A ϕ-COMPETITIVE ALGORITHM FOR SCHEDULING PACKETS WITH DEADLINES*

PAVEL VESELÝ†, MAREK CHROBAK‡, ŁUKASZ JEŻ§, AND JIŘÍ SGALL†

Abstract. In the *online packet scheduling problem with deadlines* (PacketSchD, for short), the goal is to schedule transmissions of packets that arrive over time in a network switch and need to be sent across a link. Each packet has a deadline, representing its urgency, and a nonnegative weight, which represents its priority. Only one packet can be transmitted in any time slot, so if the system is overloaded, some packets will inevitably miss their deadlines and be dropped. In this scenario, the natural objective is to compute a transmission schedule that maximizes the total weight of packets that are successfully transmitted. The problem is inherently online, with the scheduling decisions made without the knowledge of future packet arrivals. The central problem concerning PacketSchD that has been a subject of intensive study since 2001 is to determine the optimal competitive ratio of online algorithms, namely the worst-case ratio between the optimum total weight of a schedule (computed by an offline algorithm) and the weight of a schedule computed by a (deterministic) online algorithm. We solve this open problem by presenting a ϕ-competitive online algorithm for PacketSchD (where $\phi \approx 1.618$ is the golden ratio), matching the previously established lower bound.

Key words. online algorithms, packet scheduling, buffer management

MSC code. 68W27

DOI. 10.1137/21M1469753

1. Introduction. In the *online packet scheduling problem with deadlines* (abbreviated PacketSchD), the goal is to schedule transmissions of packets that arrive over time in a network switch and need to be sent across a link. The switch stores incoming packets in a buffer that is assumed to have unlimited capacity. Besides its release time r_p, each arriving packet p has two other attributes: a deadline d_p, which is the last time slot when p can be transmitted, and a nonnegative weight w_p. The deadline of p represents its urgency, while its weight represents its importance or priority. (These priorities can be used to implement various levels of service in networks with Quality of Service (QoS) guarantees.) Only one packet can be transmitted in any time slot, so if the system is overloaded, some packets will inevitably miss their deadlines and be dropped. In this scenario, the natural objective is to compute a transmission schedule that maximizes the total weight of packets that are successfully transmitted. In the literature, this problem is also occasionally referred to as *bounded-delay buffer*

† Computer Science Institute of Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic (vesely@iuuk.mff.cuni.cz, sgall@iuuk.mff.cuni.cz).

‡ Department of Computer Science, University of California at Riverside, Riverside CA 92521 USA (marek@cs.ucr.edu).

§ Institute of Computer Science, University of Wrocław, Wrocław, Poland (lje@cs.uni.wroc.pl).

management, *QoS buffering*, or as a job scheduling problem for unit-length jobs with release times, deadlines, and weights, where the objective is to maximize the weighted throughput. In the offline setting, where the information about all packets is available in advance, it is easy to find an optimal schedule by representing the set of packets as a bipartite graph, with each packet p connected to all slots in its interval $[r_p, d_p]$ by an edge of weight w_p, and applying any algorithm for the maximum-weight bipartite matching to this graph.

In practice, however, scheduling of packets must be accomplished online, with the scheduling decisions made without the knowledge of future packet arrivals. The central problem concerning PacketSchD that has been a subject of intensive study since 2001 is to determine the optimal competitive ratio of online algorithms, namely the worst-case ratio between the optimum total weight of a schedule (computed by an offline algorithm) and the weight of a schedule computed by a (deterministic) online algorithm.

This paper provides the solution of this open problem by establishing an upper bound of ϕ on the competitive ratio for PacketSchD (where $\phi = (1 + \sqrt{5})/2 \approx 1.618$ is the golden ratio), matching the previously known lower bound [16, 3, 23, 10]. Our ϕ-competitive algorithm PlanM is presented in section 4. The basic idea underlying our algorithm is relatively simple. It is based on the concept of the *optimal plan*, which, at any given time t, is the maximum-weight subset of pending packets that can be feasibly scheduled in the future (if no other packets arrive); we describe it in section 3. When some packet p from the plan is chosen to be transmitted at time t, it will be replaced in the plan by some other packet ϱ. The algorithm chooses p to maximize an appropriate linear combination of w_p and w_ϱ. Furthermore, the algorithm makes additional changes in the plan, adjusting deadlines and weights of some packets. These changes are necessary to achieve the competitive ratio ϕ and can be viewed as a subtle way to remember necessary information about the past. While the algorithm itself is not complicated, its competitive analysis, given in section 5, is quite intricate. It relies on showing a bound on amortized profit at each step, using a potential function, which quantifies the advantage of the algorithm over the adversary in future steps, and on maintaining an invariant that allows us to control decreases of the potential function. The ideas leading to the development of algorithm PlanM and its analysis are also outlined in the *SIGACT News* "Online Algorithms" column [22].

Past work. The PacketSchD problem was first introduced independently by Hajek [16] and Kesselman et al. [18], who both gave a proof that the greedy algorithm (which always transmits the heaviest packet) is 2-competitive. Hajek's paper also contains a proof of a lower bound of $\phi \approx 1.618$ on the competitive ratio. The same lower bound was later discovered independently by Andelman et al. [3, 23] and also by Chin et al. [10] in a different, but equivalent, setting. Improving over the greedy algorithm, Chrobak et al. [11, 12] gave an online algorithm with competitive ratio 1.939. This ratio was subsequently improved to 1.854 by Li et al. [20], and to 1.828 by Englert and Westermann [14]. This value of 1.828, prior to our work, has remained the best upper bound on the competitive ratio of PacketSchD in the published literature.

Algorithms with ratio ϕ have been developed for several restricted variants of PacketSchD. Li et al. [19] (see also [17]) gave a ϕ-competitive algorithm for the case of *agreeable* deadlines, which consists of instances where the deadline ordering is the same as the ordering of release times, i.e., $r_p < r_q$ implies $d_p \le d_q$ for any packets p and q. Another well-studied case is that of *s-bounded instances*, where each packet can be scheduled in at most s slots, that is $d_p \le r_p + s - 1$ for each packet p. A

ϕ-competitive algorithm for 2-bounded instances was given by Kesselman et al. [18]. This bound was later extended to 3-bounded instances by Chin et al. [9] and to 4-bounded instances by Böhm et al. [5]. The work of Bienkowski et al. [6] provides an upper bound of ϕ (in a somewhat more general setting) for the case where packet weights increase with respect to deadlines. All these results are tight, as the proof of the lower bound of ϕ in [16, 3, 23, 10] is based on an adversary strategy that uses only 2-bounded instances with increasing weights, and 2-bounded instances have the agreeable-deadline property.

In *s-uniform instances*, each packet has *exactly s* consecutive slots where it can be scheduled. (Obviously, such instances also satisfy the agreeable deadlines property.) The lower bound of ϕ in [16, 3, 23, 10] does not apply to *s*-uniform instances; in fact, as shown by Chrobak et al. [12], for 2-uniform instances, competitive ratio ≈ 1.377 is optimal.

Randomized online algorithms for PacketSchD have been studied as well, although the gap between the upper and lower bounds for the competitive ratio in the randomized case remains quite large. The best upper bound is ≈ 1.582 [4, 9, 8, 21], and it applies even to the adaptive adversary model. For the adaptive adversary, the best lower bound is ≈ 1.33 [8], while for the oblivious adversary it is 1.25 [10].

Kesselman et al. [18] originally proposed the problem in the setting with integer bandwidth $m \geq 1$, which means that m packets are sent in each step. For an arbitrary m, they proved that the greedy algorithm is 2-competitive and that there is a ϕ-competitive algorithm for 2-bounded instances [18]. Later, Chin et al. [9] gave an algorithm with ratio that tends to $\frac{e}{e-1} \approx 1.582$ for $m \to \infty$. The best lower bound for any m, also due to Chin et al. [9], equals 1.25 and holds even for randomized algorithms against the oblivious adversary. Observe that any upper bound for bandwidth 1 implies the same upper bound for an arbitrary m, by simulating an online algorithm for bandwidth 1 on an instance where each step is subdivided into m smaller steps. Hence, our algorithm in section 4 is ϕ-competitive for any m, which improves the current state-of-the-art for any $m < 13$.

There is a variety of other packet scheduling problems related to PacketSchD. The semi-online setting with lookahead was proposed in [5]. A relaxed variant of PacketSchD in which only the ordering of deadlines is known but not their exact values, was studied in [7], where a lower bound higher than ϕ was shown. In the FIFO model (see, for example, [2, 18]), packets do not have deadlines, but the switch has a buffer that can only hold a bounded number of packets, and packets must be transmitted in the first-in-first-out order. More information about PacketSchD and related scheduling problems can be found in a survey paper by Goldwasser [15].

2. Preliminaries. This section provides a formal definition of the PacketSchD problem, along with some useful assumptions used throughout the paper.

The online PacketSchD problem. The instance of PacketSchD is specified by a set of *packets*, with each packet p represented by a triple (r_p, d_p, w_p), where integers r_p and $d_p \geq r_p$ denote the *release time* and *deadline* (or *expiration time*) of p, and $w_p \geq 0$ is the *weight* of p. (To avoid double indexing, we sometimes use notation $w(p)$ to denote w_p and $d(p)$ for d_p.) The time is discrete, with time units represented by consecutive integers that we refer to as *time slots* or *steps*. In a feasible transmission schedule, a subset of packets is transmitted. Only one packet can be transmitted in each time step, and each packet p can only be transmitted in a slot from the interval $[r_p, d_p]$. (Note that this interval includes slot d_p.) The objective is to compute a schedule whose total weight of transmitted packets (also called its *profit*) is maximized.

In the online variant of PacketSchD, which is the focus of our work, the algorithm needs to compute the solution incrementally over time. At any time step t, packets with release times equal to t are revealed and added to the set of pending packets (that is, those that are already released, but not yet expired or transmitted). Then the algorithm needs to choose one pending packet to transmit in slot t. As this decision is made without the knowledge of packets to be released in future time steps, such an online algorithm cannot, in general, be guaranteed to compute an optimal solution. The quality of the schedules it computes can be quantified using competitive analysis. We say that an online algorithm is *c-competitive* if, for each instance, the optimal profit (computed offline) is at most c times the profit of the schedule computed by the online algorithm.

Useful assumptions. The range of integers used as release times and deadlines is not important, as the whole instance can be shifted in time without affecting the competitive ratio. However, for the sake of concreteness we can assume that the whole instance fits in the interval $[0, T]$, where T is some large time horizon, and that the first step of the computation takes place at time 0. Throughout the paper, all time slots are tacitly assumed to be from this finite range. (For notational reasons, we will occasionally also use slot -1.)

Without loss of generality, we make two simplifying assumptions:

(A1) We assume that at each step t and for each $\tau \in [t, T]$, there is a pending packet with deadline τ (or more such packets, if needed). This can be achieved by releasing, at time t, virtual 0-weight packets with deadline τ, for each $\tau \in [t, T]$.

(A2) We also assume that all packets have different weights. Any instance can be transformed into an instance with distinct weights through infinitesimal perturbation of the weights, without affecting the competitive ratio. Thus the virtual packets from the previous assumption have, in fact, infinitesimal positive weights, although in the calculations we treat these values as being equal 0. The only purpose of this assumption is to facilitate consistent tie-breaking, in particular uniqueness of plans (to be defined shortly).

A careful reader may notice that assumption (A1) as written reveals the value of T to the online algorithm. This does not affect our analysis because, for any instance, if the maximum packet deadline in this instance is $d_{\max}$, then the computation of our algorithm is independent of the choice of the time horizon $T \geq d_{\max}$, provided that the ties between weights of virtual 0-weight packets are broken in favor of earlier-deadline packets, in the sense that those with larger deadlines are considered "lighter".[1]

The golden ratio. The competitive ratio of our algorithm is $\phi = \frac{1}{2}(\sqrt{5}+1) \approx 1.618$, the famous number known as the golden ratio. Its most important property is that it satisfies the equation $\phi^2 = \phi + 1$. This identity will be frequently used in calculations, usually in the form $\frac{1}{\phi^2} + \frac{1}{\phi} = 1$. Another useful property is that $\sum_{i=0}^{\infty} \phi^{-2i} = \phi$, which follows easily from the formula for summing a geometric sequence and the earlier-mentioned properties of ϕ.

---

[1] An alternative and equivalent approach would be to change the time horizon dynamically, so that it is always equal to the maximum deadline of already released packets. This, however, creates some minor but distracting technical difficulties in the analysis. As yet another option, one can show that, in fact, modifying the definition of PacketSchD by having the time horizon revealed when the computation starts does not affect the optimal competitive ratio. We don't need such a strong statement, however, for our analysis.

3. Plans and their properties. Consider an execution of some online algorithm. At any time t, the algorithm will have a set of pending packets. We now discuss properties of these pending packets and introduce the concept of plans that will play a crucial role in our algorithm.

The set of packets pending at a time t has a natural ordering, called the *canonical ordering* and denoted $\prec$, which orders packets in nondecreasing order of deadlines, breaking ties in favor of heavier packets. (By assumption (A2) the weights are distinct.) Formally, for two pending packets p and q, define $p \prec q$ iff $d_p < d_q$ or $d_p = d_q$ and $w_p > w_q$. The *earliest-deadline packet* in some subset X of pending packets is the packet that is first in the canonical ordering of X. Similarly, the *latest-deadline packet* in X is the last packet in the canonical ordering of X.

Plans. Consider a set X of packets pending at a time step t. X is called a *plan* if the packets in X can be feasibly scheduled in future time slots $t, t+1, \ldots$, where "feasibly" means that all packets in X meet their deadlines. We will typically use symbols $P, Q, \ldots$ to denote plans. We emphasize that in a plan we do not assign packets to time slots; that is, a plan is *not* a schedule. A plan has at least one schedule, but in general it may have many. (In the literature, such scheduled plans are sometimes called *provisional schedules*.) Using a standard exchange argument, if P is a plan then any schedule of P can be converted into its *canonical schedule*, in which the packets from P are assigned to the slots $t, t+1, \ldots$ in the canonical order.

For any two time slots $\sigma \geq \tau \geq t$, by $X[\tau, \sigma] = \{j \in X : d_j \in [\tau, \sigma]\}$ we denote the subset of X consisting of packets whose deadlines are in $[\tau, \sigma]$. In particular, $X[t, \tau]$ contains packets with deadlines at most τ. In a similar way, we define $X[\tau, \sigma)$, $X(\tau, \sigma]$, and $X(\tau, \sigma)$. We also define

$$\mathsf{pslack}(X, \tau) = (\tau - t + 1) - |X[t, \tau]|.$$

Note that $\tau - t + 1$ is the number of slots in interval $[t, \tau]$. For convenience, we also allow $\tau = t - 1$, in which case the above formula gives us that $\mathsf{pslack}(X, t-1) = 0$. We stress that the formula for $\mathsf{pslack}(X, \tau)$ also depends on the value of t. This t will always be the "current step" under consideration and it will be uniquely determined from context, so we do not include it as a parameter of this function, to reduce clutter.

The values of $\mathsf{pslack}(X, \tau)$ are useful for determining feasibility of X.

Observation 3.1. Let X be a subset of packets pending at some step t. X is a plan if and only if $\mathsf{pslack}(X, \tau) \geq 0$ for each $\tau \geq t$.

This observation is quite straightforward: If X is a plan, then, in any (feasible) schedule of X, for each $\tau \geq t$ all packets in $X[t, \tau]$ are scheduled in different slots of the interval $[t, \tau]$; thus $|X[t, \tau]| \leq \tau - t + 1$. This shows the necessity of the condition in the observation. To show sufficiency, assume that $\mathsf{pslack}(X, \tau) \geq 0$ for each $\tau \geq t$. This implies, by simple induction, that assigning the packets in X to slots $t, t+1, \ldots$ in the canonical order will produce a schedule (the canonical schedule of X, in fact) in which all packets will meet their deadlines.

Throughout the paper, we will tacitly assume that any plan P we consider is *full*, in the sense that it contains enough packets to fill all slots between the current time t and the time horizon T; that is, $|P[t, T]| = T - t + 1$. This assumption can be made without loss of generality because there are always sufficiently many infinitesimal-weight packets pending, by assumption (A1). With this assumption, the concepts of tight slots, segments, etc., to be introduced below will be always well defined.
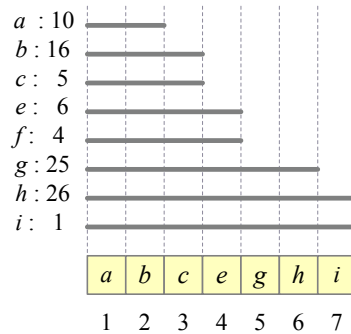
FIG. 3.1. *An example of an instance with optimal plan $P = \{a, b, c, e, g, h, i\}$ and its canonical schedule in time slots $1, 2, \ldots, 7$. Each packet is represented by one row that shows this packet's identifier and weight (separated by colon), followed by a line segment stretching from the current time slot t (assumed to be 1 here) to this packet's deadline. Note that packet f is not in P, even though it is heavier than i. The segments of P are $(0, 3]$, $(3, 4]$, $(4, 7]$. (Alternatively, in terms of packets from P, these segments are $\{a, b, c\}$, $\{e\}$, $\{g, h, i\}$.) The values of $\mathsf{minwt}(P, \tau)$ are 5 for $\tau \in \{1, 2, 3, 4\}$ and 1 for $\tau \in \{5, 6, 7\}$.*

Let P be a plan for the set of packets pending at some time t. We now discuss the structure of P. (See Figure 3.1 for an illustration.) Slot $\tau \geq t$ is called *tight* in P if $\mathsf{pslack}(P, \tau) = 0$. In particular, according to our conventions above, slots $t - 1$ and T are tight. If the tight slots of P are $t_0 = t - 1 < t_1 < t_2 < \cdots < t_c = T$, then for each $i = 1, 2, \ldots, c$ the time interval $(t_{i-1}, t_i] = \{t_{i-1} + 1, t_{i-1} + 2, \ldots, t_i\}$ is called a *segment* of P. In other words, the tight slots divide the time range into segments, each starting right after a tight slot and ending at (and including) the next tight slot. The significance of a segment $(t_{i-1}, t_i]$ is that in any schedule of P all packets in $P(t_{i-1}, t_i]$ must be scheduled in this segment. Thus, slightly abusing terminology, we occasionally think of each segment as a set of packets to be scheduled in this segment, namely the set $P(t_{i-1}, t_i]$. Within a segment, packets from P can be permuted, although only in some restricted ways. In particular, the first slot of a segment may contain any packet from that segment (see Observation A.1). Let $\alpha = t_1$ be the first tight slot of plan P (we regard tight slot $t_0 = t - 1$ as the "0th tight slot"). The first tight slot α will play a special role in our algorithm, together with the segment $[t, \alpha] = (t_0, t_1]$ of P, called the *initial segment*. Whenever we consider several plans at a time t and some ambiguity arises, we will write $\alpha = \alpha_P$, to indicate that α is the first tight slot for this specific plan P.

For a plan P and a slot $\tau \geq t$, let $\mathsf{nextts}(P, \tau)$ be the earliest tight slot $\tau' \geq \tau$ and let $\mathsf{prevts}(P, \tau)$ be the latest tight slot $\tau' < \tau$. Both notations are well defined: that $\mathsf{nextts}(P, \tau)$ is well defined follows from slot T being tight, and $\mathsf{prevts}(P, \tau)$ is well defined because $t - 1$ is a tight slot, according to our convention. (Both functions also depend on t, but t is implied, since t is the time for which P is defined. This convention will apply to other notations involving plans.)

The notion that will be crucial in the design of our ϕ-competitive algorithm is the minimum weight of a packet that can appear in a schedule of the plan in some slot between the current time t and a given slot τ. Naturally, the packets that are candidates for this minimum include all packets in segments ending before τ, but also we need to include all packets in the segment of τ, even those with deadlines larger than τ. (This is because, as explained earlier, each packet in a segment can be

---

Algorithm 3.1 Algorithm **ComputePlan**$(\mathcal{U}, t)$

Input: $\mathcal{U}$ is a set of packets pending at time t
1: $X \leftarrow \emptyset$
2: **for** each packet $j \in \mathcal{U}$ in order of decreasing weights **do**
3:      **if** $\mathsf{pslack}(X \cup \{j\}, \tau) \geq 0$ for all $\tau \geq t$ **then**
4:          $X \leftarrow X \cup \{j\}$
5: $P \leftarrow X$ $\triangleright$ P is the optimal plan for $\mathcal{U}$

---

scheduled at the beginning of that segment.) Formally, for a plan P at time t and a slot $\tau \geq t$, define

$$\mathsf{minwt}(P, \tau) = \min \{\, w_j : j \in P[t, \mathsf{nextts}(P, \tau)] \,\}.$$

We will occasionally omit P in this notation if it is understood from context. By definition, if we fix t and P and think of $\mathsf{minwt}(P, \tau)$ as a function of τ, then this function is monotonically nonincreasing over the whole range of $\tau = t, t+1, \ldots, T$, and is constant in each segment (that is, all slots τ in any given segment have the same value of $\mathsf{minwt}(P, \tau)$).

Optimal plans. Given a set of packets $\mathcal{U}$ pending at a given time t, their plan P is called *optimal* if it has the maximum weight among all plans of these packets. Since the collection of such plans forms a matroid (which is easy to verify using Observation 3.1; see also, e.g., [18]), the optimal plan at step t can be computed by the following greedy algorithm:

Assumption (A2) about different weights implies that the optimal plan P computed above is unique. See Figure 3.1 for an example of an optimal plan for a given set of pending packets.

During a computation, the set of pending packets can change. There are two types of events that can cause this change: an arrival of a new packet and a transmission of a packet, which also involves incrementing the current time and dropping all newly expired packets. This will also cause the optimal plan to change. The matroid property implies that at most one packet in the optimal plan changes after each event (not counting the transmitted packet in a transmission event). These changes are fairly intuitive and we outline them briefly below; for a formal description of these changes, correctness proofs, and figures with examples, see Appendix A.

In the discussion below, we assume that t is the current time step and P is the current optimal plan, right before an arrival or transmission event. By Q we will denote the optimal plan right after the event. For a slot $\tau \geq t$, whenever we refer to the change of $\mathsf{pslack}(\tau)$ (that it increases, decreases, or remains the same), without specifying the plan, we mean the change from $\mathsf{pslack}(P, \tau)$ to $\mathsf{pslack}(Q, \tau)$. We use the same convention for function $\mathsf{minwt}(\tau)$.

Packet arrivals. We first consider the event of a new packet s arriving at time t. As s is added to the set of pending packets, the optimal plan needs to be updated accordingly. Define $f \in P$ to be the packet with $w_f = \mathsf{minwt}(P, d_s)$, which is the lightest packet in P with $d_f \leq \mathsf{nextts}(P, d_s)$. If $w_s < w_f$, then s is not added to the optimal plan, which thus stays the same. On the other hand, if $w_s > w_f$, then s is added to the optimal plan and f is forced out, i.e., the new optimal plan is $Q = P \cup \{s\} \setminus \{f\}$. In the latter case, it is interesting to see how the values of $\mathsf{pslack}(\tau)$ and the segments change:

- If $d_s \geq d_f$, then $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) + 1$ for $\tau \in [d_f, d_s)$. Therefore, all tight slots in $[d_f, d_s)$ in P are no longer tight in Q and the segments containing d_f and d_s and all segments in between get merged into one segment of Q.
- If $d_s < d_f$, then d_f and d_s must be in the same segment of P (because $d_f \leq \mathsf{nextts}(P, d_s)$) and $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) - 1$ for $\tau \in [d_s, d_f)$. Thus, there may be new tight slots in $[d_s, d_f)$, resulting in new segments.

In both cases, the values of $\mathsf{pslack}(\tau)$ remain the same for other slots $\tau \geq t$; thus other tight slots and segments do not change. Moreover, $\mathsf{minwt}(Q, \tau) \geq \mathsf{minwt}(P, \tau)$ holds for any slot $\tau \geq t$; this property will play a significant role in our algorithm.

Transmitting a packet. Next, we discuss the changes of the optimal plan resulting from a transmission event. Throughout the paper, only packets in the optimal plan will be considered for transmission. The scenario is this: We consider the set $\mathcal{U}$ of all packets pending at time t, and the optimal plan P for $\mathcal{U}$. We choose some packet $p \in P$, transmit it at time t, and increment the current time to $t + 1$. The new set $\mathcal{U}'$ of pending packets (now at time $t + 1$) is obtained from $\mathcal{U}$ by removing p and all packets with deadline t. As before, we will use Q to denote the new optimal plan after this change, namely the optimal plan for $\mathcal{U}'$ with respect to time $t + 1$.

If p is from the initial segment $[t, \alpha]$ of P, then $Q = P \setminus \{p\}$. In this case $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) - 1$ for $\tau \in [t + 1, d_p)$ and $\mathsf{pslack}(\tau)$ remains unchanged for $t \geq d_p$. This implies that new tight slots may appear before d_p, i.e., the initial segment may get divided into more segments. Furthermore, $\mathsf{minwt}(\tau)$ does not decrease for any $\tau \geq t + 1$.

The more interesting case is when p is from a later segment. Let ℓ be the lightest packet in $P[t, \alpha]$, i.e., in the initial segment, and let ϱ be the heaviest pending packet not in P that satisfies $d_\varrho > \mathsf{prevts}(P, d_p)$ (such a packet ϱ exists by Assumption (A1)). Using the matroid property of the feasible sets of packets at time $t + 1$ and the structure of the plan, in Appendix A we prove that $Q = P \setminus \{p, \ell\} \cup \{\varrho\}$. Furthermore, in this case we have:

- $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) - 1$ for $\tau \in [t + 1, d_\ell)$. There may be new tight slots in the interval $[t + 1, d_\ell)$, resulting in new segments.
- If $d_\varrho \geq d_p$, then $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) + 1$ for $\tau \in [d_p, d_\varrho)$. Here, all segments that overlap $[d_p, d_\varrho]$ are merged into one segment of Q.
- If $d_\varrho < d_p$, then $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) - 1$ for $\tau \in [d_\varrho, d_p)$. (In this case, d_p and d_ϱ must be in the same segment of P, because $d_\varrho > \mathsf{prevts}(P, d_p)$.) As a result of decreasing some $\mathsf{pslack}(\tau)$ values, new tight slots may appear in $[d_\varrho, d_p)$, creating new segments.

For slots $\tau \geq t + 1$ not covered by the cases above, the value of $\mathsf{pslack}(\tau)$ does not change. Unlike for packet arrivals, after a packet transmission event some values of $\mathsf{minwt}(\tau)$ may decrease, either due to ϱ being included in Q or as a side-effect of segments being merged.

Substitute packets. The aforementioned updates of the plan motivate the following definition: Let $\mathcal{U}$ be the set of all packets pending at time t and $P \subseteq \mathcal{U}$ be the optimal plan at time t. For each $j \in P$ we define the *substitute packet of j*, denoted $\mathsf{sub}(P, j)$, as follows. If $j \in P[t, \alpha]$, then $\mathsf{sub}(P, j) = \ell$, where ℓ is the lightest packet in $P[t, \alpha]$. If $j \notin P[t, \alpha]$, then $\mathsf{sub}(P, j)$ is the heaviest pending packet $\varrho \in \mathcal{U} \setminus P$ that satisfies $d_\varrho > \mathsf{prevts}(P, d_j)$, which exists by assumption (A1).

By definition, all packets in a segment of P have the same substitute packet. Also, for any $j \in P$ it holds that $w_j \geq w(\mathsf{sub}(P, j))$. This is because for $j \in P[t, \alpha]$ we have $\mathsf{sub}(P, j) = \ell$ and $w_j \geq w_\ell$ (the equality will hold only when $\ell = j$), while for

$j \in P(\alpha, T]$ we have $d(\mathsf{sub}(P, j)) > \mathsf{prevts}(P, d_j)$; thus in this case the set $P - \{j\} \cup \{\mathsf{sub}(P, j)\}$ is feasible, and the optimality of P implies that $w_j > w(\mathsf{sub}(P, j))$.

4. Online algorithm. This section presents our online algorithm PlanM for PacketSchD, starting with some intuitions and additional notation, and followed by two examples where its competitive ratio is exactly ϕ.

Intuitions. Similar to other online profit maximization problems, the main challenge in achieving a small competitive ratio for PacketSchD is in finding the right balance between the immediate profit and future profits. Let P be the optimal plan at a step t. Consider the greedy algorithm for PacketSchD, which at time t transmits the heaviest pending packet h, which is always in P. We focus on the case when h is not in the initial segment. (The case when h is in the initial segment is different, but similar intuitions apply to it as well.) In the next step, after transmitting h but before new packets are released, h will be replaced in the optimal plan by its substitute packet $\varrho_h = \mathsf{sub}(P, h)$. This packet could be very light, possibly $w(\varrho_h) \approx 0$. Suppose that there is another packet g in P with $d_g < d_h$ and $w_g \approx w_h$ whose substitute packet $\varrho_g = \mathsf{sub}(P, g)$ is quite heavy, say $w(\varrho_g) \approx w_g$. Thus, instead of h we can transmit g at time t, achieving roughly the same immediate profit as from transmitting h, but with essentially no decrease in the weight of the optimal plan (which serves as a rough estimate of future profits). This example indicates that a reasonable strategy would be to choose a packet p based both on its weight and the weight of its substitute packet. Following this intuition, our algorithm chooses p that maximizes $w_p + \phi \cdot w(\mathsf{sub}(P, p))$, breaking ties arbitrarily. The choice of the coefficients in the objective function follows the intuition from analyzing the 2-bounded case; see the discussion of examples in Figure 4.2.

As it turns out, the above strategy for choosing p does not, by itself, guarantee ϕ-competitiveness. The analysis of special cases and an example where this simple approach fails lead to the second idea behind our algorithm.[2] The difficulty is related to how the values of $\mathsf{minwt}(\tau)$, for a fixed τ, vary while the current time t increases. We were able to show ϕ-competitiveness of the above strategy for certain instances where $\mathsf{minwt}(\tau)$ monotonely increases as t grows from 0 to τ. We call this property *slot-monotonicity*. To extend slot-monotonicity to instances where it does not hold, the idea is then to simply *force* it to hold by decreasing deadlines and increasing weights of some packets in the new optimal plan. (To avoid unfairly benefiting the algorithm from these increased weights, we will need to account for them appropriately in the analysis.) From this point on, the algorithm proceeds using these new weights and deadlines when computing the optimal plan and choosing a packet for transmission.

Notation. To avoid ambiguity, we will index various quantities used by the algorithm with the superscript t that represents the current time. This includes weights and deadlines of some packets, since these might change over time.

- We use notations w_j^t and d_j^t for the weight and the deadline of a packet j in step t, before the transmission of this step is implemented. (Our algorithm only changes weights and deadlines when transmitting a packet, so they are not affected by packet arrivals.) To avoid double subscripts, we occasionally write $w^t(j)$ and $d^t(j)$ instead of w_j^t and d_j^t. We will sometimes use notation w_j^0 for the original weight of packet j, when notation w_j may lead to some

---

[2]This example is described in Appendix C of the preprint version of our paper, which can be found at https://arxiv.org/abs/1807.07177. This appendix also contains examples demonstrating that certain simpler variants of algorithm PlanM are not ϕ-competitive.

Algorithm 4.1 Algorithm PlanM(t)

1: transmit packet $p \in P^t$ that maximizes $w_p^t + \phi \cdot w^t(\mathsf{sub}^t(p))$
2: **if** $d_p^t > \alpha$ **then**   ▷ "leap step"
3:     $\varrho \leftarrow \mathsf{sub}^t(p)$
4:     $w_\varrho^{t+1} \leftarrow \mathsf{minwt}^t(d_\varrho^t)$ ▷ increase w_ϱ
5:     $\gamma \leftarrow \mathsf{nextts}^t(d_\varrho^t)$ and $\tau_0 \leftarrow \mathsf{nextts}^t(d_p^t)$
6:     $i \leftarrow 0$ and $h_0 \leftarrow p$
7:     **while** $\tau_i < \gamma$ **do**
8:         $i \leftarrow i + 1$
9:         $h_i \leftarrow$ heaviest packet in $P^t(\tau_{i-1}, \gamma]$
10:         $\tau_i \leftarrow \mathsf{nextts}^t(d_{h_i}^t)$
11:         $d_{h_i}^{t+1} \leftarrow \tau_{i-1}$ and $w_{h_i}^{t+1} \leftarrow \max(w_{h_i}^t, \mathsf{minwt}^t(\tau_{i-1}))$ ▷ adjusting packet h_i
12:     $k \leftarrow i$ ▷ final value of i

---

ambiguity. We will also omit t in these notations whenever t is unambiguously implied from context.

- P^t is the optimal plan at time t after all packets j with $r_j = t$ arrive and before a packet is transmitted.
- We write $\mathsf{sub}^t(p)$ to denote $\mathsf{sub}(P^t, p)$ and we adopt similar conventions for $\mathsf{minwt}^t(\tau)$, $\mathsf{nextts}^t(\tau)$, and $\mathsf{prevts}^t(\tau)$.
- $\alpha = \mathsf{nextts}^t(t)$ is the first tight slot of P^t, so $[t, \alpha]$ is the initial segment.
- ℓ is the lightest packet in $P^t[t, \alpha]$.
- ϱ, h_i, γ, τ_i, and k will refer to the packets and values chosen in Algorithm PlanM(t).

The pseudo-code of our algorithm, called PlanM(t), is given in Algorithm 4.1. For a pending packet j, if w_j^{t+1} (resp., d_j^{t+1}) is not explicitly set in the algorithm, then its value remains the same by default, that is $w_j^{t+1} \leftarrow w_j^t$ (resp., $d_j^{t+1} \leftarrow d_j^t$).

Let p be the packet sent by PlanM in step t. If p is in the initial segment $[t, \alpha]$ of P^t, the step is called an *ordinary step*. Otherwise (if $d_p^t > \alpha$), the step is called a *leap step*, and then $\varrho = \mathsf{sub}^t(p)$ is the heaviest pending packet $\varrho \notin P^t$ with $d_\varrho^t > \mathsf{prevts}^t(d_p^t)$. We will further consider two types of leap steps. If d_p and d_ϱ are in the same segment, then this step is called a *simple leap step*; in that case $\gamma = \tau_0$, the while loop is not executed, and $k = 0$. If d_ϱ is in a later segment than d_p, then this step is called an *iterated leap step*; in that case $\gamma > \tau_0$, the while loop is executed at least once, and $k > 0$.

As all packets in the segment of P^t that contains p have the same substitute packet $\mathsf{sub}^t(p)$, p must be the heaviest packet in its segment. Furthermore, p is not too light compared to the heaviest pending packet h; specifically, we have that $w_p \geq w_h / \phi^2$. Indeed, as mentioned earlier, we have $w_p \geq w(\mathsf{sub}^t(p))$. It follows that $\phi^2 w_p = w_p + \phi w_p \geq w_p + \phi \cdot w(\mathsf{sub}^t(p)) \geq w_h + \phi \cdot w(\mathsf{sub}^t(h)) \geq w_h$, where the second inequality follows by the choice of p in line 1 of Algorithm PlanM(t).

Slot-monotonicity. Our goal is to maintain *the slot-monotonicity property*, i.e., to ensure that for any fixed slot τ the value of $\mathsf{minwt}^t(\tau)$ does not decrease as the current time t progresses from 0 to τ. For this reason, we need to increase the weight of the substitute packet ϱ in each leap step (as $w_\varrho^t < \mathsf{minwt}^t(d_\varrho^t)$), which is done in line 4. (To maintain Assumption (A2), we add an infinitesimal to the new weight of ϱ.) For the same reason, in the iterated leap step, we also need to adjust the deadlines and weights of the packets h_i, which is done in line 11. The deadlines of
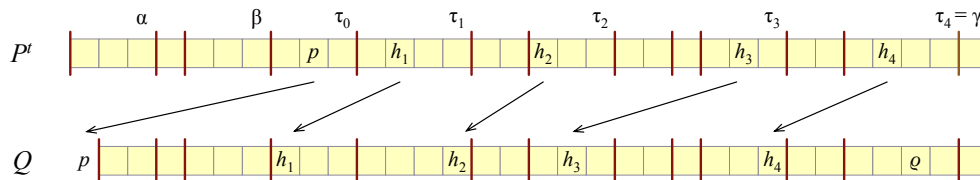
FIG. 4.1. *An illustration of the shift of packets $h_1, \ldots, h_k$ (for $k = 4$) in lines 6–11 in an iterated leap step. In this figure, $\beta = \mathsf{prevts}^t(d_p^t)$ and Q is the plan right after p is transmitted and the current time is incremented to $t+1$ (but before packets released at time $t+1$ are taken into account). Both plans are represented by their canonical schedules. Vertical dark-red lines separate the segments of the plans.*

h_i's are decreased to make sure that the segments between $\beta = \mathsf{prevts}^t(d_p^t)$ and γ do not merge (as merging could cause a decrease of some values of $\mathsf{minwt}^t(\tau)$). These deadline changes can be thought of as a sequence of substitutions, where h_1 replaces p in the segment of P ending at τ_0, h_2 replaces h_1, etc., and finally, ϱ replaces h_k in the segment ending at γ. We sometimes refer to this process as a "shift" of the h_i's. See Figure 4.1 for an illustration. Then, if the weight of some h_i is too low for its new segment, it is increased to match the earlier minimum of that segment, that is, $\mathsf{minwt}^t(\tau_{i-1})$. (Again, to maintain Assumption (A2), we add an infinitesimal to the new weight of h_i.)

In ordinary steps, the algorithm does not make any weight or deadline changes. Thus, in such steps the optimal plan changes as described earlier in section 3 and the slot monotonicity property is preserved. (See Appendix A for formal proofs.)

In leap steps, the algorithm modifies weights and deadlines of some packets, and thus the discussion from section 3 does not apply directly. The changes in the optimal plan after a leap step are elaborated in detail in Lemma B.1 in Appendix B. We briefly summarize them here. Recall that in a leap step the packet p is in some noninitial segment. Let $P = P^t$ and let Q be the optimal plan after p is transmitted, the time is incremented to $t+1$, and weights and deadlines are changed (according to lines 3–11 in the algorithm). Let also $\overline{Q}$ be the intermediate optimal plan after p is transmitted and the time is incremented, but before the algorithm adjusts weights and deadlines. As discussed in section 3, this plan is $\overline{Q} = P \setminus \{p, \ell\} \cup \{\varrho\}$, where $\varrho = \mathsf{sub}^t(p)$.

In a simple leap step, only the weight of ϱ is modified. Increasing the weight of a packet in the optimal plan does not affect its optimality and thus $Q = \overline{Q}$. Furthermore, no segments are merged, i.e., any tight slot of P is tight in Q as well.

In an iterated leap step, by the choice of p, the definition of h_i's in line 9, and the while loop condition in line 7, we have that $w_p^t = w_{h_0}^t > w_{h_1}^t > w_{h_2}^t > \cdots > w_{h_k}^t > w_\varrho^t$ and that h_k is in the segment of P ending at γ, that is $\mathsf{prevts}^t(d_\varrho^t) < d_{h_k}^t \le \gamma$. As in the simple leap step, increasing the weight of a packet does not affect the optimality of a plan. Moreover, a careful analysis of the changes of $\mathsf{pslack}()$ values yields that decreasing the deadlines of $h_1, h_2, \ldots, h_k$ (in line 11) does not change the optimal plan, so we can conclude that $Q = \overline{Q}$ holds in an iterated leap step as well. The decrease of the deadlines of h_i's also ensures that no segments are merged. (See Appendix B for complete proofs.)

The property that no segments are merged, together with the increase of the weights, allows us to prove that $\mathsf{minwt}^t(\tau)$ does not decrease for any τ even in a leap step. This slot monotonicity property is summarized in the lemma below, whose proof follows directly from Lemma A.2(c), Lemma A.4(c), and Lemma B.2.
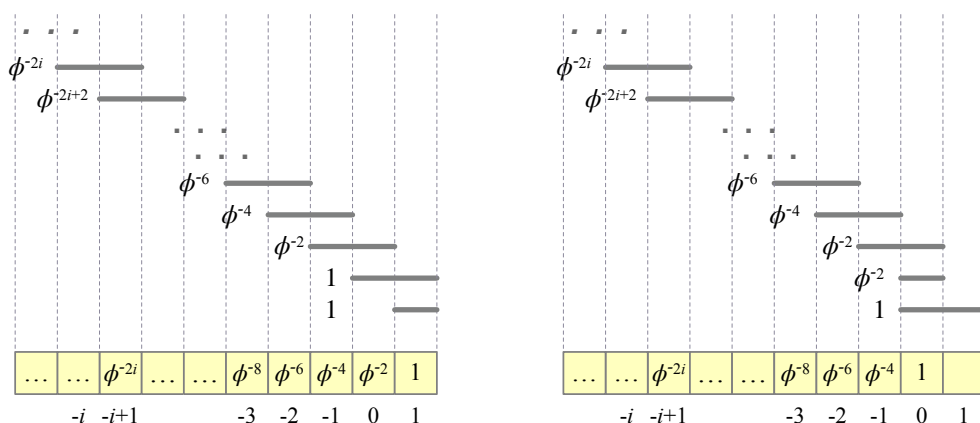
FIG. 4.2. *Two examples where the competitive ratio of Algorithm* PlanM *is* ϕ. *Packets are identified by their weight. We show the schedules of* PlanM *at the bottom and time slot indices below the schedules.*

LEMMA 4.1. *Let P be the current optimal plan in step t just before an event of either arrival of a new packet or transmitting a packet (and incrementing the current time), and let Q be the plan after the event. Then* $\mathsf{minwt}(Q,\tau) \geq \mathsf{minwt}(P,\tau)$ *for any $\tau > t$, and also for $\tau = t$ in case of a packet arrival. Hence, in the computation of Algorithm* PlanM, *for any fixed τ, the function $\mathsf{minwt}^t(\tau)$ is nondecreasing in $t \in [0,\tau]$.*

Tight examples. To illustrate the computation of Algorithm PlanM, we now give two examples of 2-bounded instances where the competitive ratio is exactly ϕ. They also give some intuition for the choice of coefficients in line 1 of Algorithm PlanM, as the chosen coefficients exactly balance the competitive ratio for these two examples. We note that, interestingly, there is another combination of the coefficients that gives ratio ϕ for the 2-bounded case, namely $\phi \cdot w_p + w(\mathsf{sub}(P,p))$; we do not know if these coefficients can yield an optimal algorithm for the general case.

Before we examine these examples, consider the instance where we have two packets pending at some step t, one tight packet q with $d_q = t$ and $w_q = c$, and one packet r with $d_r = t+1$ and $w_r = \phi^2 c$. The optimal plan is $P^t = \{q,r\}$ and $\alpha = t$. In this case we have a tie in line 1 of Algorithm PlanM: For $p = q$ the substitute packet is $\ell = q$ and the objective value is $w_q + \phi\, w_q = \phi^2\, c$, and for $p = r$ the substitute packet has weight 0 and the objective value is $w_r + \phi\, 0 = \phi^2\, c$ as well. In the examples below, we will assume that we can break this tie either way. This can be accomplished by infinitesimally perturbing the weights, without affecting the competitive ratio.

The two examples are shown in Figure 4.2. For simplicity, in these examples packets are identified by their weight and we allow negative-valued time steps. In both examples, the instance involves a sequence of packets, where at time $-i$, for $i \geq 0$, a packet of weight ϕ^{-2i} and deadline $-i+1$ is released. For simplicity, we will think of this sequence as starting at $-\infty$.

In the instance on the left, we have one additional tight packet of weight 1 released at time 1. In this instance we break the ties so that the algorithm schedules each packet ϕ^{-2i} at time $-i+1$ (thus, at each step $-i \leq 0$, we transmit the tight pending packet). As a result, the algorithm will be able to transmit only one packet of weight 1, and its profit will be $\sum_{i=0}^{\infty} \phi^{-2i} = 1/(1 - \phi^{-2}) = \phi$. The optimum solution schedules all packets, so its profit is $1 + \sum_{i=0}^{\infty} \phi^{-2i} = \phi^2$.
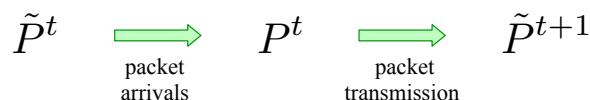
In the instance on the right, we have one additional tight packet of weight ϕ^{-2} released at time 0. Here, we also break the ties in favor of the tight pending packet until time -1, but at time 0 the algorithm transmits the packet of weight 1. The algorithm's profit will be $\sum_{i=0}^{\infty} \phi^{-2i} - \phi^{-2} = \phi - \phi^{-2} = 1 + \phi^{-1} - \phi^{-2} = 2/\phi$. The optimum solution schedules all packets, so its profit is $\sum_{i=0}^{\infty} \phi^{-2i} + \phi^{-2} = 2$.

Comparison to previous algorithms. Our algorithm shares some broad features with known algorithms in the literature. In fact, any online algorithm (with competitive ratio below 2) needs to capture the tradeoff between weight and urgency when transmitting a packet, so some similarities between these algorithms are inevitable. As we found in our earlier attempts, however, the exact mechanism of formalizing this tradeoff is critical, and minor tweaks can dramatically affect the competitive ratio.

Some prior algorithms used the notion of the *optimal provisional schedule*, which coincides with our concept of canonically ordered optimal plan; recall that in our paper a plan is defined as a feasible *set* of pending packets rather than their particular schedule. For example, the ϕ-competitive algorithm MG for instances with agreeable deadlines by Li et al. [19] (see also [17]) transmits packets from the optimal plan only, either the heaviest packet h or the earliest-deadline packet e. (Strictly speaking, this is true for the simplified variant of MG [17], whereas the original version of MG transmitted either e or another sufficiently heavy packet from the optimal plan [19].) The same authors [20] later designed a modified algorithm called DP that achieves competitive ratio $3/\phi \approx 1.854$ for arbitrary instances.

Our approach is more similar to that of Englert and Westermann [14], who designed a 1.893-competitive memoryless algorithm and an improved 1.828-competitive variant with memory. Both their algorithms are based on the notion of *suppressed packet* $\mathsf{supp}(P, p)$, for a packet p in the plan P, which, in our terminology, is the same as the substitute packet $\mathsf{sub}(P, p)$ *if* p is not in the initial segment. However, the two concepts differ for packets p in the initial segment. The memoryless algorithm in [14] identifies a packet m of maximum "benefit," which is measured by an appropriate linear combination of w_m and $w(\mathsf{supp}(P, m))$, and sends either m or e (the earliest-deadline packet in the optimal plan), based on the relation between w_e and the benefit of m. The algorithm with memory in [14] extends this approach by comparing m's benefit to e's "boosted weight" $\max(w_e, \delta(t))$, where t is the current step and $\delta(\tau)$ is the maximum value of $\mathsf{minwt}(P^{t'}, \tau)$ over $t' < t$. We remark that considering this value of $\delta(\tau)$ takes into account the previous maximal value of $\mathsf{minwt}(\tau)$; however, it does not prevent actual decreases of $\mathsf{minwt}(\tau)$.

Our algorithm involves several new ingredients that are critical to establishing the competitive ratio of ϕ. First, our analysis relies on full characterization of the evolution of the optimal plan over time, in response to packet arrivals and transmission events. This characterization is sketched in section 3 and formally treated in Appendix A. Second, we introduce a new objective function $w_p^t + \phi \cdot w^t(\mathsf{sub}^t(p))$ for selecting a packet p for transmission. This function is based on a definition of substitute packets, $\mathsf{sub}(P, p)$, that accurately reflects the changes in the optimal plan following transmission events, including the case when p is in the initial segment. Third, we introduce the concept of slot monotonicity and devise a way for the algorithm to maintain it over time, using adjustments of weights and deadlines of the packets in the optimal plan. This property is very helpful in keeping track of the optimal profit. Last but not least, we develop several tools used in the amortized analysis of Algorithm $\mathsf{PlanM}(t)$, including the concepts of a backup plan, an adversary's stash, and a novel potential function that captures the relative "advantage" of the algorithm over the adversary in terms of procuring future profits.

$$\tilde{P}^t \quad \underset{\substack{\text{packet} \\ \text{arrivals}}}{\Longrightarrow} \quad P^t \quad \underset{\substack{\text{packet} \\ \text{transmission}}}{\Longrightarrow} \quad \tilde{P}^{t+1}$$

FIG. 5.1. *Notation for snapshots of the optimal plan $\mathcal{P}$.*

5. Competitive analysis. Let ALG be the schedule computed by PlanM for the instance of PacketSchD under consideration, and let OPT be a fixed optimal schedule for this instance. (Actually, OPT can be any schedule for this instance.) For any time step t, by ALG$[t]$ and OPT$[t]$ we will denote packets scheduled by ALG and OPT, respectively, in slot t. (By assumption (A1), we can assume that ALG$[t]$ and OPT$[t]$ are defined for all steps t.) Our overall goal is to show that $\phi \cdot w^0(\text{ALG}) \geq w^0(\text{OPT})$, where, as defined earlier, we use the notation w_j^0 for the original weight of packet j.

Notational convention. The optimal plan changes in the course of the algorithm's run, as a result of new packets arriving or of packets being transmitted. In some contexts, it is convenient to think of the current optimal plan as a dynamic set, which we will denote by $\mathcal{P}$. When more formal treatment is needed, we will use letters $P, Q, \ldots$, often with appropriate subscripts or superscripts, to denote the "snapshot," i.e., the current contents, of the optimal plan before or after a particular change. Namely, as already defined in section 4, P^t is the snapshot of the optimal plan after all packets arrive and before a packet is transmitted in step t. Furthermore, by $\tilde{P}^t$ we denote the optimal plan before any packet arrives in step t. Thus, as a result of transmitting a packet in step t, the optimal plan changes from P^t to $\tilde{P}^{t+1}$. (See Figure 5.1.) We define more snapshots in the analysis of a transmission event, which will be split into more substeps. For clarity, the superscript of a particular snapshot contains the time index with respect to which the optimal plan is computed (unless it is clear from the context).

The same conventions apply to other subsets of pending packets used in the analysis: $\mathcal{A}$ and $\mathcal{B}$, which we will define shortly. In general, we think of such sets of packets as dynamic sets that change over time as new packets arrive, the algorithm transmits packets, and as we adjust the contents of the sets in the analysis. As such, the dynamic sets are denoted with calligraphic letters. Formal analysis requires that we refer to appropriate snapshots of these sets before and after the change under consideration; such snapshots are denoted with italic letters, typically with subscripts or superscripts.

Amortized analysis. We bound the competitive ratio via amortized analysis, using a combination of three accounting mechanisms:

- We use a potential function, which quantifies the advantage of the algorithm over the adversary in future steps. This potential function is defined in section 5.2.
- In leap steps, when the algorithm increases weights of some packets (the substitute packet and possibly some h_i's), we charge it a "penalty" by subtracting the total weight increase from its credit for the step.
- The optimal profit of $w^0(\text{OPT})$ is amortized over all steps using two accounting tools: an "adversary stash" $\mathcal{A}$ and values of the minwt() function. These techniques are introduced in section 5.1.

See section 5.3 for an overview of our analysis, showing how these three techniques can be combined into a formal proof of ϕ-competitiveness of Algorithm PlanM. Then

we give the analysis of packet arrival events in section 5.4 and of packet transmission events in section 5.5.

5.1. Adversary stash. In our analysis, we need a mechanism for keeping track of the adversary's future profit associated with the packets that have already been released. A natural candidate for such a mechanism would be the set of packets that are "pending" for OPT, namely the packets scheduled in OPT that have already been released but not yet transmitted by OPT. This simple definition, however, does not quite work for our purpose, in part because Algorithm PlanM modifies the weights and deadlines of packets that are pending for the adversary.

Instead of OPT, we will use two other concepts which can be defined in terms of packets that are pending for the algorithm. The first one, called the *adversary stash* and denoted $\mathcal{A}$, is used to keep track of the adversary's packets that are in the current optimal plan $\mathcal{P}$; that is, $\mathcal{A}$ is a subset of $\mathcal{P}$ (see invariant (InvA) below). $\mathcal{A}$ is a dynamic set of packets scheduled in some slots in $\{t, t+1, \ldots\}$, where t is the current time step. For $\tau \geq t$, by $\mathcal{A}[\tau]$ we denote the packet scheduled in $\mathcal{A}$ in slot τ; if there is no packet, $\mathcal{A}[\tau]$ is undefined. (Abusing notation, we will use $\mathcal{A}$ to denote both the set of packets in the adversary stash and their schedule.)

The adversary stash evolves over time, partly in response to new events and partly as a result of modifications performed in the course of our analysis. We very briefly outline this process now; a more comprehensive summary is presented later in this section, with all details given during the analysis of packet arrival events in section 5.4 and packet transmission events in section 5.5. Initially, $\mathcal{A}$ is empty, and whenever a packet j such that $j \in$ OPT arrives, we add j to $\mathcal{A}$ at the same slot that j occupies in OPT provided that j is added to $\mathcal{P}$. Sometimes in our analysis of packet arrival and transmission events, as $\mathcal{P}$ changes, we may have to modify $\mathcal{A}$ by removing or replacing packets, in which case the adversary is appropriately compensated. (As a result of such changes, for some $\tau \geq t$, $\mathcal{A}[\tau]$ may differ from OPT$[\tau]$; i.e., the former may be empty or contain a different packet than the latter, even if the latter contains a packet released at or before time t.) Finally, when we analyze the transmission event at a time step t and $\mathcal{A}[t]$ is defined (i.e., nonempty), we will remove packet $\mathcal{A}[t]$ from $\mathcal{A}$ and credit the adversary with the (current) weight of $\mathcal{A}[t]$.

The second accounting mechanism deals with the packets in OPT that are not in $\mathcal{P}$. It turns out that this can be done without explicitly keeping track of such packets. Consider a *pending* packet q that is scheduled in OPT in a future step $\tau > t$. If q is not in $\mathcal{P}$, then its weight is upper bounded by $\mathsf{minwt}(\mathcal{P}, \tau)$. Since $\mathsf{minwt}(\mathcal{P}, \tau)$ for a fixed τ does not decrease (by Lemma 4.1), its weight will be bounded by $\mathsf{minwt}(P^\tau, \tau)$ when the time reaches τ. When it happens, we will allow the adversary to obtain a profit of $\mathsf{minwt}(P^\tau, \tau) \geq w_q^\tau$. While this may seem generous, it does not affect the competitive ratio. The intuition is that in each step τ the adversary can always issue a tight packet of weight just below $\mathsf{minwt}(P^\tau, \tau)$, and this does not change the behavior of the algorithm as such a packet is not in P^τ and cannot be used as a substitute packet due to having deadline at τ. (Section 5.5.1 gives a complete analysis.)

Overall, at any time t, the adversary can receive amortized profit, also called her *credit*, in three ways. Her credit for transmitting a packet is either $w(\mathcal{A}[t])$, if $\mathcal{A}[t]$ is defined, or $\mathsf{minwt}(P^t, t)$ otherwise. In addition, the adversary receives an appropriate compensation when we decrease the total weight of packets in $\mathcal{A}$. We describe the adversary profit amortization more precisely at the end of this section.
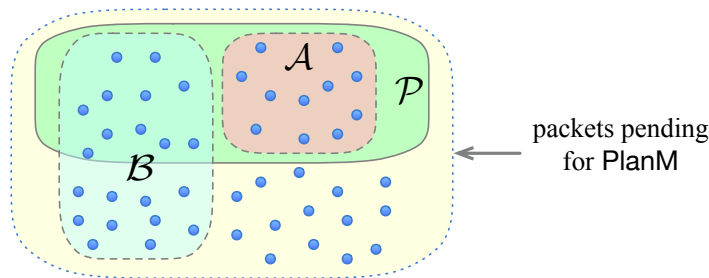
FIG. 5.2. *The sets of packets in the competitive analysis. The backup plan $\mathcal{B}$ is introduced in section* 5.2. *It has the property that* $\mathcal{P} \cap \mathcal{B} = \mathcal{P} \setminus \mathcal{A}$.

Adversary stash invariant. The following invariant, maintained throughout the analysis, captures properties of the adversary stash $\mathcal{A}$ that will be crucial for our argument (see also Figure 5.2):

(InvA)   For any time t and any snapshot A of the adversary stash $\mathcal{A}$ at time t, A contains only packets from the current optimal plan P, i.e., $A \subseteq P$, and each packet $g \in A$ is scheduled in A in a slot in interval $[t, d_g^t]$.

For $\mathcal{A}$ we adopt the same notation of snapshots as for $\mathcal{P}$, namely, $\widetilde{A}^t$ is the adversary stash before any packet arrives in step t and A^t is the adversary stash after all packet arrivals and before a packet is transmitted in step t. We ensure that invariant (InvA) is preserved after each packet arrival and after each transmission event, possibly by changing the adversary stash.

Modifications of the adversary stash. We now overview the principles guiding the maintenance of the adversary stash $\mathcal{A}$. These principles are important in understanding the details of the analysis given in sections 5.3–5.5. Let us fix some slot τ of $\mathcal{A}$. We describe all possible changes that the packet in slot τ of $\mathcal{A}$ can undergo in the course of the analysis and explain how we compensate the adversary for any such change, so that the total adversary credit from slot τ is at least $w^0(\mathsf{OPT}[\tau])$, as needed.

Adding packet $\mathsf{OPT}[\tau]$ to $\mathcal{A}$. When packet $j = \mathsf{OPT}[\tau]$ arrives at a time t, if j is added to the optimal plan $\mathcal{P}$, then we also add it to $\mathcal{A}$ in slot τ. Otherwise, the slot τ remains empty in $\mathcal{A}$ all the time. In either case, the adversary does not get any credit for this packet at this time. (The credit of at least w_j^0 from slot τ will be awarded to the adversary later in the course of computation, possibly in smaller chunks. The strategy for amortizing this credit will be described shortly.)

Replacing packet $\mathcal{A}[\tau]$. Replacement of packets in $\mathcal{A}$ may occur in an iterated leap step when, under some circumstances, we replace a packet $h_i \in \mathcal{A}$ by h_{i+1}, which is always lighter than h_i (see Lemma B.1(b)). To compensate her for this replacement, the adversary obtains credit equal to $w_{h_i}^t - w_{h_{i+1}}^{t+1}$, which is always nonnegative.

To preserve invariant (InvA), when we make packet replacements, we need to make sure that no packet in $\mathcal{A}$ is scheduled after its current deadline, which requires some care for those packets whose deadlines were decreased. We also need to avoid adding into $\mathcal{A}$ a packet that is already in $\mathcal{A}$. Thus, before a packet $h_i \in \mathcal{A}$ gets replaced by h_{i+1}, we remove h_{i+1} from its slot in $\mathcal{A}$ if it already belongs to this set.

Removing packet $\mathcal{A}[\tau]$ before step τ. As mentioned above, in some cases we remove a packet $q = \mathcal{A}[\tau]$ from $\mathcal{A}$ even though the current time has not reached τ yet. This is done in particular if q is no longer in $\mathcal{P}$, due to being ousted or transmitted by the algorithm. However, in order to preserve certain invariants, we may also remove q

from $\mathcal{A}$ even if it remains in $\mathcal{P}$. If we remove packet $q = \mathcal{A}[\tau]$ from $\mathcal{A}$ before step τ, to compensate the adversary for this change, we give her the credit whose value is at least the difference (if positive) between the current weight of q and the current value of $\mathsf{minwt}(\mathcal{P}, \tau)$; the precise formula for this credit will be given below. Importantly, once we remove $\mathcal{A}[\tau]$ from $\mathcal{A}$, $\mathcal{A}[\tau]$ will remain empty forever.

Removing packet $\mathcal{A}[\tau]$ at time τ. When processing the transmission event at time τ, if there is a packet $j = A^{\tau}[\tau]$ in slot τ, we remove j from $\mathcal{A}$ and the adversary gains its *current* weight; i.e., it obtains credit of w_j^{τ}. It follows that $\mathcal{A}$ is empty after the last (transmission) event.

Weight and deadline changes. Increasing the weight of the substitute packet ϱ in a leap step (line 4 of Algorithm PlanM) does not affect $\mathcal{A}$, as ϱ is not in $\mathcal{P}$ and thus not in $\mathcal{A}$, by invariant (InvA). However, the algorithm also changes the weights and/or deadlines of some packets h_i in an iterated leap step (line 11 of Algorithm PlanM), and these packets are in $\mathcal{P}$, so $\mathcal{A}$ might be affected too. To address this, in the analysis of an iterated leap step, we remove or replace *all* packets h_i that are in $\mathcal{A}$ and the adversary gets credit based on their *old* weights. Some of the packets h_i may be reinserted later in the analysis of the same step, but they are always reinserted with their new weights and deadlines. It follows that the weights of all packets in $\mathcal{A}$ are always current.

Amortization of the adversary profit. We now describe how the adversary profit of $w^0(\mathsf{OPT})$ is amortized. As mentioned earlier, when a packet $j = \mathsf{OPT}[\tau]$ arrives at a time t', the adversary credit for j may be awarded in smaller payments in steps $t', t' + 1, \ldots, \tau$. All the payments except the last one are called *adjustment credits* and the last one, in step τ, is called the *transmission credit*. The formulas for these credits are given below.

We note that the adversary credit when processing the arrival of a packet j is always 0. This follows from two properties of the changes in $\mathcal{A}$: First, as explained above, the adversary does not receive credit for the arrival of j, whether it is added to $\mathcal{A}$ or not. Second, the only other change of $\mathcal{A}$ associated with the arrival of j and the resulting modification of $\mathcal{P}$ may be a removal from $\mathcal{A}$ of some packet $\mathcal{A}[\tau]$ of weight at most $\mathsf{minwt}(\mathcal{P}, \tau)$, for which the adversary does not get any credit (see below and section 5.4).

In each step t, we define *the adversary credit for step t*, denoted $\mathsf{advcredit}^t$, as the sum of the following two values:

- *Transmission credit:* This is the credit given to the adversary for her packet in slot t. More precisely, it is defined by

$$\mathsf{advcredit}_t^t = \begin{cases} w^t(A^t[t]) & \text{if } A^t[t] \text{ is defined (not empty)}, \\ \mathsf{minwt}(P^t, t) & \text{otherwise}. \end{cases}$$

  Recall that $w^t(A^t[t])$ represents the weight of $A^t[t]$ at time t if $A^t[t]$ is defined.
- *Adjustment credit:* This is the credit that the adversary receives as compensation for modifications in $\mathcal{A}$. Namely, it is the sum of the following credits, one for each adjustment of $\mathcal{A}$ performed in the analysis of step t:

  - *Adjustment credit for weight decreases:* The difference between previous and new weight of $\mathcal{A}[\tau]$, for each slot $\tau \geq t$ where packet $\mathcal{A}[\tau]$ is replaced by a lighter packet.
  - *Adjustment credit for packet removal:* The value of $w_q^t - \mathsf{minwt}(P^t, d_q^t)$ for each packet $q = \mathcal{A}[\tau]$ removed from $\mathcal{A}$.

  Before proceeding, we make two simple but important observations. First, all adjustment credits are nonnegative. This is obvious for adjustment credits

for weight decreases. As for the second type of adjustments, consider a packet $q = \mathcal{A}[\tau]$ getting removed from $\mathcal{A}$ at time t. By invariant (InvA), we have that $q \in P^t$. Then the definition of function $\mathsf{minwt}()$ implies that $w_q^t \geq \mathsf{minwt}(P^t, d_q^t)$; in other words the adjustment credits for packet removals are also nonnegative.

Second, observe that the formula above for the adjustment credit for packet removals is an upper bound on the actual loss of adversary's future profit due to the removal of $q = \mathcal{A}[\tau]$ from $\mathcal{A}$, which equals $\max\{w_q^t - \mathsf{minwt}(P^\tau, \tau), 0\}$ (as the transmission credit in step τ is $\mathsf{minwt}(P^\tau, \tau)$). Indeed, we have $\mathsf{minwt}(P^t, d_q^t) \leq w_q^t$ and, since $\tau \leq d_q^t$, also $\mathsf{minwt}(P^t, d_q^t) \leq \mathsf{minwt}(P^t, \tau) \leq \mathsf{minwt}(P^\tau, \tau)$ by the slot-monotonicity property (Lemma 4.1). Therefore $w_q^t - \mathsf{minwt}(P^t, d_q^t) \geq \max\{w_q^t - \mathsf{minwt}(P^\tau, \tau), 0\}$. We use the value of $w_q^t - \mathsf{minwt}(P^t, d_q^t)$ here because it is sufficient for the analysis and easier to work with.

At each step t, these credit adjustments (if any) may be performed in multiple substeps, with each substep involving processing some time interval $(\zeta, \eta]$. We will use notation $\mathsf{advcredit}_{(\zeta, \eta]}^t$ for the adjustment credit resulting from modifications in the substep corresponding to time interval $(\zeta, \eta]$.

CLAIM 5.1. *The total adversary credit for all steps covers* $w^0(\mathsf{OPT})$, *that is,*

$$(5.1) \qquad w^0(\mathsf{OPT}) \leq \sum_t \mathsf{advcredit}^t,$$

where the sum is over all steps t.

Proof. To prove (5.1), consider again a packet $j = \mathsf{OPT}[\tau]$ that arrived at some step t'. It is sufficient to show that the sum of all adversary credits associated with slot τ for steps $t = t', t'+1, \ldots, \tau$, is at least $w_j^0 = w_j$.

At time t', if j is added to $\mathcal{P}$, then j is also added to $\mathcal{A}$; otherwise $\mathcal{A}[\tau]$ remains undefined. To streamline the proof, we think of the second case as adding j to $\mathcal{A}$ in slot τ and then removing it immediately in the same step.

At any step t when $\mathcal{A}[\tau]$ is defined, if the weight of $\mathcal{A}[\tau]$ is decreased, then, according to the definitions above, this decrease of weight contributes to the adjustment credit at step t. Consider the last step t'' when $\mathcal{A}[\tau]$ is defined, and let $q = \mathcal{A}[\tau]$ be the packet in slot τ of $\mathcal{A}$ at that time. (It could be that $q \neq j$, as the packet in $\mathcal{A}[\tau]$ may change over time.) Then the total adjustment credit in steps $t', t'+1, \ldots, t''-1$ for changing $\mathcal{A}[\tau]$ equals $w_j^0 - w_q^{t''}$.

There are two cases. If $t'' = \tau$, then the adversary will receive transmission credit of $\mathsf{advcredit}_\tau^\tau = w_q^\tau$ in step τ, and the right-hand side of (5.1) associated with slot τ (the sum of adjustment credits and the transmission credit for slot τ) is $w_j^0 - w_q^\tau + w_q^\tau = w_j^0$. If $t'' < \tau$, then q is removed from $\mathcal{A}$ in the analysis of step t''. The adjustment credits for steps $t = t', t'+1, \ldots, t''-1$, add up to $w_j^0 - w_q^{t''}$, and the adjustment credit in step t'' for removing q from $\mathcal{A}$ is $w_q^{t''} - \mathsf{minwt}(P'', d_q^{t''})$, where P'' is the snapshot of $\mathcal{P}$ when this removal of q is taking place. Then later at step τ the adversary will receive transmission credit $\mathsf{minwt}(P^\tau, \tau)$, by the definition above. Thus the total of adjustment and transmission credits associated with slot τ is at least

$$(w_j^0 - w_q^{t''}) + (w_q^{t''} - \mathsf{minwt}(P'', d_q^{t''})) + \mathsf{minwt}(P^\tau, \tau)$$
$$(5.2) \qquad\qquad \geq w_j^0 - w_q^{t''} + (w_q^{t''} - \mathsf{minwt}(P'', \tau)) + \mathsf{minwt}(P'', \tau)$$
$$= w_j^0,$$

where inequality (5.2) holds because $\mathsf{minwt}(P'', d_q^{t''}) \leq \mathsf{minwt}(P'', \tau)$, by the definition of function $\mathsf{minwt}()$ (see also the comments on adjustments credits for packet removals before Claim 5.1) and because $\mathsf{minwt}(P^\tau, \tau) \geq \mathsf{minwt}(P'', \tau)$, which follows from the slot monotonicity property, as summarized in Lemma 4.1. This concludes the proof of (5.1).   □

5.2. Backup plan and the potential function. In our analysis, we will maintain a set $\mathcal{B}$ of pending packets called a *backup plan*. $\mathcal{B}$ contains two types of packets: all packets in $\mathcal{P} \setminus \mathcal{A}$, and some pending packets not in $\mathcal{P}$. (The relation between $\mathcal{P}$, $\mathcal{B}$ and $\mathcal{A}$ is illustrated in Figure 5.2.) The packets in $\mathcal{B} \setminus \mathcal{P}$ will typically be packets that were earlier ousted from $\mathcal{P}$, either as a result of arrivals of other packets or in a leap step. These packets can also be thought of as candidates for the substitute packet $\mathsf{sub}^t(p)$ when the algorithm chooses a packet p for transmission.

The following invariant summarizes the essential properties of $\mathcal{B}$, and it will be maintained throughout the analysis:

(InvB) For any snapshot B of $\mathcal{B}$ at any time t, (i) B is a plan, i.e., a feasible set of packets pending in step t, and (ii) $B \cap P = P \setminus A$, where P and A are the current snapshots of $\mathcal{P}$ and $\mathcal{A}$, respectively.

By Observation 3.1, invariant (InvB)(i) is equivalent to the condition that for any slot $\tau \geq t$, we have $\mathsf{pslack}(B, \tau) \geq 0$. Invariant (InvB)(ii) and invariant (InvA) together imply that $B \cap A = \emptyset$ and $\mathcal{P} = (\mathcal{P} \cap \mathcal{B}) \cup \mathcal{A}$, i.e., that any packet in P is either in A or in B, but not in both. Similarly as for invariant (InvA), preserving invariant (InvB) in the course of the analysis will require making suitable changes in the adversary stash $\mathcal{A}$ and the backup plan $\mathcal{B}$.

The following observation is quite straightforward, but we state it explicitly here, as it is useful later in some proofs.

Observation 5.2. Consider the current optimal plan P, the backup plan B, and the adversary stash A at time t. Assume that invariants (InvA) and (InvB) hold. Let η be a tight slot of P. Then

(a) $A(\eta, T] \neq \emptyset$ implies that $B(\eta, T] \setminus P \neq \emptyset$, and
(b) $B[t, \eta] \setminus P \neq \emptyset$ implies that $A[t, \eta] \neq \emptyset$.

Proof. (a) From the property that all plans are full and that η is a tight slot of P, we have $|P(\eta, T]| = T - \eta \leq |B(\eta, T]|$, where T is the time horizon (see section 3). Then, using invariants (InvA) and (InvB), we obtain that $|A(\eta, T]| = |P(\eta, T] \setminus B| \leq |B(\eta, T] \setminus P|$.

(b) Since η is a tight slot of P and B is feasible, we have $|P[t, \eta]| = \eta - t + 1 \geq |B[t, \eta]|$. Therefore, if $B[t, \eta] \setminus P \neq \emptyset$, then $P[t, \eta] \setminus B \neq \emptyset$, which implies that $A[t, \eta] \neq \emptyset$, by invariant (InvB).   □

The definitions of tight slots and segments apply to $\mathcal{B}$ (as well as to any plan), and they will be helpful in our proofs. We remark that $\mathcal{B}$ may have a different segment structure than the current optimal plan $\mathcal{P}$.

Before defining our potential function, we introduce a few lemmas that will be useful in showing that invariant (InvB) is preserved after each step.

LEMMA 5.3. *Consider the current optimal plan P, the backup plan B, and the adversary stash A at time t. Assume that invariants* (InvA) *and* (InvB) *hold, and let η, η' be two time slots such that $t \leq \eta \leq \eta'$. Then*

$$(5.3) \quad \mathsf{pslack}(P, \eta') - \mathsf{pslack}(P, \eta) + |A(\eta, \eta']| \;=\; \mathsf{pslack}(B, \eta') - \mathsf{pslack}(B, \eta) + |B(\eta, \eta'] \setminus P|.$$

Equation (5.3) may appear complicated, but it's actually quite straightforward. It compares the contributions of the interval $(\eta, \eta']$ to $\mathsf{pslack}()$ values of P and B. These contributions differ by $-|P(\eta, \eta')| + |B(\eta, \eta')|$, and canceling out the common contribution $|P(\eta, \eta') \cap B|$ yields (5.3). A formal proof follows.

Proof. (InvB) implies that $A(\eta, \eta') = P(\eta, \eta') \setminus B$. Using the definition of $\mathsf{pslack}()$, and canceling out the contributions of packets with deadline at most η, we get

$$
\begin{aligned}
\mathsf{pslack}(P, \eta') - \mathsf{pslack}(P, \eta) + |A(\eta, \eta')| &= \eta' - \eta - |P(\eta, \eta')| + |P(\eta, \eta') \setminus B| \\
&= \eta' - \eta - |P(\eta, \eta') \cap B| \\
&= \eta' - \eta - |B(\eta, \eta')| + |B(\eta, \eta') \setminus P| \\
&= \mathsf{pslack}(B, \eta') - \mathsf{pslack}(B, \eta) + |B(\eta, \eta') \setminus P|,
\end{aligned}
$$

completing the proof. $\qquad\square$

LEMMA 5.4. *Consider the current optimal plan P, the backup plan B, and the adversary stash A at time t. Assume that invariants* (InvA) *and* (InvB) *hold, and let ζ be a tight slot of P (possibly, $\zeta = t - 1$). Let f^* be the earliest-deadline packet in $B(\zeta, T] \setminus P$ and let g^* be the latest-deadline packet in $A[t, \zeta]$. (We allow here the possibility that f^* or g^* is undefined). Then*

> (a) *If f^* is defined, then $\mathsf{pslack}(B, \tau) \geq \mathsf{pslack}(P, \tau) + |A(\zeta, \tau]|$ for any $\tau \in (\zeta, d_{f^*})$. Otherwise this inequality holds for any $\tau > \zeta$.*
> (b) *If g^* is defined, then $\mathsf{pslack}(B, \tau) \geq \mathsf{pslack}(P, \tau) + |B(\tau, \zeta] \setminus P|$ for any $\tau \in [d_{g^*}, \zeta]$. Otherwise this inequality holds for any $\tau \in [t, \zeta]$.*

Proof. We first observe that, since B is feasible and ζ is a tight slot for P, we have $\mathsf{pslack}(B, \zeta) \geq 0 = \mathsf{pslack}(P, \zeta)$.

(a) If packet f^* exists, let $\theta = d_{f^*}$; otherwise let $\theta = T + 1$, where T is the time horizon, as defined in section 2. Let $\tau \in (\zeta, \theta)$. By the definition of f^*, there is no packet in $B \setminus P$ with deadline in (ζ, θ); in particular $B(\zeta, \tau] \setminus P = \emptyset$. Using this, equation $\mathsf{pslack}(P, \zeta) = 0$, and Lemma 5.3 (with $\eta = \zeta$ and $\eta' = \tau$), we obtain

$$
\mathsf{pslack}(P, \tau) + |A(\zeta, \tau]| = \mathsf{pslack}(B, \tau) - \mathsf{pslack}(B, \zeta) \leq \mathsf{pslack}(B, \tau),
$$

which implies claim (a).

(b) If packet g^* exists, let $\lambda = d_{g^*}$; otherwise let $\lambda = t$. Let $\tau \in [\lambda, \zeta]$. By the definition of λ, there is no packet in A with deadline in $(\lambda, \zeta]$, which implies that $A(\tau, \zeta] = \emptyset$. Using this, equation $\mathsf{pslack}(P, \zeta) = 0$, and Lemma 5.3 (with $\eta = \tau$ and $\eta' = \zeta$), we obtain

$$
-\mathsf{pslack}(P, \tau) = \mathsf{pslack}(B, \zeta) - \mathsf{pslack}(B, \tau) + |B(\tau, \zeta] \setminus P|.
$$

Using $\mathsf{pslack}(B, \zeta) \geq 0$, we get $-\mathsf{pslack}(P, \tau) \geq -\mathsf{pslack}(B, \tau) + |B(\tau, \zeta] \setminus P|$ and claim (b) follows. $\qquad\square$

In some cases of the analysis, we will have situations when a packet $g \in \mathcal{A}$ needs to be removed from $\mathcal{A}$. By the definition of the backup plan, this causes g to be added to $\mathcal{B}$, making $\mathcal{B}$ infeasible. The lemma below shows that we can restore the feasibility of $\mathcal{B}$ after such a change by removing from $\mathcal{B}$ a suitably chosen packet that is not in $\mathcal{P}$.

LEMMA 5.5. *Consider the current optimal plan P, the backup plan B, and the adversary stash A at time t. Assume that invariants* (InvA) *and* (InvB) *hold. Let $g \in A$, $\eta = \mathsf{prevts}(P, d_g)$, and $\eta' = \mathsf{nextts}(B, d_g)$. Then*

(a) $B(\eta, \eta'] \setminus P \neq \emptyset$; and

(b) *for any $f \in B(\eta, \eta'] \setminus P$, the set $C = B \setminus \{f\} \cup \{g\}$ is feasible and $w_f \leq w_g$*
(thus $w(C) \geq w(B)$).

Proof. Note that the choice of $g \in A$ implies that $g \notin B$. We use Lemma 5.3
with η and η' as defined here. We can apply this lemma because B satisfies invariant
(InvB). From that lemma, substituting $\mathsf{pslack}(P, \eta) = \mathsf{pslack}(B, \eta') = 0$, we get

$$\mathsf{pslack}(P, \eta') + |A(\eta, \eta']| \;=\; -\mathsf{pslack}(B, \eta) + |B(\eta, \eta'] \setminus P| \;\leq\; |B(\eta, \eta'] \setminus P|,$$

where the last inequality follows from $\mathsf{pslack}(B, \eta) \geq 0$, which is a consequence of
the feasibility of B. The existence of g implies $A(\eta, \eta'] \neq \emptyset$, and hence, using the
feasibility of P, we obtain that $B(\eta, \eta'] \setminus P \neq \emptyset$ holds as well, proving claim (a).

Pick any $f \in B(\eta, \eta'] \setminus P$. From $d_f \leq \eta' = \mathsf{nextts}(B, d_g)$ we obtain that C is
feasible. Inequality $d_f > \eta = \mathsf{prevts}(P, d_g)$ and $f \notin P$ imply that $w_f \leq \mathsf{minwt}(P, d_g) \leq
w_g$, and therefore $w(C) \geq w(B)$, completing the proof of claim (b). □

As a corollary, we obtain that Lemma 5.5(b) holds for f equal to the earliest-
deadline packet f^* in $B \setminus P$ with $d_{f^*} > \mathsf{prevts}(P, d_g)$.

COROLLARY 5.6. *Consider the current optimal plan P, the backup plan B, and
the adversary stash A at time t. Assume that invariants (InvA) and (InvB) hold. Let
$g \in A$, $\eta = \mathit{prevts}(P, d_g)$, and let f^* be the earliest-deadline packet in $B(\eta, T] \setminus P$.
Then packet f^* exists, the set $C = B \setminus \{f^*\} \cup \{g\}$ is feasible, and $w_{f^*} \leq w_g$ (thus
$w(C) \geq w(B)$).*

Proof. Applying Lemma 5.5(a) to packet g and $\eta' = \mathsf{nextts}(B, d_g)$ gives us that
$B(\eta, \eta'] \setminus P \neq \emptyset$, which in turn implies that $d_{f^*} \in (\eta, \eta']$. Then, using Lemma 5.5(b)
for $f = f^*$ implies the corollary. □

Potential function. We use the backup plan $\mathcal{B}$ to define a potential function
needed for the amortized analysis of Algorithm PlanM. If B is the current snapshot
of $\mathcal{B}$, then the potential value at time t is

$$(5.4) \qquad\qquad \Psi(B) \;=\; \frac{1}{\phi}\, w^t(B)\,.$$

For brevity, we will use notation $\widetilde{\Psi}^t = \Psi(\widetilde{B}^t)$ for the potential at the beginning of step
t, before any packet arrives, and $\Psi^t = \Psi(B^t)$ for the potential just before a packet is
transmitted in step t.

The intuition behind this definition is as follows. In order to be ϕ-competitive,
the average (per step) profit of Algorithm PlanM should be at least $1/\phi$ times the
adversary's average profit. However, due to the choice of coefficients in line 1 of the
algorithm, Algorithm PlanM tends to postpone transmitting heavy packets with large
deadlines. (For example, given just two packets, a tight packet with weight 1 and a
nontight packet with weight $2.6 < \phi^2$, the algorithm will transmit the tight packet
in the current step.) As a result, in tight instances, PlanM's actual profit per step,
throughout most of the game, is often smaller than $1/\phi$ times the adversary's, and
only near the end of the instance, when delayed heavy packets are transmitted, the
algorithm can make up for this deficit.

In our amortized analysis, if there is a deficit in a given step, we pay for it
with a "loan" that is represented by an appropriate increase of the potential function.
Eventually, of course, these loans need to be repaid; the potential eventually decreases
to 0, and this decrease must be covered by excess profit. The formula for the potential

Ψ is designed to guarantee such future excess profits. To see this, imagine that no more packets arrive. Since $\mathcal{B} \cap \mathcal{A} = \emptyset$, the packets in $\mathcal{B}$ will not be transmitted in the future by the adversary. If the algorithm does not execute any more leap steps, then it will collect all the packets in $\mathcal{P}$, which are in total heavier than the packets in $\mathcal{B}$ (as $\mathcal{B}$ is feasible by invariant (InvB)(i) and each packet $f \in \mathcal{B} \backslash \mathcal{P}$ satisfies $w_f^t \leq \mathsf{minwt}(\mathcal{P}, d_f^t)$ at each time step t). On the other hand, if the algorithm executes a leap step, instead of a packet ousted from $\mathcal{P}$ in this step, the algorithm will collect a packet from $\mathcal{B} \backslash \mathcal{P}$ (or a better packet) after it is added to the optimal plan as a substitute packet.

5.3. Overview of the analysis. This section gives an overview of the analysis, states the main theorem, and shows how it follows from results that will be established in the sections that follow.

Initial and final state. At the beginning, per assumption (A1), we assume that the optimal plan is prefilled with virtual 0-weight packets, each in a slot equal to its deadline, and none of them scheduled by the adversary for transmission. The adversary stash $\mathcal{A}$ is empty; i.e., before the first step (at time 0) we have $\widetilde{A}^0 = \emptyset$, and the backup plan is the same as the optimal plan, i.e., $\widetilde{B}^0 = \widetilde{P}^0$. Thus invariants (InvA) and (InvB) clearly hold, and $\widetilde{\Psi}^0 = 0$. At the end, after all (nonvirtual) packets expire, the potential equals 0 as well, i.e., $\widetilde{\Psi}^{T+1} = 0$, where T is the time horizon (the last step).

Amortized analysis. At the core of our analysis are bounds relating amortized profits of the algorithm and the adversary in each step t. For packet arrivals in step t, we will show the following *packet-arrivals inequality*:

$$(5.5) \qquad \qquad \Psi^t - \widetilde{\Psi}^t \geq 0 \,.$$

For the transmission event in a step t, we will show that the following *packet-transmission inequality* holds:

$$(5.6) \qquad \phi \, w^t(\mathsf{ALG}[t]) - \phi \, (\Delta^t w) + (\widetilde{\Psi}^{t+1} - \Psi^t) \ \geq \ \mathsf{advcredit}^t,$$

where $\mathsf{ALG}[t]$ is the packet in slot t in the algorithm's schedule ALG (thus $w^t(\mathsf{ALG}[t])$ is the algorithm's profit), and $\Delta^t w$ is the total amount by which the algorithm increases the weights of its pending packets in step t.

We prove the packet-arrivals inequality (5.5) in section 5.4 and the packet-transmission inequality (5.6) in section 5.5. Assuming that these two inequalities hold, we now show our main result.

THEOREM 5.7. *Algorithm PlanM is ϕ-competitive.*

Proof. We show that $\phi \, w^0(\mathsf{ALG}) \geq w^0(\mathsf{OPT})$, which implies the theorem. First, by Claim 5.1, we have $w^0(\mathsf{OPT}) \leq \sum_t \mathsf{advcredit}^t$. Second, note that

$$(5.7) \qquad \qquad \sum_t (\widetilde{\Psi}^{t+1} - \widetilde{\Psi}^t) \ = \ \widetilde{\Psi}^{T+1} - \widetilde{\Psi}^0 \ = \ 0 \,,$$

where $\widetilde{\Psi}^0 = 0$ is the initial potential and $\widetilde{\Psi}^{T+1} = 0$ is the final potential after the last step T. Observe also that

$$(5.8) \qquad \qquad \sum_t [\, w^t(\mathsf{ALG}[t]) - \Delta^t w \,] \ \leq \ w^0(\mathsf{ALG}) \,.$$

This follows from the observation that if the weight of $\mathsf{ALG}[\tau]$ was increased by some value $\delta > 0$ at some step $t' < \tau$, then δ also contributes to $\Delta^{t'} w$, so such contributions

cancel out in (5.8). (There may be several such δ's, as the weight of a packet may have been increased multiple times. Note that the bound (5.8) may not be tight if some packets with increased weights are later dropped.)

Summarizing, using the above bounds yields

$$(5.9) \quad w^0(\mathsf{OPT}) \leq \sum_t \mathsf{advcredit}^t$$

$$(5.10) \qquad \leq \sum_t \left[\phi\, w^t(\mathsf{ALG}[t]) - \phi\,(\Delta^t w) + (\widetilde{\Psi}^{t+1} - \Psi^t) \right] + \sum_t (\Psi^t - \widetilde{\Psi}^t)$$

$$\qquad = \sum_t \left[\phi\, w^t(\mathsf{ALG}[t]) - \phi\,(\Delta^t w) \right] + \sum_t (\widetilde{\Psi}^{t+1} - \widetilde{\Psi}^t)$$

$$(5.11) \qquad \leq \phi\, w^0(\mathsf{ALG}),$$

where inequality (5.9) is (5.1), inequality (5.10) follows by applying (5.5) and (5.6) for each step t, and inequality (5.11) holds by (5.7) and (5.8). □

5.4. Packet arrivals. Let t be the current time step. Our aim in this section is to prove that invariants (InvA) and (InvB) can be preserved as packets arrive in step t, using appropriate modifications of sets $\mathcal{A}$ and $\mathcal{B}$. We also prove that the packet-arrivals inequality (5.5) holds for step t. To this end, it is sufficient to show how to preserve both invariants, without decreasing the value of $w(\mathcal{B})$ (and thus Ψ as well), in response to an arrival of each individual packet.

Thus, consider the arrival event of a packet s at time t. Let P be the optimal plan just before s arrives, and let Q be the optimal plan just after s arrives. Furthermore, let A and B be the snapshots of $\mathcal{A}$ and $\mathcal{B}$ just before s arrives, and let A_{+s} and B_{+s} denote, respectively, the snapshots of $\mathcal{A}$ and $\mathcal{B}$ just after s arrives and the changes described below are applied. The algorithm does not change the weights and deadlines after packet arrivals, so we will omit the superscript t in the notation for weights and deadlines, that is $w_q = w_q^t$ and $d_q = d_q^t$, for each packet q. There are two cases, depending on whether or not $s \in Q$.

Case A.1. s is not added to the optimal plan $\mathcal{P}$, i.e., $Q = P$. Lemma A.2 implies that $w_s < \mathsf{minwt}(P, d_s) = \mathsf{minwt}(Q, d_s)$. In this case, we do not change $\mathcal{A}$, i.e., $A_{+s} = A$, so invariant (InvA) is preserved. The backup plan $\mathcal{B}$ remains the same as well, so $w(\mathcal{B})$ does not change and invariant (InvB) is preserved.

Case A.2. s is added to the optimal plan $\mathcal{P}$. Let u be the lightest packet in P with $d_u \leq \mathsf{nextts}(P, d_s)$; by assumption (A1) such u exists. By the choice of u and Lemma A.2, we have $Q = P \cup \{s\} \setminus \{u\}$, $\mathsf{nextts}(P, d_u) \leq \mathsf{nextts}(P, d_s)$, and $w_s > w_u = \mathsf{minwt}(P, d_s) = \mathsf{minwt}(P, d_u)$.

Replacing u by s in $\mathcal{P}$ can also trigger changes in $\mathcal{A}$ and $\mathcal{B}$. We describe these changes in two parts:

   (i) First, we show that if $u \in A$, then we can remove u from $\mathcal{A}$, preserving the invariants and not decreasing the potential.

   (ii) Then, assuming that $u \notin \mathcal{A}$, we describe and analyze the remaining changes. Let $A_{(\mathrm{i})}$ denote the intermediate adversary stash, after the change in (i) and before the change in (ii), where we let $A_{(\mathrm{i})} = A$ if $u \notin A$, that is when change (i) does not apply. We adopt the same notation for snapshots of set $\mathcal{B}$.

   (i) *Dealing with $u \in A$.* In this case we need to remove u from $\mathcal{A}$ to preserve invariant (InvA). But this also forces us to add u to $\mathcal{B}$ in order to preserve (InvB)(ii), which in turn requires removing some packet from $\mathcal{B}$ to preserve its feasibility.

To implement these changes, we use Lemma 5.5 with $g = u$, which gives us that there is a packet $f_u \in B \setminus P$ such that the set $B \setminus \{f_u\} \cup \{u\}$ is feasible and $w(B \setminus \{f_u\} \cup \{u\}) \geq w(B)$. We then let

$$A_{(\mathrm{i})} = A \setminus \{u\} \quad \text{and} \quad B_{(\mathrm{i})} = B \setminus \{f_u\} \cup \{u\}.$$

Since u is scheduled in A in interval $[t, d_u]$ and $w_u = \mathsf{minwt}(P, d_u)$, the adversary adjustment credit associated with removing u from $\mathcal{A}$ is $w_u - \mathsf{minwt}(P, d_u) = 0$ (see section 5.1). As explained above, we also have that $B_{(\mathrm{i})}$ is feasible and $w(B_{(\mathrm{i})}) \geq w(B)$; thus invariant (InvB) is preserved and the change of $w(\mathcal{B})$ is nonnegative.

(ii) *Analysis of other changes.* By (i) we can now assume that $u \notin A_{(\mathrm{i})}$, and invariant (InvB)(ii) together with $u \in P$ imply that $u \in B_{(\mathrm{i})}$. We now consider the changes of $\mathcal{B}$ resulting from including s in $\mathcal{P}$. We analyze two subcases, depending on whether or not $s \in \mathsf{OPT}$.

Case A.2.a. $s \in \mathsf{OPT}$. We add s to $\mathcal{A}$ in the same slot as in OPT. Specifically, we take $A_{+s} = A_{(\mathrm{i})} \cup \{s\}$ and $B_{+s} = B_{(\mathrm{i})}$. Invariant (InvA) is preserved as $s \in A_{+s}$, invariant (InvB) holds as $\mathcal{B}$ remains unaffected in (ii), and the value of $w(\mathcal{B})$ does not change.

Case A.2.b. $s \notin \mathsf{OPT}$. In this case, we simply let $A_{+s} = A_{(\mathrm{i})}$, so invariant (InvA) continues to hold. However, s needs to get added to $\mathcal{B}$ to preserve invariant (InvB)(ii), so we need to remove some packet from $\mathcal{B}$ to maintain its feasibility, and this packet must be lighter than s to ensure that the potential does not decrease. Define $\xi = \mathsf{nextts}(P, d_s)$ and $\xi_B = \mathsf{nextts}(B_{(\mathrm{i})}, d_s)$. We have two cases.

If $d_u \leq \xi_B$, we replace u by s in $\mathcal{B}$. That is, we let $B_{+s} = B_{(\mathrm{i})} \cup \{s\} \setminus \{u\}$, and this satisfies invariant (InvB)(ii). The case condition implies that invariant (InvB)(i) continues to hold, and, as $w_u < w_s$ in Case A.2, the value of $w(\mathcal{B})$ does not decrease.

Next, assume that $d_u > \xi_B$. Note that $d_u \leq \xi$, by the definition of u. Let $\lambda = \mathsf{prevts}(P, d_s)$. In this case we have that $\lambda < d_s \leq \xi_B < d_u \leq \xi$, so d_u and d_s are in the same segment $(\lambda, \xi]$ of P. Since λ is a tight slot for P, ξ_B is a tight slot for $B_{(\mathrm{i})}$, and $\mathsf{pslack}(P, \xi_B) > 0$, from Lemma 5.3 with $\eta = \lambda$ and $\eta' = \xi_B$ we get that

$$|B_{(\mathrm{i})}(\lambda, \xi_B] \setminus P| = \mathsf{pslack}(B_{(\mathrm{i})}, \lambda) + \mathsf{pslack}(P, \xi_B) + |A_{(\mathrm{i})}(\lambda, \xi_B]| > 0,$$

implying $B_{(\mathrm{i})}(\lambda, \xi_B] \setminus P \neq \emptyset$. Choose any $f \in B_{(\mathrm{i})}(\lambda, \xi_B] \setminus P$ and let $B_{+s} = B_{(\mathrm{i})} \cup \{s\} \setminus \{f\}$, preserving invariant (InvB)(ii). Invariant (InvB)(i) holds for B_{+s} because $d_f \leq \xi_B = \mathsf{nextts}(B_{(\mathrm{i})}, d_s)$. The optimality of P and the choice of u and f imply that $w_f \leq \mathsf{minwt}(P, d_s) = w_u < w_s$; thus $w(\mathcal{B})$ cannot decrease.

Summarizing, we showed that in response to the arrival of any packet s in step t we can modify $\mathcal{A}$ and $\mathcal{B}$ in such a way that the invariants (InvA) and (InvB) will be preserved and the value of Ψ will not decrease. This gives us that the packet-arrivals inequality (5.5) holds for step t and that after packet arrivals both invariants (InvA) and (InvB) will hold, concluding the analysis of packet arrivals in step t.

5.5. Transmitting a packet. Let t be the current step of the computation. After all packets with release time equal to t arrive, the algorithm transmits its packet $p = \mathsf{ALG}[t]$. Recall that P^t is the optimal plan just before transmitting p and $\widetilde{P}^{t+1}$ is the optimal plan after the algorithm transmits p and possibly adjusts weights and deadlines, and after the time is incremented to $t + 1$.

We split the analysis of the transmission step into two parts, called the adversary step and the algorithm's step, defined as follows:

Adversary step: If packet $A^t[t]$ is defined, say $A^t[t] = j$, then we need to remove it from $\mathcal{A}$. (Recall that packets in $\mathcal{A}$ may have been removed or replaced during the analysis, so j may not be equal to $\mathsf{OPT}[t]$, the packet scheduled in OPT at time t.) Removing j from $\mathcal{A}$ could trigger a change in $\mathcal{B}$, but the optimal plan $\mathcal{P}$ remains unchanged. We show that these changes preserve both invariants (InvA) and (InvB). We should stress that these changes are made without advancing the current time, which will be done in algorithm's step.

In the analysis of the adversary step, given in Section 5.5.1, we establish a relation (inequality (5.12)) between the adversary transmission credit and the change of the potential. By A^t_{adv} and B^t_{adv} we will denote the snapshots of sets $\mathcal{A}$ and $\mathcal{B}$, respectively, right after the adversary step.

Algorithm's step: In the algorithm's step, the algorithm transmits p, the time is incremented to $t + 1$, and the optimal plan changes from P^t to $\tilde{P}^{t+1}$.

The analysis of this step assumes that the changes described in the adversary step have already been implemented. Using the bound (5.12), invariants (InvA) and (InvB), and other properties, we then show that the packet-transmission inequality (5.6) holds after the sets $\mathcal{P}$, $\mathcal{A}$, and $\mathcal{B}$ are updated to reflect the changes triggered by the packet's transmission. We also ensure that invariants (InvA) and (InvB) are preserved.

The analysis of the algorithm's step is given in sections 5.5.2–5.5.6. We first analyze the ordinary step in section 5.5.2. We then give a roadmap for the analysis of a leap step in section 5.5.3, as it will be divided into substeps. We analyze the particular substeps in section 5.5.4, which describes the changes in the initial segment $[t, \alpha]$ of P^t, and in sections 5.5.5–5.5.6, which contain the analysis of other changes resulting from a leap step.

In Table 1, we summarize the notation of snapshots that we use in this section.

5.5.1. Adversary step. As defined in section 5.1, if A^t contains a packet in slot t, then the adversary gains the transmission credit of $\mathsf{advcredit}^t_t = w^t(A^t[t])$. Otherwise, the adversary's transmission credit equals $\mathsf{advcredit}^t_t = \mathsf{minwt}(P^t, t)$. (The overall adversary credit for this step also includes the adjustment credit, but in this section we focus only on the relation between the transmission credit and the potential change in the adversary step.)

Except for possibly removing packet $A^t[t]$ from $\mathcal{A}$, if it is defined, we will not make other changes to $\mathcal{A}$, so invariant (InvA) will be preserved. Below we show that with appropriate changes invariant (InvB) will also be preserved after the adversary step. Further, denoting by $\Delta_{\mathrm{adv}}\Psi$ the change of the potential in the adversary step, we prove the following auxiliary inequality:

$$(5.12) \qquad \Delta_{\mathrm{adv}}\Psi - \mathsf{advcredit}^t_t \;\geq\; -\frac{1}{\phi^2}\, w^t_p - \frac{1}{\phi}\, w^t(\mathsf{sub}^t(p))\,.$$

Recall that throughout section 5.5 we denote the packet scheduled by the algorithm in step t by p.

The proof of inequality (5.12) is divided into two cases, depending on whether or not $A^t[t]$ is defined. As packet weights are not changed in the adversary step, below we omit the superscript t in the notation for weights.

TABLE 1
Notation of snapshots of sets $\mathcal{P}, \mathcal{B}$, and $\mathcal{A}$ used in the analysis. The particular snapshots for each substep of the analysis of a leap step will be defined later, in the respective sections (their numbers are given in parentheses).

				Set description	Optimal plan	Backup plan	Adversary stash
				Dynamic set	$\mathcal{P}$	$\mathcal{B}$	$\mathcal{A}$
				Beginning of step t (before packet arrivals)	$\widetilde{P}^t$	$\widetilde{B}^t$	$\widetilde{A}^t$
				Before transmission in step t (after arrivals)	P^t	B^t	A^t
Trans-mission event in step t			After the adversary step (5.5.1)		P^t	B^t_{adv}	A^t_{adv}
	Leap step (5.5.3)		After processing $[t, \alpha]$ and incrementing time (5.5.4)		P^{t+1}_{ins}	B^{t+1}_{ins}	A^{t+1}_{ins}
		Iterated leap step (5.5.6)	Before stage for group $\langle a,b\rangle$		$\dot{P}^{t+1}_{\langle a,b\rangle}$	$\dot{B}^{t+1}_{\langle a,b\rangle}$	$\dot{A}^{t+1}_{\langle a,b\rangle}$
			After stage for group $\langle a,b\rangle$		$\ddot{P}^{t+1}_{\langle a,b\rangle}$	$\ddot{B}^{t+1}_{\langle a,b\rangle}$	$\ddot{A}^{t+1}_{\langle a,b\rangle}$
			Local notation (after part (i) of changes)		$P^t_{(i)}$	$B^t_{(i)}$	$A^t_{(i)}$

Case ADV.1. A^t contains a packet in slot t. Let $j = A^t[t]$. We will now remove j from the adversary stash $\mathcal{A}$ and, since by invariant (InvA) we have $j \in P^t$, we need to add packet j to the backup plan $\mathcal{B}$ to maintain invariant (InvB)(ii), which in turn requires removing a packet from $\mathcal{B}$ to preserve its feasibility. To this end, we apply Lemma 5.5 (with $g = j$), which implies that there is a packet $f_j \in B^t \setminus P^t$ such that $d_{f_j} > \mathsf{prevts}(P^t, d_j)$, $w_{f_j} \leq w_j$, and for which set $B^t \setminus \{f_j\} \cup \{j\}$ is feasible.

We thus set $A^t_{\mathrm{adv}} = A^t \setminus \{j\}$ and $B^t_{\mathrm{adv}} = B^t \setminus \{f_j\} \cup \{j\}$. Invariant (InvA) is clearly preserved, and by Lemma 5.5, invariant (InvB) continues to hold as well. The potential change is

$$(5.13) \qquad \Delta_{\mathrm{adv}}\Psi \;=\; \frac{1}{\phi}\left(w(B^t_{\mathrm{adv}}) - w(B^t)\right) \;=\; \frac{1}{\phi}\left(-w_{f_j} + w_j\right).$$

From the definition of $\mathsf{sub}^t(j)$, $d_{f_j} > \mathsf{prevts}(P^t, d_j)$, and $f_j \notin P^t$, we have that f_j is a candidate for $\mathsf{sub}^t(j)$, and thus $w(\mathsf{sub}^t(j)) \geq w_{f_j}$. Using (5.13) and $\mathsf{advcredit}^t_t = w_j$, it follows that

$$\phi\left(\Delta_{\mathrm{adv}}\Psi - \mathsf{advcredit}^t_t\right) \;=\; -w_{f_j} + w_j - \phi\, w_j$$
$$= -\frac{1}{\phi}\, w_j - w_{f_j}$$
$$\geq -\frac{1}{\phi}\, w_j - w(\mathsf{sub}^t(j))$$
$$(5.14) \qquad\qquad \geq -\frac{1}{\phi}\, w_p - w(\mathsf{sub}^t(p)),$$

where inequality (5.14) follows from the choice of p in line 1 of the algorithm's description, using also that $j \in P^t$. This implies (5.12).

Case ADV.2. Slot t is empty in A^t. In this case, we do not change $\mathcal{A}$ and $\mathcal{B}$, so $A_{\text{adv}}^t = A^t$ and $B_{\text{adv}}^t = B^t$. Invariants (InvA) and (InvB) are trivially preserved. Recall that $\text{advcredit}_t^t = \text{minwt}(P^t, t) = w_\ell$, where ℓ denotes the lightest packet in the initial segment of P^t. Note that $\text{sub}^t(\ell) = \ell$ and that $\Delta_{\text{adv}}\Psi = 0$. Then

$$\phi\left(\Delta_{\text{adv}}\Psi - \text{advcredit}_t^t\right) = -\phi\, w_\ell$$
$$= -\frac{1}{\phi}\, w_\ell - w(\text{sub}^t(\ell))$$
$$(5.15) \qquad\qquad \geq -\frac{1}{\phi}\, w_p - w(\text{sub}^t(p)),$$

where inequality (5.15) follows from the choice of p again. Thus (5.12) holds.

This concludes the analysis of the adversary step. In particular, we have determined the snapshots A_{adv}^t and B_{adv}^t, of the adversary stash $\mathcal{A}$ and the backup plan $\mathcal{B}$, respectively, resulting from the adversary step. Next, in the following sections, we analyze the algorithm's step.

5.5.2. Ordinary step. We now assume that the adversary step, as described in the previous section, has already been implemented. The current optimal plan P^t remains unchanged in the adversary step, and the current snapshots of $\mathcal{A}$ and $\mathcal{B}$ are A_{adv}^t and B_{adv}^t. In this section we analyze the algorithm's move at step t, assuming this is an ordinary step, as defined in section 4.

In an ordinary step a packet $p \in P^t[t, \alpha]$ is transmitted, where $\alpha = \text{nextts}^t(t)$ is the first tight slot in P^t. The algorithm makes no changes in packet weights and deadlines, so $\Delta^t w = 0$. Thus, to simplify notation, for any packet q we can write $w_q = w_q^t$ and $d_q = d_q^t$, omitting the superscript t. As usual, ℓ denotes the lightest packet in the initial segment $P^t[t, \alpha]$.

By the algorithm, p is the heaviest packet in $P^t[t, \alpha]$. Since $\text{sub}^t(p) = \ell$, inequality (5.12) gives us that

$$(5.16) \qquad\qquad \Delta_{\text{adv}}\Psi - \text{advcredit}_t^t \geq -\frac{1}{\phi^2}\, w_p - \frac{1}{\phi}\, w_\ell.$$

According to Lemma A.4, the new optimal plan (starting at time slot $t+1$) is $\widetilde{P}^{t+1} = P^t \setminus \{p\}$.

We have two cases, depending on whether or not the adversary stash contains a packet in the initial segment $[t, \alpha]$ of P^t.

Case O.1. $A_{\text{adv}}^t[t, \alpha] = \emptyset$. In this case, $p \notin A_{\text{adv}}^t$ (as $p \in P^t[t, \alpha]$) and we do not further change set $\mathcal{A}$, i.e., $\widetilde{A}^{t+1} = A_{\text{adv}}^t$. So invariant (InvA) is preserved and $\text{advcredit}^t = \text{advcredit}_t^t$.

We now show that we can preserve invariant (InvB). As $p \notin A_{\text{adv}}^t$ and (InvB)(ii), we have that $p \in B_{\text{adv}}^t$. We modify the backup plan $\mathcal{B}$ by removing p; that is, $\widetilde{B}^{t+1} = B_{\text{adv}}^t \setminus \{p\}$. This immediately gives us that invariant (InvB)(ii) continues to hold.

Next, we show that invariant (InvB)(i) holds as well. Since $\widetilde{B}^{t+1}$ is with respect to time $t+1$ and B_{adv}^t is feasible, we have that $\text{pslack}(\widetilde{B}^{t+1}, \tau) = \text{pslack}(B_{\text{adv}}^t, \tau) \geq 0$ for $\tau \geq d_p$. So it remains to consider slots $\tau \in [t+1, d_p)$. The case condition and invariant (InvB)(ii) imply that $P^t[t, \alpha] \subseteq B_{\text{adv}}^t$. This, together with the feasibility of B_{adv}^t and α being a tight slot in P^t, gives us that in fact $P^t[t, \alpha] = B_{\text{adv}}^t[t, \alpha]$. From this and $\text{pslack}(P^t, \tau) \geq 1$ for $\tau \in [t+1, d_p)$ (as $\alpha \geq d_p$ is the first tight slot), we get

that $\mathsf{pslack}(\widetilde{B}^{t+1}, \tau) = \mathsf{pslack}(B_{\mathrm{adv}}^t, \tau) - 1 = \mathsf{pslack}(P^t, \tau) - 1 \geq 0$ for all $\tau \in [t+1, d_p)$, completing the proof that invariant (InvB)(i) continues to hold.

The calculation showing the packet-transmission inequality (5.6) is quite simple. Taking into account that the adversary credit is $\mathsf{advcredit}^t = \mathsf{advcredit}_t^t$, and that the change of the potential in the algorithm's step is $\Delta_{\mathrm{ALG}} \Psi = \frac{1}{\phi}(w(\widetilde{B}^{t+1}) - w(B_{\mathrm{adv}}^t)) = -\frac{1}{\phi} w_p$, we obtain

$$\phi\, w^t(\mathsf{ALG}[t]) - \phi\,(\Delta^t w) + (\widetilde{\Psi}^{t+1} - \Psi^t) - \mathsf{advcredit}^t$$

$$= \phi\, w_p - \phi \cdot 0 + [\,\Delta_{\mathrm{ALG}}\Psi + \Delta_{\mathrm{adv}}\Psi\,] - \mathsf{advcredit}_t^t$$

$$= \phi\, w_p + \Delta_{\mathrm{ALG}}\Psi + [\,\Delta_{\mathrm{adv}}\Psi - \mathsf{advcredit}_t^t\,]$$

$$(5.17) \qquad\qquad \geq \phi\, w_p - \frac{1}{\phi}\, w_p + \left[-\frac{1}{\phi^2}\, w_p - \frac{1}{\phi}\, w_\ell \right]$$

$$(5.18) \qquad\qquad = \frac{1}{\phi}\, w_p - \frac{1}{\phi}\, w_\ell \geq 0\,,$$

where inequality (5.17) uses (5.16), and the inequality in the last step in line (5.18) uses the definition of ℓ, namely that $w_p \geq w_\ell$.

Case O.2. $A_{\mathrm{adv}}^t[t, \alpha] \neq \emptyset$. (This includes the case when $p \in A_{\mathrm{adv}}^t$.) We first describe our modifications of sets $\mathcal{A}$ and $\mathcal{B}$, and then argue that with these modifications our invariants are preserved and inequality (5.6) is satisfied.

Changing sets $\mathcal{A}$ and $\mathcal{B}$. Let g^* be the latest-deadline packet in $A_{\mathrm{adv}}^t[t, \alpha]$. Packet g^* exists, by the case condition. Note that $d_{g^*} \geq t+1$, because A_{adv}^t cannot contain a packet with deadline t (such a packet would be removed from $\mathcal{A}$ when we analyze the adversary step). Let f^* be the earliest-deadline packet in $B_{\mathrm{adv}}^t \setminus P^t$. Observation 5.2(a) with $\eta = t - 1$ implies that f^* is well defined. Note that possibly $d_{f^*} = t$, in which case f^* cannot remain in $\mathcal{B}$ in the next step.

If $p \in A_{\mathrm{adv}}^t$, let $g = p$; otherwise let $g = g^*$. (In either case we have $g \in A_{\mathrm{adv}}^t$.) We remove g from $\mathcal{A}$; i.e., we set $\widetilde{A}^{t+1} = A_{\mathrm{adv}}^t \setminus \{g\}$. If we have $g = g^* \neq p$, then $p \in B_{\mathrm{adv}}^t$, so p will need to be removed from B_{adv}^t, and due to removing g from $\mathcal{A}$ we will also need to add it to $\mathcal{B}$ to satisfy invariant (InvB)(ii). In either case, we remove f^* from $\mathcal{B}$. Thus the new backup plan will be

$$\widetilde{B}^{t+1} = \left\{ \begin{array}{ll} B_{\mathrm{adv}}^t \setminus \{f^*\} & \text{if } g = p, \\ B_{\mathrm{adv}}^t \cup \{g^*\} \setminus \{f^*, p\} & \text{if } g = g^* \neq p. \end{array} \right.$$

Note that in both cases it holds that $\widetilde{B}^{t+1} = B_{\mathrm{adv}}^t \cup \{g\} \setminus \{f^*, p\}$.

For a warm-up, before proving that our invariants hold, let's verify that all packets in $\widetilde{B}^{t+1}$ have deadlines at least $t+1$. Indeed, as already mentioned earlier, we have $d_{g^*} \geq t+1$, and if some $q \in \widetilde{B}^{t+1} \setminus \{g^*\}$ had $d_q = t$, then, by the definitions of f^* and $\widetilde{B}^{t+1}$, this q would also be in P^t, but this implies that $\alpha = t$ and $P^t[t, t] = \{q\}$, contradicting the case condition because $q \notin A_{\mathrm{adv}}^t$, by invariant (InvB)(ii).

Preserving the invariants. We now have $p \notin \widetilde{A}^{t+1}$, so invariant (InvA) is preserved. We next show that invariant (InvB) holds after the step. For part (InvB)(ii), we need to show that $\widetilde{P}^{t+1} \cap \widetilde{B}^{t+1} = \widetilde{P}^{t+1} \setminus \widetilde{A}^{t+1}$. This can be verified quite easily by considering how $\mathcal{P} \cap \mathcal{B}$ and $\mathcal{P} \setminus \mathcal{A}$ change: If $g = p$, then both sets do not change, and if $g = g^* \neq p$, then p is replaced by g^* in both sets.

To streamline the argument for part (InvB)(i) (feasibility), we divide the process of updating and analyzing $\mathcal{B}$ into two parts: first we analyze the effects of replacing

f^* by g in $\mathcal{B}$, and then we show that we can remove p from $\mathcal{B}$ (and increment t), preserving its feasibility.

So in the first part we consider the auxiliary set $B^t_{(i)} = B^t_{\mathrm{adv}} \cup \{g\} \setminus \{f^*\}$. Directly from Corollary 5.6, we obtain that $B^t_{(i)}$ is a feasible set of packets at time t. Further, we also show that it has the following property.

CLAIM 5.8. $\mathsf{pslack}(B^t_{(i)}, \tau) \geq 1$ for any $\tau \in [t, d_p)$.

Proof. We first observe that for $\tau \in [t, d_{f^*})$, by invoking Lemma 5.4(a) with $\zeta = t - 1$, $B = B^t_{\mathrm{adv}}$, $A = A^t_{\mathrm{adv}}$, and with the current plan P^t, we can obtain that

$$(5.19) \qquad \mathsf{pslack}(B^t_{\mathrm{adv}}, \tau) \ \geq \ \mathsf{pslack}(P^t, \tau) + |A^t_{\mathrm{adv}}[t, \tau]| \,.$$

Now we consider three cases depending on the value of τ. The first case is for $\tau \in [t, \min(d_{f^*}, d_g))$. Then

$$\mathsf{pslack}(B^t_{(i)}, \tau) \ = \ \mathsf{pslack}(B^t_{\mathrm{adv}}, \tau) \ \geq \ \mathsf{pslack}(P^t, \tau) \geq 1 \,,$$

where the equality uses that $B^t_{(i)}[t, \tau] = B^t_{\mathrm{adv}}[t, \tau]$ (as $\tau < \min(d_{f^*}, d_g)$), the first inequality uses (5.19), and the last inequality follows from the fact that τ is not a tight slot of P^t, as $\tau \in [t, \alpha)$.

In the second case, for slots $\tau \in [d_g, \min(d_{f^*}, d_p))$, we note that $g \in A^t_{\mathrm{adv}}[t, \tau]$, so equation (5.19) and $\mathsf{pslack}(P^t, \tau) \geq 1$ imply

$$\mathsf{pslack}(B^t_{(i)}, \tau) \ = \ \mathsf{pslack}(B^t_{\mathrm{adv}}, \tau) - 1 \ \geq \ \mathsf{pslack}(P^t, \tau) + |A^t_{\mathrm{adv}}[t, \tau]| - 1 \ \geq \ 1 \,.$$

In the third case, we deal with $\tau \in [d_{f^*}, d_p)$. For $\tau \in [d_{f^*}, d_g)$, replacing f^* by g increases the value of $\mathsf{pslack}(\mathcal{B}, \tau)$ by 1, implying $\mathsf{pslack}(B^t_{(i)}, \tau) = \mathsf{pslack}(B^t_{\mathrm{adv}}, \tau) + 1 \geq 1$, as B^t_{adv} is feasible. In particular, we are done if $g = p$. It thus remains to consider the subcase when $g = g^* \neq p$ and $\tau \in [\max(d_{f^*}, d_{g^*}), d_p)$, for which we use Lemma 5.4(b) with $\zeta = \alpha$, $B = B^t_{\mathrm{adv}}$, $A = A^t_{\mathrm{adv}}$, and the current plan P^t, to get

$$\mathsf{pslack}(B^t_{(i)}, \tau) \ = \ \mathsf{pslack}(B^t_{\mathrm{adv}}, \tau) \ \geq \ \mathsf{pslack}(P^t, \tau) \ \geq \ 1 \,,$$

where the equality uses that $B^t_{(i)}[t, \tau] = B^t_{\mathrm{adv}}[t, \tau] \cup \{g\} \setminus \{f^*\}$ (as $\tau \geq \max(d_{f^*}, d_g)$), and the last inequality follows from the fact that τ is not a tight slot of P^t, as $\tau \in [t, \alpha)$.   □

Next, note that we can express $\widetilde{B}^{t+1}$ as $\widetilde{B}^{t+1} = B^t_{(i)} \setminus \{p\}$—indeed, this works no matter whether $g = p$ or $g = g^*$. It remains to show that removing p from $B^t_{(i)}$ and incrementing the current time to $t + 1$ preserves feasibility. These changes decrease the value of $\mathsf{pslack}(\mathcal{B}, \tau)$ for $\tau \in [t + 1, d_p)$, but Claim 5.8 ensures that we still have $\mathsf{pslack}(\widetilde{B}^{t+1}, \tau) \geq 0$ for such τ. For $\tau \geq d_p$, the value of $\mathsf{pslack}(\mathcal{B}, \tau)$ is not affected. Hence, invariant (InvB)(i) holds after the step.

Deriving inequality (5.6). Let $\Delta_{\mathrm{ALG}}\Psi$ be the change of Ψ in the algorithm's step, so $\Delta_{\mathrm{ALG}}\Psi = \frac{1}{\phi}(w(\widetilde{B}^{t+1}) - w(B^t_{\mathrm{adv}})) = \frac{1}{\phi}(-w_p - w_{f^*} + w_g)$. The adversary adjustment credit is $\mathsf{advcredit}^t_{(t, \alpha]} = w_g - \mathsf{minwt}(P^t, d^t_g) = w_g - w_\ell$, since we removed g from $\mathcal{A}$ and $\mathsf{minwt}(P^t, d^t_g) = w_\ell \leq w_g$. Thus, the total adversary credit for step t is $\mathsf{advcredit}^t = \mathsf{advcredit}^t_t + \mathsf{advcredit}^t_{(t, \alpha]} = \mathsf{advcredit}^t_t + w_g - w_\ell$. (See section 5.1 for the definition of $\mathsf{advcredit}^t$.) Note that $w_{f^*} \leq w_\ell$ as $f^* \notin P^t$ and that $w_g \leq w_p$ as p

is the heaviest packet in $P^t[t, \alpha]$ and $g \in P^t[t, \alpha]$. We show the packet-transmission inequality (5.6) by summing these changes:

$$\phi\, w^t(\mathsf{ALG}[t]) - \phi\,(\Delta^t w) + (\widetilde{\Psi}^{t+1} - \Psi^t) - \mathsf{advcredit}^t$$

$$= \phi\, w_p - \phi \cdot 0 + [\,\Delta_{\mathrm{ALG}}\Psi + \Delta_{\mathrm{adv}}\Psi\,] - [\,\mathsf{advcredit}^t_t + w_g - w_\ell\,]$$

$$= \phi\, w_p + \Delta_{\mathrm{ALG}}\Psi - w_g + w_\ell + [\,\Delta_{\mathrm{adv}}\Psi - \mathsf{advcredit}^t_t\,]$$

$$(5.20) \qquad \geq \phi\, w_p + \frac{1}{\phi}\,[\,-w_p - w_{f^*} + w_g\,] - w_g + w_\ell + \left[\,-\frac{1}{\phi^2}\,w_p - \frac{1}{\phi}\,w_\ell\,\right]$$

$$= \frac{1}{\phi}\,w_p - \frac{1}{\phi}\,w_{f^*} - \frac{1}{\phi^2}\,w_g + \frac{1}{\phi^2}\,w_\ell$$

$$(5.21) \qquad \geq \frac{1}{\phi}\,w_p - \frac{1}{\phi}\,w_\ell - \frac{1}{\phi^2}\,w_p + \frac{1}{\phi^2}\,w_\ell$$

$$(5.22) \qquad = \frac{1}{\phi^3}\,(w_p - w_\ell) \geq 0\,,$$

where inequality (5.20) uses (5.16), inequality (5.21) holds because $w_{f^*} \leq w_\ell$ and $w_g \leq w_p$, and the last inequality (5.22) follows from $w_p \geq w_\ell$. This concludes the analysis of an ordinary step.

5.5.3. Leap step: A roadmap. In the remainder of the analysis, we focus on a leap step of Algorithm $\mathsf{PlanM}(t)$, when it transmits a packet $p \in P^t(\alpha, T]$, where, as defined earlier, P^t represents the current optimal plan, already updated to take into account packet arrivals. As in this case some packet weights change, we use notation w^t_a and w^{t+1}_a (or $w^t(a)$ and $w^{t+1}(a)$) for the weights of a packet a before and after p is transmitted, respectively. For packets whose weight does not change, we may occasionally omit the superscript t. We apply the same convention to deadlines.

By Lemma B.1(a), the new optimal plan (starting at time $t + 1$) is

$$\widetilde{P}^{t+1} = P^t \setminus \{p, \ell\} \cup \{\varrho\}\,,$$

where, as usual, $\varrho = \mathsf{sub}^t(p)$ and ℓ is the lightest packet in $P^t[t, \alpha]$.

All changes in the optimal plan resulting from transmitting p are within two intervals of the plan: the initial segment $[t, \alpha]$ and the interval $(\beta, \gamma]$, where $\beta = \mathsf{prevts}^t(d^t_p)$ and $\gamma = \mathsf{nextts}^t(d^t_\varrho)$ (possibly, $\alpha = \beta$). In segment $[t, \alpha]$, packet ℓ is ousted. In interval $(\beta, \gamma]$ (that could consist of several segments), packet p gets replaced by ϱ, and Algorithm PlanM increases the weight of ϱ to $\mu = \mathsf{minwt}(P^t, d^t_\varrho)$. Furthermore, if this is an iterated leap step (i.e., $k > 1$ in the algorithm), PlanM also modifies weights and deadlines of some packets h_i. These changes in the optimal plan may change some values of $\mathsf{pslack}(\tau)$ and may force changes in $\mathcal{A}$ or $\mathcal{B}$, which might in turn trigger additional changes in these sets to restore invariants (InvA) and (InvB).

We divide the analysis of a leap step into two *substeps*, namely (i) processing the initial segment $[t, \alpha]$ and (ii) processing interval $(\beta, \gamma]$. These are outlined later in this section and described in detail in sections 5.5.4, 5.5.5, and 5.5.6. Before proceeding to that, we introduce two key inequalities bounding the potential change in these substeps, and we show how to derive from them the packet-transmission inequality (5.6).

Two key inequalities. We now define several quantities that we will use in our estimates and in the two key inequalities below:

$\Delta_{[t,\alpha]}\Psi$: The change of the potential due to the modifications in $\mathcal{A}$ and $\mathcal{B}$ implemented when processing interval $[t,\alpha]$. These modifications include the removal of ℓ from $P^t[t,\alpha]$, as well as other modifications that are triggered by it.

$\Delta_{(\beta,\gamma]}\Psi$: The change of the potential due to the modifications in $\mathcal{A}$ and $\mathcal{B}$ implemented when processing interval $(\beta,\gamma]$. These modifications include the replacement of p by ϱ in $(\beta,\gamma]$, the increases of weights and possible decreases of deadlines of some packets, as well as other modifications that are triggered by these changes.

$\mathsf{advcredit}^t_{(\beta,\gamma]}$: The adversary adjustment credit for modifications of packets in $\mathcal{A}$ implemented when processing interval $(\beta,\gamma]$, namely for removing them or replacing them by lighter packets.

$\Delta^t w$: As defined earlier, this is the total adjustment of weights of packets in P^t that is done by the algorithm in step t (always strictly positive as the algorithm increases the weight of ϱ by a nonzero amount).

Note that we do not define the adversary adjustment credit for processing the initial segment $[t,\alpha]$, as in that substep of the analysis the adversary does not need to be compensated for any change in $\mathcal{A}$.

We will derive the proof of the packet-transmission inequality (5.6) from the two key inequalities below, which bound the changes of the potential resulting from processing intervals $[t,\alpha]$ and $(\beta,\gamma]$, respectively:

$$(5.23) \qquad\qquad\qquad\qquad \Delta_{[t,\alpha]}\Psi \geq -\frac{1}{\phi}\, w^t_\ell,$$

$$(5.24) \qquad\qquad \Delta_{(\beta,\gamma]}\Psi - \phi\left(\Delta^t w\right) - \mathsf{advcredit}^t_{(\beta,\gamma]} \geq -w^t_p + w^t_\varrho.$$

Deriving the packet-transmission inequality. We now derive (5.6) from (5.23) and (5.24). We start with two simple but useful bounds. First, using inequality (5.12) and the definition of ϱ, we have

$$(5.25) \qquad\qquad \Delta_{\mathrm{adv}}\Psi - \mathsf{advcredit}^t_t \;\geq\; -\frac{1}{\phi^2}\, w^t_p - \frac{1}{\phi}\, w^t_\varrho\,.$$

Also, for any $\tau \geq t$, we have

$$(5.26) \qquad\qquad \frac{1}{\phi^2}\, w^t_p + \frac{1}{\phi}\, w^t_\varrho \;\geq\; w^t_\ell \;\geq\; \mathsf{minwt}(P^t,\tau)\,,$$

where the first inequality follows from the choice of p in line 1 of the algorithm (specifically, because the algorithm chose p over ℓ and because $\ell = \mathsf{sub}(P^t,\ell)$), and the second one follows from $w^t_\ell = \mathsf{minwt}(P^t,t) \geq \mathsf{minwt}(P^t,\tau)$, that is, the monotonicity of $\mathsf{minwt}(P^t,\tau)$ with respect to τ.

So assume now that (5.23) and (5.24) hold. The total potential change in this step is

$$\widetilde{\Psi}^{t+1} - \Psi^t \;=\; \Delta_{\mathrm{adv}}\Psi + \Delta_{[t,\alpha]}\Psi + \Delta_{(\beta,\gamma]}\Psi\,,$$

because all changes in the optimal plan and in sets $\mathcal{A}$ and $\mathcal{B}$ are accounted for (uniquely) in the terms on the right-hand side. (The changes during the adversary step, accounted for in $\Delta_{\mathrm{adv}}\Psi$, were discussed in section 5.5.1. The changes associated with the algorithm's step, which contribute to $\Delta_{[t,\alpha]}\Psi$ and $\Delta_{(\beta,\gamma]}\Psi$, will be detailed in sections 5.5.4–5.5.6.) The total adversary credit is the sum of the transmission credit,

which was analyzed in section 5.5.1, and the adjustment credits for decreasing some packet weights in $\mathcal{A}$, so

$$\mathsf{advcredit}^t \;=\; \mathsf{advcredit}^t_t + \mathsf{advcredit}^t_{(\beta,\gamma]}\,,$$

as the changes in $[t,\alpha]$ will not involve any weight decreases in the adversary stash $\mathcal{A}$. Combining it all together, we have

$$\phi\, w^t(\mathsf{ALG}[t]) - \phi\,(\Delta^t w) + (\widetilde{\Psi}^{t+1} - \Psi^t) - \mathsf{advcredit}^t$$

$$= \phi\, w^t_p - \phi\,(\Delta^t w) + [\,\Delta_{\mathrm{adv}}\Psi + \Delta_{[t,\alpha]}\Psi + \Delta_{(\beta,\gamma)}\Psi\,] - [\,\mathsf{advcredit}^t_t + \mathsf{advcredit}^t_{(\beta,\gamma]}\,]$$

$$= \phi\, w^t_p + [\,\Delta_{\mathrm{adv}}\Psi - \mathsf{advcredit}^t_t\,] + \Delta_{[t,\alpha]}\Psi + [\,\Delta_{(\beta,\gamma)}\Psi - \phi\,(\Delta^t w) - \mathsf{advcredit}^t_{(\beta,\gamma]}\,]$$

(5.27)

$$\geq \phi\, w^t_p + \left[\,-\frac{1}{\phi^2}\, w^t_p - \frac{1}{\phi}\, w^t_\varrho\,\right] + \left[\,-\frac{1}{\phi}\, w^t_\ell\,\right] + \left[\,-w^t_p + w^t_\varrho\,\right]$$

(5.28)

$$= \frac{1}{\phi^3}\, w^t_p + \frac{1}{\phi^2}\, w^t_\varrho - \frac{1}{\phi}\, w^t_\ell \;\geq\; 0\,,$$

where in inequality (5.27) we use, in this order, inequalities (5.25), (5.23), and (5.24), and in line (5.28) the equality is obtained by repeatedly applying the definition of ϕ, while the last inequality follows from (5.26).

Therefore, to complete the analysis, it is now sufficient to show that the two key inequalities (5.23) and (5.24) hold, and that invariants (InvA) and (InvB) are preserved after the step. As mentioned above, we divide the proof into two substeps, more precisely defined as follows:

Processing the initial segment $[t,\alpha]$. In this substep, presented in section 5.5.4 below, we assume that the changes described in the adversary step have already been implemented. We describe changes in $\mathcal{A}$ and $\mathcal{B}$ triggered by the removal of ℓ from the optimal plan and by incrementing the time to $t+1$. We show that inequality (5.23) holds and that invariants (InvA) and (InvB) are preserved after these changes. More precisely, let $P^{t+1}_{\mathrm{ins}} = P^t \setminus \{\ell\}$ be plan $\mathcal{P}$ after removing ℓ and incrementing the current time to $t+1$, and let A^{t+1}_{ins} and B^{t+1}_{ins} be the snapshots of sets $\mathcal{A}$ and $\mathcal{B}$ after implementing necessary changes resulting from modifying $\mathcal{P}$. We prove that invariants (InvA) and (InvB) hold for plan P^{t+1}_{ins} and sets A^{t+1}_{ins}, B^{t+1}_{ins}. (As indicated by superscript $t+1$, these sets are defined with respect to time $t+1$.)

Processing interval $(\beta,\gamma]$. In this substep, we assume that the changes described in the adversary step and in the processing of $[t,\alpha]$ have already been implemented. We describe changes in $\mathcal{A}$ and $\mathcal{B}$ triggered by the replacement of p by ϱ and, for an iterated leap step, by modifications of packets h_i. We prove that inequality (5.24) holds and that invariants (InvA) and (InvB) are preserved after these changes. In other words, we show that both invariants hold after the transmission event, i.e., for plan $\widetilde{P}^{t+1}$ and sets $\widetilde{A}^{t+1}$ and $\widetilde{B}^{t+1}$. The proof will be divided into two cases, depending on whether it is a simple or an iterated leap step (see sections 5.5.5 and 5.5.6, respectively). The proof for an iterated leap step is further divided into a number of smaller stages. Each such stage consists of modifications of $\mathcal{P}$, $\mathcal{B}$, and $\mathcal{A}$ that involve packets with deadlines in some time interval $(\zeta,\eta] \subseteq (\beta,\gamma]$, where ζ and η are tight slots of $\mathcal{P}$.

Throughout the process, while we gradually make changes in sets $\mathcal{P}$, $\mathcal{B}$, and $\mathcal{A}$, we will ensure that the following two properties hold:

Locality of changes. For a substep or stage in which an interval $(\zeta, \eta]$ of slots is processed, let $\dot{P}$ be the snapshot of $\mathcal{P}$ just before processing interval $(\zeta, \eta]$ and let $\ddot{P}$ be the optimal plan after this subset or stage; ζ and η will always be tight slots of $\dot{P}$. We will maintain the property that the changes of $\mathcal{P}$ are confined to the set $\mathcal{P}(\zeta, \eta]$ of packets, or more precisely,

$$\tag{5.29} \dot{P} \setminus \dot{P}(\zeta, \eta] = \ddot{P} \setminus \ddot{P}(\zeta, \eta].$$

We also do not change the deadlines and weights of packets in $\dot{P} \setminus \dot{P}(\zeta, \eta]$. Equation (5.29) implies that $|\dot{P}(\zeta, \eta]| = |\ddot{P}(\zeta, \eta]|$, since all plans are full. Therefore, η and ζ are tight slots of $\ddot{P}$ and all values of $\mathsf{pslack}(\mathcal{P}, \tau)$ remain unaffected for $\tau \notin (\zeta, \eta)$. We remark that substeps and stages in the analysis of each case will have pairwise disjoint intervals $(\zeta, \eta]$.

Monotonicity of $\mathcal{B} \setminus \mathcal{P}$. No packets will be added to $\mathcal{B}(\alpha, T] \setminus \mathcal{P}$ throughout the analysis. (In fact, the only packet that may be added to $\mathcal{B}$ is ℓ. This could happen when segment $[t, \alpha]$ is processed.)

We remark that some "intermediate snapshots" of $\mathcal{P}$ in the analysis of a leap step, such as P_{ins}^{t+1}, mentioned above, are only auxiliary concepts used to capture the changes of $\mathcal{P}$ after each substep or stage of the analysis. These snapshots satisfy the definition of plans and are full, but they may not be optimal for the current set of pending packets. As we do not rely on these intermediate plans being optimal, this presents no issue for the analysis.

5.5.4. Leap step: Processing the initial segment. We now focus on the first substep of the analysis of the leap step, namely on processing segment $[t, \alpha]$ of the optimal plan $\mathcal{P}$. As explained earlier in section 5.5.3, we assume that the changes in sets $\mathcal{A}$ and $\mathcal{B}$ described in the adversary step (section 5.5.1) have already been implemented. Recall that A_{adv}^t and B_{adv}^t represent the snapshots of sets $\mathcal{A}$ and $\mathcal{B}$, respectively, resulting from the adversary step. As the adversary step does not affect the optimal plan, the current plan is still P^t.

In this substep, we advance the time to $t+1$ and remove ℓ from $\mathcal{P}$; that is, plan P^t is changed to $P_{\mathrm{ins}}^{t+1} = P^t \setminus \{\ell\}$. P_{ins}^{t+1} is a plan for the set of packets pending at time $t+1$, that consists of the packets that were pending at time t, but excluding ℓ and packets that expired at time t. Since $d_\ell \leq \alpha = \mathsf{nextts}(P^t, t)$, the values of $\mathsf{pslack}(\mathcal{P}, \tau)$ may only change for $\tau < \alpha$, i.e., $\mathsf{pslack}(P_{\mathrm{ins}}^{t+1}, \tau) = \mathsf{pslack}(P^t, \tau)$ for $\tau \geq \alpha$. We may also need to make changes in sets $\mathcal{A}$ and $\mathcal{B}$, in order to preserve the invariants. We refer to this substep as "processing the initial segment", although some modifications of $\mathcal{B}$ may involve pending packets with deadlines after α. We divide this substep into two parts:

(i) first we show that if $\ell \in A_{\mathrm{adv}}^t$ then we can remove it from $\mathcal{A}$, preserving the invariants and not decreasing the potential, and then

(ii) assuming that $\ell \notin \mathcal{A}$, we analyze the effect of removing ℓ from $\mathcal{P}$ and incrementing the time.

Let $A_{(i)}^t$ denote the intermediate adversary stash, after the change in (i) and before the change in (ii), where we let $A_{(i)}^t = A_{\mathrm{adv}}^t$ if $\ell \notin A_{\mathrm{adv}}^t$, that is when change (i) does not apply. We adopt the same notation for set $\mathcal{B}$. Note that $\mathcal{P}$ does not change in (i), i.e., $P_{(i)}^t = P^t$.

(i) *Dealing with the case $\ell \in A_{\mathrm{adv}}^t$.* We now consider the case when $\ell \in A_{\mathrm{adv}}^t$. Since $\ell \notin P_{\mathrm{ins}}^{t+1}$, we need to remove ℓ from $\mathcal{A}$ to preserve invariant (InvA). In order to satisfy invariant (InvB), we are then forced to add ℓ to $\mathcal{B}$, which in turn requires us to remove some packet from $\mathcal{B} \setminus \mathcal{P}$ to restore feasibility of $\mathcal{B}$. To identify this packet we use Lemma 5.5 (with $g = \ell$), which shows that there is a packet $f_\ell \in B_{\mathrm{adv}}^t \setminus P^t$ for which the set $B_{\mathrm{adv}}^t \setminus \{f_\ell\} \cup \{\ell\}$ is feasible and satisfies $w(B_{\mathrm{adv}}^t \setminus \{f_\ell\} \cup \{\ell\}) \geq w(B_{\mathrm{adv}}^t)$.

We thus set $A_{(i)}^t = A_{\mathrm{adv}}^t \setminus \{\ell\}$ and $B_{(i)}^t = B_{\mathrm{adv}}^t \setminus \{f_\ell\} \cup \{\ell\}$, and this preserves both invariants (InvA) and (InvB). Note that the adversary does not get any adjustment credit for the removal of ℓ, as $w_\ell = \mathsf{minwt}(P^t, d_\ell)$, by the definition of ℓ. Further, since the contribution of this change of $\mathcal{B}$ to $\Delta_{[t,\alpha]}\Psi$ is positive, when estimating $\Delta_{[t,\alpha]}\Psi$ we will account for this contribution without an explicit reference.

(ii) *Removing ℓ from $\mathcal{P}$ and incrementing the time.* By (i) we have $\ell \notin A_{(i)}^t$, and thus also $\ell \in B_{(i)}^t$, by invariant (InvB)(ii). Now, the plan is changed from P^t to $P_{\mathrm{ins}}^{t+1} = P^t \setminus \{\ell\}$. As we do not further change $\mathcal{A}$, i.e., $A_{\mathrm{ins}}^{t+1} = A_{(i)}^t$, invariant (InvA) holds. (Note that after the adversary step, analyzed in section 5.5.1, $A_{(i)}^t$ cannot contain a packet that expires at time t.) However, we need to preserve invariant (InvB). Denoting by f_1 the earliest-deadline packet in $B_{(i)}^t \setminus P^t$, we consider two cases.

Case L.InSeg.1. $d_{f_1} \geq d_\ell$. We let $B_{\mathrm{ins}}^{t+1} = B_{(i)}^t \setminus \{\ell\}$. Inequality (5.23) holds as $\Delta_{[t,\alpha]}\Psi \geq -\frac{1}{\phi} w_\ell^t$. (This inequality could be strict, due to the contribution from replacing f_ℓ by ℓ, as discussed in part (i) above.) Invariant (InvB)(ii) holds because $\ell \notin P_{\mathrm{ins}}^{t+1}$ and $\mathcal{A}$ does not change in (ii). To show that (InvB)(i), i.e., the feasibility of $\mathcal{B}$, is preserved, note that $\mathsf{pslack}(B_{\mathrm{ins}}^{t+1}, \tau) = \mathsf{pslack}(B_{(i)}^t, \tau) \geq 0$ for $\tau \geq d_\ell$. Furthermore, $\mathsf{pslack}(B_{(i)}^t, \tau) \geq \mathsf{pslack}(P^t, \tau) \geq 1$ for $\tau \in [t, d_\ell)$, where the first inequality follows from the definition of f_1, $d_{f_1} \geq d_\ell > \tau$, and Lemma 5.4(a) with $\zeta = t - 1$, and the second inequality follows from the fact that τ is before the first tight slot α of P^t. Hence, for $\tau \in [t+1, d_\ell)$ we have $\mathsf{pslack}(B_{\mathrm{ins}}^{t+1}, \tau) = \mathsf{pslack}(B_{(i)}^t, \tau) - 1 \geq 0$, concluding the proof that invariant (InvB) is maintained.

Case L.InSeg.2. $d_{f_1} < d_\ell$. In this case, we let $B_{\mathrm{ins}}^{t+1} = B_{(i)}^t \setminus \{f_1\}$. Then $\Delta_{[t,\alpha]}\Psi \geq -\frac{1}{\phi} w_{f_1}^t \geq -\frac{1}{\phi} w_\ell^t$, where the second inequality follows from $\ell \in P^t[t,\alpha]$ and $f_1 \notin P^t$. Thus inequality (5.23) holds. Regarding invariant (InvB), its part (ii) follows from the changes of sets $\mathcal{A}$ and $\mathcal{B}$. To show invariant (InvB)(i), note that $\mathsf{pslack}(B_{\mathrm{ins}}^{t+1}, \tau) = \mathsf{pslack}(B_{(i)}^t, \tau) \geq 0$ for $\tau \geq d_{f_1}$. For $\tau \in [t, d_{f_1})$, we first get $\mathsf{pslack}(B_{(i)}^t, \tau) \geq \mathsf{pslack}(P^t, \tau) \geq 1$ by Lemma 5.4(a) with $\zeta = t - 1$, using the fact that τ is before the first tight slot α of P^t, as $\tau < d_{f_1} < d_\ell \leq \alpha$ (here, we use the case condition in the second inequality). Then, for $\tau \in [t+1, d_{f_1})$, we have $\mathsf{pslack}(B_{\mathrm{ins}}^{t+1}, \tau) \geq \mathsf{pslack}(B_{(i)}^t, \tau) - 1 \geq 0$. Thus, invariant (InvB) is preserved after processing the initial segment $[t, \alpha]$.

5.5.5. Processing $(\beta, \gamma]$ in a simple leap step. We now address the second substep of analyzing a leap step, namely processing the interval $(\beta, \gamma]$, focusing on the case of a simple leap step. (The case of an iterated leap step will be handled separately, in section 5.5.6.) Recall that the algorithm replaces p by ϱ in $\mathcal{P}$, and that in a simple leap step we have $k = 0$, that is $(\beta, \gamma]$ is a segment of P^t and $d_p, d_\varrho \in (\beta, \gamma]$ (see section 4).

We assume that the changes in $\mathcal{P}$ and sets $\mathcal{A}$ and $\mathcal{B}$ described in sections 5.5.1 and 5.5.4 have already been implemented. (We note that these changes might have involved some packets considered in this section; for example, it is possible that ϱ is the same as packet f_j removed from $\mathcal{B}$ when processing the adversary step in section

5.5.1.) In particular, the current time is already advanced to $t + 1$, and we assume that both invariants (InvA) and (InvB) hold for sets A_{ins}^{t+1}, B_{ins}^{t+1}, and P_{ins}^{t+1}, all three defined with respect to time $t + 1$. It now remains to describe how we process the interval $(\beta, \gamma]$. In this substep, the plan changes from P_{ins}^{t+1} to $\widetilde{P}^{t+1} = P_{\text{ins}}^{t+1} \setminus \{p\} \cup \{\varrho\}$ and the weight of ϱ is increased. The replacement of p by ϱ may trigger modifications in $\mathcal{A}$ and $\mathcal{B}$, in order to restore the invariants.

P_{ins}^{t+1} only differs from P^t in that in P_{ins}^{t+1} the current time is incremented to $t + 1$, packet ℓ is removed, and the interval $[t + 1, \alpha]$ of P^t may be split into multiple segments. Slot $\alpha = \text{nextts}(P^t, t)$ is also tight in P_{ins}^{t+1}, and there were no changes in $\mathcal{P}$ after slot α; that is, $P^t(\alpha, T] = P_{\text{ins}}^{t+1}(\alpha, T]$. Therefore, d_ϱ and d_p are still in the same segment $(\beta, \gamma]$ of P_{ins}^{t+1}, so $\text{prevts}(P_{\text{ins}}^{t+1}, d_\varrho) = \text{prevts}(P_{\text{ins}}^{t+1}, d_p) = \beta$ and $\text{nextts}(P_{\text{ins}}^{t+1}, d_\varrho) = \text{nextts}(P_{\text{ins}}^{t+1}, d_p) = \gamma$.

In order to build some intuition behind our modifications of $\mathcal{A}$ and $\mathcal{B}$, before we give a formal argument, let us consider a few illustrative scenarios, focusing on maintaining invariant (InvB). The simplest case is when $p \notin B_{\text{ins}}^{t+1}$ and $\varrho \in B_{\text{ins}}^{t+1}$, as then we can simply let $\widetilde{B}^{t+1} = B_{\text{ins}}^{t+1}$.

Suppose next that $p \in B_{\text{ins}}^{t+1}$ and $\varrho \notin B_{\text{ins}}^{t+1}$. In this case we are forced to remove p from $\mathcal{B}$, and to add ϱ to $\mathcal{B}$. If $d_\varrho \geq d_p$, then we don't need to make any more changes, that is we can let $\widetilde{B}^{t+1} = B_{\text{ins}}^{t+1} \setminus \{p\} \cup \{\varrho\}$, and $\widetilde{B}^{t+1}$ will be feasible. In fact, even when $d_\varrho < d_p$, using the properties of $\mathcal{B}$ established in section 5.2, we can take $\widetilde{B}^{t+1} = B_{\text{ins}}^{t+1} \setminus \{p\} \cup \{\varrho\}$ as long as $A_{\text{ins}}^{t+1}(\beta, \gamma] = \emptyset$.

In the remaining situations, some additional modifications of $\mathcal{B}$ are needed. In essence, what will happen is this: If $p \in B_{\text{ins}}^{t+1}$, we will replace it in $\mathcal{B}$ by the latest-deadline packet g^* in $A_{\text{ins}}^{t+1}[t + 1, \gamma]$, which will be removed from $\mathcal{A}$. Similarly, if $\varrho \notin B_{\text{ins}}^{t+1}$, then ϱ will replace in $\mathcal{B}$ the earliest-deadline packet f^* in $B_{\text{ins}}^{t+1}(\beta, T] \setminus P_{\text{ins}}^{t+1}$. Using the results from section 5.2, we show that such packets exist and that these changes will indeed preserve the feasibility of $\mathcal{B}$.

We now give a formal argument. The proof is organized into two subcases, depending on whether some changes in $\mathcal{A}$ are needed or not.

Case $\text{L}.(\beta, \gamma].\text{S}.1.$ $\varrho \notin B_{\text{ins}}^{t+1}$ and $A_{\text{ins}}^{t+1}(\beta, \gamma] = \emptyset$. (In particular, the case condition implies that $p \notin A_{\text{ins}}^{t+1}$, as $p \in P_{\text{ins}}^{t+1}(\beta, \gamma]$, and that $P_{\text{ins}}^{t+1}(\beta, \gamma] \subseteq B_{\text{ins}}^{t+1}$; it is possible though that this inclusion is strict.) Then we do not further change set $\mathcal{A}$, i.e., $\widetilde{A}^{t+1} = A_{\text{ins}}^{t+1}$. Since $p \notin \widetilde{A}^{t+1}$ and deadlines of all packets remain unchanged, invariant (InvA) holds.

Regarding $\mathcal{B}$, note that $\varrho \notin A_{\text{ins}}^{t+1}$ by invariant (InvA), and $p \in B_{\text{ins}}^{t+1}$, by invariant (InvB)(ii) and $p \notin A_{\text{ins}}^{t+1}$. Thus, we let $\widetilde{B}^{t+1} = B_{\text{ins}}^{t+1} \setminus \{p\} \cup \{\varrho\}$, which preserves invariant (InvB)(ii). Next, we need to show the feasibility of $\widetilde{B}^{t+1}$, i.e., invariant (InvB)(i). If $d_\varrho \geq d_p$, then replacing p by ϱ in $\mathcal{B}$ preserves feasibility, and thus, as B_{ins}^{t+1} is feasible, the new backup plan $\widetilde{B}^{t+1}$ is feasible as well. Otherwise, $d_\varrho < d_p$. For $\tau \notin [d_\varrho, d_p)$, it holds that $\text{pslack}(\widetilde{B}^{t+1}, \tau) = \text{pslack}(B_{\text{ins}}^{t+1}, \tau) \geq 0$, by the feasibility of B_{ins}^{t+1}. Next, consider some $\tau \in [d_\varrho, d_p) \subseteq (\beta, \gamma)$. By the case condition, $A_{\text{ins}}^{t+1}(\beta, \gamma] = \emptyset$; thus using Lemma 5.4(b) with $\zeta = \gamma$ we get that $\text{pslack}(B_{\text{ins}}^{t+1}, \tau) \geq \text{pslack}(P_{\text{ins}}^{t+1}, \tau) \geq 1$, where the second inequality holds because $\tau \in (\beta, \gamma)$ is not a tight slot in P_{ins}^{t+1}. Hence, $\text{pslack}(\widetilde{B}^{t+1}, \tau) = \text{pslack}(B_{\text{ins}}^{t+1}, \tau) - 1 \geq 0$ for $\tau \in [d_\varrho, d_p)$, and we can conclude that invariant (InvB) holds after the step.

Finally, we prove inequality (5.24). According to the algorithm, we have $w_\varrho^{t+1} = \text{minwt}(P^t, d_\varrho^t)$. Then the terms on the left-hand side of (5.24) are

$$\Delta_{(\beta,\gamma]}\Psi = \frac{1}{\phi}\left(-w_p^t + w_\varrho^{t+1}\right), \quad \Delta^t w = w_\varrho^{t+1} - w_\varrho^t, \quad \text{and} \quad \mathsf{advcredit}_{(\beta,\gamma]}^t = 0.$$

Plugging these in, we obtain

$$
\begin{aligned}
\Delta_{(\beta,\gamma]}\Psi - \phi\left(\Delta^t w\right) - \mathsf{advcredit}_{(\beta,\gamma]}^t &= \frac{1}{\phi}\left(-w_p^t + w_\varrho^{t+1}\right) - \phi\left(w_\varrho^{t+1} - w_\varrho^t\right) - 0 \\
&= -\frac{1}{\phi}\,w_p^t - w_\varrho^{t+1} + \phi\,w_\varrho^t \\
&\geq -\frac{1}{\phi}\,w_p^t - \left(\frac{1}{\phi^2}\,w_p^t + \frac{1}{\phi}\,w_\varrho^t\right) + \phi\,w_\varrho^t \\
&= -w_p^t + w_\varrho^t,
\end{aligned}
$$

(5.30)

where inequality (5.30) follows from $w_\varrho^{t+1} = \mathsf{minwt}(P^t, d_\varrho^t) \leq \frac{1}{\phi^2}w_p^t + \frac{1}{\phi}w_\varrho^t$, by (5.26). This shows that (5.24) holds.

Case L.$(\beta,\gamma]$.S.2. $\varrho \in B_{\mathrm{ins}}^{t+1}$ or $A_{\mathrm{ins}}^{t+1}(\beta,\gamma] \neq \emptyset$. (Thus, there is a packet in $(B_{\mathrm{ins}}^{t+1} \setminus P_{\mathrm{ins}}^{t+1}) \cup A_{\mathrm{ins}}^{t+1}$ which has deadline in $(\beta,\gamma]$.)
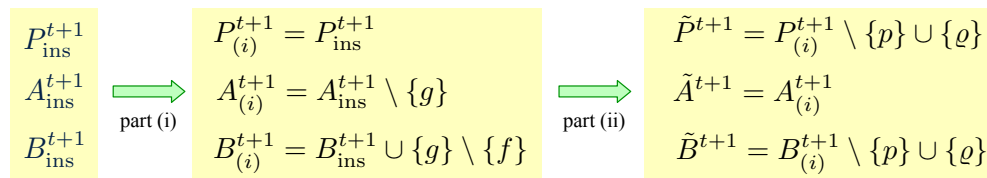
Changing sets $\mathcal{A}$ and $\mathcal{B}$. Let g^* be the latest-deadline packet in $A_{\mathrm{ins}}^{t+1}[t+1,\gamma]$. Packet g^* is well defined: this is trivially true if the second condition of the case is satisfied, and otherwise we have $\varrho \in B_{\mathrm{ins}}^{t+1} \setminus P_{\mathrm{ins}}^{t+1}$, in which case Observation 5.2(b) with $\eta = \gamma$ implies that $A_{\mathrm{ins}}^{t+1}[t+1,\gamma] \neq \emptyset$. (We remark that $d_{g^*} \leq \beta$ if $A_{\mathrm{ins}}^{t+1}(\beta,\gamma] = \emptyset$.) Similarly, let f^* be the earliest-deadline packet in $B_{\mathrm{ins}}^{t+1}(\beta,T] \setminus P_{\mathrm{ins}}^{t+1}$. We show that packet f^* is well defined. This is trivially true if $\varrho \in B_{\mathrm{ins}}^{t+1}$, while otherwise it holds that $A_{\mathrm{ins}}^{t+1}(\beta,\gamma] \neq \emptyset$, and then Observation 5.2(a) with $\eta = \beta$ gives us that $B_{\mathrm{ins}}^{t+1}(\beta,T] \setminus P_{\mathrm{ins}}^{t+1} \neq \emptyset$. (It is possible that $d_{f^*} > \gamma$ if $\varrho \notin B_{\mathrm{ins}}^{t+1}$.)

We now define packets g and f, and we modify $\mathcal{B}$ and $\mathcal{A}$ as follows. If $p \in A_{\mathrm{ins}}^{t+1}$, let $g = p$; otherwise let $g = g^*$. If $\varrho \in B_{\mathrm{ins}}^{t+1}$, let $f = \varrho$; otherwise let $f = f^*$. We remove g from $\mathcal{A}$; i.e., we set $\widetilde{A}^{t+1} = A_{\mathrm{ins}}^{t+1} \setminus \{g\}$. To preserve invariant (InvB)(ii), we must add ϱ to $\mathcal{B}$ (if it's not there already) because $\varrho \in \widetilde{P}^{t+1} \setminus \widetilde{A}^{t+1}$. Hence, we let $\widetilde{B}^{t+1} = B_{\mathrm{ins}}^{t+1} \cup \{g\} \setminus \{f,p\} \cup \{\varrho\}$; this definition applies no matter whether $g = p$ or $g = g^*$ nor whether $f = \varrho$ or $f = f^*$. (In particular, if $g = p$ and $f = \varrho$, then $\mathcal{B}$ will not change.)

Deriving inequality (5.24). We first claim that $w_f^t \leq w_\varrho^t$. As $f \in \{f^*, \varrho\}$, it is enough to prove that $w_{f^*}^t \leq w_\varrho^t$. We have $d_{f^*} > \beta \geq \alpha$, and, by the monotonicity property of $\mathcal{B} \setminus \mathcal{P}$ (see section 5.5.3), no packets were added to $\mathcal{B}(\alpha,T] \setminus \mathcal{P}$ when processing segment $[t,\alpha]$, implying that $f^* \in B^t(\beta,T] \setminus P^t$. Then the inequality $w_{f^*}^t \leq w_\varrho^t$ follows from the choice of $\varrho = \mathsf{sub}^t(p)$ as the heaviest pending packet not in P^t with deadline after slot β.

We also claim that $w_g^t \leq w_p^t$. As $g \in \{g^*, p\}$, it is enough to prove that $w_{g^*}^t \leq w_p^t$. Indeed, as we do not add any packet to $\mathcal{A}$ when processing segment $[t,\alpha]$, it holds that $g^* \in A^t$ and, thus, $g^* \in P^t$ by invariant (InvA). The choice of g^* implies that $\mathsf{prevts}(P^t, d_{g^*}) \leq \mathsf{prevts}(P^t, d_p^t)$, which in turn implies that $w(\mathsf{sub}(P^t, g^*)) \geq w_\varrho^t$. Therefore, if we had $w_{g^*}^t > w_p^t$ then the algorithm would transmit g^* instead of p. (Strictly speaking, here were rely on the "locality of changes" invariant from section 5.5.3, namely that $\mathcal{P}(\alpha,T]$ does not change when processing segment $[t,\alpha]$.)

According to the algorithm, $w_\varrho^{t+1} = \mathsf{minwt}(P^t, d_\varrho^t)$. Then the terms on the left-hand side of (5.24) are

$$P_{\text{ins}}^{t+1}$$
$$A_{\text{ins}}^{t+1}$$
$$B_{\text{ins}}^{t+1}$$

$\xrightarrow{\text{part (i)}}$

$$P_{(i)}^{t+1} = P_{\text{ins}}^{t+1}$$
$$A_{(i)}^{t+1} = A_{\text{ins}}^{t+1} \setminus \{g\}$$
$$B_{(i)}^{t+1} = B_{\text{ins}}^{t+1} \cup \{g\} \setminus \{f\}$$

$\xrightarrow{\text{part (ii)}}$

$$\tilde{P}^{t+1} = P_{(i)}^{t+1} \setminus \{p\} \cup \{\varrho\}$$
$$\tilde{A}^{t+1} = A_{(i)}^{t+1}$$
$$\tilde{B}^{t+1} = B_{(i)}^{t+1} \setminus \{p\} \cup \{\varrho\}$$

FIG. 5.3. *Changing $\mathcal{B}$ in two parts.*

$$\Delta_{(\beta,\gamma]}\Psi = \frac{1}{\phi}\left(w_g^t - w_p^t - w_f^t + w_\varrho^{t+1}\right),$$

$$\Delta^t w = w_\varrho^{t+1} - w_\varrho^t, \text{ and}$$

$$\mathsf{advcredit}_{(\beta,\gamma]}^t = w_g^t - \mathsf{minwt}(P^t, d_g^t)$$

(5.31)
$$\leq w_g^t - \mathsf{minwt}(P^t, d_\varrho^t) = w_g^t - w_\varrho^{t+1},$$

where the inequality in (5.31) follows from $\mathsf{nextts}(P^t, d_g^t) \leq \gamma = \mathsf{nextts}(P^t, d_\varrho^t)$. (For simplicity, the estimate for $\mathsf{advcredit}_{(\beta,\gamma]}^t$ is given above with respect to P^t, as described in section 5.1. To be completely formal, in this substep this credit would be with respect to the intermediate plan P_{ins}^{t+1}. However, using P^t instead of P_{ins}^{t+1} cannot make the adversary credit smaller since $\mathsf{minwt}(P_{\text{ins}}^{t+1}, d_g^t) \geq \mathsf{minwt}(P^t, d_g^t)$, which holds as $P_{\text{ins}}^{t+1} = P^t \setminus \{\ell\}$ and as any tight slot of P^t is also tight in P_{ins}^{t+1}.) Thus the changes described above give us that

$$\Delta_{(\beta,\gamma]}\Psi - \phi\left(\Delta^t w\right) - \mathsf{advcredit}_{(\beta,\gamma]}^t$$

$$\geq \frac{1}{\phi}\left(w_g^t - w_p^t - w_f^t + w_\varrho^{t+1}\right) - \phi\left(w_\varrho^{t+1} - w_\varrho^t\right) - \left(w_g^t - w_\varrho^{t+1}\right)$$

$$= -\left(\frac{1}{\phi^2}\,w_g^t + \frac{1}{\phi}\,w_p^t\right) + \left(-\frac{1}{\phi}\,w_f^t + \phi\,w_\varrho^t\right)$$

(5.32)
$$\geq -w_p^t + w_\varrho^t,$$

using bounds $w_f^t \leq w_\varrho^t$ and $w_g^t \leq w_p^t$ in step (5.32). This shows (5.24).

Preserving the invariants. Since after the changes it holds that $p \notin \tilde{A}^{t+1}$, invariant (InvA) is maintained. We now show that invariant (InvB) holds. That condition (InvB)(ii) is true follows directly from the way we update $\mathcal{A}$ and $\mathcal{B}$. Therefore, we will focus on condition (InvB)(i), that is, the feasibility of $\mathcal{B}$. If we have both $f = \varrho$ and $g = p$, then $\tilde{B}^{t+1} = B_{\text{ins}}^{t+1}$, and the feasibility of $\tilde{B}^{t+1}$ is trivial. Hence, from now on we assume that $f = f^* \neq \varrho$ or $g = g^* \neq p$ (or both).

We split the changes in $\mathcal{B}$ into two parts (see Figure 5.3):

   (i) First, we show that invariant (InvB) holds for the intermediate backup plan $B_{(i)}^{t+1} = B_{\text{ins}}^{t+1} \cup \{g\} \setminus \{f\}$, which we consider with respect to adversary stash $A_{(i)}^{t+1} = \tilde{A}^{t+1} = A_{\text{ins}}^{t+1} \setminus \{g\}$ and plan $P_{(i)}^{t+1} = P_{\text{ins}}^{t+1}$. In other words, we first implement the removal of g from $\mathcal{A}$ together with the corresponding changes in $\mathcal{B} \setminus \mathcal{P}$ but postpone the change in the plan.
   (ii) Second, we replace p by ϱ in $\mathcal{P}$, so $\tilde{P}^{t+1} = P_{(i)}^{t+1} \setminus \{p\} \cup \{\varrho\}$ and $\tilde{B}^{t+1} = B_{(i)}^{t+1} \setminus \{p\} \cup \{\varrho\}$, and we show that $\tilde{B}^{t+1}$ is feasible as well.

   CLAIM 5.9. *Assume that $f = f^* \neq \varrho$ or $g = g^* \neq p$. Set $B_{(i)}^{t+1} = B_{\text{ins}}^{t+1} \cup \{g\} \setminus \{f\}$ as a feasible set of packets with respect to time $t+1$.*

Proof. The claim clearly holds if $d_f \leq d_g$, so for the rest of the proof we assume that $d_g < d_f$. Recall that, according to the assumptions of our Case L.$(\beta, \gamma]$.S.2, we have that either $\varrho \in B_{\text{ins}}^{t+1}$ or $A_{\text{ins}}^{t+1}(\beta, \gamma] \neq \emptyset$ holds.

It is sufficient to show that $\mathsf{pslack}(B_{\text{ins}}^{t+1}, \tau) \geq 1$ for any $\tau \in [d_g, d_f)$, as this will imply that $d_f \leq \mathsf{nextts}(B_{\text{ins}}^{t+1}, d_g)$, which in turn implies the feasibility of $B_{(i)}^{t+1}$. We consider two cases.

First, assume that $\varrho \notin B_{\text{ins}}^{t+1}$. This implies $f = f^*$ and $A_{\text{ins}}^{t+1}(\beta, \gamma] \neq \emptyset$, which means that $d_g \in (\beta, \gamma]$, no matter whether $g = p$ or $g = g^*$. Then, for $\tau \in [d_g, d_f)$, Lemma 5.4(a) with $\zeta = \beta$ gives us

$$\mathsf{pslack}(B_{\text{ins}}^{t+1}, \tau) \geq \mathsf{pslack}(P_{\text{ins}}^{t+1}, \tau) + |A_{\text{ins}}^{t+1}(\beta, \tau]| \geq 1,$$

since $g \in A_{\text{ins}}^{t+1}(\beta, \tau]$ and $\mathsf{pslack}(P_{\text{ins}}^{t+1}, \tau) \geq 0$.

In the second case we have $\varrho \in B_{\text{ins}}^{t+1} \setminus P_{\text{ins}}^{t+1}$, so $f = \varrho$. We then have $g = g^*$ (as we assume that $f \neq \varrho$ or $g \neq p$). For $\tau \in [d_g, d_f)$, we use Lemma 5.4(b) with $\zeta = \gamma$ to obtain

$$\mathsf{pslack}(B_{\text{ins}}^{t+1}, \tau) \geq \mathsf{pslack}(P_{\text{ins}}^{t+1}, \tau) + |B_{\text{ins}}^{t+1}(\tau, \gamma] \setminus P_{\text{ins}}^{t+1}| \geq 1,$$

where we use $f = \varrho \in B_{\text{ins}}^{t+1}(\tau, \gamma] \setminus P_{\text{ins}}^{t+1}$ in the second inequality. $\square$

From the definitions of $P_{(i)}^{t+1}$, $A_{(i)}^{t+1}$, and $B_{(i)}^{t+1}$, and using Claim 5.9, invariants (InvA) and (InvB) hold for these three intermediate sets.

Before replacing p by ϱ in the optimal plan and in $\mathcal{B}$, we prove the following claim.

CLAIM 5.10. *Assume that* $f = f^* \neq \varrho$ *or* $g = g^* \neq p$. *If* $d_\varrho < d_p$, *then* $\mathsf{pslack}(B_{(i)}^{t+1}, \tau) \geq 1$ *for any* $\tau \in [d_\varrho, d_p)$.

Proof. Let $\hat{f}$ be the earliest-deadline packet in $B_{(i)}^{t+1}(\beta, T] \setminus P_{(i)}^{t+1}$, and let $\hat{g}$ be the latest-deadline packet in $A_{(i)}^{t+1}[t+1, \gamma]$. These definitions are similar to the definitions of f^* and g^*, but they are not equivalent because in part (i) we could have removed f^* from $\mathcal{B}$ and g^* from $\mathcal{A}$. For this reason, packets $\hat{f}$ and $\hat{g}$ may not actually exist, in which case, abusing notation, we will introduce artificial notations for their deadlines: if $\hat{f}$ does not exist, we let $d_{\hat{f}} = T$, and, similarly, if $\hat{g}$ does not exist, we let $d_{\hat{g}} = t+1$. We now observe that

$$(5.33) \qquad \mathsf{pslack}(B_{(i)}^{t+1}, \tau) \geq \mathsf{pslack}(P_{(i)}^{t+1}, \tau) \geq 1$$
$$\text{for } \tau \in (\beta, \min(d_{\hat{f}}, \gamma)) \cup [\max(d_{\hat{g}}, \beta+1), \gamma).$$

Indeed, the first inequality follows from Lemma 5.4, which we can apply here because both invariants (InvA) and (InvB) are true for $P_{(i)}^{t+1}$, $A_{(i)}^{t+1}$, and $B_{(i)}^{t+1}$. Specifically, apply Lemma 5.4(a) with $\zeta = \beta$, obtaining the first inequality for $\tau \in (\beta, d_{\hat{f}})$, and Lemma 5.4(b) with $\zeta = \gamma$, to obtain the first inequality for $\tau \in (d_{\hat{g}}, \gamma)$. To justify the second inequality in (5.33), note that, as $(\beta, \gamma]$ is a segment of $P_{(i)}^{t+1} = P_{\text{ins}}^{t+1}$, we have $\mathsf{pslack}(P_{(i)}^{t+1}, \tau) \geq 1$ for $\tau \in (\beta, \gamma)$.

Since $[d_\varrho, d_p) \subseteq (\beta, \gamma)$, inequality (5.33) directly implies Claim 5.10 when $d_{\hat{f}} \geq \gamma$ or $d_{\hat{g}} \leq \beta$. Therefore, for the rest of the argument, we will assume that $d_{\hat{f}} < \gamma$ and $d_{\hat{g}} > \beta$.

Next, observe that, since $B_{(i)}^{t+1}$ is obtained from B_{ins}^{t+1} by replacing f by g, we have

$$(5.34) \qquad \mathsf{pslack}(B_{(i)}^{t+1}, \tau) = \mathsf{pslack}(B_{\text{ins}}^{t+1}, \tau) + 1 \geq 1 \quad \text{for } \tau \in [d_f, d_g),$$

where the inequality follows from the feasibility of B_{ins}^{t+1}. (It is possible that $d_g \leq d_f$, in which case this interval is empty.)

From these two observations, (5.33) and (5.34), it is now sufficient to show that

$$(5.35) \qquad [d_\varrho, d_p) \subseteq (\beta, d_{\hat{f}}) \cup [d_f, d_g) \cup [d_{\hat{g}}, \gamma).$$

Note that $d_{\hat{g}} \leq d_{g^*}$ and $d_{\hat{f}} \geq d_{f^*}$, by the definitions of these packets and of $B_{(i)}^{t+1}$ and $A_{(i)}^{t+1}$ (note that g is not a candidate for $\hat{f}$ as $g \in P_{(i)}^{t+1}$).

Now we consider three cases.

If $f = \varrho$, then it holds that $g = g^*$ by the claim's assumption, and interval $[d_\varrho, d_p)$ is covered by $[d_f, d_g) = [d_\varrho, d_{g^*})$ and $[d_{\hat{g}}, \gamma)$, because $d_{\hat{g}} \leq d_{g^*}$ and $d_p \leq \gamma$.

Similarly, if $g = p$, then we have $f = f^*$, and interval $[d_\varrho, d_p)$ is covered by $[d_f, d_g) = [d_{f^*}, d_p)$ and $(\beta, d_{\hat{f}})$, because $d_{\hat{f}} \geq d_{f^*}$.

The last case is when $f = f^*$ and $g = g^*$. Then interval $[d_\varrho, d_p) \subseteq (\beta, \gamma)$ is covered by $(\beta, d_{f^*}) \cup [d_{f^*}, d_{g^*}) \cup [d_{g^*}, \gamma)$, which implies (5.35) because $d_{\hat{f}} \geq d_{f^*}$ and $d_{\hat{g}} \leq d_{g^*}$. □

Finally, we show that the new backup plan $\widetilde{B}^{t+1} = B_{(i)}^{t+1} \setminus \{p\} \cup \{\varrho\}$ is feasible. If $d_\varrho \geq d_p$, then the feasibility of $\mathcal{B}$ is clearly preserved after replacing p by ϱ. Otherwise, $d_\varrho < d_p$, and for $\tau \in [d_\varrho, d_p)$ we have $\textsf{pslack}(\widetilde{B}^{t+1}, \tau) = \textsf{pslack}(B_{(i)}^{t+1}, \tau) - 1 \geq 0$ by Claim 5.10, while $\textsf{pslack}(\widetilde{B}^{t+1}, \tau) = \textsf{pslack}(B_{(i)}^{t+1}, \tau) \geq 0$ for $\tau \notin [d_\varrho, d_p)$ by Claim 5.9. Hence, invariant (InvB) holds after the step.

5.5.6. Processing $(\beta, \gamma]$ in an iterated leap step. Here we address the last (and by far most involved) part of our argument, namely the analysis of an iterated leap step. As a reminder, this step occurs when $k \geq 1$ in the algorithm, or, equivalently, when d_ϱ is in a later segment of P^t than d_p. The initial comments in section 5.5.5 apply here as well. To recap: We assume that $\mathcal{A}$ and $\mathcal{B}$ have already been modified as described in sections 5.5.1 (the adversary step) and 5.5.4 (processing segment $[t, \alpha]$). In particular, the time step has already been incremented to $t + 1$ and the current snapshots of $\mathcal{P}$, $\mathcal{B}$, and $\mathcal{A}$ are denoted P_{ins}^{t+1}, B_{ins}^{t+1}, and A_{ins}^{t+1}, respectively. Recall that $P_{\text{ins}}^{t+1} = P^t \setminus \{\ell\}$. We assume that both invariants (InvA) and (InvB) hold for these sets (with respect to time step $t + 1$).

As before, $\beta = \textsf{prevts}^t(d_p)$ and $\gamma = \textsf{nextts}^t(d_\varrho)$. As this is an iterated leap step, we have $d_\varrho > \textsf{nextts}^t(d_p)$, so the interval $(\beta, \gamma]$ of P_{ins}^{t+1} is a union of two or more consecutive segments of P_{ins}^{t+1}. (The content of this interval is identical to the same interval in P^t; in fact, P_{ins}^{t+1} and P^t are identical starting right after $\alpha = \textsf{nextts}^t(t)$, the first tight slot of P^t.) Let $h_0 = p, h_1, \ldots, h_k$ be the packets from Algorithm PlanM (line 9), and let $h_{k+1} = \varrho$. All packets $h_0, h_1, \ldots, h_k$ are in different segments of P_{ins}^{t+1}, not necessarily consecutive, while $h_{k+1} = \varrho$ has deadline in the same segment as h_k, but $\varrho \notin P_{\text{ins}}^{t+1}$. The deadlines and weights of packets ϱ and $h_1, \ldots, h_k$ are still unchanged before processing $(\beta, \gamma]$ (that is, they are the same as in step t).

We need to estimate the potential change due to weight increases and the changes triggered by the replacement of p by ϱ and by the "shifting" of h_i's. We also need to prove key inequality (5.24) and that invariants (InvA) and (InvB) hold after the step.

The overall structure of the argument in this case is similar to Case L.(β, γ).S (simple leap step), with the analysis split according to whether changes in $\mathcal{A}$ are needed (Case L.(β, γ).I.2) or not (Case L.(β, γ).I.1). In Case L.(β, γ).I.2, the analysis will be divided into several stages corresponding to processing of different groups of segments.

For $i = 0, \ldots, k$, let $\mu_i = \mathsf{minwt}(P^t, d_{h_i}^t)$; note that $w_\ell^t \geq \mu_0 \geq \mu_1 \geq \cdots \geq \mu_k = \mathsf{minwt}(P^t, d_\varrho)$ and the algorithm ensures that $w_{h_i}^{t+1} \geq \mu_{i-1}$ for $i = 1, \ldots, k+1$. In both cases below, we use the following simple bound on the change of weights of h_i's.

LEMMA 5.11. *For any $1 \leq a' \leq b' \leq k$, let $\Delta^t w(h_{a'}, \ldots, h_{b'})$ be the total amount by which the algorithm increases the weights of packets $h_{a'}, \ldots, h_{b'}$ in step t. Suppose that there exists $i \in [a', b']$ such that $w_{h_i}^t < \mu_{i-1}$, i.e., the algorithm increases the weight of h_i in Line 11 (and thus $\Delta^t w(h_{a'}, \ldots, h_{b'}) > 0$). Then $\Delta^t w(h_{a'}, \ldots, h_{b'}) \leq \mu_{a'-1} - w_{h_{b'}}^t$.*

Proof. Let $c \in [a', b']$ be the maximum index such that $w_{h_c}^t < \mu_{c-1}$; such c exists by the assumption of the lemma. We show the claim as follows:

$$\Delta^t w(h_{a'}, \ldots, h_{b'}) = \sum_{i=a'}^{c} \max(\mu_{i-1} - w_{h_i}^t, 0)$$

$$(5.36) \qquad \leq \sum_{i=a'}^{c-1} \max(\mu_{i-1} - \mu_i, 0) + \max(\mu_{c-1} - w_{h_c}^t, 0)$$

$$(5.37) \qquad = \sum_{i=a'}^{c-1} (\mu_{i-1} - \mu_i) + \mu_{c-1} - w_{h_c}^t$$

$$= \mu_{a'-1} - w_{h_c}^t$$

$$(5.38) \qquad \leq \mu_{a'-1} - w_{h_{b'}}^t,$$

where inequality (5.36) follows from $w_{h_i}^t \geq \mu_i$, equality (5.37) from $\mu_{i-1} \geq \mu_i$ and from $\mu_{c-1} > w_{h_c}^t$ (by the choice of c), and inequality (5.38) from $w_{h_c}^t \geq w_{h_{b'}}^t$ by Lemma B.1(b) and $c \leq b'$. $\qquad\square$

Case L.$(\beta, \gamma]$.I.1. $\varrho \notin B_{\mathrm{ins}}^{t+1}$ and $A_{\mathrm{ins}}^{t+1}(\beta, \gamma] = \emptyset$. In this case, we do not make any changes in $\mathcal{A}$, i.e., $\widetilde{A}^{t+1} = A_{\mathrm{ins}}^{t+1}$. From the case condition, $p = h_0, h_1, \ldots, h_k \notin A_{\mathrm{ins}}^{t+1}$, because each h_i is in $P_{\mathrm{ins}}^{t+1}(\beta, \gamma]$. It follows that $\mathcal{A}$ is indeed not affected by the decrease of the deadlines of h_i's. Thus invariant (InvA) holds. Note that $\varrho \notin A_{\mathrm{ins}}^{t+1}$ by invariant (InvA). From the case condition and $p \in P_{\mathrm{ins}}^{t+1}(\beta, \gamma]$, we have $p \notin A_{\mathrm{ins}}^{t+1}$; therefore $p \in B_{\mathrm{ins}}^{t+1}$ by invariant (InvB)(ii). This means that in order to preserve invariant (InvB)(ii) we need to remove p from $\mathcal{B}$ and add ϱ to it. Thus we set $\widetilde{B}^{t+1} = B_{\mathrm{ins}}^{t+1} \setminus \{p\} \cup \{\varrho\}$.

Deriving inequality (5.24). To prove this inequality, we first show that

$$(5.39) \qquad \Delta^t w(h_1, \ldots, h_k) \leq \mu_0 - \mu_k.$$

(The weight increase $\Delta^t w(h_1, \ldots, h_k)$ was defined in Lemma 5.11.) Indeed, if there is no $i \in [1, k]$ such that $w_{h_i}^t < \mu_{i-1}$, then $\Delta^t w(h_1, \ldots, h_k) = 0 \leq \mu_0 - \mu_k$ as $\mu_0 \geq \mu_k$. Otherwise, we use Lemma 5.11 with $a' = 1$ and $b' = k$ to get $\Delta^t w(h_1, \ldots, h_k) \leq \mu_0 - w_{h_k}^t \leq \mu_0 - \mu_k$, where the last inequality follows from $w_{h_k}^t \geq \mu_k$.

In this case, the terms on the left-hand side of inequality (5.24) are

$$\Delta_{(\beta, \gamma]} \Psi = \frac{1}{\phi} \left(-w_p^t + w_\varrho^{t+1} + \Delta^t w(h_1, \ldots, h_k) \right),$$

$$\Delta^t w = w_\varrho^{t+1} - w_\varrho^t + \Delta^t w(h_1, \ldots, h_k),$$

$$\mathsf{advcredit}_{(\beta, \gamma]}^t = 0.$$

The equation for $\Delta_{(\beta,\gamma]}\Psi$ is true because all packets $h_1, \ldots, h_k$ are both in P_{ins}^{t+1} and in $\widetilde{P}^{t+1}$, by Lemma B.1(a), and none of them is in $A_{\text{ins}}^{t+1} = \widetilde{A}^{t+1}$, so we keep all of them in $\mathcal{B}$ to preserve invariant (InvB)(ii). The last equation holds because we have not changed the adversary stash. Plugging in these formulas, taking into account that $w_{\varrho}^{t+1} = \mu_k$, and using inequality (5.39), we can obtain inequality (5.24) with an easy calculation:

$$
\begin{aligned}
\Delta_{(\beta,\gamma]}\Psi &- \phi\left(\Delta^t w\right) - \mathsf{advcredit}_{(\beta,\gamma]}^t \\
&= \frac{1}{\phi}\left(-w_p^t + w_{\varrho}^{t+1} + \Delta^t w(h_1, \ldots, h_k)\right) \\
&\qquad - \phi\left(w_{\varrho}^{t+1} - w_{\varrho}^t + \Delta^t w(h_1, \ldots, h_k)\right) - 0 \\
&= -\frac{1}{\phi}\,w_p^t - \mu_k + \phi\,w_{\varrho}^t - \Delta^t w(h_1, \ldots, h_k) \\
&\geq -\frac{1}{\phi}\,w_p^t - \mu_k + \phi\,w_{\varrho}^t - (\mu_0 - \mu_k) \\
&= -\frac{1}{\phi}\,w_p^t + \phi w_{\varrho}^t - \mu_0 \\
&\geq -\frac{1}{\phi}\,w_p^t + \phi\,w_{\varrho}^t - \left(\frac{1}{\phi^2}\,w_p^t + \frac{1}{\phi}\,w_{\varrho}^t\right) \\
&= -w_p^t + w_{\varrho}^t\,,
\end{aligned}
$$

(5.40)

where inequality (5.40) follows from $\mu_0 = \mathsf{minwt}(P^t, d_p) \leq \frac{1}{\phi^2}\,w_p^t + \frac{1}{\phi}\,w_{\varrho}^t$, which is inequality (5.26) with $\tau = d_p$.

Preserving invariant (InvB). That invariant (InvB)(ii) is preserved follows immediately from the definitions of $\widetilde{P}^{t+1}$, $\widetilde{A}^{t+1}$, and $\widetilde{B}^{t+1}$. Next, we show invariant (InvB)(i), which is the feasibility of $\widetilde{B}^{t+1}$, and we split the argument into two parts. First, we show that the invariant holds for the intermediate backup plan resulting from replacing p by ϱ, and then later we show that modifying the deadlines of the h_i's preserves the invariant as well.

So consider an intermediate plan $P_{(i)}^{t+1} = P_{\text{ins}}^{t+1} \setminus \{p\} \cup \{\varrho\}$ and backup plan $B_{(i)}^{t+1} = B_{\text{ins}}^{t+1} \setminus \{p\} \cup \{\varrho\}$, both with respect to time $t + 1$ but with the deadlines of packets $h_1, h_2, \ldots, h_k$ still unchanged. Since this is an iterated leap step, we have that $d_{\varrho} > d_p$, so replacing p by ϱ in $\mathcal{B}$ preserves feasibility. Therefore, $B_{(i)}^{t+1}$ is feasible.

Now, we consider $\widetilde{B}^{t+1}$, that is, the backup plan at step $t + 1$, which is obtained from $B_{(i)}^{t+1}$ by decreasing the deadlines of h_i's, as in the algorithm. As slots outside (β, γ) are not affected, we have $\mathsf{pslack}(\widetilde{B}^{t+1}, \tau) = \mathsf{pslack}(B_{(i)}^{t+1}, \tau) \geq 0$ for $\tau \notin (\beta, \gamma)$. So it remains to consider slots $\tau \in (\beta, \gamma)$.

Recall that packets $h_1, \ldots, h_k$ belong to both B_{ins}^{t+1} and $\widetilde{B}^{t+1}$. We also have that $\mathsf{pslack}(P_{(i)}^{t+1}, \tau) \geq 1$ for any $\tau \in (\beta, \gamma)$, which holds as replacing p by ϱ merges all segments between $\beta = \mathsf{prevts}^t(d_p)$ and $\gamma = \mathsf{nextts}^t(d_{\varrho})$. Since $A_{(i)}^{t+1}(\beta, \gamma] = A_{\text{ins}}^{t+1}(\beta, \gamma] = \emptyset$, Lemma 5.4(b) with $\zeta = \gamma$ gives us that for any $\tau \in (\beta, \gamma)$,

$$
(5.41) \qquad \mathsf{pslack}(B_{(i)}^{t+1}, \tau) \geq \mathsf{pslack}(P_{(i)}^{t+1}, \tau) \geq 1\,.
$$

Next, we claim that for $\tau \in (\beta, \gamma)$,

$$
(5.42) \qquad \mathsf{pslack}(\widetilde{B}^{t+1}, \tau) \geq \mathsf{pslack}(B_{(i)}^{t+1}, \tau) - 1\,.
$$

To justify this inequality, we observe that $d_{h_i}^{t+1} = \tau_{i-1} \geq d_{h_{i-1}}^t$ for $i = 1, \ldots, k$, which gives us that intervals $[d_{h_i}^{t+1}, d_{h_i}^t) \subset (\beta, \gamma)$ are disjoint. This implies in turn that, for $\tau \in (\beta, \gamma)$, decreasing the deadlines of all h_i's can decrease $\mathsf{pslack}(\mathcal{B}, \tau)$ at most by one, which yields inequality (5.42).

Inequalities (5.41) and (5.42) yield $\mathsf{pslack}(\widetilde{B}^{t+1}, \tau) \geq \mathsf{pslack}(B_{(i)}^{t+1}, \tau) - 1 \geq 0$ for $\tau \in (\beta, \gamma)$, completing the proof that $\widetilde{B}^{t+1}$ satisfies invariant (InvB)(i).

Case L.$(\beta, \gamma]$.I.2. $\varrho \in B_{\mathrm{ins}}^{t+1}$ or $A_{\mathrm{ins}}^{t+1}(\beta, \gamma] \neq \emptyset$. In this case, both sets $\mathcal{A}$ and $\mathcal{B}$ will be changed. We focus on the segments of P_{ins}^{t+1} which contain the packets $h_0 = p, h_1, \ldots, h_k$. These are the segments of the current optimal plan that are modified by the algorithm when p is transmitted. Specifically, for $i = 0, \ldots, k$, let S_i be the segment of P_{ins}^{t+1} that ends at $\tau_i = \mathsf{nextts}^t(d_{h_i}^t)$, that is the segment containing $d_{h_i}^t$. (See Figure 4.1.) Recall that $\mathsf{prevts}^t(d_p) = \beta$, $\tau_k = \gamma$, and that we defined $h_{k+1} = \varrho$. We first organize segments S_i into groups and then split the sequence of changes in this case into stages, having one stage for each group.

Groups. We start by defining a packet $g \in A_{\mathrm{ins}}^{t+1}$. Let g^* be the latest-deadline packet in $A_{\mathrm{ins}}^{t+1}[t+1, \gamma]$. Observe that packet g^* is well defined. This is trivially true if the second part of the case condition holds, while in the case when $\varrho \in B_{\mathrm{ins}}^{t+1} \setminus P_{\mathrm{ins}}^{t+1}$, we use Observation 5.2(b) with $\eta = \gamma$ to get that $A_{\mathrm{ins}}^{t+1}[t+1, \gamma] \neq \emptyset$. (It is possible that $d_{g^*} \leq \beta$ in the second case.)

We now define g as follows: If $d_{g^*}^t$ is in a segment S_i for which $h_i \in A_{\mathrm{ins}}^{t+1}$, then let $g = h_i$; otherwise let $g = g^*$. Observe that, in particular, if $h_k \in A_{\mathrm{ins}}^{t+1}$, then $g = h_k$.

We will process segments S_i in *groups*, where each group is specified by some nonempty interval of indices $[a, b] \subseteq \{0, \ldots, k\}$ of segments S_i. We will denote such a group by $\langle a, b \rangle$ (to avoid confusion with the notation for segments and time intervals). These intervals will be chosen so that they partition the set of all indices $\{0, \ldots, k\}$. Roughly, we have a group for each $h_i \in A_{\mathrm{ins}}^{t+1}$ (that needs to be replaced in or removed from $\mathcal{A}$ because its deadline needs to be decreased), a special last group, and possibly a special group at the beginning. Let $i_1 < i_2 < \cdots < i_l$ be the indices of those packets $h_0 = p, h_1, \ldots, h_k$ that are in A_{ins}^{t+1}. Note that if $l > 0$, then $d_g^t \in [d^t(h_{i_l}), \gamma]$, because $h_{i_l} \in A_{\mathrm{ins}}^{t+1}$ is a candidate for g^*. In particular, since $g \in A_{\mathrm{ins}}^{t+1}$, we have that $g \notin \{h_0, \ldots, h_k\} \setminus \{h_{i_l}\}$; that is, among *all* packets $h_0, \ldots, h_k$, g may be possibly equal only to h_{i_l}. The definition of these groups depends on whether some packet h_i is in A_{ins}^{t+1} (that is $l > 0$) or none of them is in A_{ins}^{t+1} (that is $l = 0$):

Case $l > 0$. If $l > 1$, then for each $c = 1, \ldots, l-1$, the interval $\langle i_c, i_{c+1} - 1 \rangle$ is a *middle group*. If $i_1 > 0$ (meaning that $h_0 = p \notin A_{\mathrm{ins}}^{t+1}$), then there is a special *initial group* $\langle 0, i_1 - 1 \rangle$. If $i_1 = 0$, then there is no initial group. Next, we assign the indices in $[i_l, k]$ to one or two groups and define index $\tilde{a}$. If $g = h_{i_l}$ (in particular, if $i_l = k$), then $\langle i_l, k \rangle$ is the *terminal group*, and we let $\tilde{a} = i_l$. Otherwise, $g \neq h_{i_l}$, and we let $\tilde{a}$ be the smallest index in $0, \ldots, k$ for which $\tau_{\tilde{a}} \geq d_g^t$. The assumption that $g \neq h_{i_l}$ implies that $\tilde{a} > i_l$. Then $\langle \tilde{a}, k \rangle$ is the terminal group and $\langle i_l, \tilde{a} - 1 \rangle$ is a new middle group. (See Figure 5.4 for two examples that illustrate the case when $l > 0$.)

Case $l = 0$. We define one or two groups only. There is the terminal group $\langle \tilde{a}, k \rangle$, where $\tilde{a}$ is again the smallest index in $0, \ldots, k$ with $\tau_{\tilde{a}} \geq d_g^t$. If $\tilde{a} > 0$, we also have the initial group $\langle 0, \tilde{a} - 1 \rangle$.

Observe that we always define $\tilde{a}$ so that $\langle \tilde{a}, k \rangle$ is the terminal group. We further define index $\tilde{b}$ so that the first group is $\langle 0, \tilde{b} \rangle$; for example, if $l > 0$ and $i_1 > 0$, then $\tilde{b} = i_1 - 1$. Note that in almost every group $\langle a, b \rangle$ packet h_a is in A_{ins}^{t+1}; the only two possible exceptions are (i) the initial group, and (ii) the terminal group in case when either $l = 0$ or $l \geq 1$ and $\tilde{a} \neq i_l$. On the other hand, for any group $\langle a, b \rangle$
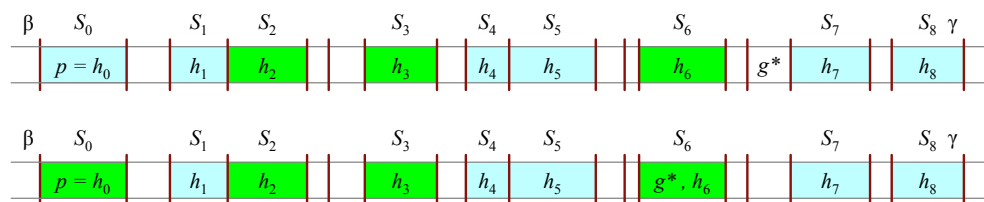
FIG. 5.4. *Two examples that illustrate assigning segments into groups. In both examples we have $k = 8$. Segments are separated by vertical bars. Segments S_i are colored (shaded). Green (or dark shaded) segments S_i are those for which packet h_i is in A_{ins}^{t+1}. In the first example, we have $l = 3$, $(i_1, i_2, i_3) = (2, 3, 6)$, $g = g^*$ (as g^* is not in S_i for any i), and $\tilde{a} = 7$. We have 5 groups: the initial group $\langle 0, 1 \rangle$, middle groups $\langle 2, 2 \rangle$, $\langle 3, 5 \rangle$, and $\langle 6, 6 \rangle$, and the terminal group $\langle 7, 8 \rangle$. In the second example we have $l = 4$, $(i_1, i_2, i_3, i_4) = (0, 2, 3, 6)$, and $g = h_6$ (because g^* is in S_6 and h_6 is in A_{ins}^{t+1}). Since $i_1 = 0$ there is no initial group. We have 4 groups: middle groups $\langle 0, 1 \rangle$, $\langle 2, 2 \rangle$, $\langle 3, 5 \rangle$, and the terminal group $\langle 6, 8 \rangle$.*
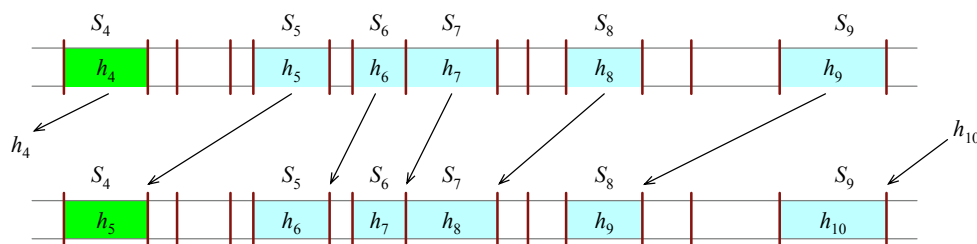

FIG. 5.5. *An illustration of the changes in $\mathcal{P}$ in the stage when a middle group $\langle a, b \rangle = \langle 4, 9 \rangle$ is processed.*

packets $h_{a+1}, \ldots, h_b$ are never in A_{ins}^{t+1}, and thus these packets are in B_{ins}^{t+1} by invariant (InvB)(ii).

Stages. The changes implemented during the processing of interval $(\beta, \gamma]$ involve the replacement of p by ϱ in $\mathcal{P}$, weight increases of ϱ and h_i's, and deadline decreases of h_i's, as described in the algorithm. These changes will also trigger changes of $\mathcal{B}$ and $\mathcal{A}$ necessary to preserve the invariants. We will organize all these changes (of $\mathcal{P}$, $\mathcal{B}$, $\mathcal{A}$, and the instance itself) into a sequence of *stages*, with one stage for each group $\langle a, b \rangle$. The stages will be implemented in the reverse order of the groups, that is, starting from the stage for the terminal group (which is defined in all cases), continuing with middle groups in the reverse order of their indices, and ending by the group containing index 0, which may be of any type.

To better understand the partition into stages, as described below, it helps to think of the substitution of p by $\varrho = h_{k+1}$ in the optimal plan as a chain of substitutions: first replace h_k by ϱ, then replace h_{k-1} by h_k, and so on, and eventually replace $p = h_0$ by h_1. In our process here, rather than processing these substitutions individually, we process the substitutions within each group $\langle a, b \rangle$ simultaneously as one stage, in which h_a is replaced by h_{b+1}. (See Figure 5.5 for an illustration.)

Formally, let $\dot{P}_{\langle a, b \rangle}^{t+1}$ be the snapshot of the intermediate plan $\mathcal{P}$ just before the stage for group $\langle a, b \rangle$, and let $\ddot{P}_{\langle a, b \rangle}^{t+1}$ be the snapshot of the intermediate plan $\mathcal{P}$ just after the stage for group $\langle a, b \rangle$. (We remark that $\dot{P}_{\langle a, b \rangle}^{t+1}$ is not necessarily an optimal plan. It is only meant to be an "intermediate plan" used to capture gradual changes in the optimal plan as stages are executed. It is, however, feasible, and it will satisfy
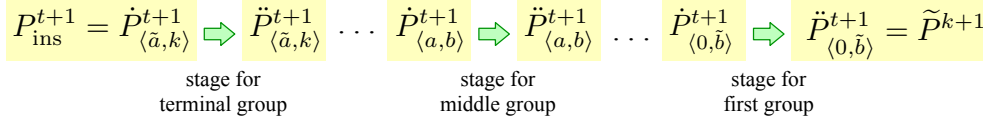
$$P_{\text{ins}}^{t+1} = \dot{P}_{\langle\tilde{a},k\rangle}^{t+1} \;\Rightarrow\; \ddot{P}_{\langle\tilde{a},k\rangle}^{t+1} \;\cdots\; \dot{P}_{\langle a,b\rangle}^{t+1} \;\Rightarrow\; \ddot{P}_{\langle a,b\rangle}^{t+1} \;\cdots\; \dot{P}_{\langle 0,\tilde{b}\rangle}^{t+1} \;\Rightarrow\; \ddot{P}_{\langle 0,\tilde{b}\rangle}^{t+1} = \widetilde{P}^{k+1}$$

<div align="center">

stage for
terminal group   stage for
middle group   stage for
first group

</div>

FIG. 5.6. *Notation for stages.*

appropriate invariants.) Naturally, if $\langle a,b\rangle$ and $\langle b+1,c\rangle$ are two consecutive groups, then $\ddot{P}_{\langle b+1,c\rangle}^{t+1} = \dot{P}_{\langle a,b\rangle}^{t+1}$. We adopt the same conventions for sets $\mathcal{B}$ and $\mathcal{A}$. In this notation (see Figure 5.6 for an illustration) we have the following:

- In the first stage, that corresponds to the terminal group $\langle\tilde{a},k\rangle$, we have $\dot{P}_{\langle\tilde{a},k\rangle}^{t+1} = P_{\text{ins}}^{t+1}$, $\dot{B}_{\langle\tilde{a},k\rangle}^{t+1} = B_{\text{ins}}^{t+1}$, and $\dot{A}_{\langle\tilde{a},k\rangle}^{t+1} = A_{\text{ins}}^{t+1}$. Note that in this stage, it holds that $h_{k+1} = \varrho \notin \dot{P}_{\langle\tilde{a},k\rangle}^{t+1}$.

- When describing the stage for each group $\langle a,b\rangle$, we assume that the changes in stages for groups $\langle a',b'\rangle$ with $a' > b$ have already been implemented. As an invariant we will ensure that $h_{b+1} \notin \dot{P}_{\langle a,b\rangle}^{t+1}$ (for $b=k$, this holds by $h_{k+1} = \varrho \notin \dot{P}_{\langle\tilde{a},k\rangle}^{t+1} = P_{\text{ins}}^{t+1}$). In this stage, the plan changes from $\dot{P}_{\langle a,b\rangle}^{t+1}$ to $\ddot{P}_{\langle a,b\rangle}^{t+1} = \dot{P}_{\langle a,b\rangle}^{t+1} \setminus \{h_a\} \cup \{h_{b+1}\}$, which will imply that $h_a \notin \ddot{P}_{\langle a,b\rangle}^{t+1}$ just after this stage is processed. Further, the weights and deadlines of packets $h_{a+1},\ldots,h_{b+1}$ are adjusted according to the algorithm (for $b=k$, the deadline of $h_{k+1} = \varrho$ remains the same). This stage also involves changes in $\mathcal{A}$ and $\mathcal{B}$ triggered by the changes in $\mathcal{P}$ and by the adjustments of weights and deadlines. These changes will convert $\dot{B}_{\langle a,b\rangle}^{t+1}$ and $\dot{A}_{\langle a,b\rangle}^{t+1}$ to $\ddot{B}_{\langle a,b\rangle}^{t+1}$ and $\ddot{A}_{\langle a,b\rangle}^{t+1}$, respectively.

- After the last stage, corresponding to the first group $\langle 0,\tilde{b}\rangle$ (which could be an initial, middle, or terminal group), we have $\ddot{P}_{\langle 0,\tilde{b}\rangle}^{t+1} = \widetilde{P}^{t+1}$, $\ddot{B}_{\langle 0,\tilde{b}\rangle}^{t+1} = \widetilde{B}^{t+1}$, and $\ddot{A}_{\langle 0,\tilde{b}\rangle}^{t+1} = \widetilde{A}^{t+1}$.

By the results of section 5.5.4, invariants (InvA) and (InvB) hold for $\dot{P}_{\langle\tilde{a},k\rangle}^{t+1}$, $\dot{B}_{\langle\tilde{a},k\rangle}^{t+1}$, and $\dot{A}_{\langle\tilde{a},k\rangle}^{t+1}$. In the stage for each group $\langle a,b\rangle$, we will show that these invariants are preserved and that an appropriate bound on amortized profits holds. This will give us that inequality (5.24) holds when processing interval $(\beta,\gamma]$ and that invariants (InvA) and (InvB) hold afterwards, that is for sets $\widetilde{P}^{t+1}$, $\widetilde{B}^{t+1}$, and $\widetilde{A}^{t+1}$.

Note that the changes of $\mathcal{P}$ implemented by the algorithm in the stage for group $\langle a,b\rangle$ satisfy

$$\dot{P}_{\langle a,b\rangle}^{t+1} \setminus \dot{P}_{\langle a,b\rangle}^{t+1}(\eta,\tau_b] = \ddot{P}_{\langle a,b\rangle}^{t+1} \setminus \ddot{P}_{\langle a,b\rangle}^{t+1}(\eta,\tau_b]$$

for $\eta = \mathsf{prevts}(\dot{P}_{\langle a,b\rangle}^{t+1}, d_{h_a}^t)$, which is the "locality of changes" invariant (5.29) from section 5.5.3. The $\mathsf{pslack}(\mathcal{P},\tau)$ values are actually affected only for $\tau \in [d_{h_a}^t, \tau_b]$. (We remark that changes of $\mathcal{P}$ involve packet h_{b+1} with $d_{h_{b+1}}^t > \tau_b$ (if $b < k$); however, $h_{b+1} \notin \dot{P}_{\langle a,b\rangle}^{t+1}$, and we change the deadline of h_{b+1} to τ_b during the stage for group $\langle a,b\rangle$.)

Finally, we will maintain the property that after the stage for each group $\langle a,b\rangle$, packet h_a will not be in $\ddot{A}_{\langle a,b\rangle}^{t+1}$, even though it may have been in $\mathcal{A}$ earlier. This is necessary for invariant (InvA), because $h_a \notin \ddot{P}_{\langle a,b\rangle}^{t+1}$. As also $\varrho = h_{k+1} \notin \dot{A}_{\langle\tilde{a},k\rangle}^{t+1} = A_{\text{ins}}^{t+1}$, this will give us that $h_{b+1} \notin \dot{A}_{\langle a,b\rangle}^{t+1}$ for any group $\langle a,b\rangle$. Recall that, importantly, packets $h_{a+1},\ldots,h_b$ are also not in $\dot{A}_{\langle a,b\rangle}^{t+1}$.

Deriving inequality (5.24). We split the potential changes and the adversary credit for replacing packets in $\mathcal{A}$ among groups in a natural way. Let $\Delta_{\langle a,b \rangle}\Psi$ be the total change of the potential due to the modifications implemented in the stage for group $\langle a,b \rangle$, and let $\mathsf{advcredit}^t_{\langle a,b \rangle}$ be the adversary adjustment credit for the changes of $\mathcal{A}$ implemented in the stage for group $\langle a,b \rangle$ (that is, for removing a packet or for replacing h_a by a lighter packet). Recall that $\Delta^t w(h_{a+1}, \ldots, h_{b+1})$ is the total amount by which the algorithm increases the weights of $h_{a+1}, \ldots, h_{b+1}$. Our goal is to prove for each group $\langle a,b \rangle$ that

$$(5.43) \qquad \Delta_{\langle a,b \rangle}\Psi - \phi\,\Delta^t w(h_{a+1}, \ldots, h_{b+1}) - \mathsf{advcredit}^t_{\langle a,b \rangle} \geq -w^t_{h_a} + w^t_{h_{b+1}}.$$

(Note that the right-hand side of (5.43) is negative, by Lemma B.1(b).)

It remains to observe that the sum of (5.43) over all groups gives us exactly the key inequality (5.24). Indeed, the right-hand sides sum to $-w^t_{h_0} + w^t_{h_{k+1}} = -w^t_p + w^t_\varrho$. Regarding the left-hand side of (5.43), $\Delta_{(\beta,\gamma]}\Psi$ equals the sum of $\Delta_{\langle a,b \rangle}\Psi$ over all groups $\langle a,b \rangle$, and similarly for $\Delta^t w$ and $\mathsf{advcredit}^t_{(\beta,\gamma]}$. (Note that the increase of the weight of $\varrho = h_{k+1}$ is accounted for in inequality (5.43) for the terminal group $\langle \tilde{a}, k \rangle$.)

Stage for the terminal group. Let $\langle \tilde{a}, k \rangle$ be the terminal group of segments. In this stage, we change plan $\mathcal{P}$ by removing $h_{\tilde{a}}$ and adding ϱ, we increase the weights of packets $h_{\tilde{a}+1}, \ldots, h_{k+1} = \varrho$, and we decrease the deadlines of packets $h_{\tilde{a}+1}, \ldots, h_k$. We thus have $\ddot{P}^{t+1}_{\langle \tilde{a}, k \rangle} = \dot{P}^{t+1}_{\langle \tilde{a}, k \rangle} \setminus \{h_{\tilde{a}}\} \cup \{\varrho\}$.

Let f^* be the earliest-deadline packet in $\dot{B}^{t+1}_{\langle \tilde{a}, k \rangle}(\beta, T] \setminus \dot{P}^{t+1}_{\langle \tilde{a}, k \rangle}$. We argue that packet f^* is well defined, which is trivially true if $\varrho \in B^{t+1}_{\mathrm{ins}}(\beta, T] = \dot{B}^{t+1}_{\langle \tilde{a}, k \rangle}(\beta, T]$. Otherwise, $A^{t+1}_{\mathrm{ins}}(\beta, \gamma] = \dot{A}^{t+1}_{\langle \tilde{a}, k \rangle}(\beta, \gamma] \neq \emptyset$ and then Observation 5.2(a) with $\eta = \beta$ implies that $\dot{B}^{t+1}_{\langle \tilde{a}, k \rangle}(\beta, T] \setminus \dot{P}^{t+1}_{\langle \tilde{a}, k \rangle} \neq \emptyset$. (Possibly, $d_{f^*} > \gamma$ in the second case.)

We then define a packet $f \in \dot{B}^{t+1}_{\langle \tilde{a}, k \rangle} \setminus \dot{P}^{t+1}_{\langle \tilde{a}, k \rangle}$, letting $f = \varrho$ if $\varrho \in \dot{B}^{t+1}_{\langle \tilde{a}, k \rangle}$, and $f = f^*$ otherwise.

Next, we modify sets $\mathcal{A}$ and $\mathcal{B}$. We remove g from $\mathcal{A}$, i.e., $\ddot{A}^{t+1}_{\langle \tilde{a}, k \rangle} = \dot{A}^{t+1}_{\langle \tilde{a}, k \rangle} \setminus \{g\}$. This ensures that invariant (InvA) is preserved, because $h_{\tilde{a}} \notin \ddot{A}^{t+1}_{\langle \tilde{a}, k \rangle}$ and none of packets $h_{\tilde{a}+1}, \ldots, h_k$ is in $\ddot{A}^{t+1}_{\langle \tilde{a}, k \rangle}$ (by the definition of groups); thus the decrease of their deadlines, implemented in this stage, does not affect $\mathcal{A}$.

Regarding $\mathcal{B}$, note that $\varrho \in \ddot{P}^{t+1}_{\langle \tilde{a}, k \rangle} \setminus \ddot{A}^{t+1}_{\langle \tilde{a}, k \rangle}$, so we need to include ϱ in $\ddot{B}^{t+1}_{\langle \tilde{a}, k \rangle}$ to preserve invariant (InvB)(ii). Hence, we let $\ddot{B}^{t+1}_{\langle \tilde{a}, k \rangle} = B^{t+1}_{\mathrm{ins}} \cup \{g\} \setminus \{f, h_{\tilde{a}}\} \cup \{\varrho\}$; this definition applies no matter whether $g = h_{\tilde{a}}$ or $g = g^*$ nor whether $f = \varrho$ or $f = f^*$.

Deriving (5.43) *for the terminal group.* Apart from the changes in the paragraph above, the algorithm changes the weights of packets $h_{\tilde{a}+1}, \ldots, h_{k+1} = \varrho$, which results in the increase of the weight of the optimal plan. Taking all these changes into account, the terms on the left-hand side of (5.43) are

$$\Delta_{\langle \tilde{a}, k \rangle}\Psi = \frac{1}{\phi}\left(w^t_g - w^t_f - w^t_{h_{\tilde{a}}} + w^{t+1}_\varrho + \Delta^t w(h_{\tilde{a}+1}, \ldots, h_k)\right)$$

$$(5.44) \qquad\qquad = \frac{1}{\phi}\left(w^t_g - w^t_f - w^t_{h_{\tilde{a}}} + \mu_k + \Delta^t w(h_{\tilde{a}+1}, \ldots, h_k)\right),$$

$$\Delta^t w(h_{\tilde{a}+1}, \ldots, h_{k+1}) = \Delta^t w(h_{\tilde{a}+1}, \ldots, h_k) + w^{t+1}_\varrho - w^t_\varrho$$

$$(5.45) \qquad\qquad = \Delta^t w(h_{\tilde{a}+1}, \ldots, h_k) + \mu_k - w^t_\varrho,$$

$$(5.46) \qquad \mathsf{advcredit}^t_{\langle \tilde{a}, k \rangle} = w^t_g - \mathsf{minwt}(P^t, d^t_g).$$

Equation (5.44) follows from the definition of $\ddot{B}^{t+1}_{\langle \tilde{a},k \rangle}$, as ϱ is added to $\mathcal{B}$ with its new weight $w^{k+1}_\varrho = \mu_k$ and packets $h_{\tilde{a}+1}, \ldots, h_k$ remain in $\mathcal{B}$, although their weights are changed. (Recall that for each group $\langle a,b \rangle$, packets $h_{a+1}, \ldots, h_b$ are always in $\mathcal{B}$.) Equation (5.45) holds because $w^{k+1}_\varrho = \mu_k$. In $\mathcal{A}$, we just removed g, implying (5.46). (See section 5.1 for the definition of the adversary adjustment credit associated with the removal of packets from $\mathcal{A}$.)

We further claim that the following inequalities hold:

$$(5.47) \qquad \Delta^t w(h_{\tilde{a}+1}, \ldots, h_k) \ \leq \ \mu_{\tilde{a}} - \mu_k,$$

$$(5.48) \qquad w^t_g \ \leq \ w^t_{h_{\tilde{a}}},$$

$$(5.49) \qquad w^t_f \ \leq \ w^t_\varrho,$$

$$(5.50) \qquad \mathsf{minwt}(P^t, d^t_g) \ \geq \ \mu_{\tilde{a}}.$$

To show (5.47), we consider two simple cases. If there is no $i \in \langle \tilde{a}+1, k \rangle$ such that $w^t_{h_i} < \mu_{i-1}$, then $\Delta^t w(h_{\tilde{a}+1}, \ldots, h_k) = 0 \leq \mu_{\tilde{a}} - \mu_k$ as $\mu_{\tilde{a}} \geq \mu_k$. Otherwise, we use Lemma 5.11 with $a' = \tilde{a}+1$ and $b' = k$ to get $\Delta^t w(h_{\tilde{a}+1}, \ldots, h_k) \leq \mu_{\tilde{a}} - w^t_{h_k} \leq \mu_{\tilde{a}} - \mu_k$, where the last inequality follows from $w^t_{h_k} \geq \mu_k$.

Inequality (5.48) is trivial if $g = h_{i_l}$, in which case also $\tilde{a} = i_l$. Otherwise, $\tilde{a} > i_l$. By the definition of the terminal group, $\tilde{a}$ is the smallest index with $\tau_{\tilde{a}} \geq d^t_g$, and thus $d^t_g > \tau_{\tilde{a}-1}$. We also have that $h_{\tilde{a}}$ is the heaviest packet in plan $\dot{P}^{t+1}_{\langle \tilde{a},k \rangle} = P^{t+1}_{\mathrm{ins}}$ with deadline in $(\tau_{\tilde{a}-1}, \gamma]$. (This property holds for plan P^t, by the algorithm. However, the changes in section 5.5.4, where we process $[t, \alpha]$, do not affect packets in the optimal plan after the first tight slot $\alpha \leq \beta$; see also the "locality of changes" invariant in section 5.5.3.) As $g \in \dot{P}^{t+1}_{\langle \tilde{a},k \rangle}$ and $d^t_g \in (\tau_{\tilde{a}-1}, \gamma]$ (by the definition of $\tilde{a}$), we conclude that $w^t_g \leq w^t_{h_{\tilde{a}}}$.

Next, we justify (5.49). Indeed, since $d^t_f > \beta$ and the only packet possibly added to $\mathcal{B}$ in the analysis of the transmission step is ℓ with $d^t_\ell \leq \alpha \leq \beta$, we have that $f \notin P^t$, and then the inequality holds by the definition of ϱ.

Finally, inequality (5.50) follows from $\tau_{\tilde{a}} \geq d^t_g$ and the monotonicity of the $\mathsf{minwt}()$ function.

Using the above equations and inequalities, we derive inequality (5.43) as follows:

$$\Delta_{\langle \tilde{a},k \rangle} \Psi \ - \phi \, \Delta^t w(h_{\tilde{a}+1}, \ldots, h_{k+1}) - \mathsf{advcredit}^t_{\langle \tilde{a},k \rangle}$$

$$= \frac{1}{\phi} \left(w^t_g - w^t_f - w^t_{h_{\tilde{a}}} + \mu_k + \Delta^t w(h_{\tilde{a}+1}, \ldots, h_k) \right)$$

$$\qquad - \phi \, (\Delta^t w(h_{\tilde{a}+1}, \ldots, h_k) + \mu_k - w^t_\varrho) - (w^t_g - \mathsf{minwt}(P^t, d^t_g))$$

$$= -\frac{1}{\phi^2} \, w^t_g - \frac{1}{\phi} \, w^t_f - \frac{1}{\phi} \, w^t_{h_{\tilde{a}}} - \Delta^t w(h_{\tilde{a}+1}, \ldots, h_k)$$

$$\qquad - \mu_k + \phi \, w^t_\varrho + \mathsf{minwt}(P^t, d^t_g)$$

$$(5.51) \qquad \geq -\frac{1}{\phi^2} \, w^t_{h_{\tilde{a}}} - \frac{1}{\phi} \, w^t_\varrho - \frac{1}{\phi} \, w^t_{h_{\tilde{a}}} - (\mu_{\tilde{a}} - \mu_k) - \mu_k + \phi \, w^t_\varrho + \mu_{\tilde{a}}$$

$$(5.52) \qquad = -w^t_{h_{\tilde{a}}} + w^t_{h_{k+1}},$$

where in inequality (5.51) we use (5.47), (5.48), (5.49), and (5.50), and in step (5.52) we substituted $w^t_\varrho = w^t_{h_{k+1}}$.

Invariant (InvB) after the stage for the terminal group. We claim that after the stage for the terminal group, invariant (InvB) holds for backup plan $\ddot{B}^{t+1}_{\langle \tilde{a},k \rangle}$. (The proof given here is an extension of the one for the case L.$[\beta, \gamma]$.S.2 in section 5.5.5.) To show this, we split all changes of this stage into three parts (see Figure 5.7):
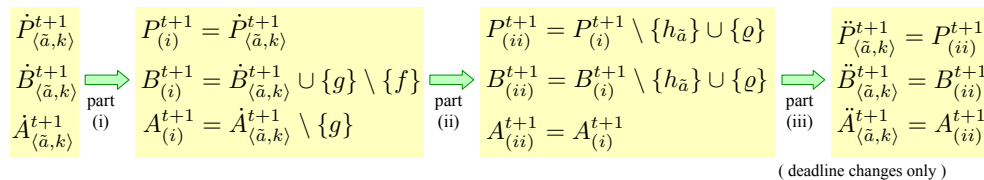
$$\dot{P}^{t+1}_{\langle \tilde{a},k \rangle} \qquad P^{t+1}_{(i)} = \dot{P}^{t+1}_{\langle \tilde{a},k \rangle} \qquad P^{t+1}_{(ii)} = P^{t+1}_{(i)} \setminus \{h_{\tilde{a}}\} \cup \{\varrho\} \qquad \ddot{P}^{t+1}_{\langle \tilde{a},k \rangle} = P^{t+1}_{(ii)}$$

$$\dot{B}^{t+1}_{\langle \tilde{a},k \rangle} \xrightarrow[\text{part}]{} B^{t+1}_{(i)} = \dot{B}^{t+1}_{\langle \tilde{a},k \rangle} \cup \{g\} \setminus \{f\} \xrightarrow[\text{part}]{} B^{t+1}_{(ii)} = B^{t+1}_{(i)} \setminus \{h_{\tilde{a}}\} \cup \{\varrho\} \xrightarrow[\text{part}]{} \ddot{B}^{t+1}_{\langle \tilde{a},k \rangle} = B^{t+1}_{(ii)}$$

$$\dot{A}^{t+1}_{\langle \tilde{a},k \rangle} \quad {}_{(i)} \quad A^{t+1}_{(i)} = \dot{A}^{t+1}_{\langle \tilde{a},k \rangle} \setminus \{g\} \quad {}_{(ii)} \quad A^{t+1}_{(ii)} = A^{t+1}_{(i)} \quad {}_{(iii)} \quad \ddot{A}^{t+1}_{\langle \tilde{a},k \rangle} = A^{t+1}_{(ii)}$$

(deadline changes only)

FIG. 5.7. *Partition of the stage for the terminal group into three parts in the proof that invariant* (InvB) *is preserved.*

(i) First, we implement the changes in $\mathcal{A}$ and $\mathcal{B} \setminus \mathcal{P}$, but postpone changes in $\mathcal{P}$; that is, we consider sets $P^{t+1}_{(i)} = \dot{P}^{t+1}_{\langle \tilde{a},k \rangle}$, $B^{t+1}_{(i)} = \dot{B}^{t+1}_{\langle \tilde{a},k \rangle} \cup \{g\} \setminus \{f\}$, and $A^{t+1}_{(i)} = \dot{A}^{t+1}_{\langle \tilde{a},k \rangle} \setminus \{g\} = \ddot{A}^{t+1}_{\langle \tilde{a},k \rangle}$. The deadlines of packets $h_{\tilde{a}+1}, \ldots, h_k$ are as yet unchanged.

(ii) Second, we replace $h_{\tilde{a}}$ by ϱ in $\mathcal{P}$ but still do not implement the deadline changes, so $P^{t+1}_{(ii)} = P^{t+1}_{(i)} \setminus \{h_{\tilde{a}}\} \cup \{\varrho\}$, $B^{t+1}_{(ii)} = B^{t+1}_{(i)} \setminus \{h_{\tilde{a}}\} \cup \{\varrho\}$, and $A^{t+1}_{(ii)} = A^{t+1}_{(i)}$.

(iii) Third, we decrease the deadlines of packets $h_{\tilde{a}+1}, \ldots, h_k$, ending up with plan $\ddot{P}^{t+1}_{\langle \tilde{a},k \rangle}$ and backup plan $\ddot{B}^{t+1}_{\langle \tilde{a},k \rangle}$. (Note that $\mathcal{A}$ and $\mathcal{B} \setminus \mathcal{P}$ are not affected by the deadline decreases, since packets $h_{\tilde{a}+1}, \ldots, h_k$ are in $P^{t+1}_{(ii)} \setminus A^{t+1}_{(ii)}$.)

In the argument for the first two parts, we can assume that $f = f^* \neq \varrho$ or $g = g^* \neq h_{\tilde{a}}$ (or both), since if both $f = \varrho$ and $g = h_{\tilde{a}}$, then $B^{t+1}_{(ii)} = \dot{B}^{t+1}_{\langle \tilde{a},k \rangle} = B^{t+1}_{\text{ins}}$, in which case invariant (InvB) holds after part (ii), as (InvB)(i) is trivial and (InvB)(ii) follows from the way $\mathcal{P}$ and $\mathcal{A}$ are modified in parts (i) and (ii). (Recall that $g \in \{g^*, h_{\tilde{a}}\}$ and that $g = h_{\tilde{a}}$ iff $\tilde{a} = i_l$.)

CLAIM 5.12. *Assume that $f = f^* \neq \varrho$ or $g = g^* \neq h_{\tilde{a}}$. Then set $B^{t+1}_{(i)} = \dot{B}^{t+1}_{\langle \tilde{a},k \rangle} \cup \{g\} \setminus \{f\}$ is a feasible set of packets pending at time $t+1$.*

Proof. Recall that $\dot{P}^{t+1}_{\langle \tilde{a},k \rangle} = P^{t+1}_{\text{ins}}$ and that the same relations hold for sets $\mathcal{B}$ and $\mathcal{A}$. Also, we are currently in Case L.(β, γ).I.2, in which either $\varrho \in B^{t+1}_{\text{ins}}$ or $A^{t+1}_{\text{ins}}(\beta, \gamma] \neq \emptyset$.

The claim clearly holds if $d^t_f \leq d^t_g$. Otherwise, by the definition of tight slots, it suffices to prove $d^t_f \leq \text{nextts}(B^{t+1}_{\text{ins}}, d^t_g)$, from which the feasibility of $B^{t+1}_{(i)}$ follows. In particular, we show that $\text{pslack}(B^{t+1}_{\text{ins}}, \tau) \geq 1$ for any $\tau \in [d^t_g, d^t_f)$.

We now consider two cases. First, assume that $\varrho \notin B^{t+1}_{\text{ins}}$. This implies $f = f^*$ and $A^{t+1}_{\text{ins}}(\beta, \gamma] \neq \emptyset$, which means that $d^t_g \in (\beta, \gamma]$, no matter whether $g = h_{\tilde{a}}$ or $g = g^*$. Then, for $\tau \in [d^t_g, d^t_f)$, Lemma 5.4(a) with $\zeta = \beta$ gives us

$$\text{pslack}(B^{t+1}_{\text{ins}}, \tau) \geq \text{pslack}(P^{t+1}_{\text{ins}}, \tau) + |A^{t+1}_{\text{ins}}(\beta, \tau]| \geq 1,$$

since $A^{t+1}_{\text{ins}}(\beta, \tau]$ contains g and $\text{pslack}(P^{t+1}_{\text{ins}}, \tau) \geq 0$.

Otherwise, $\varrho \in B^{t+1}_{\text{ins}} \setminus P^{t+1}_{\text{ins}}$, so $f = \varrho$. In this case, $g = g^*$ holds (as we assume in this part that $f = f^* \neq \varrho$ or $g = g^* \neq h_{\tilde{a}}$). For $\tau \in [d^t_g, d^t_f)$, we use Lemma 5.4(b) with $\zeta = \gamma$ to obtain

$$\text{pslack}(B^{t+1}_{\text{ins}}, \tau) \geq \text{pslack}(P^{t+1}_{\text{ins}}, \tau) + |B^{t+1}_{\text{ins}}(\tau, \gamma] \setminus P^{t+1}_{\text{ins}}| \geq 1,$$

where we use $f = \varrho \in B^{t+1}_{\text{ins}}(\tau, \gamma] \setminus P^{t+1}_{\text{ins}}$ in the second inequality. $\square$

From the definitions of $P_{(i)}^{t+1}$, $A_{(i)}^{t+1}$, and $B_{(i)}^{t+1}$, and using Claim 5.12, invariants (InvA) and (InvB) hold for these three intermediate sets.

Before considering part (ii) (that is, replacing $h_{\tilde{a}}$ by ϱ in $\mathcal{P}$ and $\mathcal{B}$), we prove the following claim.

CLAIM 5.13. *Assume that $f = f^* \neq \varrho$ or $g = g^* \neq h_{\tilde{a}}$. If $d_\varrho^t < d_{h_{\tilde{a}}}^t$, then $\tilde{a} = k$ and $\mathsf{pslack}(B_{(i)}^{t+1}, \tau) \geq 1$ for any $\tau \in [d_\varrho^t, d_{h_{\tilde{a}}}^t)$.*

Proof. Let $\chi = \mathsf{prevts}^t(d_\varrho^t)$. By the definitions of γ and ϱ and since the tight slots of P_{ins}^{t+1} are the same as those of P^t starting with slot α, d_ϱ^t is in the segment $(\chi, \gamma]$ of $P_{(i)}^{t+1} = P_{\mathrm{ins}}^{t+1}$; consequently, $d_{h_{\tilde{a}}}^t$ also belongs to this segment. So $\tilde{a} = k$, proving the first part of the claim. Moreover, it follows that $\mathsf{pslack}(P_{(i)}^{t+1}, \tau) \geq 1$ for $\tau \in [d_\varrho^t, d_{h_k}^t)$.

We now make two observations that will be used later to prove Claim 5.13. First, observe that replacing f by g in $\mathcal{B}$ results in

$$(5.53) \qquad \mathsf{pslack}(B_{(i)}^{t+1}, \tau) = \mathsf{pslack}(B_{\mathrm{ins}}^{t+1}, \tau) + 1 \geq 1 \quad \text{for } \tau \in [d_f^t, d_g^t),$$

where the inequality follows from the feasibility of B_{ins}^{t+1}. (It is possible that $d_g^t \leq d_f^t$, in which case this interval is empty.)

Next, let $\hat{f}$ be the earliest-deadline packet in $B_{(i)}^{t+1}(\chi, T] \setminus P_{(i)}^{t+1}$, and let $\hat{g}$ be the latest-deadline packet in $A_{(i)}^{t+1}[t+1, \gamma]$; if $\hat{f}$ is undefined, we let $d_{\hat{f}}^t = T$, and similarly if $\hat{g}$ is undefined, we let $d_{\hat{g}}^t = t + 1$. By Lemma 5.4 (which we can apply because invariants (InvA) and (InvB) are true for $P_{(i)}^{t+1}$, $A_{(i)}^{t+1}$, and $B_{(i)}^{t+1}$), it holds that

$$(5.54) \qquad \mathsf{pslack}(B_{(i)}^{t+1}, \tau) \geq \mathsf{pslack}(P_{(i)}^{t+1}, \tau) \geq 1$$
$$\text{for } \tau \in (\chi, \min(d_{\hat{f}}^t, \gamma)) \cup [\max(d_{\hat{g}}^t, \chi + 1), \gamma).$$

Specifically, we apply Lemma 5.4(a) with $\zeta = \chi$ to derive the first inequality for $\tau \in (\chi, d_{\hat{f}}^t)$, and Lemma 5.4(b) with $\zeta = \gamma$ to derive the first inequality for $\tau \in [d_{\hat{g}}^t, \gamma)$. The second inequality holds because $(\chi, \gamma]$ is a segment of $P_{(i)}^{t+1}$.

If $d_{\hat{f}}^t \geq \gamma$ or $d_{\hat{g}}^t \leq \chi$, then inequality (5.54) directly implies the lemma, because $[d_\varrho^t, d_{h_k}^t) \subseteq (\chi, \gamma)$. Therefore, for the rest of the proof, we will assume that $d_{\hat{f}}^t < \gamma$ and $d_{\hat{g}}^t > \chi$.

To complete the proof, it is sufficient to show that inequalities (5.53) and (5.54) cover the whole range of $\tau \in [d_\varrho^t, d_{h_k}^t)$, that is,

$$(5.55) \qquad [d_\varrho^t, d_{h_k}^t) \subseteq (\chi, d_{\hat{f}}^t) \cup [d_f^t, d_g^t) \cup [d_{\hat{g}}^t, \gamma).$$

Note that $d_{\hat{g}}^t \leq d_{g^*}^t$ and $d_{\hat{f}}^t \geq d_{f^*}^t$ by the definitions of these packets and of $B_{(i)}^{t+1}$ and $A_{(i)}^{t+1}$ (note that g is not a candidate for $\hat{f}$ as $g \in P_{(i)}^{t+1}$). If $f = \varrho$, then it holds that $g = g^*$ by the assumption, and the interval $[d_\varrho^t, d_{h_k}^t)$ is covered by $[d_f^t, d_g^t) = [d_\varrho^t, d_{g^*}^t)$ and $[d_{\hat{g}}^t, \gamma)$, because $d_{\hat{g}}^t \leq d_{g^*}^t$ and $d_{h_k}^t \leq \gamma$. Similarly, if $g = h_k$, we have $f = f^*$, and $[d_\varrho^t, d_{h_k}^t)$ is covered by $[d_f^t, d_g^t) = [d_{f^*}^t, d_{h_k}^t)$ and $(\chi, d_{\hat{f}}^t)$, since $d_{\hat{f}}^t \geq d_{f^*}^t$. Finally, consider the case of $f = f^*$ and $g = g^*$. Then the interval $[d_\varrho^t, d_{h_k}^t) \subseteq (\chi, \gamma)$ is covered by $(\chi, d_{f^*}^t) \cup [d_{f^*}^t, d_{g^*}^t) \cup [d_{g^*}^t, \gamma)$, which implies (5.55), because $d_{\hat{f}}^t \geq d_{f^*}^t$ and $d_{\hat{g}}^t \leq d_{g^*}^t$. $\square$

We now address the second part, where $B_{(i)}^{t+1}$ is changed into $B_{(ii)}^{t+1} = B_{(i)}^{t+1} \setminus \{h_{\tilde{a}}\} \cup \{\varrho\}$, and we show that $B_{(ii)}^{t+1}$ satisfies invariant (InvB). We consider the feasibility of

$B_{(ii)}^{t+1}$ first. Recall that in this part we assume $f = f^*$ or $g = g^*$. If $d_\varrho^t \geq d_{h_{\tilde{a}}}^t$, then the feasibility of $\mathcal{B}$ is clearly preserved after replacing $h_{\tilde{a}}$ by ϱ. Otherwise, $d_\varrho^t < d_{h_{\tilde{a}}}^t$, in which case, by Claim 5.13, we have $\tilde{a} = k$ and $\mathsf{pslack}(B_{(ii)}^{t+1}, \tau) = \mathsf{pslack}(B_{(i)}^{t+1}, \tau) - 1 \geq 0$ for $\tau \in [d_\varrho^t, d_{h_{\tilde{a}}}^t)$; while $\mathsf{pslack}(B_{(ii)}^{t+1}, \tau) = \mathsf{pslack}(B_{(i)}^{t+1}, \tau) \geq 0$ for $\tau \notin [d_\varrho^t, d_{h_{\tilde{a}}}^t)$, by Claim 5.12. Therefore, $B_{(ii)}^{t+1}$ is indeed feasible. In this part, we replace $h_{\tilde{a}}$ by ϱ in both $\mathcal{P}$ and $\mathcal{B}$ and $\mathcal{A}$ is unchanged, so invariant (InvB)(ii) is preserved as well.

Finally, we consider the third part of this stage, where the deadline decreases of $h_{\tilde{a}+1}, \ldots, h_k$ are implemented, converting $B_{(ii)}^{t+1}$ into $\ddot{B}_{\langle \tilde{a}, k \rangle}^{t+1}$. (Unlike in the first two parts, we perform this part also in the case $f = \varrho$ and $g = h_{\tilde{a}}$.) We only need to show that $\ddot{B}_{\langle \tilde{a}, k \rangle}^{t+1}$ remains feasible, since invariant (InvB)(ii) is not affected.

There is no deadline decrease in this part if $\tilde{a} = k$, so it remains to deal with the case when $\tilde{a} < k$. Recall that the algorithm will change the deadline $d_{h_i}^t$ to $d_{h_i}^{t+1} = \tau_{i-1}$ for each $i = \tilde{a}+1, \ldots, k$. These changes result in decreasing $\mathsf{pslack}(\mathcal{B})$ by one in each interval $[\tau_{\tilde{a}}, d_{h_{\tilde{a}+1}}^t), \ldots, [\tau_{k-1}, d_{h_k}^t)$. Other slots in $\mathcal{B}$ are not affected. These intervals are disjoint because $d_{h_i}^t \leq \tau_i$ for $i = 0, \ldots, k$. Thus, it suffices to show the following claim.

CLAIM 5.14. *Assume that* $\tilde{a} < k$. *Then* $\mathsf{pslack}(B_{(ii)}^{t+1}, \tau) \geq 1$ *for any* $\tau \in [\tau_{\tilde{a}}, \gamma)$.

Proof. We first show that $\mathsf{pslack}(P_{(ii)}^{t+1}, \tau) \geq 1$ for any $\tau \in [\tau_{\tilde{a}}, \gamma)$. To this end, we consider two cases.

The first case is when $\tau \in [\tau_{\tilde{a}}, d_\varrho^t)$. For such τ we have $\mathsf{pslack}(P_{(ii)}^{t+1}, \tau) \geq \mathsf{pslack}(P_{(i)}^{t+1}, \tau) + 1 \geq 1$, because replacing $h_{\tilde{a}}$ by ϱ causes $\mathsf{pslack}(\mathcal{P}, \tau)$ to increase by 1 for $\tau \in [d_{h_{\tilde{a}}}^t, d_\varrho^t) \supseteq [\tau_{\tilde{a}}, d_\varrho^t)$, and because $P_{(i)}^{t+1} = P_{\mathrm{ins}}^{t+1}$ is feasible.

The second case is when $\tau \in [d_\varrho^t, \gamma)$. In this case we argue as follows. Since for $\tilde{a} < k$ replacing $h_{\tilde{a}}$ by ϱ in $\mathcal{P}$ does not decrease any value $\mathsf{pslack}(\mathcal{P}, \tau)$ (i.e., there are no new tight slots in $\mathcal{P}$) and each slot $\tau \in [d_\varrho^t, \gamma)$ belongs to the segment of $P_{(ii)}^{t+1}$ ending at γ, we indeed have $\mathsf{pslack}(P_{(ii)}^{t+1}, \tau) \geq 1$ for any $\tau \in [d_\varrho^t, \gamma)$.

We thus have $\mathsf{pslack}(P_{(ii)}^{t+1}, \tau) \geq 1$ for all $\tau \in [\tau_{\tilde{a}}, \gamma)$, as claimed. To finish the proof, note that $d_{g^*}^t \leq \tau_{\tilde{a}}$ by the definitions of g^* and the terminal group $\langle \tilde{a}, k \rangle$, and that $A_{(ii)}^{t+1}(\tau_{\tilde{a}}, \gamma] = \emptyset$. Thus for $\tau \in [\tau_{\tilde{a}}, \gamma)$, we can apply Lemma 5.4(b) with $\zeta = \gamma$ to obtain $\mathsf{pslack}(B_{(ii)}^{t+1}, \tau) \geq \mathsf{pslack}(P_{(ii)}^{t+1}, \tau) \geq 1$, completing the proof. $\qquad\square$

Putting it all together and using Claim 5.14, we obtain that $\mathsf{pslack}(\ddot{B}_{\langle \tilde{a}, k \rangle}^{t+1}, \tau) \geq \mathsf{pslack}(B_{(ii)}^{t+1}, \tau) - 1 \geq 0$ for any $\tau \in [\tau_{\tilde{a}}, \gamma)$, while $\mathsf{pslack}(\ddot{B}_{\langle \tilde{a}, k \rangle}^{t+1}, \tau) = \mathsf{pslack}(B_{(ii)}^{t+1}, \tau) \geq 0$ for $\tau \notin [\tau_{\tilde{a}}, \gamma)$. This concludes the proof that invariant (InvB) holds after the stage for the terminal group.

Stage for a middle group. We now analyze the stage in which the segments represented by a middle group $\langle a, b \rangle$ of indices are processed. The current intermediate plan, backup plan, and adversary stash are $\dot{P}_{\langle a, b \rangle}^{t+1}$, $\dot{B}_{\langle a, b \rangle}^{t+1}$, and $\dot{A}_{\langle a, b \rangle}^{t+1}$, respectively, and we assume that they satisfy invariants (InvA) and (InvB). At this point, it holds that

(a) $h_a \in \dot{A}_{\langle a, b \rangle}^{t+1}$ by the definition of the middle group,

(b) $h_{b+1} \notin \dot{P}_{\langle a, b \rangle}^{t+1}$ as a result of the previous stage (recall that $b < k$ for a middle group), which implies that $h_{b+1} \notin \dot{A}_{\langle a, b \rangle}^{t+1}$ by invariant (InvA), and

(c) $h_{b+1} \notin \dot{B}_{\langle a, b \rangle}^{t+1}$, because h_{b+1} was removed from $\mathcal{P}$ in the previous stage (that is, we had $h_{b+1} \in P^t(\alpha, T]$, but now $h_{b+1} \notin \dot{P}_{\langle a, b \rangle}^{t+1}$), and because we do not add any packets to $\mathcal{B}(\alpha, T] \setminus \mathcal{P}$ while processing a leap step (see the "monotonicity of $\mathcal{B} \setminus \mathcal{P}$" invariant in section 5.5.3).

The changes of plan $\mathcal{P}$ in this stage involve removing h_a and adding back h_{b+1}; that is, $\dot{P}^{t+1}_{\langle a,b \rangle}$ is changed to $\ddot{P}^{t+1}_{\langle a,b \rangle} = \dot{P}^{t+1}_{\langle a,b \rangle} \setminus \{h_a\} \cup \{h_{b+1}\}$. We also decrease the deadlines of packets $h_{a+1}, \ldots, h_{b+1}$ and increase the weights of a subset of packets $h_{a+1}, \ldots, h_{b+1}$, according to the algorithm. Since h_a is removed from $\mathcal{P}$, we also need to remove it from $\mathcal{A}$ to preserve invariant (InvA). Similarly, as h_{b+1} is added back to $\mathcal{P}$ (and is not in $\mathcal{A}$), it needs to be added to $\mathcal{B}$ or to $\mathcal{A}$ so that invariant (InvB)(ii) continues to hold. The changes of $\mathcal{A}$ and $\mathcal{B}$ depend on two cases, which are described below.

Case L.(β, γ).M.1. There is an index $i \in [a+1, b+1]$ such that $w^t_{h_i} < \mu_{i-1}$; i.e., the algorithm (in line 11) increases the weight of h_i. We will remove h_a from $\mathcal{A}$ and add h_{b+1} to $\mathcal{B}$. To restore the feasibility of $\mathcal{B}$, we apply Lemma 5.5 for packet $g = h_a$ to identify a packet $f_a \in \dot{B}^{t+1}_{\langle a,b \rangle} \setminus \dot{P}^{t+1}_{\langle a,b \rangle}$ for which set $\dot{B}^{t+1}_{\langle a,b \rangle} \setminus \{f_a\} \cup \{h_a\}$ is feasible and $d^t_{f_a} > \mathsf{prevts}(\dot{P}^{t+1}_{\langle a,b \rangle}, d^t_{h_a})$. We then let

$$\ddot{A}^{t+1}_{\langle a,b \rangle} = \dot{A}^{t+1}_{\langle a,b \rangle} \setminus \{h_a\} \quad \text{and} \quad \ddot{B}^{t+1}_{\langle a,b \rangle} = \dot{B}^{t+1}_{\langle a,b \rangle} \setminus \{f_a\} \cup \{h_{b+1}\} .$$

With these changes, (InvA) is preserved, since also none of packets $h_{a+1}, \ldots, h_{b+1}$, whose deadlines are decreased, is in $\ddot{A}^{t+1}_{\langle a,b \rangle}$. We will show below that $\ddot{B}^{t+1}_{\langle a,b \rangle}$ is feasible, even with the deadlines of $h_{a+1}, \ldots, h_{b+1}$ decreased by the algorithm.

Deriving (5.43) *in Case* L.(β, γ).M.1. We need to take into account the addition of h_{b+1} in $\mathcal{B}$ and possible change of weights of some packets $h_{a+1}, \ldots, h_{b+1}$. Taking these changes into account, the terms on the left-hand side of (5.43) can be expressed or estimated as follows:

$$(5.56) \qquad \Delta_{\langle a,b \rangle} \Psi = \frac{1}{\phi} \left(\Delta^t w(h_{a+1}, \ldots, h_{b+1}) + w^t_{h_{b+1}} - w^t_{f_a} \right),$$

$$(5.57) \qquad \Delta^t w(h_{a+1}, \ldots, h_{b+1}) \leq \mu_a - w^t_{h_{b+1}},$$

$$(5.58) \qquad \mathsf{advcredit}^t_{\langle a,b \rangle} = w^t_{h_a} - \mathsf{minwt}(\dot{P}^{t+1}_{\langle a,b \rangle}, d^t_{h_a}) \leq w^t_{h_a} - \mu_a.$$

To justify (5.56), note that while $h_{b+1} \notin \dot{B}^{t+1}_{\langle a,b \rangle}$ is added to $\mathcal{B}$ with its new weight $w^{t+1}_{h_{b+1}}$, its weight increase of $w^{t+1}_{h_{b+1}} - w^t_{h_{b+1}}$ is already accounted for in $\Delta^t w(h_{a+1}, \ldots, h_{b+1})$.
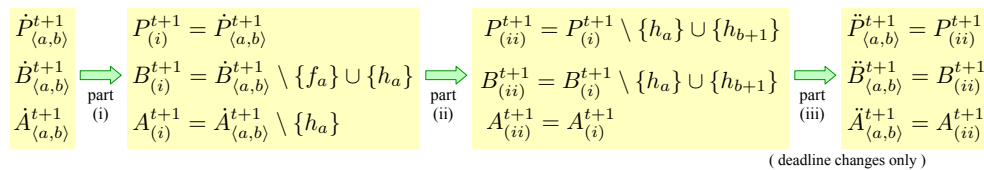
Inequality (5.57) can be derived from the case condition, which says that there is $i \in [a+1, b+1]$ such that $w^t_{h_i} < \mu_{i-1}$, so we just need to apply Lemma 5.11 with $a' = a + 1$ and $b' = b + 1$ to get (5.57).

Inequality (5.58) follows from the definition of μ_a as $\mu_a = \mathsf{minwt}(P^t, d^t_{h_a})$ and from the observation that $\mathsf{minwt}(\dot{P}^{t+1}_{\langle a,b \rangle}, d^t_{h_a}) \geq \mathsf{minwt}(P^t, d^t_{h_a})$, which in turn follows from $\mathsf{nextts}(\dot{P}^{t+1}_{\langle a,b \rangle}, d^t_{h_a}) = \mathsf{nextts}(P^t, d^t_{h_a})$ and $\dot{P}^{t+1}_{\langle a,b \rangle}[t+1, \eta] = P^t[t, \eta] \setminus \{\ell\}$, where $\eta = \mathsf{nextts}(\dot{P}^{t+1}_{\langle a,b \rangle}, d^t_{h_a})$. (The equality $\mathsf{nextts}(\dot{P}^{t+1}_{\langle a,b \rangle}, d^t_{h_a}) = \mathsf{nextts}(P^t, d^t_{h_a})$ holds because of the "locality of changes" invariant from section 5.5.3.)

Next, we claim that

$$(5.59) \qquad w^t_{f_a} \leq w^t_{h_{b+1}}.$$

Indeed, f_a is not in P^t since, by its definition, $f_a \notin \dot{P}^{t+1}_{\langle a,b \rangle}$ and $d^t_{f_a} > \mathsf{prevts}(\dot{P}^{t+1}_{\langle a,b \rangle} d^t_{h_a}) \geq \beta$, so f_a could not have been removed from $\mathcal{P}$ in this step. (Note that h_{b+1} is temporarily removed from $\mathcal{P}$, but it cannot be equal to f_a because $h_{b+1} \notin \dot{B}^{t+1}_{\langle a,b \rangle}$). Thus $w^t_{f_a} \leq w^t_{\varrho}$, because ϱ is the heaviest pending packet not in P^t with deadline after β

$$\dot{P}^{t+1}_{\langle a,b\rangle} \quad \xRightarrow[\text{(i)}]{\text{part}} \quad \begin{array}{l} P^{t+1}_{(i)} = \dot{P}^{t+1}_{\langle a,b\rangle} \\ B^{t+1}_{(i)} = \dot{B}^{t+1}_{\langle a,b\rangle} \setminus \{f_a\} \cup \{h_a\} \\ A^{t+1}_{(i)} = \dot{A}^{t+1}_{\langle a,b\rangle} \setminus \{h_a\} \end{array} \quad \xRightarrow[\text{(ii)}]{\text{part}} \quad \begin{array}{l} P^{t+1}_{(ii)} = P^{t+1}_{(i)} \setminus \{h_a\} \cup \{h_{b+1}\} \\ B^{t+1}_{(ii)} = B^{t+1}_{(i)} \setminus \{h_a\} \cup \{h_{b+1}\} \\ A^{t+1}_{(ii)} = A^{t+1}_{(i)} \end{array} \quad \xRightarrow[\text{(iii)}]{\text{part}} \quad \begin{array}{l} \ddot{P}^{t+1}_{\langle a,b\rangle} = P^{t+1}_{(ii)} \\ \ddot{B}^{t+1}_{\langle a,b\rangle} = B^{t+1}_{(ii)} \\ \ddot{A}^{t+1}_{\langle a,b\rangle} = A^{t+1}_{(ii)} \end{array}$$

(deadline changes only)

FIG. 5.8. *Partition of the stage for the middle group in Case* L.$(\beta,\gamma]$M.1. *into three parts.*

and $f_a \notin P^t$ is pending in step t. By Lemma B.1(b) we have $w^t_\varrho \le w^t_{h_{b+1}}$, which implies that $w^t_{f_a} \le w^t_{h_{b+1}}$. (As an additional clarification, the argument in this paragraph relies tacitly on two properties of earlier changes in processing the transmission event: One, no packets with deadlines after α are added to $\mathcal{B} \setminus \mathcal{P}$, and two, the tight slots of $\dot{P}^{t+1}_{\langle a,b\rangle}$ in the interval $(\beta, \tau_b]$ are the same as tight slots in P^t; cf. section 5.5.3.)

Then we prove (5.43) as follows:

$$\Delta_{\langle a,b\rangle}\Psi - \phi\,\Delta^t w(h_{a+1},\ldots,h_{b+1}) - \mathsf{advcredit}^t_{\langle a,b\rangle}$$

$$\ge \frac{1}{\phi}\left(\Delta^t w(h_{a+1},\ldots,h_{b+1}) + w^t_{h_{b+1}} - w^t_{f_a}\right)$$

$$- \phi\,\Delta^t w(h_{a+1},\ldots,h_{b+1}) - (w^t_{h_a} - \mu_a)$$

$$= \frac{1}{\phi}\left(w^t_{h_{b+1}} - w^t_{f_a}\right) - \Delta^t w(h_{a+1},\ldots,h_{b+1}) - w^t_{h_a} + \mu_a$$

$$(5.60) \qquad \ge -(\mu_a - w^t_{h_{b+1}}) - w^t_{h_a} + \mu_a$$

$$= -w^t_{h_a} + w^t_{h_{b+1}},$$

where inequality (5.60) follows from (5.59) and (5.57).

Invariant (InvB) in Case L.$(\beta,\gamma]$.M.1. We claim that after the stage for the middle group $\langle a,b\rangle$, invariant (InvB) holds, which we show by partitioning this stage into three parts:

(i) We first let $B^{t+1}_{(i)} = \dot{B}^{t+1}_{\langle a,b\rangle} \setminus \{f_a\} \cup \{h_a\}$.

(ii) Second, we modify $B^{t+1}_{(i)}$ to obtain set $B^{t+1}_{(ii)} = B^{t+1}_{(i)} \setminus \{h_a\} \cup \{h_{b+1}\}$.

(iii) Finally, we implement the decrease of the deadlines of packets $h_{a+1},\ldots,h_{b+1}$, changing $B^{t+1}_{(ii)}$ to $\ddot{B}^{t+1}_{\langle a,b\rangle}$.

In each part, we also appropriately modify the sets $\mathcal{P}$ and $\mathcal{A}$, as illustrated in Figure 5.8.

That $B^{t+1}_{(i)}$ is feasible follows from the choice of f_a and Lemma 5.5. Since $d^t_{h_a} < d^t_{h_{b+1}}$, $B^{t+1}_{(ii)}$ is also a feasible set of packets, and moreover, $\mathsf{pslack}(B^{t+1}_{(ii)}, \tau) = \mathsf{pslack}(B^{t+1}_{(i)}, \tau) + 1 \ge 1$ for any $\tau \in [d^t_{h_a}, d^t_{h_{b+1}})$, by the feasibility of $B^{t+1}_{(i)}$. Finally, we argue that $\ddot{B}^{t+1}_{\langle a,b\rangle}$ remains feasible. The deadline decreases for packets $h_{a+1},\ldots,h_{b+1}$ decrease $\mathsf{pslack}(\mathcal{B})$ by one for slots in each of the disjoint intervals $[\tau_a, d^t_{h_{a+1}}),\ldots,[\tau_b, d^t_{h_{b+1}})$. As $d^t_{h_a} \le \tau_a$, all these intervals are contained in $[d^t_{h_a}, d^t_{h_{b+1}})$; hence, by the earlier observation, it holds that $\mathsf{pslack}(\ddot{B}^{t+1}_{\langle a,b\rangle}, \tau) \ge \mathsf{pslack}(B^{t+1}_{(ii)}, \tau) - 1 \ge 0$ for any $\tau \in [d^t_{h_a}, d^t_{h_{b+1}})$. For $\tau \notin [d^t_{h_a}, d^t_{h_{b+1}})$, the values of $\mathsf{pslack}(\mathcal{B})$ do not change. It follows that $\ddot{B}^{t+1}_{\langle a,b\rangle}$ is feasible; that is, invariant (InvB)(i) holds after the stage. That invariant (InvB)(ii) holds as well follows directly from the definitions of $\ddot{P}^{t+1}_{\langle a,b\rangle}$, $\ddot{B}^{t+1}_{\langle a,b\rangle}$, and $\ddot{A}^{t+1}_{\langle a,b\rangle}$.

Case L.$(\beta,\gamma]$.M.2. For all indices $i \in [a+1, b+1]$ we have $w^t_{h_i} \geq \mu_{i-1}$. Thus in this case the algorithm does not increase the weights of packets $h_{a+1}, \ldots, h_{b+1}$, i.e., $\Delta^t w(h_{a+1}, \ldots, h_{b+1}) = 0$.

Before describing the changes in $\mathcal{A}$, we argue that $h_{a+1} \notin \dot{A}^{t+1}_{\langle a,b \rangle}$. This is trivial if $b > a$, because then packets $h_{a+1}, \ldots, h_b$ are not in $\mathcal{A}$ before processing the groups. Otherwise, we have $a = b$. Since in the stages, we process the groups backwards with respect to time slots, the stage for the group $\langle b+1, b' \rangle$ containing index $b+1$ is already completed, and it guarantees that h_{b+1} is not in $\ddot{P}^{t+1}_{\langle b+1,b' \rangle} = \dot{P}^{t+1}_{\langle a,b \rangle}$ and thus not in $\ddot{A}^{t+1}_{\langle b+1,b' \rangle} = \dot{A}^{t+1}_{\langle a,b \rangle}$ by invariant (InvA), as claimed. (See also the overview of the partition into stages, earlier in the description of Case L.$(\beta,\gamma]$.I.2.)

We now change $\mathcal{A}$ as follows: we remove h_a from $\mathcal{A}$, and we put h_{a+1} in the former slot of h_a in $\dot{A}^{t+1}_{\langle a,b \rangle}$, that is $\ddot{A}^{t+1}_{\langle a,b \rangle} = \dot{A}^{t+1}_{\langle a,b \rangle} \setminus \{h_a\} \cup \{h_{a+1}\}$. This is valid because, as we showed above, $h_{a+1} \notin \dot{A}^{t+1}_{\langle a,b \rangle}$, and the new deadline of h_{a+1} is τ_a. Then $\mathsf{advcredit}^t_{\langle a,b \rangle} = w^t_{h_a} - w^t_{h_{a+1}}$, as the weight of h_{a+1} does not change. Since packets $h_{a+2}, \ldots, h_{b+1}$ are not in $\ddot{A}^{t+1}_{\langle a,b \rangle}$, the decreases of their deadlines do not affect $\mathcal{A}$. After these changes, invariant (InvA) will be preserved.

Regarding $\mathcal{B}$, recall that $h_{b+1} \notin \dot{B}^{t+1}_{\langle a,b \rangle}$, $h_a \in \dot{A}^{t+1}_{\langle a,b \rangle}$, and $h_a \notin \ddot{P}^{t+1}_{\langle a,b \rangle}$. Also, if $b > a$, then $h_{a+1} \in \dot{B}^{t+1}_{\langle a,b \rangle}$. Then, since h_{a+1} is newly added to $\mathcal{A}$, we let $\ddot{B}^{t+1}_{\langle a,b \rangle} = \dot{B}^{t+1}_{\langle a,b \rangle} \cup \{h_{b+1}\} \setminus \{h_{a+1}\}$; it is possible here that $h_{a+1} = h_{b+1}$, in which case $\mathcal{B}$ does not change.

Deriving (5.43) *in Case* L.$(\beta,\gamma]$.M.2. In this case, the terms on the left-hand side of (5.43) are

$$\Delta_{\langle a,b \rangle} \Psi = \frac{1}{\phi}\left(-w^t_{h_{a+1}} + w^t_{h_{b+1}}\right),$$
$$\Delta^t w(h_{a+1}, \ldots, h_{b+1}) = 0,$$
$$\mathsf{advcredit}^t_{\langle a,b \rangle} = w^t_{h_a} - w^t_{h_{a+1}}$$

with all formulas following directly from the definition and the fact that the weights remain unchanged. Plugging these in, we obtain

$$\Delta_{\langle a,b \rangle} \Psi - \phi\, \Delta^t w(h_{a+1}, \ldots, h_{b+1}) - \mathsf{advcredit}^t_{\langle a,b \rangle}$$
$$= \frac{1}{\phi}\left(-w^t_{h_{a+1}} + w^t_{h_{b+1}}\right) - \phi \cdot 0 - (w^t_{h_a} - w^t_{h_{a+1}})$$
$$= -w^t_{h_a} + \frac{1}{\phi^2} w^t_{h_{a+1}} + \frac{1}{\phi} w^t_{h_{b+1}}$$
(5.61)
$$\geq -w^t_{h_a} + \frac{1}{\phi^2} w^t_{h_{b+1}} + \frac{1}{\phi} w^t_{h_{b+1}}$$
$$= -w^t_{h_a} + w^t_{h_{b+1}},$$

where inequality (5.61) follows from $w^t_{h_{a+1}} \geq w^t_{h_{b+1}}$, by Lemma B.1(b) and $a \leq b$.

Invariant (InvB) *in Case* L.$(\beta,\gamma]$.M.2. We show that $\ddot{B}^{t+1}_{\langle a,b \rangle}$ is feasible by considering first an intermediate backup plan $B^{t+1}_{(i)} = \dot{B}^{t+1}_{\langle a,b \rangle} \cup \{h_{b+1}\} \setminus \{h_{a+1}\}$, computed with packets $h_{a+1}, \ldots, h_{b+1}$ having their deadlines unchanged. Since $B^{t+1}_{(i)}$ is obtained by replacing h_{a+1} by h_{b+1} in $\mathcal{B}$, and $d^t_{h_{a+1}} \leq d^t_{h_{b+1}}$, $B^{t+1}_{(i)}$ is feasible; moreover, it satisfies

$\mathsf{pslack}(B_{(i)}^{t+1}, \tau) = \mathsf{pslack}(\dot{B}_{\langle a,b\rangle}^{t+1}, \tau) + 1 \geq 1$ for any $\tau \in [d_{h_{a+1}}^t, d_{h_{b+1}}^t)$. Next, to prove the feasibility of $\ddot{B}_{\langle a,b\rangle}^{t+1}$, we decrease the deadlines of packets $h_{a+1}, \ldots, h_{b+1}$. This decreases $\mathsf{pslack}(\mathcal{B})$ by one for slots in disjoint intervals $[\tau_{a+1}, d_{h_{a+2}}^t), \ldots, [\tau_b, d_{h_{b+1}}^t)$. (We do not consider h_{a+1} here because this packet was already removed from $\mathcal{B}$.) These intervals are contained in $[d_{h_{a+1}}^t, d_{h_{b+1}}^t)$, and we thus have $\mathsf{pslack}(\ddot{B}_{\langle a,b\rangle}^{t+1}, \tau) \geq \mathsf{pslack}(B_{(i)}^{t+1}, \tau) - 1 \geq 0$ for $\tau \in [d_{h_{a+1}}^t, d_{h_{b+1}}^t)$. For $\tau \notin [d_{h_{a+1}}^t, d_{h_{b+1}}^t)$, the values of $\mathsf{pslack}(\mathcal{B})$ do not change. Therefore, $\ddot{B}_{\langle a,b\rangle}^{t+1}$ is feasible, showing (InvB)(i). Part (ii) of invariant (InvB) is implied by the changes of sets $\mathcal{P}, \mathcal{A}$, and $\mathcal{B}$.

This completes the proof that invariants (InvA) and (InvB) are preserved after the stage for the middle group $\langle a, b\rangle$, and that inequality (5.43) holds for this stage.

Stage for the initial group. The initial group $\langle 0, \tilde{b}\rangle$ is defined when either $l > 0$ and $i_1 > 0$ or if $l = 0$ and $\tilde{a} > 0$. In either case, we have $h_0, \ldots, h_{\tilde{b}} \notin \dot{A}_{\langle 0,\tilde{b}\rangle}^{t+1}$.

In the stage in which this group is processed, we change plan $\mathcal{P}$ by removing $h_0 = p$ and adding $h_{\tilde{b}+1}$, so $\widetilde{P}^{t+1} = \dot{P}_{\langle 0,\tilde{b}\rangle}^{t+1} \setminus \{p\} \cup \{h_{\tilde{b}+1}\}$. (Recall that $\widetilde{P}^{t+1} = \ddot{P}_{\langle 0,\tilde{b}\rangle}^{t+1}$ and the corresponding relation holds for $\mathcal{A}$ and $\mathcal{B}$, since we process the stage for the initial group after all other groups.) We also decrease the deadlines of packets $h_1, \ldots, h_{\tilde{b}+1}$, and for some we increase their weights, according to the algorithm. We do not change $\mathcal{A}$; thus $\widetilde{A}^{t+1} = \dot{A}_{\langle 0,\tilde{b}\rangle}^{t+1}$ and $\mathsf{advcredit}_{\langle 0,\tilde{b}\rangle}^{t+1} = 0$. Since packets $h_1, \ldots, h_{\tilde{b}+1}$ are not in $\dot{A}_{\langle 0,\tilde{b}\rangle}^{t+1}$, it follows that invariant (InvA) is preserved. To update $\mathcal{B}$, we let $\widetilde{B}^{t+1} = \dot{B}_{\langle 0,\tilde{b}\rangle}^{t+1} \setminus \{p\} \cup \{h_{\tilde{b}+1}\}$ (recall that $h_{\tilde{b}+1} \notin \dot{B}_{\langle 0,\tilde{b}\rangle}^{t+1}$).

Deriving (5.43) *for the initial group.* The terms on the left-hand side of (5.43) can be expressed or estimated as follows:

$$(5.62) \qquad \Delta_{\langle 0,\tilde{b}\rangle}\Psi = \frac{1}{\phi}\left(\Delta^t w(h_1, \ldots, h_{\tilde{b}+1}) - w_p^t + w_{h_{\tilde{b}+1}}^t\right),$$

$$(5.63) \qquad \Delta^t w(h_1, \ldots, h_{\tilde{b}+1}) \leq \frac{1}{\phi^2}(w_p^t - w_{h_{\tilde{b}+1}}^t),$$

$$(5.64) \qquad \mathsf{advcredit}_{\langle 0,\tilde{b}\rangle}^t = 0.$$

Equations (5.62) and (5.64) follow directly from the definitions and the case assumption.

To justify (5.63), we consider two cases. If the algorithm does not increase the weight of any of the packets $h_1, \ldots, h_{\tilde{b}+1}$, then $\Delta^t w(h_1, \ldots, h_{\tilde{b}+1}) = 0 \leq \frac{1}{\phi^2}(w_p^t - w_{h_{\tilde{b}+1}}^t)$, by Lemma B.1(b). Otherwise, there is $i \in [1, \tilde{b}+1]$ such that $w_{h_i}^t < \mu_{i-1}$, and applying Lemma 5.11 with $a' = 1$ and $b' = \tilde{b} + 1$, we obtain

$$\Delta^t w(h_1, \ldots, h_{\tilde{b}+1}) \leq \mu_0 - w_{h_{\tilde{b}+1}}^t$$

$$(5.65) \qquad\qquad\qquad\qquad \leq \frac{1}{\phi^2}\, w_p^t + \frac{1}{\phi}\, w_\varrho^t - w_{h_{\tilde{b}+1}}^t$$

$$(5.66) \qquad\qquad\qquad\qquad \leq \frac{1}{\phi^2}(w_p^t - w_{h_{\tilde{b}+1}}^t),$$

where inequality (5.65) uses the bound $\mu_0 \leq \frac{1}{\phi^2}\, w_p^t + \frac{1}{\phi}\, w_\varrho^t$, which follows from (5.26), and inequality (5.66) uses $w_\varrho^t \leq w_{h_{\tilde{b}+1}}^t$, which follows from Lemma B.1(b). Thus (5.63) holds.

The calculation showing (5.43) is now quite simple:

$$
\begin{aligned}
\Delta_{\langle 0,\tilde{b}\rangle}\Psi &- \phi\,\Delta^t w(h_1,\ldots,h_{\tilde{b}+1}) - \mathsf{advcredit}^t_{\langle 0,\tilde{b}\rangle} \\
&= \frac{1}{\phi}\left(\Delta^t w(h_1,\ldots,h_{\tilde{b}+1}) - w^t_p + w^t_{h_{\tilde{b}+1}}\right) - \phi\,\Delta^t w(h_1,\ldots,h_{\tilde{b}+1}) - 0 \\
&= \frac{1}{\phi}\left(-w^t_p + w^t_{h_{\tilde{b}+1}}\right) - \Delta^t w(h_1,\ldots,h_{\tilde{b}+1}) \\
&\geq \frac{1}{\phi}\left(-w^t_p + w^t_{h_{\tilde{b}+1}}\right) - \frac{1}{\phi^2}\left(w^t_p - w^t_{h_{\tilde{b}+1}}\right) \\
&= -w^t_p + w^t_{h_{\tilde{b}+1}},
\end{aligned}
$$

(5.67)

where inequality (5.67) holds by (5.63).

Invariant (InvB) after the stage for the initial group. That invariant (InvB)(ii) is preserved follows directly from the formulas for $\widetilde{P}^{t+1}$, $\widetilde{B}^{t+1}$, and $\widetilde{A}^{t+1}$.

To show invariant (InvB)(i), that is the feasibility of $\widetilde{B}^{t+1}$, we consider an intermediate backup plan $B^{t+1}_{(i)} = \dot{B}^{t+1}_{\langle 0,\tilde{b}\rangle} \setminus \{p\} \cup \{h_{\tilde{b}+1}\}$, where the deadlines of packets $h_1,\ldots,h_{\tilde{b}+1}$ remain unchanged. Replacing p by $h_{\tilde{b}+1}$ in $\mathcal{B}$ preserves feasibility as $d^t_p < d^t_{h_{\tilde{b}+1}}$. So $B^{t+1}_{(i)}$ is feasible; moreover, it satisfies $\mathsf{pslack}(B^{t+1}_{(i)},\tau) = \mathsf{pslack}(\dot{B}^{t+1}_{\langle 0,\tilde{b}\rangle},\tau) + 1 \geq 1$ for any $\tau \in [d^t_p, d^t_{h_{\tilde{b}+1}})$.

We now decrease the deadlines of packets $h_1,\ldots,h_{\tilde{b}+1}$, as in the algorithm. This decreases $\mathsf{pslack}(\mathcal{B},\tau)$ by one for slots τ in disjoint intervals $[\tau_0, d^t_{h_1}),\ldots,[\tau_{\tilde{b}}, d^t_{h_{\tilde{b}+1}})$. These intervals are contained in $[d^t_p, d^t_{h_{\tilde{b}+1}})$, and we thus have $\mathsf{pslack}(\widetilde{B}^{t+1},\tau) \geq \mathsf{pslack}(B^{t+1}_{(i)},\tau) - 1 \geq 0$ for $\tau \in [d^t_p, d^t_{h_{\tilde{b}+1}})$. The values of $\mathsf{pslack}(\mathcal{B})$ are not affected for slots $\tau \notin [d^t_p, d^t_{h_{\tilde{b}+1}})$. Therefore, $\widetilde{B}^{t+1}$ is feasible; that is, invariant (InvB)(i) holds after the stage for the initial group and after the algorithm's step as well.

Summary of Case L.(β,γ).I.2. In this case, we processed changes in the interval $(\beta,\gamma]$ when $\varrho \in B^{t+1}_{\mathrm{ins}}$ or $A^{t+1}_{\mathrm{ins}}(\beta,\gamma] \neq \emptyset$. We divided the segments in this interval that contain packets h_i into groups, and each group was processed in a separate stage, in reverse order (i.e., backwards with respect to time slots). For each stage, we proved that (i) invariants (InvA) and (InvB) are preserved, and that (ii) inequality (5.43) holds for this stage. The last stage corresponds to a group $\langle 0,\tilde{b}\rangle$ (that may be of any type), and we proved that after this last stage the backup plan $\widetilde{B}^{t+1} = \ddot{B}^{t+1}_{\langle 0,\tilde{b}\rangle}$ satisfies invariant (InvB), and that the adversary stash $\widetilde{A}^{t+1} = \ddot{A}^{t+1}_{\langle 0,\tilde{b}\rangle}$ satisfies invariant (InvA). All inequalities (5.43) (one for each group) imply the key inequality (5.24), which is needed for amortized analysis.

This concludes the analysis of an iterated leap step and thus also the analysis of an event of packet transmission. Together with the analysis of packet arrivals given in section 5.4, this completes the proof of ϕ-competitiveness of Algorithm PlanM, as described in section 5.3.

6. Final comments. Our result establishes a tight bound of ϕ on the competitive ratio of PacketSchD in the deterministic case, settling a long-standing open problem. There is still a number of intriguing open problems about online packet scheduling that deserve further study.

Of these open problems, the most prominent one is to establish tight bounds for randomized algorithms for PacketSchD. The best known upper bound to date is $\mathrm{e}/(\mathrm{e}-1) \approx 1.582$ [4, 9, 8, 21]. This ratio is achieved by a memoryless algorithm and it holds even against an adaptive adversary. No better upper bound for the oblivious

adversary is known. (In fact, against the oblivious adversary, the same ratio can be attained for a more general problem of online vertex-weighted bipartite matching [1, 13].) The best lower bounds, to the best of our knowledge, are $4/3 \approx 1.333$ [8] against the adaptive adversary, and 1.25 [10] against the oblivious one, respectively. (Both lower bounds use 2-bounded instances and are in fact tight for 2-bounded instances.) Closing these gaps would provide insight into the power of randomization in packet scheduling.

In practical settings, the choice of the packet to transmit needs to be made at speed matching the link's rate, so the running time and simplicity of the scheduling algorithm are important factors. This motivates the study of *memoryless* algorithms for PacketSchD, as those algorithms tend to be easy to implement and fast. All known upper bounds for competitive randomized algorithms we are aware of are achieved by memoryless algorithms (see [15]). For deterministic algorithms, the only memoryless one that beats ratio 2 is the 1.893-competitive algorithm in [14]. The main question here is whether the ratio of ϕ can be achieved by a memoryless algorithm.

Among other models for packet scheduling, optimal competitiveness for the FIFO model is still wide open, both in the deterministic and randomized cases. We refer the reader to the (still mostly current) survey of Goldwasser [15], who provides a thorough discussion of various models for online packet scheduling and related open problems.

Appendix A. Properties of optimal plans. A computation of an online algorithm can be thought of as a sequence of events, with each event being either a packet arrival or transmitting a packet (which includes incrementing the current time). In this appendix, we give formal statements of "plan-update lemmas," which explain how the structure of the optimal plan changes in response to these events.

The proofs are based on analyzing Algorithm ComputePlan, given in section 3, for computing the optimal plan in a greedy fashion. In the proofs, we will refer to the packets j added to the plan in line 5 of this algorithm as *admitted* to the plan, and to the remaining packets as *rejected*.

We will use notation $\mathcal{U}$ for the current set of pending packets, which is the default argument for Algorithm ComputePlan. For a set $Z \subseteq \mathcal{U}$ of pending packets and a packet j, by $Z_{\rhd j}$ we will denote the set of packets in Z that are (strictly) heavier than j and by $Z_{\unrhd j}$ the set of packets in Z that are at least as heavy as j. (In most cases we will use this notation when Z is the optimal plan and $j \in \mathcal{U}$, but occasionally we apply it in other contexts.) If P is an optimal plan, then $P_{\rhd j}$ can be thought of, equivalently, as the set of all packets that are admitted to set X in Algorithm ComputePlan when all packets heavier than j have already been considered (and similarly for $P_{\unrhd j}$).

In the condition of admitting $j \in \mathcal{U}$ in Algorithm ComputePlan (line 4) it is sufficient to check if $\mathsf{pslack}(X \cup \{j\}, \tau) \geq 0$ only for $\tau \geq d_j$, because for smaller values of τ the addition of j does not change the value of $\mathsf{pslack}(\tau)$. This condition can be equivalently stated as $\mathsf{pslack}(X, \tau) \geq 1$ for each $\tau \geq d_j$. Note also that, letting P be the computed optimal plan, right before processing j we have $X = P_{\rhd j}$, and right after processing j we have $X = P_{\unrhd j}$.

The following observation summarizes some simple but useful properties of plans.

Observation A.1. Let $\mathcal{U}$ be the set of packets pending at time t, and let P be the optimal plan at time t. Then the following hold:

(a) P depends only on the ordering of packets in $\mathcal{U}$ with respect to weights (not on the actual values of the weights).

(b) P does not change if the weight of any packet in P is increased. It also does not change if the weight of any packet in $\mathcal{U} \setminus P$ is decreased.

(c) For a packet $j \in P$, let $\beta = \mathsf{prevts}(P, d_j)$. Then for any slot $\xi \in (\beta, d_j]$ there is a schedule of P in which j is scheduled in slot ξ.

(d) If $j \in \mathcal{U}$ is a packet and ζ a slot such that $\mathsf{pslack}(P_{\trianglerighteq j}, \zeta) = 0$, then $P_{\trianglerighteq j}[t, \zeta] = P[t, \zeta]$, and consequently $\mathsf{pslack}(P_{\trianglerighteq j}, \tau) = \mathsf{pslack}(P, \tau)$ for all $\tau \in [t, \zeta]$.

Proof. Part (a) follows directly from the correctness of Algorithm ComputePlan that computes P. Part (b) is also straightforward: P is heavier than any other plan (feasible set of pending packets) for $\mathcal{U}$. If the weight of $j \in P$ is increased by $\delta > 0$, the weight of P will increase by δ, and as the weight of any other plan X increases by at most δ, plan P remains to be optimal. The proof for the second part of (b) is similar.

To show (c), compute the desired schedule as follows. First, schedule packets in P in the canonical order. In this schedule, j will be scheduled at some slot $\xi' \in (\beta, d_j]$. If $\xi' = \xi$, we are done. If $\xi' \in (\beta, \xi)$, shift the packets in $[\xi' + 1, \xi]$ to the left by one slot and schedule j at ξ. The last case is when $\xi' \in (\xi, d_j]$. In this case, we use the fact that $\mathsf{pslack}(P, \tau) \geq 1$ for all $\tau \in (\beta, d_j)$, which means that for any $\tau \in (\beta, d_j)$ the number of packets in P with deadline at most τ is strictly less than $\tau - t + 1$, the number of slots in $[t, \tau]$. Then, by the canonical ordering, we have that none of the packets scheduled in $[\xi, \xi' - 1]$ is scheduled at its deadline. Therefore, we can shift these packets to the right by one slot, which allows us to schedule j at slot ξ.

Part (d) also follows from the correctness of ComputePlan: Consider its run that produces P. Immediately after packet j is processed we have $\mathsf{pslack}(X_{\trianglerighteq j}, \zeta) = 0$, i.e., slot ζ is already tight. Then no packet lighter than j with deadline at most ζ will be later admitted to X. Thus, $P[t, \zeta] = X[t, \zeta]$ is already fixed at that point, and so are its slack values. $\qquad \square$

In the rest of this section we now deal with plan-update lemmas. First, in section A.1 we explain how the optimal plan changes in response to packet arrivals. In section A.2 we detail how to update the optimal plan after some arbitrary packet from the plan is transmitted. The plan properties in these two sections are not specific to our algorithm; they address general properties of optimal plans. Later in section B, we give the plan-update lemma specifically for a leap step of Algorithm *PlanM*.

A.1. Updating the optimal plan after a packet arrival. In this section, we address the question of how the optimal plan changes when a new packet is released. So consider some step t, and let $\mathcal{U}$ be the current set of pending packets and P the optimal plan for $\mathcal{U}$. Suppose that now another packet s is released, and let $\mathcal{U}' = \mathcal{U} \cup \{s\}$ be the new set of pending packets. Denote by Q the optimal plan for $\mathcal{U}'$. The lemma below explains how to determine Q from P and it gives its important properties. (See Figure A.1 for an illustration.)

LEMMA A.2 (The Plan-Update Lemma for Packet Arrival). *Consider the scenario above, involving the arrival of a new packet s. Let $\gamma = \mathsf{nextts}(P, d_s)$ and define f to be the packet in $P[t, \gamma]$ with $w_f = \mathsf{minwt}(P, d_s)$; in other words, the minimum-weight packet in $P[t, \gamma]$. Also, let $\beta = \mathsf{prevts}(P, d_f)$. By definition, we have $\beta < \gamma$ and $d_f, d_s \in (\beta, \gamma]$.*

If $w_s < w_f$, then $Q = P$; i.e., s is not added to the optimal plan.

If $w_s > w_f$, then $Q = P \cup \{s\} \setminus \{f\}$; i.e., s is added to the optimal plan replacing f. In this case the following properties hold:
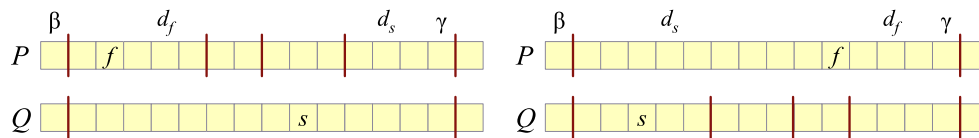
Fig. A.1. *An illustration of changes of tight slots and segments in Case* (b1) *of Lemma* A.2 *on the left and in Case* (b2) *on the right. The segments are separated by vertical dark-red line segments.*

(a) $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau)$ *for* $\tau \in [t, \min(d_f, d_s)) \cup [\max(d_f, d_s), T]$.

(b) *For slots* $\tau \in [\min(d_f, d_s), \max(d_f, d_s))$, *there are two cases:*

    (b1) *If* $d_s > d_f$, *then* $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) + 1$ *for* $\tau \in [d_f, d_s)$. *It follows that all segments of* P *in the interval* $(\beta, \gamma]$ *get merged into one segment* $(\beta, \gamma]$ *of* Q.

    (b2) *If* $d_s \leq d_f$, *then* $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) - 1$ *for* $\tau \in [d_s, d_f)$, *and thus there might be new tight slots in* $[d_s, d_f)$. *This case happens only when* $(\beta, \gamma]$ *is itself a segment of* P *and it contains both* d_f *and* d_s. *This segment may get split into multiple segments of* Q.

(c) *For any slot* $\tau \geq t$, $\mathsf{minwt}(Q, \tau) \geq \mathsf{minwt}(P, \tau)$.

We remark that packet f satisfies $w_f \neq w_s$, by assumption (A2).

Proof. We start by proving how the set of packets in the optimal plan changes. Let $\mathcal{U}$ be the set of packets pending before s is released and $\mathcal{U}' = \mathcal{U} \cup \{s\}$ be the set of pending packets after the release of s. We consider two "parallel" runs of Algorithm ComputePlan, one on $\mathcal{U}$ to compute P and the other on $\mathcal{U}'$ to compute Q.

Suppose first that $w_s < w_f$. Then both runs will be identical until right after the time when f is considered. Thus $P_{\unrhd f} = Q_{\unrhd f}$. Moreover, as f is the minimum-weight packet in P with $d_f \leq \gamma = \mathsf{nextts}(P, d_s)$, we have $P[t, \gamma] \subseteq P_{\unrhd f}$; that is, all packets in P with deadline at most γ are in $P_{\unrhd f}$. Thus $\mathsf{pslack}(Q_{\unrhd f}, \gamma) = \mathsf{pslack}(P_{\unrhd f}, \gamma) = \mathsf{pslack}(P, \gamma) = 0$. Therefore, as $d_s \leq \gamma$ and as s is considered after f, packet s will not be admitted, which implies that both runs produce the same plan $P = Q$.

For the rest of the proof, we assume that $w_s > w_f$. To prove that $Q = P \cup \{s\} \setminus \{f\}$, based on Observation A.1(b), without loss of generality we can assume that $w_s = w_f + \varepsilon$ for a tiny $\varepsilon > 0$, so that s immediately precedes f in the ordering of $\mathcal{U}'$ by decreasing weights.

Specifically, since $f \in P$, P does not change if f's weight is increased to $w_s - \varepsilon$. We will show that $Q = P \cup \{s\} \setminus \{f\}$ with this increased weight of f. But as then $f \notin Q$, decreasing f's weight back to its original value does not change Q.

With the assumption about w_f, the run on $\mathcal{U}'$ before s is considered will be identical to the run on $\mathcal{U}$ before f is considered; thus $Q_{\rhd s} = P_{\rhd f}$. In the next step of these runs, we consider packet f for P and packet s for Q. We now have two cases.

Suppose that $d_s > d_f$. Then, since f is admitted to P, we have $\mathsf{pslack}(P_{\rhd f}, \tau) \geq 1$ for $\tau \geq d_f$, which, by the paragraph above, implies that $\mathsf{pslack}(Q_{\rhd s}, \tau) \geq 1$ for $\tau \geq d_s$; and thus s will be admitted to Q. As $d_f, d_s \leq \gamma$, we also have $\mathsf{pslack}(Q_{\unrhd s}, \gamma) = \mathsf{pslack}(P_{\unrhd f}, \gamma) = 0$. This means that in the run on $\mathcal{U}'$ packet f, which is considered next, will be rejected. From this point on, both runs are again the same, because no packet with deadline at most γ is admitted, and because for $\tau > \gamma$ and any packet g

lighter than f we have $\mathsf{pslack}(P_{\rhd g}, \tau) = \mathsf{pslack}(Q_{\rhd g}, \tau)$, so g gets admitted to Q if and only if g is admitted to P.

Next, consider the case when $d_s \leq d_f$. In this case, since also $d_f \leq \gamma$, we have $\mathsf{nexttts}(P, d_f) = \gamma$, so the interval $(\beta, \gamma]$ is in fact a single segment of P. (This also proves the second claim in Case (b2), namely that $(\beta, \gamma]$ is a segment of P.) This gives us that $\mathsf{pslack}(P_{\rhd f}, \tau) \geq \mathsf{pslack}(P, \tau) \geq 1$ for all $\tau \in (\beta, \gamma)$. Since P admits f, we also have $\mathsf{pslack}(P_{\rhd f}, \tau) \geq 1$ for all $\tau \geq d_f$. Therefore, $\mathsf{pslack}(Q_{\rhd s}, \tau) = \mathsf{pslack}(P_{\rhd f}, \tau) \geq 1$ for all $\tau \geq d_s$, which implies that Q will admit s. As in the previous case, we will then have $\mathsf{pslack}(Q_{\unrhd s}, \gamma) = \mathsf{pslack}(P_{\unrhd f}, \gamma) = 0$, so the run for Q will not admit f (which is next in the ordering), and the remainders of both runs will be identical.

Next, to prove parts (a)–(c) in the case $w_s > w_f$, we analyze the changes of $\mathsf{pslack}()$ and $\mathsf{minwt}()$ caused by the arrival of s.

(a) This part is straightforward, as replacing f by s in the plan does not change any value of $|P[t, \tau]|$ for slots $\tau \in [t, \min(d_f, d_s)) \cup [\max(d_f, d_s), T]$. In particular, β and γ remain tight in Q, and slots in $(\beta, \min(d_f, d_s)) \cup [\max(d_f, d_s), \gamma)$ are not tight in Q.

(b) Recall that $Q = P \cup \{s\} \setminus \{f\}$. Suppose that $d_s > d_f$. Then $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) + 1$ for $\tau \in [d_f, d_s)$ as $|Q[t, \tau]| = |P[t, \tau]| - 1$ for such τ. So there are no tight slots in the interval $[d_f, d_s)$ in Q. This shows (b1).

The argument for (b2) is similar. Suppose that $d_s \leq d_f$. As already justified earlier, $(\beta, \gamma]$ is a segment of P. Hence, as $\beta < d_s \leq d_f \leq \gamma$, the slack values for $\tau \in [d_s, d_f)$ are strictly positive in P. Further, $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) - 1 \geq 0$ for $\tau \in [d_s, d_f)$. This means that in Q there may appear additional tight slots in $[d_s, d_f)$.

(c) For $\tau \in [t, \beta]$, the set of packets with deadline at most τ does not change and tight slots up to β remain the same. Hence, $\mathsf{minwt}(Q, \tau) = \mathsf{minwt}(P, \tau)$ for such τ. For $\tau \in (\beta, \gamma]$, it holds that $\mathsf{minwt}(P, \tau) = w_f$ and, as s replaces f and γ remains a tight slot, all packets in Q with deadlines at most $\mathsf{nexttts}(Q, \tau) \leq \gamma$ are heavier than f, so we actually have $\mathsf{minwt}(Q, \tau) > \mathsf{minwt}(P, \tau)$. Finally, consider $\tau > \gamma$. Since tight slots after γ are unchanged, the set of packets considered in the definition of $\mathsf{minwt}(\tau)$ changes only by replacing f by s with $w_s > w_f$, which implies that $\mathsf{minwt}(Q, \tau) \geq \mathsf{minwt}(P, \tau)$. $\qquad\square$

The observation below provides a characterization of optimal plans in terms of the $\mathsf{minwt}()$ function. Its ($\Rightarrow$) implication follows from the first claim of Lemma A.2. The ($\Leftarrow$) implication follows by noting that if a plan P satisfies the condition in the observation, then running Algorithm $\mathsf{ComputePlan}$ on $\mathcal{U}$ will produce exactly the set of packets in P. (The formal proof proceeds by induction, showing that packets from P will be admitted to X because P is feasible, but packets from $\mathcal{U} \setminus P$ will not, as the condition in line (4) will be violated at the time they are considered.)

Observation A.3. A plan P is optimal for a set $\mathcal{U}$ of pending packets if and only if $w_j < \mathsf{minwt}(P, d_j)$ for all packets $j \in \mathcal{U} \setminus P$.

A.2. Updating the optimal plan after transmitting a packet. Next, we analyze how the optimal plan evolves after an algorithm transmits a packet from the current optimal plan. (Only such transmissions are relevant to our analysis as in our algorithm PlanM packets from outside the optimal plan are not considered for transmission.) We stress that in this section we only analyze the effects of the event of a packet being transmitted, and the properties we establish are independent of the algorithm. In Appendix B we will deal with the changes in the plan specific to PlanM, namely with packet transmissions that also involve adjustments of the instance.
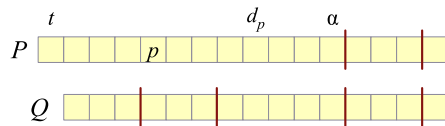
FIG. A.2. *An illustration of changes of tight slots and segments in Lemma* A.4.

So let $\mathcal{U}$ be the set of packets pending at a time t and denote by $P = P^t$ the optimal plan for $\mathcal{U}$. Suppose that we choose some packet $p \in P$, transmit it at time t, and advance the time to $t + 1$. The set $\mathcal{U}'$ of packets pending at time $t + 1$ is obtained from $\mathcal{U}$ by removing p and all packets that expire at time t. In this section, Q will denote the new optimal plan for $\mathcal{U}'$ and time step $t + 1$; that is, $Q = \widetilde{P}^{t+1}$ in the notation from section 5.5.

We now explain how to obtain Q from P. There are two cases, depending on whether or not p is in the initial segment $[t, \alpha]$ of P, where $\alpha = \mathsf{nextts}(P, t)$. We start with the case when p is in the initial segment, which is quite straightforward. See Figure A.2 for an illustration.

LEMMA A.4 (The Plan-Update Lemma for Transmitting $p \in P[t, \alpha]$). *Suppose that at time t we transmit a packet $p \in P[t, \alpha]$, where $\alpha = \mathsf{nextts}(P, t)$. Then:*

(a) $Q = P \setminus \{p\}$. *Thus the new optimal plan Q is obtained from P simply by removing p.*

(b) $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) - 1$ *for* $\tau \in [t + 1, d_p)$ *and* $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau)$ *for* $\tau \geq d_p$. *Thus new tight slots may appear in* $[t + 1, d_p)$.

(c) *For any slot $\tau \geq t + 1$, $\mathsf{minwt}(Q, \tau) \geq \mathsf{minwt}(P, \tau)$.*

Note that part (b) is meaningful in the special case when $d_p = t$, as according to our definition of $\mathsf{pslack}()$, $\mathsf{pslack}(Q, t)$ is defined and equal to 0, that is slot t is considered tight at time $t + 1$.

Proof. Observe first that $P \setminus \{p\} \subseteq \mathcal{U}'$. Indeed, if some $q \in P \setminus \{p\}$ had $d_q = t$, it would mean that the first tight slot α is t, so q would have to be equal to p.

(a) According to Observation A.1(c), P has a schedule where p is scheduled in slot t. This implies directly that $P \setminus \{p\}$ is a feasible set of packets at time $t + 1$. If $Q' \neq P \setminus \{p\}$ is any other feasible subset of $\mathcal{U}'$, then Q' would have a schedule starting at time $t + 1$, so $P' = Q' \cup \{p\}$ would have a schedule starting at time t and thus would be feasible at time t. Since P is heavier than P', $P \setminus \{p\}$ is heavier than Q', thus proving that the optimal plan at time $t + 1$ is indeed $Q = P \setminus \{p\}$.

(b) For $\tau \in [t + 1, d_p)$, $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) - 1$ as in Q the time is incremented and $|Q[t + 1, \tau]| = |P[t + 1, \tau]|$. For $\tau \geq d_p$, incrementing the time for Q is compensated by p not contributing to $Q[t+1, \tau]$ (that is, $|Q[t+1, \tau]| = |P[t+1, \tau]| - 1$), and thus $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau)$.

(c) By (b), the value of $\mathsf{nextts}(P, \tau)$ may only decrease. Since also no packet is added to the plan, in the definition of $\mathsf{minwt}(Q, \tau)$ we consider a subset of packets used to define $\mathsf{minwt}(P, \tau)$, which shows (c). $\quad\square$

For the case $p \in P(\alpha, T]$, recall that the substitute packet $\mathsf{sub}(P, p)$ for p is the heaviest pending packet $\varrho \in \mathcal{U} \setminus P$ satisfying $d_\varrho > \mathsf{prevts}(P, d_p)$. We prove that $\mathsf{sub}(P, p)$ really appears in the plan when p is transmitted. See Figure A.3 for an illustration.
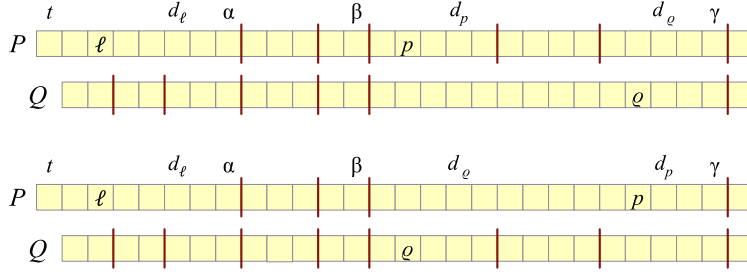
FIG. A.3. *An illustration of changes of tight slots and segments, in Case* (d1) *of Lemma* A.5 *on the top and in Case* (d2) *on the bottom.*

LEMMA A.5 (The Plan-Update Lemma for Transmitting $p \in P(\alpha, T]$). *Suppose that at time t we transmit a packet $p \in P(\alpha, T]$, where $\alpha = \mathsf{nextts}(P, t)$. Let $\varrho = \mathsf{sub}(P, p)$, $\beta = \mathsf{prevts}(P, d_p)$, and $\gamma = \mathsf{nextts}(P, d_\varrho)$, and denote by ℓ the lightest packet in $P[t, \alpha]$, i.e., in the initial segment of P. Then the following hold:*

(a) $Q = P \setminus \{p, \ell\} \cup \{\varrho\}$. *Thus the new optimal plan Q is obtained from P by removing ℓ and replacing p by ϱ.*

(b) $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) - 1$ *for $\tau \in [t+1, d_\ell)$.*

(c) $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau)$ *for $\tau \in [d_\ell, \min(d_p, d_\varrho)) \cup [\max(d_p, d_\varrho), T]$.*

(d) *For $\tau \in [\min(d_p, d_\varrho), \max(d_p, d_\varrho))$ there are two cases:*

    (d1) *If $d_\varrho > d_p$, then $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) + 1$ for $\tau \in [d_p, d_\varrho)$. This means that all segments of P in $(\beta, \gamma]$ get merged into one segment $(\beta, \gamma]$ of Q.*

    (d2) *If $d_\varrho < d_p$, then $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) - 1$ for $\tau \in [d_\varrho, d_p)$. In this case $(\beta, \gamma]$ is itself a segment of P, and in Q there might be new tight slots in $[d_\varrho, d_p)$ that partition $(\beta, \gamma]$ into smaller segments.*

Proof. (a) Let $\mathcal{U}$ be the set of packets pending at time t before transmitting p and $\mathcal{U}'$ be the packets that remain pending at time $t+1$ after p is transmitted; that is, $\mathcal{U}'$ is obtained from $\mathcal{U}$ by removing p and all packets that expire at time t. We observe first that $P \setminus \{p, \ell\} \subseteq \mathcal{U}'$. Indeed, if there was some $q \in P \setminus \{p, \ell\}$ with $d_q = t$, this would imply that $\alpha = t$, so $d_\ell = t$ and thus $\ell = q$, contradicting the choice of q. Obviously, $p \notin \mathcal{U}'$. As for ℓ, it may or may not be in $\mathcal{U}'$, depending on whether or not $d_\ell > t$. We also have $\varrho \in \mathcal{U}'$, because $d_\varrho > \beta \geq t$.

We prove that $Q = P \setminus \{p, \ell\} \cup \{\varrho\}$ in two substeps:

- First, we increment the time to $t+1$ but without transmitting p. After this substep the set of pending packets is $\overline{\mathcal{U}} = \{a \in \mathcal{U} : d_a \geq t+1\}$, i.e., we exclude packets that expire at time t. We show that the optimal plan for $\overline{\mathcal{U}}$ is $\overline{Q} = P \setminus \{\ell\}$.

- In the second substep we remove p from $\overline{\mathcal{U}}$ (and transmit it in time slot t), so the new set of pending packets will be $\mathcal{U}' = \overline{\mathcal{U}} \setminus \{p\}$, and we prove that the new optimal plan will be $Q = \overline{Q} \setminus \{p\} \cup \{\varrho\}$. (Recall that we have $p \neq \ell$ as $p \notin P[t, \alpha]$ and $\ell \in P[t, \alpha]$.)

The intuition is that these two substeps are largely independent: incrementing t to $t+1$ will squeeze out ℓ from P, no matter what packets with deadlines after α are removed from $\mathcal{U}$.

First substep. For the first substep, we use an argument similar to the one in Lemma A.4. Observe that $P \setminus \{\ell\}$ is a feasible set of packets at time $t+1$, since P

has a schedule where ℓ is in slot t, by Observation A.1(c). Let $Q' \neq P \setminus \{\ell\}$ be any other subset of $\overline{\mathcal{U}}$ that is feasible at time $t + 1$; we show that $w(Q') < w(P \setminus \{\ell\})$. Since P contains exactly $\alpha - t + 1$ packets with deadline at most α, there is a packet $q' \in P[t, \alpha]$ such that $q' \notin Q'$ (possibly $q' = \ell$). As Q' has a schedule starting at time $t + 1$, $P' = Q' \cup \{q'\}$ has a schedule starting at time t and thus is feasible at time t. Since P is at least as heavy as P' and ℓ is the lightest packet in $P[t, \alpha]$, implying $w_\ell \leq w_{q'}$, we conclude that indeed $w(P \setminus \{\ell\}) \geq w(Q')$. Hence, the plan after incrementing the time is indeed $\overline{Q} = P \setminus \{\ell\}$.

Thus we can think of $\overline{Q}$ as being obtained from P by advancing the time to $t+1$ and removing ℓ. It follows that $\mathsf{pslack}(\overline{Q}, \tau) = \mathsf{pslack}(P, \tau)$ for $\tau \geq d_\ell$. In particular, β is a tight slot of $\overline{Q}$ and $\beta = \mathsf{prevts}(\overline{Q}, d_p)$. Further, as $\mathcal{U}$ and $\overline{\mathcal{U}}$ contain the same packets with deadline after β and as $\beta \geq d_\ell$, we get that ϱ is the heaviest packet in $\overline{\mathcal{U}} \setminus \overline{Q}$ with deadline greater than β. Summarizing, we have

$$(A.1) \qquad \beta = \mathsf{prevts}(\overline{Q}, d_p),$$

$$(A.2) \qquad \varrho = \mathrm{argmax}_j \left\{ w_j : j \in \overline{\mathcal{U}} \setminus \overline{Q} \ \& \ d_j > \beta \right\},$$

$$(A.3) \qquad \gamma = \mathsf{nextts}(\overline{Q}, d_\varrho).$$

Second substep. In the second substep we establish a relationship between $\overline{Q}$ and Q. To this end, we consider two "parallel" runs of the greedy algorithm to compute plans: one run of $\mathsf{ComputePlan}(\overline{\mathcal{U}}, t + 1)$ which yields $\overline{Q}$ and another run of $\mathsf{ComputePlan}(\mathcal{U}', t+1)$ which yields Q. (Recall that $\mathcal{U}' = \overline{\mathcal{U}} \setminus \{p\}$.) The contents of the set X in these two runs after processing packets heavier than some packet j is equal to $\overline{Q}_{\triangleright j}$ and $Q_{\triangleright j}$, respectively. To avoid ambiguity, we will be using these notations when referring to the contents of X during these runs.

We divide these two runs into three phases. In the first phase we consider the two runs for packets j with $w_j > w_p$ (that is, before p gets admitted to $\overline{Q}$). Since $\overline{\mathcal{U}}_{\triangleright p} = \mathcal{U}'_{\triangleright p}$, in this phase the two runs are exactly the same. This gives us that

$$\overline{Q}_{\triangleright p} = Q_{\triangleright p}.$$

The second phase involves packets j with $w_p \geq w_j > w_\varrho$. We claim that during this phase the following invariants will be maintained:

$$(A.4) \qquad Q_{\trianglerighteq j} = \overline{Q}_{\trianglerighteq j} \setminus \{p\},$$

$$(A.5) \qquad \mathsf{pslack}(Q_{\trianglerighteq j}, \tau) = \begin{cases} \mathsf{pslack}(\overline{Q}_{\trianglerighteq j}, \tau) & \text{for } \tau \in [t+1, \beta], \\ \mathsf{pslack}(\overline{Q}_{\trianglerighteq j}, \tau) \geq 1 & \text{for } \tau \in (\beta, d_p), \\ \mathsf{pslack}(\overline{Q}_{\trianglerighteq j}, \tau) + 1 & \text{for } \tau \geq d_p. \end{cases}$$

We first show that these properties hold for $j = p$. We have $p \in \overline{Q}$ and $\mathcal{U}' = \overline{\mathcal{U}} \setminus \{p\}$; this means that p will be admitted to $\overline{Q}_{\trianglerighteq p}$ in the run for $\overline{\mathcal{U}}$ but it will not be considered in the run for $\mathcal{U}'$. Thus property (A.4) follows. Invariant (A.4) and the definition of $\mathsf{pslack}()$ imply all relations between $\mathsf{pslack}(Q_{\trianglerighteq p}, \tau)$ and $\mathsf{pslack}(\overline{Q}_{\trianglerighteq p}, \tau)$ in invariant (A.5), and the inequality $\mathsf{pslack}(\overline{Q}_{\trianglerighteq p}, \tau) \geq 1$ for $\tau \in (\beta, d_p)$ follows immediately from (A.1). (Note that $\mathsf{pslack}(\overline{Q}_{\trianglerighteq a}, \tau) \geq \mathsf{pslack}(\overline{Q}, \tau)$ holds for any packet a and slot τ.)

Next, consider packets j with $w_p > w_j > w_\varrho$. If j gets admitted to $\overline{Q}_{\triangleright j}$ then invariant (A.5) implies that it also gets admitted to $Q_{\triangleright j}$, preserving (A.4). That (A.5) is preserved follows from the same argument as for $j = p$ above.

So consider the second case, when j does not get admitted to $\overline{Q}_{\unrhd j}$. We claim that j will also not be admitted to $Q_{\unrhd j}$. To justify this, we argue as follows. By (A.2), ϱ is the heaviest packet in $\overline{\mathcal{U}}$ with deadline after β that will not be admitted to $\overline{Q}$. Therefore, since $w_j > w_\varrho$, we have $d_j \leq \beta$. Suppose for a contradiction that j is admitted to $Q_{\unrhd j}$. As j is not admitted to $\overline{Q}_{\unrhd j}$, there must be $\tau \geq d_j$ such that $\mathsf{pslack}(\overline{Q}_{\rhd j}, \tau) = 0$, although $\mathsf{pslack}(Q_{\rhd j}, \tau) > 0$ as j is admitted to $Q_{\unrhd j}$. Due to invariant (A.5) for $\tau \in [t+1, \beta]$, we must have $\tau > \beta$. As τ is already tight, by Observation A.1(d) we get that $\mathsf{pslack}(\overline{Q}_{\rhd j}, \beta) = \mathsf{pslack}(\overline{Q}, \beta) = 0$, where the second step holds by (A.1). Using invariant (A.5) for $\tau \in [t+1, \beta]$, we obtain $\mathsf{pslack}(Q_{\rhd j}, \beta) = \mathsf{pslack}(\overline{Q}_{\rhd j}, \beta) = 0$, contradicting that we must have $\mathsf{pslack}(Q_{\rhd j}, \beta) > 0$ as j is admitted to $Q_{\unrhd j}$ and $d_j \leq \beta$. Thus we indeed have that j will not be admitted to $Q_{\unrhd j}$. As in this case j is not admitted to both $\overline{Q}_{\unrhd j}$ and $Q_{\unrhd j}$, both invariants (A.4) and (A.5) are obviously preserved.

In the third phase we analyze both runs for packets j with $w_j \leq w_\varrho$. We claim that for these packets the following invariants hold:

$$(A.6) \qquad\qquad Q_{\unrhd j} = \overline{Q}_{\unrhd j} \setminus \{p\} \cup \{\varrho\},$$

$$(A.7) \qquad \mathsf{pslack}(Q_{\unrhd j}, \tau) = \mathsf{pslack}(\overline{Q}_{\unrhd j}, \tau) \quad \text{for } \tau \geq \gamma,$$

$$(A.8) \qquad \mathsf{pslack}(Q_{\unrhd j}, \gamma) = \mathsf{pslack}(\overline{Q}_{\unrhd j}, \gamma) = 0.$$

Note that property (A.7) follows directly from (A.6), because $d_p, d_\varrho \leq \gamma$. Thus we only need to show that (A.6) and (A.8) hold for all j in this phase.

We first consider $j = \varrho$. As $\varrho \notin \overline{Q}$, ϱ will not be admitted to $\overline{Q}_{\unrhd \varrho}$. From (A.5) and (A.2) we have that $\mathsf{pslack}(Q_{\rhd \varrho}, \tau) \geq 1$ for $\tau \geq d_\varrho$, and thus ϱ will be admitted to $Q_{\unrhd \varrho}$. This gives us (A.6) and (A.7). Since $\varrho \notin \overline{Q}$, all packets in $\overline{Q}[t+1, \gamma]$ are heavier than ϱ; in other words, we have $\overline{Q}[t+1, \gamma] = Q_{\unrhd \varrho}[t+1, \gamma]$, which together with (A.3) implies that $\gamma = \mathsf{nextts}(\overline{Q}_{\unrhd \varrho}, d_\varrho)$. Then, from (A.7) (which we already established for ϱ), we obtain that $\mathsf{pslack}(Q_{\unrhd \varrho}, \gamma) = \mathsf{pslack}(\overline{Q}_{\unrhd \varrho}, \gamma) = 0$.

In the rest of the third phase, for each packet j with $w_j < w_\varrho$, condition (A.8) implies that if $d_j \leq \gamma$, then j will not be admitted to $\overline{Q}_{\unrhd j}$ or to $Q_{\unrhd j}$. On the other hand, if $d_j > \gamma$, then condition (A.7) implies that j will be admitted either to both $\overline{Q}_{\unrhd j}$ and $Q_{\unrhd j}$ or to none. This completes the proof that the above invariants are preserved, and the proof of (a) as well.

(b) We now analyze the changes in the values of $\mathsf{pslack}()$. If $d_\ell > t$, it holds that $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) - 1$ for $\tau \in [t+1, d_\ell]$, as in the transmission step the time was incremented and there is no change in the set of packets taken into account.

(c) For $\tau \in [d_\ell, \min(d_\varrho, d_p))$ we have $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau)$, as in the transmission step the time was incremented, but ℓ was forced out. (In particular, since $d_\ell \leq \alpha < \min(d_\varrho, d_p)$, α is a tight slot in Q, including the special case when $\alpha = t$.) Similarly, for $\tau \geq \max(d_\varrho, d_p)$, it holds that $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau)$, since in the transmission step the time was incremented, ℓ was forced out, p was transmitted, and ϱ appeared in Q.

(d) In Case (d1), for $\tau \in [d_p, d_\varrho)$, we have $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) + 1$, as in the transmission step the time was increased, but ℓ was forced out and p was transmitted. It follows that Q has no tight slots in $[d_p, d_\varrho)$, so $\mathsf{prevts}(Q, d_\varrho) = \beta$ and $\mathsf{nextts}(Q, d_p) = \gamma$, which means that in Q the interval $(\beta, \gamma]$ forms one segment.

In Case (d2), the definitions of $\beta = \mathsf{prevts}(P, d_p)$ and $\gamma = \mathsf{nextts}(P, d_\varrho)$, together with $d_\varrho < d_p$, imply that $(\beta, \gamma]$ is indeed a segment of P. In particular, this means that $\mathsf{pslack}(P, \tau) \geq 1$ for $\tau \in [d_\varrho, d_p)$. For such slots τ we thus obtain that $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) - 1$ because in the transmission step the time was increased, ℓ was forced out, and ϱ was added to Q. $\qquad \square$

Note that after transmitting $p \in P(\alpha, T]$ (without adjusting packet weights or deadlines appropriately), the slot monotonicity property does not hold. This is because $\mathsf{minwt}(d_\varrho)$ decreases, as $\mathsf{minwt}(P, d_\varrho) > w_\varrho$ but $\mathsf{minwt}(Q, d_\varrho) = w_\varrho$.

Appendix B. Leap step of Algorithm PlanM. Algorithm PlanM artificially adjusts weights and deadlines of some pending packets in leap steps, and therefore the results from the previous section are not sufficient to fully characterize how the optimal plan evolves during its computation. Lemmas A.2 and A.4 remain valid for Algorithm PlanM, but we still need a variant of Lemma A.5 that will characterize the changes in the optimal plan after a leap step. We provide such a characterization below; see Lemma B.1. Then we use this lemma to prove Lemma B.2, the slot-monotonicity property for leap steps.

In the following, similar to the previous section, we use P for P^t (the optimal plan after all packets at time t have been released) and by $Q = \widetilde{P}^{t+1}$ we denote the optimal plan just after the event of transmitting p, incrementing the time, and changing weights and deadlines (but before new packets at time $t+1$ are released).

LEMMA B.1. *Suppose that t is a leap step of Algorithm PlanM in which p was transmitted, and let $\varrho = \mathsf{sub}(P, p)$ be the substitute packet of p. Let packets $h_0 = p, h_1, \ldots, h_k$ and tight slots $\tau_0, \ldots, \tau_k = \gamma = \mathsf{nextts}(P, d_\varrho^t)$ be as defined in the algorithm. Furthermore, let $\beta = \tau_{-1} = \mathsf{prevts}(P, d_p^t)$. Then the following hold:*

(a) $Q = P \setminus \{p, \ell\} \cup \{\varrho\}$; *in particular, if $k \geq 1$, all packets $h_1, h_2, \ldots, h_k$ are in Q.*

(b) $w_p^t = w_{h_0}^t > w_{h_1}^t > w_{h_2}^t > \cdots > w_{h_k}^t > w_\varrho^t$.

(c) *h_k's deadline is in the segment of P ending at γ, that is, $\mathsf{prevts}(P, d_\varrho^t) < d_{h_k}^t \leq \gamma$.*

(d) *The $\mathsf{pslack}()$ values change as follows (see Figure B.1):*

　(d.i) $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) - 1$ *for $\tau \in [t+1, d_\ell^t)$.*

　(d.ii) $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau)$ *for $\tau \in [d_\ell^t, \beta]$.*

　(d.iii) *If $k \geq 1$, then for $i = 0, \ldots, k-1$ we have the following changes in $(\tau_{i-1}, \tau_i]$:*

　　(d.iii.1) $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau)$ *for $\tau \in (\tau_{i-1}, d_{h_i}^t) \cup \{\tau_i\}$.*

　　(d.iii.2) $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) + 1$ *for $\tau \in [d_{h_i}^t, \tau_i)$.*

　(d.iv) $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau)$ *for $\tau \in (\tau_{k-1}, \min(d_{h_k}^t, d_\varrho^t))$.*

　(d.v) *For $\tau \in [\min(d_{h_k}^t, d_\varrho^t), \max(d_{h_k}^t, d_\varrho^t))$, there are two cases:*

　　(d.v.1) *If $d_{h_k}^t < d_\varrho^t$, then $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) + 1$ for $\tau \in [d_{h_k}^t, d_\varrho^t)$.*

　　(d.v.2) *If $d_\varrho^t < d_{h_k}^t$, then $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau) - 1$ for $\tau \in [d_\varrho^t, d_{h_k}^t)$.*

　(d.vi) $\mathsf{pslack}(Q, \tau) = \mathsf{pslack}(P, \tau)$ *for $\tau \geq \max(d_{h_k}^t, d_\varrho^t)$.*

(e) *Any tight slot of P is a tight slot of Q, but there might be new tight slots of Q in $[t+1, d_\ell^t)$ and, in Case (d.v.2), also in $[d_\varrho^t, d_{h_k}^t)$. Thus, in general, we have $\mathsf{nextts}(Q, \tau) \leq \mathsf{nextts}(P, \tau)$ for all $\tau \geq t+1$.*

(f) $w_{h_i}^{t+1} \geq \mathsf{minwt}(P, d_{h_i}^{t+1})$ *for any $i = 1, \ldots, k$, and $w_\varrho^{t+1} \geq \mathsf{minwt}(P, d_\varrho^{t+1})$.*

Proof. (a) The claim clearly holds for a simple leap step by Lemma A.5, i.e., when $k = 0$, because in this case the algorithm does not decrease deadlines, and increasing the weight of ϱ does not change the new optimal plan Q, by Observation A.1(b).

Thus consider an iterated leap step. Let $\overline{Q}$ be the optimal plan after p is transmitted and the time is increased, but before the adjustment of weights and dead-

lines in line 11 of the algorithm are taken into account. From Lemma A.5 we have $\overline{Q} = P \setminus \{p, \ell\} \cup \{\varrho\}$; in particular, packets $h_1, h_2, \ldots, h_k$ are in $\overline{Q}$. Our goal is to prove that $\overline{Q} = Q$.

By Observation A.1(b), increasing the weights of packets in the optimal plan cannot change the plan, so it is sufficient to show that the feasibility and optimality of the plan is not affected by the decrease of the deadlines of packets $h_1, h_2, \ldots, h_k$.

By Lemma A.5, all segments of $\overline{Q}$ between $\beta = \mathsf{prevts}(P, d_p^t)$ and $\gamma = \mathsf{nextts}(P, d_\varrho^t)$ get merged into one, which means that $\mathsf{pslack}(\overline{Q}, \tau) \geq 1$ for any $\tau \in (\beta, \gamma)$. For $i = 1, \ldots, k$, decreasing the deadline of h_i from $d_{h_i}^t$ to $d_{h_i}^{t+1} = \tau_{i-1}$ decreases $\mathsf{pslack}(\overline{Q}, \tau)$ by 1 for $\tau \in [\tau_{i-1}, d_{h_i}^t)$. All k intervals where these decreases occur are contained in (β, γ) and, since $d_{h_i}^t \leq \tau_i$ for all i, these intervals do not overlap. Thus, after decreasing the deadlines of the h_i's, all values of $\mathsf{pslack}()$ will remain nonnegative for $\overline{Q}$, so $\overline{Q}$ remains a feasible set of packets. Decreasing these deadlines cannot make previously unfeasible sets of packets feasible, implying that Q remains optimal. Thus $\overline{Q} = Q$, as claimed. In particular, packets $h_1, h_2, \ldots, h_k$ remain in Q.

(b) By the choice of p in the algorithm, p is the heaviest packet in the segments of P in $(\beta, \gamma]$, because ϱ is the substitute packet for any packet in P with deadline in $(\beta, \gamma]$. Thus for each $i = 1, \ldots, k$, since $d_{h_i}^t \in (\beta, \gamma]$, we get $w_p^t > w_{h_i}^t$. The ordering of weights of h_i's follows from the definition of h_i's in line 9 of the algorithm's description. Finally, for any $i = 0, \ldots, k$ inequality $w_{h_i}^t > w_\varrho^t$ holds, because $\varrho \notin P$ and $d_{h_i}^t < \gamma = \mathsf{nextts}(P, d_\varrho^t)$.

(c) This holds by the definition of h_i's in line 9 and by the condition of the while loop in line 7, which stops the loop when $\tau_i = \mathsf{nextts}(P, d_{h_i}^t) = \gamma$.

(d) For any slot $\tau \in [t+1, \beta] \cup (\gamma, T]$ the value of $\mathsf{pslack}(\tau)$ is not affected by the decrease of the deadlines of h_i's, since h_i's are both in P and in Q and since their old and new deadlines are in (β, γ). We thus get exactly the same changes of the $\mathsf{pslack}()$ values outside (β, γ) as in Lemma A.5. Regarding slots in (β, γ), Lemma A.5(d) shows that $\mathsf{pslack}(\tau)$ decreases by 1 for $\tau \in [d_\varrho^t, d_p^t)$ if $d_\varrho^t < d_p^t$, and increases by 1 for $\tau \in [d_p^t, d_\varrho^t)$ if $d_p^t < d_\varrho^t$. In the former case, we have $\tau_0 = \gamma$ as d_ϱ^t and d_p^t are in the same segment, that is $k = 0$. In the latter case, we sum the increase of values $\mathsf{pslack}(\tau)$ for $\tau \in [d_p^t, d_\varrho^t)$ with the changes of the $\mathsf{pslack}()$ values due to decreasing the deadlines of h_i's, analyzed in (a), and we get the changes summarized in (d); see Figure B.1.

Part (e) follows from (c) and (d), which imply that the value of $\mathsf{pslack}(P, \tau)$ does not increase for any tight slot τ in P.

Part (f) follows from the way the weights are changed in lines 4 and 11 of the algorithm. □

Next, we show that, thanks to the deadline adjustments in Algorithm PlanM, the slot monotonicity property for optimal plans is preserved in leap steps.

LEMMA B.2. *If step t is a leap step of Algorithm PlanM, then $\mathsf{minwt}(Q, \tau) \geq \mathsf{minwt}(P, \tau)$ for any $\tau \geq t + 1$.*

Proof. We use notation from Lemma B.1. By Lemma B.1(e) all tight slots of P are tight slots of Q (in particular, β and γ remain tight slots); thus $\mathsf{nextts}(Q, \tau) \leq \mathsf{nextts}(P, \tau)$ for all $\tau \geq t + 1$. Fix some $\tau \geq t + 1$, and let a be the packet that realizes $\mathsf{minwt}(Q, \tau)$, that is the minimum-weight packet in Q with $d_a^{t+1} \leq \mathsf{nextts}(Q, \tau)$. We need to show that $w_a^{t+1} \geq \mathsf{minwt}(P, \tau)$. We have three cases.

Case 1. $\tau \leq \beta$. Lemma B.1(a) shows that ℓ is forced out of the optimal plan, thus not in Q, and otherwise the set of packets in the optimal plan with deadline at
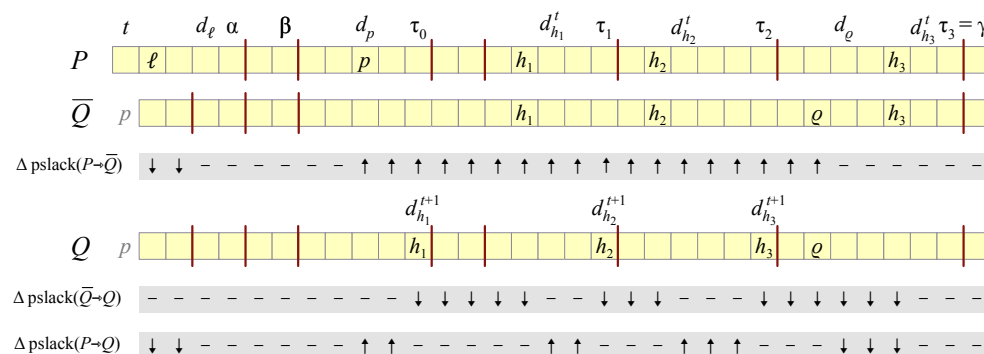
FIG. B.1. *An example with $k = 3$ illustrating the proof of Lemma B.1(d), by showing changes of the pslack() values in an iterated leap step. Here, notations $\Delta\mathsf{pslack}(P \to \overline{Q})$, $\Delta\mathsf{pslack}(\overline{Q} \to Q)$, and $\Delta\mathsf{pslack}(P \to Q)$ represent the changes of the pslack() in the two substeps (from P to $\overline{Q}$ and from $\overline{Q}$ to Q), and the overall change from P to Q, respectively. Up and down arrows represent increases and decreases of the values of pslack() in individual slots.*

most β does not change, that is, $Q[t+1, \beta] = P[t, \beta] \setminus \{\ell\}$. It follows that $a \in P$ and its weight and deadline remain the same. We thus have $d_a^t = d_a^{t+1} \leq \mathsf{nextts}(Q, \tau) \leq \mathsf{nextts}(P, \tau)$, so a is considered in the definition of $\mathsf{minwt}(P, \tau)$, which implies $w_a^{t+1} = w_a^t \geq \mathsf{minwt}(P, \tau)$.

Case 2. $\tau \in (\beta, \gamma]$. If $a \notin \{h_1, \ldots, h_{k+1}\}$ (in particular, this means that $a \neq \varrho = h_{k+1}$), then a is also in P with the same deadline and weight. As $\mathsf{nextts}(Q, \tau) \leq \mathsf{nextts}(P, \tau)$, we get $d_a^t \leq \mathsf{nextts}(P, \tau)$; hence $w_a^{t+1} \geq \mathsf{minwt}(P, \tau)$ follows as in Case 1.

Next, suppose that $a = h_i$ for some $i \in \{1, \ldots, k\}$ (excluding the case $a = h_{k+1} = \varrho$). Recall that, as a result of changes in line 11 in the algorithm, we have $w_a^{t+1} \geq \mathsf{minwt}(P, \tau_{i-1})$ and $d_a^{t+1} = \tau_{i-1}$. The definition of $\mathsf{minwt}(Q, \tau)$ gives us that $\tau_{i-1} = d_a^{t+1} = \mathsf{nextts}(Q, \tau)$, so $\tau > \mathsf{prevts}(Q, \tau_{i-1})$. Thus, since tight slots of P are tight also in Q, implying $\mathsf{prevts}(Q, \tau_{i-1}) \geq \mathsf{prevts}(P, \tau_{i-1})$, we get that $\tau > \mathsf{prevts}(P, \tau_{i-1})$. So τ is in the segment $(\mathsf{prevts}(P, \tau_{i-1}), \tau_{i-1}]$ of P. By definition, for each segment of P the value of $\mathsf{minwt}(P, \tau')$ is constant for all τ' in this segment. Therefore, $\mathsf{minwt}(P, \tau_{i-1}) = \mathsf{minwt}(P, \tau)$, and we conclude that $w_a^{t+1} \geq \mathsf{minwt}(P, \tau_{i-1}) = \mathsf{minwt}(P, \tau)$.

Finally, consider the case when $a = \varrho$. Recall that $w_\varrho^{t+1} = \mathsf{minwt}(P, d_\varrho^t)$ and $d_\varrho^t = d_\varrho^{t+1}$. We have $\tau > \mathsf{prevts}(Q, d_\varrho^t)$, which implies that $\tau > \mathsf{prevts}(P, d_\varrho^t) = \mathsf{prevts}(P, \gamma)$. Thus τ is in the segment $(\mathsf{prevts}(P, \gamma), \gamma]$ of P, just like ϱ. It follows that $w_\varrho^{t+1} = \mathsf{minwt}(P, d_\varrho^t) = \mathsf{minwt}(P, \tau)$.

Case 3. $\tau > \gamma$. The set of packets in the plan with deadline strictly after γ does not change, and also their weights and deadlines remain the same. Thus if $d_a^t > \gamma$, then using $\mathsf{nextts}(Q, \tau) \leq \mathsf{nextts}(P, \tau)$ again, we obtain that $w_a^{t+1} \geq \mathsf{minwt}(P, \tau)$. Otherwise, $d_a^t \leq \gamma$; thus $w_a^{t+1} \geq \mathsf{minwt}(Q, \gamma) \geq \mathsf{minwt}(P, \gamma) \geq \mathsf{minwt}(P, \tau)$, where the second inequality follows from Case 2 and the third one from $\gamma < \tau$. □

Acknowledgments. This work was done in part while P. Veselý was at the University of Warwick. We are all grateful to Martin Böhm for useful discussions.

REFERENCES

[1] G. AGGARWAL, G. GOEL, C. KARANDE, AND A. MEHTA, *Online vertex-weighted bipartite matching and single-bid budgeted allocations*, in Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '11), ACM, New York, SIAM, Philadelphia, 2011, pp. 1253–1264, https://doi.org/10.1137/1.9781611973082.95.

[2] W. A. AIELLO, M. YISHAY, S. RAJAGOPOLAN, AND A. ROSÉN, *Competitive queue policies for differentiated services*, J. Algorithms, 55 (2005), pp. 113–141.

[3] N. ANDELMAN, Y. MANSOUR, AND A. ZHU, *Competitive queueing policies for QoS switches*, in Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '03), ACM, New York, SIAM, Philadelphia, 2003, pp. 761–770.

[4] Y. BARTAL, F. Y. L. CHIN, M. CHROBAK, S. P. Y. FUNG, W. JAWOR, R. LAVI, J. SGALL, AND T. TICHÝ, *Online competitive algorithms for maximizing weighted throughput of unit jobs*, in Proceedings of the 21st Symposium on Theoretical Aspects of Computer Science (STACS '04), Lecture Notes in Comput. Sci. 2996, Springer, Berlin, Heidelberg, 2004, pp. 187–198.

[5] M. BÖHM, M. CHROBAK, Ł. JEŻ, F. LI, J. SGALL, AND P. VESELÝ, *Online packet scheduling with bounded delay and lookahead*, Theoret. Comput. Sci., 776 (2019), pp. 95–113.

[6] M. BIENKOWSKI, M. CHROBAK, C. DÜRR, M. HURAND, A. JEŻ, Ł. JEŻ, AND G. STACHOWIAK, *A ϕ-competitive algorithm for collecting items with increasing weights from a dynamic queue*, Theoret. Comput. Sci., 475 (2013), pp. 92–102.

[7] M. BIENKOWSKI, M. CHROBAK, C. DÜRR, M. HURAND, A. JEŻ, Ł. JEŻ, AND G. STACHOWIAK, *Collecting weighted items from a dynamic queue*, Algorithmica, 65 (2013), pp. 60–94.

[8] M. BIENKOWSKI, M. CHROBAK, AND Ł. JEŻ, *Randomized competitive algorithms for online buffer management in the adaptive adversary model*, Theoret. Comput. Sci., 412 (2011), pp. 5121–5131.

[9] F. Y. L. CHIN, M. CHROBAK, S. P. Y. FUNG, W. JAWOR, J. SGALL, AND T. TICHÝ, *Online competitive algorithms for maximizing weighted throughput of unit jobs*, J. Discrete Algorithms, 4 (2006), pp. 255–276.

[10] F. Y. L. CHIN AND S. P. Y. FUNG, *Online scheduling with partial job values: Does timesharing or randomization help?*, Algorithmica, 37 (2003), pp. 149–164.

[11] M. CHROBAK, W. JAWOR, J. SGALL, AND T. TICHÝ, *Improved online algorithms for buffer management in QoS switches*, in Proceedings of the 12th Annual European Symposium on Algorithms (ESA '04), Lecture Notes in Comput. Sci. 3221, Springer, Berlin, 2004, pp. 204–215.

[12] M. CHROBAK, W. JAWOR, J. SGALL, AND T. TICHÝ, *Improved online algorithms for buffer management in QoS switches*, ACM Trans. Algorithms, 3 (2007), 50.

[13] N. R. DEVANUR, K. JAIN, AND R. D. KLEINBERG, *Randomized primal-dual analysis of RANK-ING for online bipartite matching*, in Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '13), ACM, New York, SIAM, Philadelphia, 2013, pp. 101–107, https://doi.org/10.1137/1.9781611973105.7.

[14] M. ENGLERT AND M. WESTERMANN, *Considering suppressed packets improves buffer management in quality of service switches*, SIAM J. Comput., 41 (2012), pp. 1166–1192, https://doi.org/10.1137/110856745.

[15] M. H. GOLDWASSER, *A survey of buffer management policies for packet switches*, SIGACT News, 41 (2010), pp. 100–128.

[16] B. HAJEK, *On the competitiveness of on-line scheduling of unit-length packets with hard deadlines in slotted time*, in Proceedings of the 35th Conference on Information Sciences and Systems, 2001, pp. 434–438.

[17] Ł. JEŻ, F. LI, J. SETHURAMAN, AND C. STEIN, *Online scheduling of packets with agreeable deadlines*, ACM Trans. Algorithms, 9 (2012), 5.

[18] A. KESSELMAN, Z. LOTKER, Y. MANSOUR, B. PATT-SHAMIR, B. SCHIEBER, AND M. SVIRIDENKO, *Buffer overflow management in QoS switches*, SIAM J. Comput., 33 (2004), pp. 563–583, https://doi.org/10.1137/S0097539701399666.

[19] F. LI, J. SETHURAMAN, AND C. STEIN, *An optimal online algorithm for packet scheduling with agreeable deadlines*, in Proceedins of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '05), ACM, New York, SIAM, Philadelphia, 2005, pp. 801–802.

[20] F. LI, J. SETHURAMAN, AND C. STEIN, *Better online buffer management*, in Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '07), ACM, New York, SIAM, Philadelphia, 2007, pp. 199–208.

[21] Ł. JEŻ, *A universal randomized packet scheduling algorithm*, Algorithmica, 67 (2013), pp. 498–515.

[22] P. VESELÝ, *Packet scheduling: Plans, monotonicity, and the golden ratio*, SIGACT News, 52 (2021), pp. 72–84.

[23] A. ZHU, *Analysis of queueing policies in QoS switches*, J. Algorithms, 53 (2004), pp. 137–168.