Priority Queueing with Multiple Packet Characteristics

Pavel Chuprikov
St. Petersburg Academic University,
St. Petersburg, Russia

Sergey Nikolenko[‡§]
[‡]National Research University
Higher School of Economics,
St. Petersburg, Russia
[§]Steklov Mathematical Institute
at St. Petersburg, Russia

Kirill Kogan
IMDEA Networks Institute,
Madrid, Spain

***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 an adversarial lower bound that holds 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 [23], [26]. 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], [28], 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). 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 sort 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

https://doi.org/10.1109/INFOCOM.2015.7218519

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 $\left(1 + \frac{W+2}{V}\right)$. 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. Section VIII presents simulation results where the proposed algorithms are evaluated with synthesized traces. Section IX concludes the paper.

II. Model Description

We use a model similar to the one introduced in [1], [17] and subsequently used in [12]–[14], [20]–[22]. Consider a single queue of size B that handles the arrival of a sequence of unit-sized packets. 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 (work) $w(p) \in \{1, \ldots, W\}$ and (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 [29], 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], [30], while for $V = 1$ it becomes a single queue of unit-valued packets with different required processing, as considered, e.g., in [19]–[21]. 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.

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 $\mathrm{IB}_t^{\mathrm{ALG}}$ 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). 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 (it may be reasonable only with non-zero copying costs). 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], [30] 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], [30] 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 [15] to be at least 1.282 and at most 2-competitive. Later, the upper bound on the greedy policy was improved to 1.894 [16]; this work also considers the β-push-out case and proves that the greedy policy is at least 1.544-competitive. Policies with memory have been considered in [3], [7], [24].

Recently, packets with required processing but with uniform packet values in various settings have been considered in [8], [12]–[14], [20]–[22]. 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 sorts packets according to their required processing is known to be optimal [12]. 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], [15], [25], this line of research has received tremendous attention over the past decade. A survey by Goldwasser [10] provides an excellent overview of this field. Pruhs [27] 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.

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], [11]. 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 sorts 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 [18], with a stronger bound of 1.434 for the special case when $B = 2$ if all possible values are admissible [5], [30]. In the two-valued case, tight bound 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 $B \to \infty$ [7]. 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 $\frac{1}{2}(k + 1)$ for greedy non-push-out policies [21].

IV. Algorithms and Lower Bounds

Previous research indicates that *priority queues* with push-out are the best tools that often lead to good competitive ratios. 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 [12]. Therefore, our usual suspects are priority queues; in case of multiple characteristics, there can be different priority queues that sort packets in different orders. 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;
- PQ_f sorts 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 $f(p') > f(p)$.

In other words, PQ_f sorts and processes packets according to the function f. In particular, we consider three specific priority queues (here w denotes the current residual work):

(1) $\mathrm{PQ}_{-w,v} = \mathrm{PQ}_{-w+v/(V+1)}$ sorts packets in the increasing order of their required processing, breaking ties by value;
(2) $\mathrm{PQ}_{v,-w} = \mathrm{PQ}_{v-w/(W+1)}$ sorts packets in the decreasing order of their value, breaking ties by required processing;
(3) $\mathrm{PQ}_{v/w}$ sorts 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. 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 will unite them under the same notation.

Fig. 1 shows a sample time slot of these priority queues; all policies start with $(5 \mid 2)$, $(4 \mid 3)$, and $(1 \mid 1)$ in their queues, $B = 3$, 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 $2/5$.

Our main result in this section is that priority queues fail to provide constant or even logarithmic competitiveness with two packet characteristics, as they do in simpler cases. We show linear (in V and/or W) lower bounds on the competitive ratio of all three PQ policies. This is an interesting and somewhat discouraging result since priority queues have proven to be efficient (often with constant upper bounds on the competitive ratio) in other contexts. For a lower bound, it suffices to present a hard sequence of packets on which the optimal algorithm outperforms the one in question; we show the first proof as an example but omit the others due to space constraints.

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 $B \times (1 \mid 1)$ (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. ■

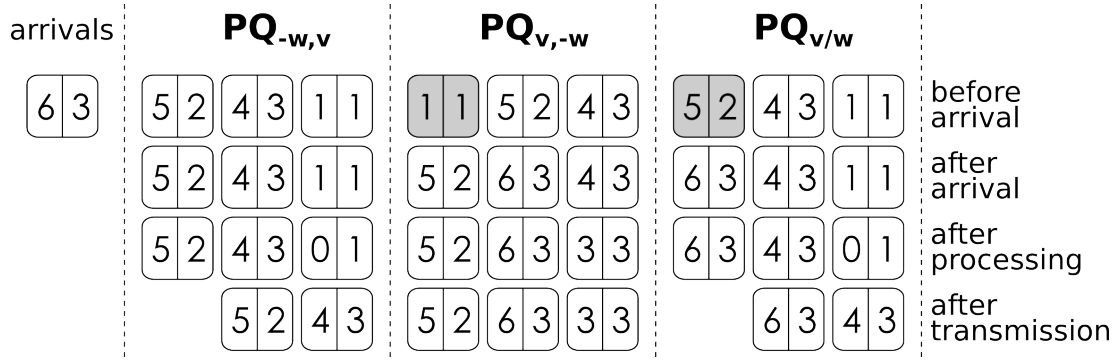

Fig. 1. A sample time slot of $\mathrm{PQ}_{-w,v}$, $\mathrm{PQ}_{v,-w}$, and $\mathrm{PQ}_{v/w}$.

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.

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.

V. General lower bound

We have already mentioned related general lower bounds for online algorithms in Section III. It turns out that in our case, when both required processing and value are allowed to vary, it becomes possible again to prove an adversarial lower bound on all online policies. We first show it for general values and then consider the two-valued case (which in this case turns out to be a generalization).

Theorem 4: For a buffer of size B, maximal packet required processing W, and maximal packet value $V \geq 2$, every online algorithm ALG is at least $\left(\frac{5}{4} - O(1/W)\right)$-competitive.

Proof: To show a lower bound, we present the corresponding hard instance. On the first step, there arrive $B \times (1 \mid 1)$ and $B \times (W \mid 2)$. ALG is forced to choose how many packets of each kind it accepts. Denote the number of $(W \mid 2)$ packets accepted by ALG by n. There are two cases.

- $\mathbf{n < \frac{3}{5}B}$: in this case, OPT chooses to accept $B \times (W \mid 2)$, and no new packets arrive. After BW time slots, OPT has processed packets with total value $2B$, and ALG has processed at most $(B-n)+2n$. The competitive ratio thus becomes $\frac{2B}{B+n} = \frac{2}{1+n/B} \geq \frac{5}{4}$ for $\frac{n}{B} < \frac{3}{5}$.
- $\mathbf{\frac{n}{B} \geq \frac{3}{5}}$: in this case, OPT accepts $B \times (1 \mid 1)$. After B time slots, OPT has processed all of its packets with total value B, while ALG has not been able to process more than $(B-n) \times (1 \mid 1)$ (by assumption) and no more than $B/W \times (W \mid 2)$ (by the time constraint). At this point, $B \times (1 \mid 2)$ arrive and then no more packets; regardless of whether ALG accepts any of them, after all buffers have been emptied OPT gets $2B$ more value and ALG no more than $2B$. The ratio is thus at least $\frac{B+2B}{(B-n)+2B/W+2B} = \frac{3}{3-n/B} - O(1/W) \geq \frac{5}{4} - O(1/W)$ for $\frac{n}{B} \geq \frac{3}{5}$.

■

In the two-valued case, the bound becomes worse; note that the bound depends on the minimal available packet value larger than 1; it does not matter what values more than V may be available.

Theorem 5: For a buffer of size B, maximal packet required processing W, and available packet values 1 and $V > 1$, every online algorithm ALG is at least $\left(1 + \frac{V-1}{V^2} - O\left(\frac{1}{W}\right)\right)$-competitive.

Proof: Similar to Theorem 4, on the first step there arrive $B \times (1 \mid 1)$ and $B \times (W \mid V)$. Suppose that ALG has accepted n of $(W \mid 2)$ 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 $\frac{VB}{B-n+Vn} = \frac{V}{1+(V-1)\frac{n}{B}} \geq \frac{V^2+V-1}{V^2}$ for $\frac{n}{B} < \frac{V^2-1}{V^2+V-1}5$.
- $\mathbf{n \geq \frac{V^2-1}{V^2+V-1}B}$: 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 gives us the ratio $\frac{B+VB}{(B-n)+VB/W+VB} = \frac{V+1}{V+1-n/B} - O(1/W) \geq \frac{V^2+V-1}{V^2} - O(1/W)$ for $\frac{n}{B} \geq \frac{V^2-1}{V^2+V-1}$.

■

In the next section, we look at the case of two possible values, 1 and V, in more detail. In the main result of this work, we show that in this special case some priority queues do admit an interesting upper bound on their competitive ratio.

VI. Upper bound for the two-valued case

Although the results of Section IV do not leave much hope about the general case, we can still prove positive results in special cases. In this section, we consider a special case when there are only two possible values, 1 and V, so there are two kinds of packets, $(w \mid 1)$ and $(w \mid V)$; the required processing can still be arbitrary 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 [20]. We will show in Theorem 7 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 $W < V$. However, we begin with negative results; Theorem 6 provides matching tight lower bounds for the main result that follows; again, proofs of lower bounds are omitted due to lack of space.

Theorem 6: 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 $W \geq V$ 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.

In the next theorem, we show the main result of this work, an upper bound on the competitive ratio of $\mathrm{PQ}_{v,-w}$.

Theorem 7: 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:

(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) $\mathrm{Trans}_v^{\mathrm{ALG}} = \{p \text{ transmitted by ALG} : \mathrm{value}(p) = v\}$ 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; $\mathrm{IBT}_v^{\mathrm{ALG}} = \bigcup_v \mathrm{IBT}_v^{\mathrm{ALG}}$.

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 let OPT transmit certain packets immediately, "for free", thus improving its throughput. These packets do not fall into $\mathrm{Trans}^{\mathrm{OPT}}$ but comprise a separate set $\mathrm{Free}(\mathrm{OPT})$. Throughout the proof, we will preserve the property that $\forall p \in \mathrm{Free}(\mathrm{OPT})\ \mathrm{value}(p) = 1$, i.e., we only give out packets of value 1 for free. We begin with a technical statement (proof omitted due to space constraints).

Lemma 8: 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 $\forall l \in \{1, \ldots, m\}\ \sum_{i=1}^{l} a_i \geq \sum_{i=1}^{l} b_i$. Let also a^* and b^* be any two numbers, such that $a^* \geq b^*$. Consider two sequences: $a'_1, a'_2, \ldots, a'_{m+1}$ and $b'_1, b'_2, \ldots, b'_{m+1}$, that result after insertion of a^* and b^* into $a_1, \ldots, a_m$ and $b_1, \ldots, b_m$ respectively (nondecreasing order preserved). Then we have, that $\forall l \in \{1, \ldots, m+1\}\ \sum_{i=1}^{l} a'_i \geq \sum_{i=1}^{l} b'_i$.

We now prove the crucial lemma of this upper bound.

Lemma 9: There exists an algorithm OPT that works no worse than the optimal algorithm on any sequence of inputs for which on every sequence of inputs at every time moment it holds that:

(1) for all $l \in \{1, \ldots, |\mathrm{IBT}_1^{\mathrm{OPT}}|\}\ \Phi_1^{\mathrm{OPT}}(l) \geq \Phi_1^{\mathrm{PQ}}(l)$;
(2) for all $l \in \{1, \ldots, |\mathrm{IBT}_V^{\mathrm{OPT}}|\}\ \Phi_V^{\mathrm{OPT}}(l) \geq \Phi_V^{\mathrm{PQ}}(l)$;
(3) $|\mathrm{Free}(\mathrm{OPT})| \leq (W+2)|\mathrm{Trans}_V(\mathrm{PQ})|$.

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 $|\mathrm{Trans}_v(\mathrm{OPT})| \leq |\mathrm{Trans}_v(\mathrm{PQ})|$ since $\Phi_v^{\mathrm{OPT}}(l) = 0$ for all $l < |\mathrm{Trans}_v(\mathrm{OPT})|$ and consequently $\Phi_v^{\mathrm{PQ}}(l) = 0$; moreover, $|\mathrm{IBT}_v^{\mathrm{OPT}}| \leq |\mathrm{IBT}_v^{\mathrm{PQ}}|$ since Φ_v must be defined. We now remove from $\mathrm{IBT}_v^{\mathrm{PQ}}$ the $(|\mathrm{IBT}_v^{\mathrm{PQ}}| - |\mathrm{IBT}_v^{\mathrm{OPT}}|)$ packets with the highest priority, denoting the resulting set by $\widetilde{\mathrm{IBT}}_v^{\mathrm{PQ}}$ and the corresponding set of packets in the buffer by $\widetilde{IB}^{\mathrm{PQ}}$. Then all of the above implies that $|\mathrm{IB}_v^{\mathrm{OPT}}| \geq |\widetilde{IB}_v^{\mathrm{PQ}}|$.

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

$$|\mathrm{overflow}(\mathrm{OPT})| \leq |\mathrm{IB}_V^{\mathrm{PQ}}| - |\widetilde{IB}_V^{\mathrm{PQ}}|.$$

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 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 largest 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 $B > |\mathrm{IB}_V^{\mathrm{OPT}}| \geq |\widetilde{IB}_V^{\mathrm{PQ}}|$, so the sequence $\mathrm{IBT}_V^{\mathrm{PQ}}$ will receive the smallest of packets $(\mathrm{IB}_V^{\mathrm{PQ}} \setminus \widetilde{IB}_V^{\mathrm{PQ}}) \cup \{p\}$ (according to the push-out rules). Therefore, by Lemma 8 inequalities for Φ_V still hold. Note, that $(|\mathrm{IB}_V^{\mathrm{PQ}}| - |\widetilde{IB}_V^{\mathrm{PQ}}|)$ could only increase.
- $\mathbf{value(p) = 1}$. 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 $\widetilde{IB}_1^{\mathrm{PQ}}$ the smallest packet from $(\mathrm{IB}_1^{\mathrm{PQ}} \setminus \widetilde{IB}_1^{\mathrm{PQ}}) \cup \{p\}$ (according to push-out rules), and by Lemma 8 the inequalities are preserved. (ii) $|\mathbf{IB}_\mathbf{V}^{\mathbf{PQ}}| + |\widetilde{\mathbf{IB}}_\mathbf{1}^{\mathbf{PQ}}| = \mathbf{B}$. Then, since $|\mathrm{IB}^{\mathrm{OPT}}| \geq |\widetilde{IB}^{\mathrm{PQ}}|$ and $|\mathrm{IB}^{\mathrm{OPT}}| + |\mathrm{overflow}(\mathrm{OPT})| < B$, we get that $|\mathrm{IB}_V^{\mathrm{PQ}}| - |\widetilde{IB}_V^{\mathrm{PQ}}| > |\mathrm{overflow}(\mathrm{OPT})|$. Now, if OPT accepts p then p is added to the $\mathrm{overflow}(\mathrm{OPT})$.

Processing of packet p by OPT. In this case, 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** $\mathbf{IB_V^{PQv/w} = \emptyset}$. This case is completely similar to the previous one.

$\mathbf{value(p) = 1}$ **and** $\mathbf{IB_V^{PQv/w} \neq \emptyset}$. In this case, the packet p being processed is sent to $\mathrm{Free(OPT)}$. It remains to show that in this case $\Phi_1^{\mathrm{OPT}}(l)$ 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, during transmission the value $(|\mathrm{IB}_V^{\mathrm{PQ}}| - |\widetilde{IB}_V^{\mathrm{PQ}}|)$ may decrease by one. In order to preserve the invariant on $\mathrm{overflow(OPT)}$, we move an arbitrary packet from $\mathrm{overflow(OPT)}$ (if it is nonempty) to $\mathrm{Free(OPT)}$.

To see the last statement of the lemma, it suffices to note that a packet may fall into $\mathrm{Free(OPT)}$ 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 $|\mathrm{IBT}_1^{\mathrm{OPT}}| + V|\mathrm{IBT}_V^{\mathrm{OPT}}| + |\mathrm{Free(OPT)}|$. Thus,

$$\alpha = \frac{|\mathrm{IBT}_1^{\mathrm{OPT}}| + V|\mathrm{IBT}_V^{\mathrm{OPT}}| + |\mathrm{Free(OPT)}|}{|\mathrm{IBT}_1^{\mathrm{PQ}}| + V|\mathrm{IBT}_V^{\mathrm{PQ}}|}$$

$$\leq 1 + \frac{|\mathrm{Free(OPT)}|}{|\mathrm{IBT}_1^{\mathrm{PQ}}| + V|\mathrm{IBT}_V^{\mathrm{PQ}}|} \leq 1 + \frac{|\mathrm{Free(OPT)}|}{V|\mathrm{IBT}_V^{\mathrm{PQ}}|}$$

$$\leq 1 + \frac{(W+2)|\mathrm{IBT}_V^{\mathrm{PQ}}|}{V|\mathrm{IBT}_V^{\mathrm{PQ}}|} \leq 1 + \frac{W+2}{V}.$$

∎

Corollary 10: 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 $\mathrm{PQ}_{v,-w}$. ∎

VII. The β-push-out case

In some systems it is important to control the number of pushed out packets, for at least two reasons: each admitted packet incurs some *copying cost* α, and pushing a packet out is not a simple operation. The addition of a copying cost α covers both cases. As a result, [12] 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 $\max(0, T - \alpha A)$. Thus, in extreme cases, the transmitted value of a push-out policy may even go down to zero, and copying cost provides an additional control on the number of pushed out packets. To implement such a control mechanism, Keslassy et al. [12] 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. In our context, 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 7 does not go through for β-push-out policies; it remains an open problem to show nontrivial (less than linear) upper bounds for β-push-out policies with two packet characteristics.

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

(1) $\mathrm{PQ}_{-w,v}^\beta$ is at least V-competitive in both in the case of arbitrary packet values and in the two-valued case;

(2) $\mathrm{PQ}_{v/w}^\beta$ 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 $\mathrm{PQ}_{v/w}^\beta$ is at least V-competitive;

(4) $\mathrm{PQ}_{v,-w}^\beta$ 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 $\mathrm{PQ}_{-w,v}^\beta$. (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 6 since $\beta > 1$. The same construction works: $\mathrm{PQ}_{v/w}$ fills its buffer with $B \times (1 \mid 1)$ and then drops incoming $B \times (W \mid V)$. $\mathrm{PQ}_{v/w}^\beta$ also drops them since $\frac{V}{W} \leq \frac{\beta V}{V} = \beta$. (4) Again, the constructions from Theorem 2 and Theorem 6 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 [9] are of little use for our model. They cannot provide information on required processing and time scale (i.e., which packets go in the same time slot) since this information is heavily dependent on a specific network processor. Therefore, in our experiments we have used synthetic traces.

We have conducted five series of experiments on synthetic traces, studying how performance depends on maximal required processing k, buffer size B, maximal value V, intensity λ_{on}, and β (in the model with copying cost). The actual optimal online algorithm in our model would be

computationally prohibitive, so to find out the competitive ratios of our algorithms we have used an algorithm which is actually better than optimal: a single priority queue that breaks each packet into single priority queue that processes smallest packets first and has n cores. This algorithm has been proven optimal in the single queue model, so in our model, where some queues may be congested and others idle, it performs even better than optimal. In our experiments, synthetic traffic was generated by 500 independent sources, where each source was a Markov-modulated Poisson process (MMPP) with packet rate λ_{on} in the "on" state and 0 in the "off" state; the probability of switching the state from "off" to "on" was 0.02 (a source switches on with probability $1/50$ on each tick), and the probability of switching from "on" to "off" was set to 0.2. The values and processing ratios of packets were chosen uniformly from $\{1, \ldots, k\}$ 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.

As the OPT algorithm, we used a single priority queue of size B that holds all packets regardless of their queue and makes n processing steps per time slot; this algorithm is obviously better than the optimal algorithm in our model.

Figure 2 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. 2 that will be described in subsections below; we have tested the four algorithms used in this work: $\text{PQ}_{v/w,-w}$, $\text{PQ}_{v/w,v}$, $\text{PQ}_{-w,v}$, and $\text{PQ}_{v,-w}$. Note that in all cases, OPT, $\text{PQ}_{v/w,-w}$, and $\text{PQ}_{v/w,v}$ are virtually indistiguishable: the competitive ratio of both $\text{PQ}_{v/w,-w}$ and $\text{PQ}_{v/w,v}$ was below 1.01 (that is, they stayed within one percent of OPT) across all settings we have tried in the experiments except at very large buffer sizes.

A. Maximal required processing

In the first set of simulations (Fig. 2(1-3)), we study performance as a function of the maximal required processing k. As k grows, all algorithms deteriorate (packets become heavier), but it is clear that $\text{PQ}_{-w,v}$, which pays more attention to required processing, fares better and becomes closer to OPT while $\text{PQ}_{v,-w}$ loses badly. We see that $\text{PQ}_{v/w}$ is uniformly the best policy, virtually indistinguishable from OPT.

B. Buffer size B

In the second set of simulations (Fig. 2(4-6)), we study performance as a function of the buffer size B. For relatively small sized buffers, $\text{PQ}_{v/w,-w}$ and $\text{PQ}_{v/w,v}$ remain indistinguishable. However, as B grows larger, $\text{PQ}_{v/w,v}$ stars to overcome $\text{PQ}_{v/w,-w}$, and for very large buffers, especially for low values of V, the difference becomes significant; also, both ratio-based priority queues start to lose somewhat to the fractional OPT algorithm. Thus, for large buffers we recommend $\text{PQ}_{v/w,v}$.

C. Maximal value V

In the third set of experiments (Fig. 2(7-9)), we look at performance as a function of the maximal value V. It turns out that the performance level of all our algorithms does not significantly depend on V in realistic cases; there is only a slight improvement in performance as V increases, noticeable only for $\text{PQ}_{v,-w}$, as expected since $\text{PQ}_{v,-w}$ emphasizes value over required processing. The relative order of algorithms remains unchanged.

D. Incoming stream intensity

The fourth set of experiments (Fig. 2(10-12)) shows how performance depends on the intensity λ_{on} of switched on incoming packet sources. Here, we see that all algorithms, including OPT, show approximately the same rate of decline under higher and higher input intensities, and their relative standings remain the same across all intensities.

E. β for β-push-out policies

The last, fifth set of experiments (Fig. 2(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. On the graphs for $\text{PQ}_{v/w}$, we see that in each case, performance is maximized at some $\beta^* > 1$; in these three cases, $\beta^* \in [1.2, 1.5]$. Thus, β-push-out policies indeed provide an improved alternative for regular policies in the model with copying cost.

To summarize, in this section we have shown a comprehensive simulations study on synthetic traces. The main result is that the $\text{PQ}_{v/w}$ policy that we have introduced in this work is uniformly the best policy across all tested settings. Another interesting result is that β-push-out policies do have an advantage (albeit slight) in settings with nonzero copying cost.

IX. Conclusion

In this work, we have considered a single queue processing packets with two different characteristics: processing requirement and value. We have investigated various packet processing orders and found that they have linear lower bounds on the competitive ratio, which makes them unattractive. However, we have provided positive results in the special case

Processing policy	General case	Two-valued case	
	Lower bound	Lower bound	Upper bound
Any	$5/4$	$1+\frac{V-1}{V^2}-O\left(\frac{1}{W}\right)$	
$\mathrm{PQ}_{-w,v}$, $\mathrm{PQ}^{\beta}_{-w,v}$	V	V	V
$\mathrm{PQ}_{v,-w}$, $\mathrm{PQ}^{\beta}_{v,-w}$	$\frac{(V-1)}{V}W-o(1)$	$\frac{W}{V}+o(1)$	$1+\frac{W+2}{V}$
$\mathrm{PQ}_{v/w}$, $\mathrm{PQ}^{\beta}_{v/w}$, $\beta W \geq V$	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.

of two different values, 1 and V. Namely, we have shown a $(1+(W+2)/V)$ upper bound for a priority queue $\mathrm{PQ}_{v,-w}$ that sorts 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 sorts packets by unit processing (ratio of value to processing). Besides, we have introduced a general lower bound of $5/4$ (in case when value 2 is available) that holds for all online policies; this lower bound distinguishes the model considered in this work from previously studied models with uniform processing or uniform values, where certain online policies have been proven to be optimal. Results of this work are summarized in Table I; note that all algorithms in the table employ push-out and use different heuristics for it.

For the two-valued case, we have proven tightly matching lower and upper bounds on the competitive ratio (they differ by $1+o(1)$). It still remains a very interesting open problem to prove upper bounds for the general case of multiple possible values, and another interesting problem would be to prove upper bounds for β-push-out policies. However, the really crucial question here is whether it is possible to devise a processing policy with better than linear competitive ratio for the general case of two characteristics. We suggest this problem for further study.

Acknowledgements: This work was partially supported by the Government of the Russian Federation (grant 14.Z50.31.0030).

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

[8] 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.

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

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

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

[12] 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.

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

[14] 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.

[15] 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.

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

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

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

[19] 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.

[20] 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.

[21] 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.

[22] Kirill Kogan, Alejandro López-Ortiz, Sergey I. Nikolenko, Alexander V. Sirotkin, and Denis Tugaryov. FIFO queueing policies for packets with heterogeneous processing. In *Proceedings of the* 1^{st} *Mediterranean Conference on Algorithms (MedAlg 2012), Lecture Notes in Computer Science*, volume 7659, pages 248–260, 2012. arXiv:1204.5443 [cs.NI].

[23] 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.

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

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

[26] 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.

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

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

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

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

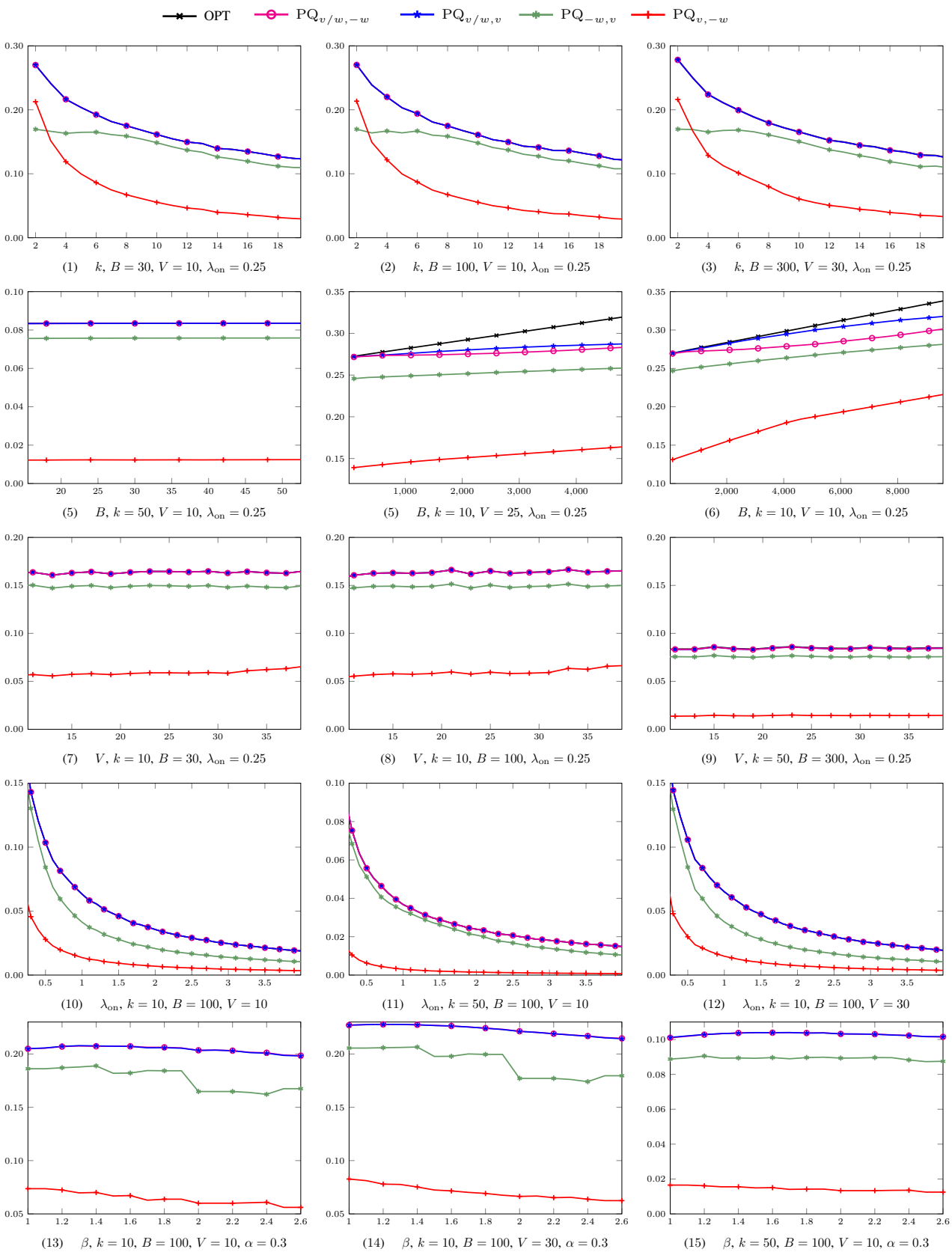

Fig. 2. Simulation results: share of successfully transmitted packets in the required processing model as a function of (1-3) maximal required processing k, (4-6) buffer size B, (7-9) maximal value V, (10-12) packet stream intensity λ_{on}. Specific simulation parameters are shown in graph captions.