t_h-1]$ T is also not in B, by the definition of t_b. $\square$

In the rest of this subsection, we let $s = t_h - t_b - 1$, the number of slots in $[t_b+1, t_h-1]$.

We now determine a lower bound on the number of subtasks of tasks in I that may be scheduled in $[t_b+1, t_h-1]$ as a function of a_1^0, a_1^1, h, and s. For this purpose, we assign subtasks scheduled in $[t_b, t_h-1]$ to processors in a systematic way. This assignment is only for accounting purposes; subtasks are not bound to processors in the actual schedule.

Processor groups. The assignment is based on the tasks scheduled at t_h. We first divide the M processors into four groups, P_1, P_2, P_3, and P_4, based on the tasks scheduled at t_h, as follows.

P_1 By Lemma 17, there is at least one subtask W_l such that $e(W_l) \leq t_b$ and $S(W,t) = 0$, for t in $[t_b+1, t_h-1]$. We assign one such subtask to the lone processor in this group. Hence, $|P_1| = 1$. We let $\mathcal{T}_{h1}$ denote the single task assigned to P_1 at t_h.

P_2 The h processors that are idle at t_h comprise this group. Thus, $|P_2| = h$.

P_3 This group consists of the $a_1^0 + a_1^1$ processors on which the a_1^0 MIs and a_1^1 SMIs are scheduled. We let $\mathcal{T}_{h3}$ denote the subset of all tasks scheduled on processors in P_3 at t_h.

P_4 Processors not assigned to P_1, P_2, or P_3 belong to this group. $\mathcal{T}_{h4}$ denotes the subset of all tasks scheduled on P_4 at t_h.

Subtask assignment in $[t_b, t_h-1]$. We assign subtasks scheduled in $[t_b, t_h-1]$ to processors by the following rules. Tasks in $\mathcal{T}_{h3}$ and $\mathcal{T}_{h4}$ are assigned to the same processor that they are assigned to in t_h, in every slot in which they are scheduled in $[t_b, t_h-1]$. Subtasks of tasks not in $\mathcal{T}_{h3}$ or $\mathcal{T}_{h4}$ may be assigned to any processor.

The next three lemmas will be used to bound the number of subtasks of tasks in I scheduled in $[t_b+1, t_h-1]$. These lemmas assume that the assignment of subtasks to processors in $[t_b+1, t_h-1]$ follows the rules described above.

Lemma 28 *The tasks of all s subtasks scheduled in $[t_b+1, t_h-1]$ on each processor in P_1 or P_2 are inactive at t_h.*

Proof: By our assignment of subtasks to processors, tasks assigned to processors in P_1 or P_2 in $[t_b+1, t_h-1]$ are not scheduled at t_h. By Lemma 17, there are no holes in $[t_b+1, t_h-1]$. Hence, by Lemma 27, all s subtasks assigned to a processor in P_1 or P_2 in the s slots in $[t_b+1, t_h-1]$ are subtasks of tasks in I. $\square$

Lemma 29 *At least one of the subtasks assigned to each processor in P_3 in $[t_b+1, t_h-1]$ is a subtask of a task in I.*

Proof: Let P_{3i} be any processor in P_3, and let T_i be the subtask scheduled on P_{3i} at t_h. Then, T_i is either an MI or an SMI. In the former case, $S(T, t_h-1) = 0$, and in the latter, $S(T, t_h-2) = 0$. By (A4), $t_b \leq t_h - 3$. Thus, there is at least in one slot in $[t_b+1, t_h-1]$ in which a subtask of a task V other than T is assigned to P_3. By our subtask assignment, V is not scheduled at t_h; thus, by Lemma 27, $V \in I$. $\square$

Lemma 30 *The number of subtasks of tasks in I that are scheduled in $[t_b+1, t_h-1]$ is at least $s \cdot (h+1) + (a_1^0 + a_1^1)$.*

Proof: Follows from Lemmas 28 and 29. $\square$

Lemma 31 *The sum of the weights of the tasks in I is at least $(h+1) \cdot \frac{s}{s+2} + \frac{a_1^0 + a_1^1}{s+2}$.*

Proof: Let V_k be a subtask of a task V in I that is scheduled in $[t_b+1, t_h-1]$. Then, by Def. 3, $d(V_k) \leq t_h$. By Lemma 16 and Def. 6, $d(U_j) > t_h$ and U_j is scheduled at t_b. Because V_k with an earlier deadline than U_j is scheduled later than t_b, either $r(V_k) \geq t_b + 1$ or V_k's predecessor V_j, where $j < k$, is scheduled at t_b. In the latter case, by (A2), $tardiness(V_j) \leq 1$, and hence, $d(V_j) \geq t_b$, which by (4) and (5) implies $r(V_k) \geq t_b - 1$. Thus, we have the following.

$$\begin{aligned}&(\forall V_k : V \in I :: \\ &\qquad (\exists u :: u \in [t_b+1, t_h-1] \ \wedge \ S(V_k, u) = 1 \\ &\qquad \Rightarrow \ (r(V_k) \geq t_b - 1 \ \wedge \ d(V_k) \leq t_h)) \qquad (29)\end{aligned}$$

We next show that if $V.n$ is the number of subtasks of V scheduled in $[t_b+1, t_h-1]$, then $wt(V) \geq \frac{V.n}{s+2}$. Let V_k and V_l denote the first and last subtasks of V scheduled in $[t_b+1, t_h-1]$. Then $r(V_k) \geq t_b - 1$ and $d(V_l) \leq t_h$, by (29). Hence,

$$d(V_l) - r(V_k) \leq t_h - t_b + 1 = s + 2. \tag{30}$$

Also, $r(V_k) = \left\lfloor \frac{k-1}{wt(V)} \right\rfloor + \theta(V_k)$ and $d(V_l) = \left\lceil \frac{l}{wt(V)} \right\rceil + \theta(V_l)$, by (4) and (5). Therefore,

$$\begin{aligned} & d(V_l) - r(V_k) \\ & = \left\lceil \frac{l}{wt(V)} \right\rceil - \left\lfloor \frac{k-1}{wt(V)} \right\rfloor + \theta(V_l) - \theta(V_k) \\ & \geq \left\lceil \frac{l}{wt(V)} \right\rceil - \left\lfloor \frac{k-1}{wt(V)} \right\rfloor \qquad (31) \\ & \qquad \text{(from } l > k \text{ and (6)).} \end{aligned}$$

From (30) and (31), we have $\left\lceil \frac{l}{wt(V)} \right\rceil - \left\lfloor \frac{k-1}{wt(V)} \right\rfloor \leq s+2$, which implies $\frac{l}{wt(V)} - \frac{k-1}{wt(V)} \leq s+2$, or

$$wt(V) \geq \frac{l-k+1}{s+2}. \tag{32}$$

Note that $V.n = l-k+1$ if V is periodic, and $V.n \leq l-k+1$, if V is GIS. Therefore, $wt(V) \geq \frac{V.n}{s+2}$, from which we have $\sum_{W \in I} wt(W) \geq \sum_{W \in I} \frac{W.n}{s+2} \geq (h+1) \cdot \frac{s}{s+2} + \frac{a_1^0 + a_1^1}{s+2}$, by Lemma 30. □

Lemma 32 *If $LAG(\tau, t_h+1) > LAG(\tau, t_h)$ and $a_1^0 \geq 1$, then $a_1^0 + a_1^1 \leq \min(2h-3, M-h-1)$.*

Proof: By Lemma 25, if $LAG(\tau, t_h+1) > LAG(\tau, t_h)$, then $\sum_{V \in I} wt(V) < h$. By Lemma 31, $(h+1) \cdot \frac{s}{s+2} + \frac{a_1^0 + a_1^1}{s+2} \leq \sum_{V \in I} wt(V)$. Therefore, $(h+1) \cdot \frac{s}{s+2} + \frac{a_1^0 + a_1^1}{s+2} < h$, which implies that $a_1^0 + a_1^1 < 2h - s$. Because $s \geq 2$, $2h - s \leq 2h - 2$. Therefore,

$$a_1^0 + a_1^1 < 2h - 2. \tag{33}$$

Also, there are h holes in t_h, and by Lemma 17, $a_0 \geq 1$. Therefore,

$$a_1^0 + a_1^1 \leq M - h - 1. \tag{34}$$

(33) and (34) imply that $a_1^0 + a_1^1 \leq \min(2h-3, M-h-1)$. □

We now conclude Case B by establishing the following.

Lemma 33 *If $a_1^0 > 0$, then $LAG(\tau, t_h+1) < M+1$.*

Proof: From (24),

$$\begin{aligned} & LAG(\tau, t_h+1) \\ & \quad < a_0 \cdot wt + a_1^0 \cdot 2wt + \\ & \qquad a_1^1 \cdot (3wt - 1) + a_1^2 \cdot (4wt - 2) \\ & \quad \leq a_0 \cdot wt + 2wt \cdot (a_1^0 + a_1^1) + \\ & \qquad a_1^2 \cdot (4wt - 2) \qquad \text{(because } wt < 1\text{).} \end{aligned} \tag{35}$$

By Lemma 32, if $LAG(\tau, t_h+1) > LAG(\tau, t_h)$, then $a_1^0 + a_1^1 \leq \min(2h-3, M-h-1)$. Because the lag bounds for tasks in $A_1^0 \cup A_1^1$ are higher than those for the other tasks, $LAG(\tau, t_h+1)$ is maximized when $a_1^0 + a_1^1 = \min(2h-3, M-h-1)$. We assume this is the case. Note that

$$\min(2h-3, M-h-1) = \begin{cases} 2h-3, & h \leq \frac{M+1}{3} \\ M-h-1, & \text{otherwise.} \end{cases} \tag{36}$$

Based on (36), we consider two cases.

Case 1: $h > \frac{M+1}{3}$.
For this case, $a_1^0 + a_1^1 = M - h - 1$, and hence, $a_0 + a_1^2 = M - h - (a_1^0 + a_1^1) = 1$, which, by (25) implies that $M \geq 3$. Because, by Lemma 17, $a_0 > 0$, we have $a_0 = 1$, and hence, $a_1^2 = 0$. Therefore, by (35), $LAG(\tau, t_h+1) < a_0 \cdot wt + 2wt \cdot (a_1^0 + a_1^1) = wt + 2wt \cdot (M-h-1) \leq wt + 2wt \cdot \left(\frac{2M}{3} - 2\right)$. If $M+1 \leq LAG(\tau, t_h+1)$, then $wt + 2wt \cdot \left(\frac{2M}{3} - 2\right) \geq M+1$, which implies that $wt \geq \frac{3M+3}{4M-9}$, which is greater than $\frac{3}{4}$, for all $M \geq 3$. This violates (C) and (D).

Case 2: $h \leq \frac{M+1}{3}$.
For this case, letting $a_1^0 + a_1^1 = 2h - 3$, we have $a_1^2 = M - h - (a_0 + a_1^0 + a_1^1) = M - 3h - a_0 - 3$. Therefore, by (35), $LAG(\tau, t_h+1) < a_0 \cdot wt + 2wt \cdot (a_1^0 + a_1^1) + (4wt - 2) \cdot a_1^2 = a_0 \cdot wt + 2wt \cdot (2h-3) + (4wt-2)(M - 3h - a_0 + 3)$. If $M+1 \leq LAG(\tau, t_h+1)$, then $a_0 \cdot wt + (2h-3) \cdot 2 \cdot wt + (M - 3h - a_0 + 3) \cdot (4 \cdot wt - 2) \geq M+1$, which implies that $wt \geq \frac{3M - 6h - 2a_0 + 7}{4M - 8h - 3a_0 + 6}$. If $LAG(\tau, t_h+1) > M+1$, then $wt > f = \frac{3M - 6h - 2a_0 + 7}{4M - 8h - 3a_0 + 6}$. Because $a_1^0 > 0$, we have $a_1^0 + a_1^1 = 2h - 3 > 0$. This implies that $h > \frac{3}{2}$; hence, because h is integral, $h \geq 2$. Therefore, M, h, and a_0 in f are constrained by $M > h + a_0 \geq 2$, $2 \leq h \leq \frac{M+1}{3}$, and $a_0 > 0$. The first constraint is from (25) and because $a_1^0 + a_1^1 > 1$, and the last constraint is by Lemma 17. It can be shown that f is minimized when $h = 2$ and $a_0 = 1$. In this case, $f = \frac{3M-7}{4M-13} > \frac{3}{4}$, for all $M > 2$. Hence, $wt > \frac{3}{4}$, which is a violation of (C) and (D). □

3.4 Case C ($A_1^0 = \phi$; $A_1^1 \neq \phi$)

In this case, *no* miss initiators are scheduled at t_h, but *at least one* successor of miss initiator is. An upper bound on $LAG(\tau, t_h+1)$ for this case is given below. From (24),

$$\begin{aligned}
&LAG(\tau, t_h+1) \\
&\quad < a_0 \cdot wt + a_1^0 \cdot 2wt + \\
&\qquad a_1^1 \cdot (3wt-1) + a_1^2 \cdot (4wt-2) \\
&\quad = a_0 \cdot wt + a_1^1 \cdot (3wt-1) + a_1^2 \cdot (4wt-2) \\
&\qquad\qquad \text{(because } a_1^0 = 0\text{)} \\
&\quad = a_0 \cdot wt + a_1^1 \cdot (3wt-1) + \\
&\qquad (M-h-a_0-a_1^1) \cdot (4wt-2) \qquad \text{(by (25))}. \quad (37)
\end{aligned}$$

The following lemma establishes the sufficiency of (C) for this case.

Lemma 34 *If* $A_1^0 = \phi$, *then* $LAG(\tau, t_h+1) < M+1$.

From (37) and $A_1^0 = \phi$,

$$\begin{aligned}
&LAG(\tau, t_h+1) \\
&\quad = a_0 \cdot wt + a_1^1 \cdot (3wt-1) + \\
&\qquad (M-h-a_0-a_1^1) \cdot (4wt-2) \\
&\quad = a_0 \cdot wt + (M-h-a_0) \cdot (3wt-1) \\
&\qquad \text{(because } wt < 1 \Rightarrow (4wt-2 < 3wt-1)\text{)}.
\end{aligned}$$

$LAG(\tau, t_h+1) \geq M+1$ implies that $a_0 \cdot wt + (M-h-a_0) \cdot (3wt-1) > M+1$, which implies that $wt > f = \frac{2M-h-a_0+1}{3M-3h-2a_0}$. It can be shown that f is minimized when $h = a_0 = 1$. Because $a_1^1 > 0$, this implies that $M \geq 3$, by (25). For $h = a_0 = 1$, $f = \frac{2M-1}{3M-5} > \frac{2}{3}$, for all $M \geq 2$. Thus, (C) is violated. □

In the rest of this section, we establish the sufficiency of (D) for this case, and hence, assume that for any subtask T_i, $e(T_i) \geq r(T_i)$, which by (8), implies the following.

$$(\forall T_i :: e(T_i) = r(T_i)) \qquad (38)$$

We prove a series of lemmas that contradict the existence of t_h, if (D) holds. The first lemma gives a lower bound on a_0, for $LAG(\tau, t_h+1)$ to be less than $M+1$.

Lemma 35 *If* $a_0 \geq a_1^1 - \left(\frac{14h}{3}+5\right)$, *then* $LAG(\tau, t_h+1) < M+1$.

Proof: From (37), $LAG(\tau, t_h+1) < a_0 \cdot wt + a_1^1 \cdot (3wt-1) + (M-h-a_0-a_1^1) \cdot (4wt-2)$. If $M+1 \leq LAG(\tau, t_h+1)$, then $a_0 \cdot wt + a_1^1 \cdot (3wt-1) + (M-h-a_0-a_1^1) \cdot (4wt-2) > M+1$, which implies that $wt > f = \frac{3M-2h-a_1^1-2a_0+1}{4M-4h-a_1^1-3a_0}$. Because (D) holds, $wt \leq \frac{11}{15}$, which implies that $f < \frac{11}{15}$, or $a_0 < a_1^1 - (\frac{M}{3} - \frac{a_1^1}{3} + \frac{14h}{3} + 5) < a_1^1 - (\frac{14h}{3} + 5)$ (because $M > a_1^1$). Taking the contrapositive, we have the condition stated in the lemma. □

The previous lemma gave a lower bound on a_0 as a function of h, while the next lemma gives a bound as a function of M and h.

Lemma 36 *If* $LAG(\tau, t_h+1) \geq M+1$, *then* $a_0 < \frac{3M-18h-15}{7}$.

Proof: From (37),

$$\begin{aligned}
&LAG(\tau, t_h+1) \\
&\quad < a_0 \cdot wt + a_1^1 \cdot (3wt-1) \\
&\qquad +(M-h-a_0-a_1^1) \cdot (4wt-2) \\
&\quad < a_0 \cdot wt + a_1^1 \cdot (3wt-1) \\
&\qquad +(M-h-a_0-a_1^1) \cdot (3wt-1) \\
&\qquad\qquad \text{(because } wt < 1\text{)} \\
&\quad = a_0 \cdot wt + (M-h-a_0) \cdot (3wt-1).
\end{aligned}$$

If $LAG(\tau, t_h+1) \geq M+1$, then $a_0 \cdot wt + (M-h-a_0) \cdot (3wt-1) > M+1$, which implies that $wt > \frac{2M+1-h-a_0}{3M-3h-2a_0}$. Because (D) holds, $wt \leq \frac{11}{15}$, which implies that $\frac{2M+1-h-a_0}{3M-3h-2a_0} < \frac{11}{15}$, from which we have $a_0 < \frac{3M-18h-15}{7}$. □

t_b, the latest time that a subtask of a task in B is scheduled, can be as late as $t_h - 1$ for this case, *i.e.*, $t_b \leq t_h - 1$. If $t_b < t_h - 1$, then using reasoning similar to that of Lemma 26, it can be shown that $t_b \leq t_h - 4$. Case C can then be reasoned in exactly the same way as Case B was reasoned. Hence, assume $t_b = t_h - 1$ for the rest of this section.

In the next two lemmas, we derive a lower bound on the number of tasks that are inactive at $t_h - 1$ and t_h.

Lemma 37 *Let* V_k *be a subtask scheduled at* $t_h - 2$. *Then,* $d(V_k) \leq t_h - 1$, *and there are no holes in* $t_h - 2$.

Proof: Because a_1^1 SMIs are scheduled at t_h, at least a_1^1 subtasks that are scheduled at $t_h - 1$ are MIs (by the definition of an SMI). Let X_k be one such MI scheduled at t_h-1. Then, $d(X_k)=t_h-1$, and X is not scheduled at t_h-2 (by the definition of an MI). Therefore, every subtask scheduled at t_h-2 has its deadline at or before t_h-1, and there are no holes in t_h-2 (otherwise, EPDF would schedule X_k in t_h-2). □

Lemma 38 *If $LAG(\tau,t_h+1)\geq M+1$, then at least $\frac{14h}{3}+5$ tasks scheduled at t_h-2 are inactive in slots t_h-1 and t_h.*

Proof: If $a_1^1 \leq a_0$, then by Lemma 35, $LAG(\tau,t_h+1) < M+1$. Therefore, assume $a_1^1 > a_0$. We first show that at least $a_1^1-a_0$ tasks scheduled at t_h-2 are inactive at t_h-1 and t_h, as follows.

By the assumption for this case ($a_1^0=0$) and by the definition of set A, a task scheduled at t_h is in A_0, A_1^1, or A_1^2, *i.e.*

$$S(T,t_h)=1 \;\Rightarrow\; T\in A_0\cup A_1^1\cup A_1^2. \tag{39}$$

The following holds for a task in A_1^1 by the definition of an SMI.

$$\begin{aligned}(\forall T\in A_1^1 :: S(T,t_h) = \\ S(T,t_h-1)=1 \;\wedge\; S(T,t_h-2)=0).\end{aligned} \tag{40}$$

By Lemma 37, there are no holes in t_h-2, which along with (40) implies that there are at least a_1^1 tasks scheduled at t_h-2, that are not scheduled in t_h-1. Let τ_s be the set of all such tasks. Then, by the previous argument, the following hold.

$$\tau_s=\{T\in\tau : S(T,t_h-2)=1\wedge S(T,t_h-1)=0\}, \tag{41}$$

and

$$|\tau_s|\geq a_1^1. \tag{42}$$

By Lemma 37, the deadline of every subtask scheduled at t_h-2 is at or before t_h-1. Let X be any task in τ_s whose subtask X_k is scheduled at t_h-2. Therefore,

$$S(X_k,t_h-2)=1 \;\wedge\; d(X_k)\leq t_h-1. \tag{43}$$

By Def. 3, if X is active at t_h-1, then there exists an X_l, where $l>k$, such that $e(X_l)\leq t_h-1 \;\wedge\; d(X_l)\geq t_h$. By (43), X_l is not scheduled before t_h-1, and by (41), X_l is not scheduled at t_h-1. The hole in t_h implies that X_l, if present, should be scheduled at t_h. Hence, if X is not scheduled at t_h, then we can conclude that subtask X_l of X does not exist, and that X is inactive at t_h-1. By a similar argument it can be shown that X is inactive at t_h also. Thus, we have the following.

$$X\in\tau_s\wedge S(X,t_h)=0 \;\Rightarrow\; X \text{ is inactive at } t_h-1 \text{ and } t_h. \tag{44}$$

Having shown that X is inactive if not scheduled at t_h, we next show that if X is scheduled at t_h, then X is in A_0.

Because X is in τ_s, by (41) and (40), X is not in A_1^1. By the definition of A_1^2,

$$\begin{aligned}(\forall T\in A_1^2 :: S(T,t_h) &= S(T,t_h-1)\\ &= S(T,t_h-2)=1),\end{aligned} \tag{45}$$

which along with (41) implies that X is not in A_1^2. Therefore, by (39), X is in A_0. Thus, at most a_0 tasks in τ_s can be scheduled at t_h, which by (42) and (44) implies that at least $a_1^1-a_0$ tasks scheduled at t_h-2 are inactive at t_h-1 and t_h.

If $LAG(\tau,t_h+1)\geq M+1$, then by Lemma 35, $a_0 < a_1^1-(\frac{14h}{3}+5)$. Hence, $a_1^1-a_0$, the number of tasks that are inactive at t_h-1 and t_h, is at least $\frac{14h}{3}+5$. □

Definition 7: I' denotes the set of tasks scheduled at t_h-2 that are inactive at t_h-1 and t_h.

In the next three lemmas, we bound the sum of the weights of the tasks in I' from above, and use it to determine the latest time that the critical subtask at t_h of a task in B may be scheduled, if it is not scheduled at t_h-1.

Lemma 39 *If $LAG(\tau,t_h+1) > LAG(\tau,t_h-1)$, then $\sum_{T\in I'} wt(T) < h/2$.*

Proof: From (16), $LAG(\tau,t_h+1)$ can be expressed in terms of $LAG(\tau,t_h-1)$ as follows.

$$\begin{aligned}
&LAG(\tau,t_h+1)\\
&= LAG(\tau,t_h-1)-\sum_{T\in\tau}(S(T,t_h-1)+S(T,t_h))\\
&\quad +\sum_{T\in\tau}(share(T,t_h-1)+share(T,t_h))\\
&= LAG(\tau,t_h-1)-\sum_{T\in\tau}(S(T,t_h-1)+S(T,t_h))\\
&\quad +\sum_{T\in\tau-I'}(share(T,t_h-1)+share(T,t_h))\\
&\qquad\qquad \text{(tasks in } I' \text{ are inactive at } t_h-1 \text{ and } t_h)\\
&= LAG(\tau,t_h-1)-(2M-h)\\
&\quad +\sum_{T\in\tau-I'}(share(T,t_h-1)+share(T,t_h))\\
&\qquad\qquad \text{(there are } h \text{ holes in } t_h\text{, and by Lemma 17,}\\
&\qquad\qquad \text{there are no holes in } t_h-1)\\
&\leq LAG(\tau,t_h-1)+2\sum_{T\in\tau-I'}wt(T)-(2M-h)\\
&\leq LAG(\tau,t_h-1)+2(M-\sum_{T\in I'}wt(T))-(2M-h)
\end{aligned}$$

(from (11))

$$= LAG(\tau, t_h - 1) - 2\sum_{T \in I'} wt(T) + h.$$

If $LAG(\tau, t_h + 1) > LAG(\tau, t_h - 1)$, then $h - 2\sum_{T \in I'} wt(T) > 0$, which implies that $\sum_{T \in I'} wt(T) < h/2$. □

Lemma 40 *If $LAG(\tau, t_h + 1) < LAG(\tau, t_h - 1)$, then there is at least one task V in I', with $wt(V) < 1/9$.*

Proof: By Lemma 38, $|I'| \geq \frac{14h}{3} + 5$, and by Lemma 39, $\sum_{T \in I'} wt(T) < h/2$, which together imply that the average weight of a task in I' is less than 3/28, which is less than 1/9. Hence, the weight of at least one task in I' is less than 1/9. □

Lemma 41 *If the critical subtask at t_h, V_k, of a task V in B is not scheduled at $t_h - 1$, then V_k is scheduled at or before $t_h - 10$.*

Proof: We prove this lemma by contradiction. Assume that V_k is scheduled at t, where $t_h - 10 < t < t_h - 1$. We first show that $t \neq t_h - 2$.

By Lemma 16,

$$d(V_k) = t_h + 1, \quad (46)$$

and by Lemma 37, we have

$$(\forall T_i :: S(T_i, t_h - 2) = 1 \Rightarrow d(T_i) \leq t_h - 1). \quad (47)$$

Therefore,

$$S(V_k, t_h - 2) = 0. \quad (48)$$

We next show that $t \notin [t_h - 9, t_h - 3]$. By Lemma 40 and Def. 7, we have

$$\left(\exists T_i : S(T_i, t_h - 2) = 1 \ \wedge \ wt(T) < \frac{1}{9}\right). \quad (49)$$

Let X_i be a subtask of a task X in I' scheduled at $t_h - 2$ with $wt(X) < \frac{1}{9}$. Then, by Lemma 1, $|w(X_i)| > \left\lceil \frac{1}{\frac{1}{9}} \right\rceil = 9$, *i.e.*,

$$|w(X_i)| \geq 10. \quad (50)$$

(47), (50), and (10) imply that

$$r(X_i) \leq t_h - 11. \quad (51)$$

If X_h is the predecessor of X_i, then by (4), (5), and (51),

$$d(X_h) \leq t_h - 10. \quad (52)$$

By (A2), $tardiness(X_h) \leq 1$, which by (52) implies that X_h is scheduled no later than $t_h - 10$. Thus, X_i can be scheduled in every slot in $[t_h - 9, t_h - 3]$. If t is in $[t_h - 9, t_h - 3]$, then by (46) and (47), it implies that V_k with a lower priority than X_i is scheduled at t, which is a contradiction. Therefore, V_k is scheduled at or before $t_h - 10$. □

Definition 8: In the rest of this section, the following additional notation shall be used.

B' : Includes $T \in B$, iff the critical subtask at t_h of T is scheduled before $t_h - 1$.
t'_b : Latest time at which a critical subtask at t_h of any task in B' is scheduled.
B'' : Includes $T \in B$, iff T is not scheduled at t'_b and the critical subtask at t_h, T_k, of T is scheduled at $t_h - 1$, and is eligible at t'_b.
b'' : Number of tasks in $B'' = |B''|$.
A''_0 : Includes $T \in A_0$, iff T is not scheduled at t'_b and T's subtask scheduled at t_h is eligible at t'_b.
a''_0 : Number of tasks in $A''_0 = |A''_0|$.
I'' : Includes $T \in I$, iff T is active at $t_h - 1$ and its critical subtask at $t_h - 1$ is scheduled before $t_h - 1$.
i'' : Number of tasks in $I'' = |I''|$.

The above definitions imply that

$$B' \cup B'' \cup (B - B' - B'') = B \quad (53)$$

$$B' \cap B'' = B'' \cap (B - B' - B'') = (B - B' - B'') \cap B' = \phi \quad (54)$$

From Lemma 41, we have the following.

$$t'_b \leq t_h - 10. \quad (55)$$

Lemma 42 *There are no holes in $[t'_b + 1, t_h - 1]$.*

Proof: The proof is by contradiction. Assume that there is a hole at t, where $t'_b + 1 \leq t \leq t_h - 1$. Let V_k be a subtask that is critical at t_h of a task V in B' that is scheduled at t'_b. We show that if there is a hole anywhere in the interval specified, then V_k can be removed without affecting the tardiness of subtasks scheduled after the hole. Let τ' be the system obtained by removing V_k from τ, and let S' be the schedule for τ'. Let $\Delta_1, \Delta_2, \ldots, \Delta_n$

be the chain of displacements caused by removing V_k, where $\Delta_i = \langle X^{(i)}, t_i, X^{(i+1)}, t_{i+1}\rangle$, $X^{(1)} = V_k$ and $t_1 = t'_b$. By Lemma 11, $t_{i+1} > t_i$, for $1 \leq i < n$. Hence, the priority of $X^{(i)}$ is greater than or equal to the priority of $X^{(i+1)}$, for $1 \leq i < n$, which in turn implies that $d(X^{(i)}) \leq d(X^{(i+1)}$. By Lemma 16, $d(V_k) = t_h + 1$, and hence,

$$(\forall i : 1 \leq i \leq n :: d(X^{(i)}) \geq t_h + 1). \tag{56}$$

We now show that the chain of displacements does not extend beyond t. Assume to the contrary that it extends beyond t, and let Δ_k be the first displacement in the chain such that

$$t_{k+1} > t. \tag{57}$$

Then, $t_k \leq t$. Since Δ_k is valid,

$$e(X^{(k+1)}) \leq t_k \leq t. \tag{58}$$

Because there is a hole in t, $X^{(k+1)}$ should be scheduled at t in S rather than at $t_{k+1} > t$. Therefore, $t_k = t$. Because there is a hole at t, by Lemma 12, $X^{(k+1)}$ is the successor of $X^{(k)}$. Therefore, by (56), (4), and (5), $r(X^{(k+1)}) \geq t_h$, which by Lemma 4(a) implies that $e(X^{(k+1)}) \geq \min(t_h, t_{k+1})$. This is in contradiction to (58), which implies that our assumption in (57) is false. In other words, no subtask scheduled to the right of t in S gets shifted left in S'. Therefore, the tardiness of subtasks scheduled after t remain the same in S', which contradicts (T2). Thus, there are no holes in $[t'_b + 1, t_h - 1]$. □

In the rest of this section, we first determine a lower bound on the weights of the tasks in B', and then use it to determine a lower bound on the sum of the lags of the tasks scheduled at t'_b. Finally, we show that if $LAG(\tau, t_h + 1) \geq M + 1$, then the latter bound implies a violation of (D). We begin by deriving an upper bound on the weights of the tasks in B'' and A''_0.

Lemma 43 *The weight of a task V in B'' or A''_0 is at most* $1/9$.

Proof: We give the proof for V in B''. The proof for when V is in A''_0 is similar.

Let V_k be the critical subtask at t_h of V. Then, because V is in B'', by Def. 8, $e(V_k) \leq t'_b$. Because V_k is scheduled at $t_h - 1$, by Lemma 4(a),

$$\begin{aligned} r(V_k) &\leq t'_b \\ &\leq t_h - 10, \qquad \text{(from 55)} \end{aligned} \tag{59}$$

and

$$d(V_k) = t_h + 1 \qquad \text{(by Lemma 16).} \tag{60}$$

The above two inequalities imply that $|w(V_k)| = d(V_k) - r(V_k) \geq 11$. Hence, by Lemma 1,

$$\begin{aligned} &\left\lceil \frac{1}{wt(T)} \right\rceil \geq 10 \\ &\Rightarrow \quad \frac{1}{wt(V)} + 1 \geq 10 \\ &\Rightarrow \quad \frac{1}{wt(V)} \geq 9 \\ &\Rightarrow \quad wt(V) \leq \frac{1}{9}. \end{aligned}$$

□

The next lemma is concerned with the deadline of the predecessor of a critical subtask at t_h of a task in B, and is used in later lemmas.

Lemma 44 *Let V_k be the critical subtask at t_h of a task V in B, and let V_h, where $h < k$, be V_k's predecessor in τ. Then, $d(V_h) \leq t_h - 1$.*

Proof: By Lemma 16,

$$d(V_k) = t_h + 1, \tag{61}$$

which by (5) implies that $d(V_h) \leq t_h$. Hence, it is sufficient to show that $d(V_h) \neq t_h$. Assume to the contrary that

$$d(V_h) = t_h. \tag{62}$$

Then, (61), (62), (4), and (5) imply that $r(V_k) = t_h - 1$. Hence,

$$|w(V_k)| = d(V_k) - r(V_k) = 2. \tag{63}$$

By Lemma 14, V_k is scheduled at or before $t_h - 1$. Therefore, V_h is not scheduled at $t_h - 1$. We next show that V_h is not scheduled at $t_h - 2$ or $t_h - 3$ either.

By Lemma 37, the deadline of every subtask scheduled at $t_h - 2$ is at most $t_h - 1$. This, by (62) implies that V_h is not scheduled at $t_h - 2$. Let X_i be a subtask of a task X in I' that is scheduled at $t_h - 2$, with $wt(X) < \frac{1}{9}$. By Lemma 40, such a task X exists, which by (4) and (5) implies that $r(X_i) \leq t_h - 11$. It is easy to show that X_i's predecessor, if it exists, cannot be scheduled any later than $t_h - 10$, and hence, that no subtask with lower priority than X_i can be scheduled in any slot in $[t_h - 9, t_h - 3]$. In particular, V_h is scheduled prior to $t_h - 9$, which by (38) implies that $r(V_h) \leq t_h - 10$, which by (62) implies that

$$|w(V_h)| \geq 10. \quad (64)$$

By (9), the difference between the lengths of the windows of any two subtasks of a task is at most one, which is in contradiction to (63) and (64). Therefore, our assumption is false, and $d(V_h) \leq t_h - 1$. □

Lemma 45 *The number of tasks that are scheduled at $t_h - 1$, and are not scheduled at t_h is at most $h + a_0$.*

Proof: By the assumption of this case, tasks scheduled at t_h are in $A_0 \cup A_1^1 \cup A_1^2$. Because there are h holes in t_h, $M = h + a_0 + a_1^1 + a_1^2$. Every task in A_1^1 or A_1^2 is also scheduled at $t_h - 1$ (by the definition of these sets). Therefore, the number of tasks scheduled at t_h, and not at $t_h - 1$ is at most $M - a_1^1 - a_1^2 = h + a_0$. □

In the next lemma, we determine a lower bound on the excess allocation that τ receives in S in time slots $t_h - 1$ and t_h in comparison to the ideal system. This will be used in turn to determine a lower bound on the sum of the weights of the tasks in $B' \cup I''$.

Lemma 46 *Let C denote the set of all tasks that are scheduled at $t_h - 1$, or t_h, or both. Then, the difference between the total service $2M-h$ provided by S in slots $t_h - 1$ and t_h, and the share in the ideal system of the tasks in C in the same two slots, is at least $\frac{8M-8h}{15} - a_0 + \frac{7b''}{9} + \frac{168a_0''}{135}$. Further, if $LAG(\tau, t_h + 1) \geq M + 1$, then $\frac{8M-8h}{15} - a_0 > 0$.*

Proof: We first derive an upper bound on the total share in the ideal system, of all the tasks in C in slots t_h and $t_h - 1$. The set of all tasks that are scheduled at t_h is given by A and tasks in $C - A$ are scheduled at $t_h - 1$, but not at t_h. Thus, $C = (C - A) \cup A$, where $(C - A) \subseteq B \cup I$, by (18). Hence,

$$\begin{aligned} C - A &= (C - A) \cap (B \cup I) \\ &= ((C - A) \cap B) \cup ((C - A) \cap I). \quad (65) \end{aligned}$$

No task in B' is scheduled at $t_h - 1$ *i.e.*, is in $(C - A)$. Hence,

$$\begin{aligned} ((C - A) \cap B) &= (C - A) \cap (B - B') \\ &= (C - A) \cap ((B - B' - B'') \cup B'') \\ &= ((C - A) \cap (B - B' - B'')) \cup \\ &\quad ((C - A) \cap B''). \quad (66) \end{aligned}$$

Thus, from (65) and (66), we have

$$\begin{aligned} C - A &= ((C - A) \cap (B - B' - B'')) \cup \\ &\quad ((C - A) \cap B'') \cup ((C - A) \cap I). \quad (67) \end{aligned}$$

For every V that is in $(C - A) \cap I$, $share(V, t_h) = 0$. Hence, $share(V, t_h - 1) + share(V, t_h) = share(V, t_h - 1)$. From Lemma 2(c), $share(V, t_h - 1) \leq wt(V) < 1$, *i.e.*,

$$(\forall V \in (C-A) \cap I :: share(V, t_h - 1) + share(V, t_h) < 1.) \quad (68)$$

For every V in B, the share of V in $t_h - 1$ and t_h is determined as follows. Let V_k be the critical subtask at t_h of V, and let V_h be V_k's predecessor in τ. Then, by Lemma 44, $d(V_h) \leq t_h - 1$. Hence, no other subtask of V, except V_k, has a positive share in $t_h - 1$ or t_h in the ideal system. Thus, the share of V in $t_h - 1$ and t_h is given by the share of V_k in the same two slots. That is, $share(V, t_h - 1) + share(V, t_h) = f(V_k, t_h - 1) + f(V_k, t_h)$. If V is in B'', then $wt(V) \leq 1/9$ by Lemma 43, and hence, $share(V, t_h - 1) + share(V, t_h) = f(V_k, t_h - 1) + f(V_k, t_h) \leq 2/9$, from Lemma 2(d), *i.e.*

$$\begin{aligned} &(\forall V \in (C - A) \cap B'' :: \\ &\quad share(V, t_h - 1) + share(V, t_h) \leq 2/9.) \quad (69) \end{aligned}$$

On the other hand, if V is in $B - B' - B''$, then by Lemma 2(c) and Lemma 2(e), $share(V, t_h - 1) + share(V, t_h) = f(V_k, t_h - 1) + f(V_k, t_h) \leq 1$, *i.e.*,

$$\begin{aligned} &(\forall V \in (C - A) \cap (B - B' - B'') :: \\ &\quad share(V, t_h - 1) + share(V, t_h) \leq 1.) \quad (70) \end{aligned}$$

Similarly, $A = (A - A_0'') \cup A_0''$. By Lemma 43, weight of a task in A_0'' is at most 1/9, and hence by Lemma 2(c), the following holds.

$$\begin{aligned} &(\forall V \in A_0'' :: \\ &\quad share(V, t_h - 1) + share(V, t_h) \leq 2/9.) \quad (71) \end{aligned}$$

Therefore, the total share of all the tasks in C is bounded from above as follows.

$$\begin{aligned} &\sum_{T \in C} (share(T, t_h - 1) + share(T, t_h)) \\ &= \sum_{T \in A} (share(T, t_h - 1) + share(T, t_h)) + \\ &\quad \sum_{T \in C-A} (share(T, t_h - 1) + share(T, t_h) + \\ &= \sum_{T \in A - A_0''} (share(T, t_h - 1) + share(T, t_h)) + \\ &\quad \sum_{T \in A_0''} (share(T, t_h - 1) + share(T, t_h)) + \\ &\quad \sum_{T \in (C-A) \cap ((B - B' - B'') \cup I)} share(T, t_h - 1) + \end{aligned}$$

$$\sum_{T\in(C-A)\cap((B-B'-B'')\cup I)} share(T,t_h) + \sum_{T\in(C-A)\cap B''} (share(T,t_h-1)+share(T,t_h))$$

(from (67))

$$\leq 2\sum_{T\in A-A''_0} wt(T) + \frac{2a''_0}{9} + |(C-A)\cap((B-B'-B'')\cup I)| + \frac{2b''}{9}$$

(from Lemma 2(c), (71), (70), (68), and (69))

$$\leq 2\sum_{T\in A-A''_0} wt(T) + \frac{2a''_0}{9} + (a_0+h-b'') + \frac{2b''}{9}$$

(by Lemma 45 and $|B''| = b''$)

$$\leq 2(M-h-a''_0)\cdot wt + a_0 + \frac{2a''_0}{9} + h - \frac{7b''}{9}$$

(there are $M-h$ tasks scheduled at t_h with weight at most wt; a''_0 of those are in A''_0)

$$\leq 2(M-h-a''_0)\cdot\frac{11}{15} + a_0 + \frac{2a''_0}{9} + h - \frac{7b''}{9}$$

(by (D))

$$= 2(M-h)\cdot\frac{11}{15} + a_0 + h - \frac{7b''}{9} - \frac{168a''_0}{135}.$$

The difference between the actual allocation and the total ideal share of tasks in C is therefore given by $2M-h-\sum_{T\in C}(share(T,t_h-1)+share(T,t_h)) \geq 2M-h-\left(\frac{22(M-h)}{15}+a_0+h-\frac{7b''}{9}-\frac{168a''_0}{135}\right) = \frac{8M}{15}-\frac{8h}{15}-a_0+\frac{7b''}{9}+\frac{168a''_0}{135}$. By Lemma 36, if $LAG(\tau,t_h+1)\geq M+1$, then $a_0 < \frac{3M-18h}{7}-\frac{15}{7}$. Because $\frac{3M}{7}-\frac{18h}{7}-\frac{15}{7} < \frac{8M}{15}-\frac{8h}{15}$, $\frac{8M-8h}{15}-a_0 > 0$. □

Lemma 47 *If $LAG(\tau,t_h+1) \geq M+1 > LAG(\tau,t_h-1)$, then the sum of the weights of the tasks in $B'\cup I''$ is greater than $\frac{4M-4h}{15}-\frac{a_0}{2}+\frac{7b''}{18}+\frac{84a''_0}{135} > 0$.*

Proof: Expressing $LAG(\tau,t_h+1)$ in terms of $LAG(\tau,t_h-1)$ using (16), we have

$$\begin{aligned}
&LAG(\tau,t_h+1)\\
&= LAG(\tau,t_h-1) - \sum_{T\in\tau}(S(T,t_h-1)+S(T,t_h))\\
&\quad + \sum_{T\in\tau}(share(T,t_h-1)+share(T,t_h))\\
&= LAG(\tau,t_h-1) - \sum_{T\in\tau}(S(T,t_h-1)+S(T,t_h))\\
&\quad + \sum_{T\in C}(share(T,t_h-1)+share(T,t_h))\\
&\quad + \sum_{T\in\tau-C}(share(T,t_h-1)+share(T,t_h))
\end{aligned}$$

$(C = \{T\in\tau \wedge (S(T,t_h-1)=1 \vee S(T,t_h)=1)\})$

$$\leq LAG(\tau,t_h-1) - \left(\frac{8M-8h}{15}+\frac{7b''}{9}+\frac{168a''_0}{135}-a_0\right) + \sum_{T\in\tau-C}(share(T,t_h-1)+share(T,t_h))$$

(from Lemma 46)

$$= LAG(\tau,t_h-1) - \left(\frac{8M-8h}{15}+\frac{7b''}{9}+\frac{168a''_0}{135}-a_0\right) + \sum_{T\in B'\cup I''} share(T,t_h-1)+share(T,t_h).$$

(if a task has nonzero share in t_h-1 or t_h, and is not scheduled in either slot, then it is in B' or I'', by Def. 8)

$$\leq LAG(\tau,t_h-1) - \left(\frac{8M-8h}{15}+\frac{7b''}{9}+\frac{168a''_0}{135}-a_0\right) + \sum_{T\in B'\cup I''} 2wt(T), \qquad \text{(from Lemma 2(b))}.$$

If $LAG(\tau,t_h+1) > LAG(\tau,t_h-1)$, then $\sum_{T\in B'\cup I''} 2wt(T) - \left(\frac{8M-8h}{15}+\frac{7b''}{9}+\frac{168a''_0}{135}-a_0\right) > 0$, which implies that $\sum_{T\in B'\cup I''} wt(T) > \frac{4M-4h}{15}-\frac{a_0}{2}+\frac{7b''}{18}+\frac{84a''_0}{135}$. By Lemma 46, $\frac{8M-8h}{15}-a_0 > 0$, which implies that $\frac{4M-4h}{15}-\frac{a_0}{2}$, and hence, $\frac{4M-4h}{15}-\frac{a_0}{2}+\frac{7b''}{18}+\frac{84a''_0}{135} > 0$. □

Lemma 48 *If $LAG(\tau,t_h+1) > LAG(\tau,t_h-1)$, then $\sum_{T\in B'\cup I''} lag(T,t'_b+1) \leq -\frac{40M}{15}+\frac{40h}{15}+5a_0-\frac{70b''}{18}-\frac{840a''_0}{135}$.*

Proof: Let T be any task in B'; then, its critical subtask at t_h, T_k, has its deadline at or after t_h+1. Because T_k is scheduled at or before t'_b in S (by Def. 8), T_k receives its entire share of one quantum, by $t'_b+1 < t_h+1$ in S. Because no later subtask of T is eligible, and hence by (8), is not released before t_h+1, *i.e.*,

$$(\forall l : l > k :: r(T_l) \geq t_h+1), \qquad (72)$$

we have

$$lag(T,t_h+1) \leq 0. \qquad (73)$$

Also, because T_k is scheduled at or before t'_b, $e(T_k)$, and hence, by (38), $r(T_k) \leq t'_b$. Therefore, by (4) and (5),

$$(\forall h : h < k :: d(T_h) \leq t'_b + 1). \tag{74}$$

From (15), we have

$$\begin{aligned}
& lag(T, t'_b + 1) \\
&= lag(T, t_h + 1) - \sum_{u=t'_b+1}^{t_h} share(T, u) + \\
&\quad \sum_{u=t'_b+1}^{t_h} S(T, u) \\
&\leq \sum_{u=t'_b+1}^{t_h} S(T, u) - \sum_{u=t'_b+1}^{t_h} share(T, u) \quad \text{(from (73))} \\
&= -\sum_{u=t'_b+1}^{t_h} share(T, u) \\
&\qquad \text{(by Def. 8, } S(T, u) = 0 \text{, for } u \text{ in } [t'_b + 1, t_h]) \\
&= -\sum_{u=t'_b+1}^{t_h} wt(T)
\end{aligned}$$

(By (72) and (74), only subtask T_k of T contributes to the *share* of T in $[t'_b + 1, t_h - 1]$. Because $r(T_k) \leq t_b$ and $d(T_k) > t_h$, by (13), *share* of T in the interval equals $wt(T)$.)

$$\begin{aligned}
&= -(t_h - t'_b) \cdot wt(T) \\
&\leq -10 \cdot wt(T) \qquad \text{(from (55))}.
\end{aligned}$$

It can be shown in a similar manner that for a task T in I'', $lag(T, t'_b + 1) \leq -10 \cdot wt(T)$. Hence, the sum of the lags of all the tasks in $B' \cup I''$ at $t'_b + 1$ is given by

$$\begin{aligned}
& \sum_{T \in B' \cup I''} lag(T, t'_b + 1) \\
&\leq \sum_{T \in B' \cup I''} -10 \cdot wt(T) \\
&\leq -10 \cdot \left(\frac{4M - 4h}{15} - \frac{a_0}{2} + \frac{7b''}{18} + \frac{84a''_0}{135} \right) \\
&\qquad \text{(from Lemma 47)} \\
&= \frac{-40M + 40h}{15} + 5a_0 - \frac{70b''}{18} - \frac{840a''_0}{135}. \qquad \square
\end{aligned}$$

Finally, we determine a lower bound on the sum of the lags of the tasks scheduled at t'_b. We then prove Theorem 2 by showing that if the bound holds, then (D) is violated.

Lemma 49 *Let D denote the set of all the tasks scheduled at t'_b. Then, $lag(T, t'_b + 1) \leq 0$, for every T not in $D \cup B'' \cup A''_0$, or every T in $(\tau - D) \cap (\tau - B'' - A''_0)$.*

Proof: Let U be a task that is not in D. Then, U is in one of the following sets: A''_0, B'', B', $B - B' - B''$, $A - A''_0$ or I (by the definitions of the sets, they are pairwise disjoint and their union is τ). Thus, $\tau - D = ((\tau - D) \cap A''_0) \cup ((\tau - D) \cap B'') \cup ((\tau - D) \cap B') \cup ((\tau - D) \cap (B - B' - B'')) \cup ((\tau - D) \cap (A - A''_0)) \cup ((\tau - D) \cap I)$. In what follows, we reason about the lags of the tasks in the last four sets in the right-hand side of the above expression. (Because the first two sets are subsets of A''_0 and B'', respectively, for which the lag bound need not be proved, by the statement of the lemma.) In the cases that follow, let W be a task in B' whose critical subtask at t_h, W_l, is scheduled at t'_b. Therefore, by Defs. 8 and 4, and Lemma 16,

$$d(W_l) = t_h + 1. \tag{75}$$

Lags of tasks in $(\tau - D) \cap B'$: Let V be a task in this set and let V_k be its critical subtask at t_h. Then, by Def. 8, if V_k is not scheduled at t'_b, it is scheduled before. By Def. 4, $e(V_l)$ and hence, by (8), $r(V_l)$ is at or after $t_h + 1$ for any $l > k$. Thus, all subtasks released prior to $t'_b + 1$ receive their entire share by $t'_b + 1$ in S, while they may or may not in the ideal system. Hence, $lag(V, t'_b + 1) \leq 0$.

Lags of tasks in $(\tau - D) \cap (B - B' - B'')$: Let V be a task in this set and let V_k be its critical subtask at t_h. Then, by Def. 8, $e(V_k) > t'_b$, and hence, by (8),

$$r(V_k) > t'_b. \tag{76}$$

By Lemma 16 , $d(V_k) = t_h + 1$. Therefore, by (5), $d(V_j) \leq t_h$, for $j < k$, and in particular, if V_i is the critical subtask at t'_b of V, then,

$$d(V_i) \leq t_h. \tag{77}$$

Because V is not scheduled at t'_b, (75) and (77) imply that V_i is sheduled before t'_b. Thus, all subtasks prior to V_k complete execution before t'_b in S, while they may or may not complete before t'_b in the ideal system. By (76), V_k is not released before $t'_b + 1$, and hence, its share in the ideal system is zero. Thus, the actual allocation that V receives in S until $t'_b + 1$ is greater than or equal to its share until the same time in the ideal system. Hence, $lag(V, t'_b + 1) \leq 0$, for every V in this set.

Lags of tasks in $(\tau - D) \cap I$: Let V be a task in this set, and let V_k be the critical subtask of V at t'_b. By Def. 4, and because V is not active at t_h,

$$t'_b + 1 \leq d(V_k) \leq t_h. \quad (78)$$

Because V is not scheduled at t'_b but W_l is, (75) and (78) imply that V_k is scheduled earlier than t'_b. Thus, V_k completes execution before t'_b in S, while it may or not complete before t'_b in the ideal system. Since the successor of V_k is not released until $t'_b + 1$ (by Def. 4), the service provided by S to V is at least the service provided by the ideal system. Hence, $lag(V, t'_b + 1) \leq 0$, for all V in this set.

Lags of tasks in $(\tau - D) \cap (A - A''_0)$**:** Let V be a task in this set, V_k its critical subtask at t'_b, and V_l its subtask scheduled at t_h. Because V is not in A''_0, by Def. 8, V_l is not eligible at t'_b. Hence, $k \neq l$, which implies that $k < l$. By Def. 4, $d(V_k) \geq t'_b + 1$ and $r(V_l) \geq t'_b + 1$. Therefore, using the same reasoning of the previous case, we can conclude that the lag of every task in this set at t'_b is at most zero.

The set of all tasks not in $(D \cup B'' \cup A''_0)$ is given by

$$\begin{aligned} &\tau - (D \cup B'' \cup A''_0) \\ &\quad = \quad (\tau - D) \cap (\tau - (B'' \cup A''_0)) \\ &\quad = \quad (\tau - D) \cap (\tau - B'' - A''_0). \end{aligned}$$

Thus, a task not in $D \cup B'' \cup A''_0$ is in $(\tau - D) \cap (\tau - B'' - A''_0)$. □

Lemma 50 *The lag at $t'_b + 1$ of any task in A''_0 or B'' is at most one.*

Proof: We prove the lemma for tasks in A''_0. The proof for tasks in B'' is similar. Let T be a task in A''_0, T_k its subtask scheduled at t_h, and T_l, the successor of T_k in τ. By Def. 8, $e(T_k) \leq t'_b$, which by (38), and because T_k is scheduled at t_h implies that

$$r(T_k) \leq t'_b. \quad (79)$$

By Lemma 15, $d(T_k) = t_h + 1$, and hence, by (5),

$$r(T_l) \geq t_h, \quad (80)$$

which because T_l is not scheduled before t_h, by Lemma 4(a), implies that $e(T_l) \geq t_h$. Therefore, by Def. 4, T_k is the critical subtask at t'_b of T. (79), (4), and (5) imply that the following holds for T_h, which is T_k's predecessor in τ.

$$d(T_h) \leq t'_b + 1. \quad (81)$$

Let W_l be the critical subtask at t_h of W in B' which is scheduled at t'_b. Then, by Def. 8 and Lemma 16,

$$d(W_l) = t_h + 1. \quad (82)$$

(81) and (82), along with the fact that W_l is scheduled at t'_b imply that T_h is scheduled before t'_b in S. Therefore, T_h and all prior subtasks of T receive their entire share by $t'_b + 1$ in both S and in the ideal system. (80) implies that T_k is the only subtask released after subtask T_h that can have a positive share in the ideal system prior to $t'_b + 1$. This positive share is at most one by Lemma 2(e). Therefore, the maximum lag of T at $t'_b + 1$ is one (which is possible if T_k completes execution by $t'_b + 1$ in the ideal system). □

Lemma 51 *If $LAG(\tau, t_h + 1) \geq M + 1$, then $LAG(\tau, t'_b + 1) \geq M + 1 - h$.*

Proof: From (16),

$$\begin{aligned} &LAG(\tau, t_h + 1) \\ &\quad = \quad LAG(\tau, t_b + 1) + \\ &\qquad\qquad \sum_{u=t_b+1}^{u=t_h} \sum_{T \in \tau} (share(T, u) - S(T, u)) \\ &\quad \leq \quad LAG(\tau, t_b + 1) + \sum_{u=t_b+1}^{u=t_h} \sum_{T \in \tau} wt(T) - \\ &\qquad\qquad \sum_{u=t_b+1}^{u=t_h} \sum_{T \in \tau} S(T, u) \qquad \text{(from Lemma 2(c))} \\ &\quad \leq \quad LAG(\tau, t_b + 1) + (t_h - t_b) \cdot M - \\ &\qquad\qquad \sum_{u=t_b+1}^{u=t_h} \sum_{T \in \tau} S(T, u) \qquad \text{(from 11)} \\ &\quad \leq \quad LAG(\tau, t_b + 1) + (t_h - t_b) \cdot M - \\ &\qquad\qquad ((t_h - t_b) \cdot M - h) \\ &\quad \text{(by Lemma 42, there are no holes in } [t'_b + 1, t_h - 1]) \\ &\quad = \quad LAG(\tau, t_b + 1) + h. \end{aligned}$$

The last inequality above implies that $LAG(\tau, t_b + 1) \geq LAG(\tau, t_h + 1) - h$. Hence, if $LAG(\tau, t_h + 1) \geq M + 1$, then $LAG(\tau, t_b + 1) \geq M + 1 - h$. □

Lemma 52 *Let D be the set of tasks scheduled at t'_b. If $LAG(\tau, t_h + 1) \geq M + 1$, then the sum of* *the lags of the tasks in* $D - (B' \cup I'')$ *at* $t'_b + 1$ *is at least* $\frac{160M}{105} + \frac{965h}{105} + \frac{82}{7} + \frac{52b''}{18} + \frac{605a''_0}{135}$.

Proof: The set $D - (B' \cup I'')$ can be expressed as $\tau - (\tau - (D - (B' \cup I'')))$, where $(\tau - (D - (B' \cup I''))) = (\tau - D) \cup B' \cup I''$ (all tasks in τ that are not in D but may be in $B' \cup I''$). Hence,

$$
\begin{aligned}
& D - (B' \cup I'') \\
&= \tau - (\tau - (D - (B' \cup I''))) \\
&= \tau - ((\tau - D) \cup B' \cup I'') \\
&= \tau - (B' \cup I'' \cup ((\tau - D) \cap (A''_0 \cup B' \cup \\
&\quad B'' \cup I'' \cup (\tau - A''_0 - B'' - B' - I'')))) \\
&\quad (A''_0 \cup B' \cup B'' \cup I'' \cup (\tau - A''_0 - B' - B'' - I'') = \tau) \\
&= \tau - (B' \cup I'' \cup ((\tau - D) \cap A''_0) \cup ((\tau - D) \cap B') \\
&\quad \cup ((\tau - D) \cap I'') \cup ((\tau - D) \cap B'') \\
&\quad \cup ((\tau - D) \cap (\tau - A''_0 - B' - B'' - I''))) \\
&= \tau - ((B' \cup ((\tau - D) \cap B')) \cup (I'' \cup ((\tau - D) \cap I'')) \\
&\quad \cup ((\tau - D) \cap B'') \cup ((\tau - D) \cap A''_0) \\
&\quad \cup ((\tau - D) \cap (\tau - A''_0 - B' - B'' - I''))) \\
&\qquad \text{(set union is associative and commutative)} \\
&= \tau - (B' \cup I'' \cup ((\tau - D) \cap B'') \cup ((\tau - D) \cap A''_0) \\
&\quad \cup ((\tau - D) \cap (\tau - A''_0 - B' - B'' - I'')) \\
&\quad \text{(union of } B' \text{ and tasks not in } D \text{ but in } B' \text{ is } B') \\
&\quad \text{(union of } I'' \text{ and tasks not in } D \text{ but in } I'' \text{ is } I'')
\end{aligned}
$$

The sets A''_0, B', B'',I'', and $(\tau - A''_0 - B' - B'' - I'')$ are pairwise disjoint, and hence, so are their intersections with $(\tau - D)$. Therefore,

$$
\begin{aligned}
& \sum_{T \in D - (B' \cup I'')} lag(T, t'_b + 1) \\
&= LAG(\tau, t'_b + 1) - \Bigg(\sum_{T \in B' \cup I''} lag(T, t'_b + 1) + \\
&\quad \sum_{T \in (\tau - D) \cap B''} lag(T, t'_b + 1) + \\
&\quad \sum_{T \in (\tau - D) \cap A''_0} lag(T, t'_b + 1) + \\
&\quad \sum_{T \in ((\tau - D) \cap (\tau - B' - B'' - A''_0 - I''))} lag(T, t'_b + 1) \Bigg) \\
&= LAG(\tau, t'_b + 1) - \sum_{T \in B' \cup I''} lag(T, t'_b + 1) \\
&\quad - \sum_{T \in (\tau - D) \cap B''} lag(T, t'_b + 1) \\
&\quad - \sum_{T \in (\tau - D) \cap A''_0} lag(T, t'_b + 1) \\
&\quad - \sum_{T \in ((\tau - D) \cap (\tau - B' - B'' - A''_0 - I''))} lag(T, t'_b + 1) \\
&= LAG(\tau, t'_b + 1) - \sum_{T \in B' \cup I''} lag(T, t'_b + 1) \\
&\quad - \sum_{T \in (\tau - D) \cap B''} lag(T, t'_b + 1) \\
&\quad - \sum_{T \in (\tau - D) \cap A''_0} lag(T, t'_b + 1) - 0 \\
&\qquad \text{(By Lemma 49, lags of tasks in} \\
&\qquad (\tau - D) \cap (\tau - B'' - A''_0) \text{ are at most 0,} \\
&\qquad \text{and } (\tau - D) \cap (\tau - B'' - A''_0 - B' - I'') \\
&\qquad \text{is a subset of } (\tau - D) \cap (\tau - B'' - A''_0).) \\
&\geq M + 1 - h - \sum_{T \in B' \cup I''} lag(T, t'_b + 1) \\
&\quad - \sum_{T \in (\tau - D) \cap B''} lag(T, t'_b + 1) \\
&\quad - \sum_{T \in (\tau - D) \cap A''_0} lag(T, t'_b + 1) \\
&\qquad \text{(from Lemma 51)} \\
&\geq M + 1 - h - \\
&\quad \left(\frac{-40M}{15} + \frac{40h}{15} + 5a_0 - \frac{70b''}{18} - \frac{840a''_0}{135} \right) \\
&\quad - \sum_{T \in (\tau - D) \cap B''} lag(T, t'_b + 1) \\
&\quad - \sum_{T \in (\tau - D) \cap A''_0} lag(T, t'_b + 1) \\
&\qquad \text{(from Lemma 48)} \\
&\geq M + 1 - h \\
&\quad - \left(\frac{-40M}{15} + \frac{40h}{15} + 5a_0 - \frac{70b''}{18} - \frac{840a''_0}{135} \right) \\
&\quad - b'' - a''_0 \\
&\qquad \text{(By Lemma 50, tasks in } B'' \text{ and } A''_0 \text{ have} \\
&\qquad \text{lags at most 1.)} \\
&= \frac{55M}{15} - \frac{55h}{15} + 1 + \frac{52b''}{18} + \frac{605a''_0}{135} - 5a_0 \\
&\geq \frac{55M}{15} - \frac{55h}{15} + 1 + \frac{52b''}{18} + \frac{605a''_0}{135} \\
&\quad -5 \cdot \left(\frac{3M - 18h - 15}{7} \right) \\
&\qquad (a_0 \leq \tfrac{3M - 18h - 15}{7} \text{ by Lemma 36}) \\
&= \frac{160M}{105} + \frac{965h}{105} + \frac{82}{7} + \frac{52b''}{18} + \frac{605a''_0}{135}. \qquad \square
\end{aligned}
$$

Lemma 53 *If* $LAG(\tau, t_h + 1) > LAG(\tau, t'_b + 1)$, *then,* $LAG(\tau, t_h + 1) < M + 1$.

Proof: Assume to the contrary that $LAG(\tau, t_h + 1) \leq M+1)$. Let D be the set of tasks scheduled at t'_b. Then $|D - B' - I''| \leq M - 1$, from the definition of B' and t'_b. Therefore, $\sum_{T \in (D - B' - I'')} lag(T, t'_b + 1) < \sum_{T \in (D - B' - I'')} 2wt(T) = 2 \cdot wt(M -$

1), from Lemma 10. From Lemma 52, $\sum_{T \in (D-B'-I'')} lag(T, t'_b + 1) \geq \frac{160M}{105} + \frac{965h}{105} + \frac{82}{7} + \frac{52b''}{18} + \frac{605a''_0}{105}$. Therefore, $2 \cdot wt \cdot (M-1) > \frac{160M}{105} + \frac{965h}{105} + \frac{82}{7} + \frac{52b''}{18} + \frac{605a''_0}{105} > \frac{160M+965h}{105}$, or $wt > \frac{160M+965h}{210M-210} > \frac{160}{210} > \frac{11}{15}$, for all M. Thus, (D) is violated. Therefore, $LAG(\tau, t_h + 1) < M + 1$. □

3.4.1 Case D ($A_1^0 = A_1^1 = \phi$)

Lemma 54 *If $A_1^0 = A_1^1 = \phi$, then $LAG(\tau, t_h + 1) < M + 1$.*

From (24) and $A_1^0 = A_1^1 = \phi$, $LAG(\tau, t_h + 1) < a_0 \cdot wt + (4wt - 2) \cdot a_1^2$, which, by (35), equals $a_0 \cdot wt + (M - h - a_0) \cdot (4wt - 2)$. $LAG(\tau, t_h + 1) \geq M + 1$ implies that $a_0 \cdot wt + (M - h - a_0) \cdot (4wt - 2) > M + 1$, which implies that $wt > f = \frac{3M-2h-2a_0+1}{4M-4h-3a_0}$. M, h, and a_0 are constrained by $M \geq h + a_0$, and $M, h, a_0 > 0$. It can be shown that the value of f is minimized when $h = a_0 = 1$, for which $f = \frac{3M-3}{4M-7} > \frac{3}{4}$, for all $M > 1$. This violates (C) and (D). □

By Lemmas 24, 33, 34, 53, and 54, if (C) or (D) is satisfied, then $LAG(\tau, t_h + 1) < M + 1$, which is a contradiction to (A3). Thus, Theorem 2 is proved.

3.5 Other Results

The above approach can be extended to obtain a general per-task weight restriction of $\frac{1+q}{2+q}$, and a restriction of $\frac{7+4q}{11+4q}$, if no subtask is allowed to become eligible before its release time, for ensuring a tardiness of q. The proof is the same, except for generalizations to allow subtasks with tardiness up to q to be scheduled in any slot.

It can be shown that a tardiness of two less than the largest difference between successive group deadlines of any task can be ensured, in the absence of any restrictions. A formal proof is omitted due to space constraints. However, note that the key to the proof we have presented is dealing with the impact of cascades of deadline misses in heavy tasks. Such cascades must end by the next group deadline, regardless of any restrictions.

4 Conclusion

We have presented counterexamples that show that tardiness under the EPDF Pfair algorithm can exceed a small constant number of quanta for task systems that are not restricted, thereby proving false the conjecture that EPDF ensures a tardiness of one quantum. We have also presented sufficient utilization restrictions that are more liberal than those previously known.

References

[1] J. Anderson and A. Srinivasan. Mixed Pfair/ERfair scheduling of asynchronous periodic tasks. *Journal of Computer and System Sciences*. To appear.

[2] 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*, pages 297–306, Dec. 2000.

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

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

[5] A. Srinivasan. *Efficient and Flexible Fair Scheduling of Real-time Tasks on Multiprocessors*. PhD thesis, University of North Carolina at Chapel Hill, December 2003.

[6] A. Srinivasan and J. Anderson. Optimal rate-based scheduling on multiprocessors. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 189–198, May 2002.

[7] A. Srinivasan and J. Anderson. Efficient scheduling of soft real-time applications on multiprocessors. In *Proceedings of the 15th Euromicro Conference on Real-time Systems*, pages 51–59, July 2003.