Priority Queueing for Packets with Two Characteristics

Pavel Chuprikov, Sergey I. Nikolenko, Alex Davydow, Kirill Kogan

***Abstract*—Modern network elements are increasingly required to deal with heterogeneous traffic. Recent works consider processing policies for buffers that hold packets with different processing requirement (number of processing cycles needed before a packet can be transmitted out) but uniform value, aiming to maximize the throughput, i.e., the number of transmitted packets. Other developments deal with packets of varying value but uniform processing requirement (each packet requires one processing cycle); the objective here is to maximize the total transmitted value. In this work, we consider a more general problem, combining packets with both nonuniform processing and nonuniform values in the same queue. We study the properties of various processing orders in this setting. We show that in the general case natural processing policies have poor performance guarantees, with linear lower bounds on their competitive ratio. Moreover, we show several adversarial lower bounds for every priority queue and even for every online policy. On the positive side, in the special case when only two different values are allowed, 1 and V, we present a policy that achieves competitive ratio $\left(1+\frac{W+2}{V}\right)$, where W is the maximal number of required processing cycles. We also consider copying costs during admission.**

I. Introduction

Modern networks require implementation of advanced economic models that can be represented by desired objectives, network topology, buffering architecture, and its management policy. The current Internet architecture is mostly built for fairness, while consideration of other objectives such as network utilization, throughput, profit and others is required [27], [32]. For a given network topology and buffering architecture, design of management policies that optimize a desired objective is extremely important; a management policy of a single network element includes admission control and scheduling policies. Admission control is one of the critical elements of management policy. Most admission control policies are based on a simple characteristic such as buffer occupancy, whereas traffic has additional important characteristics such as processing requirements or value that are either not taken into account at all or a separate queue is allocated per traffic type. Incorporation of new characteristics (e.g., required processing per packet) in admission decisions and implementation of additional objectives beyond fairness lead to new challenges in design and implementation of traditional network elements.

In this work, we consider a single-queue switch where a buffer of size B is shared among all types of traffic. We do not assume any specific traffic distribution but rather analyze our switching policies against adversarial traffic using competitive analysis [6], [35], which provides a uniform throughput guarantee for online algorithms under all possible traffic patterns. An online algorithm ALG is *α-competitive* for some $\alpha \geq 1$ if for any arrival sequence σ the total value transmitted by ALG is at least $1/\alpha$ times the total value transmitted in an optimal solution obtained by an offline clairvoyant algorithm (denoted OPT). If an online algorithm is not α-competitive for any constant α independent of the input, it is said to be *non-competitive*. Note that a *lower bound* on the competitive ratio can be proven with a specific hard example while an *upper bound* represents a general statement that should hold over all possible inputs. In practice, the choices of processing order, implementation of push-out mechanisms etc. are likely to be made at design time. From this point of view, our study of worst-case behaviour aims to provide a robust estimate on the settings that can handle all possible loads.

The purpose of this work is to study the impact of both packet values and required processing on weighted throughput; to the best of our knowledge, this is the first attempt to study such impact. The paper is organized as follows. In Section II, we formally introduce the model we will use in this work, a model with both required processing and values. In Section III, we survey previous work in related buffer processing algorithms. In Section IV, we introduce several algorithms based on priority queueing that appear promising for this setting; these algorithms differ in the way how they order packets: by required processing, by value, or by a ratio of these numbers (i.e., by value per one processing cycle). In Section IV we begin with a negative result: we show that all of these algorithms have at least linear competitive ratio in the general case. Moreover, in Section V we proceed to show a general lower bound for *any* online algorithm proven in an adversarial fashion; this is an important new result for this model as previously considered special cases (uniform values with heterogeneous processing and uniform processing with variable values) allowed for optimal online policies. However, in Section VI we introduce an important special case when there are only two different possible values, i.e., packets may have different required processing but their value is limited to 1 and V. The maximal number of required processing cycles is W. In our main result, we present a policy based on a priority queue that achieves competitive ratio $\left(1+\frac{W+2}{V}\right)$.

A previous version of this work has appeared at INFOCOM 2015. Compared to the conference version, we have added several novel results including general lower bounds shown in Section V, in particular, a general adversarial lower bound in Theorem 10, significantly extended the discussion parts of the paper, added Figure 4 and published software used for experiments.

P. Chuprikov is with the Steklov Mathematical Institute at St. Petersburg and IMDEA Networks Institute, Madrid, Spain; e-mail: pschuprikov@gmail.com. S.I. Nikolenko is with the Steklov Mathematical Institute at St. Petersburg; e-mail: sergey@logic.pdmi.ras.ru. A. Davydow is with the Steklov Mathematical Institute at St. Petersburg; e-mail: adavydow@gmail.com. K. Kogan is with IMDEA Networks Institute; e-mail: kirill.kogan@imdea.org.

Note that while it may appear suspicious to compare packet values with required processing, in fact we are comparing ratios of the most valuable (resp., heaviest) packet to the least valuable (resp., lightest) packet because the minimal required processing and minimal value are always set to 1. In Section VII, we consider the β-push-out case, which takes copying cost into account by introducing additional penalties for push-out (a detailed explanation of β-push-out is given in Section VII). Section VIII presents simulation results where the proposed algorithms are evaluated with synthesized traces, and Section IX concludes the paper.

II. Model Description

We use a model similar to the one introduced in [1], [20] and subsequently used in [14]–[16], [23]–[25]. Consider a single queue that is able to hold B unit-sized packets and that handles the arrival of a sequence of packets, each of which is unit-sized. A new part of the problem setting in this work is to combine two different characteristics of a packet. Namely, we assume that each arriving packet p is branded with:

(1) the number of required processing cycles (required work, or weight) $w(p) \in \{1, \ldots, W\}$;
(2) its processing value $v(p) \in \{1, \ldots, V\}$.

These numbers are known for every arriving packet; for a motivation of why required processing may be available see [36], and values are usually defined externally. Although the values of W and V will play a fundamental role in our analysis, our algorithms will not need to know W or V in advance. Note that for $W = 1$ the model degenerates into a single queue of uniform packets with nonuinform value, as considered in, e.g., [5], [37], while for $V = 1$ it becomes a single queue of unit-valued packets with different required processing, as considered, e.g., in [22]–[24]. We will denote a packet with required processing w and value v by $(w \mid v)$, and a sequence of n packets with the same parameters w and v by $n \times (w \mid v)$.

The queue performs three main tasks, namely:

(1) *buffer management*, i.e., admission control of newly arrived packets;
(2) *processing*, i.e., deciding which of the currently stored packets will be processed;
(3) *transmission*, i.e., deciding if already processed packets should be transmitted and transmitting those that should.

A packet is *fully processed* if the processing unit has scheduled the packet for processing for at least its required number of cycles. Even though a fully processed packet is eligible for transmission, in some settings it can be deliberately delayed, e.g., if FIFO transmission order is required [24], [25]. We consider transmission order constraints only once in Theorem 5 and assume that the packet is transmitted as soon as it is fully processed.

We assume discrete slotted time, where each time slot consists of three phases (see Fig. 1 for an illustration):

(i) *arrival*: new packets arrive, and admission control decides if a packet should be dropped or, possibly, an already admitted packet should be pushed out;
(ii) *processing*: one packet is selected for processing by the scheduling unit;
(iii) *transmission*: at most one fully processed packet is selected for transmission and leaves the queue.

If a packet is *dropped* prior to being transmitted (while it still has a positive number of required processing cycles), it is lost. A packet may be dropped either upon arrival or due to a push-out decision while it is stored in the buffer. A packet contributes its value to the objective function only upon being successfully transmitted; note that only one packet may be transmitted per time slot. The goal is to devise buffer management algorithms that maximize the overall throughput, i.e., the total value of all packets transmitted out of the queue.

For an algorithm ALG and time slot t, we denote by $\mathrm{IB}^{\mathrm{ALG}}(t)$ the set of packets stored in ALG's buffer at time slot t after arrival but before processing (i.e., the buffer state shown in the second row of Fig. 1); if the timeslot is clear from the context we write simply $\mathrm{IB}^{\mathrm{ALG}}$. For every time slot t and every packet p currently stored in the queue, its number of *residual processing cycles*, denoted $w_t(p)$, is defined to be the number of processing cycles it requires before it can be successfully transmitted, and its *value*, denoted $v(p)$, is the number it contributes to the objective function upon transmission.

Three fundamental properties are often used in online algorithms. First, a policy is called *greedy* if it always accepts packets in the queue whenever it has free space. Greedy algorithms are usually amenable to efficient implementation and transmit everything if there is no congestion. Second, a policy is called *work-conserving* if it is always processing as long as it has packets with nonzero required processing in the buffer. Third, a policy is called *push-out* if it is allowed to drop packets that already reside in its queue; note that it does not make sense for a push-out policy to be non-greedy in the basic setting, but in the setting of Section VII where admitted packets incur nonzero copying cost this may not be the case. In what follows, we will assume that all push-out policies are greedy and all policies are work-conserving.

III. Related Work

Rich literature has been devoted to special cases of our model where one characteristic is assumed to be uniform. In particular, admission control policies for the case of single-queued buffers where packets with uniform processing and varying intrinsic value arrive have been thoroughly studied. In the case of two values (1 and V) and First-In-First-Out (FIFO) processing order, the works [5], [37] present a deterministic non-push-out policy with competitive ratio $(2 - \frac{1}{V})$, i.e., bounded by a constant. For the more general case, when packet values vary between 1 and V, the works [5], [37] prove that the competitive ratio cannot be better than $\Theta(\log V)$. In [4], this upper bound was improved to $2 + \ln V + O(\ln^2 V/B)$. In the push-out case with two packet values, the greedy policy was shown in [17] to be at least 1.282 and at most 2-competitive. Later, the upper bound on the greedy policy was improved to 1.894 [19]; this work also considers the β-push-out case and proves that the greedy policy is at least 1.544-competitive.

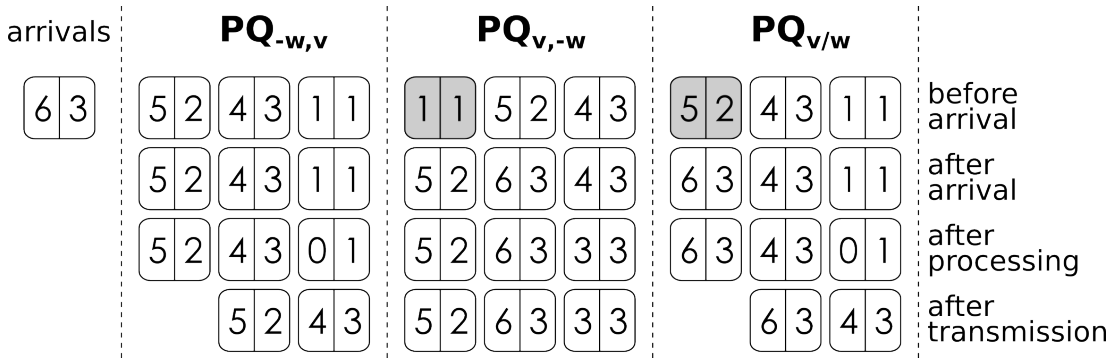

Fig. 1. A sample time slot of $\mathrm{PQ}_{-w,v}$, $\mathrm{PQ}_{v,-w}$, and $\mathrm{PQ}_{v/w}$.

Policies with memory have been considered in [3], [8], [26], [28].

Recently, packets with required processing but with uniform packet values in various settings have been considered in [9], [14]–[16], [23]–[25]. These works also follow the paradigm of competitive analysis, and their main results usually constitute good processing policies that have constant or logarithmic upper bounds on the competitive ratio. For a buffer with one queue of packets with uniform value, priority queue that orders packets according to their required processing is known to be optimal [14]. Our current work can be viewed as part of a larger research effort concentrated on studying competitive algorithms for management of bounded buffers. Initiated in [2], [17], [29], this line of research has received tremendous attention over the past decade. A survey by Goldwasser [11] provides an excellent overview of this field. Pruhs [33] provides a comprehensive overview of a related field of competitive online scheduling for server systems; however, scheduling for server systems usually concentrates on average response time and does not allow jobs to be dropped, while we focus mostly on throughput and allow push-out.

To control increasing queueing delays introduced by packet buffers, the bounded-delay model with varying intrinsic value was introduced by Kesselman et al. [18]. In that model, each packet is associated with a *slack* value s, which denotes a hard deadline when a packet should be processed. The greedy algorithm that always processes a packet with the earliest deadline is known to be 2-competitive [13], [18], and the best known competitive ratio is $2\sqrt(2) - 2 \approx 1.828$, as shown by Englert and Westermann [7]. A recent experimental study [34] evaluated the performance of different algorithms under a compatible deadline model. Note that a maximal slack value implicitly bounds a buffer size even if the buffer is theoretically unlimited. For this reason, the bounded-delay model appears to be more attractive for competitive analysis than the model, where a buffer is bounded explicitly. In this work, we not only consider an explicitly bounded buffer but also take into account required processing, which has a huge impact on the performance as both our theoretical results and simulation study will show.

Another very interesting class of results in competitive analysis are adversarial lower bounds that hold over all algorithms. Such bounds, when they can be proven, indicate that one cannot hope to get an optimal online algorithm, and a clairvoyant offline algorithm will always be able to outperform it. One well-known example of such a bound is the lower bound of $\frac{4}{3}$ on the competitive ratio of any algorithm in the model with multiple queues in a shared memory buffer and uniform packets (i.e., packets with identical value and required processing) [1], [12]. For the case of a single queue, previous works have considered two cases: variable value with uniform processing and variable processing with uniform values. In both cases, a single priority queue that orders packets with respect to the variable characteristic (largest value and smallest required processing first, respectively) is optimal, so there can be no nontrivial general lower bound regardless of transmission order. In the FIFO model, for the case of variable values and uniform processing there has been a line of adversarial lower bounds culminating in the lower bound of 1.419 that applies to all algorithms [21], with a stronger bound of 1.434 for the special case when $B = 2$ if all possible values are admissible [5], [37]. In the two-valued case, tight bounds are known: an adversarial lower bound of $r = \frac{1}{2}(\sqrt{13} - 1) \approx 1.303$ for any $B \geq 2$ and $r_\infty = \sqrt{2} - \frac{1}{2}(\sqrt{5 + 4\sqrt{2}} - 3) \approx 1.282$ for $B \to \infty$ and an online algorithm that achieves competitive ratio r for arbitrary B and r_∞ for $B \to \infty$ [8]. In the case of variable processing with uniform values, no general lower bounds for FIFO order are known apart from a simple lower bound of $\frac{1}{2}(W + 1)$ for greedy non-push-out policies [24].

IV. Algorithms and Lower Bounds

The ultimate goal of this entire line of research is to choose the right buffer management policy in every problem setting, i.e., for each possible switch configuration and every objective. For instance, previous works have studied in detail the interrelations between policies with and without *push-out*, the capability to drop previously accepted packets from the buffer [11], [31]; while non-push-out policies are simpler to implement in practice, they often turn out to be *non-competitive* in terms of weighted throughput with lower bounds on the competitive ratio linear in problem parameters such as buffer size B, maximal possible value V, or maximal required processing W. Note that even this is not quite the whole story: although non-push-out policies are usually clearly inferior with respect to (weighted) throughput, they can still

come out ahead for other objectives, e.g., minimizing the total power consumption (push-out might be a costly procedure).

Still, in this work we concentrate on the weighted throughput objective, and consider a setting where worst-case lower bounds, i.e., hard examples, are relatively easy to come by: we have two characteristics to play with for every packet, value and required processing, instead of just a single one as in majority of previous works [11], [31]. For this reason, we concentrate on studying the best class of natural algorithms available for the single queue setting, and previous research indicates that *priority queues with push-out* are the best tools that often lead to good results. In particular, Keslassy et al. showed that a single priority queue with push-out is optimal for packets with varying required processing and unit value [14].

Note, however, that though in previous work a single priority queue was usually the best algorithm, sometimes simply optimal, and the goal often was to try and achieve comparable throughput under additional constraints such as FIFO transmission order or multiple separate queues; in the setting with two different characteristics it is not even clear what a priority queue sorts its packets on: if one packet has less value but also less required processing than another, which one should we prefer? To capture different possible orderings in a priority queue, we introduce the following definition.

Definition 4.1: Let f be a function of packets, $f(w,v) \in \mathbb{R}$, with the intuition that better packets have larger values of f. Then the PQ_f processing policy is defined as follows:

- PQ_f is greedy, i.e., it accepts incoming packets as long as there is space in its buffer;
- PQ_f is work-conserving, i.e., it processes a packet as long as its buffer is not empty;
- PQ_f orders and processes packets in its queue in the order of decreasing values of f;
- PQ_f pushes out a packet p and adds a new packet p' to the queue at time slot t if the buffer is full, p is currently the worst packet in the buffer and p' is better than p: $f(p) = \min_{q \in \mathrm{IB}^{\mathrm{PQ}_f}} f(q)$, and $f(p') > f(p)$. Here $\mathrm{IB}^{\mathrm{PQ}_f}$ is $\mathrm{IB}^{\mathrm{PQ}_f}(t)$ for the current time slot t.

In other words, PQ_f orders and processes packets according to the function f. Note that this definition, again, restricts the space of possible algorithms. In theory, we could separate admission and processing order, accepting and pushing out packets with respect to one ordering but processing and transmitting them with respect to a different ordering. In [24], in the setting with one characteristic and FIFO transmission order constraint, a similar idea—decouple transmission order from processing order—has led to significant improvements in competitive ratios, including a constant upper bound on the competitiveness of a policy which admitted and processed its packets as a priority queue but transmitted them in FIFO order. However, throughout this paper we simplify our considerations and assume that admission and processing orders are the same.

In particular, we consider three specific priority queues (here w denotes the *current* residual work and v denotes the packet's value):

(1) $\mathrm{PQ}_{-w,v} = \mathrm{PQ}_{-w+v/(V+1)}$ orders packets in the increasing order of their required processing, breaking ties by value;
(2) $\mathrm{PQ}_{v,-w} = \mathrm{PQ}_{v-w/(W+1)}$ orders packets in the decreasing order of their value, breaking ties by required processing;
(3) $\mathrm{PQ}_{v/w}$ orders packets in the decreasing order of their value-to-work ratio, i.e., it prioritizes packets that yield the best value per one time slot of processing.[1]

Fig. 1 shows a sample time slot of these priority queues; in this case $B = 3$, all policies start with $(5 \mid 2)$, $(4 \mid 3)$, and $(1 \mid 1)$ in their queues, and a $(6 \mid 3)$ packet arrives. $\mathrm{PQ}_{-w,v}$ rejects the $(6 \mid 3)$ since it has the largest processing requirement, $\mathrm{PQ}_{-v,w}$ pushes out $(1 \mid 1)$ since it has the smallest value, and $\mathrm{PQ}_{v/w}$ pushes out $(5 \mid 2)$ since it has the worst v/w ratio of $2/5$ compared to $3/4$, 1, and $3/6$ of the other three packets.

One of the goals of this work is to explore which order performs best. Our main result in this part is that, in general, priority queues fail to provide constant or even logarithmic competitiveness in the setting with two packet characteristics, as they do in cases, where there is only a single characteristic. For all three specific PQ policies shown above, we prove linear (in V and/or W) lower bounds on their competitive ratios against an optimal algorithm. This is an interesting and somewhat discouraging result since priority queues have proven to be efficient when each characteristic is considered separately, often with constant upper bounds on the competitive ratio or even shown to be optimal policies. Note that while it may seem intuitive that $\mathrm{PQ}_{v/w}$ should be best at least among these three, we will see lower bounds for all of them in this section, and later an even more counterintuitive result in a special case with two values.

For a lower bound, it suffices to present a hard sequence of packets on which the optimal algorithm outperforms the one in question; so in the theorems below we simply describe this sequence. We also show matching upper bounds when applicable.

Theorem 1: For a buffer of size B, maximal packet required processing W, and maximal packet value V, $\mathrm{PQ}_{-w,v}$ is at least V-competitive and at most V-competitive.

Proof: First, there arrive $B \times (1 \mid 1)$ packets, i.e., B packets with required processing 1 and value 1; $\mathrm{PQ}_{-w,v}$ accepts them while OPT does not. Then there arrive $B \times (2 \mid V)$ packets accepted by OPT; $\mathrm{PQ}_{-w,v}$ skips them since they have larger required processing than already admitted. No more packets arrive, so in $2B$ steps $\mathrm{PQ}_{-w,v}$ processes packets with total value B; OPT, with total value VB. The same sequence is repeated to get the asymptotic bound. The upper bound follows since PQ is optimal for uniform values and variable required processing; this means that $\mathrm{PQ}_{-w,v}$ processes as many packets as OPT, so it cannot lose by a factor of more than V. ■

[1]Note that here we also have two possibilities for breaking ties, $\mathrm{PQ}_{v/w,-w} = \mathrm{PQ}_{v/w-w/(W^2+1)}$ and $\mathrm{PQ}_{v/w,v} = \mathrm{PQ}_{v/w+v/(WV+1)}$, but in this case the tie-breakers will be irrelevant for all our statements, so we unite them under the same notation.

Theorem 2: For a buffer of size B, maximal packet required processing W, and maximal packet value V, $\mathrm{PQ}_{v,-w}$ is at least $\left(\frac{(V-1)}{V}W - o(1)\right)$-competitive.

Proof: In the first burst, there arrive $B \times (W \mid V)$ which $\mathrm{PQ}_{v,-w}$ accepts but OPT does not. Then, over the next W time slots there arrives a $(1 \mid V-1)$ packet on every time slot. OPT accepts, processes, and transmits this packet immediately, while $\mathrm{PQ}_{v,-w}$ drops it since it has worse value than the ones in its queue. On time slot $W+1$, when $\mathrm{PQ}_{v,-w}$ has processed one $(W \mid V)$ packet, another $(W \mid V)$ arrives, to be accepted by $\mathrm{PQ}_{v,-w}$, and this brings us back to the same state as on the first time slot. Thus, over this sequence $\mathrm{PQ}_{v,-w}$ has processed packets with total value V, OPT has processed packets with total value $W(V-1)$, and the sequence can be repeated. After we repeat the sequence C times, we finish by "flushing" both buffers with $B \times (1 \mid V)$. Thus, both algorithms will end with VB more processed value, and the competitive ratio over this sequence is

$$\frac{(V-1)WC + VB}{V(B+C)}.$$

It remains to let $C \to \infty$. ∎

So is $\mathrm{PQ}_{v/w}$ that combines characteristics and aims for the best "value per timeslot" any better in the worst case? Unfortunately, no.

Theorem 3: For a buffer of size B, maximal packet required processing W, and maximal packet value V, $\mathrm{PQ}_{v/w}$ is at least $\min\{V, W\}$-competitive.

Proof: We denote $m = \min\{V, W\}$. In the first burst, there arrive $B \times (1 \mid 1)$ packets which $\mathrm{PQ}_{v/w}$ accepts but OPT does not. Then on the same time slot there arrive $B \times (m \mid m)$ packets which OPT accepts but $\mathrm{PQ}_{v/w}$ does not since they have work-to-value ratio worse than 1. Thus, in Bm steps OPT transmits total value mB, while $\mathrm{PQ}_{v/w}$ only transmits total value B, and this sequence can be repeated. ∎

V. General lower bounds

Theorems 1–3 have established that $\mathrm{PQ}_{-w,v}$, $\mathrm{PQ}_{v,-w}$, and $\mathrm{PQ}_{v/w}$ are non-competitive. But perhaps we have just failed to be inventive enough in designing these priority queues? Maybe we can devise a better priority queue, or simply a better online algorithm that will achieve constant competitiveness or even be optimal? In this section, we dash these hopes by proving *general lower bounds* on all online algorithms. They are proven in an *adversarial* way: we construct a sequence of inputs where further inputs depend on the choices an online algorithm makes, so in the end we find a "bad" input for every possible choice.

Note that some of the bounds below are nontrivial only in the extreme cases of $W, V > B$ and even $W, V \gg B$, but it still shows that we cannot hope for constant upper bounds unless we explicitly assume $B > W, V$ and somehow use it in the proof. On the positive side, note that most of these lower bounds only need two kinds of packets, so they also work in restricted settings where value and/or work can only take some of the values in their respective intervals.

Later we will see that an important special case is the *two-valued case*, when required processing can be an arbitrary integer $1 \le w \le W$ but there are only two possible values, 1 and V; we will prove an upper bound for this case in Section VI. Therefore, we note special cases of lower bounds for this case as well. Note that all lower bounds trivially extend from the two-valued case to the general case but not vice versa.

We begin with a very simple case to illustrate basic ideas. It turns out that even reducing the buffer to a single slot does not let us construct a competitive online algorithm.

The basic idea for the following and most other general lower bounds is to have "light" and "heavy" packets such that it is x times better to process "light" packets per time slot, but a "heavy" packet is x times better than a single "light" packet, so if the algorithm pushes out the "heavy" packet, we can stop the arrivals and win x times the value again. This is where the $\sqrt{W}$ that appears here and in many further bounds comes from: since a "heavy" packet is x times worse per time slot but x times better overall, it must have x^2 times the processing of a "light" packet.

Theorem 4 ($B = 1$): For $B = 1$ and $V > \sqrt{W}$, any online algorithm ALG is at least $\sqrt{W}$-competitive. Further, in the two-valued case or if $V \le \sqrt{W}$ any online algorithm ALG is at least $\min\{V, W/V\}$-competitive for $B = 1$.

Proof: On the first step, two packets arrive, $(W \mid V)$ and $\left(1 \mid \frac{V}{\sqrt{W}}\right)$. If ALG accepts $\left(1 \mid \frac{V}{\sqrt{W}}\right)$, no other packets arrive, OPT accepts $(W \mid V)$ and wins by a factor of $\sqrt{W}$ in value. If ALG accepts $(W \mid V)$, the same pair of packets, $\left(1 \mid \frac{V}{\sqrt{W}}\right)$ and $(W \mid V)$, continue to arrive every tick, OPT processes light packets and earns value $V\sqrt{W}$ while ALG earns V. If ALG decides to switch to a lighter packet in the process, arrivals stop immediately, OPT accepts the current $(W \mid V)$, and the result is even worse for ALG. For the case when the value of $V/\sqrt{W}$ is either unavailable or is less than or equal to one, replace $\left(1 \mid \frac{V}{\sqrt{W}}\right)$ with $(1 \mid 1)$ and observe that in the first case we get a ratio of V and in the second, W/V. ∎

In traditional networking, most buffers implement FIFO processing order because of its simplicity and desired properties. The following theorem demonstrates that under the FIFO constraint on processing and transmission order, lower bounds may significantly deteriorate and become non-competitive. Note that the admission order is not constrained in the theorem, ALG is free to push out any packet.

Theorem 5 (FIFO order): For arbitrary B and $V > \sqrt{W}$, any online algorithm ALG that preserves FIFO processing and transmission order is at least $\left(\frac{\sqrt{W}}{B} + 1 - \frac{1}{B}\right)$-competitive. In the two-valued case or if $V \le \sqrt{W}$, any online algorithm ALG with FIFO processing and transmission order is at least $\frac{V+B-1}{W+B-1}$-competitive.

Proof: The construction is very similar: $(W \mid V)$ and $B \times \left(1 \mid \frac{V}{\sqrt{W}}\right)$ arrive on the first step. Then ALG either:

(a) drops $(W \mid V)$, in which case arrivals stop, and OPT earns total value $V + (B-1)\frac{V}{\sqrt{W}}$ while ALG earns $B\frac{V}{\sqrt{W}}$, for the total ratio of $\frac{V+(B-1)V/\sqrt{W}}{BV/\sqrt{W}} = \frac{\sqrt{W}}{B} + 1 - \frac{1}{B}$; or

(b) keeps processing $(W \mid V)$ while we feed OPT with more $\left(1 \mid \frac{V}{\sqrt{W}}\right)$s; then arrivals stop, ALG finishes the other $B-1$ of his packets and earns total value $V + (B-1)\frac{V}{\sqrt{W}}$

while OPT has earned $(W+B-1)\frac{V}{\sqrt{W}}$ in value; the ratio in this case is worse,

$$\frac{(W+B-1)V/\sqrt{W}}{V+(B-1)V/\sqrt{W}} = \frac{W+B-1}{\sqrt{W}+B-1} > \frac{\sqrt{W}}{B}+1.$$

For the second part, replace $\left(1 \mid \frac{V}{\sqrt{W}}\right)$ with $(1 \mid 1)$ and observe that in case (a) we get a competitive ratio of $\frac{V+B-1}{B}$, and in case (b) the competitive ratio is $\frac{V+B-1}{W+B-1}$, which is smaller. ∎

Next, we turn to priority queues. We have already mentioned that priority queues arise naturally as candidates for best possible policies, but, again, by restricting our consideration to the class of priority queues (i.e., algorithms that have deterministic linear orderings on the packets) we get a larger general lower bound, regardless of what this order specifically is. We would like to emphasize that not all algorithms can be represented as PQ_f. For example, some algorithms may base their decisions on buffer occupancy or statistical data collected over previous timeslots. In particular, Theorem 4 is not a special case of the following theorem.

Theorem 6 (arbitrary PQ): For $V > \sqrt{W}$ and any priority function f such that $f(w,v) \in \mathbb{R}$, the algorithm PQ_f is at least $\sqrt{W}$-competitive. In the two-valued case or if $V \le \sqrt{W}$, algorithm PQ_f is at least $\min\{V, W/V\}$-competitive.

Proof: Again, we only need two kinds of packets, $(W \mid V)$ and $\left(1 \mid V/\sqrt{W}\right)$. There are two cases:

(1) if PQ_f prefers $(W \mid V)$, i.e. $f(W,V) > f(1, V/\sqrt{W})$, we keep feeding the algorithms with both kinds of packets; PQ_f chooses and processes $(W \mid V)$s while OPT is processing $\left(1 \mid V/\sqrt{W}\right)$s, getting $\sqrt{W}$ times more value per time slot;

(2) if, on the other hand, PQ_f prefers $\left(1 \mid V/\sqrt{W}\right)$ to $(W \mid V)$, then $B \times \left(1 \mid V/\sqrt{W}\right)$ and $B \times (W \mid V)$ arrive on the first burst and then arrivals stop; ALG fills its buffer with light packets, OPT takes the heavy ones, and after BW time slots OPT has again transmitted $\sqrt{W}$ times more value.

For the second part, again, replace $\left(1 \mid \frac{V}{\sqrt{W}}\right)$ with $(1 \mid 1)$. ∎

Finally, we consider general lower bounds for all deterministic online algorithms. We begin with the two-valued case. The idea of the following lower bound is to send in plenty of both "light" packets $(1 \mid 1)$ and "heavy" packets $(W \mid V)$. If ALG accepts few "heavy" packets, OPT can accept all of them, halt arrivals, and win a lot with the sequence. If ALG accepts a lot of "heavy" packets, OPT fully processes all "light" packets, winning over ALG that has to begin processing "heavy" ones, and then the buffers are flushed out with the "best" possible packets $(1 \mid V)$.

Theorem 7: For a buffer of size B, maximal packet required processing W, and available packet values 1 and $V > 1$, every online deterministic algorithm ALG is at least $\left(1 + \frac{V-1}{V^2} - O\left(\frac{1}{W}\right)\right)$-competitive.

Proof: On the first step there arrive $B \times (1 \mid 1)$ and $B \times (W \mid V)$. Suppose that ALG has accepted n of $(W \mid V)$ packets. Again, there are two cases.

- $\mathbf{n < \frac{V^2-1}{V^2+V-1}B}$: in this case, OPT chooses to accept $B \times (W \mid V)$, and no new packets arrive. After BW time slots, OPT has processed packets with total value VB, and ALG has processed at most $(B-n)+Vn$, yielding competitive ratio

$$\frac{VB}{B-n+Vn} = \frac{V}{1+(V-1)\frac{n}{B}},$$

which is at least $\frac{V^2+V-1}{V^2}$ for $\frac{n}{B} < \frac{V^2-1}{V^2+V-1}$.

- $\mathbf{n \ge \frac{V^2-1}{V^2+V-1}B}$: in this case, OPT accepts $B \times (1 \mid 1)$. After B time slots, OPT has transmitted total value B, while ALG has processed at most $(B-n) \times (1 \mid 1)$ and $B/W \times (W \mid V)$. Now $B \times (1 \mid V)$ arrive and then no more packets; after all buffers have been emptied OPT gets VB more value and ALG at most VB, which yields the ratio

$$\frac{B+VB}{(B-n)+VB/W+VB} = \frac{V+1}{V+1-n/B} - O\left(\frac{1}{W}\right) \ge$$
$$\ge \frac{V^2+V-1}{V^2} - O\left(\frac{1}{W}\right) \text{ for } \frac{n}{B} \ge \frac{V^2-1}{V^2+V-1}.$$

∎

These same ideas can lead to a general lower bound on all deterministic online algorithms. We first show the proof for $B = 1$ here and then show the proofs for $B = 2$ as a characteristic special case and then for the general case of arbitrary B. These proofs essentially rely on the availability of a packet whose value is a specific fraction of V, and thus they are not applicable in the two-valued case. The proof for the general case is technically quite involved so we consider in detail the case of $B = 2$ and then show a proof sketch for arbitrary B. The basic idea of "light" and "heavy" packets remains the same, but for buffer size B we will need $B+1$ "levels" of different packets to get a recursive construction and an inductive proof of the bound. This leads to $\sqrt[4]{W}$ in the case of $B = 2$ and $\sqrt[2B]{W}$ in the general case.

Theorem 8 (arbitrary online ALG, $B = 1$): For $B = 1$, the competitive ratio of any deterministic online algorithm ALG is at least $\min\{\sqrt{W}, V\}$.

Proof: There are only two kinds of packets involved in the lower bound: "heavy" packets $(W \mid V)$ and "medium" packets $\left(\frac{W}{l^2} \mid \frac{V}{l}\right)$, where l is a parameter to be defined later. On the first burst, both packets arrive, $1 \times (W \mid V)$ and $1 \times \left(\frac{W}{l^2} \mid \frac{V}{l}\right)$, and then on every time slot $1 \times \left(\frac{W}{l^2} \mid \frac{V}{l}\right)$ arrives until ALG accepts it. Denote by t the time when ALG accepts a "medium" packet instead of $(W \mid V)$. There are two cases.

1. $t = W$, i.e., ALG processes $(W \mid V)$ to completion. In this case, we repeat the sequence by sending another $(W \mid V)$ after W time slots. OPT will process "medium" packets all the time, getting total value of Vl per W time slots while ALG obtains value V per W time slots.
2. At some $t < W$, ALG accepts a "medium" packet, pushing out the "heavy" one. In this case, "medium" packets immediately stop, and OPT processes only the "heavy" packet with value V while ALG processes one "medium" packet with total value $\frac{V}{l}$. Then both buffers become empty, and the sequence can be repeated.

We now take $l = \min\{\sqrt{W}, V\}$ to get the bound. ■

Theorem 9 (arbitrary online ALG, $B = 2$): For $B = 2$, the competitive ratio of any deterministic online algorithm ALG is at least $\frac{1}{2}\min\{\sqrt[4]{W}, \sqrt{V}\}$.

Proof: The basic idea is to preserve the following invariant: on every step except a small fraction OPT will obtain at least l times more value than ALG. There are three kinds of packets in the sequence: "heavy" $(W \mid V)$, "medium" $\left(\frac{W}{l^2} \mid \frac{V}{l}\right)$, and "light" $\left(\frac{W}{l^4} \mid \frac{V}{l^2}\right)$, where l is a parameter to be defined later. On the first burst, there arrive a "heavy" and a "medium" packet, $1 \times (W \mid V)$ and $1 \times \left(\frac{W}{l^2} \mid \frac{V}{l}\right)$, followed by two "light" packets, $2 \times \left(\frac{W}{l^4} \mid \frac{V}{l^2}\right)$. Then, on every time step two more "light" packets arrive, $2 \times \left(\frac{W}{l^4} \mid \frac{V}{l^2}\right)$.

There are several cases. Note that in what follows, OPT always keeps one "heavy" packet in its buffer, and "heavy" packets never arrive during the period we are counting the packets in. A new "heavy" packet only arrives when the sequence is repeated from the start.

1. ALG does not push out either "heavy" or "medium" packet in favor of "light" ones. Then OPT keeps one "heavy" packet in the buffer while it processes "light" packets, getting value $\frac{Vl^2}{W}$ per processing cycle while ALG is getting at most $\frac{Vl}{W}$ (from the "medium" packet). As soon as ALG finishes the "medium" or "heavy" packet, another packet of the same kind arrives, and the sequence is repeated. Note that this ratio of l in favor of OPT occurs every time OPT is able to process a "light" packet while ALG is processing a different one.
2. At some timeslot t, ALG pushes out the "heavy" packet for a "light" one. In this case, OPT accepts the last arriving "medium" packet (note that there may have been several "medium" packets arriving due to ALG processing them to completion), and there are no more arrivals after timeslot t. OPT finishes the "heavy" packet residing in its buffer and the last arriving "medium" packet while ALG can process at most a "medium" and a "light" one. As a result, before time t OPT had l times more value per timeslot by case 1, and after time t ALG has earned at most $V(\frac{1}{l} + \frac{1}{l^2})$ total value while OPT has earned $V(1 + \frac{1}{l})$, for the total ratio of l over the entire sequence.
3. At some timeslot t, ALG pushes out the "medium" packet for a "light" one. This is a slightly more complicated case, with two subcases depending on the time t' when the pushed out "medium" packet had arrived:
   (i) if $t - t' < \frac{W}{l^2}$ (the pushed out "medium" packet arrived less than $\frac{W}{l^2}$ timeslots ago), OPT accepts it at time t', and all arrivals stop until time $t' + \frac{W}{l^2}$, i.e., until OPT finishes processing it; over these $\frac{W}{l^2}$ timeslots:
   - OPT is getting value $\frac{Vl}{W}$ per time slot every time, for a total value $\frac{V}{l}$;
   - ALG can get total value $\frac{V}{l^2}$ once by processing this "light" packet, but on all other timeslots it could not get more than $\frac{V}{W}$ value per time slot (assuming it was processing the "heavy" packet); hence, over this period of time ALG gets no more than $\frac{V}{l^2} + \frac{V}{l^2}$ total value;

   hence, the competitive ratio over these timeslots is at least $\frac{l}{2}$;
   (ii) if $t - t' \geq \frac{W}{l^2}$ (the pushed out "medium" packet arrived at least $\frac{W}{l^2}$ timeslots ago), OPT is processing "light" packets all this time, and as soon as ALG has finished the "light" packet, another "medium" packet arrives, reverting to the original situation; in this case:
   - OPT has processed $\frac{(t-t')l^4}{W}$ "light" packets plus the one final "light" packet, obtaining total value at least $\frac{(t-t')Vl^2}{W} + \frac{V}{l^2}$;
   - ALG has obtained total value at most $\frac{(t-t')V}{W}$ over the past $t - t'$ timeslots (if ALG was processing the "medium" packet it is now worth nothing since it has been pushed out, so value can only come from the "heavy" packet) plus the final "light" packet, for a total of at most $\frac{(t-t')V}{W} + \frac{V}{l^2}$;

   since $t - t' \geq \frac{W}{l^2}$, the competitive ratio is at least

$$\frac{\frac{(t-t')Vl^2}{W} + \frac{V}{l^2}}{\frac{(t-t')V}{W} + \frac{V}{l^2}} \geq \frac{V + \frac{V}{l^2}}{2\frac{V}{l^2}} = \frac{l^2 + 1}{2}.$$

As a result, during the entire sequence the competitive ratio is never smaller than $\frac{l}{2}$, and the constraint on l is that $l \leq \min\{\sqrt[4]{W}, \sqrt{V}\}$. ■

Theorem 10 (general case): For arbitrary B, the competitive ratio of any deterministic online algorithm ALG is at least $\frac{1}{2}\left(\min\{\sqrt[2B]{W}, \sqrt[B]{V}\} - 1\right)$.

Proof of Theorem 10: We proceed by induction with a construction similar to the proof of Theorem 9. The induction base for $B = 1$ and $B = 2$ has already been considered in Theorems 8 and 9.

For the induction step, consider $k + 1$ types of packets that differ by a factor of v from each other in value and by a factor of v^2 in required processing: the first packet has value 1 and work 1, the second value v and work v^2, and so on until value v^k and work v^{2k}; for this to work we have to have $v \leq \min\{\sqrt[2B]{W}, \sqrt[B]{V}\}$. In the arrivals, we will preserve the invariant that ALG's buffer can only have two packets of identical value if they are the cheapest; i.e., we will not give "repeated" packets to the algorithm but send in a new packet of a certain size only after the previous one has finished processing or has been dropped; the cheapest and lightest packets are coming in with a steady stream, and in most cases OPT keeps processing them.

If ALG at some point pushes out the heaviest packet, arrivals stop immediately, and OPT finishes the heaviest packet; in this case, ALG loses by a factor of at least $\frac{1+v+v^2+\cdots v^{k-1}}{v^k}$. If ALG is working on the heaviest packet, OPT is working on the lightest packet and has unit gain over ALG by a factor of at least $v^k - 1$.

If, otherwise, ALG keeps the heaviest packet in the buffer and is working on some other packet, arrivals and OPT are operating as in the induction hypothesis for $B - 1$, with one slot in the buffer reserved for the heaviest packet. There is only one exception to this operation: if ALG has pushed out the second heaviest packet (which is heaviest in the $B - 1$ case), arrivals do not stop completely but cease temporarily for $v^{2(k-1)}$, i.e., for the time it takes to process the second

heaviest packet. Afterwards, we again send in one packet of each type except the heaviest.

The technicality here is that by sending in these packets to "reset" the buffer, we are violating the declared invariant. To fix this, we count all packets except the heaviest one as "processed" by the algorithm and assume that it has gotten full value for them (afterwards, if ALG is working on one of the "old" packets, we can simulate it as idle time). In total, OPT has processed v^{k-1} total value, and ALG has not processed more than $1 + v + v^2 + \cdots v^{k-2}$ which constitutes at most $v^k/v^2 = v^{k-2}$ of the "unit" cost of the heaviest packet.

Now the minimal competitive ratio for ALG out of the three cases is the last one:

$$\frac{v^{k-1}}{v^{k-2} + \sum_{0 \le i \le k-2} v^i} = \frac{v^{k-1}}{v^{k-2} + \frac{v^{k-1}-1}{v-1}} \ge \frac{1}{2}(v-1),$$

and we take $v = \min\{ \sqrt[2B]{W},\ \sqrt[B]{V}\}$ to obtain the lower bound. ∎

Note that while this bound is not too large in practical cases, it is still non-constant, that is, we now cannot hope for an online algorithm with constant competitiveness unless we impose and use some additional constraints on the problem setting. One fruitful constraint turns out to be the constraint that the only two allowed values are 1 and V.

VI. Upper bound for the two-valued case

Given the pessimistic results of previous two sections, it remains only to impose additional constraints on at least one of the characteristic and try to distinguish important special cases under which a "good" upper bound may exist. In this section, we consider an important special case when there are only two possible packet values, 1 and V, so there are two kinds of packets, $(w \mid 1)$ and $(w \mid V)$; the required processing can still vary from 1 to W. This case often occurs in practice; for instance, $(w \mid 1)$ may represent "commodity" packets while $(w \mid V)$ corresponds to "golden" packets that have paid more to be processed. Similar special cases have been considered, e.g., in [23].

We will show in Theorem 12 that in this special case, the $\mathrm{PQ}_{v,-w}$ policy has an attractive upper bound on the competitive ratio, which implies a constant upper bound on the competitive ratio of both $\mathrm{PQ}_{v,-w}$ and $\mathrm{PQ}_{v/w}$ in case when $W < V$. This upper bound is fundamentally different from the lower bounds presented earlier: instead of showing that an algorithm (or a set of those) sometimes performs badly, it shows that these particular algorithms *always* perform well. However, we begin with negative results; Theorem 11 provides matching tight lower bounds for the main result that follows. A plot summarizing different lower bounds for the two-valued case is shown on Figure 2.

Theorem 11: Consider a buffer of size B with maximal required processing W and possible packet values 1 or V. Then:

(1) $\mathrm{PQ}_{-w,v}$ is at least V-competitive;
(2) if $W \ge V$ then $\mathrm{PQ}_{v/w}$ is at least V-competitive;
(3) $\mathrm{PQ}_{v,-w}$ is at least $\left(\frac{W}{V} + o(1)\right)$-competitive.

Proof:

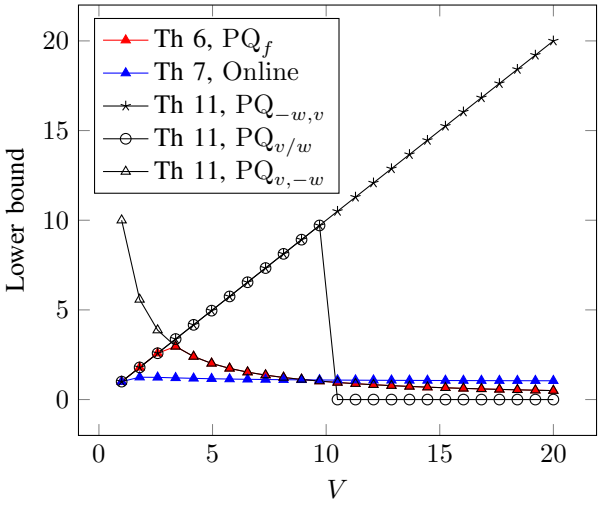

Fig. 2. Lower bounds on the competitive ratios for the two-valued case with fixed $W = 10$. Higher value of a lower bound is better.

(1) The construction from Theorem 1 uses only packets of values 1 and V.
(2) Again, we present a hard sequence of arrivals. In the first burst, there arrive $B \times (1 \mid 1)$ accepted by $\mathrm{PQ}_{v/w}$ but not OPT. Then, on the same time slot there arrive $B \times (W \mid V)$ which $\mathrm{PQ}_{v/w}$ has to miss since $\frac{W}{V} \ge \frac{V}{V} = \frac{1}{1}$ but which OPT accepts. Then, in BW steps, $\mathrm{PQ}_{v/w}$ will have processed total value B while OPT will have processed total value BV, which implies the bound.
(3) In the first time slot, there arrive $B \times (W \mid V)$, which $\mathrm{PQ}_{v,-w}$ accepts, but OPT does not. Then, over the next W time slots there arrives a $(1 \mid 1)$ on every time slot. OPT transmits it, and $\mathrm{PQ}_{v,-w}$ drops it. On time slot $W+1$, when $\mathrm{PQ}_{v,-w}$ has processed one $(W \mid V)$ packet, another $(W \mid V)$ packet arrives, which brings us back to the same state as on the first time slot. Over this sequence $\mathrm{PQ}_{v,-w}$ has processed packets with total value V, and OPT with total value W. Repeating this sequence C times, we get competitive ratio $\frac{WC+O(1)}{VC+O(1)}$ and let $C \to \infty$.

∎

In the next theorem, we show one of the main results of this work, an upper bound on the competitive ratio of $\mathrm{PQ}_{v,-w}$. Note that the lower bounds from Theorem 11 and previous sections, which were linear in V, do not work for $\mathrm{PQ}_{v,-w}$ since in the two-valued case, it always processes packets with value V first and with value 1 last, so intuitively we cannot lose more than the worst possible packet with value V, $(W \mid V)$, against the best possible packet with value 1, $(1 \mid 1)$ (Theorem 11 (3) shows that we cannot lose any less). The following proof captures this intuition.

Theorem 12: Consider a buffer of size B with maximal required processing W and possible packet values 1 or V. Then $\mathrm{PQ}_{v,-w}$ is at most $\left(1 + \frac{W+2}{V}\right)$-competitive.

Proof: By the definition of the $\mathrm{PQ}_{v,-w}$ queue, any packet with value V pushes out any packet with value 1. This is the crucial property that we need to prove this upper bound. For brevity, throughout this proof we denote $\mathrm{PQ} = \mathrm{PQ}_{v,-w}$. We define the following sets of packets:

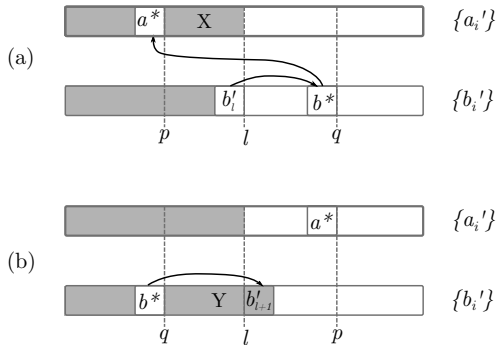

Fig. 3. Illustration for the subcases of Lemma 13: subcase $\mathbf{p} \le \mathbf{q}$ is shown on (a), and subcase $\mathbf{q} < \mathbf{p}$ is shown on (b). Arrows represent $\le$ relation, and shaded areas denote sets of elements that sum to either A_l or B_l (note that lemma's premise contains a $A_l \ge B_l$ inequality); dotted lines denote specific position in a sequence.

(1) $\mathrm{IB}_v^{\mathrm{ALG}} = \{p \in \mathrm{IB}^{\mathrm{ALG}} : \mathrm{value}(p) = v\}$ contains packets with value v in $\mathrm{IB}^{\mathrm{ALG}}$;
(2) $\mathrm{Trans}_v^{\mathrm{ALG}} = \{p$ transmitted by ALG : $\mathrm{value}(p) = v\}$ is the set of packets with value v transmitted by ALG; $\mathrm{Trans}^{\mathrm{ALG}} = \bigcup_v \mathrm{Trans}_v^{\mathrm{ALG}}$;
(3) $\mathrm{IBT}_v^{\mathrm{ALG}} = \mathrm{IB}_v^{\mathrm{ALG}} \cup \mathrm{Trans}_v^{\mathrm{ALG}}$ is the set of packets with value v either already transmitted by ALG or currently residing in its buffer; $\mathrm{IBT}^{\mathrm{ALG}} = \bigcup_v \mathrm{IBT}_v^{\mathrm{ALG}}$.

We also define $\Phi_v^{\mathrm{ALG}}(l) = \sum_{i=1}^{l} w(p_i)$, where p_i is the ith packet from $\mathrm{IBT}_v^{\mathrm{ALG}}$ in PQ order. Here $w(p)$ is the residual processing time of a packet at the current time moment; in particular, we let $w(p) = 0$ for already transmitted packets.

In the proof, we will sometimes force OPT to transmit certain packets immediately, "for free", thus improving its throughput. We denote the set of these packets at the current timeslot as $\mathrm{Free}^{\mathrm{OPT}}$; they do not fall into $\mathrm{Trans}^{\mathrm{OPT}}$ but rather contribute to the objective separately. The only requirement is that for any $p \in \mathrm{Free}^{\mathrm{OPT}}$ we must have $\mathrm{value}(p) = 1$, i.e., we only give out packets of value 1 for free. We begin with a technical statement.

Lemma 13: Let $a_1, a_2, \ldots, a_m$ and $b_1, b_2, \ldots, b_m$ be two sequences of numbers in nondecreasing order, and, moreover, suppose that $\forall l \in \{1, \ldots, m\}$ the prefix sums of length l satisfy the following inequality: $\sum_{i=1}^{l} a_i \ge \sum_{i=1}^{l} b_i$. Let also a^* and b^* be any two numbers, such that $a^* \ge b^*$. If a^* is inserted into $a_1, \ldots, a_m$, and b^* is inserted into $b_1, \ldots, b_m$ then the prefix sums of resulting sequences satisfy the same inequality. Formally, if the result of a^*'s insertion is $a_1', a_2', \ldots, a_{m+1}'$ and the result of b^*'s insertion is $b_1', b_2', \ldots, b_{m+1}'$, then we have, that $\forall l \in \{1, \ldots, m+1\}$ $\sum_{i=1}^{l} a_i' \ge \sum_{i=1}^{l} b_i'$.

Proof: Denote right and left hand sides of inequalities before (after) insertion as A_l and B_l (A_l' and B_l') respectively. Let p and q be positions of a^* and b^* in the new sequences. Assume that $l < \min\{p, q\}$, then inequalities hold since *none* of the inserted values lie in the prefix of length l, consequently, $A_l' = A_l$ and $B_l' = B_l$. If $l \ge \max\{p, q\}$ then *both* inserted values lie in the prefix of length l, thus we have $A_l' = A_{l-1} + a^*$ and $B_l' = B_{l-1} + b^*$, and it is easy to see that $A_l' \ge B_l'$. The remaining case splits into two subcases (see Figure 3).

$\mathbf{p \le q}$. Denote $X = \sum_{i=p+1}^{l} a_i'$. See next that $A_l' = A_{p-1} + a^* + X$, $B_l' = B_{l-1} + b_l'$, and also $A_{p-1} + X = A_{l-1} \ge B_{l-1}$. Due to nondecreasing order: $a^* \ge b^* \ge b_l'$, and we easily get the required $A_l' \ge B_l'$.

$\mathbf{q < p}$. Denote $Y = \sum_{i=p+1}^{l} b_i'$. This gives us: $A_l' = A_l$, $B_l' = B_{p-1} + b^* + Y$, and we have $A_l \ge B_l = B_{p-1} + Y + b_{l+1}'$. Again, due to nondecreasing order: $b_{l+1}' \ge b^*$, and claimed inequality can be easily derived.

■

We now prove the crucial lemma for this upper bound.

Lemma 14: There exist an algorithm OPT that works no worse than the optimal algorithm on any sequence of inputs and such a choice of $\mathrm{Free}^{\mathrm{OPT}}$, that on every sequence of inputs at every time moment it holds that:

(1) $|\mathrm{IBT}_1^{\mathrm{PQ}}| \ge |\mathrm{IBT}_1^{\mathrm{OPT}}|$, and for all l, s.t. $l \le |\mathrm{IBT}_1^{\mathrm{OPT}}|$ it holds that $\Phi_1^{\mathrm{OPT}}(l) \ge \Phi_1^{\mathrm{PQ}}(l)$;
(2) $|\mathrm{IBT}_V^{\mathrm{PQ}}| \ge |\mathrm{IBT}_V^{\mathrm{OPT}}|$, and for all l, s.t. $l \le |\mathrm{IBT}_V^{\mathrm{OPT}}|$ it holds that $\Phi_V^{\mathrm{OPT}}(l) \ge \Phi_V^{\mathrm{PQ}}(l)$;

Proof: We prove these estimates by induction on the number of "events" such as receiving, processing, or transmitting a packet. At the initial time moment all conditions hold trivially. Note that whenever conditions (1) and (2) hold, it also necessarily holds that $|\mathrm{Trans}_v^{\mathrm{OPT}}| \le |\mathrm{Trans}_v^{\mathrm{PQ}}|$ since $\Phi_v^{\mathrm{OPT}}(l) = 0$ for all $l \le |\mathrm{Trans}_v^{\mathrm{OPT}}|$ and consequently $\Phi_v^{\mathrm{PQ}}(l) = 0$. We now remove from $\mathrm{IBT}_v^{\mathrm{PQ}}$ the $(|\mathrm{IBT}_v^{\mathrm{PQ}}| - |\mathrm{IBT}_v^{\mathrm{OPT}}|)$ packets with the lowest priority, denoting the resulting set by $\widetilde{\mathrm{IBT}}_v^{\mathrm{PQ}}$ and the corresponding set of packets in the buffer by $\widetilde{\mathrm{IB}}_v^{\mathrm{PQ}}$. Then all of the above implies that $|\mathrm{IB}_v^{\mathrm{OPT}}| \ge |\widetilde{\mathrm{IB}}_v^{\mathrm{PQ}}|$.

Upon acceptance we may mark a packet admitted to OPT buffer as "causing overflow". The set of such packets is denoted as $\mathrm{Over}^{\mathrm{OPT}}$, and it does not contribute to $\mathrm{IBT}^{\mathrm{OPT}}$. The induction step will guarantee that every packet in $\mathrm{Over}^{\mathrm{OPT}}$ is moved eventually to $\mathrm{Free}^{\mathrm{OPT}}$ and the following invariant holds: $|\mathrm{Over}^{\mathrm{OPT}}| \le |\mathrm{IB}_V^{\mathrm{PQ}}| - |\widetilde{\mathrm{IB}}_V^{\mathrm{PQ}}|$.

Let us now consider all possible events one by one and show that none of them violates the conditions of the theorem.

Arrival of a new packet $\mathbf{p}$. There are two subcases.

$\mathbf{value(p) = V}$. If PQ has accepted the packet and has pushed out a packet from $\mathrm{IB}_1^{\mathrm{PQ}}$, we move the heaviest packet from $\mathrm{IB}_1^{\mathrm{OPT}}$ (if it is nonempty) to $\mathrm{Free}^{\mathrm{OPT}}$. Thus, inequalities for Φ_1 are not violated since we have removed largest elements from both $\mathrm{IB}_1^{\mathrm{OPT}}$ and $\mathrm{IB}_1^{\mathrm{PQ}}$. Further, if OPT accepts p, then $B > |\mathrm{IB}_V^{\mathrm{OPT}}| \ge |\widetilde{\mathrm{IB}}_V^{\mathrm{PQ}}|$, so the sequence $\mathrm{IBT}_V^{\mathrm{PQ}}$ will receive the lightest of packets $(\mathrm{IB}_V^{\mathrm{PQ}} \setminus \widetilde{\mathrm{IB}}_V^{\mathrm{PQ}}) \cup \{p\}$ (according to the push-out rules). Therefore, by Lemma 13 inequalities for Φ_V still hold. The only time $|\mathrm{IB}_V^{\mathrm{PQ}}| - |\widetilde{\mathrm{IB}}_V^{\mathrm{PQ}}|$ increases is when $|\mathrm{IB}_V^{\mathrm{PQ}}| = B$ and OPT accepts, but $|\mathrm{IB}_V^{\mathrm{OPT}}| \ge |\widetilde{\mathrm{IB}}_V^{\mathrm{PQ}}|$ and $|\mathrm{Over}^{\mathrm{OPT}}| + |\mathrm{IB}_V^{\mathrm{OPT}}| < B$ together give $|\mathrm{Over}^{\mathrm{OPT}}| < |\mathrm{IB}_V^{\mathrm{PQ}}| - |\widetilde{\mathrm{IB}}_V^{\mathrm{PQ}}|$.

$\mathbf{value(p) = 1}$. We consider two subcases separately. (i) $|\mathbf{IB_V^{PQ}}| + |\widetilde{\mathbf{IB}}_\mathbf{1}^{\mathbf{PQ}}| < \mathbf{B}$. In this case, we add to $\widetilde{\mathrm{IB}}_1^{\mathrm{PQ}}$ the lightest packet from $(\mathrm{IB}_1^{\mathrm{PQ}} \setminus \widetilde{\mathrm{IB}}_1^{\mathrm{PQ}}) \cup \{p\}$ (according to push-out rules), and by Lemma 13 the inequalities are preserved. (ii) $|\mathbf{IB_V^{PQ}}| + |\widetilde{\mathbf{IB}}_\mathbf{1}^{\mathbf{PQ}}| = \mathbf{B}$. Then, since

$|\mathrm{IB}^{\mathrm{OPT}}| \geq |\widetilde{\mathrm{IB}}^{\mathrm{PQ}}|$ and $|\mathrm{IB}^{\mathrm{OPT}}|+|\mathrm{Over}^{\mathrm{OPT}}| < B$, we get that $|\mathrm{IB}_V^{\mathrm{PQ}}| - |\widetilde{\mathrm{IB}}_V^{\mathrm{PQ}}| > |\mathrm{Over}^{\mathrm{OPT}}|$. Now, if OPT accepts p then p is added to the $\mathrm{Over}^{\mathrm{OPT}}$.

OPT **processes a packet p**. There are three subcases.

- $\mathbf{value(p) = V}$. This is a simple case. If $\mathrm{IB}_V^{\mathrm{PQ}} \neq \emptyset$ then each nonzero term in $\Phi_V^{\mathrm{PQ}}(l)$ reduces exactly by one, while each nonzero term in $\Phi_V^{\mathrm{OPT}}(l)$ reduces by at most one, so the inequalities are obviously preserved. If otherwise $\mathrm{IB}_V^{\mathrm{PQ}} = \emptyset$ then all $\Phi_V^{\mathrm{PQ}} = 0$.
- $\mathbf{value(p) = 1}$ **and** $\mathbf{IB_Y^{PQ} = \emptyset}$. Similar to the previous.
- $\mathbf{value(p) = 1}$ **and** $\mathbf{IB_V^{PQ} \neq \emptyset}$. In this case, p is sent to $\mathrm{Free}^{\mathrm{OPT}}$. It remains to note that $\Phi_1^{\mathrm{OPT}}(l)$ do not decrease since we have merely removed an element from an ordered sequence.

Transmitting a packet. Inequalities on Φ obviously remain unchanged upon transmission; however, the value of $(|\mathrm{IB}_V^{\mathrm{PQ}}| - |\widetilde{\mathrm{IB}}_V^{\mathrm{PQ}}|)$ can decrease by one. If $\mathrm{Over}^{\mathrm{OPT}}$'s invariant is violated, we move an arbitrary packet from $\mathrm{Over}^{\mathrm{OPT}}$ to $\mathrm{Free}^{\mathrm{OPT}}$.

■

Lemma 15: The set $\mathrm{Free}^{\mathrm{OPT}}$ constructed in Lemma 14 satisfies the inequality $|\mathrm{Free}^{\mathrm{OPT}}| \leq (W+2)|\mathrm{Trans}_V^{\mathrm{PQ}}|$ after algorithms finish processing the input sequence.

Proof: It suffices to note that in the proof of Lemma 14 a packet may fall into $\mathrm{Free}^{\mathrm{OPT}}$ only when PQ receives, processes, or transmits a packet with value V. ■

Now, after both OPT and PQ have processed the entire sequence of packets, the total value of packets transmitted by PQ equals $|\mathrm{IBT}_1^{\mathrm{PQ}}| + V|\mathrm{IBT}_V^{\mathrm{PQ}}|$. The total value of packets transmitted by OPT is $|\mathrm{IBT}_1^{\mathrm{OPT}}| + V|\mathrm{IBT}_V^{\mathrm{OPT}}| + |\mathrm{Free}^{\mathrm{OPT}}|$. Thus, using the lemma's inequalities, the competitive ratio α can be bounded as follows:

$$\begin{aligned}\alpha &\leq \frac{|\mathrm{IBT}_1^{\mathrm{OPT}}| + V|\mathrm{IBT}_V^{\mathrm{OPT}}| + |\mathrm{Free}^{\mathrm{OPT}}|}{|\mathrm{IBT}_1^{\mathrm{PQ}}| + V|\mathrm{IBT}_V^{\mathrm{PQ}}|} \\ &\leq 1 + \frac{|\mathrm{Free}^{O}PT|}{|\mathrm{IBT}_1^{\mathrm{PQ}}| + V|\mathrm{IBT}_V^{\mathrm{PQ}}|} \leq 1 + \frac{|\mathrm{Free}^{O}PT|}{V|\mathrm{IBT}_V^{\mathrm{PQ}}|} \\ &\leq 1 + \frac{(W+2)|\mathrm{IBT}_V^{\mathrm{PQ}}|}{V|\mathrm{IBT}_V^{\mathrm{PQ}}|} \leq 1 + \frac{W+2}{V}.\end{aligned}$$

■

Corollary 16: If $W < V$, $\mathrm{PQ}_{v,-w}$ and $\mathrm{PQ}_{v/w}$ are at most $\left(2 + \frac{2}{V}\right)$-competitive.

Proof: Since $(W \mid V)$ pushes out $(1 \mid 1)$ in the $\mathrm{PQ}_{v/w}$ queue, any packet with value V pushes out any packet with value 1, so for $W < V$ $\mathrm{PQ}_{v/w}$ is equivalent to $\mathrm{PQ}_{v,-w}$. ■

Figure 4 shows a contour plot of the $\mathrm{PQ}_{v,-w}$ competitive ratio upper bound $1 + \frac{W+2}{V}$ for the two-valued case; naturally, for large V the bound is very good.

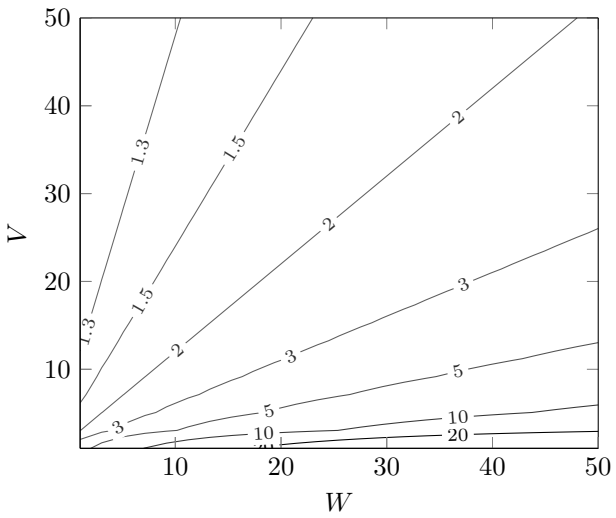

Fig. 4. Contour plot of the $\mathrm{PQ}_{v,-w}$ competitive ratio $1 + \frac{W+2}{V}$ as a function of W and V.

VII. The β-push-out case

In many networking systems, there arises an additional motivation to avoid pushouts and prioritize packets that are already in the buffer. For instance, newly admitted packets incur higher costs than packets that have already resided in the buffer since they require more access bandwidth to packet memories: a new packet incurs computational costs for constructing and updating the corresponding data structures in the network processor, and immediate push-out of a less-preferable packet can lead to increased computational overhead [14].

To represent these effects in the formal model, Keslassy et al. [14] introduced the notion of *copying cost* in the performance of transmission algorithms for packets with heterogeneous processing requirements but uniform values: if an algorithm accepts A packets and transmits packets with total value T, its transmitted value is $\max\{0, T - \alpha A\}$, i.e., each admitted packet incurs a cost α subtracted from the throughput in the objective function. Thus, in extreme cases the transmitted value of a push-out policy may even go down to zero; copying cost provides an additional control on the number of pushed out packets to avoid pathological cases. To implement such a control mechanism, Keslassy et al. [14] introduced the greedy push-out work-conserving policy PQ_β that processes a packet with minimal required work first and in the case of congestion such a policy pushes out only if a new arrival has at least β times less work then the maximal residual work in PQ_β.

However, the work [14] only dealt with a single packet characteristic, namely processing requirements. To generalize their ideas to our problem settings with two characteristics, we extend PQ_f to PQ_f^β and then show several lower bounds for β-push-out counterparts of our policies. Unfortunately, the proof of Theorem 12 cannot be directly applied to β-push-out policies; it remains an interesting open problem to show nontrivial (less than linear) upper bounds for β-push-out policies even in the two-valued case.

Definition 7.1: Let f be a function of packets, $f(w, v) \in \mathbb{R}$, with better packets corresponding to larger values of f. The PQ_f^β processing policy for $\beta > 1$ is defined as PQ_f with the following difference: PQ_f^β can push out a packet p and add a new packet p' to the queue at time slot t if p is currently the worst packet in the buffer and p' is better than p at least by a factor of β: $f(p) = \min_{q \in \mathrm{IB}^{\mathrm{PQ}_f}} f(q)$, and $f(p') > \beta f(p)$.

Theorem 17: Consider a buffer of size B with maximal required processing W and maximal packet value V. Then:

(1) $\mathrm{PQ}^{\beta}_{-w,v}$ is at least V-competitive both in the case of arbitrary packet values and in the two-valued case;
(2) $\mathrm{PQ}^{\beta}_{v/w}$ is at least $\min\{V, W\}$-competitive in the case of arbitrary packet values;
(3) in the two-valued case, if $\beta W \geq V$ then $\mathrm{PQ}^{\beta}_{v/w}$ is at least V-competitive;
(4) $\mathrm{PQ}^{\beta}_{v,-w}$ is at least $\left(\frac{(V-1)}{V}W - o(1)\right)$-competitive in the case of arbitrary packet values and at least $\left(\frac{W}{V} + o(1)\right)$-competitive in the two-valued case.

Proof: (1) In the construction from Theorem 1, packets are never pushed out from $\mathrm{PQ}_{-w,v}$ buffer, so the result still holds for $\mathrm{PQ}^{\beta}_{-w,v}$. (2) Construction from Theorem 3 also works because packets are never pushed out from $\mathrm{PQ}_{v/w}$ buffer in tnis construction. (3) This result can be seen as a relaxation of the second part of Theorem 11 since $\beta > 1$. The same construction works: $\mathrm{PQ}_{v/w}$ fills its buffer with $B \times (1 \mid 1)$ and then drops incoming $B \times (W \mid V)$. $\mathrm{PQ}^{\beta}_{v/w}$ also drops them since $\frac{V}{W} \leq \frac{\beta V}{V} = \beta$. (4) Again, the constructions from Theorem 2 and Theorem 11 work since they do not force packets to be pushed out from $\mathrm{PQ}_{-v,w}$ buffer. ∎

VIII. Simulations

In this section, we present the results of a comprehensive simulation study intended to validate our theoretical results. Naturally, it would be desirable to compare the proposed algorithms on real life network traces. Unfortunately, available datasets such as CAIDA [10] are of little use for packet characteristics in our model since they do not provide data on required processing and intrinsic values of the packets. Nevertheless, we have used CAIDA traces [10] to model the incoming stream of packets, breaking down the timestamps into equal timeslots and counting the packets in each timeslot; hence, the intensity of the incoming stream below is measured in milliseconds, the size of a single timeslot.

We have conducted six series of experiments, studying how performance depends on maximal required processing W, buffer size B, maximal value V, size of a CAIDA trace timeslot t, and β (in the model with copying cost). The actual optimal online algorithm in our model would be computationally prohibitive, so to estimate and compare the competitive ratios of our algorithms we have used an algorithm which is actually better than optimal: a single priority queue with size BW that breaks each packet $(v \mid w)$ into "fractional" packets that each have required work 1 and value $\frac{v}{w}$ and then orders and processes them by this value. Since the priority queue has been proven optimal in the model with values and no required processing, it performs even better than optimal.

In our experiments, the values and processing ratios of packets were chosen uniformly from $\{1, \ldots, W\}$ and $\{1, \ldots, V\}$ respectively. We ran all experiments for $5 \cdot 10^5$ time slots with periodic "flushouts" (wait for all queues to finish their packets and then continue from an empty state), which in our experiments has proven to be sufficient for stable results. We have also performed simulations without flushouts; since the results are very close to the ones with flushouts in all settings, we do not show them separately. Note that all of our experiments venture into the values of parameters that yield high system load with large dropout rates for all algorithms; these are precisely the situations where we would like to compare performance since without heavy load and frequent congestion all reasonable algorithms perform identically. We have made the code for our experimental evaluation publicly available at GitHub [30].

Figure 5 shows simulation results presented in terms of the fraction of successfully transmitted packets: each graph shows the "better than optimal" reference algorithm in black alongside with the ratio of transmitted packets for other policies. There are five sets of experiments corresponding to the rows of Fig. 5 that will be described in subsections below; we have tested the four algorithms used in this work: $\mathrm{PQ}_{v/w,-w}$, $\mathrm{PQ}_{v/w,v}$, $\mathrm{PQ}_{-w,v}$, and $\mathrm{PQ}_{v,-w}$. Note that in all cases, $\mathrm{PQ}_{v/w,-w}$ and $\mathrm{PQ}_{v/w,v}$ are virtually indistiguishable across all settings. Thus, below we will sometimes refer to them collectively as $\mathrm{PQ}_{v/w}$.

A. Maximal required processing

In the first set of simulations (Fig. 5(1-3)), we study performance as a function of the maximal required processing W. As W grows, all algorithms deteriorate in absolute terms (packets become heavier), but it is clear that $\mathrm{PQ}_{-w,v}$, which pays more attention to required processing, fares better while $\mathrm{PQ}_{v,-w}$ loses badly as W grows. This is expected since $\mathrm{PQ}_{v,-w}$ cares little about W and therefore is likely to get stuck with very heavy packets. We see that $\mathrm{PQ}_{v/w}$ is uniformly the best policy, performing very close to OPT and deteriorating only slightly.

B. Buffer size B

In the second set of simulations (Fig. 5(4-6)), we study performance as a function of the buffer size B.

In this setting as well, $\mathrm{PQ}_{v/w,-w}$ and $\mathrm{PQ}_{v/w,v}$ remain indistinguishable, and since these experiments were done in the relatively low ranges of the W/V ratio, $\mathrm{PQ}_{v,-w}$ is also very close to $\mathrm{PQ}_{v/w}$. $\mathrm{PQ}_{v,-w}$, on the other hand, is able to store, in a larger buffer, more high-value packets and do so for longer, so as B increases and congestion decreases, $\mathrm{PQ}_{v,-w}$ becomes closer to the other three. Note, in this setting, all algorithms become significantly worse off compared to the fractional OPT through no fault of their own: the "unfairness" of fractional OPT becomes much more pronounced with large B (it has more and more extra space to store packets).

C. Maximal value V

In the third set of experiments (Fig. 5(7-9)), we look at performance as a function of the maximal value V. It turns out that while the relative performance to fractional OPT drops, the performance level of the three leading algorithms does not significantly depend on V in realistic cases, and only $\mathrm{PQ}_{v,-w}$ drops significantly in relative quality. This, again, can be explained by the fact that $\mathrm{PQ}_{v,-w}$ suffers from a wider variety of packets, getting stuck with valuable yet heavy ones. The relative order of algorithms remains unchanged.

D. Incoming stream intensity

The fourth set of experiments (Fig. 5(10-12)) shows how performance depends on the intensity of the packet source, expressed in terms of the timeslot size t (naturally, more packets on average arrive during a longer t). This setting lets us explore the most congested settings: all algorithms join together at the high end of intensity simply because now there are, on average, enough $(1 \mid V)$ packets arriving to keep all algorithms busy only with the obviously best packets. The relative standings of all algorithms remain the same throughout this increasing congestion.

E. β for β-push-out policies

The fifth set of experiments (Fig. 5(13-15)) studies a different situation; here, we have introduced nonzero copying cost α (on all three graphs, $\alpha = 0.3$) and have studied how performance depends on β for β-push-out counterparts of our policies (as introduced in Section VII); since the number of admitted packets is not well defined for our fractional OPT, OPT did not participate in these experiments; we have taken the results of $\mathrm{PQ}_{v/w,-w}$ for $\beta = 1$ as the starting point, dividing all the rest by this value. We see that in all cases, $\beta = 1$ appears to be the perfect or almost perfect choice in practice: sometimes $\beta = 1.2$ or $\beta = 1.3$ yield better results, but only slightly.

F. W for the two-valued case

The last, sixth set of experiments (Fig. 5(16-18)) deals with the two-valued case, when the intrinsic value of a packet can only take values in $\{1, V\}$ while required work can still be an arbitrary integer from 1 to W. We repeated the experiments from Section VIII-A with this additional restriction, and the results closely match our theoretical results from Section VI: contrary to the general case, now $\mathrm{PQ}_{v,-w}$ is not the obviously worst algorithm but performs on par with $\mathrm{PQ}_{v/w}$ policies, while $\mathrm{PQ}_{-w,v}$ diverges from them for larger W in exactly the same way as in the general case (compare to Section VIII-A, Fig. 5(1-3)). Since $\mathrm{PQ}_{v,-w}$ may be easier to implement than $\mathrm{PQ}_{v/w}$ (required work does not have to be considered or even known), for the two-valued case we recommend to use $\mathrm{PQ}_{v,-w}$. Again, as a side effect we see that the fractional OPT performs better (relative to other algorithms) when more buffer space is provided.

To summarize, in this section we have shown a comprehensive simulations study on synthetic traces. The main result is that the $\mathrm{PQ}_{v/w}$ policy that we have introduced in this work is uniformly the best policy across all tested settings, and there is little difference between tie-breaking variations of it, while in the two-valued case experimental results supported the theoretical conclusion that $\mathrm{PQ}_{v,-w}$ is a good policy.

IX. Conclusion

In this work, we have begun the study of buffer management for processing packets with two different characteristics: *processing requirement* and *value*. In these settings we have considered a single queue buffering architecture and have mostly studied algorithms based on priority queues; in the setting with two characteristics, there may be different reasonable priority queues that have different policies. We have investigated various packet processing orders and found that they have linear lower bounds on the competitive ratio, which makes them unattractive in the general case. However, we have provided positive results in the special case of two different values, 1 and V and heterogeneous processing requirements.

Processing policy	**General case**	**Two-valued case**	
Adversarial general lower bounds			
Any online algorithm	$\frac{\min\{\sqrt[2B]{W},\, \sqrt[B]{V}\}-1}{2}$	$1 + \frac{V-1}{V^2} - O\left(\frac{1}{W}\right)$	
Any priority queue	$\sqrt{W}$	$\min\{V, W/V\}$	
Any FIFO online algorithm	$\left(\frac{\sqrt{W}}{B} + 1 - \frac{1}{B}\right)$	$\frac{V+B-1}{W+B-1}$	
Lower and upper bounds for specific algorithms			
	Lower bound	Lower	Upper
$\mathrm{PQ}_{-w,v}$, $\mathrm{PQ}^{\beta}_{-w,v}$	V	V	V
$\mathrm{PQ}_{v,-w}$, $\mathrm{PQ}^{\beta}_{v,-w}$	$\frac{W(V-1)}{V} - o(1)$	$\frac{W}{V} + o(1)$	$1 + \frac{W+2}{V}$
$\mathrm{PQ}_{v/w}$, $W \geq V$	V	V	
$\mathrm{PQ}^{\beta}_{v/w}$, $\beta W \geq V$	V	V	
$\mathrm{PQ}_{v/w}$, $W < V$	W	$\frac{W}{V} + o(1)$	$2 + \frac{2}{V}$

TABLE I
RESULTS SUMMARY: LOWER AND UPPER BOUNDS.

The results of our work are summarized in Table I; note that all algorithms in the table employ push-out (albeit with different heuristics for it). In the main result of this work, we have shown a $(1 + (W + 2)/V)$ upper bound for the buffer management policy $\mathrm{PQ}_{v,-w}$ that orders packets first by value and then by required processing. For $W < V$, this also becomes a constant upper bound on the competitive ratio of $\mathrm{PQ}_{v/w}$ which orders packets by unit processing (ratio of value to processing). This result has been somewhat counterintuitive since the intuition would be that the $\mathrm{PQ}_{v/w}$ policy that optimizes for value per timeslot would be best, but in the general two-valued case it has a non-competitive lower bound.

In addition, we have shown a number of general lower bounds, for the cases of any deterministic online algorithm with FIFO processing and transmission order, for any priority queue, and even for any deterministic online algorithm at all; while these lower bounds are relatively weak, they are non-constant and show that it is impossible to achieve constant upper bounds in these cases without additional assumptions on the relations between parameters such as B, V, and W.

For the two-valued case, we have shown tightly matching lower and upper bounds on the competitive ratio (they differ by $1+o(1)$). It still remains an interesting open problem to prove upper bounds for the general case of arbitrary values; another interesting problem would be to prove upper bounds for β-push-out policies. However, the really crucial question here is whether there exists a processing policy with better than linear competitive ratio for the general case of two characteristics: our general lower bounds are not constant but they are far from linear too. We suggest this problem for further study.

Acknowledgements: We thank both IEEE INFOCOM 2015 and *IEEE/ACM Transactions on Networking* reviewers for their insightful comments. This work was supported by the Russian Science Foundation grant 17-11-01276 "Networking and distributed systems and algorithms and related fundamental problems".

REFERENCES

[1] William Aiello, Alexander Kesselman, and Yishay Mansour. Competitive buffer management for shared-memory switches. *ACM Transactions on Algorithms*, 5(1), 2008.

[2] William Aiello, Yishay Mansour, S. Rajagopolan, and Adi Rosén. Competitive queue policies for differentiated services. In *INFOCOM*, pages 431–440, 2000.

[3] Nir Andelman. Randomized queue management for diffserv. In *SPAA*, pages 1–10, 2005.

[4] Nir Andelman and Yishay Mansour. Competitive management of non-preemptive queues with multiple values. In *DISC*, pages 166–180, 2003.

[5] Nir Andelman, Yishay Mansour, and An Zhu. Competitive queueing policies for QoS switches. In *SODA*, pages 761–770, 2003.

[6] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[7] Matthias Englert and Matthias Westermann. Considering suppressed packets improves buffer management in qos switches. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 209–218, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[8] Matthias Englert and Matthias Westermann. Lower and upper bounds on FIFO buffer management in QoS switches. *Algorithmica*, 53(4):523–548, 2009.

[9] Patrick Th. Eugster, Kirill Kogan, Sergey I. Nikolenko, and Alexander Sirotkin. Shared memory buffer management for heterogeneous packet processing. In *IEEE 34th International Conference on Distributed Computing Systems, ICDCS 2014, Madrid, Spain, June 30 - July 3, 2014*, pages 471–480, 2014.

[10] CAIDA The Cooperative Association for Internet Data Analysis. [Online] http://www.caida.org/.

[11] Michael Goldwasser. A survey of buffer management policies for packet switches. *SIGACT News*, 41(1):100–128, 2010.

[12] Ellen L. Hahne, Alexander Kesselman, and Yishay Mansour. Competitive buffer management for shared-memory switches. In *SPAA*, pages 53–58, 2001.

[13] B. Hajek. On the competitiveness of on-line scheduling of unit-length packets with hard deadlines in slotted time. In *In Proceedings of the 2001 Conference on Information Sciences and Systems*, 2001.

[14] Isaac Keslassy, Kirill Kogan, Gabriel Scalosub, and Michael Segal. Providing performance guarantees in multipass network processors. *IEEE/ACM Trans. Netw.*, 20(6):1895–1909, 2012.

[15] Alexander Kesselman, Kirill Kogan, and Michael Segal. Improved competitive performance bounds for CIOQ switches. In *ESA*, pages 577–588, 2008.

[16] Alexander Kesselman, Kirill Kogan, and Michael Segal. Packet mode and QoS algorithms for buffered crossbar switches with FIFO queuing. *Distributed Computing*, 23(3):163–175, 2010.

[17] Alexander Kesselman, Zvi Lotker, Yishay Mansour, Boaz Patt-Shamir, Baruch Schieber, and Maxim Sviridenko. Buffer overflow management in QoS switches. *SIAM Journal on Computing*, 33(3):563–583, 2004.

[18] Alexander Kesselman, Zvi Lotker, Yishay Mansour, Boaz Patt-Shamir, Baruch Schieber, and Maxim Sviridenko. Buffer overflow management in qos switches. *SIAM Journal on Computing*, 33(3):563–583, 2004.

[19] Alexander Kesselman and Yishay Mansour. Loss-bounded analysis for differentiated services. *J. Algorithms*, 46(1):79–95, 2003.

[20] Alexander Kesselman and Yishay Mansour. Harmonic buffer management policy for shared memory switches. *Theor. Comput. Sci.*, 324(2-3):161–182, 2004.

[21] Alexander Kesselman, Yishay Mansour, and Rob van Stee. Improved competitive guarantees for QoS buffering. *Algorithmica*, 43(1-2):63–80, 2005.

[22] Kirill Kogan, Alejandro López-Ortiz, Sergey I. Nikolenko, Gabriel Scalosub, and Michael Segal. Balancing work and size with bounded buffers. In *Sixth International Conference on Communication Systems and Networks, COMSNETS 2014, Bangalore, India, January 6-10, 2014*, pages 1–8, 2014.

[23] Kirill Kogan, Alejandro López-Ortiz, Sergey I. Nikolenko, and Alexander Sirotkin. Multi-queued network processors for packets with heterogeneous processing requirements. In *Proceedings of the 5^{th} International Conference on Communication Systems and Networks (COMSNETS 2013)*, pages 1–10, 2013.

[24] Kirill Kogan, Alejandro López-Ortiz, Sergey I. Nikolenko, and Alexander V. Sirotkin. A taxonomy of semi-FIFO policies. In *Proceedings of the 31^{st} IEEE International Performance Computing and Communications Conference (IPCCC 2012)*, pages 295–304, 2012.

[25] Kirill Kogan, Alejandro López-Ortiz, Sergey I. Nikolenko, and Alexander V. Sirotkin. Online scheduling FIFO policies with admission and push-out. *Theory Comput. Syst.*, 58(2):322–344, 2016.

[26] Fei Li. A near-optimal memoryless online algorithm for fifo buffering two packet classes. *Theoretical Computer Science*, 497:164 – 172, 2013.

[27] Hongqiang Harry Liu, Xin Wu, Ming Zhang, Lihua Yuan, Roger Wattenhofer, and David A. Maltz. zupdate: updating data center networks with zero loss. In *SIGCOMM*, pages 411–422, 2013.

[28] Zvi Lotker and Boaz Patt-Shamir. Nearly optimal FIFO buffer management for two packet classes. *Comp. Netw.*, 42(4):481–492, 2003.

[29] Yishay Mansour, Boaz Patt-Shamir, and Ofer Lapid. Optimal smoothing schedules for real-time streams. *Distrib. Comp.*, 17(1):77–89, 2004.

[30] Sergey I. Nikolenko. Code for simulation experiments. http://github.com/snikolenko/sim-twocharacteristics.

[31] Sergey I. Nikolenko and Kirill Kogan. Single and multiple buffer processing. In *Encyclopedia of Algorithms*, pages 1988–1994. Springer, 2016.

[32] George Porter, Richard D. Strong, Nathan Farrington, Alex Forencich, Pang-Chen Sun, Tajana Rosing, Yeshaiahu Fainman, George Papen, and Amin Vahdat. Integrating microsecond circuit switching into the data center. In *SIGCOMM*, pages 447–458, 2013.

[33] Kirk Pruhs. Competitive online scheduling for server systems. *SIGMETRICS Performance Evaluation Review*, 34(4):52–58, 2007.

[34] Nourhan Sakr and Cliff Stein. An empirical study of online packet scheduling algorithms. In *Proceedings of the 15th International Symposium on Experimental Algorithms - Volume 9685*, SEA 2016, pages 278–293, New York, NY, USA, 2016. Springer-Verlag New York, Inc.

[35] Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update and paging rules. *Comm. ACM*, 28(2):202–208, 1985.

[36] Tilman Wolf, Prashanth Pappu, and Mark A. Franklin. Predictive scheduling of network processors. *Comp. Netw.*, 41(5):601–621, 2003.

[37] An Zhu. Analysis of queueing policies in QoS switches. *J. Algorithms*, 53(2):137–168, 2004.

Pavel Chuprikov Pavel Chuprikov is a research assistant at the IMDEA Networks Institute, Spain and a Ph.D. student at the Steklov Institute of Mathematics at St. Petersburg, Russia. Previously, he worked as a software developer at JetBrains Research. Pavel received his M.Sc from St. Petersburg Academic University of Russian Academy of Sciences. His research interests include software defined networking, online algorithm design, and dependent types.

Alex Davydow Alex Davydow is a researcher at the Steklov Institute of Mathematics at St. Petersburg. He obtained his M.Sc. from the St. Petersburg Academic University and graduated from its Ph.D. studies. His research interests includes networking algorithms and systems, tropical algebraic geometry and its application to scheduling algorithms.

Sergey Nikolenko Sergey Nikolenko is a researcher at the Steklov Institute of Mathematics at St. Petersburg (PDMI RAS). He received his M.Sc. summa cum laude from St. Petersburg State University (2005); Ph.D., from PDMI RAS (2009). His research interests include networking algorithms and systems, machine learning and probabilistic inference, bioinformatics, and theoretical CS, with projects funded by major companies and research funds such as RSF, CRDF, INTAS, Mail.Ru, RFBR, RAS, and others.

Kirill Kogan Kirill Kogan is a Research Assistant Professor at IMDEA Networks Institute. He received his PhD from Ben-Gurion University (Israel) at 2012. He was a Technical Leader at Cisco Systems, where he worked in 2000-2012. He was a Postdoctoral Fellow at University of Waterloo and Purdue University during 2012-2014. His current research interests are in design, analysis, and implementation of networked systems, broadly defined.

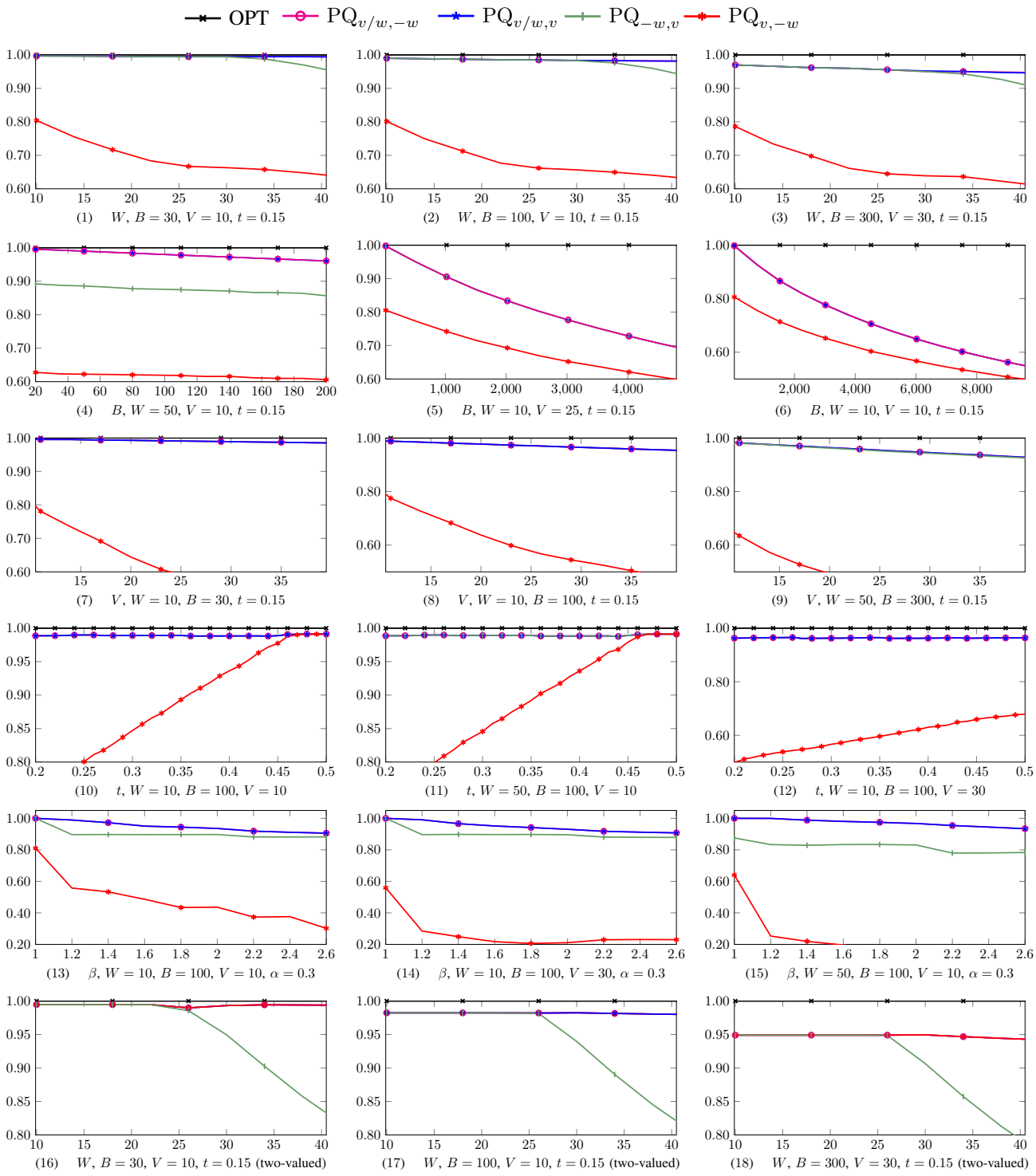

Fig. 5. Simulation results: share of successfully transmitted packets in the required processing model as a function of (1-3) maximal required processing W, (4-6) buffer size B, (7-9) maximal value V, (10-12) timeslot size t used for the CAIDA traces, (13-15) β parameter for β-push-out policies, (14-16) maximal required processing W in the two-valued case. Specific simulation parameters are shown in graph captions.