Geometric Aspects of Online Packet Buffering: An Optimal Randomized Algorithm for Two Buffers*

Marcin Bienkowski[1] and Aleksander Mądry[1,2]

[1] Institute of Computer Science, University of Wroclaw, Poland
[2] CSAIL, MIT, Cambridge, MA, USA

Abstract. We study packet buffering, a basic problem occurring in network switches. We construct an optimal 16/13-competitive randomized online algorithm PB for the case of two input buffers of arbitrary sizes. Our proof is based on geometrical transformations which allow to identify the set of sequences incurring extremal competitive ratios. Later we may analyze the performance of PB on these sequences only.

Key words: online algorithms, network problems, packet buffering.

1 Introduction

Nowadays, the performance of network backbones depends on the speed, with which network devices can switch data packets arriving at the input ports to the appropriate output ports. Since the traffic is usually bursty, the rate of arriving packets might be much higher than the rate with which the device can transmit them, and in result packets might get lost. This motivates the use of buffers attached to the input ports; these buffers can accumulate incoming packets and store them for later transmission. The capacity of buffers — although usually large — is limited, which makes buffer management techniques crucial for minimizing the data loss.

We study a basic problem in this context. We consider a network device which has m input ports and one output port. Each input port has an attached buffer which can store up to B packets; we assume that all packets are of unit size. Time is slotted into time steps. At any time step, any number of packets may arrive at the input ports and they are appended to the appropriate buffers. If a buffer cannot accommodate all the packets, the excess is lost. At any time step, the device can transmit one packet from one buffer; the buffer managing algorithm has to choose which buffer to send from. The scenario described above is typical for input-queued switches or routers. Additionally, this model, in which packets are equally important, is typical for current IP networks.

In our setting no information about the future is available to the algorithm. In particular, we make no probabilistic assumptions about the input. For analyzing the efficiency of our algorithms we use competitive analysis [13], and — on any input sequence — compare the throughput (the number of packets transmitted) of our algorithm and the optimal offline schedule. For any algorithm A and any sequence of packets arrival τ, we denote the throughput of A on τ by $T_{\mathrm{A}}(\tau)$. We call a deterministic algorithm Alg c-competitive if for all sequences τ, it holds that $c \cdot T_{\mathrm{ALG}}(\tau) \geq T_{\mathrm{OPT}}(\tau)$, where Opt denotes the optimal *offline* algorithm. Number c is called a *competitive ratio* of the algorithm Alg. If Alg is a randomized algorithm, then in the definition above we replace $T_{\mathrm{ALG}}(\tau)$ with its expected value.

* Full version. Extended abstract appeared in the proceedings of LATIN 2008. Research supported by MNiSW grant number N206 001 31/0436, 2006–2008, MNiSW grant number N N206 1723 33, 2007–2010, and by an Akamai Presidential Fellowship.

Previous Results. There are several results for the basic model described above. As the optimal competitive ratios can differ depending on the values B and m, the results address particular classes of these values.

First, we consider deterministic algorithms. The general upper bound holding for all values of B and m was given by Azar and Richter [2]. They proved that any deterministic *work-conserving* (i.e. serving a non-empty queue) algorithm is 2-competitive. They showed that for $B = 1$ no deterministic strategy can be better than $(2 - \frac{1}{m})$-competitive and presented a lower bound on the competitive ratio of $1.366 - \Theta(\frac{1}{m})$ which holds for any fixed B. Albers and Schmidt [1] improved that bound showing that for any fixed B and for large m the lower bound can be arbitrarily close to $e/(e-1) \approx 1.582$. They also showed an algorithm *Semi-Greedy* which is 1.944-competitive for $B \geq 2$ [12]. For $B = 2$ this algorithm is optimal and 1.857-competitive; for $B \to \infty$ the algorithm is 1.889-competitive. For $m = 2$ and $B \to \infty$, Schmidt [11] demonstrated a lower bound of $16/13 = 1.231$ and proved that a greedy algorithm achieves a ratio of $9/7 \approx 1.286$.

Randomized algorithms were also considered: Schmidt [11] showed a 3/2-competitive *Random Permutation* algorithm; the competitive ratio holds for any values of B and m. For the lower bound, define $h(n, k)$ as

$$h(n,k) = \frac{k+n}{k+1+\frac{(n-1)^{k+1}}{n^k}} \quad \text{and} \quad h(n) = \min_{k \in \mathbb{N}} h(n,k) \ . \tag{1}$$

A lower bound claimed by Albers and Schmidt in [1], whose proof can be found in [12], states that for any value of B the competitive ratio of any randomized algorithm is at least $h(m)$. This value is equal to $16/13$ for $m = 2$ and approaches 1.466 for $m \to \infty$.

Our Contribution and Paper Outline. In this paper, we present a randomized online algorithm which — for the case of $m = 2$ buffers with arbitrary buffer size B — achieves the optimal competitive ratio of $16/13 \approx 1.231$.

Most papers on packet buffering concentrate around developing a smart algorithm and then comparing its behavior to the optimal one. Hence, the optimal algorithm is considered only in the analysis. We employ a different approach. In each step, we trace the set of possible states of the algorithm, which would so far serve the sequence in optimal manner. Then, by keeping the state of our online algorithm as close to the center of this set as possible, we ensure that it performs well compared to the optimal solution. This technique bears some similarities to the well-known *work-function* technique, used for constructing many optimal or almost optimal online algorithms (for example, for k-server [7] or page migration [3]).

Initially, we construct and analyze a deterministic algorithm PBF in a setting that allows PBF to have fractional number of packets in its buffers. We note that the lower bound on the competitive ratio of $h(2) = 16/13$ holds also for such model. The proof of PBF optimality consists of two parts. In Sect. 3.2, we show how the hardest sequences for PBF look like. We show that these input sequences (we call them *regular*) have very special structure, which we exploit to bound the competitiveness of PBF. We prove this by developing a geometric view on the packet buffering problem; such approach turns out to be surprisingly successful. Finally, using a potential function-like argument, in Sect. 3.3, we show that the performance ratio of PBF on any regular sequence is at most 16/13. We note that the idea of reductions of arbitrary sequences to the most difficult ones can be found in the previous papers, for example in [11].

As we mentioned above, PBF is a deterministic algorithm which is optimal in an extended, fractional model. We note that the most straightforward translation of this solution into a randomized non-fractional one does not work. Instead, using techniques similar to randomized

rounding [10], we construct a two-dimensional rounding technique, which yields an optimal algorithm PB.

For clarity, the proofs of technical lemmas were moved to the appendix.

Related Work. One of the most straightforward generalizations of the considered simple scenario is the model in which packets have values and the objective is to maximize the total value of packets sent. Although yet not commonly seen in practice, these *Differentiated Services* allow Internet Service Providers to assign different levels of *Quality of Service* to different data streams.

There are several results concerning the case of maintaining a single buffer, where packets have to be transmitted in FIFO order and where preemption (eviction of packets already in buffer) is allowed. Currently, the best deterministic *preemptive greedy* algorithm due to Englert and Westermann [4] achieves a competitive ratio of 1.732 and the best known lower bound for this problem, 1.419, is due to Kesselman et al [6]. There has also been work on a so-called *bounded-delay model*, in which no FIFO order is enforced but packets have deadlines (see, e.g., [5]). Azar and Richter [2] showed how to cope with multiple queues, presenting a general technique of transforming algorithms for single queue into multiple queue algorithms, losing factor 2 in the competitive ratio.

The packet buffering problems were also considered under some probabilistic assumptions on the input sequence (see e.g. [9]). However, there is an observed evidence that the nature of data traffic in networks is chaotic [8] and does not follow standard patterns like Poisson arrival model.

2 Preliminaries

First, let us formally define the input sequence. We transform a description of packets arrivals τ into a sequence of requests σ with more convenient form. For each time step in which there are no new incoming packets, we append a request IDLE to sequence σ. For a step in which there are new packets at input ports, say x_0 packets at buffer 0 and x_1 packets at buffer 1, we append $\text{ADD}(0)^{x_1}\,\text{ADD}(1)^{x_2}\,\text{IDLE}$ to σ.

By σ_t we understand the t-th element of σ and by $\sigma|_a^b$ the contiguous subsequence of σ starting at position (step) a and ending at b. Additionally we define σ^t as the first t elements of σ, i.e. $\sigma|_1^t$; in particular, σ^0 denotes an empty sequence ϵ. We say that the request σ_t is processed in *step* t. For any two sequences σ and σ' we denote their concatenation by $\sigma\sigma'$.

Semantics of Request Sequence. For any algorithm ALG, the *state* of its buffers at the end of a given step can be described by a pair $\boldsymbol{x}^{\text{ALG}} \in \{0,\ldots,B\}\times\{0,\ldots,B\}$, where the coordinates denote the number of packets in the respective buffers. For any request (sub)sequence σ, we use $\boldsymbol{x}^{\text{ALG}}(\sigma)$ to denote the state of ALG after it starts with empty buffers and serves the sequence σ. In particular, by $\boldsymbol{x}^{\text{ALG}}(\sigma^t)$ we mean the state of ALG after processing the first t steps of σ, and thus $\boldsymbol{x}^{\text{ALG}}(\sigma^0) = (0,0)$.

The semantics of the requests from σ sequence is straightforward. Fix any step t. For an IDLE request, ALG may choose a non-empty buffer i and transmit one packet from it. Although the algorithm may also choose not to transmit a packet, any such algorithm can be transformed to a *work-conserving* one, which sends a packet whenever possible, and the competitiveness of the obtained algorithm is not worse. The way of choosing the buffer for transmission is called *pivoting rule*. Note that this rule is the only factor that determines the behavior of the algorithm.

If σ_t is an ADD(i) request, a new packet is *added* to the buffer i. If the number of packets at the i-th buffer is already B, then the packet is immediately lost. Let $\ell^{\mathrm{ALG}}(\sigma^t)$ be the number of packets that are actually added to the buffer in step t, $\ell^{\mathrm{ALG}}(\sigma^t) \in \{0,1\}$. We extend this definition to IDLE steps, setting $\ell^{\mathrm{ALG}}(\sigma^t) = 0$ for them.

At the end of the input sequence, the algorithm empties all the buffers; we may assume that they are all transmitted in one batch.

Let $\mathcal{H} = [0,B] \times [0,B]$. We may view $\boldsymbol{x}^{\mathrm{ALG}}$, the state of ALG, as a point from $\mathcal{H} \cap \mathbb{N}^2$. Then, the IDLE and ADD operations described above have obvious geometric interpretation. For any state $\boldsymbol{x}$, x_0 and x_1 denote the number of packets in the respective buffers, and $\|\boldsymbol{x}\| = x_0 + x_1$.

Fractional Model. It is now straightforward to generalize the above description to a *fractional model*, where the state of ALG can be any point from $\mathcal{H}$ (also the one with fractional coordinates). In particular, the algorithm serving IDLE request may choose to transmit fractional parts of packets from different buffers, with the only requirement that the total mass of transmitted packets is at most 1. More formally, during an IDLE request in step t of σ, ALG chooses a vector $\Delta(t) = (\delta_0, \delta_1) \in [0,1] \times [0,1]$ such that $\delta_0 + \delta_1 \leq 1$ and $x_i^{\mathrm{ALG}}(\sigma^{t-1}) \geq \delta_i(t)$ for $i \in \{0,1\}$. Then the effect of this request is reflected by a new state $\boldsymbol{x}^{\mathrm{ALG}}(\sigma^t) = \boldsymbol{x}^{\mathrm{ALG}}(\sigma^{t-1}) - \Delta(t)$ of the algorithm. Note that the definition of $\ell^{\mathrm{ALG}}(\sigma^t)$ can be extended to the fractional model in a straightforward manner.

Although we allow online algorithms to use fractional parts of the packets, to simplify our analysis we compare them to the optimal offline algorithm which still works in the standard model.[3] We note that the lower bound of 16/13 holds in the fractional model, as well.

Competitiveness. For any sequence σ and any deterministic (not necessarily online) algorithm ALG, we define a function $\mathsf{loss}_{\mathrm{ALG}}(\sigma)$ as the number of packets lost by the strategy ALG on σ, under the condition that ALG starts with empty buffers. For any algorithm ALG and two sequences σ and τ we define $\Delta_\sigma \mathsf{loss}_{\mathrm{ALG}}(\tau) = \mathsf{loss}_{\mathrm{ALG}}(\sigma\tau) - \mathsf{loss}_{\mathrm{ALG}}(\sigma)$.

Obviously, the losses can occur only due to ADD requests which overflow some buffer. Therefore, if σ is the whole sequence, $\mathsf{loss}_{\mathrm{ALG}}(\sigma) = \sum_{t:\sigma_t=\mathrm{ADD}} (1 - \ell^{\mathrm{ALG}}(\sigma^t))$. Let $S(\sigma)$ denote the total number of packets added in σ, i.e. the number of ADD requests. The throughput of ALG (the number of transmitted packets), denoted $T_{\mathrm{ALG}}(\sigma)$, is then equal to $S(\sigma) - \mathsf{loss}_{\mathrm{ALG}}(\sigma)$.

Consider a sequence σ. Let OPT be an optimal (offline) algorithm for the packet buffering problem, i.e. the one which minimizes the number of packets lost. We define the *performance ratio* of (an online) algorithm ALG on σ as

$$\mathcal{R}_{\mathrm{ALG}}(\sigma) = \frac{T_{\mathrm{OPT}}(\sigma)}{T_{\mathrm{ALG}}(\sigma)} = \frac{S(\sigma) - \mathsf{loss}_{\mathrm{OPT}}(\sigma)}{S(\sigma) - \mathsf{loss}_{\mathrm{ALG}}(\sigma)} \ . \tag{2}$$

If Σ is any set of sequences, then $\mathcal{R}_{\mathrm{ALG}}(\Sigma) = \sup_{\sigma \in \Sigma}\{R_{\mathrm{ALG}}(\sigma)\}$. Let $\mathcal{Q}$ be the set of all possible sequences; then the competitive ratio can be defined as $\mathcal{R}_{\mathrm{ALG}} = \mathcal{R}_{\mathrm{ALG}}(\mathcal{Q})$. If $\mathcal{R}_{\mathrm{ALG}} \leq \alpha$, then we call ALG *$\alpha$-competitive*.

OPT State Space As mentioned in the introduction, we would like to rely the behavior of our algorithm on tracing the state of some optimal off-line algorithm buffers in each step. Obviously, the complete knowledge about this state is not available to an on-line algorithm. Instead, we will focus on extrapolating a set of possible OPT states which can be inferred from the already

[3] Although it is not needed for our reasoning, it appears that this restriction does not constrain the power of OPT.

seen prefix of σ. In each step we trace a certain set $\mathcal{I}(\sigma^t)$ of possible states whose relation to an optimal solution is presented in Lemma 2.

We define set $\mathcal{I}$ inductively as $\mathcal{I}(\sigma^0) = \{(0,0)\}$, and

$$\mathcal{I}_{\text{pre}}(\sigma^t) = \begin{cases} (\mathcal{I}(\sigma^{t-1}) - (1,0)) \cup (\mathcal{I}(\sigma^{t-1}) - (0,1)) & \text{if } \sigma_t = \textsc{Idle} \\ \mathcal{I}(\sigma^{t-1}) + (1,0) & \text{if } \sigma_t = \textsc{Add}(0) \\ \mathcal{I}(\sigma^{t-1}) + (0,1) & \text{if } \sigma_t = \textsc{Add}(1) \end{cases}, \tag{3}$$

$$\mathcal{I}(\sigma^t) = \begin{cases} \mathcal{I}_{\text{pre}}(\sigma^t) \cap \mathcal{H} & \text{if } \mathcal{I}_{\text{pre}}(\sigma^t) \cap \mathcal{H} \neq \emptyset \\ \mathcal{I}(\sigma^{t-1}) & \text{otherwise} \end{cases}. \tag{4}$$

An intuition behind the set $\mathcal{I}(\sigma^t)$ is that it contains states of all algorithms which try to greedily reduce their loss, i.e. postpone losing packets. A straightforward induction shows that the number of packets in each state from the set $\mathcal{I}$ is the same (we denote this number by $\|\mathcal{I}\|$) and $\mathcal{I}$ consists of non-fractional states contained in an anti-diagonal interval (see Fig. 1a). We call an $\textsc{Add}$ request *hit* if it reduces the number of elements in $\mathcal{I}$.

We say that a state (k_1, k_2) *is majorized* by (k_1', k_2') (we write $(k_1, k_2) \leq (k_1', k_2')$) if $k_1 \leq k_1'$ and $k_2 \leq k_2'$. We define *shadow of* $\mathcal{I}$, denoted $\mathcal{SH}(\mathcal{I})$, as the set of all the states which are majorized by some state from $\mathcal{I}$. The following technical lemmas show the relation between the set $\mathcal{I}$, optimal solutions and other work-conserving algorithms.

Lemma 1. *For any input σ and any work-conserving algorithm $\textsc{Alg}$, $\boldsymbol{x}^{\text{ALG}}(\sigma) \in \mathcal{SH}(\mathcal{I}(\sigma))$.*

Lemma 2. *There exists an algorithm A, such that $\boldsymbol{x}^A(\sigma^t) \in \mathcal{I}(\sigma^t)$ for any step t. Every algorithm with such property is optimal and loses a packet in step t on $\textsc{Add}$ request if and only if $\mathcal{I}(\sigma^t) = \mathcal{I}(\sigma^{t-1})$.*

The main implication of Lemma 2 is that we get a convenient description of the loss of an optimal algorithm. Namely, we can compute $\mathsf{loss}_{\text{OPT}}(\sigma)$ by counting all $\textsc{Add}$ requests, for which $\mathcal{I}(\sigma^t) = \mathcal{I}(\sigma^{t-1})$. Note that such equality may occur only when $\mathcal{I}(\sigma^{t-1})$ is a singleton set at one of the upper boundaries.

Of course, we would like to achieve the performance of A, but the main difficulty in constructing an *online* algorithm mimicking such A is that usually knowing only σ^t, we can neither predict the exact state of A in step $t+1$, nor the future shape of $\mathcal{I}$ sets. However, as we already said, on the basis of computed $\mathcal{I}(\sigma^t)$, we can in some way extrapolate $\mathcal{I}(\sigma^{t+1})$ and the behavior of an optimal algorithm A.

3 Algorithm PBF

In this section, we present an algorithm PBF, which is optimal in the fractional model. Let

$$r_i(\sigma) = x_i^{\text{PBF}}(\sigma) - \min_{z \in \mathcal{I}(\sigma)} z_i \quad \text{for } i \in \{0,1\}. \tag{5}$$

Assume that the adversary decides to issue a maximum number of $\textsc{Add}(i)$ requests without incurring a loss to $\textsc{Opt}$. Then r_i would be the number of packets lost by PBF. If $r_i \leq 0$, PBF cannot lose packets in this way (see r_1 in Fig. 1b). We note that r_i can be efficiently computed by an online algorithm (as $\mathcal{I}$ can be described by a few parameters). Let $\mathsf{bal}(\sigma) = r_0(\sigma) - r_1(\sigma)$ be called *balance*.

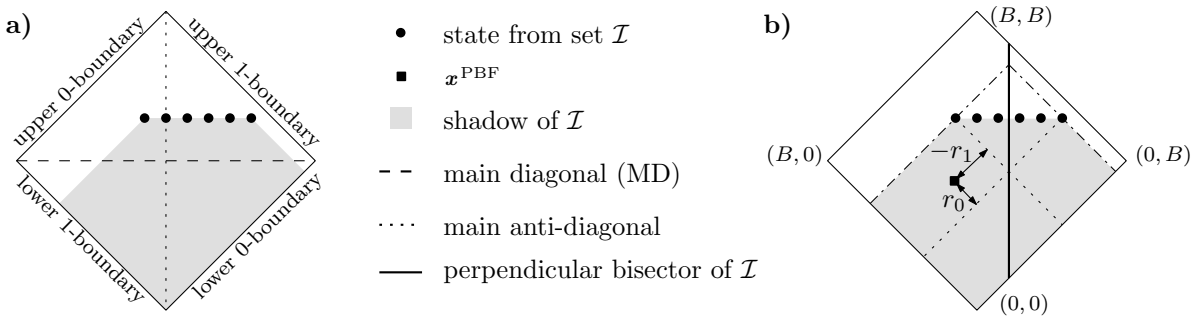

Fig. 1. Illustration of the set $\mathcal{I}$ and PBF parameters; $r_0 > 0$, $r_1 < 0$

When PBF encounters IDLE request in step t of σ, it computes a new shape of the set $\mathcal{I}(\sigma^t)$ first. Then it transmits a total mass of 1 packet, so that the resulting value of $|\mathsf{bal}(\sigma^t)|$ is as small as possible. This rule can be interpreted geometrically as choosing a new state $\boldsymbol{x}^{\mathrm{ALG}}(\sigma^t)$ as close to the perpendicular bisector (hence the abbreviation PB; F stands for fractional model) of the set $\mathcal{I}$ as possible.

3.1 Outline of the Proof

In the remaining part Sect. 3, we prove the following theorem.

Theorem 1. *For any buffer size B,* PBF *on two buffers is* 16/13*-competitive.*

We prove it inductively on B. Obviously, for $B = 0$, the theorem trivially holds (with competitiveness 1). Below we present the roadmap of the proof. As we mentioned in the introduction, the proof is divided into two parts. In the first one, we narrow down instances on which PBF has high competitive ratio. In the second part, we restrict our analysis only to these instances.

For any integers x and y, by a *block* $\mathcal{B}(x, y)$ we denote a subsequence $\textsc{Idle}^x\,\textsc{Add}(0)^y$. We also denote the sequence $\textsc{Add}(0)^B\,\textsc{Add}(1)^B$ by $\mathcal{A}$. By *main diagonal* (MD) we mean the diagonal of the square, which contains all fractional states $\boldsymbol{z}$ such that $\|\boldsymbol{z}\| = B$. We say that $\mathcal{I}$ is *above*, *at*, or *below* MD if $\|\mathcal{I}\|$ is, respectively, greater, equal, or less than B. We introduce the following classes of sequences.

Definition 1. *We denote the set of all sequences by $\mathcal{Q}$. We also define the following sets of sequences.*

- $\mathcal{Q}_N$*:* non-trivial.
  $\sigma \in \mathcal{Q}_N$ *if it ends with* ADD *incurring loss to* PBF *and* $\boldsymbol{x}^{\mathrm{PBF}}(\sigma^t) \neq (0,0)$ *for* $t > 0$.
- $\mathcal{Q}_P$*:* proper.
  $\sigma \in \mathcal{Q}_P$ *if it is non-trivial, starts with* $\mathcal{A}$*, and* $\mathcal{I}(\sigma^t)$ *is above MD for all* $t \geq |\mathcal{A}|$.
- $\mathcal{Q}_U$*:* uniform.
  $\sigma \in \mathcal{Q}_U$ *if* σ *is proper and after initial* $\mathcal{A}$*, consists only of* ADD(0) *and* IDLE *requests.*
- $\mathcal{Q}_R$*:* regular.
  $\sigma \in \mathcal{Q}_R$ *if* σ *is uniform and has the form* $\mathcal{A}\,\mathcal{B}(x_1, y_1)\,\mathcal{B}(x_2, y_2)\,\ldots\,\mathcal{B}(x_n, y_n)$*, where after each block* $\mathcal{B}(x_i, y_i)$*,* $\boldsymbol{x}_0^{\mathrm{PBF}} = B$.

The course of the proof is to show each of consecutive relations below.

$$\mathcal{R}_{\mathrm{PBF}} \leq \mathcal{R}_{\mathrm{PBF}}(\mathcal{Q}_N) \leq \mathcal{R}_{\mathrm{PBF}}(\mathcal{Q}_P) = \mathcal{R}_{\mathrm{GR}}(\mathcal{Q}_P) \leq \mathcal{R}_{\mathrm{GR}}(\mathcal{Q}_U) \leq \mathcal{R}_{\mathrm{GR}}(\mathcal{Q}_R) \tag{6}$$

$\mathcal{R}_{\mathrm{GR}}$ denotes the performance ratio of a natural Greedy algorithm (see [11]), which is defined formally later. We show these inequalities in Sect. 3.2. Then, in Sect. 3.3, we prove that $\mathcal{R}_{\mathrm{GR}}(\mathcal{Q}_R) \leq 16/13$. On the other hand, on the sequence $\nu(B) = \mathcal{A}\mathcal{B}(B,B)\,\mathcal{B}(B,B)$. PBF transmits $\frac{13}{4} \cdot B$ packets, whereas Opt transmits $4 \cdot B$. Hence, for $B \geq 1$, it holds that $\mathcal{R}_{\mathrm{PBF}} \geq \mathcal{R}_{\mathrm{PBF}}(\nu(B)) = \frac{16}{13}$, and therefore all inequalities above can be replaced by equalities. We note that by [1,12], the competitive ratio of any online algorithm Alg is at least $h(2) = \frac{16}{13}$, and therefore PBF is optimal.

3.2 Worst-case Sequences

In this section, we consecutively prove all inequalities of (6) but the last one. In general, in order to show that $\mathcal{R}_{\mathrm{ALG}}(\mathcal{Q}_1) \leq \mathcal{R}_{\mathrm{ALG}}(\mathcal{Q}_2)$, we show that for any $\sigma \in \mathcal{Q}_1$, we can transform it to obtain a sequence $\widehat{\sigma} \in \mathcal{Q}_2$, such that the performance ratio of Alg does not decrease. Intuitively, $\widehat{\sigma}$ is more difficult for Alg than σ.

Changes in Balance. We start from several simple definitions and observations. We define $\mathsf{len}(\mathcal{I})$ as the length of the smallest interval containing set $\mathcal{I}$. This amount is equal to the number of $\mathcal{I}$ elements minus 1.

We conceptually divide each step into two parts. In the first one, the adversary issues a request and as a result the set $\mathcal{I}$ is changed. In case of an Add request, $\boldsymbol{x}^{\mathrm{PBF}}$ is changed as well. In the second part, which is present only for Idle requests, PBF transmits some packets.

As a result of an Add(i) request, r_i may decrease by one. This can happen only if just before this request the set $\mathcal{I}$ *touches* (i.e. has non-empty intersection with) the upper i-boundary of $\mathcal{H}$, where *upper i-boundary* is $\mathcal{H} \cap \{(k_0, k_1) : k_i = B\}$. We call such an Add($i$) request a *hit*; such an Add reduces the value of $\mathsf{len}(\mathcal{I})$ by one.

If $\mathcal{I}$ is above MD, then in a step with Idle request, both r_i increase by 1, and therefore the balance remains unchanged. On the other hand, if $\mathcal{I}$ is at or below MD and it touches the lower i-boundary of $\mathcal{H}$, the corresponding r_i remains unchanged. If $\mathcal{I}$ touches only one boundary, the balance may therefore change by one. In total, upon an Idle request, in the first part of a step $\mathsf{len}(\mathcal{I})$ increases by 1 minus the number of lower boundaries $\mathcal{I}$ touches and in the second part PBF changes its state according to its pivoting rule. These observations lead to the following technical lemmas.

Lemma 3. *For any sequence σ, if $\mathcal{I}(\sigma)$ is below or at the main diagonal, then $\mathsf{bal}(\sigma^t) = 0$.*

Lemma 4. *For any sequence σ, there exists a non-trivial sequence $\widehat{\sigma}$, such that $\mathcal{R}_{\mathrm{PBF}}(\sigma) \leq \mathcal{R}_{\mathrm{PBF}}(\widehat{\sigma})$.*

Proper Sequences. Consider a non-trivial sequence σ. By the definition, at the end of σ, set $\mathcal{I}$ touches an upper boundary (in the following informal description, we assume that it touches both boundaries). However, if we look from the adversary point of view, it is not obvious when this should happen for the first time. It is also not clear that keeping the set $\mathcal{I}$ below the main diagonal is not preferable — even if this constraints the growth of the set $\mathcal{I}$, the resulting constraints on PBF's behavior might be beneficial for the adversary.

We address the issues above by showing that proper sequences incur the worst performance ratio of PBF. Namely, we prove that for any non-trivial sequence σ, there exists another sequence $\widehat{\sigma}$ with not smaller performance ratio, such that $\widehat{\sigma}$ starts with filling the buffers with packets and later it keeps the set $\mathcal{I}$ all the time above the diagonal.

We construct $\widehat{\sigma}$ on the basis of σ, so that after initial filling the buffers, $\widehat{\sigma}$ contains almost the same steps as σ. We can imagine that we have two instances of PBF running "in parallel" on σ and on $\widehat{\sigma}$. We show that it is possible to maintain an invariant that the set $\mathcal{I}$ and the point $\boldsymbol{x}^{\mathrm{PBF}}$ for $\widehat{\sigma}$ are equal to $\mathcal{I}$ and $\boldsymbol{x}^{\mathrm{PBF}}$ for σ translated by some vector. This invariant allows us to prove that the performance of PBF can only worsen by replacing σ with $\widehat{\sigma}$.

How do we create $\widehat{\sigma}$? If the request of σ does not change $\mathsf{len}(\mathcal{I})$ and the spatial relation between $\mathcal{I}$ and $\boldsymbol{x}^{\mathrm{PBF}}$, we do not append anything to $\widehat{\sigma}$. Otherwise, if we have an Add request in σ, which is a hit, then this Add appended to $\widehat{\sigma}$ is a hit as well (as $\mathcal{I}(\widehat{\sigma})$ touches both boundaries). The only problem arises when we have an Idle request in σ occurring when $\mathcal{I}$ touches both lower boundaries, as in this case $\mathsf{len}(\mathcal{I})$ decreases. To simulate such change also on the instance $\widehat{\sigma}$, we introduce an additional request Lift. We justify this enhancement later in this section, by showing that the adversary does not need Lift to impose the worst competitive ratio.

Making Sequences Proper. For the formal proof, we have to extend the notion of proper sequences. Let $\mathcal{H}(k_0, k_1)$ be the largest square contained entirely in $\mathcal{H}$, with an upper corner at (k_0, k_1), i.e., the edge length of such square is $\min\{k_0, k_1\}$. Note that $\mathcal{H}(B, B) = \mathcal{H}$. We say that a sequence σ is (k_0, k_1)-*proper* (or just *proper* if $k_0 = k_1 = B$) if it begins with $\textsc{Add}(0)^{k_0}\textsc{Add}(1)^{k_1}$ and $\mathcal{I}(\sigma^t)$ is always contained in the triangle corresponding to the half of $\mathcal{H}(k_0, k_1)$ above its main diagonal. By a straightforward induction, we get the following fact.

Fact 2 *If σ is (k_0, k_1)-proper, then $\mathcal{I}(\sigma^t)$ has always maximal possible length, i.e., it contains all states of $\mathcal{H}(k_0, k_1)$ with $\|\mathcal{I}(\sigma^t)\|$ packets.*

Let $b_i(\sigma^t) = \min_{\boldsymbol{z} \in \mathcal{I}(\sigma^t)} B - z_i$ be the distance between $\mathcal{I}$ and the upper i-boundary in step t. Let $b_i^*(\sigma) = \min_{1 \le t \le |\sigma|} b_i(\sigma^t)$. We note that both $\|\boldsymbol{b}\|$ and $\|\boldsymbol{b}^*\|$ start from $2 \cdot B$. Moreover, b_i^* never increases during runtime.

We also introduce a new request Lift, which adds a half of a packet to both buffers of PBF. When it is issued in step t, it changes set $\mathcal{I}$ in a way opposite to the effect an Idle request would have. Namely, we extend definition (3) by the following case.

$$\mathcal{I}_{\mathrm{pre}}(\sigma^t) = (\mathcal{I}(\sigma^{t-1}) + (1,0)) \cap (\mathcal{I}(\sigma^{t-1}) + (0,1)) \cap \mathcal{H}(b_0(\sigma^{t-1}), b_1(\sigma^{t-1})) \quad \text{if } \sigma_t = \textsc{Lift} \tag{7}$$

It appears that this artificial request helps us to normalize the behavior of PBF. Our goal is to show a transformation from an arbitrary sequence without Lifts to a proper sequence possibly containing Lifts. The most important property of this transformation is that in each step t it preserves the spatial relation between $\mathcal{I}(\sigma^t)$ and $\boldsymbol{x}^{\mathrm{PBF}}(\sigma^t)$.

Let $\mathcal{P}_\sigma(\sigma)$ be constructed in the following manner. We define $\Delta_\sigma \mathsf{len}(a) \equiv \mathsf{len}(\mathcal{I}(\sigma a)) - \mathsf{len}(\mathcal{I}(\sigma))$. Let $\mathcal{P}_\sigma(\epsilon) = \textsc{Add}(0)^{B - b_0^*(\sigma)}\textsc{Add}(1)^{B - b_1^*(\sigma)}$. Let

$$\mathcal{P}_\sigma(\sigma^{t+1}) = \begin{cases} \mathcal{P}_\sigma(\sigma^t)\,\textsc{Idle} & \text{if } \sigma_{t+1} = \textsc{Idle} \text{ and } \Delta_{\sigma^t}\mathsf{len}(\sigma_{t+1}) = 1, \\ \mathcal{P}_\sigma(\sigma^t)\,\textsc{Lift} & \text{if } \sigma_{t+1} = \textsc{Idle} \text{ and } \Delta_{\sigma^t}\mathsf{len}(\sigma_{t+1}) = -1, \\ \mathcal{P}_\sigma(\sigma^t)\,\textsc{Add}(i) & \text{if } \sigma_{t+1} = \textsc{Add}(i) \text{ and this } \textsc{Add}(i) \text{ is a hit}, \\ \mathcal{P}_\sigma(\sigma^t) & \text{otherwise.} \end{cases} \tag{8}$$

Although this is not necessary in our reasoning, we observe that if σ is proper, then $\mathcal{P}_\sigma(\sigma) = \sigma$.

Lemma 5. *For any σ and corresponding $\mathcal{P}_\sigma(\sigma)$, the following invariants hold for all $0 \le t \le |\sigma|$:*

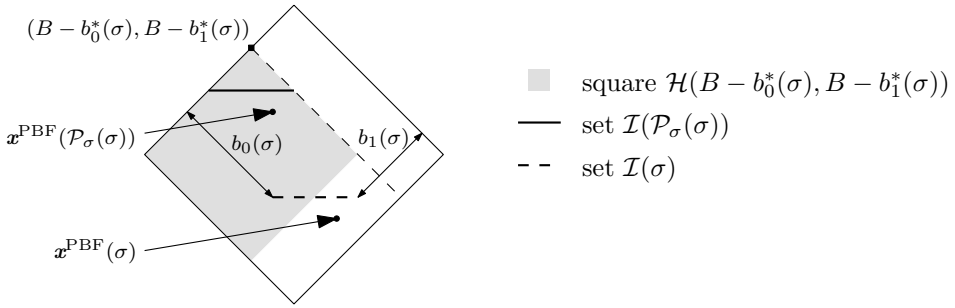

Fig. 2. Transformation $\mathcal{P}$

(i) $\mathit{len}(\mathcal{P}_\sigma(\sigma^t)) = \mathit{len}(\sigma^t)$;
(ii) $r_i(\mathcal{P}_\sigma(\sigma^t)) = r_i(\sigma^t)$ *for* $i \in \{0, 1\}$;
(iii) $S(\sigma^t) - S(\mathcal{P}_\sigma(\sigma^t)) + \|\boldsymbol{b}^*(\sigma)\| + \min_i\{b_i(\sigma^t) - b_i^*(\sigma^t)\} \geq 0$;
(iv) $\mathit{loss}_{\mathrm{OPT}}(\mathcal{P}_\sigma(\sigma^t)) = \mathit{loss}_{\mathrm{OPT}}(\sigma^t)$ *and* $\mathit{loss}_{\mathrm{PBF}}(\mathcal{P}_\sigma(\sigma^t)) = \mathit{loss}_{\mathrm{PBF}}(\sigma^t)$;
(v) $\mathcal{P}_\sigma(\sigma^t)$ *is* $(B - b_0^*(\sigma), B - b_1^*(\sigma))$*-proper.*

Proof. We prove the invariants inductively. At the very beginning, for $t = 0$, invariants (i), (ii), (iv) and (v) trivially hold. For invariant (iii), we note that $S(\epsilon) = 0$, $S(\mathcal{P}_\sigma(\epsilon)) = \|\boldsymbol{b}^*(\sigma)\|$, and $\boldsymbol{b}^*(\sigma^0) = \boldsymbol{b}(\sigma^0)$.

Assume that invariants hold for some $t < |\sigma|$; we show that they hold also for $t + 1$. The cases are depicted on Fig. 3. By MD' we mean the main diagonal of $\mathcal{H}(B - b_0^*(\sigma), B - b_1^*(\sigma))$.

1. $\sigma_{t+1} = \textsc{Idle}$. Invariant (iv) holds trivially as Idle request cannot generate loss. Since σ is non-trivial, it holds that $\|\boldsymbol{x}^{\mathrm{PBF}}(\sigma^t)\| \geq 1$. We prove invariants (i), (ii), (iii) and (v) for the following three subcases, depending on the value of $\Delta_{\sigma^t}\mathsf{len}(\sigma_{t+1})$.
   (a) If $\Delta_{\sigma^t}\mathsf{len}(\sigma_{t+1}) = 1$, then an Idle request is appended to $\mathcal{P}_\sigma(\sigma^t)$. We note that in this case $\mathsf{len}(\sigma^t)$ has to be shorter than the length of MD'. By invariant (i), $\mathsf{len}(\mathcal{P}_\sigma(\sigma^t)) = \mathsf{len}(\sigma^t)$, and thus Fact 2 implies that $\mathcal{I}(\mathcal{P}_\sigma(\sigma^t))$ lies strictly above MD'. Therefore, $\Delta_{\mathcal{P}(\sigma^t)}\mathsf{len}(\textsc{Idle}) = 1$, which proves invariants (i) and (v). Moreover, both values of r_i for σ^t and $\mathcal{P}_\sigma(\sigma^t)$ increase by 1, which yields invariant (ii). Invariant (iii) is fulfilled trivially, as none of the terms on its left hand side changes.
   (b) If $\Delta_{\sigma^t}\mathsf{len}(\sigma_{t+1}) = 0$, nothing is appended to $\mathcal{P}_\sigma(\sigma^t)$, and thus $\mathsf{len}(\mathcal{P}_\sigma(\sigma^t))$ does not change. This proves invariants (i) and (v). The considered case may occur only if $\mathcal{I}(\sigma^t)$ is at or below the diagonal of $\mathcal{H}(B - b_0^*(\sigma), B - b_1^*(\sigma))$, which, by Lemma 3, implies that $\mathsf{bal}(\sigma^t) = \mathsf{bal}(\sigma^{t+1}) = 0$. As $\|\boldsymbol{x}^{\mathrm{PBF}}(\sigma^t)\| \geq 1$, PBF is able to transmit a packet, and it can compensate the increase of one of r_i caused by the shift of $\mathcal{I}$. This proves invariant (ii). For proving invariant (iii), we note that only b_i can change, but they may only increase.
   (c) If $\Delta_{\sigma^t}\mathsf{len}(\sigma_{t+1}) = -1$, then a Lift is appended to $\mathcal{P}_\sigma(\sigma^t)$. Since $\mathcal{P}_\sigma(\sigma^t)$ is $(B - b_0^*(\sigma), B - b_1^*(\sigma))$-proper, by Fact 2, the corresponding set $\mathcal{I}(\mathcal{P}_\sigma(\sigma^t))$ has maximal possible length and therefore $\Delta_{\mathcal{P}_\sigma(\sigma^t)}\mathsf{len}(\textsc{Lift}) = -1$, which shows invariant (i) and (v). The proof of invariant (ii) is the same as in the previous case, because $\mathsf{bal}(\mathcal{P}_\sigma(\sigma^t)\,\textsc{Lift}) = \mathsf{bal}(\mathcal{P}_\sigma(\sigma^t))$. For proving invariant (iii) , we note that the increase of $S(\mathcal{P}_\sigma(\sigma^t))$ is compensated by the increase of b_0 and b_1.
2. $\sigma_{t+1} = \textsc{Add}(j)$ request. We consider two subcases, depending on whether this request was a hit.

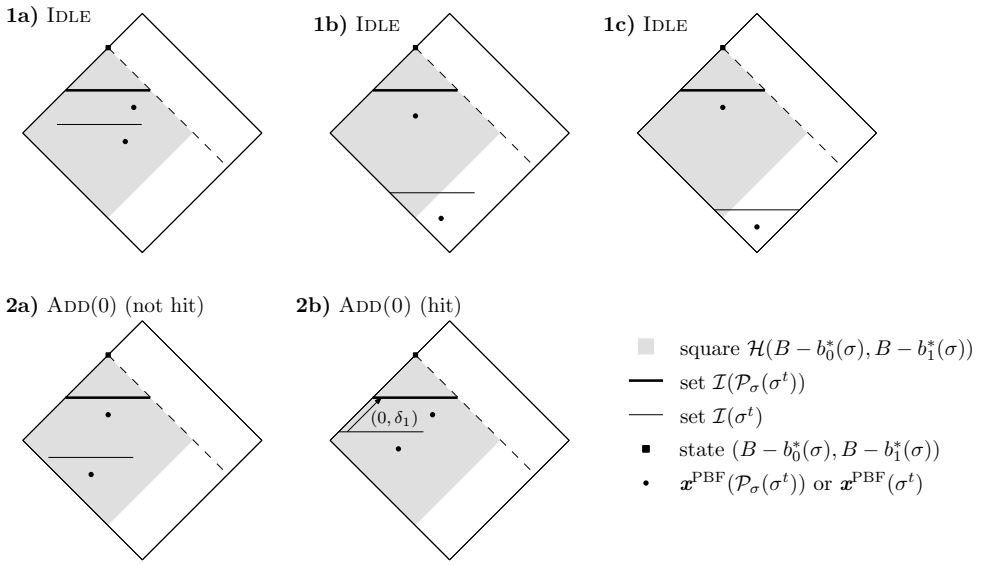

Fig. 3. Illustration for the case analysis of Lemma 5.

(a) If $\textsc{Add}(j)$ is not a hit, then $\mathcal{P}_\sigma(\sigma^{t+1}) = \mathcal{P}_\sigma(\sigma^t)$, and thus the increase of $\mathsf{len}(\mathcal{I}(\mathcal{P}(\sigma))$ and $\mathsf{len}(\mathcal{I}(\sigma))$ is equal to 0. In this case, r_i values remain unchanged both for sequence σ^t and $\mathcal{P}_\sigma(\sigma^t)$. Additionally, there is no loss incurred neither on Opt nor on PBF since by Lemma 1 only a hit can incur loss to PBF. This proves invariants (i), (ii), (iv), and (v). For invariant (iii), we note that $S(\sigma^t) - S(\mathcal{P}_\sigma(\sigma^t))$ increases by 1 and b_j decreases by 1. If $b_j^*(\sigma^t)$ does not change, then the invariant holds. If b_j^* decreases, it may happen only if $b_j^*(\sigma^t) = b_j(\sigma^t)$. In this case, the last term on the left side of the inequality is equal to zero, and the invariant holds as well.

(b) If $\textsc{Add}(j)$ is a hit, then $\textsc{Add}(j)$ is appended to $\mathcal{P}_\sigma(\sigma^t)$ sequence. Obviously, invariant (v) is preserved. By Fact 2, $\textsc{Add}(j)$ is also a hit in $\mathcal{P}_\sigma(\sigma)$, i.e., both $\mathcal{I}(\sigma^t)$ and $\mathcal{I}(\mathcal{P}(\sigma^t))$ touch the upper j-boundary. By invariant (i), we have that $\mathcal{I}(\mathcal{P}_\sigma(\sigma^t)) = \mathcal{I}(\sigma^t) + (\delta_0, \delta_1)$, where $\delta_{1-j} \geq 0$ and $\delta_j = 0$. Furthermore, this observation combined with invariant (ii) for step t implies $\boldsymbol{x}^{\mathrm{PBF}}(\mathcal{P}_\sigma(\sigma^t)) = \boldsymbol{x}^{\mathrm{PBF}}(\sigma^t) + (\delta_0, \delta_1)$. Because the shift caused by $\textsc{Add}(j)$ is perpendicular to the vector (δ_0, δ_1), $\mathsf{loss}_{\mathrm{OPT}}$ and $\mathsf{loss}_{\mathrm{PBF}}$ incurred in this step depend only on the length $\mathsf{len}(\mathcal{I})$ and r_i values. We conclude that invariant (ii) and (iv) are preserved. Finally, since $\textsc{Add}(j)$ is a hit, $\boldsymbol{b}^*$ and $\boldsymbol{b}$ do not change and both $S(.)$ increase by one, which proves invariant (iii).

This finishes the proof of all invariants. □

We use the lemma above to show that in the analysis of PBF, we may restrict our consideration to proper sequences only.

Lemma 6. *For any non-trivial sequence σ, there exists a proper sequence $\widehat{\sigma}$ (possibly containing* Lift *requests) such that $\mathcal{R}_{\mathrm{PBF}}(\sigma) \leq \mathcal{R}_{\mathrm{PBF}}(\widehat{\sigma})$.*

Proof. As σ is non-trivial, by Lemma 1, it ends with a hit. W.l.o.g., we may assume that it ends with $\textsc{Add}(0)$. Therefore, $b_0^*(\sigma) = b_0(\sigma) = 0$, and by the technical lemma above, $\mathcal{P}_\sigma(\sigma)$ is $(B, B - b_1^*(\sigma))$-proper. If additionally $b_1^*(\sigma) = 0$, then by invariant (iii) of Lemma 5, $S(\sigma) \geq S(\mathcal{P}_\sigma(\sigma))$. This combined with invariant invariant (iv) of the same lemma allows to take $\widehat{\sigma} = \mathcal{P}_\sigma(\sigma)$ and obtain $\mathcal{R}_{\mathrm{PBF}}(\sigma) \leq \mathcal{R}_{\mathrm{PBF}}(\widehat{\sigma})$.

The problem arises if $b_1^*(\sigma) > 0$. Consider $\mathcal{P}'_\sigma(\sigma)$ equal to $\mathcal{P}_\sigma(\sigma)$ but with $b_1^*(\sigma)$ of ADD(0) requests removed from its beginning. By Lemma 5, $\mathcal{P}'_\sigma(\sigma)$ is a proper sequence in square $[0, B - b_1^*(\sigma)] \times [0, B - b_1^*(\sigma)]$. The only difference between $\mathcal{P}'_\sigma(\sigma)$ and $\mathcal{P}_\sigma(\sigma)$ is that $\mathcal{P}'_\sigma(\sigma)$ uses $\boldsymbol{b}_1^*(\sigma)$ packets less and thus, by invariant (iii) of Lemma 5, $S(\sigma) \geq S(\mathcal{P}_\sigma(\sigma))$. In effect, $\mathcal{R}_{\mathrm{PBF}}(\sigma) \leq \mathcal{R}_{\mathrm{PBF}}(\mathcal{P}'_\sigma(\sigma))$. We note that the latter performance ratio holds if it is computed for smaller buffer sizes, namely for $B' = B - b_1^*(\sigma)$. However, by the inductive assumption, PBF is 16/13-competitive for B', and hence $\mathcal{R}_{\mathrm{PBF}}(\sigma) \leq 16/13$. At this point we may stop our analysis, because this is what we want to prove in Thm. 1. On the other hand, we may prove the lemma itself by choosing $\widehat{\sigma} = \nu(B)$ or any other proper sequence with the performance ratio equal to 16/13. □

Uniform Sequences. An important observation at this stage is that when we constrain our consideration only to proper sequences σ (possibly containing LIFTs), PBF behaves like a GREEDY algorithm. The set $\mathcal{I}$ always touches both upper boundaries of $\mathcal{H}$ (and in fact, the whole set $\mathcal{I}$ is therefore defined by a single variable $\|\mathcal{I}\|$). Thus, its perpendicular bisector always coincides with the main anti-diagonal of $\mathcal{H}$ and PBF just tries to move as close to it as possible, i.e., transmits packets to minimize the maximal level of packets in its buffers. This is exactly the pivoting rule of GREEDY. Note that it does not mean that PBF is just a GREEDY algorithm, which is known to be not optimal. It only means that it behaves in such manner on special kind of sequences (the proper ones). The worst-case behavior for the GREEDY algorithm is incurred by a sequence which do not start from full buffers (see [11]). Now, we want to further simplify the structure of the worst-case instances.

Lemma 7. *For any proper sequence σ, which may contain* LIFT *requests, there exists a uniform sequence $\widehat{\sigma}$, such that $\mathcal{R}_{\mathrm{GR}}(\sigma) \leq \mathcal{R}_{\mathrm{GR}}(\widehat{\sigma})$.*

Proof. We show a series of transformations of σ, which eventually lead to a uniform sequence $\widehat{\sigma}$. If σ is not uniform, it contains some LIFT and/or ADD(1) requests; let σ_{k+1} be the last of them. We show that if we replace σ_{k+1} by ADD(0) and possibly exchange the roles of ADD(0) and ADD(1) in σ^k, then the resulting sequence, called $\widetilde{\sigma}$, incurs non-smaller loss on GREEDY. As the number of injected packets is the same for σ and $\widetilde{\sigma}$ and on proper sequences the set $\mathcal{I}$ behaves identically for any injection request (ADD(0), ADD(1) or LIFT), we obtain $\mathcal{R}_{\mathrm{GR}}(\widetilde{\sigma}) \geq \mathcal{R}_{\mathrm{GR}}(\sigma)$. After repeating the above operation at most $|\sigma|$ times, we end up with a desired uniform sequence $\widehat{\sigma}$. (This sequence is equal to σ with all ADD(1) and LIFT requests replaced by ADD(0).)

In the following, for succinctness, we omit GR subscripts and superscripts. If $x_0(\sigma^k) \geq x_1(\sigma^k)$, then we set $\widetilde{\sigma}^k = \sigma^k$, otherwise we set $\widetilde{\sigma}^k$ to be equal to σ^k with the roles of ADD(0) and ADD(1) exchanged. This way, it is guaranteed that $x_0(\widetilde{\sigma}^k) \geq x_1(\widetilde{\sigma}^k)$ and $\|\boldsymbol{x}(\widetilde{\sigma}^k)\| = \|\boldsymbol{x}(\sigma^k)\|$. As already stated above, $\widetilde{\sigma}_{k+1} = \mathrm{ADD}(0)$. Moreover, $\widetilde{\sigma}$ has the same length as σ and $\widetilde{\sigma}|_{k+2}^{|\sigma|} = \sigma|_{k+2}^{|\sigma|}$.

Recall that $x_0(\widetilde{\sigma}^k) \geq x_1(\widetilde{\sigma}^k)$. Furthermore, all the requests within $\widetilde{\sigma}|_{k+1}^{|\sigma|}$ are either ADD(0) that increase only x_0 coordinate or IDLE requests that (for GREEDY) cannot change the ordering of x_0 and x_1. Therefore,

$$x_0(\widetilde{\sigma}^t) \geq x_1(\widetilde{\sigma}^t) \tag{9}$$

for any $t \in \{k, \ldots, |\sigma|\}$. Now, we show inductively show that the following invariants hold for any $t \in \{k, \ldots, |\sigma|\}$.

(i) $\|\boldsymbol{x}(\sigma^t)\| \geq \|\boldsymbol{x}(\widetilde{\sigma}^t)\|$
(ii) $x_0(\widetilde{\sigma}^t) - x_1(\widetilde{\sigma}^t) \geq x_0(\sigma^t) - x_1(\sigma^t)$

The induction basis holds trivially by the choice of $\widetilde{\sigma}^k$. We assume that these invariants hold for some $t < |\sigma|$ and we consider two cases.

1. $\widetilde{\sigma}_{t+1} = \textsc{Idle}$. In this case, invariant (i) is trivially preserved. For showing invariant (ii), we define $\delta = x_0(\sigma^t) - x_1(\sigma^t)$. If $\delta \geq 1$, then both instances of Greedy transmit a packet from buffer 0 and thus invariant (ii) is preserved. If $0 \leq \delta < 1$, then $x_0(\sigma^{t+1}) - x_1(\sigma^{t+1}) = 0$. If $\delta < 0$, then again we use the fact that Greedy does not change the relation between x_0 and x_1, and therefore $x_0(\sigma^{t+1}) - x_1(\sigma^{t+1}) \leq 0$. But (9) implies that $x_0(\widetilde{\sigma}^{t+1}) - x_1(\widetilde{\sigma}^{t+1}) \geq 0$, which settles invariant (ii).
2. $\widetilde{\sigma}_{t+1} = \textsc{Add}(0)$. By adding invariants (ii)) and (i) for step t and reorganizing terms, we obtain $x_1(\sigma^t) \geq x_1(\widetilde{\sigma}^t)$. As the buffer 1 is left untouched in step $t+1$, $x_1(\sigma^{t+1}) \geq x_1(\widetilde{\sigma}^{t+1})$. For showing invariant (i), we observe that if $x_0(\sigma^t) \leq B-1$, then the LHS of invariant (i) increases by 1 and the RHS increases by at most 1. Otherwise, $\|\boldsymbol{x}(\sigma^{t+1})\| = B + x_1(\sigma^{t+1}) \geq B + x_1(\widetilde{\sigma}^{t+1}) \geq \|\boldsymbol{x}(\widetilde{\sigma}^{t+1})\|$. For showing invariant (ii), we observe that if $x_0(\widetilde{\sigma}^t) \leq B-1$, then the LHS of invariant (ii) increases by 1 and the RHS increases by at most 1. Otherwise, $x_0(\widetilde{\sigma}^{t+1}) - x_1(\widetilde{\sigma}^{t+1}) = B - x_1(\widetilde{\sigma}^{t+1}) \geq B - x_1(\sigma^{t+1}) \geq x_0(\sigma^{t+1}) - x_1(\sigma^{t+1})$.

Finally, we may compare the losses of Greedy on σ and $\widetilde{\sigma}$. During any packet injection, the loss of an algorithm plus the increase of $\|\boldsymbol{x}\|$ is equal to 1. Therefore, invariant (i) implies that $\mathsf{loss}(\widetilde{\sigma}) \geq \mathsf{loss}(\sigma)$. □

Regular Sequences. We cannot analyze uniform sequences easily, because Idle and Add(0) requests can be arbitrarily mixed. In order to alleviate this problem, we show how to change a uniform sequence into a regular one.

Lemma 8. *For any uniform sequence σ, there exists a regular sequence $\widehat{\sigma}$, such that $\mathcal{R}_{\mathrm{GR}}(\sigma) \leq \mathcal{R}_{\mathrm{GR}}(\widehat{\sigma})$.*

Proof. We denote a subsequence $\textsc{Add}(0)\,\textsc{Idle}$ by $\mathcal{F}$. We process σ from the beginning to the end, looking for $\mathcal{F}$. We show that if σ contains $\mathcal{F}$ and $x_0^{\mathrm{GR}} \leq B-1$ before processing this $\mathcal{F}$, then we show that such $\mathcal{F}$ can be removed from σ without decreasing the performance ratio. Moreover, after the removal the sequence remains proper (and thus also uniform). By applying this removal inductively to any occurrence of $\mathcal{F}$ in σ, we eventually get a regular $\widehat{\sigma}$.

Assume that $\sigma = \sigma_{\mathrm{prec}}\,\mathcal{F}\,\sigma_{\mathrm{succ}}$. We look separately at the change of the throughput of Opt and Greedy. If $x_0^{\mathrm{GR}}(\sigma_{\mathrm{prec}}) \leq B-1$, then obviously the removal of $\mathcal{F}$ from σ does not change the state of Greedy and decreases its throughput by 1. The change of T_{OPT} is twofold. First, it decreases by 1, since one Idle was removed. Second, the throughput on σ_{succ} may only increase, because $\|\mathcal{I}(\sigma_{\mathrm{prec}})\| \geq \|\mathcal{I}(\sigma_{\mathrm{prec}}\,\mathcal{F})\|$. Moreover, in either case, $\|I\|$ can only increase after the removal, which implies that the new sequence is also uniform. □

3.3 Performance Ratio on Regular Sequences

In this section we prove that the performance ratio of Greedy on any regular sequence is at most 16/13. By the previous section, this will prove the competitiveness of PBF. We begin with an observation on the behavior of Greedy on regular sequences.

Lemma 9. *Fix any regular sequence $\sigma = \mathcal{A}\,\mathcal{B}(x_1, y_1), \mathcal{B}(x_2, y_2) \ldots \mathcal{B}(x_n, y_n)$. Then $\boldsymbol{x}^{\mathrm{GR}}(\sigma) = (B, B-\gamma_i)$, where $\gamma_0 = 0$ and $\gamma_i \in [0, B)$ for all $i \leq n$. Moreover, we have the following recurrence relation for γ_i*

$$\gamma_i = \begin{cases} \gamma_{i-1} & \text{if } x_i \leq \gamma_{i-1} \\ \frac{\gamma_{i-1}+x_i}{2} & \text{if } x_i > \gamma_{i-1} \end{cases}.$$

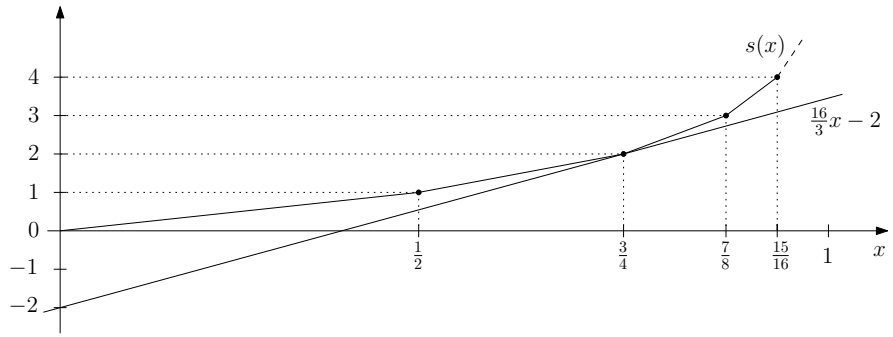

Fig. 4. Function $s(x)$

Proof. We prove the lemma inductively. For $i = 0$, GREEDY ends at (B, B) and the lemma follows trivially.

Assume that the lemma holds for $i-1$. If $x_i \leq \gamma_{i-1}$, then during IDLE^{x_i} steps the algorithm transmits from the first buffer, ending in the state $(B - x_i, B - \gamma_i)$. On the other hand, if $x_i > \gamma_{i-1}$, then during the IDLE^{x_i} requests GREEDY first transmits γ_{i-1} packets from the buffer 0 and then $\frac{1}{2} \cdot (x_i - \gamma_{i-1})$ packets from each buffer, ending at the state $(B - \frac{1}{2}(\gamma_{i-1} + x_i), B - \frac{1}{2}(\gamma_{i-1} + x_i))$.

Since σ is regular, within the following $\text{ADD}(0)^{y_i}$ requests, GREEDY fills up buffer 0, ending at state $(B, B - \gamma_{i-1})$ or $(B, B - \frac{1}{2}(\gamma_{i-1} + x_i))$, respectively. As σ is proper, $x_i \leq B$, and therefore $\gamma_i < B$ for all $i \leq n$. □

Function s and the Competitive Ratio. Before we continue with the proof of the competitiveness, we need to introduce a function s on interval $[0, 1)$. We use it to lower-bound the number of packets needed by the adversary to achieve a specific state of the algorithm. We refrain from giving a closed-form formula, as it makes the construction unnecessarily complicated. Fix any $x \in [0, 1)$. Let i be an integer such that $x \in [1 - 2^{-i}, 1 - 2^{-(i+1)})$. We define

$$s(x) = i + \frac{x - (1 - 2^{-i})}{2^{-(i+1)}} \quad . \tag{10}$$

It is easy to check that $s(x)$ is continuous, piecewise linear, and monotonically increasing. Its plot is presented in Fig. 4. In the appendix, we present the proofs of the two following technical lemmas.

Lemma 10. *For any $0 \leq a < b \leq 1$, it holds that $s(\frac{a+b}{2}) - s(a) \leq b$.*

Lemma 11. *For any $a \in [0, 1)$, it holds that $s(a) \geq \frac{16}{3} \cdot a - 2$.*

Lemma 12. *For any regular sequence $\sigma = \mathcal{A}\,\mathcal{B}(x_1, y_1)\,\mathcal{B}(x_2, y_2) \ldots \mathcal{B}(x_n, y_n)$, and a corresponding sequence $\gamma_1, \gamma_2, \ldots, \gamma_n$, it holds that $\sum_{i=1}^{n} x_i/B \geq s(\gamma_n/B)$.*

Proof. We note, without proof, that if all x_i were equal to B, then in the lemma statement we would have an equality. We prove the lemma inductively, i.e., we show that $\sum_{i=1}^{j} x_i/B \geq s(\gamma_j/B)$ holds for any $1 \leq j \leq n$. For $j = 0$, we have $s(\gamma_0) = s(0) = 0$.

Assume that the bound holds for $j - 1$. We consider two cases. If $x_j \leq \gamma_{j-1}$, then by Lemma 9, $\gamma_j = \gamma_{j-1}$ and the lemma follows. If $x_j > \gamma_{j-1}$, then by Lemma 10 and Lemma 9, it holds that

$$x_j/B \geq s\left(\frac{x_j/B + \gamma_{j-1}/B}{2}\right) - s(\gamma_{j-1}/B) = s(\gamma_j/B) - s(\gamma_{j-1}/B) \quad .$$

By adding the inequality above to the induction hypothesis, we get $\sum_{i=1}^{j} x_i/B \geq s(\gamma_j/B)$. □

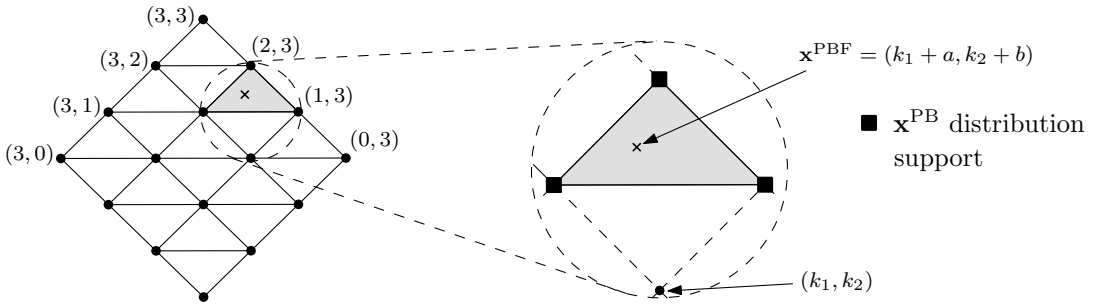

Fig. 5. Distribution of PB possible states

Lemma 13. *For any regular sequence σ, $\mathcal{R}_{\mathrm{GR}}(\sigma) \leq 16/13$.*

Proof. Let $\sigma = \mathcal{A}\mathcal{B}(x_1, y_1)\,\mathcal{B}(x_2, y_2)\,\ldots\,\mathcal{B}(x_n, y_n)$. This determines the sequence $\gamma_1, \gamma_2, \ldots, \gamma_n$. Since σ is non-trivial, both OPT and GREEDY transmit packets during all $\sum_{i=1}^{n} x_i$ IDLE requests of σ. Afterwards, $\boldsymbol{x}^{\mathrm{GR}} = (B - \gamma_n, B)$ and OPT has at most $2 \cdot B$ packets; these packets are transmitted at the end of σ. Therefore, the reciprocal of the performance ratio on σ is

$$\frac{1}{\mathcal{R}(\sigma)} = \frac{(\sum_{i=1}^{n} x_i) + B + (B - \gamma_n)}{(\sum_{i=1}^{n} x_i) + B + B} = 1 - \frac{\gamma_n/B}{2 + \sum_{i=1}^{n} x_i/B} \geq 1 - \frac{\gamma_n/B}{2 + s(\gamma_n/B)} \geq 1 - \frac{3}{16} = \frac{13}{16}\ ,$$

where the last two inequalities follow by Lemma 12 and Lemma 11, respectively. □

We note that (6) (and thus Thm. 1) follows directly from combining Lemmas 4, 6, 7, 8, and 13.

4 Randomization

In this section, we describe a randomized algorithm PB, which works in a standard model and whose expected loss on a sequence σ is exactly the same as the loss of PBF on σ in the fractional model.

PB traces the current state of PBF. In case of IDLE requests, PB tries to transmit a packet in such a way, that the expected number of packets in its buffers is equal to the actual number of packets in the buffers of PBF. In the appendix we show an argument why a straightforward randomization of the fractional solution is inappropriate. We define PB algorithm implicitly, i.e., in each step we show what its probability distribution over possible states should be. First, we define this distribution for any possible state of PBF. Later, we show how to infer the actual behavior of the algorithm on the basis of the distribution.

Definition 2. *Fix any (fractional) state of the buffers $\boldsymbol{x} = (k_0 + a, k_1 + b)$, where $k_0, k_1 \in \mathbb{N}$ and $a, b \in [0, 1]$. Let $\mu(\boldsymbol{x})$ be a random variable with the following distribution.*

– *If $a + b \leq 1$, then*

$$\mu(\boldsymbol{x}) = (k_0, k_1) + \begin{cases} (1,0) & \text{with probability } a \\ (0,1) & \text{with probability } b \\ (0,0) & \text{with probability } 1 - a - b \end{cases}$$

– *If* $a + b \geq 1$, *then*

$$\mu(\boldsymbol{x}) = (k_0, k_1) + \begin{cases} (1,0) & \text{with probability } 1-b \\ (0,1) & \text{with probability } 1-a \\ (1,1) & \text{with probability } a+b-1 \end{cases}$$

We observe that $\mathbf{E}[\mu(\boldsymbol{x})] = \boldsymbol{x}$. We note that this definition is purposely made ambiguous. In particular, it can be easily verified that if a number of packets in the first buffer is an integer, then the resulting value of μ is the same, no matter whether we pick $a = 0$ or $a = 1$. The same holds for the second buffer. Also, if $a + b = 1$, we may choose any of the two rules above for setting $\mu(\boldsymbol{x})$ above, and both would yield the same distribution. For more intuitions about function μ, see Fig. 5. Each point from the square represents a legal state of an algorithm in the fractional model. Legal states of the algorithm in the standard model are represented by dots (points with integer coordinates). Then $\mu(\boldsymbol{x})$ is just a function which assigns probabilities to the vertices of a triangle enclosing $\boldsymbol{x}$. If $\boldsymbol{x}$ lies on a triangle edge or at a triangle vertex, then $\mu(\boldsymbol{x})$ has non-zero probability on two points (ends of the edge) or at one point (the vertex), respectively.

The following lemma shows that whenever PBF changes is state from $\boldsymbol{x}$ to $\boldsymbol{x}'$, it is possible for PB to change the probability distribution from $\mu(\boldsymbol{x})$ to $\mu(\boldsymbol{x}')$.

Lemma 14. *It is possible to construct a randomized online algorithm* PB *for the standard model, such that* $\boldsymbol{x}^{\mathrm{PB}}(\sigma) = \mu(\boldsymbol{x}^{\mathrm{PBF}}(\sigma))$ *for any sequence* σ. *In effect,* $\mathsf{loss}_{\mathrm{PB}}(\sigma) = \mathsf{loss}_{\mathrm{PBF}}(\sigma)$.

Theorem 3. PB *is* $\frac{16}{13}$*-competitive for the packet buffering problem on two buffers.*

5 Conclusions

Although we presented an optimal algorithm for a specific case of two buffers, its main idea of tracing set $\mathcal{I}$ and trying to stay close to its center can be generalized; the game is then played in a m-dimensional cube $\mathcal{H}$. The main open question is whether such algorithms can approach the lower bound of $h(m)$.

References

1. S. Albers and M. Schmidt. On the performance of greedy algorithms in packet buffering. In *Proc. of the 36th ACM Symp. on Theory of Computing (STOC)*, pages 35–44, 2004.
2. Y. Azar and Y. Richter. Management of multi-queue switches in QoS networks. In *Proc. of the 35th ACM Symp. on Theory of Computing (STOC)*, pages 82–89, 2003.
3. M. Chrobak, L. L. Larmore, N. Reingold, and J. Westbrook. Page migration algorithms using work functions. *Journal of Algorithms*, 24(1):124–157, 1997. Also appeared in *Proc. of the 4th ISAAC*, pages 406–415, 1993.
4. M. Englert and M. Westermann. Lower and upper bounds on FIFO buffer management in QoS switches. In *Proc. of the 14th European Symp. on Algorithms (ESA)*, pages 352–363, 2006.
5. 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.
6. A. Kesselman, Y. Mansour, and R. van Stee. Improved competitive guarantees for QoS buffering. *Algorithmica*, 43(1–2):63–80, 2005. Also appeared in *Proc. of the 11th ESA*, pages 361–372, 2003.
7. E. Koutsoupias and C. H. Papadimitriou. On the k-server conjecture. *Journal of the ACM*, 42(5):971–983, 1995. Also appeared in *Proc. of the 26th STOC*, pages 507–511, 1994.
8. W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.
9. M. May, J. Bolot, A. Jean-Marie, and C. Diot. Simple performance models of differentiated services schemes for the internet. In *Proc. of the IEEE INFOCOM*, pages 1385–1394, 1999.

10. P. Raghavan and C. D. Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.
11. M. Schmidt. Packet buffering: Randomization beats deterministic algorithms. In *Proc. of the 22nd Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 293–304, 2005.
12. M. Schmidt. *Online Packet Buffering.* PhD thesis, Albert-Ludwigs-Universität Freiburg, 2006.
13. D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

A Proofs of Technical Lemmas

A.1 The Set $\mathcal{I}$

Proof (of Lemma 1). We prove the lemma inductively. At the very beginning $\boldsymbol{x}^{\mathrm{ALG}} = (0,0) \in \mathcal{I} \subseteq \mathcal{SH}(\mathcal{I})$. Assume that this lemma holds for sequence σ^{t-1}. It means that there exists a state $(k_0, k_1) \in \mathcal{I}(\sigma^{t-1})$, which majorizes $\boldsymbol{x}^{\mathrm{ALG}}(\sigma^{t-1})$.

- If $\sigma_t = \textsc{Idle}$, then we consider two cases. If $\boldsymbol{x}^{\mathrm{ALG}}(\sigma^{t-1}) = (0,0)$, then Alg remains in this state and the lemma trivially holds. Otherwise Alg transmits a packet from a non-empty buffer, say from buffer 0, which implies $k_0 \geq 1$. But $(k_0 - 1, k_1) \in \mathcal{I}(\sigma^t)$, and this state majorizes $\boldsymbol{x}^{\mathrm{ALG}}(\sigma^{t-1}) - (1,0) = \boldsymbol{x}^{\mathrm{ALG}}(\sigma^t)$.
- If $\sigma_t = \textsc{Add}(0)$, then we consider three subcases. Analogous reasoning applies if $\sigma_t = \textsc{Add}(1)$.
  - $k_0 < B$. In this case, $\boldsymbol{x}^{\mathrm{ALG}}(\sigma^t) \leq \boldsymbol{x}^{\mathrm{ALG}}(\sigma^{t-1}) + (1,0) \leq (k_0+1, k_1) \in \mathcal{I}(\sigma^t)$.
  - $k_0 = B$ and $\mathcal{I}(\sigma^{t-1})$ is a singleton set. In this case $\mathcal{I}(\sigma^t) = \mathcal{I}(\sigma^{t-1})$. We get $x_0^{\mathrm{ALG}}(\sigma^t) \leq B = k_0$ and $x_1^{\mathrm{ALG}}(\sigma^t) = x_1^{\mathrm{ALG}}(\sigma^{t-1}) \leq k_1$, and therefore $\boldsymbol{x}^{\mathrm{ALG}}(\sigma^t) \leq (k_0, k_1) \in \mathcal{I}(\sigma^t)$.
  - $k_0 = B$ and $\mathcal{I}(\sigma^{t-1})$ is not a singleton set. By the construction of $\mathcal{I}$, $\mathcal{I}(\sigma^{t-1})$ contains also (k_0-1, k_1+1), and therefore $\mathcal{I}(\sigma^t)$ contains (k_0, k_1+1). Similarly to the previous subcase, $x_0^{\mathrm{ALG}}(\sigma^t) \leq B = k_0$, $x_1^{\mathrm{ALG}}(\sigma^t) = x_1^{\mathrm{ALG}}(\sigma^{t-1}) \leq k_1 < k_1 + 1$, and thus $\boldsymbol{x}^{\mathrm{ALG}}(\sigma^t) \leq (k_0, k_1+1)$.

Hence, in either case the lemma holds for σ^t. □

Proof (of Lemma 2). In order to prove the first part of the lemma, we show that for each step $t \geq 1$ and $\boldsymbol{x} \in \mathcal{I}(\sigma^t)$, there exists a state $\boldsymbol{x}_{\mathrm{prev}} \in \mathcal{I}(\sigma^{t-1})$, such that after request σ_t and a proper choice of packet transmitted, $\boldsymbol{x}_{\mathrm{prev}}$ becomes $\boldsymbol{x}$. Thus, if we choose any sequence σ and any final state from $\mathcal{I}(\sigma)$, we can reconstruct the sequence of states of an algorithm A, which remains for the whole sequence σ in the corresponding $\mathcal{I}$ set.

We introduce a notion of ℓ^* which is the same measure for set $\mathcal{I}$ as ℓ^{ALG} is for an algorithm Alg, i.e., $\ell^*(\sigma^t)$ is the number of packets added during an Add request in step t to a state from $\mathcal{I}$. For an Idle request in step t, $\ell^*(\sigma^t) = 0$. This means that for an $\textsc{Add}(0)$ request in step t, $\mathcal{I}(\sigma^t) = \mathcal{I}(\sigma^{t-1}) + (\ell^*, 0)$ and for $\textsc{Add}(1)$, $\mathcal{I}(\sigma^t) = \mathcal{I}(\sigma^{t-1}) + (0, \ell^*)$.

If $\sigma_t = \textsc{Idle}$, then it follows from the definition of $\mathcal{I}(\sigma^t)$ that either $\boldsymbol{x} + (1,0)$, $\boldsymbol{x} + (0,1)$ or $\boldsymbol{x}$ itself belongs to $\mathcal{I}(\sigma^{t-1})$. Choosing $\boldsymbol{x}_{\mathrm{prev}}$ as such element of $\mathcal{I}(\sigma^{t-1})$ guarantees that $\boldsymbol{x}$ can be reached from $\boldsymbol{x}_{\mathrm{prev}}$ by transmitting packet from an appropriate buffer or not transmitting at all. On the other hand, if $\sigma_t = \textsc{Add}(0)$, then we choose $\boldsymbol{x}_{\mathrm{prev}} = \boldsymbol{x} - (\ell^*, 0)$. By the observation above, $\boldsymbol{x}_{\mathrm{prev}} \in \mathcal{I}(\sigma^{t-1})$. The case of $\sigma_t = \textsc{Add}(1)$ is analogous.

Moreover, we observe that for all the requests, it holds that $\ell^A \equiv \ell^*$, which means that A loses a packet during an Add request which does not change the set $\mathcal{I}$.[4]

For proving the second part of the lemma, we fix any algorithm A, which is always in the set $\mathcal{I}$. Note that $\|\boldsymbol{x}^A\| \equiv \|\mathcal{I}\|$. For proving the optimality of A, we show that for any algorithm

[4] It may only happen if $\mathcal{I}$ is a singleton set.

ALG and any input sequence σ, $\mathsf{loss}^{\mathrm{ALG}}(\sigma) \geq \mathsf{loss}^{A}(\sigma)$. Without loss of generality, we may assume that ALG is work-conserving.

When we look at the number of packets lost by ALG in the consecutive steps, we notice that the fewer packets it has, the harder it is to force it to lose another packets. Thus, one may try a strategy of losing more packets than strictly necessary at some step, and then winning back not only the surplus lost, but also some additional ones. We show that no such strategy is feasible.

We inductively prove that for any input σ and step $0 \leq t \leq |\sigma|$, it holds that

$$\|\boldsymbol{x}^{\mathrm{ALG}}(\sigma^t)\| + \mathsf{loss}_{\mathrm{ALG}}(\sigma^t) \geq \|\boldsymbol{x}^{A}(\sigma^t)\| + \mathsf{loss}_{A}(\sigma^t) \ . \tag{11}$$

Obviously, it holds for $t = 0$. For an ADD requests, both sides of inequality increase by 1, i.e., either the ADD request incurs a loss to an algorithm or shifts its position in appropriate direction. For an IDLE request, there is no loss generated. By Lemma 1, $\|\boldsymbol{x}^{\mathrm{ALG}}\| \leq \|\boldsymbol{x}^{A}\|$, and therefore if ALG transmits a packet, A does this as well. Hence, (11) holds.

Finally, $\|\boldsymbol{x}^{\mathrm{ALG}}\| \leq \|\boldsymbol{x}^{A}\|$ together with (11) implies $\mathsf{loss}_{\mathrm{ALG}}(\sigma) \geq \mathsf{loss}_{A}(\sigma)$, which finishes the proof. □

A.2 Non-trivial Sequences

Proof (of Lemma 3). It is sufficient to show that the balance is at most $\max\{\|\mathcal{I}\| - B, 0\}$. Initially, this condition is trivially fulfilled. Assume that this condition holds at the end of step t.

If $\sigma_{t+1} = \textsc{Add}(i)$, then the balance changes by 1 only if this $\textsc{Add}(i)$ is a hit, in which case $\mathcal{I}$ has to be above or at MD. As a result, $\max\{\|\mathcal{I}\| - B, 0\}$ increases as well.

If $\sigma_{t+1} = \textsc{Idle}$, then the balance may increase only if $\mathcal{I}$ is below or at MD and touches exactly one lower boundary. In this case, the PBF decreases the balance back in the second part of the step. □

Proof (of Lemma 4). First, we show that we may cut off our sequence σ after the last ADD request which incurs a loss to PBF. Indeed, if σ does not end with ADD request inflicting loss to PBF, then let $\sigma' = \sigma^{|\sigma|-1}$. We have $S(\sigma') \leq S(\sigma)$, $\mathsf{loss}_{\mathrm{OPT}}(\sigma') \leq \mathsf{loss}_{\mathrm{OPT}}(\sigma)$, and $\mathsf{loss}_{\mathrm{PBF}}(\sigma') = \mathsf{loss}_{\mathrm{PBF}}(\sigma)$. Therefore, $\mathcal{R}_{\mathrm{PBF}}(\sigma') \geq \mathcal{R}_{\mathrm{PBF}}(\sigma)$. By proceeding inductively, we obtain a sequence ending with an ADD request which incurs a loss to PBF.

For proving the second part of the non-triviality property, we show that if σ contains a step t, such that $\boldsymbol{x}^{\mathrm{PBF}}(\sigma^t) = (0, 0)$, then there exists a strictly shorter sequence $\widetilde{\sigma}$, such that $\mathcal{R}_{\mathrm{PBF}}(\widetilde{\sigma}) \geq \mathcal{R}_{\mathrm{PBF}}(\sigma)$. If a newly obtained sequence $\widetilde{\sigma}$ is not non-trivial, then we may apply this scheme inductively on $\widetilde{\sigma}$. After finitely many steps, we get a non-trivial sequence $\widehat{\sigma}$, such that $\mathcal{R}_{\mathrm{PBF}}(\widehat{\sigma}) \geq \mathcal{R}_{\mathrm{PBF}}(\sigma)$.

To show this, we take the last step t for which $\boldsymbol{x}^{\mathrm{PBF}}(\sigma^t) = (0, 0)$. By a straightforward induction, we observe that it may only happen when $\mathcal{I}(\sigma^t)$ touches both lower boundaries. Let $L = \|\mathcal{I}(\sigma^t)\|$. Our goal is now to show that if we insert L IDLE requests directly after step t, then we may change the remaining suffix of σ, so that the performance ratio of PBF does not decrease. Sequences $\tau = \sigma^t$ and $\lambda = \sigma_{t+1}^{|\sigma|}$ fulfill the requirements of Lemma 15 (see below). By applying this lemma L times, we obtain a sequence $\sigma^t\,\textsc{Idle}^L\,\eta$, such that $\mathcal{R}_{\mathrm{PBF}}(\sigma^t\,\textsc{Idle}^L\,\eta) \geq \mathcal{R}_{\mathrm{PBF}}(\sigma)$ and $|\eta| = |\lambda|$. Moreover, $\boldsymbol{x}^{\mathrm{PBF}}(\sigma^t\,\textsc{Idle}^L) = (0, 0)$ and $\mathcal{I}(\sigma^t\,\textsc{Idle}^L) = \{(0, 0)\}$, which means that after the first $t + L$ steps the game between PBF and the adversary starts over. In effect, $\mathcal{R}_{\mathrm{PBF}}(\sigma^t\,\textsc{Idle}^L\,\eta) \leq \max\{\mathcal{R}_{\mathrm{PBF}}(\sigma^t), \mathcal{R}_{\mathrm{PBF}}(\eta)\}$. We pick $\widetilde{\sigma}$ to be either σ^t or η, choosing the one with higher performance ratio. Finally we note that for any such choice $|\widetilde{\sigma}| < |\sigma|$, and thus the lemma follows. □

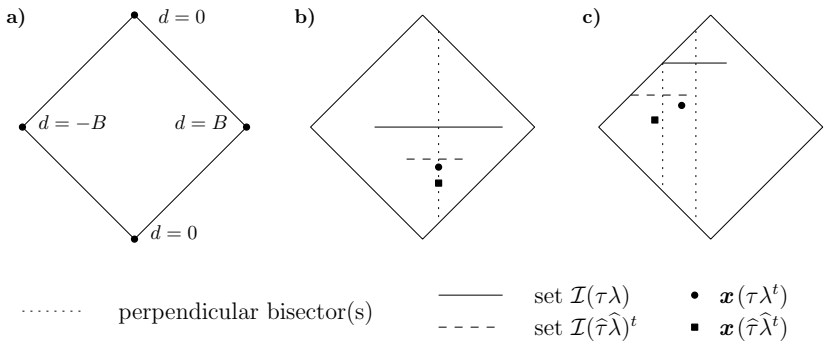

Fig. 6. Types of steps defined in the proof of Lemma 15

Lemma 15. *Fix any sequence τ, such that $\boldsymbol{x}^{\mathrm{PBF}}(\tau) = (0,0)$, $1 \le \|\mathcal{I}(\tau)\| \le B$, and $\mathcal{I}(\tau)$ touches both lower boundaries. Let $\widehat{\tau} = \tau\,\textsc{Idle}$. Fix any sequence λ, such that $\boldsymbol{x}^{\mathrm{PBF}}(\tau\lambda^t) \neq (0,0)$ for any $t = 1, \ldots, |\lambda|$. There exists a sequence $\widehat{\lambda}$, such that $|\widehat{\lambda}| = |\lambda|$ and $\mathcal{R}_{\mathrm{PBF}}(\widehat{\tau}\widehat{\lambda}) \ge \mathcal{R}_{\mathrm{PBF}}(\tau\lambda)$.*

Proof. Note that values of S, $\mathsf{loss}_{\mathrm{OPT}}$ and $\mathsf{loss}_{\mathrm{PBF}}$ are the same on τ and on $\widehat{\tau}$. In the following, we omit subscripts and superscripts at $\boldsymbol{x}$.

We create sequence $\widehat{\lambda}$ iteratively on the basis of λ, thinking that λ is created by an adversary. We refer to λ as the *original* sequence and to $\widehat{\lambda}$ as the *modified* one. To simplify the analysis later, we want to make $\widehat{\lambda}$ quite similar to λ. In particular, $\widehat{\lambda}$ will have the same length as λ and will inject packets in exactly same steps. That said, $\widehat{\lambda}$ may inject packets to different buffers than λ.

Additionally, we employ the following *swapping operation*. As at the beginning $\mathcal{I}(\tau)$ and $\mathcal{I}(\widehat{\tau})$ touch both lower boundaries and $\boldsymbol{x}(\tau) = \boldsymbol{x}(\widehat{\tau}) = (0,0)$, the buffers are completely symmetric. This means that at step t we may replace all $\textsc{Add}(0)$ requests in $\lambda|_1^t$ by $\textsc{Add}(1)$ and vice versa. Note that this does not restrict the power of the adversary of creating an arbitrary sequence: we may think that upon swapping operation the adversary performs the same replacement in the not yet processed part of λ. This way, we merely swapped the meaning of two buffers in the original sequence.

We define a couple of general invariants.

(I1) $\mathsf{loss}_{\mathrm{OPT}}(\tau\lambda^t) = \mathsf{loss}_{\mathrm{OPT}}(\widehat{\tau}\widehat{\lambda}^t)$;
(I2) $\|\mathcal{I}(\tau\lambda^t)\| = \|\mathcal{I}(\widehat{\tau}\widehat{\lambda}^t)\| + 1$.

Additionally, we divide steps into two different types and define invariants for each type. To this end, we first introduce a few notions. By $\mathsf{pb}(\mathcal{I})$ we denote the perpendicular bisector of the smallest interval containing set $\mathcal{I}$. For any point of $\boldsymbol{y} \in \mathcal{H}$, we define its *horizontal position* as $d(\boldsymbol{y}) = y_1 - y_0$. Thus, the horizontal position of the leftmost point of $\mathcal{H}$ is $-B$ and the rightmost one is B, cf. Fig. 6a. Note that points from the same vertical lines of $\mathcal{H}$ (in particular from the $\mathsf{pb}(\mathcal{I})$) have the same horizontal positions. We call a difference between the values of d of two points the *horizontal distance* between these two points.
A T1-type step t is depicted in Fig. 6b. Its invariants are:

(P1a) $\mathsf{len}(\mathcal{I}(\tau\lambda^t)) = \mathsf{len}(\mathcal{I}(\widehat{\tau}\widehat{\lambda}^t)) + 1$;
(P1b) $\mathsf{pb}(\mathcal{I}(\tau\lambda^t)) = \mathsf{pb}(\mathcal{I}(\widehat{\tau}\widehat{\lambda}^t))$;
(P1c) $\|\boldsymbol{x}(\widehat{\tau}\widehat{\lambda}^t)\| \le \|\boldsymbol{x}(\tau\lambda^t)\| \le \|I(\widehat{\tau}\widehat{\lambda}^t)\|$;
(P1d) $\boldsymbol{x}(\tau\lambda^t)$ and $\boldsymbol{x}(\widehat{\tau}\widehat{\lambda}^t)$ lie on $\mathsf{pb}(\mathcal{I}(\tau\lambda^t)) = \mathsf{pb}(\mathcal{I}(\widehat{\tau}\widehat{\lambda}^t))$.

A T2-type step t is depicted in Fig. 6c. Its invariants are:

(P2a) $\mathsf{len}(\mathcal{I}(\tau\lambda^t)) = \mathsf{len}(\mathcal{I}(\widehat{\tau}\widehat{\lambda}^t))$;
(P2b) either $\mathcal{I}(\tau\lambda^t) = \mathcal{I}(\widehat{\tau}\widehat{\lambda}^t) + (0,1)$ and both $\mathcal{I}(\tau\lambda^t)$ and $\mathcal{I}(\widehat{\tau}\widehat{\lambda}^t)$ touch the upper 0-boundary (we call $\mathcal{I}(\tau\lambda^t)$ *right-shifted*),
or $\mathcal{I}(\tau\lambda^t) = \mathcal{I}(\widehat{\tau}\widehat{\lambda}^t) + (1,0)$ and both $\mathcal{I}(\tau\lambda^t)$ and $\mathcal{I}(\widehat{\tau}\widehat{\lambda}^t)$ touch the upper 1-boundary (we call $\mathcal{I}(\tau\lambda^t)$ is *left-shifted*);
(P2c) $\|\boldsymbol{x}(\widehat{\tau}\widehat{\lambda}^t)\| \leq \|\boldsymbol{x}(\tau\lambda^t)\|$;
(P2d) $d(\boldsymbol{x}(\widehat{\tau}\widehat{\lambda}^t)) \leq d(\boldsymbol{x}(\tau\lambda^t))$;
(P2e) $d(\boldsymbol{x}(\widehat{\tau}\widehat{\lambda}^t)) \leq d(\mathsf{pb}(\mathcal{I}(\widehat{\tau}\widehat{\lambda}^t)))$;
(P2f) $d(\boldsymbol{x}(\tau\lambda^t)) \leq d(\mathsf{pb}(\mathcal{I}(\tau\lambda^t)))$.

The inequalities in invariants (P2d)–(P2f) hold for a right-shifted $\mathcal{I}(\tau\lambda^t)$. For a left-shifted one, they are reversed.

We inductively show that the invariants hold. Clearly, step $t = 0$ is of type T1: all invariants are trivially preserved as $\boldsymbol{x}(\tau) = \boldsymbol{x}(\widehat{\tau}) = (0,0)$. We assume that the invariants hold for step $t < T$ and we show them for step $t+1$. We refrain for formally showing invariants related solely to sets $\mathcal{I}$ (i.e., (I1), (I2), (P1a), (P1b), (P2a), (P2b)) as they will follow immediately from our construction. Unless specified otherwise, we choose $\widehat{\lambda}_{t+1} = \lambda_{t+1}$,

Preserving invariants when t is of type T1.

1. $\lambda_{t+1} = \textsc{Add}$ and this $\textsc{Add}$ is not a hit in sequence $\tau\lambda^{t+1}$. In this case, this $\textsc{Add}$ is also not a hit in sequence $\widehat{\tau}\widehat{\lambda}^{t+1}$. By invariants (P1c) and (P1d), no loss is incurred to any of the two instances of PBF. Hence, this $\textsc{Add}$ just shifts both sets $\mathcal{I}$ and points $\boldsymbol{x}$ and all invariants are preserved.
2. $\lambda_{t+1} = \textsc{Add}$ and this $\textsc{Add}$ is a hit in sequence $\tau\lambda^{t+1}$. We show that $t+1$ is of type T2. As there is no hit for the sequence $\widehat{\tau}\widehat{\lambda}^{t+1}$, invariants (P1c) and (P1d) guarantee no loss for both instances of PBF. This implies invariant (P2c). As $d(\boldsymbol{x}(\tau\lambda^{t+1})) = d(\boldsymbol{x}(\widehat{\tau}\widehat{\lambda}^{t+1})) = d(\mathsf{pb}(\mathcal{I}(\widehat{\tau}\widehat{\lambda}^{t+1}))) = d(\mathsf{pb}(\mathcal{I}(\tau\lambda^{t+1}))) \pm 1$, invariants (P2d)–(P2f) follow.
3. $\lambda_{t+1} = \textsc{Idle}$. By the lemma assumption, $\|\boldsymbol{x}(\tau\lambda^{t+1})\| \geq 0$. In this case, $\|\mathcal{I}\|$ and the number of packets in the buffers of PBF run on the original sequence decrease by 1. The number of packets in the buffers of PBF run on the modified sequence may decrease by a smaller amount if $\|\boldsymbol{x}(\widehat{\tau}\widehat{\lambda}^{t+1})\| = 0$. In either case, invariant (P1c) follows. Invariant (P1d) follows trivially.

Preserving invariants when t is of type T2.

Without loss of generality, we assume that at step t, $\mathcal{I}(\tau\lambda^t)$ is *right-shifted*, i.e., $\mathcal{I}(\tau\lambda^t) = \mathcal{I}(\widehat{\tau}\widehat{\lambda}^t) + (0,1)$. Recall that this means that both $\mathcal{I}(\tau\lambda^t)$ and $\mathcal{I}(\widehat{\tau}\widehat{\lambda}^t)$ touch upper 0-boundary and the horizontal distance between their pbs is exactly 1.

1. $\lambda_{t+1} = \textsc{Idle}$. If $\mathcal{I}(\widehat{\tau}\widehat{\lambda}^t)$ lies strictly above the main diagonal, then pbs remain intact. As both instances of PBF are trying to reduce their distance to the respective pbs, whose mutual horizontal distance is 1, invariants (P2d)–(P2f) are preserved. Invariant (P2c) follows trivially. If, however, $\mathcal{I}(\widehat{\tau}\widehat{\lambda}^t)$ lies on the main diagonal, then step $t+1$ is of type T1, where invariant (P1c) follows trivially and invariant (P1d) follows by Lemma 3.
2. $\lambda_{t+1} = \textsc{Add}(0)$. Such request is clearly a hit. If both $\mathcal{I}$ are points on the boundary then this incurs a loss of a packet for $\textsc{Opt}$ in both instances. The pbs remain in place, while $\boldsymbol{x}$ corresponding to PBF instances may be shifted. This preserves invariants (P2d)–(P2f). Invariant (P2c) of step t states that there exists $c \geq 0$ such that $\|\boldsymbol{x}(\widehat{\tau}\widehat{\lambda}^t)\| + c = \|\boldsymbol{x}(\tau\lambda^t)\|$. On the other hand, invariant (P2d) of step t states $x_1(\widehat{\tau}\widehat{\lambda}^t) - x_0(\widehat{\tau}\widehat{\lambda}^t) \leq x_1(\tau\lambda^t) - x_0(\tau\lambda^t)$. Therefore, $x_0(\widehat{\tau}\widehat{\lambda}^t) \geq x_0(\tau\lambda^t) - c/2$. By adding a packets to the buffer 0 in step $t+1$, the loss incurred to the modified instance can be at most $c/2$ smaller than the loss incurred to the original one. Therefore, $\|\boldsymbol{x}(\widehat{\tau}\widehat{\lambda}^{t+1})\| + c/2 \leq \|\boldsymbol{x}(\tau\lambda^{t+1})\|$, which yields invariant (P2c).

3. $\lambda_{t+1} = \textsc{Add}(1)$ and $\mathcal{I}(\tau\lambda^t)$ does not touch the upper 1-boundary. As this request is not a hit, sets $\mathcal{I}$ and points $\boldsymbol{x}$ are just shifted and all invariants hold.
4. $\lambda_{t+1} = \textsc{Add}(1)$ and $\mathcal{I}(\tau\lambda^t)$ touches the upper 1-boundary. Note that in such case $d(\mathsf{pb}(\mathcal{I}(\tau\lambda))) = 0$ and $d(\mathsf{pb}(\mathcal{I}(\widehat{\tau}\widehat{\lambda}))) = -1$. We consider two subcases.
   (a) If $d(\boldsymbol{x}(\tau\lambda_t)) \leq -1$, then we set $\widehat{\lambda}^{t+1} = \textsc{Add}(0)$. For such choice, both pbs remain intact, and invariants (P2d)–(P2f) follow. Furthermore, this step incurs no loss on PBF on the original sequence, and therefore invariant (P2c) follows as well. Note that this step may incur a loss of $\textsc{Opt}$ on the modified sequence in case when set $\mathcal{I}(\widehat{\tau}\widehat{\lambda}^t)$ contains a single point (on the upper boundary). However, in such case $\mathcal{I}(\tau\lambda)$ consists of a single point $\{(B, B)\}$ and λ_{t+1} incurs a loss on $\textsc{Opt}$ on the original sequence, too. Hence invariant (I1) follows.
   (b) If $d(\boldsymbol{x}(\tau\lambda^t)) \in (-1, 0]$, then we perform swapping operation. Note that the operation does not change the shape of $\mathcal{I}(\tau\lambda^t)$ as this set is buffer symmetric. It does however change the position of $\boldsymbol{x}(\tau\lambda^t)$, so that $d(\boldsymbol{x}(\tau\lambda^t)) \in [0, 1)$ and $\lambda_{t+1} = \textsc{Add}(0)$. In particular, after performing this request, $d(\boldsymbol{x}(\tau\lambda_t)) < 0$, and thus invariant (P2f) is preserved. In this case, we set $\widehat{\lambda}^{t+1} = \textsc{Add}(0)$ and apply the analysis from case 2.

After the input sequence ends, we analyze the performance ratio on the original and the modified sequence. Clearly, $S(\widehat{\tau}\widehat{\lambda}) = S(\tau\lambda)$. Within λ, PBF run on the original instance of PBF transmitted load 1 for any $\textsc{Idle}$ request and when run on the modified instance transmitted load at most 1. Hence, by a simple induction we obtain the following relation: $\|\boldsymbol{x}(\tau\lambda)\| - \|\boldsymbol{x}(\tau)\| + \Delta_\tau \mathsf{loss}_{\mathrm{PBF}}(\lambda) \leq \|\boldsymbol{x}(\widehat{\tau}\widehat{\lambda})\| - \|x(\widehat{\tau})\| + \Delta_{\widehat{\tau}} \mathsf{loss}_{\mathrm{PBF}}(\widehat{\lambda})$. As $\mathsf{loss}_{\mathrm{PBF}}(\tau) = \mathsf{loss}_{\mathrm{PBF}}(\widehat{\tau})$ and $\|\boldsymbol{x}(\tau)\| = \|\boldsymbol{x}(\widehat{\tau})\|$, it holds that $\mathsf{loss}_{\mathrm{PBF}}(\widehat{\tau}\widehat{\lambda}) - \mathsf{loss}_{\mathrm{PBF}}(\tau\lambda) \geq \|\boldsymbol{x}(\tau\lambda)\| - \|\boldsymbol{x}(\widehat{\tau}\widehat{\lambda})\|$.

If the sequence ends with a step of type T1, then $\mathsf{loss}_{\mathrm{OPT}}(\widehat{\tau}\widehat{\lambda}) = \mathsf{loss}_{\mathrm{OPT}}(\tau\lambda)$ (by invariant (I1)) and $\mathsf{loss}_{\mathrm{PBF}}(\widehat{\tau}\widehat{\lambda}) \geq \mathsf{loss}_{\mathrm{PBF}}(\tau\lambda)$ (by invariant (P1c)). Same relations hold if a sequence ends with a step of type T2 (by (I1) and (P2c)). Therefore, $\mathcal{R}(\widehat{\tau}\widehat{\lambda}) \geq \mathcal{R}(\tau\lambda)$. □

A.3 Properties of Function s()

Proof (of Lemma 10). We fix any $0 \leq a < b \leq 1$. Let i be an integer, such that $1 - 2^{-i} \leq a < 1 - 2^{-(i+1)}$. Note that for any $b > a$, it holds that $\frac{a+b}{2} < 1 - 2^{-(i+2)}$. We consider two cases.

(i) If $\frac{a+b}{2} < 1 - 2^{-(i+1)}$, then both a and $\frac{a+b}{2}$ belong to the same linear segment of the function s. Then,

$$\begin{aligned} s\left(\frac{a+b}{2}\right) - s(a) &= \left(i + \frac{\frac{a+b}{2} - (1 - 2^{-i})}{2^{-(i+1)}}\right) - \left(i + \frac{a - (1 - 2^{-i})}{2^{-(i+1)}}\right) \\ &= 2^{i+1} \cdot \left(\frac{a+b}{2} - a\right) \\ &= 2^i \cdot (b - a) \\ &\leq 2^i \cdot \left[b - (1 - 2^{-i})\right] \\ &= (2^i - 1) \cdot (b - 1) + b \\ &\leq b \ . \end{aligned}$$

(ii) If $1-2^{-(i+1)} \le \frac{a+b}{2} < 1-2^{-(i+2)}$, then a and $\frac{a+b}{2}$ belong to the two consecutive linear fragments of the function s. In this case,

$$\begin{aligned} s\left(\frac{a+b}{2}\right) - s(a) &= \left((i+1) + \frac{\frac{a+b}{2} - (1-2^{-(i+1)})}{2^{-(i+2)}}\right) - \left(i + \frac{a-(1-2^{-i})}{2^{-(i+1)}}\right) \\ &= 1 + \frac{(a+b)-(2-2^{-i})}{2^{-(i+1)}} - \frac{a-(1-2^{-i})}{2^{-(i+1)}} \\ &= 1 + 2^{i+1}\cdot(b-1) \\ &= (2^{i+1}-1)\cdot(b-1)+b \\ &\le b \ . \end{aligned}$$

Thus, in both cases it holds that $s(\frac{a+b}{2}) - s(a) \le b$. □

Proof (of Lemma 11). Let $g(a) = \frac{16}{3}\cdot a - 2$. It suffices to show that $s(a) \ge g(a)$ for any $a \in [0,1]$. The plot of both functions is depicted in Fig. 4, but the relation between $s(a)$ and $g(a)$ can be also proved analytically.

For $a \in [0, \frac{1}{2})$, $s(a) = 2a$, and for $a \in [\frac{1}{2}, \frac{3}{4})$, $s(a) = 4a - 1$. It can be checked that for these cases $s(a) \ge g(a)$. On the other hand, $s(\frac{3}{4}) = g(\frac{3}{4})$ and for $a \ge \frac{3}{4}$, $s(a)$ is growing faster than $g(a)$. To show this, it is sufficient to observe than the function s is continuous, differentiable on the whole $(0,1)$ except for the points $1-2^{-i}$, and its first derivative on $(\frac{3}{4}, 1)$ is at least 4, which is more than $\frac{16}{3}$, the first derivative of $g(a)$. □

A.4 Randomization

Proof (of Lemma 14). We prove the lemma by induction on the number of steps. At the beginning, the invariant is trivially fulfilled, as both algorithms start with empty buffers. Assume that $\boldsymbol{x}^{\mathrm{PB}}(\sigma^{t-1}) = \mu(\boldsymbol{x}^{\mathrm{PBF}}(\sigma^{t-1}))$. Fix $k_0, k_1 \in \mathbb{N}$, $a, b \in [0,1]$, such that $\boldsymbol{x}^{\mathrm{PBF}}(\sigma^{t-1}) = (k_0+a, k_1+b)$. To keep the description of a random variable concise, we introduce the following notation. We write $\boldsymbol{x}^{\mathrm{PB}} = \{(t_1, s_1) : p_1, (t_2, s_2) : p_2, \ldots\}$ meaning that $\boldsymbol{x}^{\mathrm{PB}}$ is equal to (t_i, s_i) with probability p_i. Furthermore, we call a triangle corresponding to the case $a+b \ge 1$ *upper*; the other ones are called *lower*.

Assume that $\sigma_t = \textsc{Add}(0)$ (for $\sigma_t = \textsc{Add}(1)$, the reasoning is analogous). If $\boldsymbol{x}_1^{\mathrm{PBF}}(\sigma^{t-1}) \le B-1$, then adding a packet to the buffer of PBF and PB just changes the triangle, without violating the relations between $\boldsymbol{x}^{\mathrm{PBF}}$ and $\boldsymbol{x}^{\mathrm{PB}}$. The only problem may occur if $x_1^{\mathrm{PBF}}(\sigma^{t-1}) > B-1$. So let $k_0 = B-1$ and assume first that $\boldsymbol{x}^{\mathrm{PBF}}(\sigma^{t-1}) = (B-1+a, k_1+b)$ lies in an upper triangle. Hence, $\boldsymbol{x}^{\mathrm{PB}}(\sigma^{t-1}) = \{(B, k_1) : 1-b, (B-1, k_1+1) : 1-a, (B, k_1+1) : a+b-1\}$. After adding a packet to the first buffer, $\boldsymbol{x}^{\mathrm{PBF}}(\sigma^t) = (B, k_1+b)$. On the other hand, PB adds the packet only if it happens to be in the state $(B-1, k_1+1)$, in other cases it just loses this packet. Therefore, $\boldsymbol{x}^{\mathrm{PB}}(\sigma^t) = \{(B, k_1) : 1-b, (B, k_1+1) : b\} = \mu(\boldsymbol{x}^{\mathrm{PBF}}(\sigma^t))$ and $\mathsf{loss}_{\mathrm{PB}}(\sigma^t) = \mathsf{loss}_{\mathrm{PBF}}(\sigma^t)$. Similar reasoning can be applied if PBF lies in a lower triangle.

The reasoning for $\sigma_t = \textsc{Idle}$ is more complicated. Essentially, we have to show that it is possible to choose a strategy of (randomly) transmitting packets, such that the probability distribution of PB changes from $\mu(\boldsymbol{x}^{\mathrm{PBF}}(\sigma^{t-1}))$ to $\mu(\boldsymbol{x}^{\mathrm{PBF}}(\sigma^t))$. Obviously, each possible state of PB can transmit once from a single buffer only.

First, we note that PBF is work-conserving, i.e., the total mass of packets sent in step t is equal to 1. The only exception of this rule occurs if PBF ends step t with the buffers completely empty. This may occur only if $\boldsymbol{x}^{\mathrm{PBF}}(\sigma^{t-1})$ belongs to the lower triangle with vertices $(0,0)$, $(0,1)$, and $(1,0)$. By simply transmitting from a non-empty buffer if PB is at state $(0,1)$ or $(1,0)$, we may assure that $\boldsymbol{x}^{\mathrm{PB}}(\sigma^t) = 0 = \boldsymbol{x}^{\mathrm{PBF}}(\sigma^t)$.

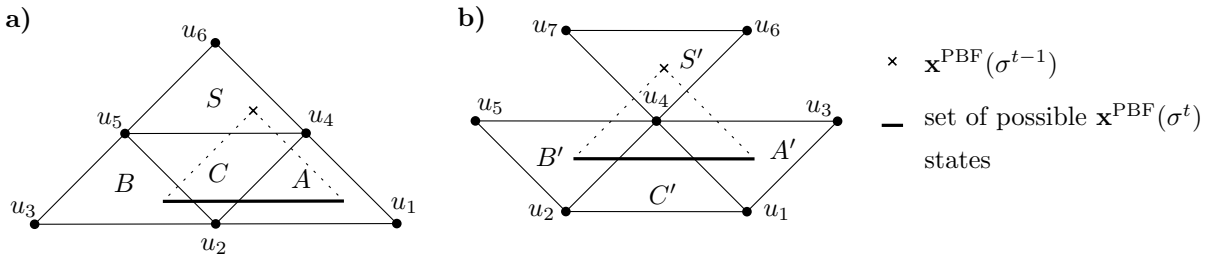

Fig. 7. State transitions of PB

For the remaining part of the proof, we consider two cases, depicted on Fig. 7a and Fig 7b. $\boldsymbol{x}^{\mathrm{PBF}}(\sigma^{t-1})$ belongs either to an upper triangle S or to a lower triangle S'. Therefore, $\boldsymbol{x}^{\mathrm{PBF}}(\sigma^t)$ may belong to one of the three triangles A, B, or C (or respectively A', B', or C' if it starts from a lower triangle). A set of possible $\boldsymbol{x}^{\mathrm{PBF}}(\sigma^t)$ positions is an interval depicted with a thick line in the figure. We fix $k'_0, k'_1 \in \mathbb{N}$, $a', b' \in [0,1]$, such that $\boldsymbol{x}^{\mathrm{PBF}}(\sigma^t) = (k'_0 + a', k'_1 + b')$. The behavior of PB is described by the table below. First two columns denote the starting and ending triangle for $\boldsymbol{x}^{\mathrm{PBF}}$. Instead of writing from which buffer PB has to transmit a packet we write in which state it has to end (we guarantee that to achieve that it has to transmit a packet from exactly one buffer). A rule $u_i \to u_j$ means that if PB is in state u_i then it has to end in state u_j (see Fig. 7). A rule $u_i \to (u_j : p_j, u_k : p_k)$ means that if PB is in state u_i, it should end in state u_j with probability p_j and in state u_k with probability p_k.

$\boldsymbol{x}^{\mathrm{PBF}}(\sigma^{t-1})$	$\boldsymbol{x}^{\mathrm{PBF}}(\sigma^t)$	Rule	Notes
S	A	$u_5 \to u_2;\ \ u_6 \to u_4;\ \ u_4 \to (u_1 : \frac{1-a'}{1-a}, u_2 : \frac{a'-a}{1-a})$	$a' \geq a$
	B	$u_6 \to u_5;\ \ u_4 \to u_2;\ \ u_5 \to (u_3 : \frac{1-b'}{1-b}, u_2 : \frac{b'-b}{1-b})$	$b' \geq b$
	C	$u_4 \to u_2;\ \ u_5 \to u_2;\ \ u_6 \to (u_5 : \frac{a'}{a+b-1}, u_2 : \frac{b'}{a+b-1})$	$a' + b' = a + b - 1$
S'	A'	$u_4 \to u_1;\ \ u_7 \to u_4;\ \ u_6 \to (u_3 : \frac{b'}{b}, u_4 : \frac{b-b'}{b})$	$b' \leq b$
	B'	$u_4 \to u_2;\ \ u_6 \to u_4;\ \ u_7 \to (u_5 : \frac{a'}{a}, u_4 : \frac{a-a'}{a})$	$a' \leq a$
	C'	$u_6 \to u_4;\ \ u_7 \to u_4;\ \ u_4 \to (u_1 : \frac{1-a'}{1-a-b}, u_2 : \frac{1-b'}{1-a-b})$	$a' + b' = a + b + 1$

To illustrate the concept, we consider the case $\boldsymbol{x}^{\mathrm{PBF}}(\sigma^{t-1}) \in S$ and $\boldsymbol{x}^{\mathrm{PBF}}(\sigma^t) \in A$ more thoroughly. First, we note that $a' \geq a$, and therefore the probabilities occurring in the PB rule are legal. Additionally, $a + b = a' + b'$. Second, we compute PB distribution at the end of step t.

$$\begin{aligned}
\mathbf{Pr}[\boldsymbol{x}^{\mathrm{PB}}(\sigma^t) = u_1] &= \frac{1-a'}{1-a} \cdot \mathbf{Pr}[\boldsymbol{x}^{\mathrm{PB}}(\sigma^{t-1}) = u_4] = 1 - a' \ , \\
\mathbf{Pr}[\boldsymbol{x}^{\mathrm{PB}}(\sigma^t) = u_4] &= \mathbf{Pr}[\boldsymbol{x}^{\mathrm{PB}}(\sigma^{t-1}) = u_6] = a + b - 1 = a' + b' - 1 \ ,
\end{aligned}$$

which implies that

$$\mathbf{Pr}[\boldsymbol{x}^{\mathrm{PB}}(\sigma^t) = u_2] = 1 - (1 - a') - (a' + b' - 1) = 1 - b' \ .$$

The proof for the remaining five cases is analogous. □

A.5 Why naive randomization does not work?

Let us take a look at a naive approach. When PBF transmits a fraction δ from the first buffer and $1 - \delta$ from the other, then the naive randomized algorithm PBN chooses a transmitting

buffer randomly with probabilities δ and $1-\delta$. Such approach does not guarantee that the expected numbers of packets in PBN buffers are equal to the fractional amounts of packets in the corresponding buffers of PB.

For example, consider a sequence $\textsc{Add}(0), \textsc{Add}(1), \textsc{Idle}, \textsc{Idle}$ for $B = 1$. PBF ends with empty buffers, whereas after the prefix $\textsc{Add}(0), \textsc{Add}(1), \textsc{Idle}$, state $\boldsymbol{x}^{\mathrm{PBN}}$ is equal to $(0,1)$ or $(1,0)$, both with probability $1/2$. For the last $\textsc{Idle}$ request, PBN tries to transmit a packet from an empty buffer with probability $1/2$, which results in expected non-zero number of packets in the buffers.

Similar situation occurs for $\textsc{Add}$ requests: consider a sequence $\textsc{Add}(0)^2, \textsc{Add}(1)^2, \textsc{Idle}^2$, $\textsc{Add}(0)$ for $B = 2$. Obviously, PBF serves this sequence without losing any packets. On the other hand, after processing everything but the last $\textsc{Add}(0)$, PBN has non-zero probability of having two packets in the first buffer, which leads to non-zero expected loss due to the last $\textsc{Add}(0)$ request.