

Randomized Algorithms for Buffer Management with 2-Bounded Delay

Marcin Bienkowski¹, Marek Chrobak², and Lukasz Jeż¹

¹ Institute of Computer Science, University of Wrocław, 50-383 Wrocław, Poland. *

² Department of Computer Science, University of California, Riverside, CA 92521, USA **

Abstract. In the problem of buffer management with bounded delay, packets with weights and deadlines arrive at a network switch over time, and the goal is to send those packets on the outgoing link while maximizing the total weight of the packets that are sent before their deadlines expire. In the 2-bounded delay case, each packet has to be sent either in the step of its release or in the next step. In the deterministic case, the optimal competitive ratio for this case is $\phi \approx 1.618$. In the randomized case, against oblivious adversaries, the optimal competitive ratio is 1.25. The only yet unresolved case is that of randomized algorithms against adaptive adversaries. For this case, we give a complete solution by proving that the optimal competitive ratio is $4/3$. Additionally, we give a lower bound of 1.2 for the 2-uniform case.

1 Introduction

In this paper, we consider the problem of *buffer management with bounded delay*, introduced by Kesselman et al. [9]. This problem models the behavior of a single network switch. We assume that time is slotted and divided into steps. At the beginning of a time step, any number of packets may arrive at a switch and are stored in its *buffer*. A packet has a positive weight and a deadline, which is an integer denoting the last step in which packet may be transmitted. Only one packet can be transmitted in a single step. Packets whose deadline already passed are lost and removed from the buffer. The goal is to maximize the *gain*, defined as the total weight of the transmitted packets.

We note that this problem is equivalent to a scheduling problem, in which packets are jobs of unit length, with given release time, deadline and weight. Release times and deadlines are restricted to integer values. In this setting, the goal is to maximize the total weight of jobs which are completed before their deadlines.

As the process of managing packet queue is inherently a real-time task, we model it as an online problem. This means that the algorithm, deciding which

* Supported by MNiSW grants number N206 001 31/0436, 2006–2008 and N N206 1723 33, 2007–2010.

** Supported by NSF grants OISE-0340752 and CCF-0729071.

packets to transmit, has to base its decision solely on the packets which have already arrived at a switch, without the knowledge of the future.

Competitive Analysis. To measure the performance of an online algorithm, we use a standard notion of the competitive analysis [3], which, roughly speaking, compares the gain of the algorithm to the gain of the optimal solution on the same input sequence. For any algorithm ALG, we denote its gain on input sequence I by $\mathcal{G}_{\text{ALG}}(I)$; we denote the optimal offline algorithm by OPT. We say that a deterministic algorithm ALG is \mathcal{R} -competitive if on any input sequence I , it holds that $\mathcal{G}_{\text{ALG}}(I) \geq \frac{1}{\mathcal{R}} \cdot \mathcal{G}_{\text{OPT}}(I)$.

When analyzing the performance of an online algorithm ALG, we view the process as a game between ALG and an *adversary*. The adversary controls what packets are injected into the buffer and chooses which of them to send. The goal is then to show that the adversary's gain is at most \mathcal{R} times ALG's gain.

If the algorithm is randomized, we consider its expected gain, $\mathbf{E}[\mathcal{G}_{\text{ALG}}(I)]$, where the expectation is taken over all possible random choices made by ALG. However, in the randomized case, the power of the adversary has to be further specified. Following Ben-David et al. [2], we distinguish between an *oblivious* and *adaptive-online* adversary (called adaptive for short). An oblivious adversary has to construct the whole input sequence in advance, not taking into account the random bits used by an algorithm. The expected gain of ALG is compared to the gain of the optimal offline solution on I . An adaptive adversary may decide about the next packets injected into the buffer upon seeing which packets are transmitted by the algorithm. However, it has to provide an answering entity ADV, which creates a solution in parallel to ALG. ADV's solution may not be changed afterwards. We say that ALG is \mathcal{R} -competitive against an adaptive adversary if for any input sequence I created adaptively and any answering algorithm ADV, it holds that $\mathbf{E}[\mathcal{G}_{\text{ALG}}(I)] \geq \frac{1}{\mathcal{R}} \cdot \mathbf{E}[\mathcal{G}_{\text{ADV}}(I)]$. We note that ADV is deterministic, but as ALG is randomized, so is the input sequence I .

In the literature on online algorithms (see e.g. [3]), the definition of the competitive ratio sometimes allows an additive constant, i.e., a deterministic algorithm is \mathcal{R} -competitive if there exists a constant $\alpha \geq 0$ such that for input sequence I , it holds that $\mathcal{G}_{\text{ALG}}(I) \geq \frac{1}{\mathcal{R}} \cdot \mathcal{G}_{\text{OPT}}(I) - \alpha$. An analogous definition applies to randomized case. In this paper, our upper bound has $\alpha = 0$, while our lower bounds hold for any constant α .

Bounded Sequences. A packet *delay* is the number of steps in which the packet may be transmitted. In an *s-bounded sequence* the delay of each packet is at most s , whereas in an *s-uniform sequence* the delay of each packet is exactly s .

1.1 Related Work

The currently best, 1.828-competitive deterministic algorithm for general instances was given by Englert and Westermann [7]. The lower bound of $\phi \approx 1.618$ for the competitive ratio of deterministic algorithms was shown in [1, 5, 8]. The

	deterministic	(rand.) adaptive	(rand.) oblivious
general input upper	1.828 [7]	1.582 [4]	1.582
general input lower	1.618	1.25 1.333	1.25
2-bounded upper	1.618 [9]	1.582 1.333	1.25 [4]
2-bounded lower	1.618 [1, 5, 8]	1.25 1.333	1.25 [5]
2-uniform upper	1.377 [6]	1.377 1.333	1.25
2-uniform lower	1.377 [6]	1.172 1.2	1.172 [4]

Fig. 1. Comparison of known and new results. The results of this paper are written in boldface. The results without citations are implied by other entries of the table.

input sequences which incur this lower bound are 2-bounded. On the other hand, for 2-bounded and 3-bounded instances, deterministic ϕ -competitive algorithms are known (see [9] and [4], respectively). If we restrict the set of inputs to 2-uniform instances, the optimum competitive ratio equals approximately 1.377 [6]. These bounds are summarized in Fig. 1.

The best known randomized solution is the algorithm RMIX by Chin et al. [4]. RMIX achieves the competitive ratio $\frac{e}{e-1} \approx 1.582$, which holds even against an adaptive adversary. The best lower bound of 1.25 for randomized algorithms was given by Chin and Fung [5]. This lower bound holds even against an oblivious adversary and uses 2-bounded instances. A matching upper bound for 2-bounded instances and oblivious adversaries was given by Chin et al. [4]. For 2-uniform instances, currently best known lower bound for competitive ratio against oblivious adversary is approximately 1.172 [4].

1.2 Our Contribution

In our paper we consider 2-bounded and 2-uniform sequences and adaptive adversaries. In addition to purely theoretical interest, the adaptive adversary model is in fact quite reasonable in the network traffic setting. This is because, in reality, the traffic through a switch is not at all independent of the packet scheduling algorithm. For example, lost packets are typically resent, and low throughput in a node can affect the choice of routes for data streams in a network. These phenomena are not captured by the oblivious adversary model.

Prior to this work, the only result on packet scheduling, which involved these adversaries, was the algorithm RMIX by Chin et al. [4]. The lower bounds for this variant are implied by the corresponding lower bounds for oblivious adversaries. In particular, the currently best lower bound for general input sequence was 1.25 [5].

We improve this bound by presenting an adaptively created 2-bounded instance which forces any randomized algorithm to have a competitive ratio at least $4/3$. We present an optimal algorithm RAND, which achieves this ratio for 2-bounded sequences. Obviously, the $4/3$ upper bound applies to 2-uniform instances, as well. We also give a lower bound of 1.2 for 2-uniform instances.

1.3 Preliminaries

For any algorithm A , let \mathcal{B}_A denote the state of its buffer, that is, the set of packets stored in the buffer. In particular, \mathcal{B}_{ADV} denotes the state of the buffer of the adversary ADV. We denote the (expected) gain of algorithm A by \mathcal{G}_A .

We assume that time is divided into steps. In step t , first the adversary injects any number of packets. By (w, d) we denote a packet with weight w and deadline d . For 2-bounded instances, $d \in \{t, t + 1\}$, whereas for 2-uniform ones, $d = t + 1$. Then ADV and the algorithm decide, at the same time, which packet to transmit. We do a fine-grained description of the step (packet disappearances due to deadlines or removal of superfluous packets) in the upper bound section.

2 Lower Bound for 2-Bounded Sequences

First, we present a finite strategy of the adversary, which forces any randomized algorithm ALG to have gain at least $4/3 - \epsilon$ times smaller than the gain of the adversary. As after executing this strategy, buffers of both the algorithm and the adversary are empty, we may repeat it, achieving arbitrarily high gain of the adversary. This and the fact that the ϵ term can be made arbitrarily small imply the lower bound of $4/3$ on the ratio of any randomized algorithm. Without loss of generality, we assume that ALG always sends a packet in a step if its buffer is non-empty.

Input Construction. First, we show how the adversary creates a finite input sequence of length at most n . Our construction preserves the following invariant: at the very beginning of each step t , before the adversary injects new packets, both ALG and ADV have at most one pending packet, $(2^t, t)$. To this end, we assume that before we start both ALG and ADV have packet $(2, 1)$ in their buffers (this packet is simply injected by the adversary in the first step). If at the beginning of a step $\mathcal{B}_{\text{ALG}} = \emptyset$, the sequence is finished. Otherwise, we describe what packets are injected in step t , distinguishing three disjoint cases.

Case 1. $\mathcal{B}_{\text{ALG}} = \{(2^t, t)\}$ and $t < n$. Then the adversary injects $(2^t, t)$ and $(2^{t+1}, t + 1)$.

Case 2. $\mathcal{B}_{\text{ALG}} = \emptyset$. Then no new packet is injected.

Case 3. $\mathcal{B}_{\text{ALG}} = \{(2^t, t)\}$ and $t = n$. Then the adversary injects $(2^t, t)$.

If case i occurs in step t , we call such step an i -step. Since after any 2-step or 3-step, $\mathcal{B}_{\text{ALG}} = \mathcal{B}_{\text{ADV}} = \emptyset$, the sequence ends immediately. It might happen that in a 2-step the adversary has no packet. In this case, neither the algorithm nor the adversary transmits anything in this step; we call such step an *empty 2-step*.

We present the strategy of the adversary in a single step t . In a 2-step, the adversary simply transmits a packet if he has one; in a 3-step, the adversary has one or two packets $(2^t, t)$; he transmits one of them. In a 1-step, both the

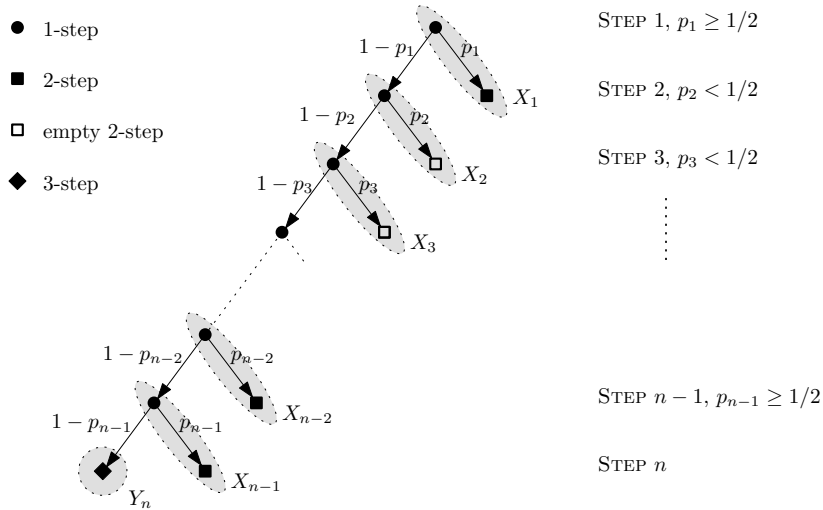


Fig. 2. Tree of the game between ALG and ADV for 2-bounded sequences

algorithm and the adversary have packet $(2^{t+1}, t + 1)$ and at least one copy of packet $(2^t, t)$. In this case, the adversary tries to transmit packet different from the one transmitted by the algorithm. Formally, let p_t denote the probability that ALG transmits the packet with later deadline. If $p_t \geq 1/2$, then ADV transmits the packet with the earlier deadline, otherwise it chooses the packet with the later deadline.

Game Tree. Consider the following thought experiment: assume that n is infinite, i.e., case 3 never occurs and inspect a 1-step t . With probability $1 - p_t$, ALG transmits $(2^t, t)$ and leaves $(2^{t+1}, t + 1)$ in the buffer. In this case the next step is a 1-step. With probability p_t , the algorithm transmits $(2^{t+1}, t + 1)$ and ends this step with empty buffer. In such case, the next step is a 2-step, after which the sequence ends. Thus, we may identify ALG with an infinite sequence of $(p_t)_t$. Note that this observation holds also if the sequence is finite, as the algorithm does not know the value of n until step n .

We view the whole process as a game between the algorithm ALG and the adversary ADV. If we fix the sequence $(p_t)_t$ of ALG and n , the length of the game, this uniquely defines the game tree T_n . An example of such tree is depicted in Fig. 2. The tree represents all the possible outcomes of the random choices made by ALG, i.e. a possible run of ALG is represented by a path starting from the root and ending in a leaf. We say that a node v occurs in the game if it is on such a path. Level t of the tree corresponds to step t ; the root is on level 1.

The probability that a particular state represented by a tree node v actually appears throughout the game is equal to the product of probabilities assigned to edges on the path from the root to v . Let \mathcal{E}_t denote an event of reaching a step t on the leftmost path of the tree, hence $\Pr[\mathcal{E}_t] = \prod_{i=1}^{t-1} (1 - p_i)$. We note

that tree T_n itself is well defined even if $\Pr[\mathcal{E}_n] = 0$, i.e., if the algorithm never reaches the n -th level of the tree.

We split the expected gain of ALG on the sequence into the gains in individual steps (corresponding to appropriate tree nodes). We define $\mathcal{G}_{\text{ALG}}(v)$ as contribution to the expected gain of ALG gathered in the step represented by v . In other words, $\mathcal{G}_{\text{ALG}}(v)$ is the (unconditioned) expected gain of ALG in this step. For any subset $K \subseteq T_n$, we define $\mathcal{G}_{\text{ALG}}(K) = \sum_{v \in K} \mathcal{G}_{\text{ALG}}(v)$. Although this gain is well defined for any subset of vertices K , we consider only subsets which are either single nodes, paths or whole subtrees. We introduce analogous definition for \mathcal{G}_{ADV} . In these terms, $\mathcal{G}_{\text{ALG}}(T_n)$ and $\mathcal{G}_{\text{ADV}}(T_n)$ are expected gains of ALG and ADV, respectively, on the whole input sequence of length n . Thus, the lower bound is implied by the following theorem, proved in the subsection below.

Theorem 1. *For any $\epsilon > 0$ and any probability sequence $(p_t)_t$ of ALG, there exists an integer n , such that $\mathcal{G}_{\text{ADV}}(T_n) \geq (\frac{4}{3} - \epsilon) \cdot \mathcal{G}_{\text{ALG}}(T_n)$.*

Comparing Expected Gains. To prove Theorem 1, we partition T_n into the following parts. Let X_t denote a set consisting of a node corresponding to a 1-step in step t and a 2-step in step $t+1$. Let Y_n be the node corresponding to the only 3-step in tree T_n , see Figure 2. Below, we compute the expected gains on these parts.

Lemma 1. *For any algorithm ALG, the corresponding game tree T_n and any $1 \leq t < n$, we have*

- (i) $\mathcal{G}_{\text{ALG}}(X_t) = (1 + p_t) \cdot 2^t \cdot \Pr[\mathcal{E}_t]$,
- (ii) $\mathcal{G}_{\text{ADV}}(X_t) = \max\{2, 1 + 2 \cdot p_t\} \cdot 2^t \cdot \Pr[\mathcal{E}_t]$,
- (iii) $\mathcal{G}_{\text{ADV}}(Y_n) = \mathcal{G}_{\text{ALG}}(Y_n) = 2^n \cdot \Pr[\mathcal{E}_n]$.

Proof. Fix any set X_t . The probability of reaching its 1-step is $\Pr[\mathcal{E}_t]$. In this 1-step, both ALG and ADV have $(2^t, t)$ and $(2^{t+1}, t+1)$ in their buffers.

The algorithm transmits the latter with probability p_t and, by the construction, does not have any packet in the 2-step. Thus (i) follows, as $\mathcal{G}_{\text{ALG}}(X_t) = \Pr[\mathcal{E}_t] \cdot (p_t \cdot 2^{t+1} + (1 - p_t) \cdot 2^t)$.

As for (ii), if $p_t < 1/2$, then ADV transmits packet $(2^{t+1}, t+1)$ and the subsequent 2-step is empty. If $p_t \geq 1/2$, then ADV transmits packet $(2^t, t)$ and is left with $(2^{t+1}, t+1)$ in the buffer. This packet is then transmitted in the next 2-step. Therefore, the gain of the adversary is $\Pr[\mathcal{E}_t] \cdot 2^{t+1}$ if $p_t < 1/2$ and $\Pr[\mathcal{E}_t] \cdot (2^t + p_t \cdot 2^{t+1})$ if $p_t \geq 1/2$. Hence, (ii) holds.

As in the only 3-step of Y_n , both ALG and ADV transmit $(2^n, n)$, property (iii) follows. \square

The following lemmas imply that the ratio of the expected gains of ALG and ADV on T_n is $4/3$ if we neglect either the gain in Y_n or some constant gain.

Lemma 2. *For any integer n , $\mathcal{G}_{\text{ADV}}(T_n) \geq \frac{4}{3} \cdot \mathcal{G}_{\text{ALG}}(T_n) - \frac{1}{3} \cdot \mathcal{G}_{\text{ALG}}(Y_n)$.*

Proof. From Lemma 1, parts (i) and (ii), and using inequality $\max\{2, 1+2 \cdot p_t\} \geq \frac{4}{3}(1+p_t)$, we get $\mathcal{G}_{\text{ADV}}(X_t) \geq \frac{4}{3} \cdot \mathcal{G}_{\text{ALG}}(X_t)$. The lemma follows immediately by noting that $T_n = \uplus_{t=1}^{n-1} X_t \uplus Y_n$ and applying Lemma 1, part (iii). \square

Lemma 3. *For any integer n , $\mathcal{G}_{\text{ADV}}(T_n) \geq \frac{4}{3} \cdot \mathcal{G}_{\text{ALG}}(T_n) - \frac{2}{3}$.*

Proof. We implicitly exploit the idea that if the gain on Y_n is large, then the probabilities p_t are small (at least on average), implying that the ratio of gains of ADV and ALG on X_t is in fact higher than $4/3$.

Formally, let T_n^t be the subtree of T_n rooted at a node corresponding to a 1-step t . We prove, by a backward induction on t , that for each $1 \leq t \leq n$,

$$3 \cdot \mathcal{G}_{\text{ADV}}(T_n^t) \geq 4 \cdot \mathcal{G}_{\text{ALG}}(T_n^t) - 2^t \cdot \Pr[\mathcal{E}_t] . \quad (1)$$

Since $\Pr[\mathcal{E}_1] = 1$, this inequality with $t = 1$ immediately implies the lemma.

As $T_n^n = Y_n$, (1) holds for $t = n$. Assuming (1) holds for $t + 1$, we prove it holds for t as well.

$$\begin{aligned} & 3 \cdot \mathcal{G}_{\text{ADV}}(T_n^t) \\ &= 3 \cdot \mathcal{G}_{\text{ADV}}(X_t) + 3 \cdot \mathcal{G}_{\text{ADV}}(T_n^{t+1}) \\ &\geq 3 \cdot \max\{2, 1 + 2 \cdot p_t\} \cdot 2^t \cdot \Pr[\mathcal{E}_t] + 4 \cdot \mathcal{G}_{\text{ALG}}(T_n^{t+1}) - 2^{t+1} \cdot \Pr[\mathcal{E}_{t+1}] \\ &= 4 \cdot \mathcal{G}_{\text{ALG}}(T_n^{t+1}) + 2^t \cdot \Pr[\mathcal{E}_t] \cdot (3 \cdot \max\{2, 1 + 2 \cdot p_t\} - 2 \cdot (1 - p_t)) \\ &\geq 4 \cdot \mathcal{G}_{\text{ALG}}(T_n^{t+1}) + 2^t \cdot \Pr[\mathcal{E}_t] \cdot (4 \cdot (1 + p_t) - 1) \\ &= 4 \cdot \mathcal{G}_{\text{ALG}}(T_n^{t+1}) + 4 \cdot \mathcal{G}_{\text{ALG}}(X_t) - 2^t \cdot \Pr[\mathcal{E}_t] \\ &= 4 \cdot \mathcal{G}_{\text{ALG}}(T_n^t) - 2^t \cdot \Pr[\mathcal{E}_t] , \end{aligned}$$

where the second inequality follows as $3 \cdot \max\{2, 1 + 2 \cdot p_t\} \geq 5 + 2 \cdot p_t$. \square

Proof (of Theorem 1). First, we compare the expected gain of ALG if the adversary cuts the sequence after n steps to the case where the sequence is cut after $n + 1$ steps. In the former case, if \mathcal{E}_n occurs, step n is a 3-step, in which ALG transmits packet $(2^n, n)$. In the latter case, if \mathcal{E}_n occurs, step n is a 1-step, in which ALG has packets $(2^n, n)$ and $(2^{n+1}, n + 1)$ and transmits one of them. Hence $\mathcal{G}_{\text{ALG}}(X_n) \geq \mathcal{G}_{\text{ALG}}(Y_n)$, which implies

$$\mathcal{G}_{\text{ALG}}(T_{n+1}) \geq \mathcal{G}_{\text{ALG}}(T_n) + \mathcal{G}_{\text{ALG}}(Y_{n+1}) . \quad (2)$$

Fix any $\epsilon > 0$. We consider a non-decreasing sequence $(\mathcal{G}_{\text{ALG}}(T_n))_n$ and distinguish two cases, depending on whether $\{\mathcal{G}_{\text{ALG}}(T_n)\}_n$ converges or not.

If $\lim_{n \rightarrow \infty} \mathcal{G}_{\text{ALG}}(T_n) = \infty$, choose n such that $\mathcal{G}_{\text{ALG}}(T_n) \geq \frac{2}{3\epsilon}$. Lemma 3 implies $\mathcal{G}_{\text{ADV}}(T_n) \geq (\frac{4}{3} - \epsilon) \cdot \mathcal{G}_{\text{ALG}}(T_n)$. Note that in this case, to obtain a high gain, the adversary needs not repeat this strategy; it may simply choose sufficiently large n .

Otherwise, let $g = \lim_{n \rightarrow \infty} \mathcal{G}_{\text{ALG}}(T_n)$. Without loss of generality, we may assume that $\epsilon \leq 1/3$. In this case, for a sufficiently large n , $\mathcal{G}_{\text{ALG}}(T_n) \geq g - \frac{3}{2} \cdot \epsilon \cdot g \geq g/2$. By (2), it holds that $\mathcal{G}_{\text{ALG}}(Y_{n+1}) \leq \mathcal{G}_{\text{ALG}}(T_{n+1}) - \mathcal{G}_{\text{ALG}}(T_n) \leq$

$g - (g - \frac{3}{2} \cdot \epsilon \cdot g) = \frac{3}{2} \cdot \epsilon \cdot g \leq 3 \cdot \epsilon \cdot \mathcal{G}_{\text{ALG}}(T_n) \leq 3 \cdot \epsilon \cdot \mathcal{G}_{\text{ALG}}(T_{n+1})$. Hence, by Lemma 2, $\mathcal{G}_{\text{ADV}}(T_{n+1}) \geq (\frac{4}{3} - \epsilon) \cdot \mathcal{G}_{\text{ALG}}(T_{n+1})$. If this case occurs, the adversary has to repeat the process, to ensure that the overall gain is arbitrarily high. \square

3 Lower Bound for 2-Uniform Sequences

In this section we adapt the previous lower bound construction, so that it holds for 2-uniform sequences as well. The strategy of the adversary has to be changed, because injecting packets with deadline t in step t is no longer possible. Adapting the adversary's strategy to the 2-uniform setting results in a weaker lower bound of $6/5$ on the competitive ratio.

Let us start with an informal description of the changes to the adversary's strategy in reference to the game tree T_n . Suppose that at some 1-step t of the game tree in Fig. 2, the algorithm uses $p_t < \frac{1}{2}$ as the probability of transmitting $(2^{t+1}, t+1)$. Consequently, the adversary transmits $(2^{t+1}, t+1)$. In the following 1-step $t+1$, the adversary is unable to inject $(2^{t+1}, t+1)$. Instead —before each 1-step in the tree— we add a new kind of step, a 0-step, obtaining a tree depicted in Fig. 3. In a 0-step the adversary injects two copies of the packet from the algorithm's buffer, ensuring both players transmit the same packet and are left with essentially the same buffers afterwards. Thus, in every 1-, 2-, or 3-step, the adversary can behave as previously. As each 1-step and the sole 3-step is preceded by a 0-step, we treat each 0-step and the step following it as a single entity.

Input Construction. For sake of similarity with the previous lower bound, we number the steps differently. The steps' numbers now form the sequence: $\frac{1}{2}, 1, 1\frac{1}{2}, 2, 2\frac{1}{2}, 3, \dots$. A packet injected in step t has deadline $t + \frac{1}{2}$. Our construction preserves the following invariant: at the very beginning of each step t , before the adversary injects new packets: (a) if t is integer, both ALG and ADV have at least one copy of a packet $(2^t, t)$ in their buffers, and (b) if t is not integer, both ALG and ADV have at most one copy of a packet $(2^{t+1/2}, t)$ in their buffers, and no packet of other weight.

If $\mathcal{B}_{\text{ALG}} = \emptyset$ at the very beginning of some step $t \geq 1$, the sequence is finished. We describe in detail what packets are injected at step t , distinguishing four disjoint cases.

Case 0. $t < n$ is not an integer and $\mathcal{B}_{\text{ALG}} = \{(2^{t+1/2}, t)\}$, or $t = \frac{1}{2}$. Then the adversary injects 2 copies of $(2^{t+1/2}, t + \frac{1}{2})$.

Case 1. $t < n$ is an integer (in this case the algorithm has at least one copy of $(2^t, t)$ in its buffer). Then the adversary injects $(2^{t+1}, t + \frac{1}{2})$.

Case 2. $t > 1$ is not integer and $\mathcal{B}_{\text{ALG}} = \emptyset$. Then the adversary injects nothing.

Case 3. $t = n$. Then the adversary injects nothing.

If case i occurs in step t , then we call such a step an i -step. We present the strategy of the adversary in a single step t . In a 2-step, the adversary transmits

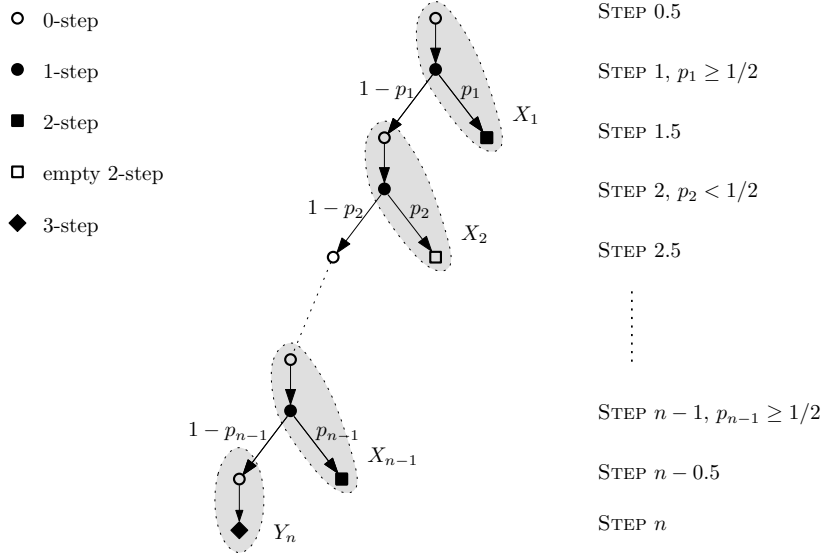


Fig. 3. Tree of the game between ALG and ADV for 2-uniform sequences

a packet if it has one (by the invariant, this packet is $(2^{t+1/2}, t)$); if it has no packet, then we call this 2-step empty. In a 3-step, the adversary transmits the packet $(2^t, t)$, which it has by the invariant. Note that such steps end the sequence. In a 0-step, both the adversary and the algorithm have two copies of $(2^{t+1/2}, t + \frac{1}{2})$. Each of them may also have the packet $(2^{t+1/2}, t)$. Both can transmit any of these packets and are left with at least one copy of $(2^{t+1/2}, t + \frac{1}{2})$. In a 1-step, both the algorithm and the adversary have packet $(2^{t+1}, t + \frac{1}{2})$ and at least one copy of packet $(2^t, t)$. In this case, the adversary tries to transmit packet different from the one transmitted by the algorithm. Formally, let p_t be the probability that ALG transmits the packet with later deadline. If $p_t \geq 1/2$, ADV transmits the packet with earlier deadline, otherwise it chooses the packet with later deadline.

Again, to obtain the lower bound, it suffices to prove the following theorem.

Theorem 2. *For any $\epsilon > 0$ and any probability sequence $(p_t)_t$ of ALG, there exists an integer n , such that $\mathcal{G}_{\text{ADV}}(T_n) \geq (\frac{6}{5} - \epsilon) \cdot \mathcal{G}_{\text{ALG}}(T_n)$.*

Comparing Expected Gains. Let us now define analogues of \mathcal{E}_t , X_t and Y_t from the previous section. By \mathcal{E}_t , for integer t , we denote the event of reaching the step t . Therefore, $\Pr[\mathcal{E}_t] = \prod_{i=1}^{t-1} (1 - p_i)$. Let X_t denote the set consisting of the node corresponding to 0-step in step $t - \frac{1}{2}$, 1-step in step t and 2-step in step $t + \frac{1}{2}$. Let Y_n denote the set consisting of two nodes corresponding to the only 3-step in tree T_n and the 0-step preceding it.

We remark that the only difference between the sets X_t and Y_n defined now and in the previous section are 0-steps. Thus, the gains of both ALG and ADV on X_t increase by $\Pr[\mathcal{E}_t] \cdot 2^t$. The same holds for Y_n . Therefore, the following result holds.

Lemma 4. *For any algorithm ALG, the corresponding game tree T_n and any $1 \leq t < n$,*

- (i) $\mathcal{G}_{\text{ALG}}(X_t) = (2 + p_t) \cdot 2^t \cdot \Pr[\mathcal{E}_t]$
- (ii) $\mathcal{G}_{\text{ADV}}(X_t) = \max\{3, 2 + 2p_t\} \cdot 2^t \cdot \Pr[\mathcal{E}_t]$.
- (iii) $\mathcal{G}_{\text{ALG}}(Y_n) = \mathcal{G}_{\text{ADV}}(Y_n) = 2^{n+1} \cdot \Pr[\mathcal{E}_n]$

The following analogues of Lemmas 2 and 3 are proved in the full version of the paper. Together, these lemmas yield Theorem 2. Its proof, a straightforward modification of proof of Theorem 1, is left out.

Lemma 5. *For any integer n , $\mathcal{G}_{\text{ADV}}(T_n) \geq \frac{6}{5} \cdot \mathcal{G}_{\text{ALG}}(T_n) - \frac{1}{5} \cdot \mathcal{G}_{\text{ALG}}(Y_n)$.*

Lemma 6. *For any integer n , $\mathcal{G}_{\text{ADV}}(T_n) \geq \frac{6}{5} \cdot \mathcal{G}_{\text{ALG}}(T_n) - \frac{4}{5}$.*

4 Upper Bound

In this section, we present our memoryless algorithm RAND, whose competitive ratio against an adaptive adversary is $4/3$.

Without loss of generality, we assume that in each step t the adversary injects at most one packet with deadline t and at most two packets with deadline $t + 1$, as any reasonable algorithm will drop superfluous packets with smaller weights. In fact, we assume that such three packets are injected in each step, as the adversary may use 0-weight packets.

Algorithm RAND. We describe the algorithm's behavior in step t . We can assume that at the beginning of the step the algorithm has exactly one packet and its deadline is t . (At the very beginning, we can assume it is a dummy packet of weight 0.) We divide the step into two stages:

1. The adversary adds a packet with deadline t . RAND drops the lighter of its packets with deadline t (breaking ties arbitrarily). The remaining one is denoted $\mathbf{a} = (a, t)$.
2. The adversary adds two packets $\mathbf{c} = (c, t + 1)$ and $\mathbf{d} = (d, t + 1)$, where $c \leq d$. RAND transmits \mathbf{a} with probability $\min\{a/d, 1\}$ and \mathbf{d} with the remaining probability.

Finally, at the end of step t , RAND removes expired packets (all packets with deadline t) and superfluous packets (all but the most valuable packet with deadline $t + 1$).

When we look at the definition of RAND, we observe that \mathbf{c} , the lighter of two packets with deadline $t + 1$ is never transmitted by RAND in step t . Below, we prove that, without loss of generality, \mathbf{c} is also not transmitted in step t by ADV. Recall that ADV is a deterministic online algorithm, the answering part of the adversary.

Lemma 7. *For any online (deterministic) algorithm ALG, there is an online (deterministic) algorithm $\overline{\text{ALG}}$, with the following properties:*

- (i) *the gain of $\overline{\text{ALG}}$ on every sequence is at least the gain of ALG,*
- (ii) *if, at step t , $\mathcal{B}_{\overline{\text{ALG}}}$ contains $\mathbf{x} = (x, t + 1)$ and $\mathbf{y} = (y, t + 1)$ where $x > y$, then \mathbf{y} is not transmitted in step t .*

Due to space limitations, the proof can be found in the full version of the paper. By the lemma above, the adversary may as well inject $\mathbf{d} = (d, t + 1)$ at step t and $\mathbf{c} = (c, t + 1)$ at step $t + 1$, rather than injecting them both at step t . Such a change does not influence the behavior of RAND and, by Lemma 7, does not hinder the performance of ADV. We obtain the following corollary.

Corollary 1. *Without loss of generality, at each step t the adversary injects exactly one packet with deadline t and exactly one packet with deadline $t + 1$.*

4.1 Computing Competitive Ratio.

We introduce a potential function Φ . It is well-defined at the beginning and at the end of any stage, i.e., when $|\mathcal{B}_{\text{ADV}}| = |\mathcal{B}_{\text{RAND}}| = 1$. Let *rand* and *adv* be the weights of the packets in the buffers of RAND and ADV, respectively. Then

$$\Phi = \max\{\text{adv} - \text{rand}, 0\} .$$

Let Φ_t be the potential at the very beginning of step $t + 1$. We prove the following lemma.

Lemma 8. *For any step t , $\mathbf{E}[\Phi_t - \Phi_{t-1}] + \mathcal{G}_{\text{ADV}}(t) \leq \frac{4}{3} \cdot \mathbf{E}[\mathcal{G}_{\text{RAND}}(t)]$.*

Proof. We show that for any stage of any step t , it holds that $\mathbf{E}[\Delta\Phi] + \mathcal{G}_{\text{ADV}} \leq \frac{4}{3} \cdot \mathbf{E}[\mathcal{G}_{\text{RAND}}]$, where $\Delta\Phi$ is the change of the potential in this stage.

In the first stage, the adversary injects a packet with deadline t . Since the algorithm and the adversary are not transmitting anything, $\mathcal{G}_{\text{RAND}} = \mathcal{G}_{\text{ADV}} = 0$. Without loss of generality, we assume, that both RAND and ADV drop their lighter packet at this moment. The potential cannot increase by such an action.

Assume that at the beginning of the second stage, $\mathcal{B}_{\text{RAND}} = \{\mathbf{a}\}$, $\mathcal{B}_{\text{ADV}} = \{\mathbf{b}\}$, and the adversary injects a packet \mathbf{d} , where $\mathbf{a} = (a, t)$, $\mathbf{b} = (b, t)$, and $\mathbf{d} = (d, t + 1)$. We consider two cases.

- (i) $a \geq d$. In this case, RAND transmits \mathbf{a} , gaining a , and ADV transmits \mathbf{b} or \mathbf{d} , and thus $\mathcal{G}_{\text{ADV}} \leq \max\{b, d\}$. Additionally after this step $\mathcal{B}_{\text{RAND}} = \{\mathbf{d}\}$ and $\mathcal{B}_{\text{ADV}} \subseteq \{\mathbf{d}\}$, hence $\Phi_t = 0$. Therefore, it holds that

$$\begin{aligned} \mathcal{G}_{\text{ADV}} + \Delta\Phi &\leq \max\{b, d\} - \max\{b - a, 0\} \\ &= \max\{b, d\} - \max\{b, a\} + a \\ &\leq a \\ &\leq (4/3) \cdot a . \end{aligned}$$

- (ii) $a < d$. In this case, RAND transmits \mathbf{a} with probability a/d and \mathbf{d} with the remaining probability. Thus,

$$\mathbf{E}[\mathcal{G}_{\text{RAND}}] = \frac{a}{d} \cdot a + \left(1 - \frac{a}{d}\right) \cdot d = \frac{(a - d/2)^2 + \frac{3}{4} \cdot d^2}{d} \geq \frac{3}{4} \cdot d$$

It remains to prove that $\mathcal{G}_{\text{ADV}} + \mathbf{E}[\Delta\Phi] \leq d$. We consider two cases.

- (a) If the adversary transmits \mathbf{d} , then $\mathcal{G}_{\text{ADV}} = d$ and $\Phi_t = 0$. Therefore, $\mathcal{G}_{\text{ADV}} + \Delta\Phi \leq \mathcal{G}_{\text{ADV}} = d$.
- (b) If the adversary transmits \mathbf{b} , then $\mathcal{G}_{\text{ADV}} = b$. In this case, with probability a/d at the end of this step $\mathcal{B}_{\text{RAND}} = \mathcal{B}_{\text{ADV}} = \{\mathbf{d}\}$ and $\Phi_t = 0$. With the remaining probability $\mathcal{B}_{\text{RAND}} = \emptyset$ and hence $\Phi_t = d$. Thus, $\mathbf{E}[\Phi_t] = (1 - a/d) \cdot d = d - a$. Summing up, we obtain $\mathcal{G}_{\text{ADV}} + \mathbf{E}[\Delta\Phi] = b + (d - a) - \max\{b - a, 0\} \leq d$.

□

Finally, as $\Phi_0 = 0$ and Φ is always non-negative, summing the inequality yielded by Lemma 8 over all steps from the input sequence, we obtain the following bound.

Theorem 3. RAND is $\frac{4}{3}$ -competitive against an adaptive-online adversary.

References

1. N. Andelman, Y. Mansour, and A. Zhu. Competitive queueing policies for QoS switches. In *Proc. of the 14th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 761–770, 2003.
2. S. Ben-David, A. Borodin, R. M. Karp, G. Tardos, and A. Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11:2–14, 1994. Also appeared in *Proc. of the 22nd STOC*, pages 379–386, 1990.
3. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
4. 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. *Journal of Discrete Algorithms*, 4:255–276, 2006.
5. F. Y. L. Chin and S. P. Y. Fung. Online scheduling for partial job values: Does timesharing or randomization help? *Algorithmica*, 37:149–164, 2003.
6. M. Chrobak, W. Jawor, J. Sgall, and T. Tichý. Improved online algorithms for buffer management in QoS switches. *ACM Transactions on Algorithms*, 3(4):50, 2007. Also appeared in *Proc. of the 12th ESA*, pages 204–215, 2004.
7. M. Englert and M. Westermann. Considering suppressed packets improves buffer management in QoS switches. In *Proc. of the 18th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 209–218, 2007.
8. B. Hajek. On the competitiveness of online scheduling of unit-length packets with hard deadlines in slotted time. In *Conference in Information Sciences and Systems*, pages 434–438, 2001.
9. A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko. Buffer overflow management in QoS switches. *SIAM Journal on Computing*, 33(3):563–583, 2004. Also appeared in *Proc. of the 33rd STOC*, pages 520–529, 2001.

A Appendix

Proof (of Lemma 5). By Lemma 4, $\mathcal{G}_{\text{ADV}}(X_t) \geq \frac{6}{5} \cdot \mathcal{G}_{\text{ALG}}(X_t)$. The lemma follows immediately by noting that $T_n = \uplus_{t=1}^{n-1} X_t \uplus Y_n$ and applying Lemma 4. \square

Proof (of Lemma 6). Again, let T_n^t be the subtree of T_n rooted at the node corresponding to 0-step $t - \frac{1}{2}$. We prove, by a backward induction on t , that for each $1 \leq t \leq n$,

$$5 \cdot \mathcal{G}_{\text{ADV}}(T_n^t) \geq 6 \cdot \mathcal{G}_{\text{ALG}}(T_n^t) - 2^{t+1} \cdot \Pr[\mathcal{E}_t] . \quad (3)$$

Since $\Pr[\mathcal{E}_1] = 1$, for $t = 1$ this inequality immediately implies the lemma.

As $T_n^n = Y_n$, (3) holds for $t = n$. Assuming (3) holds for $t + 1$, we prove it holds for t as well.

$$\begin{aligned} & 5 \cdot \mathcal{G}_{\text{ADV}}(T_n^t) \\ &= 5 \cdot \mathcal{G}_{\text{ADV}}(X_t) + 5 \cdot \mathcal{G}_{\text{ADV}}(T_n^{t+1}) \\ &\geq 5 \cdot \max\{3, 2 + 2 \cdot p_t\} \cdot 2^t \cdot \Pr[\mathcal{E}_t] + 6 \cdot \mathcal{G}_{\text{ALG}}(T_n^{t+1}) - 2^{t+2} \cdot \Pr[\mathcal{E}_{t+1}] \\ &= 6 \cdot \mathcal{G}_{\text{ALG}}(T_n^{t+1}) + 2^t \cdot \Pr[\mathcal{E}_t] \cdot (5 \cdot \max\{3, 2 + 2 \cdot p_t\} - 4 \cdot (1 - p_t)) \\ &\geq 6 \cdot \mathcal{G}_{\text{ALG}}(T_n^{t+1}) + 2^t \cdot \Pr[\mathcal{E}_t] \cdot (6 \cdot (2 + p_t) - 2) \\ &= 6 \cdot \mathcal{G}_{\text{ALG}}(T_n^{t+1}) + 6 \cdot \mathcal{G}_{\text{ALG}}(X_t) - 2^{t+1} \cdot \Pr[\mathcal{E}_t] \\ &= 6 \cdot \mathcal{G}_{\text{ALG}}(T_n^t) - 2^{t+1} \cdot \Pr[\mathcal{E}_t] , \end{aligned}$$

where the second inequality follows since $5 \cdot \max\{3, 2 + 2 \cdot p_t\} \geq 14 + 2 \cdot p_t$. \square

Proof (of Lemma 7). We transform ALG into $\overline{\text{ALG}}$ iteratively, i.e. we take the minimum t_0 such that ALG first violates property (ii) in step t_0 and we transform it into an algorithm ALG' with gain no smaller than that of ALG, which satisfies property (ii) up to step t_0 , possibly violating it in further steps.

Let t_0 be the first step in which property (ii) is violated. Let $y = (y, t_0 + 1)$ be the packet transmitted by ALG and $x = (x, t_0 + 1)$ be the heaviest packet with deadline $t_0 + 1$. Then ALG' transmits the same packets as ALG up to step $t_0 - 1$, but in step t_0 it transmits x , and in the remaining steps tries to transmit the same packets as ALG. It is impossible only in step $t_0 + 1$ and only if ALG transmits x in that step. In this case ALG' transmits y in step $t_0 + 1$. Clearly, the gain of ALG' is at least as large as the gain of ALG. \square