Optimal Rate-based Scheduling on Multiprocessors*

Anand Srinivasan and James H. Anderson
Department of Computer Science, University of North Carolina Chapel Hill, NC

Abstract

The PD^2 Pfair/ERfair scheduling algorithm is the most efficient known algorithm for optimally scheduling periodic tasks on multiprocessors. In this paper, we prove that PD^2 is also optimal for scheduling "rate-based" tasks whose processing steps may be highly jittered. The rate-based task model we consider generalizes the widely-studied sporadic task model.

1 Introduction

In the real-time scheduling literature, the periodic [10] and sporadic [11] task models have received the most attention. In the periodic model, each task is invoked repeatedly, with consecutive invocations, or *jobs*, being spaced apart by a fixed amount; in the sporadic model, a lower bound on the time between invocations is assumed. In practice, however, event occurrences often are neither periodic nor sporadic. For example, in an application that services packets arriving over a network, packet arrivals may be highly jittered. Rate-based scheduling schemes are more seamlessly able to cope with jitter. In such schemes, there is no restriction on a task's instantaneous rate of execution, but an average rate is assumed. If a task's instantaneous rate exceeds its average rate, then it is dealt with by using simple mechanisms such as postponing deadlines. In this paper, we investigate rate-based scheduling on multiprocessors. The starting point for our work is recent research on Pfair and ERfair scheduling algorithms, which are known to be optimal for scheduling periodic tasks on multiprocessors [2, 4, 5, 6].

Under *Pfair scheduling*, each task is required to execute at a uniform rate, while respecting a fixed allocation quantum. 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 is effectively subdivided into quantum-length *subtasks* that must execute within *windows* of approximately equal lengths: if a subtask of a task T executes outside of its window, then T's error bounds are exceeded. *Early-release fair* (*ERfair*) differs from Pfair scheduling in a rather simple way: under ERfair scheduling, a subtask may become eligible for execution *early*, *i.e.*, before its Pfair window. By allowing early releases, response times can often be reduced.

In [3], we proposed the *intra-sporadic task model*, which generalizes both the sporadic and early-release models. In the intra-sporadic model, subtasks may become eligible either early or late, *i.e.*, there may be separation between consecutive windows of the same task. As explained later, the intra-sporadic notion of a rate is quite similar to that found in the recently-proposed *uniprocessor* rate-based execution model [9]. In [3], we presented an algorithm that optimally schedules intra-sporadic tasks on two processors. However, we left open the problem of optimally scheduling intra-sporadic tasks on systems of more than two processors.

Contributions of this paper. In this paper, we close this problem by showing that the PD^2 Pfair algorithm [2, 4] correctly schedules any feasible intra-sporadic task system on M processors. Because the intra-sporadic model is a generalization of the sporadic model, our work also shows that PD^2 is optimal for scheduling sporadic tasks on multiprocessors. Since periodic task systems represent a "worst-case" scenario in the spectrum of intra-sporadic (or sporadic) task systems, one may think that the optimality of PD^2 would follow as a simple corollary from previous work. However, previously-presented proofs for Pfair and ERfair scheduling algorithms do not easily extend beyond the periodic task model. In this paper, we provide a new approach for dealing with Pfair- or ERfair-scheduled systems and use it to show that intra-sporadic tasks can be optimally scheduled on multiprocessors. This paper breaks new ground by being the first to show that sporadic or intra-sporadic tasks can be optimally scheduled on systems of more than two processors.

Some example schedules. The length and alignment of a task's Pfair windows is determined by its *weight*, which is defined as the ratio of its per-job execution cost and period. Fig. 1 shows some schedules in-

*Supported by NSF grants CCR 9972211, CCR 9988327, and ITR 0082866. Email: {anands,anderson}@cs.unc.edu.

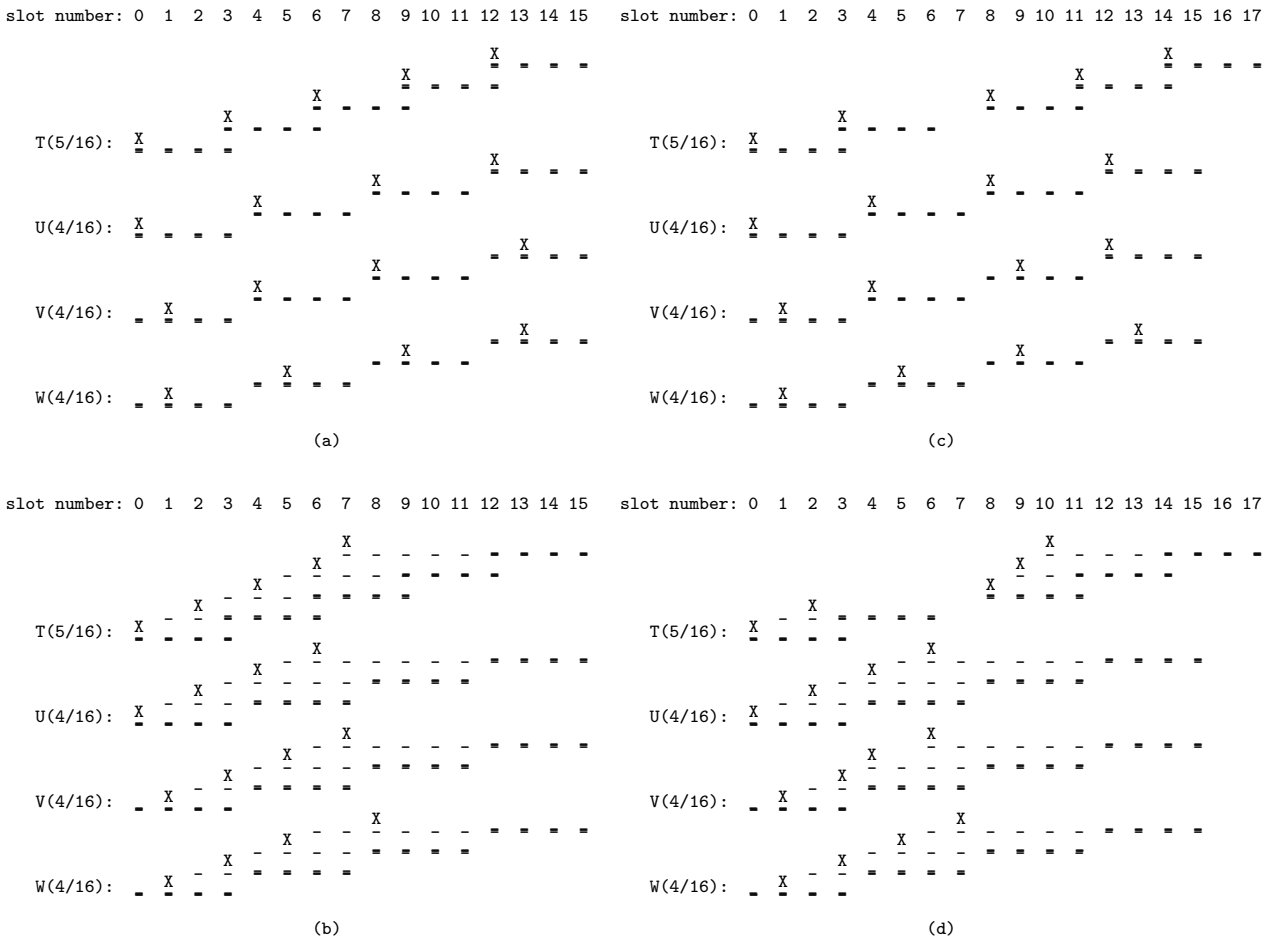

Figure 1: A partial schedule for an intra-sporadic task set under various conditions. In each of these schedules, subtasks of the same task are shown on different lines. Each subtask has an *eligibility interval* corresponding to the sequence of time slots in which it can be scheduled; a subtask's eligibility interval must include its Pfair window. In each schedule, each subtask's eligibility interval is denoted by a sequence of dashes, with its Pfair window denoted in bold; the time slot in which the subtask is scheduled is denoted by an 'X'. **(a)** All tasks are periodic and are Pfair-scheduled. **(b)** All tasks are periodic and are ERfair-scheduled. **(c)** Task T has a late-released subtask and no subtask is eligible before its Pfair window. **(d)** Task T has a late-released subtask and all subtasks become eligible as early as possible.

volving four tasks: T with weight 5/16, and U, V, and W with weight 4/16. Insets (a) and (b) show schedules for Pfair- and ERfair-scheduled systems, respectively. Insets (c) and (d) show similar schedules, except that the third subtask of task T is released two time units late (perhaps this late release represents a packet that arrived late). In the intra-sporadic model, each subtask must be scheduled within an *eligibility interval* that includes its Pfair window. For example, the eligibility interval of T's second subtask in inset (b) is $[1, 6]$, whereas its Pfair window is $[3, 6]$.

In the rest of this paper, we present needed definitions (Sec. 2), describe the PD^2 algorithm (Sec. 3), prove that PD^2 optimally schedules intra-sporadic tasks (Sec. 4), and then conclude (Sec. 5).

2 Definitions

In the scheduling disciplines we consider, processor time is allocated in discrete quanta; the time interval $[t, t+1)$, where t is a nonnegative integer, is called *slot* t. A task T may be allocated time on different processors, but not in the same slot. The sequence of allocation decisions over time defines a *schedule* S. Formally, $S\colon \tau \times \mathcal{Z} \mapsto \{0, 1\}$, where τ is a set of tasks and $\mathcal{Z}$ is the set of nonnegative integers. $S(T, t) = 1$ means that task T is scheduled in slot t. In any schedule for M processors, $\sum_{T \in \tau} S(T, t) \leq M$ for all t. Note that the definition of a schedule given here permits "invalid" scheduling decisions (*e.g.*, missed deadlines). Our validity condition for schedules is defined later in Sec. 2.2.

2.1 Pfair and ERfair Scheduling

In this subsection, we define notions relevant to Pfair and ERfair scheduling. For now, we limit attention to periodic tasks. A periodic task T with a *period* $T.p$ and an *execution cost* $T.e$ has a *weight* of $T.e/T.p$. Every $T.p$ time units, T releases a new *job* (*i.e.*, instance) with cost $T.e$. Each job of T must complete execution before the next job of T is released. We require that $T.e \leq T.p$ and hence $wt(T) \leq 1$. A task with weight less than (at least) 1/2 is called a *light* (*heavy*) task.

The notion of a Pfair schedule is defined by comparing to an ideal system with an infinitesimally small quantum that allocates exactly $(T.e/T.p)t$ time to each task T over $[0, t]$ for any integer t. Deviance from the ideal system is formally captured by the concept of *lag*. Formally, the *lag of task T at time t in a schedule S* is

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

(For conciseness, we leave the schedule implicit and use $lag(T, t)$ instead of $lag(T, t, S)$.) A schedule is *Pfair* iff

$$(\forall T, t :: -1 < lag(T, t) < 1). \quad (2)$$

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

The lag bounds above have the effect of breaking each task T into an infinite sequence of unit-time *subtasks*. We denote the i^{th} subtask of task T as T_i, where $i \geq 1$. As in [5], we associate a *pseudo-release* $r(T_i)$ and *pseudo-deadline* $d(T_i)$ with each subtask T_i, as follows.

$$r(T_i) = \left\lfloor \frac{i-1}{wt(T)} \right\rfloor \quad (3)$$

$$d(T_i) = \left\lceil \frac{i}{wt(T)} \right\rceil - 1 \quad (4)$$

(For brevity, we often drop the prefix "pseudo-.") In a Pfair-scheduled system, subtask T_i can be scheduled only within the interval $[r(T_i), d(T_i)]$, termed its *window*, and denoted $w(T_i)$. The *length* of $w(T_i)$, denoted $|w(T_i)|$, is $d(T_i) - r(T_i) + 1$. As an example, consider a task T with weight $T.e/T.p = 8/11$. Each job of T consists of eight windows, one for each of its unit-length subtasks, as shown in Fig 1(a). It can be shown that, in general, consecutive windows of a task are either disjoint or overlap by one slot.

The notion of ERfair scheduling [2] is obtained by simply dropping the -1 constraint in (2). With this change, a subtask can become eligible before its window. Note that any Pfair schedule is ERfair, but not necessarily vice versa. It is easy to show that, in any Pfair or ERfair schedule, all job deadlines are met [2].

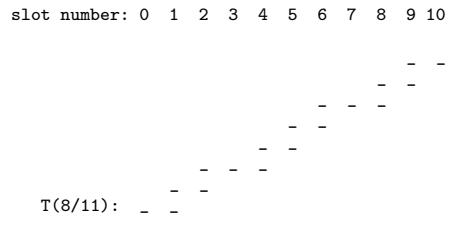

Figure 2: The eight windows of the first job of a task T with weight $T.e/T.p = 8/11$. This job consists of subtasks $T_1, \ldots, T_8$, each of which must be allocated processor time during its window, or else a lag-bound violation will result.

2.2 Intra-sporadic Tasks

The intra-sporadic task model generalizes the sporadic model by allowing separation between consecutive subtasks of a task. In addition, early releases are allowed.

Formally, an *intra-sporadic task system* is defined by a pair (τ, e), where τ is a task set, and e is a function that indicates when each subtask becomes eligible. Each task may release either a finite or infinite number of subtasks. We assume that $e(T_i) \geq e(T_{i-1})$ for all $i \geq 2$, *i.e.*, no subtask is eligible before its predecessor.

In the intra-sporadic model, each subtask has both a *Pfair window* (PF-window) and an *intra-sporadic window* (IS-window). A subtask's IS-window, which includes its PF-window, defines the interval during which it is eligible to be scheduled. For example, the fourth subtask of T in Fig. 1(d) has an IS-window of $[9, 14]$ and a PF-window of $[11, 14]$. Formally, T_i's IS-window is defined as $[e(T_i), d(T_i)]$ and its PF-window is defined as $[r(T_i), d(T_i)]$. As we shall see, the terms $r(T_i)$ and $d(T_i)$ have a similar interpretation to that given previously for periodic task systems. As before, we will use $w(T_i)$ to denote the PF-window of subtask T_i.

$r(T_i)$ and $d(T_i)$ are defined inductively by examining the PF-windows of T in a periodic system. As mentioned earlier, consecutive PF-windows of a periodic task are either disjoint or overlap by one slot. The bit $b(T_i)$ distinguishes between these two possibilities.

$$b(T_i) = \begin{cases} 1, & \text{if } \left\lceil \frac{i}{wt(T)} \right\rceil - 1 = \left\lfloor \frac{i}{wt(T)} \right\rfloor \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

By (3) and (4), if T is *periodic*, then $w(T_i)$ and $w(T_{i+1})$ overlap (by one slot) if $b(T_i) = 1$, and do not overlap if $b(T_i) = 0$. For intra-sporadic tasks, we define $b(T_i)$ exactly as above. Given this definition, we can define $r(T_i)$, which defines the beginning of T_i's PF-window.

$$r(T_i) = \begin{cases} e(T_i), & \text{if } i = 1 \\ max(e(T_i), d(T_{i-1}) + 1 - b(T_{i-1})), & \text{if } i \geq 2 \end{cases} \quad (6)$$

Thus, if T_i becomes eligible *during* T_{i-1}'s PF-window, then $r(T_i) = d(T_{i-1})+1-b(T_{i-1})$, and hence, the spacing between $r(T_{i-1})$ and $r(T_i)$ is exactly as in a periodic task system.[1] 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 (6) 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 $r(T_i) + |w(T_i)| - 1$. PF-window lengths here are as in periodic systems. Thus, by (3) and (4), we have the following.

$$|w(T_i)| = \left\lceil \frac{i}{wt(T)} \right\rceil - \left\lfloor \frac{i-1}{wt(T)} \right\rfloor \tag{7}$$

$$d(T_i) = r(T_i) + \left\lceil \frac{i}{wt(T)} \right\rceil - \left\lfloor \frac{i-1}{wt(T)} \right\rfloor - 1 \tag{8}$$

Thus, if T_i becomes eligible early, *i.e.*, $e(T_i) < r(T_i)$, then its deadline is postponed to where it would have been if $e(T_i) = r(T_i)$.

To summarize, our notion of an intra-sporadic task is obtained by allowing a task's PF-windows to be right-shifted from where they would appear if the task were periodic. In addition, we allow a subtask to become eligible before its PF-window, as in ERfair scheduling. A schedule for a system of intra-sporadic tasks is said to be *valid* iff each subtask is scheduled in its IS-window.

In [3], we proved that an intra-sporadic task system τ has a valid schedule on M processors (*i.e.*, is *feasible*) iff

$$\sum_{T \in \tau} \frac{T.e}{T.p} \leq M. \tag{9}$$

Our proof in fact shows that a valid schedule exists in which each subtask is scheduled in its PF-window.

Relation to the RBE task model. In the recently-proposed uniprocessor *rate-based execution* (RBE) model [9], each task is characterized by four parameters: (x, y, d, c). A task is *expected* to release x jobs every y time units; each job has an execution cost of c and a relative deadline of d. In the intra-sporadic model, a task with parameters (e, p) is *expected* to release e subtasks every p time units; each subtask has an execution cost of one and a relative deadline of approximately p/e. An RBE task may release more than x jobs every y time units, but the deadlines of jobs released early are postponed in a way that ensures the system is still feasible. Deadlines of early intra-sporadic subtasks are similarly postponed using (6) and (8).

[1]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.

Lag values in an intra-sporadic task system. The lag of an intra-sporadic task at time t can be defined in the same way as it is defined for periodic tasks:

$$lag(T, t) = ideal(T, t) - \sum_{u=0}^{t-1} S(T, u), \tag{10}$$

where $ideal(T, t)$ is the amount of processor time task T receives in an ideal system[2] in $[0, t)$. For a periodic task, $ideal(T, t) = (T.e/T.p)t$. To define $ideal(T, t)$ for an intra-sporadic task, we consider the feasibility proof given in [3]. There, a valid schedule is shown to exist by constructing a flow network with a certain real-valued flow. $ideal(T, t)$ is defined based on this flow:

$$ideal(T, t) = \sum_{u=0}^{t-1} flow(T, u). \tag{11}$$

Here, $flow(T, u)$ is the flow (or share) assigned to task T in slot u. We formally define $flow(T, u)$ below. For motivation, consider a task of weight 5/16. In any valid schedule, each subtask of this task must receive a share of one unit processor time over its IS-window. In the ideal system, each subtask gets a share of 5/16 in each slot of its PF-window, except maybe the first and last slots of the window. This is illustrated in Fig. 3. Inset (a) shows the shares assigned in each slot of the PF-window for a periodic task of weight 5/16, and inset (b) shows the shares in each slot for an intra-sporadic task of weight 5/16 in which some subtasks are released late. Note that the shares for each subtask sum to one (*e.g.*, $5/16 + 5/16 + 5/16 + 1/16 = 1$ for the first subtask). Also, note that the share in each slot is at most 5/16, the weight of the task. For the periodic task, the share in each slot is *exactly* 5/16, whereas for the intra-sporadic task, it may be less (see slot 3 in inset (b)). In the flow network, each subtask has flows corresponding to these shares.

Formally, $flow(T, u)$ is defined in terms of a function f, which indicates the share assigned to each subtask T_i in each slot u. The function f is defined as follows.

$$\begin{aligned} f(T_i, r(T_i)) &= \left(\left\lfloor \tfrac{i-1}{wt(T)} \right\rfloor + 1\right) \times wt(T) - (i-1) \\ f(T_i, d(T_i)) &= i - \left(\left\lceil \tfrac{i}{wt(T)} \right\rceil - 1\right) \times wt(T) \\ f(T_i, u) &= \begin{cases} wt(T), & \text{if } u \in [r(T_i)+1, d(T_i)-1] \\ 0, & \text{if } u \notin [r(T_i), d(T_i)]. \end{cases} \end{aligned} \tag{12}$$

For example, consider the last slot of the second sub-

[2]We assume the ideal system gives each task T a share of exactly $T.e/T.p$, and does not distribute any excess processor capacity. There may be other ways to define an "ideal" system, but the definition given here is sufficient for our proof.

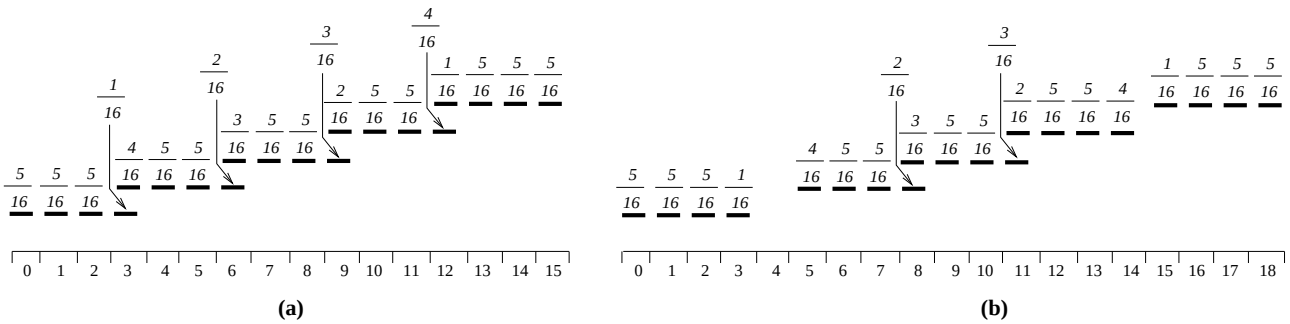

Figure 3: Windows of a task of weight 5/16 and the share of each subtask in the slots of its window. **(a)** No subtask is released late. **(b)** The second and fifth subtasks are released late.

task in Fig. 3(b) and also the first slot of the third subtask. $f(T_2, d(T_2)) = f(T_2, 8)$, which by (12) equals $2-(\left\lceil \frac{2}{5/16} \right\rceil -1)\times(5/16) = 2/16$, as shown in the figure. Similarly, $f(T_3, r(T_3)) = f(T_3, 8)$, which by (12) equals $(\left\lfloor \frac{2}{5/16} \right\rfloor +1)\times(5/16)-(3-1) = 3/16$. Note that these two flows sum to 5/16, the weight of the task.

The function $flow(T, u)$ is defined as $flow(T, u) = \sum_i f(T_i, u)$. From this, (P1) below follows.

(P1) For all time slots t, $flow(T, t) \leq wt(T)$.

From (10) and (11), we get

$$\begin{aligned} lag(T, t+1) &= \sum_{u=0}^{t} (flow(T, u) - S(T, u)) \qquad (13) \\ &= lag(T, t) + flow(T, t) - S(T, t). \end{aligned}$$

Similar to the notion of *lag* for tasks, we can define the total lag of a task system. The total lag for a schedule S and task system τ at time $t+1$, denoted by $LAG(\tau, t+1)$, is defined as follows.

$$LAG(\tau, t+1) = LAG(\tau, t) + \sum_{T \in \tau} (flow(T, t) - S(T, t)) \qquad (14)$$

$LAG(\tau, 0)$ is defined to be 0. Note that the definitions of *lag* and *LAG* do not make any assumptions about the validity of the corresponding schedule.

3 Algorithm PD^2

PD^2 prioritizes subtasks by their deadlines. Any ties are broken using two tie-break parameters: the b-bit defined in (5), and the "group deadline," defined next.

The group deadline. It can be shown that all windows of a heavy task are of length two or three. Consider a sequence $T_i, \ldots, T_j$ of subtasks of a heavy task T (without any late releases) such $|w(T_k)| = 2$ for all $i < k \leq j$, $b(T_k) = 1$ for all $i \leq k < j$, and either $b(T_j) = 0$ or $|w(T_{j+1})| = 3$ (*e.g.*, T_1, T_2 or T_3, T_4, T_5 or T_6, T_7, T_8 in Fig. 2). If any of $T_i, \ldots, T_j$ is scheduled in the last slot of its window, then each subsequent subtask in this sequence must be scheduled in its last slot. In effect, $T_i, \ldots, T_j$ must be considered as a single schedulable entity subject to a "group" deadline. Formally, we define the *group deadline* for the subtasks $T_i, \ldots, T_j$ to be $d(T_j)$ if $b(T_j) = 0$, and $d(T_j) + 1$ if $|w(T_{j+1})| = 3$. Intuitively, if we imagine a job of T in which each subtask is scheduled in the first slot of its window, then the remaining empty slots exactly correspond to the group deadlines of T. For example, in Fig. 2, T has group deadlines at slots 3, 7, and 10.

We let $D(T_i)$ denote the group deadline of subtask T_i. Formally, if T is heavy, then $D(T_i) = (\textbf{min}\ u :: u \geq d(T_i)$ and u is a group deadline of $T)$. For example, in Fig. 2, $D(T_1) = 3$ and $D(T_6) = 10$. If T is light, then $D(T_i) = 0$. For an intra-sporadic task, the group deadline is defined in the same way, assuming that all the future subtasks are released as early as possible. The group deadline can be calculated using a simple formula (ommitted here due to space limitations).

Having explained the notion of a group deadline, we can now state the PD^2 priority definition.

PD^2 Priority Definition: Subtask T_i's priority at slot t is defined to be $(d(T_i), b(T_i), D(T_i))$, if it is eligible at t. Priorities are ordered using the following relation.

$$\begin{aligned} (d', b', D') \preceq (d, b, D) \equiv [d < d'] \vee [(d = d') \wedge (b > b')] \\ \vee [(d = d') \wedge (b = b') \wedge (D \geq D')] \end{aligned}$$

If T_i and U_j are both eligible at t, then T_i's priority is at least U_j's at t if $(d(U_j), b(U_j), D(U_j)) \preceq (d(T_i), b(T_i), D(T_i))$. □

According to the definition above, T_i has higher priority than U_j if it has an earlier deadline. If T_i and U_j have equal deadlines, but $b(T_i) = 1$ and $b(U_j) = 0$, then the tie is broken in favor of T_i. This is because the window of T_i may overlap with that of its successor (and hence *not* scheduling it may reduce the number of slots available for its successor by one, constraining

the future schedule). If T_i and U_j have equal deadlines and b-bits, then their group deadlines are inspected to break the tie. If one is heavy and the other light, then the tie is broken in favor of the heavy task (by the definition of the group deadline). If both are heavy and their group deadlines differ, then the tie is broken in favor of the one with the later group deadline. Note that the subtask with the later group deadline can force a longer cascade of scheduling decisions in the future. Thus, choosing to schedule such a subtask early places fewer constraints on the future schedule. Any ties not resolved by PD^2 can be broken arbitrarily.

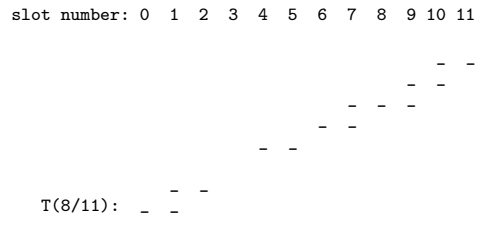

Figure 4: The PF-windows of a generalized intra-sporadic task T with weight $T.e/T.p = 8/11$. Subtask T_3 is removed and T_5 becomes eligible one unit late. (Because T_3 is missing, this is not an intra-sporadic task.)

4 Proof of Optimality of PD^2

We now prove that PD^2 optimally schedules intra-sporadic tasks. Assume to the contrary that there exists a feasible task set not schedulable using PD^2. We consider one such task system with certain properties that are in some sense "minimal." By reasoning about lags in the PD^2 schedule, we derive a contradiction. Due to space limitations, only light tasks are considered in detail in this section; needed lemmas involving heavy tasks are proved in an appendix. The following property follows from the fact that light tasks have windows of length at least three.

(P2) For a light task T, $d(T_{i+1}) > d(T_i) + 1$ for all i.

Generalized intra-sporadic task systems. In our proof, we consider task systems obtained by removing subtasks from an intra-sporadic task system. Note that such a task system may no longer be intra-sporadic (see Fig. 4). To circumvent this problem, we define a more general model called the *generalized* intra-sporadic task model, and show that PD^2 can optimally schedule task systems in this model. In a generalized intra-sporadic 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 $T_{i+1}, T_{i+2}, \ldots, T_{k-1}$ 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 $\left\lfloor \frac{k-1}{wt(T)} \right\rfloor$. Thus, for every generalized intra-sporadic task system τ, there exists an intra-sporadic task system τ' such that τ can be obtained by simply removing certain subtasks in τ'. For generalized intra-sporadic task systems, the parameters in the PD^2 priority definition are defined and calculated in the same way as for intra-sporadic systems.

A task T of a generalized intra-sporadic task system is *active* at slot t if there exists a subtask T_i such that $e(T_i) \leq t \leq d(T_i)$. (Note that a task that is active at t is not necessarily scheduled at t.) If a task T is inactive and its most-recently released subtask is T_i, then it can resume execution with any subtask T_k such that $r(T_i) + \left\lfloor \frac{k-1}{wt(T)} \right\rfloor - \left\lfloor \frac{i-1}{wt(T)} \right\rfloor \leq t$. 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 (*e.g.*, T_4 is the successor of T_2 in Fig. 4).

The following are properties of generalized intra-sporadic task systems ((P4) is proved in the appendix).

(P3) Removing a subtask from a generalized intra-sporadic task system results in another generalized intra-sporadic task system. (Follows from the definition of such systems.)

(P4) Let T_i be a subtask such that $b(T_i) = 1$, and let T_k be the successor of T_i. If $r(T_k) > d(T_i)$, then $\mathit{flow}(T, d(T_i)) + \mathit{flow}(T, d(T_i) + 1) \leq wt(T)$.

Henceforth, when we refer to a task system, we mean a generalized intra-sporadic task system.

Given the above definitions, we are now ready to prove the optimality of PD^2 for scheduling generalized intra-sporadic tasks. If PD^2 is not optimal, then there exists a task system that is feasible and yet misses a deadline under PD^2. This implies that there exists a time slot t_d and a task system τ defined as follows.

Definition 1 t_d is defined as the minimal time slot at which some task system misses a deadline under PD^2, *i.e.*, some such task system misses a deadline at t_d, and no such task system misses a deadline prior to t_d. □

Let the *rank* of a task system be the sum of the eligibility times of all subtasks with deadlines at most t_d.

Definition 2 τ is a feasible task system with the following properties.

(T1) τ misses a deadline under PD^2 at t_d.

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

(T3) No feasible task system satisfying (T1) and (T2) has a larger rank than τ.[3] □

By (T1), (T2), and Def. 1, exactly one subtask in τ misses its deadline: if several subtasks miss their deadline, all but one can be removed and the remaining subtask will still miss its deadline, contradicting (T2).

In the rest of this section, we use S to denote the PD^2 schedule of τ. (Note that we have assumed that S is an invalid schedule and our aim is to show that it is valid.) We henceforth assume that ties among subtasks in τ, not resolved by PD^2, are resolved consistently (for example, by using task identifiers).

We now prove some properties about τ and S. The following lemma is concerned with idle processors: 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.

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

Proof: Let k be the number of subtasks scheduled in slot t. Then, by (14), $LAG(\tau, t+1) = LAG(\tau, t) + \sum_{T \in \tau} \mathit{flow}(T, t) - k$. If $LAG(\tau, t) < LAG(\tau, t+1)$, then $k < \sum_{T \in \tau} \mathit{flow}(T, t)$. Because $\mathit{flow}(T, t) \leq wt(T)$ (by (P1)), we have $\sum_{T \in \tau} \mathit{flow}(T, t) \leq \sum_{T \in \tau} wt(T)$, which by (9) implies that $\sum_{T \in \tau} \mathit{flow}(T, t) \leq M$. Therefore, $k < M$, *i.e.*, there is a hole in slot t. □

Lemma 2 *The following properties hold for τ and S, the PD^2 schedule for τ, where T_i is any subtask in S.*

(a) *Let t be the slot in which T_i is scheduled. Then, $e(T_i) \geq min(r(T_i), t)$.*

(b) *Let t be the slot in which T_i is scheduled. If either $t < d(T_i)$ or $t = d(T_i) \wedge b(T_i) = 0$, then the successor of T_i is not eligible before $t+1$.*

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

(d) *There are no holes in slot t_d.*

(e) *$LAG(\tau, t_d + 1) > 0$.*

(f) *$LAG(\tau, t_d) > 0$.*

(g) *There exists $u \in [0, t_d - 1]$ such that $LAG(\tau, u) \leq 0$ and $LAG(\tau, u+1) > 0$.*

[3]Note that these conditions are being applied *in sequence*. We are not, for example, claiming that τ is of maximal rank; rather, it has the largest rank *among those task systems satisfying* (T1) *and* (T2).

Proof of (a): Suppose that $e(T_i)$ is less than $min(r(T_i), t)$. Consider the task system τ' obtained from τ with the following change: the eligibility time $e'(T_i)$ in τ' is $min(r(T_i), t)$. (Note that τ' is feasible. This follows from the feasibility proof for intra-sporadic task systems [3], which yields a schedule in which each subtask is scheduled in its PF-window.) Note that τ' has a larger rank than τ. It is easy to see that the relative priorities of the subtasks do not change for any slot $u \in [0, t_d]$, and therefore, the PD^2 schedules for τ' and τ are the same. Therefore, τ' misses a deadline at t_d as well. This contradicts (T3).

Proof of (b): Because T_i is scheduled at t, the successor of T_i is scheduled after slot t. Also, if either $d(T_i) > t$ or $d(T_i) = t \wedge b(T_i) = 0$, then by (6), we have $r(T_{i+1}) > t$ and hence, $r(T_k) > t$ for all $k > i$. In particular, if T_k is the successor of T_i, then $r(T_k) \geq t + 1$. Therefore, by (a), $e(T_k) \geq t + 1$.

Proof of (c): Suppose τ contains a subtask U_j with a deadline greater than t_d. U_j can be removed without affecting the scheduling of subtasks with higher priority. Thus, if U_j is removed, then a deadline is still missed at t_d. This contradicts (T2).

Proof of (d): If there were a hole in slot t_d, then the subtask that misses its deadline at t_d would have been scheduled there by PD^2. Contradiction.

Proof of (e): By (14), we have

$$LAG(\tau, t_d + 1) = \sum_{t=0}^{t_d} \sum_{T \in \tau} \mathit{flow}(T, t) - \sum_{t=0}^{t_d} \sum_{T \in \tau} S(T, t).$$

The first term on the left hand side of the above equation is the total flow in $[0, t_d]$, which is equal to the total number of subtasks in τ. The second term corresponds to the number of subtasks scheduled by PD^2 in $[0, t_d]$. Since exactly one subtask misses its deadline under PD^2, the difference between the two terms is 1. Therefore, $LAG(\tau, t_d + 1) = 1 > 0$.

Proof of (f): By (d), there are no holes in slot t_d. Hence, by Lemma 1, $LAG(\tau, t_d) \geq LAG(\tau, t_d + 1)$. Therefore, by (e), $LAG(\tau, t_d) > 0$.

Proof of (g): This follows from the fact that $LAG(\tau, 0) = 0$ and $LAG(\tau, t_d) > 0$ (from (f)). □

Later, using Lemmas 9 and 10, we show that $LAG(\tau, t_d + 1) \leq 0$, contradicting part (e) of Lemma 2. To establish these properties, we consider schedules obtained by removing some subtasks. (By (P3), the resulting system is a generalized intra-sporadic task system as well). Removing a subtask may cause other

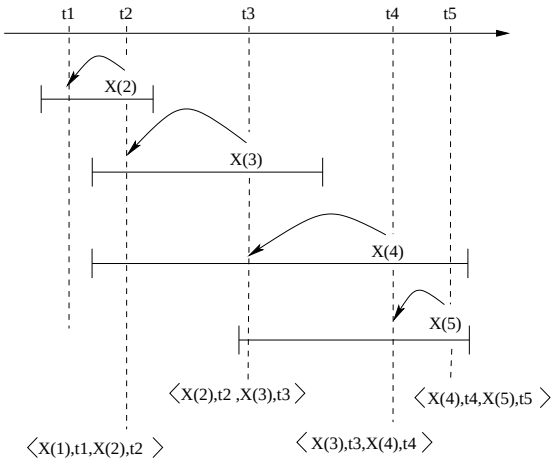

Figure 5: A chain of four displacements, caused by removing $X^{(1)}$, which was scheduled in slot t_1.

subtasks to shift elsewhere in the schedule.

Let $X^{(i)}$ denote a subtask of any task in τ. Assume that removing subtask $X^{(1)}$ scheduled at slot t_1 in S causes the subtask $X^{(2)}$ to be shifted from slot t_2 to slot t_1, where $t_1 \neq t_2$, which in turn may cause other shifts. We call this shift a *displacement* and represent it 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. An example of this is given in Fig. 5.

The lemmas below concern displacements. Lemma 3, proved in the appendix, states that a subtask removal can only cause left-shifts, as in Fig. 5. Lemma 4 indicates when a left-shift into a slot with a hole can occur.

Lemma 3 *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 4 *Let $\Delta = \langle X^{(1)}, t_1, X^{(2)}, t_2\rangle$ be a valid displacement in any PD^2 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 $X^{(1)}$.*

Proof: Because Δ is valid, $e(X^{(2)}) \leq t_1$. Since there is a hole in slot t_1 and $X^{(2)}$ is not scheduled there by PD^2, $X^{(2)}$ must be the successor of $X^{(1)}$. □

The next three lemmas state conditions that must hold for S. In each of these lemmas, we show that if these conditions do not hold, then a subtask can be removed from τ without causing the missed deadline in S to be met, contradicting (T2). (Note that removing a subtask could cause a sequence of left-shifts that results in the missed deadline being met.) We remind the reader of our assumption that ties among subtasks are resolved consistently. Thus, if task system τ' is obtained by removing a subtask from τ, then the relative priorities of two subtasks in τ' are the same as in τ.

Lemma 5 *Suppose there is a hole in slot $t \in [0, t_d]$ in S. Let U_j be a subtask scheduled at $t' < t$. If the eligibility time of the successor of U_j is at least $t+1$, then either $d(U_j) < t$ or $d(U_j) = t \ \wedge \ b(U_j) = 1$.*

Proof: We prove this by contradiction. Assume that the following is true.

$$d(U_j) > t \ \text{ or } \ d(U_j) = t \ \wedge \ b(U_j) = 0 \qquad (15)$$

Let τ' be the task system obtained by removing the subtask U_j from τ. Let S' be the PD^2 schedule for τ'.

Let the chain of displacements caused by removing U_j be $\Delta_1, \Delta_2, \ldots, \Delta_k$, where $\Delta_i = \langle X^{(i)}, t_i, X^{(i+1)}, t_{i+1}\rangle$, $X^{(1)} = U_j$ and $t_1 = t'$. By Lemma 3, $t_{i+1} > t_i$ for all $i \in [1, k]$. Note that at slot t_i, the priority of subtask $X^{(i)}$ is higher than the priority of $X^{(i+1)}$, because $X^{(i)}$ was chosen over $X^{(i+1)}$ in S. Thus, because $X^{(1)} = U_j$, by (15), for each subtask $X^{(i)}, i \in [1, k+1]$, either $d(X^{(i)}) > t$ or $d(X^{(i)}) = t \ \wedge \ b(X^{(i)}) = 0$. Therefore, by part (b) of Lemma 2, we have the following property.

(E) The eligibility time of the successor of $X^{(i)}$ (if it exists in τ) is at least $t+1$ for all $i \in [1, k+1]$.

We now show that the displacements do not extend beyond slot t, which implies that a deadline is still missed at t_d in S', contradicting (T2). Suppose that these displacements extend beyond slot t, *i.e.*,

$$t_{k+1} > t. \qquad (16)$$

Refer to Fig. 6(a). Let h be the smallest $i \in [2, k+1]$ such that $t_i > t$. Then, $t_{h-1} \leq t$. Since Δ_{h-1} is valid,

$$e(X^{(h)}) \leq t_{h-1}.$$

Now, if $t_{h-1} < t$, then by the above expression, $X^{(h)}$ is eligible at t. Because there is a hole in slot t, this implies that $X^{(h)}$ should have been scheduled at t in S instead of at $t_h > t$. Therefore, $t_{h-1} = t$. Because there is a hole in slot t, by Lemma 4, $X^{(h)}$ is the successor of $X^{(h-1)}$. However, because $e(X^{(h)}) \leq t_{h-1} = t$, by (E), $X^{(h)}$ cannot be the successor of $X^{(h-1)}$.

Thus, we have a contradiction of (16), implying that no subtask scheduled after t can get left-shifted. Hence, a deadline is still missed at t_d in S', contradicting (T2). Thus, either $d(U_j) < t$ or $d(U_j) = t \ \wedge \ b(U_j) = 1$. □

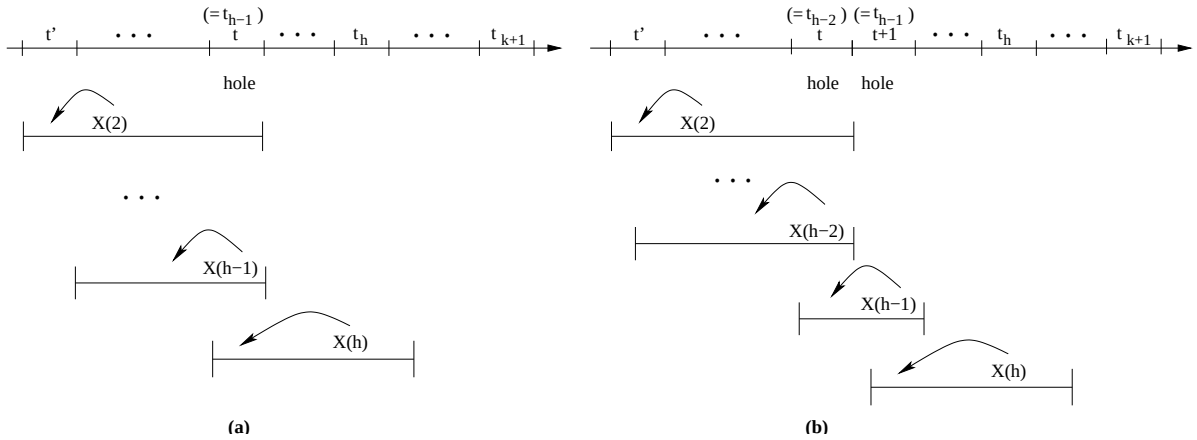

Figure 6: **(a)** Lemma 5. There is a hole in slot t. $X^{(h)}$ must be the successor of $X^{(h-1)}$. **(b)** Lemma 6. $X^{(h-2)}$, $X^{(h-1)}$, and $X^{(h)}$ must be consecutive subtasks of the same task because there are holes in slots t and $t+1$.

Lemma 6 *Let U_j be a subtask of a light task scheduled at $t' < d(U_j)$ in S. If the eligibility time of the successor of U_j is at least $d(U_j)+1$, then there cannot be holes in both $d(U_j)$ and $d(U_j)+1$.*

Proof: Note that by part (c) of Lemma 2, $d(U_j) \leq t_d$. Therefore, $t' < t_d$. Let $d(U_j) = t$. If $t = t_d$, then t satisfies the stated requirement because there is no hole in slot t_d (by part (d) of Lemma 2). In the rest of the proof, we assume that $t < t_d$, and hence $t+1 \leq t_d$. Suppose that there are holes in both t and $t+1$. Because there is a hole in slot t and (from the statement of the lemma) the eligibility time of the successor of U_j is at least $t+1$, by Lemma 5, either $d(U_j) < t$ or $d(U_j) = t \ \wedge \ b(U_j) = 1$. Because $d(U_j) = t$, the latter in fact must hold, *i.e.*, $d(U_j) = t \ \wedge \ b(U_j) = 1$. We now show that U_j can be removed without causing the missed deadline to be met, contradicting (T2). In particular, we show that the sequence of left-shifts caused by removing U_j does not extend beyond slot $t+1$.

Let the chain of displacements caused by removing U_j be $\Delta_1, \Delta_2, \ldots, \Delta_k$, where $\Delta_i = \langle X^{(i)}, t_i, X^{(i+1)}, t_{i+1} \rangle$, $X^{(1)} = U_j$ and $t_1 = t'$. By Lemma 3, we have $t_{i+1} > t_i$ for all $i \in [1,k]$. Also, the priority of $X^{(i)}$ is greater than the priority of $X^{(i+1)}$ at t_i, because $X^{(i)}$ was chosen over $X^{(i+1)}$ in S. Because U is light and $d(U_j) = t \ \wedge \ b(U_j) = 1$, this implies the following.

(P) For all $i \in [1, k+1]$, either **(i)** $d(X^{(i)}) > t$ or **(ii)** $d(X^{(i)}) = t$ and $X^{(i)}$ is the subtask of a light task.

Suppose this chain of displacements extends beyond $t+1$, *i.e.*, $t_{k+1} > t+1$. Let h be the smallest $i \in [1, k+1]$ such that $t_i > t+1$. Then, $t_{h-1} \leq t+1$.

If $t_{h-1} < t+1$, then $X^{(h)}$ is eligible to be scheduled in slot $t+1$ because $e(X^{(h)}) \leq t_{h-1}$ (by the validity of displacement Δ_{h-1}). Because there is a hole in slot $t+1$ in S, $X^{(h)}$ should have been scheduled there in S. Therefore, $t_{h-1} = t+1$ and by Lemma 4, $X^{(h)}$ must be the successor of $X^{(h-1)}$. By similar reasoning, because there is a hole in slot t, $t_{h-2} = t$ and $X^{(h-1)}$ must be the successor of $X^{(h-2)}$ (see Fig. 6(b)).

By (P), either $d(X^{(h-2)}) > t$ or $d(X^{(h-2)}) = t$ and $X^{(h-2)}$ is the subtask of a light task. In either case, $d(X^{(h-1)}) > t+1$. To see this, note that if $d(X^{(h-2)}) > t$, then because $X^{(h-1)}$ is the successor of $X^{(h-2)}$, by (6), $d(X^{(h-1)}) > t+1$. On the other hand, if $d(X^{(h-2)}) = t$ and $X^{(h-2)}$ is the subtask of a light task, then, by (P2), $d(X^{(h-1)}) > t+1$. Now, because $X^{(h-1)}$ is scheduled at $t+1$, by part (b) of Lemma 2, the successor of $X^{(h-1)}$ is not eligible before $t+2$, *i.e.*, $e(X^{(h)}) \geq t+2$. This implies that the displacement Δ_{h-1} is not valid. Thus, the chain of displacements cannot extend beyond $t+1$ and because $t+1 \leq t_d$, removing U_j cannot cause a missed deadline at t_d to be met. This contradicts (T2). Therefore, there cannot be holes in both t and $t+1$. □

The following lemma is the counterpart of Lemma 6 for heavy tasks. It is proved in the appendix.

Lemma 7 *Let U_j be a subtask of a heavy task scheduled in slot t' in S. If $t' < d(U_j) \wedge b(U_j) = 1$ and U_j's successor is not eligible before $d(U_j)+1$, then there exists a slot $t \in [d(U_j), min(D(U_j), t_d)]$ with no hole.*

Lemma 8 *If $LAG(\tau, t) < LAG(\tau, t+1)$ for some t in S, then there exists a task that is active at t but not scheduled at t.*

Proof: Let I denote the set of tasks that are inactive at slot t. Let A denote the set of tasks that are active but not scheduled in slot t, and let B denote the set of active tasks scheduled in slot t. Note that these three

sets are disjoint and $I \cup A \cup B = \tau$. Our aim is to show that A is non-empty.

By Lemma 1, there is at least one hole in slot t. Let the number of these holes be h. Then, the number of allocations in slot t in S is $M-h$, *i.e.*, $\sum_{T\in\tau} S(T,t) = M - h$. By (14), $LAG(\tau, t+1) = LAG(\tau,t) + \sum_{T\in\tau}(\mathit{flow}(T,t) - S(T,t))$. Thus, because $LAG(\tau,t) < LAG(\tau,t+1)$, we have $\sum_{T\in\tau} \mathit{flow}(T,t) > M - h$. Because $\sum_{T\in I} \mathit{flow}(T,t) = 0$, it follows that $\sum_{T\in A\cup B} \mathit{flow}(T,t) > M-h$. Therefore, by (P1), $\sum_{T\in A\cup B} wt(T) > M-h$. Because the number of tasks scheduled at t is $M-h$, $|B| = M-h$. Because each task's weight is at most one, $\sum_{T\in B} wt(T) \le M-h$. Thus, $\sum_{T\in A} wt(T) > 0$. Hence, A is not empty. □

The following definition is used in the next lemma.

Definition 3 Subtask U_j is the *critical subtask of U at slot t* iff $e(U_j) \le t \le d(U_j)$ and no other subtask U_k of U, where $k > j$, satisfies $e(U_k) \le t \le d(U_k)$. □

The next lemma is used later to contradict the fact that $LAG(\tau, t_d+1) > 0$ (part (e) of Lemma 2).

Lemma 9 *Suppose that for some $t \in [0, t_d - 1]$, $LAG(\tau,t) \le 0$ and $LAG(\tau,t+1) > 0$. Let A denote the set of tasks that are active but not scheduled in slot t. If any task in A is light, then $LAG(\tau,t+2) \le 0$.*

Proof: Because $LAG(\tau,t) \le 0$ and $LAG(\tau,t+1) > 0$,

$$LAG(\tau,t) < LAG(\tau,t+1) \tag{17}$$

By Lemma 1, we have the following for some $h \ge 1$.

(H) There are h holes in slot t.

Let I denote the set of tasks that are inactive at slot t, and let B denote the set of tasks scheduled in slot t. Note that I, A, and B are disjoint and $I \cup A \cup B = \tau$. By (17) and Lemma 8, A is non-empty.

Let U be any task in A. Because U is active at t, t lies within the eligibility interval of some subtask of U. Therefore, there exists a subtask U_j that is critical at slot t. Let t' be the slot at which U_j is scheduled.

Let U_k be any subtask satisfying $e(U_k) \le t \le d(U_k)$ such that U_k is scheduled after t and its predecessor (if it exists) is scheduled before t. (Because $U \in A$, no subtask of U is scheduled at t.) Because there is a hole in slot t (by (H)), and because U_k is eligible at t, it should have been scheduled there by PD^2. Thus, for all subtasks U_k with $e(U_k) \le t \le d(U_k)$, U_k is scheduled before t. In particular, $t' < t$.

Note that, by Def. 3, the successor of U_j has an eligibility time of at least $t+1$. Because there is a hole in slot t, by Lemma 5, either $d(U_j) < t$ or $d(U_j) = t \ \wedge\ b(U_j) = 1$. Also, by Def. 3, $d(U_j) \ge t$. Therefore, $d(U_j) = t \ \wedge\ b(U_j) = 1$. This is true for the critical subtask of each task in A. Let C denote the set of critical subtasks of all the tasks in A. Then,

$$\forall U_j \in C,\ d(U_j) = t \ \wedge\ b(U_j) = 1. \tag{18}$$

Because A is not empty, by the statement of the lemma, A contains at least one light task. Let $U \in A$ be light. Because there is a hole in slot $t = d(U_j)$, by Lemma 6, there is no hole in slot $t+1 = d(U_j)+1$. We are now ready to show that $LAG(\tau,t+2) \le 0$.

Let the sum of the weights of all tasks in set B be W. Then, the total sum of the weights of all tasks in $I \cup A$ is at most $M - W$. For all tasks $T \in I$, $\mathit{flow}(T,t) = 0$. By (P1), this implies that for all tasks $T \in I$, $\mathit{flow}(T,t) + \mathit{flow}(T,t+1) \le wt(T)$. Consider $U \in A$. Let U_j denote its critical subtask, and let U_k denote the successor of U_j. Because the eligibility time of U_k is at least $t+1$, $r(U_k) \ge t+1 = d(U_j)+1$. Therefore, by (P4), $\mathit{flow}(U,t) + \mathit{flow}(U,t+1) \le wt(U)$ for all $U \in A$. Because $\sum_{U\in I\cup A} wt(U) \le M - W$, $\sum_{U\in I\cup A}(\mathit{flow}(U,t) + \mathit{flow}(U,t+1)) \le M - W$. Also, by (P1), $\sum_{U\in B}(\mathit{flow}(U,t) + \mathit{flow}(U,t+1)) \le 2(\sum_{U\in B} wt(U)) = 2W$. Therefore,

$$\sum_{U\in\tau}(\mathit{flow}(U,t) + \mathit{flow}(U,t+1)) \le M + W. \tag{19}$$

By (H), there are h holes in slot t. Because there are no holes in slot $t+1$,

$$\sum_{U\in\tau}(S(U,t)+S(U,t+1)) = M-h+M = 2M-h. \tag{20}$$

Because the number of tasks scheduled at t is $M-h$, and the weight of each task is at most one, we have $\sum_{U\in B} wt(U) \le M-h$, *i.e.*, $W \le M-h$. This relation and (19) and (20) together imply that $\sum_{U\in\tau} \mathit{flow}(U,t) + \mathit{flow}(U,t+1) \le \sum_{U\in\tau} S(U,t) + S(U,t+1)$. Using this relation in the identity (obtained from (14)), $LAG(\tau,t+2) = LAG(\tau,t) + \sum_{U\in\tau}(\mathit{flow}(U,t) + \mathit{flow}(U,t+1)) - \sum_{U\in\tau}(S(U,t) + S(U,t+1))$, and the fact that $LAG(\tau,t) \le 0$, we have $LAG(\tau,t+2) \le 0$. □

The following lemma, which is proved in the appendix, generalizes Lemma 9 by allowing heavy tasks.

Lemma 10 *For any $t \in [0, t_d-1]$, if $LAG(\tau,t) \le 0$ and $LAG(\tau,t+1) > 0$, then there exists $u \in [t+2, t_d+1]$ such that $LAG(\tau,u) \le 0$.*

By Lemma 2, part (g), there exists a $u \in [0, t_d-1]$ such that $LAG(\tau,u) \le 0$ and $LAG(\tau,u+1) > 0$. Let v be the largest such u. Because $v \le t_d - 1$, we have

$v + 2 \leq t_d + 1$. By Lemma 10, $LAG(\tau, t) \leq 0$ for some $t \in [v + 2, t_d + 1]$. By Lemma 2, parts (e) and (f), t cannot be t_d or $t_d + 1$. Thus, $t \in [0, t_d - 1]$. Because $LAG(\tau, t_d) > 0$, this contradicts the maximality of v. Hence, τ cannot exist, *i.e.*, our original assumption that PD^2 is not optimal is incorrect. Thus, we have the following.

Theorem 1 *PD^2 is optimal for scheduling generalized intra-sporadic tasks on M processors.*

Corollary 1 *PD^2 is optimal for scheduling intra-sporadic tasks on M processors.*

5 Concluding Remarks

In this paper, we have shown that PD^2, the most efficient optimal Pfair scheduling algorithm proposed to date, correctly schedules any feasible intra-sporadic task system on M processors. This paper is the first to show that either sporadic or intra-sporadic tasks can be optimally scheduled on systems of more than two processors.

Two key insights led to our proof: the development of a notion of lag for intra-sporadic systems that can be used to sufficiently predict where holes exist in a schedule, and the identification of certain minimality conditions (Defs. 1 and 2) that facilitate the reasoning. It is these notions that distinguish our proof from previous proofs for Pfair/ERfair scheduling algorithms.

Its rate-based properties make PD^2 potentially useful in several application domains. In joint work with researchers at NC State University, we are currently investigating the potential of using PD^2 to schedule rate-based packet flows in wave-division-multiplexing (WDM) networks. In WDM networks, optical multiplexing techniques are used to send multiple packets over the same link in parallel. In a similar vein, PD^2 can be used to solve the parallel switching problem in ATM networks mentioned in [1].

Rate-based scheduling algorithms are also useful for multiplexing independently-authored applications on the same server. This is because such algorithms ensure *temporal isolation* among applications (no "misbehaving" application can execute faster than its proscribed rate, unless there is spare processing capacity). Many systems that could benefit from the use of rate-based scheduling algorithms have workloads that necessitate the use of multiple processors. Consider, for example, the proliferation of Internet service providers that host third-party websites on multiprocessor servers [7]. One such service provider, Ensim Corp., has in fact deployed multiprocessor rate-based scheduling algorithms in its product line; these algorithms have been evaluated empirically by Chandra *et al.* [7, 8]. Unfortunately, Chandra *et al.* give no formal correctness proofs for the algorithms they consider. In this paper, we have given the first ever general optimality proof for a multiprocessor rate-based scheduling algorithm. The techniques in our proof are not unique to PD^2 and should be applicable to other rate-based algorithms as well.

References

[1] J. Anderson, S. Baruah, and K. Jeffay. Parallel switching in connection-oriented networks. *Proc. of the 20th IEEE Real-Time Systems Symp.*, pp. 200–209, 1999.

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

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

[4] J. Anderson and A. Srinivasan. Mixed Pfair/ERfair scheduling of asynchronous periodic tasks. *Proc. of the 13th Euromicro Conf. on Real-Time Systems*, pp. 76–85, June 2001.

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

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

[7] A. Chandra, M. Adler, P. Goyal, and P. Shenoy. Surplus fair scheduling: A proportional-share cpu scheduling algorithm for symmetric multiprocessors. In *Proc. of the Fourth Symposium on Operating System Design and Implementation (OSDI)*, 2000.

[8] A. Chandra, M. Adler, and P. Shenoy. Deadline fair scheduling: Bridging the theory and practice of proportionate-fair scheduling in multiprocessor servers. *Proc. of the 7th IEEE Real-Time Technology and Applications Symp.*, 2001.

[9] K. Jeffay and S. Goddard. The rate-based execution model. *Proc. of the 20th IEEE Real-Time Systems Symp.*, pp. 304–314, 1999.

[10] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *JACM*, 20(1):46–61, 1973.

[11] A. Mok. Fundamental design problems of distributed systems for the hard-real-time environment. Technical Report MIT/LCS/TR-297, 1983.

Appendix: Remaining Proofs

In this appendix, we present all proofs omitted earlier. We begin with Lemma 3.

Lemma 3 *Let $X^{(1)}$ be a subtask that is removed from τ, and let the resulting chain of displacements in S (the PD^2 schedule for τ) be $\Delta_1, \Delta_2, \ldots, \Delta_k$, where $\Delta_i = \langle X^{(i)}, t_i, X^{(i+1)}, t_{i+1} \rangle$. Then $t_{i+1} > t_i$ for all $i \in [1, k]$.*

Proof: Let τ' be the task system obtained by removing $X^{(1)}$ from τ, and let S' be its PD^2 schedule. Note that the last displacement creates a hole at t_{k+1} in S'. Suppose $t_{i+1} > t_i$ is not true for some $i \in [1, k]$. Let

$$t_j = \mathbf{min}(t_i \mid t_{i+1} < t_i).$$

(Informally, the leftmost right-shift occurs when $X^{(j+1)}$ scheduled at t_{j+1} shifts to t_j.) We consider two cases depending on whether j is equal to k. If $j = k$, then the last displacement will be as shown in Fig. 7(a). Note that $X^{(k+1)}$ is eligible to be scheduled in slot t_{k+1} in S', because it is scheduled there in S and no subtask (in particular, its predecessor) scheduled before t_{k+1} is shifted to t_{k+1} (by choice of j). Because there will be a hole in slot t_{k+1} in S' and $t_{k+1} < t_k$, this contradicts the greedy behavior of the PD^2 algorithm.

If $j < k$, then by our choice of j, $t_{j+1} < t_j$ and the displacements are as in Fig. 7(b). By the minimality of t_j, $t_{j+2} > t_{j+1}$. Thus, at t_{j+1}, $X^{(j+1)}$ was chosen over $X^{(j+2)}$ in S. After the displacements, $X^{(j+1)}$ is scheduled at t_j and $X^{(j+2)}$ at t_{j+1}. This contradicts our assumption that ties are broken consistently in S and S'. Hence, $t_{i+1} > t_i$ for all $i \in [1, k]$. □

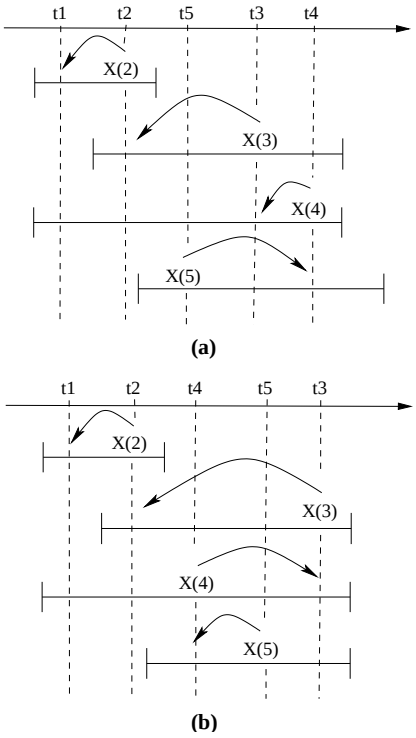

Figure 7: Lemma 3. A chain of $k = 4$ displacements is shown. **(a)** The leftmost right shift occurs when $X^{(5)}$ shifts from t_5 to t_4, *i.e.*, $j = k$. **(b)** The leftmost right shift occurs when $X^{(4)}$ shifts from t_4 to t_3, *i.e.*, $j < k$ (here, $t_j = t_3, t_{j+1} = t_4$, and $t_{j+2} = t_5$).

We now prove (P4), from Sec. 4, and several properties that are used later in proof of Lemma 10. With the exception of (B3) (see below), all of the properties below apply to generalized intra-sporadic task systems.

(B1) Let T_i be a subtask with $b(T_i) = 1$. If T_{i+1} exists, then $f(T_i, d(T_i)) + f(T_{i+1}, r(T_{i+1})) = wt(T)$.

Proof: By (12), $f(T_i, d(T_i)) = i - (\lceil i/wt(T) \rceil - 1) \times wt(T)$, and $f(T_{i+1}, r(T_{i+1})) = (\lfloor i/wt(T) \rfloor + 1) \times wt(T) - i$. Since $b(T_i) = 1$, by (5), $\lceil i/wt(T) \rceil = \lfloor i/wt(T) \rfloor + 1$. This implies that $f(T_{i+1}, r(T_{i+1})) = \lceil i/wt(T) \rceil \times wt(T) - i$. Hence, $f(T_i, d(T_i)) + f(T_{i+1}, r(T_{i+1})) = wt(T)$. (See Fig. 3.) □

(B2) Let T_i be a subtask such that $b(T_i) = 1$. If T_{i+1} exists and is released late, *i.e.*, $r(T_{i+1}) \geq d(T_i) + 1$, then $\mathit{flow}(T, d(T_i)) + \mathit{flow}(T, d(T_i) + 1) \leq wt(T)$.

Proof: By (6) and (8), it follows that $r(T_k) > d(T_i)+1$ for all $k > i + 1$. Similarly, $d(T_j) < d(T_i)$ for all $j < i$. This implies that the slot $d(T_i)$ lies within the PF-window of only one subtask, namely, T_i, and the slot $d(T_i) + 1$ can lie within the PF-window of only one subtask, namely, T_{i+1}. Thus, the contribution to the flow in slot $d(T_i)$ is $f(T_i, d(T_i))$ and the contribution to slot $d(T_i) + 1$ is at most $f(T_{i+1}, r(T_{i+1}))$. Hence, by (B1), $\mathit{flow}(T, d(T_i)) + \mathit{flow}(T, d(T_i) + 1) \leq wt(T)$. □

(B3) If T_i and T_k are subtasks of an *intra-sporadic* heavy task T such that $k > i$ and $r(T_k) \leq D(T_i)$, then $f(T_i, d(T_i)) + f(T_k, r(T_k)) \leq wt(T)$.

Proof: If $b(T_i) = 0$, then $D(T_i) = d(T_i)$. In this case, $r(T_k) > D(T_i)$ holds, since (6) implies $r(T_k) > d(T_i)$ (thus, no task T_k exists such that $k > i$ and $r(T_k) \leq D(T_i)$). In the rest of the proof, we assume that $b(T_i) = 1$. Note that, by the definition of D (the group deadline), because $r(T_k) \leq D(T_i)$, for all $j \in [i + 1, k - 1]$, $|w(T_j)| = 2$ and $b(T_j) = 1$. Because $|w(T_j)| = 2$, we have $d(T_j) = r(T_j) + 1$. Because the total flow for a subtask is one, this implies that

$$\forall j \in [i+1, k-1], f(T_j, r(T_j)) + f(T_j, d(T_j)) = 1. \quad (21)$$

Because $b(T_j) = 1$, by (B1), we have for all $j \in [i, k - 1], f(T_j, d(T_j)) + f(T_{j+1}, r(T_{j+1})) = wt(T)$. There-

fore, $\sum_{j=i}^{k-1} f(T_j, d(T_j)) + f(T_{j+1}, r(T_{j+1})) = (k-i) \times wt(T)$. Rewriting, we get $f(T_i, d(T_i)) + f(T_k, r(T_k)) + \sum_{j=i+1}^{k-1} (f(T_j, r(T_j)) + f(T_j, d(T_j))) = (k-i) \times wt(T)$. By (21), this implies that

$$f(T_i, d(T_i)) + f(T_k, r(T_k)) + k - i - 1 = (k-i) \times wt(T).$$

Therefore, $f(T_i, d(T_i)) + f(T_k, r(T_k)) = wt(T) + (k - i - 1)(wt(T) - 1)$. Because $k \geq i+1$ and $wt(T) \leq 1$ for all T, we have $f(T_i, d(T_i)) + f(T_k, r(T_k)) \leq wt(T)$. □

(B4) Let T_i be a subtask of a heavy task T and let T_k ($k > i$) be a subtask such that $r(T_k) \leq D(T_i)$. Then, $f(T_i, d(T_i)) + f(T_k, r(T_k)) \leq wt(T)$.

Proof: Because T is a generalized intra-sporadic task, there is an intra-sporadic task U such that $wt(U) = wt(T)$, all subtasks between U_i and U_k are present, and $r(U_k) = r(T_k)$. Hence, $r(U_k) \leq D(U_i)$. By (B3), $f(U_i, d(U_i)) + f(U_k, r(U_k)) \leq wt(U)$. Corresponding subtasks in T and U have identical flows. Thus, $f(T_i, d(T_i)) + f(T_k, r(T_k)) \leq wt(T)$. □

(B5) Let T_i be a subtask of a heavy task T such that $b(T_i) = 1$ and let T_k be the successor of T_i. If $r(T_k) \geq u$, where $u \in [d(T_i) + 1, D(T_i)]$, then $\mathit{flow}(T, d(T_i)) + \mathit{flow}(T, u) \leq wt(T)$.

Proof: Since $b(T_i) = 1$, by the definition of D, $D(T_i) > d(T_i)$. Since $u > d(T_i)$ and T_k is T_i's successor, if $r(T_k) > u$, then $\mathit{flow}(T, u) = 0$. Thus, by (P1), $\mathit{flow}(T, d(T_i)) + \mathit{flow}(T, u) \leq wt(T)$. The other possibility is $r(T_k) = u$, which implies $r(T_k) \leq D(T_i)$. In this case, by (B4), $f(T_i, d(T_i)) + f(T_k, r(T_k)) \leq wt(T)$. Thus, $\mathit{flow}(T, d(T_i)) + \mathit{flow}(T, u) \leq wt(T)$. □

(P4) Let T_i be a subtask such that $b(T_i) = 1$ and let T_k be the successor of T_i. If $r(T_k) \geq d(T_i) + 1$, then $\mathit{flow}(T, d(T_i)) + \mathit{flow}(T, d(T_i) + 1) \leq wt(T)$.

Proof: If $k = i + 1$, then by (B2), $\mathit{flow}(T, d(T_i)) + \mathit{flow}(T, d(T_i) + 1) \leq wt(T)$. Also, if $r(T_k) > d(T_i) + 1$, then $\mathit{flow}(T, d(T_i) + 1) = 0$. Hence, by (P1), $\mathit{flow}(T, d(T_i)) + \mathit{flow}(T, d(T_i) + 1) \leq wt(T)$.

In the rest of the proof, we assume that $k > i+1$ and $r(T_k) = d(T_i)+1$. We first show that T must be heavy. If T is light, then by (P2), we have $d(T_{i+1}) > d(T_i)+1$. By (6), we also have $r(T_k) \geq d(T_{i+1})$ and therefore, $r(T_k) > d(T_i)+1$, which contradicts $r(T_k) = d(T_i)+1$.

Thus, T is heavy. Because $b(T_i) = 1$, by the definition of D, $D(T_i) > d(T_i)$. Hence, because $r(T_k) = d(T_i) + 1$, we have $r(T_k) \leq D(T_i)$. Thus, by (B4), $\mathit{flow}(T, d(T_i)) + \mathit{flow}(T, d(T_i) + 1) \leq wt(T)$. □

Finally, we prove Lemmas 7 and 10.

Lemma 7 *Let U_j be a subtask of a heavy task scheduled in slot t' in S. If $t' < d(U_j) \wedge b(U_j) = 1$ and U_j's successor is not eligible before $d(U_j) + 1$, then there exists a slot $t \in [d(U_j), min(D(U_j), t_d)]$ with no hole.*

Proof: By part (c) of Lemma 2, $d(U_j) \leq t_d$, and hence, $t' < t_d$. If $min(D(U_j), t_d) = t_d$, then by part (f) of Lemma 2, $t = t_d$ satisfies the stated requirement. In the rest of the proof, assume that $D(U_j) < t_d$. Let $u = d(U_j)$ and $v = D(U_j)$. Since $b(U_j) = 1$, by the definition of D, $D(U_j) > d(U_j)$, *i.e.*,

$$u < v. \tag{22}$$

Suppose that the following property holds.

(H) There is a hole in slot t for all $t \in [u, v]$.

Given (H), we show that removing U_j does not cause the missed deadline to be met, contradicting (T2). Let $\Delta_1, \Delta_2, \ldots, \Delta_k$ be the chain of displacements caused by removing U_j, where $\Delta_i = \langle X^{(i)}, t_i, X^{(i+1)}, t_{i+1} \rangle$, $X^{(1)} = U_j$, and $t_1 = t'$. By Lemma 3, $t_{i+1} > t_i$ for all $i \in [1, k-1]$. Also, $X^{(i)}$ has higher priority than $X^{(i+1)}$ at t_i since $X^{(i)}$ is scheduled at t_i in S. Thus, for all $i \in [2, k+1]$, one of the following holds:

(a) $d(X^{(i)}) > u$,

(b) $d(X^{(i)}) = u \ \wedge \ b(X^{(i)}) = 0$, or

(c) $d(X^{(i)}) = u \ \wedge \ b(X^{(i)}) = 1 \ \wedge \ D(X^{(i)}) \leq v$.

We now show that the displacements do not extend beyond v (which implies that U_j can be removed without causing the missed deadline to be met). Suppose, to the contrary, they do extend beyond v, *i.e.*, $t_{k+1} > v$.

Let t_g be the largest t_i such that $t_i < u$ and let t_h be the smallest t_i such that $t_i > v$. (Note that such a t_g exists because $t_1 < u$.) Then, by (H), there are holes in all slots in $[t_{g+1}, t_{h-1}]$. Thus, by Lemma 4,

$$\forall i \in [g+1, h-1],\ X^{(i+1)} \text{ is the successor of } X^{(i)}. \tag{23}$$

We now show that the situation is as shown in Fig. 8(a). By Lemma 3, $t_{i+1} \geq t_i + 1$ for all $i \in [g+1, h-2]$. If $t_{i+1} > t_i + 1$, then because **(i)** there is a hole in slot $t_i + 1$, **(ii)** $e(X^{(i+1)}) \leq t_i$, and **(iii)** $X^{(i+1)}$'s predecessor $X^{(i)}$ is scheduled at t_i, $X^{(i+1)}$ should have been scheduled at $t_i + 1$ in S, a contradiction. Thus, $t_{i+1} = t_i + 1$ for all $i \in [g+1, h-2]$. By similar reasoning, we have the following.

$$t_{g+1} = u \ \wedge \ t_{h-1} = v \tag{24}$$

$$\forall i \in [g+1, h-1],\ t_i = u + i - (g+1) \tag{25}$$

Earlier, we showed that one of (a)–(c) holds for all $i \in [2, k+1]$. If either $d(X^{(g+1)}) > u$ or $d(X^{(g+1)}) = u \wedge b(X^{(g+1)}) = 0$, then since $X^{(g+1)}$ is scheduled at u,

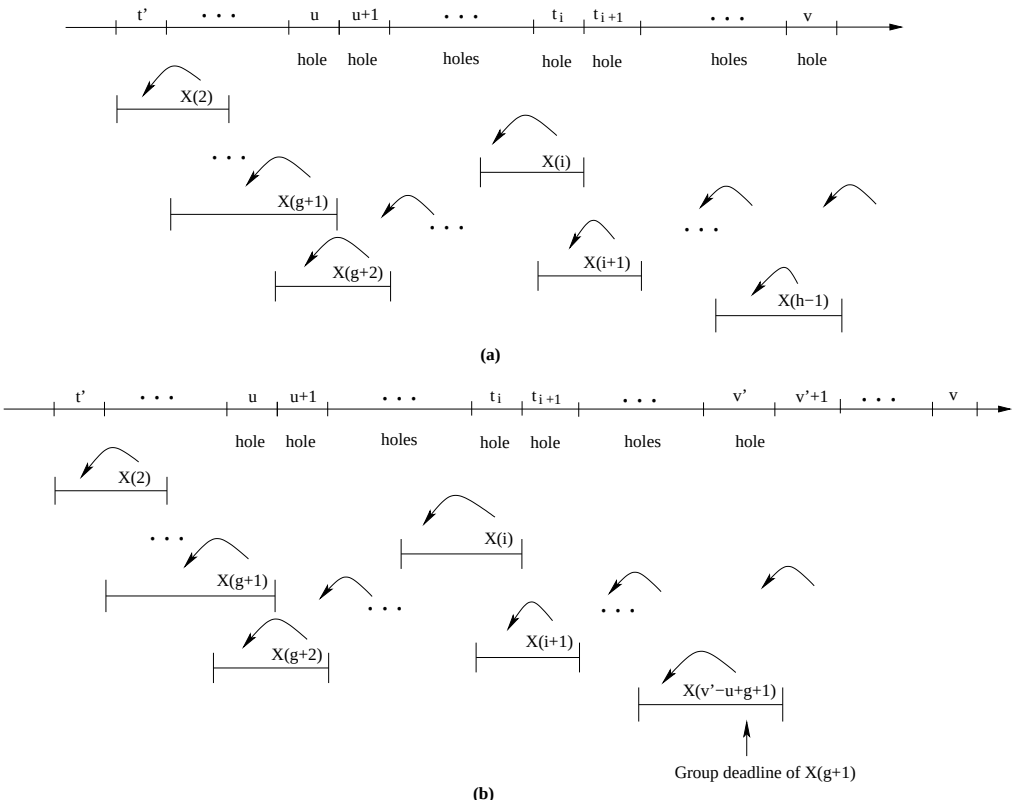

Figure 8: Lemma 7. **(a)** There are holes in all slots in $[u, v]$. $X^{(i)}$ scheduled at t_i displaces $X^{(i-1)}$ scheduled at t_{i-1}. Also, by (25), the t_i's are consecutive and satisfy $t_i = u + i - (g + 1)$. Further, $X^{(h-1)}$ is the subtask scheduled in slot v. **(b)** Case 2. $D(X^{(g+1)}) = v'$. Hence, either $d(X^{(v'-u+g+1)}) = v' \wedge b(X^{(v'-u+g+1)}) = 0$ (as depicted) or $d(X^{(v'-u+g+1)}) > v'$.

by Lemma 2, part (b), $e(X^{(g+2)}) \geq u + 1$ (recall that, by (23), $X^{(g+2)}$ is the successor of $X^{(g+1)}$). In other words, the displacement Δ_g is not valid. Therefore,

$$d(X^{(g+1)}) = u \wedge b(X^{(g+1)}) = 1 \wedge D(X^{(g+1)}) \leq v. \quad (26)$$

We now consider two cases. In each, we show that the displacements do not extend beyond v, as desired.

Case 1: $X^{(g+1)}$ is the subtask of a light task. By (22), $u + 1 \leq v$ and hence, by (H), there is a hole in both u and $u + 1$. Also, by (25) and (25), we have $v = u + (h-1) - (g+1) = u + h - g - 2$. Hence, because $u < v$ (by (22)), we have $h > g + 2$, *i.e.*,

$$h \geq g + 3.$$

We now show that the displacement Δ_{g+2} is not valid.

By (25), $X^{(i)}$ is scheduled at $u + i - (g + 1)$ in S for all $i \in [g + 1, h - 1]$. In particular, $X^{(g+1)}$ and $X^{(g+2)}$ are scheduled at u and $u + 1$, respectively. Observe that $d(X^{(g+1)}) = u$ (by 26), $X^{(g+1)}$ is the subtask of a light task (our assumption for Case 1), and $X^{(g+2)}$ is the successor of $X^{(g+1)}$ (by (23)). Thus, by (P1), $d(X^{(g+2)}) \geq u + 2$. Since $X^{(g+2)}$ is scheduled at $u + 1$, by Lemma 2, part (b), the eligibility time of $X^{(g+2)}$'s successor is at least $u + 2$, *i.e.*, $e(X^{(g+3)}) \geq u + 2$. ($X^{(g+3)}$ exists because $h \geq g + 3$.) This implies that the displacement Δ_{g+2} is not valid. Hence, the displacements do not extend beyond $u + 1$ (and hence v).

Case 2: $X^{(g+1)}$ is the subtask of a heavy task. Let $v' = D(X^{(g+1)})$. By (26), $v' \leq v$. We now show that the displacements cannot extend beyond v' (and hence v). By (25), $X^{(i)}$ is scheduled at $u + i - (g + 1)$ in S for all $i \in [g+1, h-1]$. By (23), all $X^{(i)}$ where $i \in [g + 1, h]$ are subtasks of the same heavy task. We now show that the displacement $\Delta_{v'-u+g+1}$ is not valid.

By (25), $t_{v'-u+g+1} = v'$. Because $X^{(i)}$ is scheduled at t_i, the subtask scheduled at v' is $X^{(v'-u+g+1)}$. Since $X^{(i+1)}$ is the successor of $X^{(i)}$, by (8), $d(X^{(i)}) > d(X^{(i-1)})$ for all $i \in [g + 2, v' - u + g + 1]$. Hence, because $d(X^{(g+1)}) = u \wedge b(X^{(g+1)}) = 1$ (by (26)),

$$\forall i \in [g+1, v'-u+g+1],\ d(X^{(i)}) \geq u+i-(g+1). \quad (27)$$

In particular, $d(X^{(v'-u+g+1)}) \geq v'$.

We now show that if $d(X^{(v'-u+g+1)}) = v'$, then $b(X^{(v'-u+g+1)}) = 0$. In this case, be-

cause $d(X^{(v'-u+g)}) < d(X^{(v'-u+g+1)})$, we have $d(X^{(v'-u+g)}) < v'$. By (27), $d(X^{(v'-u+g)}) \geq v' - 1$. Therefore, $d(X^{(v'-u+g)}) = v' - 1$. Similarly, by induction, $d(X^{(i)}) = u + i - (g+1)$ for all $i \in [g+1, v'-u+g+1]$. (Refer to Fig. 8(b).) Because $D(X^{(g+1)}) = v'$, by the definition of D, $b(X^{(v'-u+g+1)}) = 0$.

Thus, either $d(X^{(v'-u+g+1)}) > v'$ or $d(X^{(v'-u+g+1)}) = v' \wedge b(X^{(v'-u+g+1)}) = 0$. Since $X^{(v'-u+g+1)}$ is scheduled at v', by Lemma 2, part (b), the eligibility time of the successor of $X^{(v'-u+g+1)}$ is at least $v'+1$. Hence, $\Delta_{v'-u+g+1}$ is not valid. Thus, the displacements do not extend beyond v' (and hence v). □

The following claims are used in proving Lemma 10.

Claim 1 *If U_j is scheduled in $t \in [0, t_d]$ in S, where $t \leq d(U_j)$, and if there is a hole in t, then $d(U_j) = t$.*

Proof: We show that if $d(U_j) > t$, then we can remove subtask U_j without causing any displacements. Because $d(U_j) > t$ and U_j is scheduled at t, by part (b) of Lemma 2, its successor (if it exists) is not eligible before $t+1$. Also, any other subtask scheduled after t is not eligible at t. (If it were eligible, because there is a hole in t, PD^2 would have scheduled that subtask.) Thus, no subtask can shift into slot t if U_j is removed. This contradicts (T2). Hence, $d(U_j) \leq t$. By Lemma 2, part (d), there are no holes in t_d. Because there is a hole in slot t, we have $t < t_d$. Therefore, U_j does not miss its deadline, *i.e.*, $d(U_j) = t$. □

Claim 2 *Suppose there is a hole in slot $t \in [0, t_d]$. Let U_j be a subtask scheduled at $t' \leq t$. If the eligibility time of the successor of U_j is at least $t+1$, then $d(U_j) \leq t$.*

Proof: If $t' = t$, then by Lemma 1, $d(U_j) = t$. On the other hand, if $t' < t$, then by Lemma 5, $d(U_j) \leq t$. □

Lemma 10 *For any $t \in [0, t_d - 1]$, if $LAG(\tau, t) \leq 0$ and $LAG(\tau, t+1) > 0$, then there exists $v \in [t+2, t_d+1]$, such that $LAG(\tau, v) \leq 0$.*

Proof: Because $LAG(\tau, t) \leq 0$ and $LAG(\tau, t+1) > 0$,

$$LAG(\tau, t) < LAG(\tau, t+1). \tag{28}$$

Thus, by Lemma 1, we have the following property.

(H) There is at least one hole in slot t.

Let I denote the set of inactive tasks at slot t. Let A denote the set of tasks that are active but not scheduled in slot t and let B denote the set of tasks scheduled in slot t. Note that these three sets are disjoint and $I \cup A \cup B = \tau$. By (28) and Lemma 8, A is non-empty.

Let U be any task in A. Because U is active at t, t lies within the eligibility interval of some subtask of U. Thus, there exists a subtask U_j that is critical at slot t (refer to Def. 3 in Sec. 4). Let t' be the slot where U_j is scheduled. We now show that $t' < t$. Suppose $t' > t$. Then there exists a subtask U_k satisfying $e(U_k) \leq t \leq d(U_k)$ such that U_k is scheduled after t and its predecessor (if it exists) is scheduled before t. (Because $U \in A$, no subtask of U is scheduled at t.) Since there is a hole in slot t (by (H)), and since U_k is eligible at t, it should have been scheduled there by PD^2. Thus, there cannot exist any such subtask, implying that $t' < t$.

Note that, by Def. 3, the successor of U_j has an eligibility time of at least $t+1$. Because there is a hole in slot t, by Lemma 5, either $d(U_j) < t$ or $d(U_j) = t \ \wedge \ b(U_j) = 1$. Also, by Def. 3, $d(U_j) \geq t$. Therefore, $d(U_j) = t \ \wedge \ b(U_j) = 1$. This is true for the critical subtask of each task in A. Let C denote the set of critical subtasks of all the tasks in A. Then,

$$\forall\, U_j \in C,\ d(U_j) = t \ \wedge \ b(U_j) = 1. \tag{29}$$

If any task in A is light, then by Lemma 9, $LAG(\tau, t+2) \leq 0$, which establishes our proof obligation. We henceforth assume all tasks in A are heavy.

Let L_i be the lowest-priority subtask in C. By (29),

$$\forall U_j \in C,\ d(U_j) = t \wedge b(U_j) = 1 \wedge D(U_j) \geq D(L_i). \tag{30}$$

Because L_i is a critical subtask, the eligibility time of its successor L_k is at least $t+1$. Thus, by Lemma 7, there is a slot in $[t, \min(D(L_i), t_d)]$ with no hole. Let u be the slot defined as follows.

(U) u is the earliest slot in $[t, \min(D(L_i), t_d)]$ with no hole.

Fig. 9 depicts this situation. By (U) and (H),

$$u \geq t+1, \tag{31}$$

and there are holes in all slots in $[t, u-1]$. We now establish the following property about tasks in A.

> **Claim 3** *All tasks in A are inactive over the interval $[t+1, u-1]$.*
>
> **Proof:** If the interval $[t+1, u-1]$ is empty, then the claim is vacuously true, so assume it is nonempty. Let V be any task in A. We first show that no subtask of V is scheduled in $[t, u-1]$.
>
> Note that because $V \in A$, no subtask of V is scheduled in slot t. Let V_i be the earliest subtask of V scheduled in $[t+1, u-1]$ and let v be the slot in which it is scheduled. Because there is hole in slot v, by Lemma 1, $d(V_i) = v$. By (6) and (8), this implies that $r(V_i) < v$ and hence, $e(V_i) < v$. Thus, because there are holes in all slots in $[t, v-1]$, it should have been scheduled earlier. Contradiction. Hence, no subtask of any task in A is scheduled in

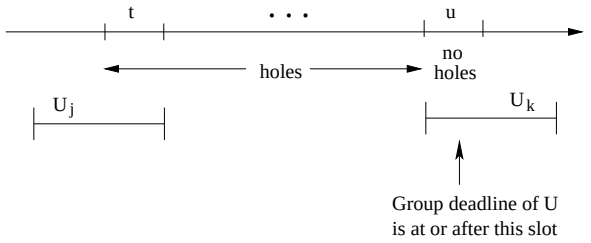

Figure 9: Lemma 10. U_j is the critical subtask of a task in A and U_k is the successor of U_j. There is a hole in each slot in $[t, u-1]$ and there is no hole in slot u. The earliest time at which U_k's PF-window starts is u, *i.e.*, $r(U_k) \geq u$.

$[t, u-1]$ (see Fig. 9). Moreover, because there are holes in all slots in $[t, u-1]$, the earliest slot after t at which a subtask of a task in A is eligible to be scheduled is u. By (29), this implies that all the tasks in A are inactive in $[t+1, u-1]$. □

Let U_j be any subtask in C, and let U_k be the successor of U_j. By Claim 3, $r(U_k) \geq u$. Furthermore, by (29)–(31) and (U), $d(U_j) = t < u \leq D(U_j)$. Hence, by (B5), $\mathit{flow}(U,t) + \mathit{flow}(U,u) \leq wt(U)$. Because this argument is applicable to all tasks in A, we have

$$\forall U \in A,\ \mathit{flow}(U,t) + \mathit{flow}(U,u) \leq wt(U). \tag{32}$$

We now show that LAG decreases over $[t+1, u-1]$.

Claim 4 *For all $v \in [t+1, u-1]$, $LAG(\tau, v+1) \leq LAG(\tau, v)$.*

Proof: If $[t+1, u-1]$ is empty, then the claim is vacuously true, so assume it is nonempty. Suppose for some $v \in [t+1, u-1]$, $LAG(\tau, v+1) > LAG(\tau, v)$. Then, by Lemma 8, there exists a task that is active at v but not scheduled at v. Let V be one such task and let V_k be the critical subtask of V at slot v. Then, by Def. 3,

$$e(V_k) \leq v \leq d(V_k). \tag{33}$$

Because no subtask of V is scheduled at v, V_k is scheduled before v. (It cannot be scheduled later by PD^2 because there is a hole at v, by (U).) By (U), there is a hole at $v-1$; moreover, because $v \in [t+1, u-1]$, we have $v-1 \in [t, u-2] \subseteq [0, t_d]$. Hence, by Claim 2, we have $d(V_k) \leq v-1$, which contradicts (33). Therefore, $LAG(\tau, v+1) \leq LAG(\tau, v)$ for all $v \in [t+1, u-1]$. □

Thus, the LAG continues to decrease in the interval $[t+1, u-1]$. We now show that $LAG(\tau, u+1) \leq 0$, which establishes our proof obligation.

For each $v \in [t, u]$, let H_v denote the number of holes in slot v. Then, $M - H_v$ tasks are scheduled in slot v.

By Claim 3, only tasks in $I \cup B$ are active at $v \in [t+1, u-1]$. Thus, by (14) and Claim 4, $\sum_{T \in \tau} \mathit{flow}(T,v) = \sum_{T \in I \cup B} \mathit{flow}(T,v) \leq \sum_{T \in I \cup B} S(T,v)$. Therefore,

$$\forall\ v \in [t+1, u-1],\ \sum_{T \in \tau} \mathit{flow}(T,v) \leq M - H_v. \tag{34}$$

Since $I \cup A \cup B = \tau$, $\sum_{T \in I \cup A \cup B} wt(T) \leq M$, by (9). Hence, by (32) and (P1), $\sum_{T \in A} (\mathit{flow}(T,t) + \mathit{flow}(T,u)) + \sum_{T \in I \cup B} \mathit{flow}(T,u) \leq M$. Thus,

$$\sum_{T \in A} \mathit{flow}(T,t) + \sum_{T \in \tau} \mathit{flow}(T,u) \leq M. \tag{35}$$

Because the tasks in B are the ones scheduled in slot t, the number of tasks in set B is $M - H_t$. Hence, because the weight of each task is at most one,

$$\sum_{T \in B} \mathit{flow}(T,t) \leq \sum_{T \in B} wt(T) \leq M - H_t. \tag{36}$$

We are now ready to show that $LAG(\tau, u+1) \leq 0$. Because $S(T,v) = M - H_v$, by (14), $LAG(\tau, u+1) - LAG(\tau, t) = R$, where $R = \sum_{v=t}^{u} \left(\sum_{T \in \tau} \mathit{flow}(T,v) \right) - \sum_{v=t}^{u} (M - H_v)$. Because, by (U), there are no holes in slot u, $H_u = 0$. Therefore,

$$R = \sum_{v=t}^{u} \left(\sum_{T \in \tau} \mathit{flow}(T,v) \right) - \sum_{v=t}^{u-1} (M - H_v) - M. \tag{37}$$

The right-hand side of (37) can be rewritten as follows.

$$\sum_{T \in \tau} (\mathit{flow}(T,t) + \mathit{flow}(T,u)) - (M - H_t) - M + \sum_{v=t+1}^{u-1} \left(\sum_{T \in \tau} \mathit{flow}(T,v) - (M - H_v) \right).$$

Rearranging terms, and using $\sum_{T \in I} \mathit{flow}(T,t) = 0$ (which follows by the definition of I), we get

$$\begin{aligned} & \sum_{T \in A} \mathit{flow}(T,t) + \sum_{T \in \tau} \mathit{flow}(T,u) - M \\ + & \sum_{T \in B} \mathit{flow}(T,t) - (M - H_t) \\ + & \sum_{v=t+1}^{u-1} \left(\sum_{T \in \tau} \mathit{flow}(T,v) - (M - H_v) \right). \end{aligned}$$

By (34)–(36), the above value is non-positive. Hence, by (37), $LAG(\tau, u+1) - LAG(\tau, t) \leq 0$. Because $LAG(\tau, t) \leq 0$, this implies that $LAG(\tau, u+1) \leq 0$.

By (U) and (31), $u \in [t+1, min(D(U_j), t_d)]$. Hence, $u+1 \in [t+2, t_d+1]$. Thus, there exists a $v \in [t+2, t_d+1]$ such that $LAG(\tau, v) \leq 0$. □