

Online Packet Scheduling with Bounded Delay and Lookahead*

Martin Böhm¹, Marek Chrobak², Łukasz Jeż³, Fei Li⁴, Jiří Sgall¹, and Pavel Veselý¹

¹ Computer Science Institute of Charles University, Prague, Czech Republic.

{bohm,sgall,vesely}@iuuk.mff.cuni.cz

² Department of Computer Science and Engineering, University of California, Riverside, USA. marek@cs.ucr.edu.

³ Institute of Computer Science, University of Wrocław, Poland. lje@cs.uni.wroc.pl.

⁴ Department of Computer Science, George Mason University, USA. lifei@cs.gmu.edu.

Abstract

We study the *online bounded-delay packet scheduling problem (PacketScheduling)*, where packets of unit size arrive at a router over time and need to be transmitted over a network link. Each packet has two attributes: a non-negative weight and a deadline for its transmission. The objective is to maximize the total weight of the transmitted packets. This problem has been well studied in the literature, yet its optimal competitive ratio remains unknown: the best upper bound is 1.828 [7], still quite far from the best lower bound of $\phi \approx 1.618$ [11, 2, 5].

In the variant of *PacketScheduling* with *s-bounded instances*, each packet can be scheduled in at most s consecutive slots, starting at its release time. The lower bound of ϕ applies even to the special case of 2-bounded instances, and a ϕ -competitive algorithm for 3-bounded instances was given in [4]. Improving that result, and addressing a question posed by Goldwasser [9], we present a ϕ -competitive algorithm for 4-bounded instances.

We also study a variant of *PacketScheduling* where an online algorithm has the additional power of *1-lookahead*, knowing at time t which packets will arrive at time $t + 1$. For *PacketScheduling* with 1-lookahead restricted to 2-bounded instances, we present an online algorithm with competitive ratio $\frac{1}{2}(\sqrt{13} - 1) \approx 1.303$ and we prove a nearly tight lower bound of $\frac{1}{4}(1 + \sqrt{17}) \approx 1.281$. In fact, our lower bound result is more general: using only 2-bounded instances, for any integer $\ell \geq 0$ we prove a lower bound of $\frac{1}{2(\ell+1)}(1 + \sqrt{5 + 8\ell + 4\ell^2})$ for online algorithms with ℓ -lookahead, that is algorithms that at time t can see all packets arriving by time $t + \ell$.

1 Introduction

Background. Optimizing the flow of packets across an IP network gives rise to a plethora of challenging algorithmic problems. In fact, even scheduling packet transmissions from a router across a specific network link can involve non-trivial tradeoffs. Several models for such tradeoffs have been formulated, depending on the architecture of the router, on characteristics of the packets, and on the objective function.

In the model that we study in this paper, each packet has two attributes: a non-negative weight and a deadline for its transmission. The time is assumed to be discrete (slotted), and

*M. Böhm, J. Sgall, and P. Veselý were supported by project 17-09142S of GA ČR and by the GAUK project 634217. M. Chrobak was supported by NSF grants CCF-1217314 and CCF-1536026. L. Jeż was supported by NCN grant 2016/21/D/ST6/02402. F. Li was supported by NSF grant CCF-1216993. The conference version of this paper appeared in ISAAC 2016 [3]

37 only one packet can be sent in each slot. The objective is to maximize the total weight of the
38 transmitted packets. We focus on the online setting, where at each time step the router needs to
39 choose a pending packet for transmission, without the knowledge about future packet arrivals.
40 This problem, which we call *online bounded-delay packet scheduling problem* (PacketScheduling),
41 was introduced by Kesselman *et al.* [12] as a theoretical abstraction that captures the constraints
42 and objectives of packet scheduling in networks that need to provide quality of service (QoS)
43 guarantees. The combination of deadlines and weights is used to model packet priorities.

44 In the literature, the PacketScheduling problem is sometimes referred to as *bounded-delay*
45 *buffer management in QoS switches*. It can also be formulated as the job-scheduling problem
46 $1|p_j = 1, r_j| \sum w_j U_j$, where packets are represented by unit-length jobs with deadlines, with the
47 objective to maximize the weighted throughput.

48 A router transmitting packets across a link needs to make scheduling decisions on the fly,
49 based only on the currently available information. This motivates the study of online competitive
50 algorithms for PacketScheduling. A simple online greedy algorithm that always schedules the
51 heaviest pending packet is known to be 2-competitive [11, 12]. In a sequence of papers [6, 8, 13, 7],
52 this ratio was gradually improved, and the best currently known ratio is 1.828 [7]. The best lower
53 bound, widely believed to be the optimal ratio, is $\phi = (1 + \sqrt{5})/2 \approx 1.618$ [11, 2, 5]. Closing the
54 gap between these two bounds is one of the most intriguing open problems in online scheduling.

55 ***s*-Bounded instances.** In an attempt to bridge this gap, restricted models have been studied.
56 In the *s*-bounded variant of PacketScheduling, each packet must be scheduled within k consecutive
57 slots, starting at its release time, for some $k \leq s$ possibly depending on the packet. The lower
58 bound of ϕ from [11, 2, 5] holds even in the 2-bounded case. A matching ϕ -competitive algorithm
59 was given Kesselman *et al.* [12] for 2-bounded instances and by Chin *et al.* [4] for 3-bounded
60 instances. Both results are based on the algorithm EDF_α , with $\alpha = \phi$, which always schedules
61 the earliest-deadline packet whose weight is at least the weight of the heaviest pending packet
62 divided by α (ties are broken in favor of heavier packets). EDF_ϕ is not ϕ -competitive for
63 4-bounded instances; however, a different choice of α yields a 1.732-competitive algorithm for
64 the 4-bounded case [4].

65 *Our contribution.* We present a ϕ -competitive online algorithm for PacketScheduling restricted to
66 4-bounded instances, matching the lower bound of ϕ (see Section 3). This improves the results
67 from [4] and answers the question posed by Goldwasser in his SIGACT News survey [9].

68 **Algorithms with lookahead.** In Sections 4 and 5, we investigate a variant of PacketScheduling
69 where an online algorithm is able to learn at time t which packets will arrive by time $t + 1$. This
70 property is known as *1-lookahead*. From a practical point of view, 1-lookahead corresponds to
71 the situation in which a router can see the packets that are just arriving in the buffer and that
72 will be available for transmission in the next time slot.

73 The notion of lookahead is quite natural and it has appeared in the online algorithm literature
74 for paging [1], scheduling [14] and bin packing [10] since the 1990s. Ours is the first paper, to
75 our knowledge, that considers lookahead in the context of packet scheduling.

76 *Our contributions.* We provide two results about PacketScheduling with 1-lookahead, restricted
77 to 2-bounded instances. First, in Section 4, we present an online algorithm for this problem
78 with competitive ratio of $\frac{1}{2}(\sqrt{13} - 1) \approx 1.303$. Then, in Section 5, we give a lower bound of
79 $\frac{1}{4}(1 + \sqrt{17}) \approx 1.281$ on the competitive ratio of algorithms with 1-lookahead which holds already
80 for the 2-bounded case. Our argument is an extension of the lower bound proof of ϕ in [11, 2, 5].
81 In fact, our lower bound result is more general: using only 2-bounded instances, for any integer
82 $\ell \geq 0$ we prove a lower bound of $\frac{1}{2(\ell+1)}(1 + \sqrt{5 + 8\ell + 4\ell^2})$ for online algorithms with ℓ -lookahead,
83 that is algorithms that at time t can see all packets arriving by time $t + \ell$. It follows that there
84 is no 1-competitive algorithm with any constant lookahead.

2 Definitions and Notation

Problem statement. Formally, we define the `PacketScheduling` problem as follows. The instance is a set of packets, with each packet p specified by a triple (r_p, d_p, w_p) , where r_p and $d_p \geq r_p$ are integers representing the *release time* and *deadline* of p , and $w_p \geq 0$ is a real number representing the *weight* of p . Time is discrete, divided into unit *time slots*, also called *steps*. A *schedule* assigns time slots to some subset of packets such that (i) any packet p in this subset is assigned a slot in the interval $[r_p, d_p]$, and (ii) each slot is assigned to at most one packet. The objective is to compute a schedule that maximizes the total weight of the scheduled packets, also called the *profit*.

In the *s-bounded* variant of `PacketScheduling`, we assume that each packet p in the instance satisfies $d_p \leq r_p + s - 1$. In other words, this packet must be scheduled within some specified number, at most s , of consecutive slots starting at its release time.

Online algorithms. In the online variant of `PacketScheduling`, which is the focus of our work, at any time t only the packets released at times up to t are revealed. Thus an online algorithm needs to decide which packet to schedule at time t (if any) without any knowledge of packets released after time t .

As is common in the area of online optimization, we measure the performance of an online algorithm by its competitive ratio. An algorithm \mathcal{A} is said to be *R-competitive* if, for all instances, the total weight of the optimal schedule (computed offline) is at most R times the weight of the schedule computed by \mathcal{A} .

We say that a packet p is *pending* for an algorithm at time t , if $r_p \leq t \leq d_p$ and p is not scheduled before time t . A (pending) packet p is *expiring* at time t if $d_p = t$, that is, it must be scheduled now or never. A packet p is *tight* if $r_p = d_p$; thus p is expiring already at its release time.

Algorithms with lookahead. In Sections 4 and 5, we investigate the `PacketScheduling` problem with ℓ -lookahead, mainly focusing on the case $\ell = 1$. With ℓ -lookahead, the problem definition changes so that at time t , an online algorithm can also see the packets that will be released at times $t + 1, t + 2, \dots, t + \ell$, in addition to the pending packets. Naturally, only a pending packet can be scheduled at time t .

Other terminology and assumptions. We will make several assumptions about our problem that do not affect the generality of our results. First, we can 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. Second, we assume that at each step there is at least one pending packet. (If not, we can always release a tight packet of weight 0 at each step.)

We define the *earliest-deadline relation* on packets, or *canonical ordering*, denoted \prec , where $x \prec y$ means that either $d_x < d_y$ or $d_x = d_y$ and $w_x > w_y$ (so the ties are broken in favor of heavier packets). At any step t , the algorithm maintains the earliest-deadline relation on the set of its pending packets. Throughout the paper, “earliest-deadline packet” means the earliest packet in the canonical ordering.

Regarding the adversary (optimal) schedule, we can assume that it satisfies the following *earliest-deadline property*: if packets p, p' are scheduled in steps t and t' , respectively, where $r_{p'} \leq t < t' \leq d_p$ (that is, p and p' can be swapped in the schedule without violating their release times and deadlines), then $p \prec p'$. This can be rephrased in the following useful way: at any step, the optimum schedule transmits the earliest-deadline packet among all the pending packets that it transmits in the future.

131 3 An Algorithm for 4-bounded Instances

132 In this section, we present a ϕ -competitive algorithm for 4-bounded instances. Ratio ϕ is of
 133 course optimal [11, 2, 5, see also Section 1]. Up until now, the best competitive ratio for
 134 4-bounded instances was $\sqrt{3} \approx 1.732$, achieved by algorithm $\text{EDF}_{\sqrt{3}}$ in [4]. Our algorithm can
 135 be seen as a modification of EDF_{ϕ} . Unlike EDF_{ϕ} , in some circumstances our algorithm may
 136 schedule a packet lighter than w_h/ϕ , where h is the heaviest pending packet.

137 We remark that our algorithm uses memory; in particular, it marks one pending packet under
 138 certain conditions. (In contrast to EDF_{α} , which is memoryless.) It is an interesting question
 139 whether there is a memoryless ϕ -competitive algorithm for 4-bounded instances.

140 **Algorithm ToggleH.** The algorithm maintains one mark that may be assigned to one of the
 141 pending packets. For a given step t , we choose the following packets from among all pending
 142 packets:

- 143 h = the heaviest packet,
- 144 s = the second-heaviest packet,
- 145 f = the earliest-deadline packet with $w_f \geq w_h/\phi$, and
- 146 e = the earliest-deadline packet with $w_e \geq w_h/\phi^2$.

147 We then proceed as follows:

```

if ( $h$  is not marked)  $\vee$  ( $w_s \geq w_h/\phi$ )  $\vee$  ( $d_e > t$ )
  schedule  $f$ 
  if there is a marked packet then unmark it
  if ( $d_h = t + 3$ )  $\wedge$  ( $d_f = t + 2$ ) then mark  $h$ 
else // ( $h$  is marked)  $\wedge$  ( $w_s < w_h/\phi$ )  $\wedge$  ( $d_e = t$ )
  schedule  $e$ 
  unmark  $h$ 

```

148 Note that when $f \neq h$, then the algorithm will always schedule f . This is because in this case f
 149 is a candidate for s , so the condition $w_s \geq w_h/\phi$ holds. The algorithm never specifically chooses
 150 s for scheduling – it is only used to determine if there is one more relatively heavy pending
 151 packet other than h . (But s may get scheduled if it so happens that $s = f$ or $s = e$.) Note
 152 also that, if $e \neq f$, then e is scheduled only in a very specific scenario, namely when all of the
 153 following conditions hold: e is expiring, h is marked, and $w_s < w_h/\phi$.

154 We distinguish two types of packets scheduled by Algorithm **ToggleH**: *f-packets*, scheduled
 155 using the first case, and *e-packets*, scheduled using the second case. Similarly, we refer to the
 156 steps as *f-steps* and *e-steps*.

157 **Intuition.** Let us give a high-level view of the analysis using charging schemes and an example
 158 that motivates both our algorithm and its analysis. The example consists of four packets
 159 j, k, f, h released in step 1, with deadlines 1, 2, 3, 4 and weights $1 - \varepsilon, 1 - \varepsilon, 1, \phi$ for a small $\varepsilon > 0$,
 160 respectively. The optimum schedules all packets.

161 Algorithm EDF_{ϕ} performs only *f-steps*; in our example it schedules f and h in steps 1 and
 162 2, while j and k are lost. Thus the ratio is larger than ϕ . (In fact, after optimizing the threshold
 163 and the weight of h , this is the tight example for $\text{EDF}_{\sqrt{3}}$ on 4-bounded instances.) **ToggleH**
 164 avoids this example by performing *e-step* in step 2 and scheduling k which has the role of e and
 165 s in the algorithm.

166 This example and its variants are also important for our analysis. We analyze the algorithms
 167 by charging schemes, where the weight of each packet scheduled by the adversary is charged
 168 to one or more of the slots of the algorithm's schedule. If the weight charged to each slot is at

169 most R times the weight of the packet scheduled by the algorithm in that slot, the algorithm
 170 is R -competitive. In the case of EDF, we charge the weight of each packet j scheduled by the
 171 adversary at time t either fully to the step where EDF schedules j , if it is before t , or fully to
 172 step t otherwise. In our example, the weight charged to step 1 is $2 - \varepsilon$ while EDF schedules
 173 only weight 1, giving the ratio 2. Considering steps 1 and 2 together leads to a better ratio and
 174 after balancing the threshold it gives the tight analysis of EDF $\sqrt{3}$.

175 Our analysis of **ToggleH** is driven by the variants of the example above where step 2 is
 176 an f -step. This may happen in several cases. One case is if in step 2 another packet s with
 177 $w_s \geq w_h/\phi$ arrives. If s is not scheduled in step 2, then s is pending in step 3, thus **ToggleH**
 178 schedules a relatively heavy packet in step 3, and we can charge a part of the weight of f ,
 179 scheduled in step 3 by the adversary, to step 3. This motivates the definition of regular up and
 180 back charges below and corresponds to Case 5.1 in the analysis. Another case is when the weight
 181 of k is charged to $1/\phi - \varepsilon$. Then **ToggleH** performs an f -step because k is not a candidate for
 182 e , thus the role of e is taken by the non-expiring packet h . However, then the weight of the
 183 four packets charged to steps 1 and 2 in the way described above is at most ϕ times the weight
 184 of f and h ; this corresponds to Case 5.2 of the analysis. Lemma 3.3 gives a subtle argument
 185 showing that in the 4-bounded case essentially these two variants of our example are the only
 186 difficult situations. Finally, in the original example, **ToggleH** schedules k in step 2 which is an
 187 e -step. Then again h is a pending heavy packet and we can charge some weight of f to step 3.
 188 Intuitively it is important that an e -step is performed only in a very specific situation where it
 189 is guaranteed that h can be scheduled in the next two steps (as it is marked) and that there is
 190 no other packet of comparable weight due to the condition $w_s < w_h/\phi$. Still, there is a case to
 191 be handled: If more packets arrive in step 3, it is also possible that the adversary schedules h
 192 already in step 2 and we need to redistribute its weight. This case motivates the definition of
 193 the special up and back charges below.

194 **Theorem 3.1.** *Algorithm **ToggleH** is ϕ -competitive on 4-bounded instances.*

195 *Proof.* Fix some optimal adversary schedule. Without loss of generality, we can assume that
 196 this schedule satisfies the earliest-deadline property (see Section 2).

197 Let t be the current step. By h , f , e , and s we denote the packets from the definition of
 198 **ToggleH**. By j we denote the packet scheduled by the adversary. By h' and h'' we denote the
 199 heaviest pending packets in steps $t + 1$ and $t + 2$, respectively. We use the same convention for
 200 packets f , e , s , and j .

201 Our analysis uses a new charging scheme which we now define. The adversary packet j
 202 scheduled in step t is charged according to the first case below that applies:

- 203 1. If t is an e -step and $j = h$, we charge w_h/ϕ to step t and w_h/ϕ^2 to step $t - 1$. We call
 204 these charges a *special up charge* and a *special back charge*, respectively. Note that the
 205 total charge is equal to $w_h = w_j$.
- 206 2. If j is pending for **ToggleH** in step t , charge w_j to step t . We call this charge a *full up*
 207 *charge*.
- 208 3. Otherwise j is scheduled before step t . We charge w_h/ϕ^2 to step t and $w_j - w_h/\phi^2$ to the
 209 step where **ToggleH** scheduled j . We call these charges a *regular up charge* and a *regular*
 210 *back charge*, respectively. We point out that the regular back charge may be negative, but
 211 this causes no problems in the proof.

212 We start with an easy observation that we use several times throughout the proof.

213 **Lemma 3.2.** *If an f -step t receives a regular back charge, then the up charge it receives is less*
 214 *than w_h/ϕ .*

215 *Proof.* For a regular up charge the lemma is trivial (with a slack of a factor of ϕ). For a full
 216 up charge, the existence of a back charge implies that the adversary schedules f after j , thus
 217 the earliest-deadline property of the adversary schedule implies that $j \prec f$, as both j and f are
 218 pending for the adversary at t . Thus **ToggleH** would schedule j if $w_j \geq w_h/\phi$. Finally, an f -step
 219 does not receive a special up charge. \square

220 We examine packets scheduled by **ToggleH** from left to right, that is in order of time. For
 221 each time step t , if p is the packet scheduled at time t , we want to show that the charge to step
 222 t is at most ϕw_p . However, as it turns out, this will not always be true. In one case we will also
 223 consider the next step $t + 1$ and the packet p' scheduled in step $t + 1$, and show that the total
 224 charge to steps t and $t + 1$ is at most $\phi(w_p + w_{p'})$.

225 Let t be the current step. We consider several cases.

226 Case 1: t is an e -step. By the definition of **ToggleH**, $w_e \geq w_h/\phi^2$ and $d_e = t$; the latter implies
 227 that step t receives no regular back charge. We further note that the heaviest pending packet h'
 228 in step $t + 1$ is either released at time $t + 1$ or it coincides with h , which is still pending and
 229 became unmarked by the algorithm in step t ; in either case h' is unmarked at the beginning of
 230 step $t + 1$, which implies that step $t + 1$ is an f -step. Thus, step t receives no special back charge,
 231 which, combined with the previous observation, implies it receives no back charge of any kind.

232 Now we claim that the up charge is at most w_h/ϕ . For a special or regular up charge this
 233 follows from its definition. For a full up charge, the job j is pending at time t for **ToggleH** and
 234 $j \neq h$ (as for $j = h$ the special charges are used). This implies that $w_j < w_h/\phi$, as otherwise
 235 $w_s \geq w_h/\phi$ and t would be an f -step. Thus the full charge is $w_j \leq w_h/\phi$ as well.

236 Using $w_e \geq w_h/\phi^2$, the charge is at most $w_h/\phi \leq \phi w_e$ and we are done.

237 Case 2: t is an f -step and t does not receive a back charge. Then t can only receive an up-charge,
 238 and this up charge is at most $w_h \leq \phi w_f$, where the inequality follows from the definition of f .

239 Case 3: t is an f -step and t receives a special back charge. From the definition of special charges,
 240 the next step is an e -step, and therefore h' is marked at its beginning. Since the only packet
 241 that may be marked after an f -step is h , we thus have $h = h' = j'$, and the special back charge
 242 is w_h/ϕ^2 . Since $f \prec h$, the adversary cannot schedule f after step t , so step t cannot receive a
 243 regular back charge.

244 We claim that the up charge to step t is at most w_f . Indeed, a regular up charge is at
 245 most $w_h/\phi^2 \leq w_f$, and a special up charge does not happen in an f -step. To show this bound
 246 for a full up charge, assume for contradiction that $w_j > w_f$. This implies that $j \neq f$ and,
 247 since **ToggleH** scheduled f , we have $d_j > d_f$. In particular j is pending at time $t + 1$. Thus
 248 $w_{s'} \geq w_j > w_f \geq w_h/\phi$, contradicting the fact that $t + 1$ is an e -step. Therefore the full charge
 249 is $w_j \leq w_f$, as claimed.

250 As $w_h \leq \phi w_f$, the total charge to t is at most $w_f + w_h/\phi^2 \leq w_f + w_f/\phi = \phi w_f$.

251 Case 4: t is an f -step, t receives a regular back charge and no special back charge, and $f = h$.
 252 The up charge is at most w_h/ϕ by Lemma 3.2 and the back charge is at most w_h , thus the total
 253 charge is at most $w_h + w_h/\phi = \phi w_h$, and we are done.

254 Case 5: t is an f -step, t receives a regular back charge and no special back charge, and $f \neq h$.
 255 Let \bar{t} be the step when the adversary schedules f . We distinguish two sub-cases.

256 Case 5.1: In step \bar{t} , a packet of weight at least w_h/ϕ is pending for the algorithm. Then the regular
 257 back charge to t is at most $w_f - (w_h/\phi)/\phi^2 = w_f - w_h/\phi^3$. As the up charge to t is at most w_h/ϕ by
 258 Lemma 3.2, the total charge to t is at most $w_h/\phi + w_f - w_h/\phi^3 = w_f + w_h/\phi^2 \leq (1 + 1/\phi)w_f = \phi w_f$,
 259 and we are done.

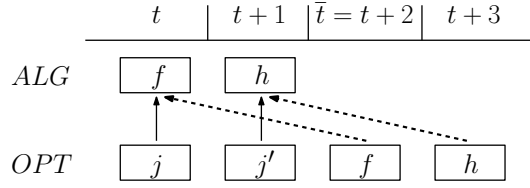


Figure 1: An illustration of the situation in Case 5.2. Up charges are denoted by solid arrows and back charges by dashed arrows.

260 **Case 5.2:** In step \bar{t} , no packet of weight at least w_h/ϕ is pending for the algorithm. In this case
 261 we consider the charges to steps t and $t + 1$ together. First, we claim the following.

262 **Lemma 3.3.** *ToggleH schedules h in step $t + 1$. Furthermore, step $t + 1$ receives no special*
 263 *charge and it receives an up charge of at most w_h/ϕ^2 .*

264 *Proof.* Since $f \neq h$, we have $f \prec h$ and thus, using also the definition of \bar{t} and 4-boundedness,
 265 $\bar{t} \leq d_f < d_h \leq t + 3$. The case condition implies that h is not pending at \bar{t} , thus **ToggleH**
 266 schedules h before \bar{t} . The only possibility is that **ToggleH** schedules h in step $t + 1$, $\bar{t} = d_f = t + 2$,
 267 and $d_h = t + 3$; see Figure 1 for an illustration. This also implies that **ToggleH** marks h in step t .

268 We claim that $w_{s'} < w_h/\phi$. Indeed, otherwise either s' is pending in step $t + 2$, contradicting
 269 the condition of Case 5.2, or $d_{s'} = t + 1 < d_h$, thus s' is a better candidate for f' than h , which
 270 contradicts the fact that the algorithm scheduled $f' = h$.

271 The claim also implies that $h' = h$, as otherwise $w_{s'} \geq w_h$. Since $h = h'$ is scheduled in step
 272 $t + 1$, there is no marked packet in step $t + 2$ and $t + 2$ is an f -step; thus there is no special back
 273 charge to $t + 1$.

274 We note that step $t + 1$ is also an f -step, since **ToggleH** schedules h in step $t + 1$ and $d_h > t + 1$.
 275 Since $h' = h$ is marked when step $t + 1$ starts and $w_{s'} < w_h/\phi$, the reason that step $t + 1$ is an
 276 f -step must be that $d_{e'} > t + 1$.

277 There is no special up charge to step $t + 1$ as it is an f -step. If the up charge to step $t + 1$ is
 278 a regular up charge, by definition it is at most $w_{h'}/\phi^2 = w_h/\phi^2$ and the lemma holds.

279 The only remaining case is that of a full up charge to step $t + 1$ from a packet j' scheduled by
 280 the adversary in step $t + 1$ and pending for **ToggleH** in step $t + 1$. Since $j' \neq h$, it is a candidate
 281 for s' , and thus $w_{j'} < w_h/\phi \leq w_f$. The earliest-deadline property of the adversary schedule
 282 implies that $j' \prec f$; together with $d_f = t + 2$ and $w_{j'} < w_f$ this implies $d_{j'} = t + 1$. Therefore
 283 $w_{j'} < w_h/\phi^2$, as otherwise j' is a candidate for e' , but we have shown that $d_{e'} > t + 1$. Thus the
 284 regular up charge is at most $w_{j'} < w_h/\phi^2$ and the lemma holds also in the remaining case. \square

285 By Lemma 3.3, step $t + 1$ receives no special charge and an up charge of at most w_h/ϕ^2 and
 286 **ToggleH** schedules h in step $t + 1$. Step $t + 1$ thus also receives a regular back charge of at most
 287 w_h . So the total charge to step $t + 1$ is at most $w_h/\phi^2 + w_h \leq w_f/\phi + w_h$. Moreover, using
 288 Lemma 3.2, the total charge to step t is at most $w_h/\phi + w_f$. Thus, the total charge to these
 289 two steps is at most $(w_h/\phi + w_f) + (w_f/\phi + w_h) = \phi(w_f + w_h)$, as f and h are the two packets
 290 scheduled by **ToggleH**.

291 In each case we have shown that a step or a pair of consecutive steps receive a total charge of
 292 at most ϕ times the weight of packets scheduled in these steps. Thus **ToggleH** is ϕ -competitive
 293 for the 4-bounded case. \square

294 4 An Algorithm for 2-Bounded Instances with Lookahead

295 In this section, we present an algorithm for *2-bounded PacketScheduling with 1-lookahead*, as
 296 defined in Section 2.

297 Consider some online algorithm \mathcal{A} . Recall that, for a time step t , packets *pending* for \mathcal{A} are
 298 those that are released at or before time t and have neither expired nor been scheduled by \mathcal{A}
 299 before time t . *Lookahead* packets at time t are the packets with release time $t + 1$. For \mathcal{A} , we
 300 define the *plan* in step t to be the optimal schedule in the time interval $[t, \infty)$ that consists of
 301 pending and lookahead packets at time t and has the earliest-deadline property. For 2-bounded
 302 instances, this plan will only use slots $t, t + 1$ and $t + 2$. We will typically denote the packets in
 303 the plan scheduled in these slots by p_1, p_2, p_3 , respectively. The earliest-deadline property then
 304 implies that if both p_1 and p_2 have release time t and deadline $t + 1$ then p_1 is heavier than p_2 .
 305 A similar condition holds for p_2 and p_3 .

306 **Algorithm** COMPAREWITHBIAS(α). Fix some parameter $\alpha > 1$. At any time step t , the
 307 algorithm proceeds as follows:

```

  let  $p_1, p_2, p_3$  be the plan at time  $t$ 
  if  $r_{p_2} = t$  and  $w_{p_1} < \min(w_{p_2}, w_{p_3}, \frac{1}{2\alpha}(w_{p_2} + w_{p_3}))$ 
    then schedule  $p_2$ 
  else schedule  $p_1$ 
  
```

308 Note that if the algorithm schedules p_2 then p_1 must be expiring, for otherwise $w_{p_1} > w_{p_2}$
 309 (by canonical ordering). Also, the scheduled packet is at least as heavy as the heaviest expiring
 310 packet q , since clearly $w_{p_1} \geq w_q$ and the algorithm schedules p_2 only if $w_{p_1} < w_{p_2}$.

311 **Theorem 4.1.** *The algorithm COMPAREWITHBIAS(α) is R -competitive for packet scheduling*
 312 *on 2-bounded instances for $R = \frac{1}{2}(\sqrt{13} - 1) \approx 1.303$ if $\alpha = \frac{1}{4}(\sqrt{13} + 3) \approx 1.651$.*

313 Let ALG be the schedule produced by COMPAREWITHBIAS. Let us consider an optimal
 314 schedule OPT (a.k.a. schedule of the adversary) satisfying the canonical ordering, i.e., if a packet
 315 x is scheduled before a packet y in OPT then either y is released after x is scheduled or $x \prec y$.
 316 Recall that we are assuming w.l.o.g. that the weights of packets are different.

317 The analysis of COMPAREWITHBIAS is based on a charging scheme. First we define a few
 packets by their schedule times:

- i = packet scheduled in step $t - 1$ in OPT,
- j = packet scheduled in step t in OPT,
- k = packet scheduled in step $t + 1$ in OPT,
- ℓ = packet scheduled in step $t + 2$ in OPT,
- e = packet scheduled in step $t - 1$ in ALG,
- f = packet scheduled in step t in ALG,
- g = packet scheduled in step $t + 1$ in ALG,
- h = packet scheduled in step $t + 2$ in ALG.

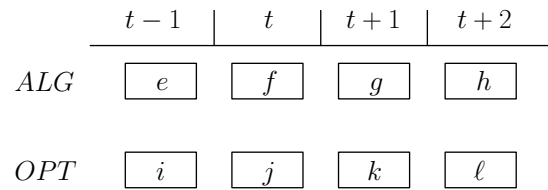


Figure 2: Packet definition.

318

319 **Informal description of charging.** We use three types of charges. The adversary's packet j
 320 in step t is charged using a *full charge* either to step $t - 1$ if ALG schedules j in step $t - 1$ or to
 321 step t if $w_f \geq w_j$ (including the case $f = j$) and f is not in step $t + 1$ in OPT; the last condition
 322 assures that step t does not receive two full charges.

323 The second type are *split charges* that occur in step t if $w_f > w_j$, j is pending in step t in
 324 ALG and f is in step $t + 1$ in OPT, i.e., step t receives a full back charge from f . In this case, we

325 distribute the charge from j to f and another relatively large packet f' scheduled in step $t + 1$
326 or $t + 2$ in **ALG**; we shall prove that one of these steps satisfies $2\alpha \cdot w_j < w_f + w'_f$. We charge to
327 step $t + 2$ only when it is necessary, which allows us to prove that split-charge pairs are pairwise
328 disjoint. Also, in this case we analyze the charges to both steps together, thus it is not necessary
329 to fix a distribution of the weight to the two steps.

330 The remaining case is when $w_f < w_j$ and j is not scheduled in $t - 1$ in **ALG**. We analyze
331 these steps in maximal consecutive intervals, called *chains* and the corresponding charges are
332 *chain charges*. Inside each chain we distribute the charge of each packet j scheduled at t in **OPT**
333 to steps $t - 1$, t and $t + 1$, if these steps are also in the chain. The distribution of weights shall
334 depend on a parameter δ . Packets at the beginning and at the end of the chain are charged in a
335 way that minimizes the charge to steps outside of the chain. In particular, the step before a
336 chain receives no charge from the chain.

Parameters and constants. We set the parameter α and constants δ and R which we will use in the analysis so that they satisfy the following equalities:

$$2 - \delta - \frac{R + 2\delta - 1}{\alpha} = R \quad (1)$$

$$1 - 2\delta + 2\alpha\delta = R \quad (2)$$

$$1 + \frac{1}{2\alpha} = R \quad (3)$$

337 By solving these equations we get $\alpha = \frac{1}{4}(\sqrt{13} + 3) \approx 1.651$, $\delta = \frac{1}{6}(5 - \sqrt{13}) \approx 0.232$, and
338 $R = \frac{1}{2}(\sqrt{13} - 1) \approx 1.303$. We will prove that the algorithm is R -competitive.

We also use the following properties of these constants:

$$2 - R - 3\delta = 0 \quad (4)$$

$$2 - R - 2\delta > 0 \quad (5)$$

$$1 - \delta - \frac{R - 1 + 2\delta}{2\alpha} > 0 \quad (6)$$

$$1 - \frac{R}{2\alpha} > 0 \quad (7)$$

$$3\alpha\delta < R \quad (8)$$

$$2 - \frac{R}{\alpha} < R \quad (9)$$

339 where (4) follows from (1) and (2) and the strict inequalities can be verified numerically.

340 **Notations and the charging scheme.** A step t for which $w_f < w_j$ and j is pending in step
341 t in **ALG** is called a *chaining step*. A maximal sequence of successive chaining steps is called a
342 *chain*. The chains with a single step are called *singleton chains*, the chains with at least two
343 steps are called *long chains*.

344 The pair of steps that receives a split charge from the same packet is called a *split-charge*
345 *pair*. The charging scheme does not specify the distribution of the weight to the two steps of the
346 split-charge pair, as the charges to them are analyzed together.

347 Packet j scheduled in **OPT** at time t is charged according to the first rule below that applies.
348 See Figures 3 and 4 for an illustration of different types of charges.

- 349 1. If j is scheduled in step $t - 1$ in **ALG** (that is, $e = j$), charge w_j to step $t - 1$. We call this
350 charge a *full back charge*.
- 351 2. If $w_f \geq w_j$ and f is not scheduled in step $t + 1$ in **OPT** (in particular, if $j = f$), charge w_j
352 to step t . We call this charge a *full up charge*.

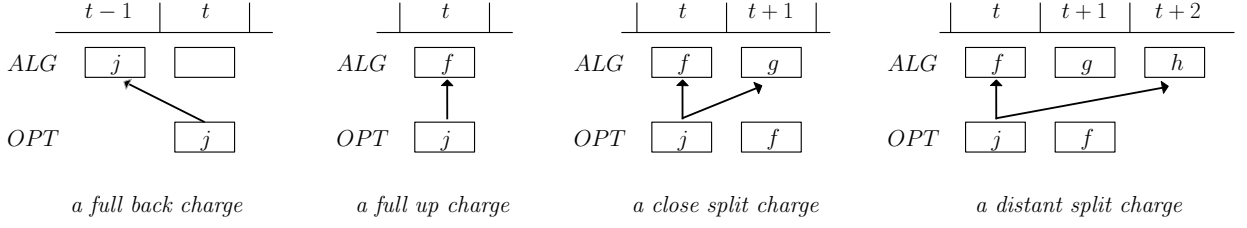


Figure 3: Non-chaining charges. Note that for split charges f is scheduled in step $t + 1$ in OPT which follows from the fact that we do not charge j using a full up charge.

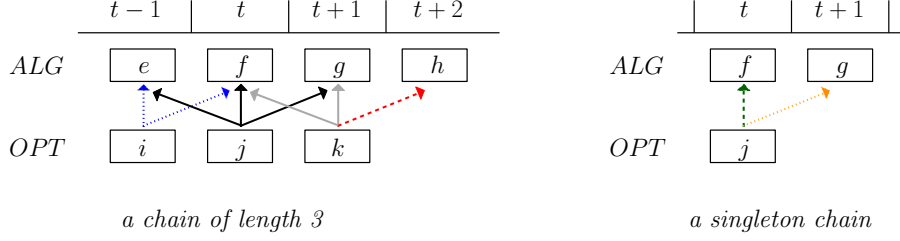


Figure 4: On the left, a chain of length 3 starting in step $t - 1$ and ending in step $t + 1$. The *chain beginning charges* are denoted by dotted (blue) lines, the *chain end charges* are denoted by gray lines and the *forward charge from a chain* is depicted by a dashed (red) arrow. Black arrows denote the *chain link charges*. On the right, an example of a singleton chain, with the *up charge from a singleton chain* denoted with a dashed (green) line and the *forward charge from a singleton chain* denoted with a dotted (orange) line.

- 353 3. If $w_f > w_j$ and at least one of the following holds (in both cases, p_1 is the first packet in
354 the plan at time t):
- 355 • $2\alpha \cdot w_{p_1} < w_f + w_g$, or
 - 356 • g does not get a full back charge and $2\alpha \cdot (w_{p_1} - w_g) < w_f + w_g$,
- 357 then charge w_j to the pair of steps t and $t + 1$. We call this charge a *close split charge*.
- 358 4. If $w_f > w_j$, then charge w_j to the pair of steps t and $t + 2$. We call this charge a *distant*
359 *split charge*.
- 360 5. Otherwise step t is a chaining step, as $w_f < w_j$ and ALG does not schedule j in step $t - 1$
361 by the previous cases. We distinguish the following subcases.
- 362 (a) If step t is (the only step of) a singleton chain, then charge $\min(w_j, R \cdot w_f)$ to step t
363 and $w_j - R \cdot w_f$ to step $t + 1$ if $w_j > R \cdot w_f$. We call these charges an *up charge from*
364 *a singleton chain* and a *forward charge from a singleton chain*.
 - 365 (b) If step t is the first step of a long chain, charge $2\delta \cdot w_j$ to step t , and $(1 - 2\delta) \cdot w_j$ to
366 step $t + 1$. We call these charges *chain beginning charges*.
 - 367 (c) If step t is the last step of a long chain, charge $\delta \cdot w_j$ to step $t - 1$, $(R - 1 + 2\delta) \cdot w_f$ to
368 step t , and $(1 - \delta) \cdot w_j - (R - 1 + 2\delta) \cdot w_f$ to step $t + 1$. We call these charges *chain end*
369 *charges*; the charge to step $t + 1$ is called a *forward charge from a chain*. (Note that
370 we always have $(1 - \delta) \cdot w_j > (R - 1 + 2\delta) \cdot w_f$, since $w_j > w_f$ and $1 - \delta = R - 1 + 2\delta$
371 which follows from (4).)
 - 372 (d) Otherwise, i.e., if step t is inside a long chain, charge $\delta \cdot w_j$ to step $t - 1$, $\delta \cdot w_j$ to step
373 t , and $(1 - 2\delta) \cdot w_j$ to step $t + 1$. We call these charges *chain link charges*.

374 To estimate the competitive ratio we need to show that each step or a pair of steps
 375 does not receive too much charge. We start with a useful observation about plans of Al-
 376 gorithm COMPAREWITHBIAS(α), that will be used multiple times in our proofs.

377 **Lemma 4.2.** *Consider a time t , where the algorithm has two pending packets a, b and a lookahead*
 378 *packet c with the following properties: $d_a = t$, $(r_b, d_b) = (t, t + 1)$, $(r_c, d_c) = (t + 1, t + 2)$, and*
 379 *$w_a < \min(w_b, w_c)$. If the algorithm schedules packet a in step t then the plan at time t is a, b, c ,*
 380 *and $2\alpha \cdot w_a \geq w_b + w_c$.*

381 *Proof.* We claim that there is no pending or lookahead packet $q \notin \{b, c\}$ heavier than a . Suppose
 382 for a contradiction that such a q exists. Then a schedule containing packets q, b, c in some order
 383 is feasible and has larger profit than a, b, c . This implies that the plan does not contain a and
 384 thus a cannot be scheduled, contradicting the assumption of the lemma.

385 The schedule a, b, c is feasible and the claim above implies that it is optimal, thus it is
 386 the plan. It remains to show that $2\alpha \cdot w_a \geq w_b + w_c$, which follows easily by a contradiction:
 387 Otherwise $2\alpha \cdot w_a < w_b + w_c$ and COMPAREWITHBIAS(α) would schedule b , contradicting the
 388 assumption of the lemma. \square

389 Next, we will provide an analysis of full, split and chain charges, starting with full and split
 390 charges. We prove several lemmas from which the analysis follows. We fix some time slot t , and
 391 use the notation from Figure 2 for packets at time slots $t - 1, t, t + 1$ and $t + 2$ in the schedule
 392 of the algorithm and the optimal schedule.

393 **Analysis of full charges.** Using Rules 1 and 2, if step t receives a full back charge, then
 394 the condition of Rule 2 guarantees that it will not receive a full up charge. This gives us the
 395 following observation.

396 **Lemma 4.3.** *Step t receives at most one full charge, i.e., a charge by Rule 1 or 2.*

397 **Analysis of split charges.** We now analyze close and distant split charges. The crucial
 398 property of split charges is that, similar to full charges, each step receives at most one split
 399 charge. Before we prove this, we establish several useful properties of split charges.

400 **Lemma 4.4.** *Let the plan at time t be p_1, p_2, p_3 . If j is charged using a close or a distant split*
 401 *charge, then the following holds:*

- 402 (a) j is not scheduled by the algorithm in step $t - 1$, i.e., j is pending for the algorithm in step t .
- 403 (b) $d_f = t + 1$ and f is scheduled in step $t + 1$ in OPT (that is, $k = f$). In particular, step t
 404 receives a full back charge.
- 405 (c) $d_j = t$ and $w_j \leq w_{p_1}$.
- 406 (d) $p_2 = f$.

407 *Proof.* By Rule 1, packet j would be charged using a full back charge if it were scheduled in
 408 step $t - 1$, implying (a). The case conditions for split charges in the charging scheme imply
 409 that OPT schedules f in step $t + 1$ and $w_f > w_j$. Now (b) follows from the fact that we do not
 410 charge j using a full up charge.

411 To show (c), note that if j is not expiring, then j and f would have equal deadlines. As
 412 we also have $w_f > w_j$, f would be scheduled before j in OPT by the canonical ordering, a
 413 contradiction. The inequality $w_j \leq w_{p_1}$ now follows from the definition of the plan.

414 It remains to prove (d). Towards contradiction, suppose that $f = p_1$. We know that j is
 415 expiring and thus it is not in the plan. If $d_{p_2} = t + 1$ then the optimality of the plan implies
 416 $w_{p_2} > w_j$ (otherwise j, f, p_3 would be a better plan), so, since p_2 is not in OPT, we could improve
 417 OPT by scheduling f in step t and p_2 in step $t + 1$.

418 Next, assume that $d_{p_2} = t + 2$. The optimality of the plan implies that $w_{p_2} > w_j$ and
 419 $w_{p_3} > w_j$. Since both p_2, p_3 have deadline $t + 2$, at least one of them is not scheduled in OPT.
 420 So OPT could be improved by scheduling f in step t and one of p_2 or p_3 in step $t + 1$. In both
 421 cases we get a contradiction with the optimality of OPT. \square

422 We show a useful lemma about a distant split charge from which we derive an upper bound
 423 on w_j , similar as the upper bound in the definition of close split charge.

424 **Lemma 4.5.** *If j is charged using a distant split charge, then $w_g < w_{p_3}$ where p_3 is the third
 425 packet in the plan at time t , and $d_g = t + 1$.*

426 *Proof.* Suppose that $w_g \geq w_{p_3}$. Then, from Lemma 4.4(d) and the choice of $p_2 = f$ in the
 427 algorithm, we have that $2\alpha w_{p_1} < w_{p_2} + w_{p_3} \leq w_f + w_g$, so we would use the close split charge in
 428 step t , not the distant one. Thus $w_g < w_{p_3}$, as claimed.

429 To prove the second part, if we had $d_g = t + 2$ then, since the algorithm chose g in step $t + 1$
 430 and also $d_{p_3} = t + 2$, we would also have $w_g \geq w_{p_3}$ – a contradiction. \square

431 **Lemma 4.6.** *If j is charged using a distant split charge then $2\alpha \cdot w_j < w_f + w_h$. (Recall that h
 432 is the packet scheduled in step $t + 2$ in ALG.)*

433 *Proof.* Let p_1, p_2, p_3 be the plan in step t . By Lemma 4.4(d) we have that $f = p_2$. Thus
 434 $2\alpha \cdot w_{p_1} < w_{p_2} + w_{p_3}$ by the definition of the algorithm. By Lemma 4.4(c), j is expiring and
 435 $w_j \leq w_{p_1}$. As $g \neq p_3$ by Lemma 4.5, the algorithm has p_3 pending in step $t + 2$ where it is expiring,
 436 implying that $w_{p_3} \leq w_h$. Putting it all together, we get $2\alpha w_j \leq 2\alpha w_{p_1} < w_{p_2} + w_{p_3} \leq w_f + w_h$. \square

437 For a split charge from j in step t , let t' be the other step that receives the split charge from
 438 j ; that is, $t' = t + 1$ for a close split charge and $t' = t + 2$ for a distant split charge. We now
 439 show that split-charge pairs are pairwise disjoint.

440 **Lemma 4.7.** *If j is charged using a split charge to a pair of steps t and t' , then neither of t
 441 and t' is involved in another pair that receives a split charge from a packet $j' \neq j$.*

442 *Proof.* No matter which split charge we use for j , using Lemma 4.4(b), step $t + 1$ does not
 443 receive a split charge from $k = f$. By a similar argument, since j is not scheduled in step $t - 1$ in
 444 ALG, step t does not receive a close split charge from the packet scheduled in step $t - 1$ in OPT.

445 It remains to prove that if j is charged using a distant split charge, then the packet ℓ
 446 scheduled in step $t + 2$ in OPT is not charged using a split charge. (This also ensures that step t
 447 does not receive a distant split charge from a packet scheduled in step $t - 2$ in OPT.)

448 For a contradiction, suppose that packet ℓ is charged using a split charge. Let p_1, p_2, p_3 be
 449 the plan in step t . Recall that g and h are the packets scheduled in steps $t + 1$ and $t + 2$ in ALG.

450 From Lemma 4.5, step $t + 1$ does not receive a full back charge. Since we did not apply the
 451 close split charge for j in Rule 3, we must have

$$2\alpha(w_{p_1} - w_g) \geq w_f + w_g \geq w_f. \quad (10)$$

452 By Lemma 4.4(b) applied to step $t + 2$, we get $d_h = t + 3$. Since $d_{p_3} = t + 2$, we get
 453 $w_{p_3} < w_h$. We now use Lemma 4.2 for step $t + 1$ with $a = g, b = p_3$, and $c = h$. We note that
 454 all the assumptions of the lemma are satisfied: we have $d_g = t + 1$, $(r_{p_3}, d_{p_3}) = (t + 1, t + 2)$,
 455 $(r_h, d_h) = (t + 2, t + 3)$, and $w_g < w_{p_3} < w_h$. This gives us that $2\alpha w_g \geq w_{p_3} + w_h > w_{p_3}$.

456 Since the algorithm schedules $f = p_2$ in step t , we have $2\alpha w_{p_1} < w_f + w_{p_3}$. Subtracting the
 457 inequality derived in the previous paragraph, we get $2\alpha(w_{p_1} - w_g) < (w_f + w_{p_3}) - w_{p_3} = w_f - \alpha$
 458 contradiction with (10). This completes the proof. \square

459 The lemmas above allow us to estimate the total of full and split charges.

460 **Lemma 4.8.** *If j is charged using a split charge to a pair of steps t and t' , then the total of full*
 461 *and split charges to steps t and t' does not exceed $R \cdot (w_f + w_{f'})$ where f' is the packet scheduled*
 462 *in step t' in ALG.*

463 *Proof.* Each of steps t and t' may receive a full charge, but each step at most one full charge
 464 from a packet of smaller or equal weight by Lemma 4.3 and charging rules.

First suppose that we use a distant split charge, or we use a close split charge and $2\alpha \cdot w_{p_1} < w_f + w_{f'}$. Then we have $2\alpha \cdot w_j < w_f + w_{f'}$ by Lemma 4.6 for a distant split charge or by $w_j \leq w_{p_1}$ from Lemma 4.4(c) for a close split charge. Thus the total of full and split charges to steps t and t' is upper bounded by

$$w_f + w_{f'} + w_j < w_f + w_{f'} + \frac{w_f + w_{f'}}{2\alpha} = \left(1 + \frac{1}{2\alpha}\right) \cdot (w_f + w_{f'}) = R \cdot (w_f + w_{f'})$$

465 where we used (3) in the last step.

Otherwise, i.e., if we use a close split charge and $2\alpha \cdot w_{p_1} \geq w_f + w_{f'}$, then step $t' = t + 1$ does not get a full back charge and we have $2\alpha \cdot (w_{p_1} - w_{f'}) < w_f + w_{f'}$ by Rule 3. By Lemma 4.4(c) we have $w_j \leq w_{p_1}$ and $2\alpha \cdot (w_j - w_{f'}) < w_f + w_{f'}$. Also, step $t + 1$ does not receive a full up charge by Lemma 4.4(b). We thus bound the total of full and split charges to steps t and $t + 1$ by

$$w_f + w_j < w_f + \frac{w_f + (2\alpha + 1) \cdot w_{f'}}{2\alpha} = \left(1 + \frac{1}{2\alpha}\right) \cdot (w_f + w_{f'}) = R \cdot (w_f + w_{f'})$$

466 using (3) in the last step again. □

467 **Analysis of chain charges.** We now analyze chaining steps starting with a lemma below
 468 consisting of several useful observations. In particular, Part (c) motivates the name “chaining”
 469 for such steps.

470 **Lemma 4.9.** *If step t is a chaining step, then the following holds:*

471 (a) $d_j = t + 1$,

472 (b) $d_f = t$.

473 *Moreover, if step $t + 1$ is also a chaining step, then*

474 (c) j is scheduled by the algorithm in step $t + 1$, i.e., $g = j$,

475 (d) $2\alpha \cdot w_f \geq w_j + w_k$ (recall that k is the packet scheduled in step $t + 1$ in *OPT*).

476 *Proof.* Recall that Algorithm COMPAREWITHBIAS(α) never schedules a packet lighter than the
 477 heaviest expiring packet. As in step t it schedules f with $w_f < w_j$ (by Rule 5 for chain charges),

478 (a) follows. Furthermore, it follows that f is expiring in step t , because otherwise the algorithm
 479 would schedule j , since both would have the same deadline and j is heavier. Thus (b) holds as
 480 well.

481 Now assume that step $t + 1$ is also in the chain and for a contradiction suppose that $g \neq j$.
 482 Since j is expiring and pending for the algorithm in step $t + 1$, we have $w_g > w_j$ and $w_k > w_g$
 483 as step $t + 1$ is in the chain.

484 Summarizing, the algorithm sees all packets f, j, g, k in step t (some are pending and some
 485 may be lookahead packets), and they are all distinct packets with $w_f < w_j < w_g < w_k$, $d_f = t$,
 486 $(r_j, d_j) = (t, t + 1)$, and both g and k can be feasibly scheduled at time $t + 1$. Thus, independently
 487 of the release times and deadlines of g and k , the plan at time t containing f would not be
 488 optimal – a contradiction. This proves that (c) holds.

489 Finally, we show (d). Since f is expiring in step t by (b) and both j and k are considered for
 490 the plan at time t and satisfy $(r_j, d_j) = (t, t + 1)$, $(r_k, d_k) = (t + 1, t + 2)$, $w_f < w_j < w_k$, we use
 491 Lemma 4.2 with $a = f, b = j$, and $c = k$ and get the inequality in (d). □

492 First we show that chaining steps does not receive charges of other types.

493 **Lemma 4.10.** *If step t is a chaining step, then t does not receive a full charge or a split*
 494 *charge.*

495 *Proof.* By Lemma 4.9, f is expiring, thus step t does not receive a full back charge. As $w_j > w_f$,
 496 the step also does not get a full up charge or a split charge from step t . So it remains to show
 497 that f does not receive a split charge.

498 First observe that step t cannot receive a close split charge from step $t - 1$ in OPT, because
 499 j is pending in step t in ALG, while Lemma 4.4(b) states that a split charge from step $t - 1$
 500 would require j to be scheduled at time $t - 1$ in ALG.

501 Finally, we show that step t does not receive a distant split charge. For a contradiction,
 502 suppose that step t receive a distant split charge from the packet x scheduled in step $t - 2$
 503 in OPT. Let p_1, p_2, p_3 be the plan in step $t - 2$. According to Lemma 4.4(d) and (b), p_2 is
 504 scheduled in step $t - 2$ in ALG and in step $t - 1$ in OPT. Moreover, by Lemma 4.4(c), x is
 505 pending and expiring in step $t - 2$ and $w_{p_1} \geq w_x$. As the algorithm scheduled p_2 in step $t - 2$
 506 we get $r_{p_2} = t - 2$ and $w_{p_1} < w_{p_3}$.

507 Observe that p_3 is not scheduled in OPT, since it is expiring in step t and j is not expiring,
 508 by Lemma 4.9(a). Thus we could increase the weight of OPT if we scheduled p_2 in step $t - 2$
 509 instead of x and p_3 in step $t - 1$. This contradicts the optimality of OPT. \square

510 We now analyze how much charge does each chaining step get.

511 **Lemma 4.11.** *If step t is a chaining step, then it receives a charge of at most $R \cdot w_f$.*

512 *Proof.* By Lemma 4.10, step t does not receive any full or split charges; therefore we just need
 513 to prove that the total of chain charges to step t does not exceed $R \cdot w_f$.

514 Case 1: t is the last step of a chain. If t is the only step in the chain then Rule 5a implies directly
 515 that the charge to t is at most $R \cdot w_f$. Otherwise, Lemma 4.9(c) implies that f is scheduled in
 516 step $t - 1$ in OPT, and thus the charge from step $t - 1$ is $(1 - 2\delta) \cdot w_f$. The charge from step t is
 517 $(R - 1 + 2\delta) \cdot w_f$ by Rule 5c. So the total charge is at most $R \cdot w_f$.

518 Case 2: t is not the last step of a chain. Since step $t + 1$ is also in the chain, by Lemma 4.9(c)
 519 we have that j is scheduled in step $t + 1$ in ALG and OPT has a packet k with $w_k > w_j$ in step
 520 $t + 1$. From Lemma 4.9(d) we know that $2\alpha \cdot w_f \geq w_j + w_k$.

There are two sub-cases. If t is the first step of the chain, then the charge to t is at most

$$2\delta \cdot w_j + \delta \cdot w_k \leq \frac{3}{2}\delta \cdot (w_j + w_k) \leq 3\alpha\delta \cdot w_f < R \cdot w_f,$$

where the last inequality follows from (8). Otherwise, using Lemma 4.9(c), f is scheduled in
 step $t - 1$ in OPT, so the total charge to step t is at most

$$(1 - 2\delta) \cdot w_f + \delta \cdot w_j + \delta \cdot w_k \leq (1 - 2\delta) \cdot w_f + 2\alpha\delta \cdot w_f = R \cdot w_f$$

521 where the last equality follows from (2). \square

522 **Analysis of forward charges from chains.** We now show that a forward charge from a chain
 523 does not cause an overload on the step just after the chain which may also get both a full charge
 524 and a split charge. (This is the only case when a step receives charges of all three types.)

525 For the following lemmas we assume that step $t - 1$ is a chaining step. Recall that i is the
 526 packet scheduled in step $t - 1$ in OPT and e is the packet scheduled in step $t - 1$ in ALG. First,
 527 we prove some useful observations.

528 **Lemma 4.12.** *If step t receives a forward charge from a chain, then the following holds*

529 (a) $j \neq e$ (that is, j is not charged using a full back charge),

530 (b) $w_f \geq w_j$,

531 (c) $w_f \geq w_i$.

532 *Moreover, if step t is not in a split-charge pair:*

533 (d) j is charged using a full up charge to step t ,

534 (e) step t does not receive a back charge.

535 *Proof.* Part (a) holds because e is expiring in step $t - 1$, by Lemma 4.9(b). Part (b) follows from
536 (a) and the fact that step t is not chaining.

537 To show (c), Lemma 4.9(a) implies that $d_i = t$. Also, since $i \neq e$, i is pending in ALG in step
538 t . Now (c) follows, because each packet scheduled by the algorithm is at least as heavy as the
539 the heaviest expiring packet.

540 Part (d) follows from (a) and (b) and the assumption that t is not in a split-charge pair.

541 Part (e) follows from (d) and Lemma 4.3. \square

542 Note that f may be the same packet as i or j . We start with the case in which f is not in a
543 split-charge pair.

544 **Lemma 4.13.** *If step t receives a forward charge from a chain C and t is not in a split-charge*
545 *pair, then the total charge to step t is at most $R \cdot w_f$.*

546 *Proof.* The proof is by case analysis, depending on the relative weights of j and e , and on
547 whether C is a singleton or a long chain. In all cases we use Lemma 4.12 and the charging rules
548 to show upper bounds on the total charge.

549 Case 1: $w_j < w_e$.

Case 1.1: The chain C is long. The charge to step t is then at most

$$\begin{aligned}
 w_j + (1 - \delta) \cdot w_i - (R - 1 + 2\delta) \cdot w_e &< w_j + (1 - \delta) \cdot w_i - (R - 1 + 2\delta) \cdot w_j \\
 &= (2 - R - 2\delta) \cdot w_j + (1 - \delta) \cdot w_i \\
 &\leq (2 - R - 2\delta) \cdot w_f + (1 - \delta) \cdot w_f && (11) \\
 &= (3 - R - 3\delta) \cdot w_f = w_f. && (12)
 \end{aligned}$$

550 To justify inequality (11), note that $2 - R - 2\delta \geq 0$ by (5) and $1 - \delta \geq 0$ by the choice of δ , so
551 we can apply inequalities $w_j \leq w_f$ and $w_i \leq w_f$ from Lemma 4.12(b) and (c). The last step (12)
552 follows from equation (4).

Case 1.2: The chain C is singleton. We assume that $w_i > R \cdot w_e$, otherwise there is no forward
charge from the chain. Then the charge to step t is

$$w_j + w_i - R \cdot w_e \leq w_j + w_i - R \cdot w_j \leq w_i \leq w_f,$$

553 where in the last step we used Lemma 4.12(c).

554 Case 2: $w_j > w_e$. We claim first that j is not expiring in step t , that is $d_j = t + 1$. Indeed, if we
555 had $d_j = t$, then in step $t - 1$ the algorithm would have pending packets e and i , plus packet j
556 (pending or lookahead), that need to be scheduled in slots $t - 1$ and t . Since $w_e < w_i$ (because
557 step $t - 1$ is a chaining step) and $w_e < w_j$ (by the case assumption), packet e could not be in
558 the plan in step $t - 1$ which is a contradiction. Thus $d_j = t + 1$.

559 Recall that e is expiring in step $t - 1$ by Lemma 4.9(b) and both i and j are considered
560 for the plan in step $t - 1$. Moreover, we know that $w_i > w_e$, $w_j > w_e$, $(r_i, d_i) = (t - 1, t)$ (by
561 Lemma 4.9(a)), and $(r_j, d_j) = (t, t + 1)$. We thus use Lemma 4.2 for step $t - 1$ with $a = e$, $b = i$,
562 and $c = j$, to get that $2\alpha \cdot w_e \geq w_i + w_j$.

Case 2.1: The chain C is long. The charge to step t is

$$\begin{aligned}
w_j + (1 - \delta) \cdot w_i - (R - 1 + 2\delta) \cdot w_e & \\
&\leq w_j + (1 - \delta) \cdot w_i - (R - 1 + 2\delta) \cdot \frac{w_i + w_j}{2\alpha} \\
&= \left(1 - \frac{R - 1 + 2\delta}{2\alpha}\right) \cdot w_j + \left(1 - \delta - \frac{R - 1 + 2\delta}{2\alpha}\right) \cdot w_i \\
&\leq \left(1 - \frac{R - 1 + 2\delta}{2\alpha}\right) \cdot w_f + \left(1 - \delta - \frac{R - 1 + 2\delta}{2\alpha}\right) \cdot w_f \quad (13) \\
&= \left(2 - \delta - \frac{R - 1 + 2\delta}{\alpha}\right) \cdot w_f = R \cdot w_f. \quad (14)
\end{aligned}$$

563 To justify inequality (13), we note that $1 - \delta - (R - 1 + 2\delta)/(2\alpha) \geq 0$, by (6), so we can again
564 apply inequalities $w_j \leq w_f$ and $w_i \leq w_f$ from Lemma 4.12(b) and (c). In the last step (14) we
565 used equation (1).

Case 2.2: The chain C is singleton. We assume that $w_i > R \cdot w_e$, otherwise there is no forward charge from the chain. Then the charge to step t is

$$\begin{aligned}
w_j + w_i - R \cdot w_e &\leq w_j + w_i - R \cdot \frac{w_i + w_j}{2\alpha} \\
&= \left(1 - \frac{R}{2\alpha}\right) \cdot (w_i + w_j) \\
&\leq \left(1 - \frac{R}{2\alpha}\right) \cdot (2w_f) \quad (15)
\end{aligned}$$

$$= \left(2 - \frac{R}{\alpha}\right) \cdot w_f < R \cdot w_f. \quad (16)$$

566 Inequality (15) is valid, because $w_i \leq w_f$ and $w_j \leq w_f$, by Lemma 4.12, and $1 - R/(2\alpha) \geq 0$
567 by (7). In step (16) we used (9). \square

568 We now analyze how the forward charge from a chain combines with split charges. First we
569 observe that only the first step from a split-charge pair may receive a forward charge from a
570 chain.

571 **Lemma 4.14.** *If j is charged using a split charge to a pair of steps t and t' (where t' is $t + 1$ or*
572 *$t + 2$), then t' does not receive a forward charge from a chain.*

573 *Proof.* By Lemma 4.4(b) we have $k = f$, which implies that steps t and $t + 1$ are not chaining
574 steps. \square

575 **Lemma 4.15.** *If j is charged using a split charge to a pair of steps t and t' , f' is the packet*
576 *scheduled in t' in ALG , and step t receives a forward charge from a chain C , then the total charge*
577 *to steps t and t' is at most $R \cdot (w_f + w_{f'})$.*

578 *Proof.* First we note that j is expiring in step t by Lemma 4.4(c). Furthermore, i is expiring in
579 step t by Lemma 4.9(a) and f is not expiring in step t by Lemma 4.4(b), so $f \neq i$.

580 We claim $w_j < w_e$. Indeed, if $w_j > w_e$, then in step $t - 1$ the algorithm would have pending
581 packets e and i , plus packet j (pending or lookahead), that need to be scheduled in slots $t - 1$
582 and t . Since $w_e < w_i$ (because step $t - 1$ is a chaining step) and $w_e < w_j$, packet e could not be
583 in the plan in step $t - 1$ which is a contradiction. Therefore $w_j < w_e$.

584 Let p_1, p_2, p_3 be the plan at time t . We split the proof into two cases, both having two
585 subcases, one for long chains and one for singleton chains.

586 Case 1: j is charged using a distant split charge or f' gets a full back charge.

587 We claim that $2\alpha \cdot w_i < w_f + w_{f'}$. Indeed, since i is expiring and pending in step t by
 588 Lemma 4.9(a), we have $w_i \leq w_{p_1}$. As the algorithm scheduled $f = p_2$ by Lemma 4.4(d) we get
 589 $2\alpha \cdot w_{p_1} < w_f + w_{p_3}$. To prove the claim it remains to show $w_{f'} \geq w_{p_3}$.

590 If j is charged using a distant split charge, then by Lemma 4.5 we have $w_g < w_{p_3}$ and in
 591 particular, $g \neq p_3$. Thus p_3 is pending and expiring in step $t + 2$, hence $w_{p_3} \leq w_{f'}$. Otherwise, if
 592 j is charged using a close split charge, then $f' = g$ gets a full back charge. Hence $d_{f'} = t + 2$.
 593 Since also $d_{p_3} = t + 2$ and the algorithm chooses the heaviest such packet, we have $w_{p_3} \leq w_{f'}$.

594 The claim follows, since

$$2\alpha \cdot w_i \leq 2\alpha \cdot w_{p_1} < w_f + w_{p_3} \leq w_f + w_{f'}. \quad (17)$$

Case 1.1: The chain C is long. We upper bound the total charge to steps t and t' by

$$\begin{aligned} w_f + w_{f'} + w_j + (1 - \delta) \cdot w_i - (R - 1 + 2\delta) \cdot w_e \\ \leq w_f + w_{f'} + (2 - R - 2\delta) \cdot w_e + (1 - \delta) \cdot w_i \\ < w_f + w_{f'} + (2 - R - 3\delta) \cdot w_i + w_i \end{aligned} \quad (18)$$

$$= w_f + w_{f'} + w_i \quad (19)$$

$$< w_f + w_{f'} + \frac{w_f + w_{f'}}{2\alpha} \quad (20)$$

$$= R \cdot (w_f + w_{f'}).$$

595 We can use $w_e < w_i$ in (18), because $2 - R - 2\delta \geq 0$ by (5). Equality (19) follows from
 596 $2 - R - 3\delta = 0$ by (4) and inequality (20) from (17). In the last step we use (3).

Case 1.2: The chain C is singleton. We suppose that $w_i > R \cdot w_e$, otherwise there is no forward
 charge from the chain. We upper bound the total charge to steps t and t' by

$$\begin{aligned} w_f + w_{f'} + w_j + w_i - R \cdot w_e &\leq w_f + w_{f'} + (1 - R) \cdot w_e + w_i \\ &< w_f + w_{f'} + w_i \\ &< w_f + w_{f'} + \frac{w_f + w_{f'}}{2\alpha} \\ &= R \cdot (w_f + w_{f'}), \end{aligned} \quad (21)$$

597 where we apply Equation 17 in (21), and we use (3) in the last step.

598 Case 2: j is charged using a close split charge and $f' = g$ does not get a full back charge. We
 599 have (i) $2\alpha \cdot w_{p_1} < w_f + w_g$, or (ii) $2\alpha \cdot (w_{p_1} - w_g) < w_f + w_g$ by the definition of the close split
 600 charge. We suppose that we have (ii), since (i) is stronger than (ii).

601 Since i is expiring and pending in step t by Lemma 4.9(a), we have $w_i \leq w_{p_1}$. Hence
 602 $2\alpha \cdot (w_i - w_g) < w_f + w_g$. This is equivalent to

$$w_i < \frac{w_f + (2\alpha + 1) \cdot w_g}{2\alpha}. \quad (22)$$

Case 2.1: The chain C is long. The total charge to steps t and $t' = t + 1$ is

$$\begin{aligned} w_f + w_j + (1 - \delta) \cdot w_i - (R - 1 + 2\delta) \cdot w_e & \\ & \leq w_f + (2 - R - 2\delta) \cdot w_e + (1 - \delta) \cdot w_i & (23) \\ & < w_f + (2 - R - 3\delta) \cdot w_i + w_i & (23) \\ & = w_f + w_i & (24) \end{aligned}$$

$$< w_f + \frac{w_f + (2\alpha + 1) \cdot w_{f'}}{2\alpha} \quad (25)$$

$$= w_f + w_{f'} + \frac{w_f + w_{f'}}{2\alpha}$$

$$= R \cdot (w_f + w_{f'}),$$

603 We can use $w_e < w_i$ in (23), because $2 - R - 2\delta \geq 0$ by (5). Then we use $2 - R - 3\delta = 0$ by (4)
604 in (24), Equation 22 in (25), and Equation 3 in the last step.

Case 2.2: The chain C is singleton. We again suppose that $w_i > R \cdot w_e$, as otherwise there is no forward charge from the chain. We upper bound the total charge to steps t and $t + 1$ by

$$\begin{aligned} w_f + w_j + w_i - R \cdot w_e & \leq w_f + (1 - R) \cdot w_e + w_i \\ & < w_f + w_i \\ & < w_f + \frac{w_f + (2\alpha + 1) \cdot w_{f'}}{2\alpha} & (26) \\ & = w_f + w_{f'} + \frac{w_f + w_{f'}}{2\alpha} \\ & = R \cdot (w_f + w_{f'}), \end{aligned}$$

605 where we apply (22) in inequality (26), and (3) in the last step. \square

606 We now summarize our analysis of COMPAREWITHBIAS(α). If t is not in a split-charge pair,
607 we show upper bounds on the total charge to step t . For each split-charge pair (t, t') , we show
608 upper bounds on the total charge to both steps t and t' . This is sufficient, since split-charge
609 pairs are pairwise disjoint by Lemma 4.7, thus summing all the bounds gives the result.

610 For each step t , we distinguish three cases according to whether t is in a split-charge pair
611 and whether t is a chaining step. In all cases, let f be the packet scheduled at time t in ALG
612 and let j be the packet scheduled at time t in OPT.

613 Case 1: Step t is not chaining and it is not in a split-charge pair. Then t receives at most one
614 full charge from a packet p such that $w_p \leq w_f$ (by Lemma 4.3 and charging rules) and possibly
615 a forward charge from a chain C ; then Lemma 4.13 shows that the sum of a forward charge from
616 a chain and a full charge is at most $R \cdot w_f$.

617 Case 2: Step t is a chaining step. Then it does not receive a split charge or a full charge, by
618 Lemma 4.10. Lemma 4.11 implies that step t receives a charge of at most $R \cdot w_f$.

619 Case 3: (t, t') is a split-charge pair, i.e., t is the first step of the split-charge pair and $t' = t + 1$,
620 or $t' = t + 2$. Thus j is charged using a split charge. Let f' be the packet scheduled in step t' in
621 ALG.

622 By Lemma 4.14 step t' does not receive a forward charge from a chain. If step t also does
623 not receive a forward charge from a chain, then the total charge to steps t and t' is at most
624 $R \cdot (w_f + w_{f'})$ by Lemma 4.8. Otherwise, step t receives a forward charge from a chain and we
625 apply Lemma 4.15 to show that the total charge to steps t and t' is again at most $R \cdot (w_f + w_{f'})$.

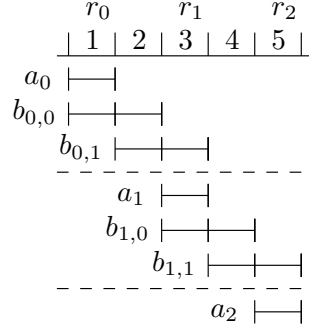


Figure 5: The instance for $\ell = 1$ and $k = n = 2$. Each packet has a row which shows slots between its release time and deadline. Packets from different phases are separated by a dashed line.

5 A Lower Bound for 2-bounded Instances with Lookahead

In this section, we prove that there is no online algorithm for `PacketScheduling` with ℓ -lookahead that has competitive ratio smaller than $R := \frac{1}{2(\ell+1)}(1 + \sqrt{5 + 8\ell + 4\ell^2})$ for any $\ell \geq 0$, even for 2-bounded instances. We note that $R > 1$ for any $\ell \geq 0$ and that R is the positive root of the quadratic equation

$$(\ell + 1)R^2 - R - (\ell + 1) = 0. \quad (27)$$

The idea of our proof is similar to the proof of the lower bound of ϕ for `PacketScheduling` [11, 2, 5] and, indeed, for $\ell = 0$ our lower bound is equal to ϕ . For the case of 1-lookahead we obtain a lower bound of $\frac{1}{4}(1 + \sqrt{17}) \approx 1.281$.

Theorem 5.1. *Let $\ell \geq 0$ be an integer and $R = \frac{1}{2(\ell+1)}(1 + \sqrt{5 + 8\ell + 4\ell^2})$. For each $\varepsilon > 0$, no deterministic online algorithm for `PacketScheduling` with ℓ -lookahead can be $(R - \varepsilon)$ -competitive, even for 2-bounded instances.*

Proof. Fix some online algorithm \mathcal{A} and some $\varepsilon > 0$. We will show that, for some sufficiently large integer n and sufficiently small $\delta > 0$, there is a 2-bounded instance of `PacketScheduling` with ℓ -lookahead, parameterized by n and δ , for which the optimal profit is at least $(R - \varepsilon)$ times the profit of \mathcal{A} .

Our instance will consist of phases $0, \dots, k$, for some $k \leq n$. The number k of phases is determined by the adversary based on the behavior of \mathcal{A} . Each phase (except phase n) will involve $\ell + 2$ packets. The weights of these packets will grow roughly exponentially from one phase to next.

The adversary strategy is as follows. We start with phase 0. Suppose that some phase i , where $0 \leq i < n$, has been reached. Let $r_i = (\ell + 1)i + 1$ be the first slot of phase i . In phase i the adversary releases the following $\ell + 2$ packets:

- A packet a_i with weight w_i , release time r_i and deadline r_i , i.e., a tight packet.
 - Packets $b_{i,j}$ for $j = 0, \dots, \ell$ with weight w_{i+1} , release time $r_i + j$ and deadline $r_i + j + 1$.
- (The weights w_i will be specified later.) Now, if \mathcal{A} schedules an expiring packet in step r_i (a tight packet a_i or $b_{i-1,\ell}$, which may be pending from the previous phase), then the game continues; the adversary will proceed to phase $i + 1$. Otherwise, the algorithm schedules packet $b_{i,0}$, in which case the adversary lets $k = i$ and the game ends. Note that in steps $r_i + 1, \dots, r_i + \ell$ the algorithm may schedule only $b_{i,j}$ (for some j) of weight w_{i+1} . Also, importantly, in step r_i the algorithm cannot yet see whether the packets from phase $i + 1$ will arrive or not.

656 If phase $i = n$ is reached, then $k = n$, and in phase n the adversary releases a single tight
657 packet a_n with weight w_n and release time and deadline r_n . See Figure 5 for an illustration.

658 We calculate the ratio between the weight of packets in an optimal schedule and the weight
659 of packets sent by the algorithm. Let $S_k = \sum_{i=0}^k w_i$. There are two cases: either $k < n$, or
660 $k = n$.

661 Case 1: $k < n$. In all steps r_i for $i < k$ algorithm \mathcal{A} scheduled an expiring packet of weight w_i and
662 in step r_k it scheduled packet $b_{k,0}$ of weight w_{k+1} . In steps $r_i + 1, \dots, r_i + \ell$ for $i < k$ it scheduled
663 packets of weight w_{i+1} . Finally, in phase k the algorithm scheduled $\ell + 1$ packets of weight w_{k+1}
664 (including $b_{k,0}$). Overall, \mathcal{A} scheduled packets of total weight $S_{k-1} + \ell \cdot (S_k - w_0) + (\ell + 1) \cdot w_{k+1} =$
665 $(\ell + 1) \cdot S_{k+1} - w_k - \ell \cdot w_0$.

The adversary schedules packets of weight w_{i+1} in steps $r_i, \dots, r_i + \ell$ for $i < k$ and all
packets from phase k in steps $r_k, \dots, r_k + \ell + 1$. In total, the optimum has a schedule of weight
 $(\ell + 1) \cdot (S_k - w_0) + w_k + (\ell + 1) \cdot w_{k+1} = (\ell + 1) \cdot S_{k+1} + w_k - (\ell + 1) \cdot w_0$. The ratio is

$$R_k = \frac{(\ell + 1) \cdot S_{k+1} + w_k - (\ell + 1) \cdot w_0}{(\ell + 1) \cdot S_{k+1} - w_k - \ell \cdot w_0}.$$

666 Case 2: $k = n$. As before, in all steps r_i for $i < n$ algorithm \mathcal{A} scheduled an expiring packet of
667 weight w_i and in steps $r_i + 1, \dots, r_i + \ell$ for $i < n$ it scheduled a packet of weight w_{i+1} . In the
668 last step r_n it scheduled a packet of weight w_n as there is no other choice. Overall, the total
669 weight of the \mathcal{A} 's schedule is $S_{n-1} + \ell \cdot (S_n - w_0) + w_n = (\ell + 1) \cdot S_n - \ell \cdot w_0$.

The adversary schedules packets of weight w_{i+1} in steps $r_i, \dots, r_i + \ell$ for $i < n$ and a packet
of weight w_n in the last step r_n which adds up to $(\ell + 1) \cdot S_n + w_n - (\ell + 1) \cdot w_0$. The ratio is

$$\widehat{R}_n = \frac{(\ell + 1) \cdot S_n + w_n - (\ell + 1) \cdot w_0}{(\ell + 1) \cdot S_n - \ell \cdot w_0}.$$

670 We normalize the instances so that $w_0 = 1$. It remains to show that we can set the weights
671 so that $R_k \geq R - \varepsilon$ for all $k \geq 0$ and $\widehat{R}_n \geq R - \varepsilon$.

We first define a sequence of weights, parametrized by some parameter $\delta \geq 0$, such that
 $R_k = R$ for all $k \geq 1$. Using $w_k = S_k - S_{k-1}$ for $k \geq 1$ and $w_0 = 1$, the condition $R_k = R$ for
 $k \geq 1$ is rewritten as

$$R = \frac{(\ell + 1) \cdot S_{k+1} + S_k - S_{k-1} - (\ell + 1)}{(\ell + 1) \cdot S_{k+1} - S_k + S_{k-1} - \ell},$$

672 or equivalently as

$$(\ell + 1)(R - 1)S_{k+1} - (R + 1)S_k + (R + 1)S_{k-1} = \ell R - (\ell + 1). \quad (28)$$

673 By (27) we get that $(\ell + 1)(R - 1) = R/(R + 1)$ and similarly $\ell R - (\ell + 1) = -R^2/(R + 1)$.
674 Substituting and multiplying by $R + 1$, we obtain that (28) is equivalent to

$$R \cdot S_{k+1} - (R + 1)^2 S_k + (R + 1)^2 S_{k-1} = -R^2. \quad (29)$$

675 To define our instance, we set $w_0 = 1$ and for $i = 1, 2, \dots$,

$$w_i = (\gamma + 1)\alpha^{i-1}(\alpha - 1) + \delta[\beta^{i-1}(\beta - 1) - \alpha^{i-1}(\alpha - 1)],$$

676 where

$$\alpha = 1 + \frac{1}{R} = \frac{R + 1}{R}, \quad \beta = R + 1, \quad \gamma = R,$$

677 and $\delta > 0$ is a parameter to be chosen later. Summing the geometric sequences in $S_k = \sum_{i=0}^k w_i$,
 678 we obtain that, for $k = 0, \dots, n$,

$$S_k = (\gamma + 1)\alpha^k + \delta(\beta^k - \alpha^k) - \gamma. \quad (30)$$

679 It can be verified that (29) holds and thus, for any choice of δ and any $k \geq 1$, we have $R_k = R$.
 680 In fact, (30) describes a general solution of the linear recurrence (29) that satisfies one initial
 681 condition $S_0 = w_0 = 1$, as α, β are the two roots of $Rx^2 - (R + 1)^2x + (R + 1)^2$ which is the
 682 characteristic polynomial of the recurrence and $S_0 = S_1 = \dots = S_n = -\gamma$ is a particular solution
 683 of the recurrence; furthermore, for $\delta = 0$ (30) gives a particular solution satisfying $S_0 = 1$ and
 684 changing δ does not change S_0 .

We now show that for $\delta = 0$ the solution would satisfy $R_0 = R$. We first calculate w_1 :

$$w_1 = (\gamma + 1)(\alpha - 1) = (R + 1) \cdot \frac{1}{R} = \alpha.$$

By (27) we have $(\ell + 1)\alpha = 1/(R - 1)$. Using this we can calculate R_0 as

$$R_0 = \frac{(\ell + 1)w_1 + 1}{(\ell + 1)w_1} = 1 + \frac{1}{(\ell + 1)\alpha} = R.$$

685 By continuity of the dependence of w_1 and R_0 on δ , for a sufficiently small $\delta > 0$ we have
 686 $R_0 \geq R - \varepsilon$. We fix such a $\delta > 0$.

Since $1 < \alpha < \beta$, for $n \rightarrow \infty$, the dominating term in S_n is $\delta\beta^n$ and w_0 is negligible compared
 to S_n . Thus we obtain

$$\lim_{n \rightarrow \infty} \widehat{R}_n = \lim_{n \rightarrow \infty} \frac{(\ell + 1)S_n + S_n - S_{n-1}}{(\ell + 1)S_n} = \lim_{n \rightarrow \infty} \frac{(\ell + 2)\delta\beta^n - \delta\beta^{n-1}}{(\ell + 1)\delta\beta^n} = \frac{(\ell + 2)\beta - 1}{(\ell + 1)\beta} = R.$$

687 The last equality follows from (27). Actually, this is the equation that defines R as the optimal
 688 ratio for our construction (if β is expressed in terms of R as the root of the characteristic
 689 polynomial). Consequently, for some sufficiently large n , we have that $\widehat{R}_n \geq R - \varepsilon$. Fix this n .

690 Summarizing, we showed that for any $\varepsilon > 0$ the adversary can choose $\delta > 0$ and n , and an
 691 instance with up to n phases, such that for this instance we have $R_0 \geq R - \varepsilon$, $R_k = R$ for all
 692 $k \geq 1$, and $\widehat{R}_n \geq R - \varepsilon$. This implies that the competitive ratio of \mathcal{A} is at least R , completing
 693 the proof. \square

694 References

- 695 [1] Susanne Albers. On the influence of lookahead in competitive paging algorithms. *Algorith-*
 696 *mica*, 18(3):283–305, 1997.
- 697 [2] Nir Andelman, Yishay Mansour, and An Zhu. Competitive queueing policies for QoS
 698 switches. In *Proc. 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'03)*,
 699 pages 761–770, 2003.
- 700 [3] Martin Böhm, Marek Chrobak, Lukasz Jez, Fei Li, Jiri Sgall, and Pavel Vesely. Online packet
 701 scheduling with bounded delay and lookahead. In *Proc. 27th International Symposium on*
 702 *Algorithms and Computation (ISAAC'16)*, pages 21:1–21:13, 2016.
- 703 [4] Francis Y. L. Chin, Marek Chrobak, Stanley P. Y. Fung, Wojciech Jawor, Jiri Sgall, and
 704 Tomáš Tichý. Online competitive algorithms for maximizing weighted throughput of unit
 705 jobs. *J. of Discrete Algorithms*, 4(2):255–276, 2006.

- 706 [5] Francis Y. L. Chin and Stanley P. Y. Fung. Online scheduling with partial job values: Does
707 timesharing or randomization help? *Algorithmica*, 37(3):149–164, 2003.
- 708 [6] Marek Chrobak, Wojciech Jawor, Jiří Sgall, and Tomáš Tichý. Improved online algorithms
709 for buffer management in QoS switches. In *Proc. 12th Annual European Symposium*
710 *(ESA '04)*, pages 204–215, 2004.
- 711 [7] Matthias Englert and Matthias Westermann. Considering suppressed packets improves
712 buffer management in QoS switches. In *Proc. 18th Annual ACM-SIAM Symposium on*
713 *Discrete Algorithms (SODA '07)*, pages 209–218, 2007.
- 714 [8] Matthias Englert and Matthias Westermann. Lower and upper bounds on FIFO buffer
715 management in QoS switches. *Algorithmica*, 53(4):523–548, 2009.
- 716 [9] Michael H. Goldwasser. A survey of buffer management policies for packet switches. *SIGACT*
717 *News*, 41(1):100–128, 2010.
- 718 [10] Edward F. Grove. Online bin packing with lookahead. In *Proc. 6th Annual ACM-SIAM*
719 *Symposium on Discrete Algorithms (SODA '95)*, pages 430–436, 1995.
- 720 [11] Bruce Hajek. On the competitiveness of on-line scheduling of unit-length packets with hard
721 deadlines in slotted time. In *Proc. Conference on Information Sciences and Systems*, pages
722 434–438, 2001.
- 723 [12] Alexander Kesselman, Zvi Lotker, Yishay Mansour, Boaz Patt-Shamir, Baruch Schieber,
724 and Maxim Sviridenko. Buffer overflow management in QoS switches. *SIAM Journal on*
725 *Computing*, 33(3):563–583, 2004.
- 726 [13] Fei Li, Jay Sethuraman, and Clifford Stein. An optimal online algorithm for packet
727 scheduling with agreeable deadlines. In *Proc. 16th Annual ACM-SIAM Symposium on*
728 *Discrete Algorithms (SODA '05)*, pages 801–802, 2005.
- 729 [14] Rajeev Motwani, Vijay Saraswat, and Eric Torng. Online scheduling with lookahead:
730 Multipass assembly lines. *INFORMS J. on Computing*, 10(3):331–340, 1998.