Noname manuscript No.
(will be inserted by the editor)

Admission Control in Shared Memory Switches

Patrick Eugster · Alex Kesselman · Kirill Kogan · Sergey Nikolenko · Alexander Sirotkin

the date of receipt and acceptance should be inserted later

Abstract Cloud applications bring new challenges to the design of network elements, in particular the burstiness of traffic workloads. A shared memory switch is a good candidate architecture to exploit buffer capacity; in this work, we analyze the performance of this architecture. Our goal is to explore the impact of additional traffic characteristics such as varying processing requirements and packet values on objective functions. The outcome of this work is a better understanding of the relevant parameters for buffer management to achieve better performance in dynamic environments of data centers. We consider a model that captures more of the properties of the target architecture than previous work and consider several scheduling and buffer management algorithms that are specifically designed to optimize its performance. In particular, we provide analytic guarantees for the throughput performance of our algorithms that are independent from specific distributions of packet arrivals. We furthermore report on a comprehensive simulation study which validates our analytic results.

This work is an extended version of [11].

The work of P. Eugster was partially supported by the German Research Foundation (DFG) project "MAKI" and by ERC grant "LiveSoft". The work of Kirill Kogan was partially supported by a grant from the Cisco University Research Program Fund, an advised fund of Silicon Valley Community Foundation. The work of S.I. Nikolenko was supported by the Basic Research Program of the National Research University Higher School of Economics, 2015, grant No 78.

P. Eugster
University of Lugano, Purdue University, and TU Darmstadt
E-mail: eugstp@usi.ch

A. Kesselman
Google Inc.
E-mail: alx@google.com

K. Kogan
IMDEA Networks Institute
E-mail: kirill.kogan@imdea.org

S.I. Nikolenko
National Research University Higher School of Economics, St. Petersburg, Russia
Steklov Mathematical Institute at St. Petersburg
E-mail: sergey@logic.pdmi.ras.ru

A. Sirotkin
National Research University Higher School of Economics, St. Petersburg, Russia
E-mail: alexander.sirotkin@gmail.com

1 Introduction

The Internet is built around a large variety of network switches that transfer data packets toward their destinations. If a burst of packets destined to the same output port arrives, not all packets can be transmitted immediately, and some should be buffered or dropped. Allocation and availability of buffer resources to input and output ports, determined by the buffering architecture, affects burst absorption capabilities and performance characteristics of a network switch.

In the *output queuing* buffering architecture, packets arriving from input ports immediately join a corresponding queue at the switch output port; buffering at the output ports allows to control how these packets are transmitted out. In the *shared-memory* switch architecture, output queues are dynamically allocated from the same shared memory pool. The main benefit of shared-memory switches is that the same memory space can serve all output ports. However, complete sharing may perform poorly under overload conditions [16]: packets destined to the same output port can take over most of the memory, preventing admission of packets destined to other output ports that causes the total switch throughput to drop. At the other extreme, shared memory can also simulate complete memory partition on the port-by-port basis. In this case the traffic of each output port is fully isolated;

this comes at the cost of memory underutilization, risking higher drops during incoming traffic bursts. The flexibility of shared memory allows the policies to balance tradeoffs between complete sharing and complete partition.

Incorporation of additional input traffic characteristics can significantly affect desired objectives. Packets can require different processing, so as a result each admitted packet should be processed by the corresponding output port during several "processing cycles" before it is actually transmitted. Increased required processing in some packets may cause congestion even for traffic with relatively modest burstiness characteristics. In addition, traffic flows can have different constraints, and to further differentiate incoming traffic each packet may carry an intrinsic value. The need to account for additional input characteristics leads to increased complexity of buffer management policies.

In this work we consider a shared memory switch with multiple output ports, where a buffer of size B is shared among all traffic. Each arriving packet is labeled with an output queue. Arrivals can be adversarial. The number of input ports has no significance for the model since we process all arriving packets at the same time, as a single burst. In stark contrast to the seminal work of Aiello et al. [1], where packets have uniform values and processing requirements, in our model each arriving packet has either processing requirement or *value*, that is, how much transmitting that packet adds to the objective function.

In our study of these settings, we use the paradigm of *competitive analysis* [6, 33]: an algorithm ALG is α-*competitive* for some $\alpha \geq 1$ if for any arrival sequence the total value of packets transmitted by ALG is at least $1/\alpha$ times the total value of packets transmitted by an optimal offline clairvoyant algorithm OPT. The final practical goal here is to show how additional workload characteristics should be taken into account for buffer management. Since all policies we consider are greedy (they accept and transmit all traffic if there is no congestion), extreme cases during congestion periods can actually happen in such scenarios as big data processing [8, 34, 35]. So worst-case results are of particular importance here since they cover all possible cases, including extreme congestion or even adversarial traffic, that helps define buffer management "rules of thumb" independent of specific arrival distributions.

In the first part of this paper every incoming packet has unit value but has processing requirement varying from 1 to k and has an output port label from 1 to n. In this case, the objective reduces to maximizing the number of transmitted packets. We show that the Longest-Queue-Drop (LQD) policy [1] is at least $(n/2 - o(n))$-competitive for sufficiently large buffer size B and maximal possible processing k (in cycles); we also show that Biggest-Packet-Drop (BPD), a policy that pushes out packets with maximal processing requirement in case of congestion, degrades to at least $(n+1)/2$-competitiveness. In addition we introduce a natural Biggest-Average-Drop (BAD) policy that pushes out a packet with maximal required processing from a queue with maximal average processing requirements in the case of congestion; we show the same lower bound of $(n+1)/2$ for BAD. All lower bounds hold even for PQ (priority queueing) processing order, where packets are ordered in the non-decreasing order of their processing requirements, which was shown to be optimal in the single-queue case [17]. The main result of this work is the 2-competitiveness of a variant of the Longest-Work-Drop (LWD) policy of [12] that holds in our general model when packets with heterogeneous processing requirements are processed in PQ order in each queue (Section 5). In addition we show that in the FIFO case of our general model, LWD is at least $\Big((\log_{B/n} k)(1 - 1/B) + 1\Big)$-competitive.

In the second part of this paper we consider a model where each incoming packet has, in addition to an output port label, a heterogeneous value from 1 to V (and uniform processing). In this case the objective is to maximize transmitted value. This problem was presented as an open question in SIGACT News [13, p. 22]. It might seem that the model with values should be similar to the model with required processing since the greedy algorithm with PQ order and push-outs is optimal in both cases. However, we show that in the shared memory case the Maximal-Total-Value-Drop (MTVD) policy, which is similar to LWD, is at least V-competitive. We also turn to policies that combine several characteristics and consider the Minimal-Ratio-Drop (MRD) policy introduced in [12] that considers both queue occupancy and the average value in the same queue. MRD was conjectured in [12] to have constant competitiveness by analogy with the case with required processing; here we disprove the conjecture, showing that MRD is at least V-competitive. This indicates that models with values and required processing generalize from the single-queue case in a different way.

The paper is organized as follows. Section 2 contains a brief survey of related work. Section 3 details the model underlying our work. Section 4 considers lower bounds of several algorithms for packets with heterogeneous processing requirements. The main result of this paper — 2-competitiveness of LWD policy for packets with heterogeneous requirements — is presented in Section 5. Section 6 considers a model with heterogeneous packet values. In Section 7 we evaluate the empirical performance of the considered algorithms on synthetic traces, and Section 8 concludes the paper.

2 Related Work

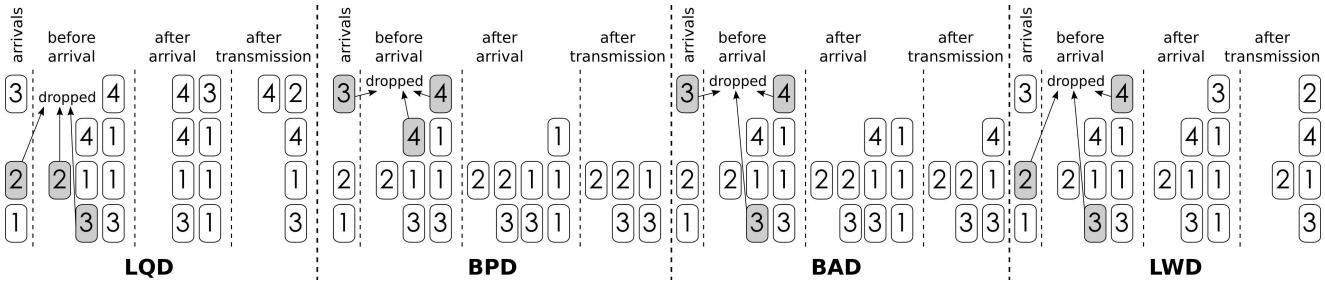

Fig. 1: A sample time slot of Longest Queue Drop (LQD), Biggest Packet Drop (BPD), Biggest Average Drop (BAD), and Largest Work Drop (LWD) policies with maximal processing $k = 4$, $n = 4$ output ports, and a shared buffer of size $B = 8$. Queues for each output port are shown horizontally. Shaded packets are dropped during arrival.

Kesselman and Mansour [1] propose a non-push-out buffer management policy called Harmonic that is at most $O(\log n)$-competitive and establish a lower bound of $\Omega(\frac{\log n}{\log \log n})$ on the performance of any online non-push-out deterministic policy, where n is the number of output ports. Aiello et al. [24] demonstrate that the LQD policy is at most 2- and at least $\sqrt{2}$-competitive. Both works consider homogeneous packet processing, where each packet requires a single processing cycle.

Eugster et al. [12] consider a limited variant of our model where all incoming packets for the same output port have identical processing requirements. But even in this case the work [12] proves that LQD is at least $\left(\sqrt{k} - o(\sqrt{k})\right)$-competitive. Fortunately, in [12] a generalization of LQD, namely Longest-Work-Drop (LWD), was proposed for this limited model; LWD is at most 2-competitive in case when packets are processed with minimal current required processing first (PQ order). Besides, it was shown in [12] that Biggest-Packet-Drop (BPD), which in case of congestion prefers to drop packets with maximal processing requirements, is at least $\log k$ competitive for $B > \frac{k(k+1)}{2}$. Unlike the model in [12] the model we consider in this paper allows for packets with heterogeneous processing requirements to be admitted to the same queue. This generalization has significant impact on the efficiency of considered policies and is applicable to a number of real-world scenarios. The model in [12] suggests to allocate a separate queue per processing requirement per output port, but in this case constraints on the maximal number of supported queues can significantly limit applicability once k and n are growing.

Previous research efforts concentrated on studying algorithms for management of bounded buffers with provable competitiveness guarantees [2–5,10,18–22,28,29], but none of these models cover the case of packets with heterogeneous processing requirements. Survey by Goldwasser [13] provides a comprehensive overview of results for these models. Adding into account heterogeneous processing requirements for the single queue case has been initiated in [17,27] and later extended for the multiple separated queues in [26]. Recently, [7,9] studied a combination of heterogeneous processing requirements with other traffic characteristics such as packet values and deadlines for a single queue case; see the survey [30] for an overview.

Pruhs [31] provides a comprehensive overview of competitive online scheduling for server systems. Note that in server systems, scheduling usually concentrates on average response time, while we focus mostly on throughput. Furthermore, scheduling of server systems does not allow jobs to be dropped, which is an inherent aspect of our model due to size limitations on buffers.

3 Model Description

We consider an $1 \times n$ shared memory switch with one input and n output ports and a buffer of size B, that is, the total length of all queues is bounded by B. We assume that $B \geq n$. The number of input ports have no significance for the model: we process all arriving packets at the same time, as a single burst. Each output port, on the other hand, manages a separate output queue, denoted Q_i for port i, $1 \leq i \leq n$; the number of packets in Q_i is denoted by $|Q_i|$. Each packet arriving at the input port is labeled with the output port number d and its required work w in processing cycles ($1 \leq d \leq n$ and $1 \leq w \leq k$), where k denotes the global upper bound on required work per packet. Each Q_i implements either (i) priority queueing (PQ) processing order, where packets are processed in non-decreasing order of required processing, or (ii) first-in-first-out (FIFO) processing order, where packets are processed in order of arrival. In what follows, we denote by $\boxed{w \mid i}$ a packet with required work w intended for output port i; by $h \times \boxed{w \mid i}$, a burst of h $\boxed{w \mid i}$ packets arriving at the same time. We also denote by $r_t(p)$ the remaining required processing of a packet p at time unit t. Time is slotted; we divide each time slot into two phases (see Fig. 1). During the (1) *arrival phase* a burst of new packets arrives at the input port, and the buffer management policy decides which packets should be admitted

(more than n arrivals are allowed at the same time slot). The arrivals are adversarial and do not assume any specific traffic distribution. An accepted packet can be later dropped from the buffer when another packet is accepted; in this case we say that a packet p is *pushed out* by another packet q, and a policy that allows this is called a *push-out* policy. During the (2) *transmission phase*, required work of the head-of-line packet according to the supported processing order (PQ or FIFO) at each non-empty queue is reduced by one, and every packet with zero residual work is transmitted. To facilitate our proofs, we use some properties of ordered (multi-)sets. These notions, as well as the properties we recall in this section, will enable us to compare the performance of our proposed algorithms. In the following, we consider multi-sets of real numbers, where we assume each multi-set is ordered in non-decreasing order. We will refer to such multi-sets as *ordered sets*. We denote the i-th element of a set A by a_i. Given two ordered sets A and B, we say $A \succeq B$ if for every i for which both a_i and b_i exist, $a_i \geq b_i$. The following lemma, and its corollary, will be used in our analysis; their proofs can be found in [25] (Lemma 1 and Corollary 2).

Lemma 1 *For any two ordered sets A and B satisfying $A \succeq B$, and any two real numbers a, b such that $a \geq b$, if (i) $b \leq b_{|B|}$ or (ii) $|A| \leq |B|$ then the ordered sets $A' = A \cup \{a\}$, $B' = B \cup \{b\}$ satisfy $A' \succeq B'$.*

Corollary 1 *For any two ordered sets A, B satisfying $A \succeq B$, and any real number b, if (i) $b \leq b_{|B|}$ or (ii) $|A| \leq |B|$ then the ordered set $B' = B \cup \{b\}$ satisfies $A \succeq B'$.*

4 The Quest for a Perfect Policy with Heterogeneous Processing

In this section, we consider several possible candidates for the "ideal" policy that might provide constant competitiveness in the model presented above. These algorithms either look like natural candidates or have been proven to be efficient for uniform processing [1,24]. Note that in our model, each algorithm has two versions, with PQ and FIFO processing order in each output queue. By default, we assume that every queue implements PQ order.

Longest-Queue-Drop (LQD): during the arrival of a packet p with output port i, denote by $j^* = \arg\max_j\{|Q_j| + [i = j]\}$ where $[i = j] = 1$ if $i = j$ and 0 otherwise (i.e., Q_{j^*} is the longest queue once we virtually add p to Q_i; we choose one with largest total required processing if there are several); then do the following: (1) if the buffer is not full, accept p into Q_i; (2) if the buffer is full, accept p into Q_i and push out last packet from Q_{j^*}; else drop p. Note that in case (2), if $i = j^*$ and p is the largest packet in Q_{j^*}, p will be "pushed out" itself, i.e., dropped.

Note that the proposed here version of LQD is not oblivious to processing requirements since it will drop a packet with a maximal processing from the longest queue in the case of congestion. In case of homogeneous processing, LQD is at least $\sqrt{2}$- and at most 2-competitive [1]. For heterogeneous required processing, the situation is worse.

Theorem 1 *For $k \geq n(n-1)$ and $B \geq k + n$, LQD is at least $(n/2 - o(n))$-competitive.*

Proof Over the first burst, there arrive B packets of each of the following kinds: $\boxed{1 \mid 1}$, $\boxed{k \mid 2}$, $\boxed{k \mid 3}$, …, $\boxed{k \mid n}$. LQD evenly distributes the packets among queues and has B/n packets in each of its nonempty queues (throughout the proof we assume that B is large and is divisible by everything we need it to be). OPT accepts $(B-n+1)\times\boxed{1 \mid 1}$ and one each in the remaining queues. Every k processing cycles there arrive $1 \times \boxed{k \mid 2}$, $1 \times \boxed{k \mid 3}$, …, $1 \times \boxed{k \mid n}$, so OPT always has packets in these queues to work on, but there are no more $\boxed{1 \mid 1}$s. OPT spends $(B - n + 1)$ time to process all $\boxed{1 \mid 1}$s (after that the arrival iteration is restarted); let us count the number of processed packets by this time. OPT will have processed $(B-n+1)$ in queue 1 and $(B-n+1)/k$ in each one of $n - 1$ other queues. LQD will have the same $(B-n+1)/k$ processed packets in each queue but the first, and in the first queue LQD will have processed B/n since there are no more packets in first queue. Thus, the overall competitive ratio is

$$\frac{(B - n + 1) + (n - 1) \times \frac{B-n+1}{k}}{\frac{B}{n} + (n - 1) \times \frac{B-n+1}{k}},$$

and for $k = n(n - 1)$ we get

$$\frac{(B - n + 1) + \frac{B-n+1}{n}}{\frac{B}{n} + \frac{B-n+1}{n}} \geq \frac{(B - n + 1)}{2\frac{B}{n}} \approx n/2,$$

as needed. □

The next two algorithms drop packets with the largest processing requirement in case of congestion.

Biggest-Packet-Drop (BPD): during the arrival of a packet p with required work w and output port i, denote by Q_j the nonempty queue that contains a packet $p_{\max}$ with the largest processing requirement $w_{\max}$; then do the following: (1) if the buffer is not full, accept p into Q_i; (2) if the buffer is full and $w < w_{\max}$, push out $p_{\max}$ from Q_j and accept p into Q_i; (3) if the buffer is full and $w > w_{\max}$, drop p.

Biggest-Average-Drop (BAD): during the arrival of a packet p with required work w and output port i, denote by Q_j the nonempty queue with largest average processing requirement $\bar{w}_{\max}$; then do the following: (1) if the buffer is not full, accept p into Q_i; (2) if the buffer is full and $w < \bar{w}_{\max}$, push out packet with maximal work from Q_j and accept p into Q_i; (3) if the buffer is full and $w > \bar{w}_{\max}$, drop p.

Theorem 2 *BPD and BAD are both at least* $(n+1)/2$*-competitive.*

Proof The hard instance is as follows: every time slot, there arrive $B \times \boxed{1 \mid 1}$ followed by $B \times \boxed{2 \mid 2}, \ldots, B \times \boxed{2 \mid n}$ (a full set of packets); BPD and BAD both accept only $B \times \boxed{1 \mid 1}$ and keep processing one packet per time slot, i.e., 2 packets per 2 time slots, while OPT is free to accept the packets evenly and get $2 + 2/2 + \ldots + 2/2$ packets per 2 time slots, getting the bound as $\frac{n+1}{2}$. □

Largest-Work-Drop (LWD): during the arrival of a packet p with output port i and required processing w, denote by $j^* = \arg\max_j\{W_j + [i=j]w\}$ where $[i=j] = 1$ if $i = j$ and 0 otherwise, and W_j is the total required processing of all packets in queue Q_j (i.e., Q_{j^*} is the queue with the largest total required processing once we virtually add p to Q_i; we choose the one with the largest single packet if there are several queues with largest work). Then do the following: (1) if the buffer is not full, accept p into Q_i; (2) if the buffer is full and w is smaller than the required processing of at least one packet in Q_{j^*}, push out the largest packet from Q_{j^*} and accept p into Q_i; else drop p.

Theorem 3 *LWD with FIFO processing order is at least* $(\log_{B/n} k)(1 - n/B) + 1$*-competitive.*

Proof Consider LWD with n output ports and suppose that n divides B. Let $a = B/n$. For every output port i, there arrive $1 \times \boxed{k \mid i}$ followed by $(a-1) \times \boxed{k/a \mid i}$. OPT discards $\boxed{k \mid i}$ and accepts all $\boxed{k/a \mid i}$. After $(a-1)k/a$ processing steps, LWD has $a \times \boxed{k/a \mid i}$ in every queue and has not yet transmitted any packets, while OPT has transmitted the additional $(a-1)$ packets. The next arrival is $(a-1) \times \boxed{k/a^2 \mid i}$ for every i. Since the processing order is FIFO, after accepting all these packets LWD has $\boxed{k/a \mid i}$ as HOL (head of line packet) followed by $(a-1) \times \boxed{k/a^2 \mid i}$ and OPT has only $(a-1) \times \boxed{k/a^2 \mid i}$ in every queue. After $(a-1)k/a^2$ processing steps, LWD has $a \times \boxed{k/a^2 \mid i}$ in each queue and has not yet transmitted any packets, but OPT has transmitted all $(a-1)$ packets. Next we repeat the above arrival sequence for packets of size $k/a^3, \ldots, k/a^m$, until $k/a^m = 1$, i.e., for $log_a(k)$ steps. On every step, OPT transmits $(a-1) \times n$ packets and LWD transmits nothing. After all these steps, $a \times \boxed{1 \mid i}$ has arrived in every queue, so after a processing cycles both OPT and LWD transmit B packets and finish with empty buffers. Thus, the total number of packets that LWD transmits is B, and the total number of packets transmitted by OPT is $n(a-1)\log_a k + B$, getting the ratio $\frac{n\cdot((a-1)\log_a k)}{B} + 1$. Recall that we had $a = B/n$, so the final ratio is $\frac{n\cdot(B/n-1)\log_{B/n} k}{B} + 1 = (\log_{B/n} k)(1 - n/B) + 1$. □

5 Scheduling with Heterogeneous Processing

To avoid ambiguity during the arrival phase, a reference time, usually denoted by t, should be interpreted as the arrival *time unit* of a single packet. If several packets arrive at the same time slot, we consider them independently, in the sequence in which they arrive. The arrival phase operates with the time unit resolution, while processing and transmission phases still operate only with the time slot resolution.

We introduce the class of *semi-greedy* algorithms $\mathcal{SG}$. A semi-greedy algorithm $G \in \mathcal{SG}$ accepts a packet if G's buffer is not full. G is defined by an *iteration*. An iteration begins during the first time unit t_s, when G's buffer becomes full and ends on the first time unit t_e when G has transmitted at least B packets since t_s. To simplify analysis, we also assume that any $G \in \mathcal{SG}$ drops the content of its buffer at the end of an iteration at time t_d, $t_e \le t_d < t_e + 1$, without gain to its throughput.

In what follows we consider an artificially enhanced version of the optimal algorithm OPT to analyze an online algorithm G: (1) OPT never pushes out admitted packets (since OPT is offline, it is clear that this property can be satisfied); (2) at the end of an iteration, OPT flushes out all packets residing in its buffer with extra gain to its throughput (in this case, the throughput of OPT is no worse than any other optimal algorithm); (3) if at time t, G transmits out of port i, then OPT immediately transmits the first packet q (in PQ order) out of port i (if q exists) regardless of its remaining work value $r_t(q)$, with extra gain to OPT's throughput; again, clearly we only make OPT better.

Note that by definition, for a given sequence of inputs all algorithms in $\mathcal{SG}$ with the same processing order accept and transmit the same number of packets between starting with an empty buffer and the first moment of congestion (first moment when the buffer is full). With PQ processing order, moreover, no algorithm can transmit more packets from this sequence over this time. And, by definition, at the end of an iteration an $\mathcal{SG}$ algorithm has an empty buffer. The difference in the number of packets remaining at the end of an iteration (just before t_d) is irrelevant since all these packets are dropped at time t_d. Since during $[t_s, t_d)$ any semi-greedy algorithm G transmits at least B packets, adding less than B packets to OPT's throughput for free by property (3), increasing the total competitiveness of G by at most 1. The general idea of our analysis here is similar to [23] but the definition of an iteration and the analysis of what happens during an iteration and between two consecutive iterations are completely new.

We denote by t_b the first time unit after the end of a previous iteration or the time unit of the first arrival in the system. Since a semi-greedy G and OPT both clean their buffers at time t_d, it suffices to compare performance of G versus OPT only during $[t_b, t_e]$. The class of semi-greedy algorithms is based on a well-structured accounting infrastructure that makes it possible to analyze online buffer management policies with various characteristics rigorously. We assume that every queue implements priority queueing (PQ), where all packets in the same queue are ordered in non-decreasing order of required processing. For simplicity, we denote queue Q_i of an algorithm ALG by A_i, where A is the the first letter of the name of the considered algorithm (e.g, O_i and G_i are the i-th queue of OPT and a semi-greedy G). We treat queues as ordered sets in the sense of Lemma 1 and correspondingly write $A_i \leq B_i$ for two queues if for every slot in the queue where both A_i and B_i have packets p_A and p_B respectively, $w(p_A) \leq w(p_B)$.

The *latency* $\mathrm{lat}_t^A(p)$ of a packet $p \in A_i$ at time t is the number of time slots currently needed to transmit p out of A_i, that is, the sum of required processings r_t for p and all packets that are before p in queue A_i. We define the latency of an already transmitted packet as -1 and the latency of a packet that has not yet arrived as ∞. An i-th port or queue of ALG's buffer is called *active* at time unit t if it transmits during t; otherwise, it is called *idle*. Note that due to property (3) in the definition of OPT, at this point it is unclear if our version of OPT can transmit more packets than a semi-greedy G during $[t_b, t_s)$, and theoretically it can happen. To show that OPT does not transmit more packets than a semi-greedy algorithm G during $[t_b, t_s)$, we formulate the following lemma; the lemma also proves an even stronger result (2) which will be used in the proof of the key Lemma 3. Consider an interval of time I, $I \subseteq [t_b, t_d)$. We denote by S_I^A the set of packets transmitted by algorithm A during I.

Lemma 2 *For a semi-greedy algorithm G with PQ processing and time unit $t \in [t_b, t_s)$ between two consecutive iterations, (1) $\left|S^{OPT}_{[t_b,t]}\right| \leq \left|S^{G}_{[t_b,t]}\right|$; (2) for every $1 \leq i \leq n$, at time unit t $G_i \preceq O_i$ and $|G_i| \geq |O_i|$.*

Proof The proof proceeds by induction on the number of time units. During the first arrival of a packet p to Q_i at time t_b, since G is greedy, G accepts p, so the induction base follows. Assume that the lemma holds during $[t_b, t)$, $t \in [t_b, t_s - 1]$. We are to show that the lemma holds during time unit t.

Processing and transmission: the induction step holds by induction hypothesis for all empty queues or queues with head-of-line packet whose remaining processing is at least one. Consider any active queue index i, $1 \leq i \leq n$. If both O_i and G_i are nonempty just before t, by the induction hypothesis we have $O_i[1] \succeq G_i[1]$. If G_j is active at time unit $t+1$, then by definition of OPT (property (3)) O_j is also active (even if there are additional processing cycles in the HOL packet), and the induction step follows.

A packet p arrives to Q_i: during the arrival phase, the number of transmitted packets is unchanged, so condition (1) follows. Since there is no congestion during $[t_b, t_s)$, G accepts all arrivals. By the induction hypothesis, at the end of time unit $t-1$ we had $G_i \preceq O_i$ and $|G_i| \geq |O_i|$. Thus, if OPT accepts, by condition (2) follows Lemma 1(ii) at time unit t. If OPT does not accept p to O_i, condition (2) follows by Corollary 1(ii). □

The Largest Work Drop (LWD) policy belongs to the $\mathcal{SG}$ class. The rationale behind LWD is to minimize the duration of an iteration. It can be done by optimizing a "local" state of LWD buffer, and that is why we suggest to drop packets from a queue with the largest total required processing. Our plan is as follows. By Lemma 2, between two consecutive iterations OPT does not transmit more than LWD. We denote by T the number of packets transmitted by LWD between two consecutive iterations. Later we are to show that during $[t_s, t_d)$ LWD transmits B packets no later than OPT transmits B packets. Since at t_d OPT contains at most B packets, during $[t_b, t_e]$ OPT transmits at most $T+2B$, whereas LWD transmits $T+B$ packets.

Next, we define red and white packets in a buffer. For any time interval $I' = [t_s - 1, t]$, $t \in [t_s - 1, t_d)$, during an iteration we say that the $B - \left|S_{I'}^{LWD}\right|$ packets with minimal latency in LWD buffer are colored in *red*; those are the first $B - \left|S_{I'}^{LWD}\right|$ packets that will be transmitted out by LWD unless new packets arrive and change the situation. Every other packet in LWD buffer is colored in *white*. Packets that cease to be red are immediately recolored in white again. We denote by R_i the set of all red packets in L_i. Lemma 3 contains the main ideas of this upper bound.

Observation 4 *If a packet $p_j \in L_i$ is red then every $p_l \in L_i$ is red for $1 \leq l \leq j-1$.*

Lemma 3 *For every packet $p_l \in O_i$, $1 \leq l \leq |O_i|$ and $1 \leq i \leq n$, at time unit $t \in (t_s, t_e)$ either (1) there is a red packet $q_l \in L_i$ such that $r_t(p_l) \geq r_t(q_l)$, or (2) for every red packet q at LWD buffer, $r_t(p_l) + W(R_i) \geq \mathrm{lat}_t^{LWD}(q)$, where $W(A)$ is the total remaining processing time for all packets in a set A.*

Proof The proof is by induction on time units.

Base: Consider time unit $t_s - 1$. By definition of iteration, at the end of $t_s - 1$ LWD's buffer is full. Since LWD is semi-greedy, by Lemma 2 at time $t_s - 1$ $L_i \leq O_i$ and, therefore, $R_i \leq O_i$ for every $i \in [1, n]$. Thus, the induction base follows.

Hypothesis: Assume that the lemma holds for every time unit $t' \in [t_{s-1}, t)$, $t < t_d$. We are to show that it holds at the t-th time unit.

Induction step.

Processing and Transmission: consider a time unit t that represents the processing and transmission phase. At time t, every nonempty queue L_i is either active (in this case O_i is active too regardless of how many processing cycles remains in HOL packet of O_i by property (3) in the definition of OPT) or the processing cycles of HOL packets of L_i and O_i are decreased by one. Assume that during $t \in (t_s, t_e)$, O_j is active and transmits a packet p; while L_i is idle. In this case by condition (2) LWD's buffer does not contain any red packet that means the iteration is already over, hence, $t \geq t_e$, which is a contradiction.

Arrival of a packet p to Q_i: Note that if OPT accepts p, its buffer has free space since by definition OPT never pushes out already accepted packets.

OPT and LWD reject p: the hypothesis holds at time t.

OPT accepts p, but LWD rejects: LWD's buffer is congested. Furthermore, since p is rejected by LWD, its required processing exceeds that of any packet in L_i. Suppose that p is at the l's position in O_i after acceptance, $l \leq |O_i|$. If $q_l \in L_i$ is red, condition (1) holds (the required processing of p is at least the required processing of any packet in L_i, including all red packets in L_i). If $q_l \in L_i$ is white or $l > |L_i|$, assume that there is a red packet whose latency is more than $r_t(p)+W(R_i)$. If $l > |L_i|$, $r_t(p)+W(R_i) = r_t(p)+W(L_i)$ that is (by definition of LWD) at least $W(L_j)$ since p is rejected. Thus, condition (2) holds. If $q_l \in L_i$ is white then $r_t(q_l) + W(R_i) \geq W(R_j)$, for any $j \in [1, n]$ (by definition of red packet); $r_t(q_l) \leq r_t(p)$ (otherwise, LWD will not drop p). Therefore, condition (2) holds, and the induction hypothesis holds too.

OPT and LWD accept p: 1. If $r_t(p)$ is less than at least one red packet in L_i then p is recolored in red and the last red packet in L_i is recolored in white. Since no new red packets are added to the queues other than L_i, condition (1) holds in these queues. By Theorem 1(i), condition (1) holds for any red packet in R_i. Next we show that condition (2) continues to hold for any OPT packet that is not covered by condition (1). Since the maximal latency among red packets does not increase for any queue except j, condition (2) holds. Consider a packet $u_l \in O_j$ corresponding to q_l recolored from red to white; by condition (1) of the induction hypothesis, $r_t(u_l) \geq r_t(q_l)$. Therefore, $r_t(u_l) + W(R_j) \geq r_t(q_l) + W(R_j)$, and (2) holds.

2. If the value of $r_t(p)$ is at least the required processing of any red packet in L_i then if $r_t(p)+W(R_i)$ is less than the latency of some red packet in LWD's buffer, recolor p in red, but the red packet q_l with a maximal latency in LWD's buffer recolor in white. Otherwise, p remains white. If p is white then the condition (1) follows by the induction hypothesis. Since p is white, $r_t(p)+W(R_i)$ is at least the latency of any red packet in LWD's buffer (otherwise, p is recolored in red). If p is recolored in red, condition (2) follows similar to the case 1. Since only Q_i is affected, condition (1) is satisfied for any Q_m, $m \neq i$ and holds for Q_i by Lemma 1(ii).

OPT rejects, LWD accepts: 1. Consider the case when LWD's buffer is not congested. (i) If $r_t(p)$ is at least the remaining processing of some white packet in Q_i, the set of the red packets is not changed. Also since OPT rejects p the set of OPT's packets is not changed also. Hence, conditions (1) and (2) hold. (ii) Otherwise, if $r_t(p) + W(R_i)$ is less than the latency of the red packet q with a maximal latency in LWD's buffer then recolor p in red and q in white. Denote by p an OPT packet in the position $|R_i|+1$ of O_i. If $|O_i| > |R_i|$ just before p is arrived, $r_t(p) + W(R_i)$ is more than the latency of q. Hence, $r_t(p) + W(R_i) > r_t(p) + W(R_i)$ and therefore, $r_t(p) > r_t(p)$. Thus, condition (1) holds. Condition (2) holds similar to case 1.2. LWD's buffer is congested. If a white packet is pushed out, we can drop it and run the case when the congestion did not occur as in case 1. If the pushed out packet is red then recolor a new packet p in red and apply case (ii). □

We can now state the main result of this section.

Theorem 5 *For a shared memory $n \times n$ switch with a buffer B, LWD is at most $1+\frac{B}{T+B}$-competitive, where T is the minimal number of packets transmitted between any two consecutive iterations.*

Proof By Lemma 3, during (t_s, t_e) OPT cannot transmit more packets than LWD. Note that during t_e it is possible that OPT transmits L more packets than LWD, $0 \leq L < N$. By definition of OPT, at the end of an iteration OPT gets all remaining $B-L$ packets for free, and its buffer is empty. By Lemma 2, between two consecutive iterations OPT cannot transmit more than LWD. So if OPT transmits $T \geq 0$ packets between two consecutive iterations, P packets during the iteration, the OPT's throughput is at most $T+P+B-L = T+2B$, whereas LWD transmits $T+P-L = T+B$. Thus, LWD is at most $1+\frac{B}{T+B}$-competitive. □

6 Scheduling with Heterogeneous Values

In this section, we consider a model with values: each incoming packet has an output port from 1 to n and an intrinsic value from 1 to V; in this model all packets have uniform processing requirements. The objective is to maximize the total transmitted value. Similar to the model with heterogeneous processing requirements, the work [12] showed that in the model with values LQD is at least $\left(\sqrt[3]{k} - o\left(\sqrt[3]{k}\right)\right)$-competitive. In Section 5, we have shown that LWD with PQ processing is 2-competitive in the model with heterogeneous processing requirements. Therefore, we begin with LWD's counterpart for this model: the Minimal-Total-Value-Drop policy (MTVD) that has packets in each queue sorted in

non-increasing order of values; MTVD tries to process and transmit packets with maximal value first but in case of congestion MTVD drops a packet with minimal value. Proofs of all results in this section can be found in the Appendix.

Minimal-Total-Value-Drop (MTVD): during the arrival of a packet p with output port i and value v, (1) if the buffer is not full, accept p into Q_i; (2) if the buffer is full and v exceeds the minimal value of some packet, push out a packet with the smallest value from the buffer and accept p into Q_i; else drop p.

For a single queue, MTVD is optimal by reasoning similar to LWD. Unfortunately, this does not generalize to the shared memory switch, as the following theorem shows.

Theorem 6 *The Minimal-Total-Value-Drop (MTVD) algorithm is at least* $\frac{Vn-(n-1)}{V}$*-competitive in the model with values (this is* $n - o(n)$ *unless* $V = o(n)$*).*

Proof In the first burst, there arrive B packets with value V for output port 1 and B packets with value $V - 1$ for every other output port $2..n$. MTVD accepts B packets to the first queue, while OPT accepts B/n packets to each queue. In B/n steps, MTVD will have transmitted total value BV/n, while OPT will have transmitted total value $(V+(V-1)(n-1))B/n$, and the first burst repeats, getting the bound. □

Theorem 6 shows that in the model with values the total value characteristic is insufficient and additional parameters should be included if an "ideal" online policy that achieves a constant competitiveness exists. This is why the work [12] introduced the Maximal-Ratio-Drop policy that considers both buffer occupancy and values as a potential policy that achieves constant competitiveness.

Maximal-Ratio-Drop (MRD): during the arrival of a packet p with output port i and value v, denote

$$j^* = \arg\max_j\{|Q_j|/V_j\},$$

where V_j is the total value of packets in queue j and $|Q_j|$ is the queue length; then: (1) if buffer is not full, accept p into Q_i; (2) if buffer is full and v exceeds the minimal value of a packet from queue Q_{j^*}, push out a packet from Q_{j^*} with minimal value and accept p into Q_i; else drop p.

Theorem 7 *The Maximal-Ratio-Drop (MRD) algorithm is at least* V*-competitive if* $n \geq B - V^2 + 1$.

Proof In the first burst, there arrive $2(m-1)$ packets of value 1 destined to output ports $[1, m-1]$, 2 packets per port, followed by B packets of value V destined to output port m, where $B > V$ is the buffer size and $m = B-V^2+1$. OPT accepts only packets of value V accruing the total value of BV. On the other hand, MRD accepts just V packets of value V at which point the ratio of the length to the average value becomes 1 and it retains $m-1$ packets of value 1 gaining the total value of V^2+m-1. Thus, the competitive ratio of MRD is $\frac{BV}{V^2+m-1} = V$. □

7 Simulations

To validate our theoretical results, we have conducted an extensive simulation study. While it would be best to use real life traces for testing, available datasets such as CAIDA [15] cannot be used for our setting. The problem is that real life traces, which describe packets independently of the networking hardware, neither can contain processing requirements, as these requirements heavily depend on specific router configurations, nor can provide sufficient information about the hardware and network processor configuration to determine the time scale (which packets go into the same time slot), which directly affects traffic burstiness. We have opted for simulations as opposed to deployments to understand the impact of traffic characteristics such as required processing since infrastructures are optimized with fairness in mind.

We have thus conducted four series of experiments on synthetic traces, studying how performance depends on maximal required processing k, buffer size B, number of queues n, and packet stream intensity λ_{on}. The actual optimal algorithm in our model is hard to compute, so as OPT we used an algorithm which is better than optimal: a single priority queue of size B that holds all packets, processes smallest packets first, and has n output ports (processing cores), where every port can process packets on every time slot regardless of output port label. This algorithm is optimal in the single queue model [17], so in our model it performs even better than optimal. Traffic was generated by 100 independent sources, each source a Markov-modulated Poisson process (MMPP) with intensity λ_{on} in the "on" state and 0 in the "off" state; MMPP generation is known to be an adequate synthetic model for network traffic [14, 32].

To model congested situations when some output queues become congested but others do not, we have used skewed heavy-tailed distributions: for the output port, we used the zeta (Zipf) distribution, $p(k \mid s) \propto k^{-s}$); for required processing, we randomly chose a subset of output ports where processing is expected to be high (with mean $3k/4$) and skewed the other ports to have packets with smaller required processing (with mean $k/4$). This setup lets us better model real life situations where peak loads often contain packets destined to a few output ports, and some ports have heavier packets than others.

We ran all experiments for 10^6 time slots with periodic "flushouts", in effect averaging the competitive ratio over 10 runs of 10^5 time slots each. In our experiments, this has proven to be sufficient for stable results.

Figure 2 shows simulation results in terms of competitive ratio, i.e., ratio between the number of packets transmitted by a given algorithm to the number of packets transmitted by the "better than optimal" reference algorithm OPT. Note that if the competitive ratio with "better than optimal" OPT is close to 1, and in our experiments the ratios are of-

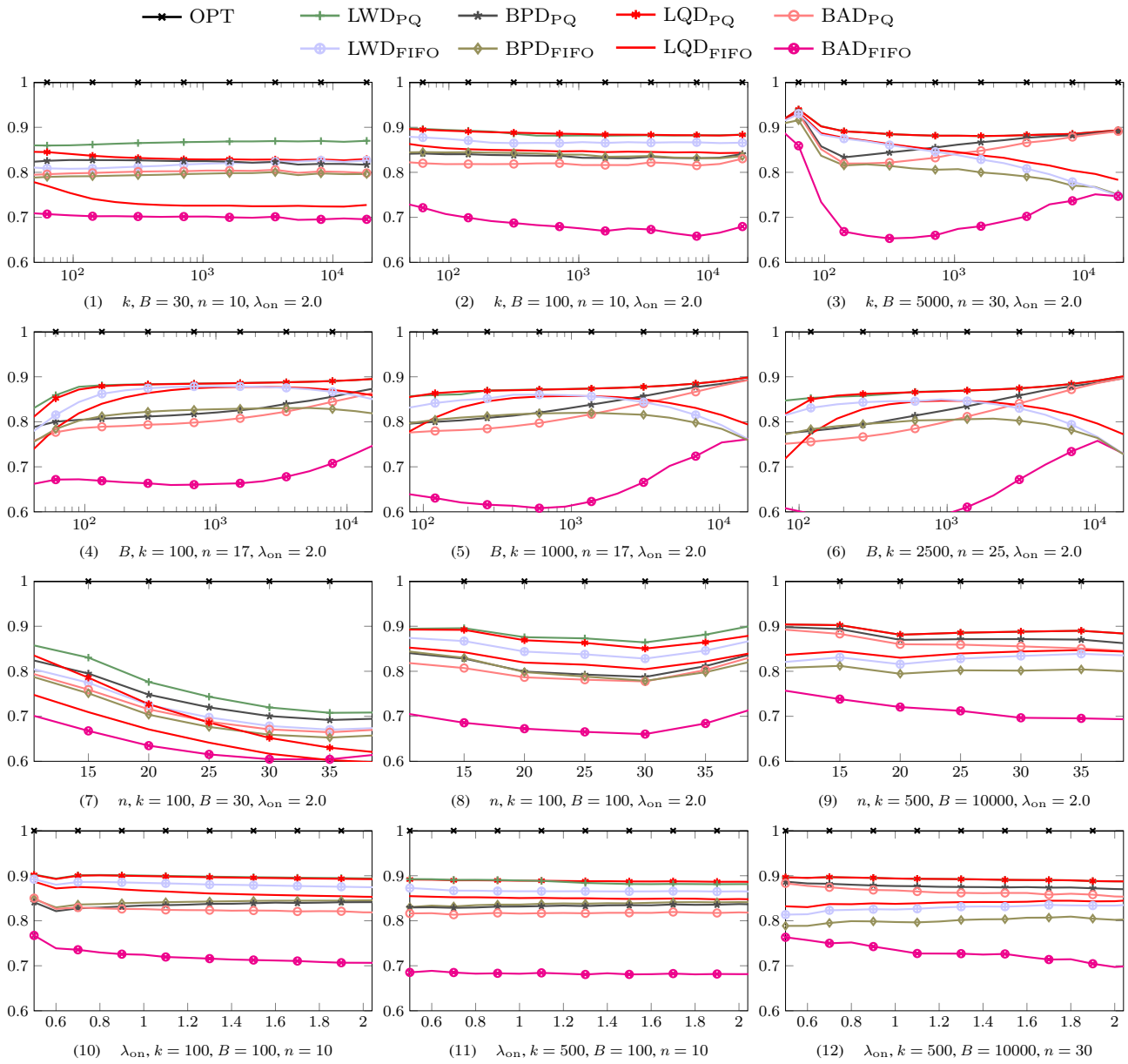

Fig. 2: Simulation results: ratio between number of successfully transmitted packets for processing models and number of packets transmitted by the optimal algorithm OPT as a function of (1-3) maximal required processing k, (4-6) buffer size B, (7-9) number of queues n, (10-12) packet stream intensity λ_{on}. Specific simulation parameters are shown in graph captions.

ten around 0.9, it is clear that the policy achieves excellent results in absolute terms, not only in comparison with other (perhaps simply equally bad) policies.

In the first set of simulations (Fig. 2(1-3)), we study performance as a function of the maximal required processing k. As expected, algorithms with PQ queues consistently outperform their FIFO counterparts. Among the algorithms themselves, the worst algorithm by far is the appropriately named BAD. With PQ queues, for smaller B BPD and LQD compete for second place and LWD outperforms them both; for large values of B LQD catches up with LWD: for larger buffers heavy packets are pushed out even by this version of LQD. With FIFO queues, LQD becomes noticeably worse and LWD has a much larger margin; for small B, LWD with FIFO queues exhibits performance close to the PQ-based policies. In the second set of simulations (Fig. 2(4-6)), we study performance as a function of the buffer size B; (4) and (5) show relatively small values of k, and (6) covers the case of large buffers and large required processing. Again, we see that FIFO queues lose to PQ queues, and BAD remains

the worst algorithm overall. LWD is again the best policy in this case both with PQ and FIFO queues for smaller values of B, but for very large buffers LQD catches up, and the FIFO version in some cases even overtakes LWD by showing the tradeoff between maximization of active ports versus minimization of total required processing in these settings. In the third set of experiments (Fig. 2(7-9)), we look at performance as a function of the number of queues n. All algorithms obviously degrade in performance as n increases compared to OPT because for OPT the increase in performance is virtually linear. The relative order of algorithms nevertheless remains the same; only LQD's performance drops in comparison to LWD and other policies as n increases. Again, for very large buffers, such as the one on graph (9), there is virtually no difference between LWD and LQD, but for smaller B LWD outperforms LQD quite significantly. Still, $\mathrm{LWD}_{\mathrm{PQ}}$ remains the best algorithm under all circumstances. The final, fourth set of experiments (Fig. 2(10-12)) shows how performance depends on the intensity λ_{on} of incoming packet sources when switched on. Here all algorithms, including OPT, show approximately the same rate of decline under increasingly high input intensities, and their relative standings remain the same across all intensities with the exception of LQD.

To sum up, we have tested the proposed algorithms against an "optimal" algorithm which is obviously superior to our model, across a wide variety of congested settings. In all experiments, the $\mathrm{LWD}_{\mathrm{PQ}}$ policy remained the best, and $\mathrm{LWD}_{\mathrm{FIFO}}$ was uniformly best across all policies with FIFO queues for small and medium B, while for very large buffers LQD performs as well as LWD and the FIFO version even outperforms the FIFO version of LWD. One of the most interesting aspects arising from our simulation results is the fact that they seem to imply that our worst case analysis has been beneficial in designing algorithms that work well also on average. In addition we observe that incorporation of required processing in heterogeneous environments does have a significant impact on performance (recall that even LQD is not oblivious to processing requirements during congestion).

8 Conclusion

Over the recent years, there has been a growing interest in understanding the impact of buffer architecture on network performance. The needs and (bursty) behavior of many modern data center applications further add incentive to fill this knowledge gap. In this work, we study the impact of heterogeneous processing on throughput in the shared memory switch architecture. We demonstrate that policies attractive under uniform processing requirements perform poorly in the worst case, which provides new insights to the practice of admission control policies. Our main result is a constant upper bound on the competitiveness of the LWD policy that drops packets from the queues with largest total processing in case of congestion; this is a significant improvement over [12], as our generalized model requires different proof methods. In addition, we consider a model with heterogeneous packet values and provide lower bounds that show that many natural candidates, proven to work well in other settings, fail to achieve constant competitiveness. Simulations confirm our analytical findings and in particular demonstrate the relevance of worst-case analysis results for understanding overall (average) performance.

References

1. Aiello, W., Kesselman, A., Mansour, Y.: Competitive buffer management for shared-memory switches. ACM Transactions on Algorithms **5**(1) (2008)
2. Aiello, W., Mansour, Y., Rajagopolan, S., Rosén, A.: Competitive queue policies for differentiated services. J. Algorithms **55**(2), 113–141 (2005)
3. Albers, S., Schmidt, M.: On the performance of greedy algorithms in packet buffering. SIAM Journal on Computing **35**(2), 278–304 (2005)
4. Azar, Y., Litichevskey, A.: Maximizing throughput in multi-queue switches. Algorithmica **45**(1), 69–90 (2006)
5. Azar, Y., Richter, Y.: An improved algorithm for CIOQ switches. ACM Transactions on algorithms **2**(2), 282–295 (2006)
6. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press (1998)
7. Chuprikov, P., Nikolenko, S.I., Kogan, K.: Priority queueing with multiple packet characteristics. In: INFOCOM, pp. 1418–1426 (2015)
8. Costa, P., Donnelly, A., Rowstron, A.I.T., O'Shea, G.: Camdoop: Exploiting in-network aggregation for big data applications. In: Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2012, San Jose, CA, USA, April 25-27, 2012, pp. 29–42 (2012)
9. Davydow, A., Chuprikov, P., Nikolenko, S.I., Kogan, K.: Throughput optimization with latency constraints. In: INFOCOM, pp. 1–9 (2017)
10. Englert, M., Westermann, M.: Lower and upper bounds on FIFO buffer management in QoS switches. Algorithmica **53**(4), 523–548 (2009)
11. Eugster, P., Kesselman, A., Kogan, K., Nikolenko, S.I., Sirotkin, A.: Essential traffic parameters for shared memory switch performance. In: SIROCCO, pp. 1–15 (2015)
12. Eugster, P.T., Kogan, K., Nikolenko, S.I., Sirotkin, A.: 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, pp. 471–480 (2014)
13. Goldwasser, M.: A survey of buffer management policies for packet switches. SIGACT News **41**(1), 100–128 (2010)
14. Heffes, H., Lucantoni, D.: A markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance. IEEE Journal on Selected Areas in Communications **4**(6), 856–868 (1986). DOI 10.1109/JSAC.1986.1146393
15. for Internet Data Analysis, C.T.C.A.: (2015). [Online] `http://www.caida.org/`
16. Irland, M.: Best effort and priority queuing policies for buffered crossbar switches. IEEE Transactions on Communications **26**(3), 328–337 (1978)
17. Keslassy, I., Kogan, K., Scalosub, G., Segal, M.: Providing performance guarantees in multipass network processors. IEEE/ACM Trans. Netw. **20**(6), 1895–1909 (2012)

18. Kesselman, A., Kogan, K.: Nonpreemptive scheduling of optical switches. IEEE Transactions on Communications **55**(6), 1212–1219 (2007)
19. Kesselman, A., Kogan, K., Segal, M.: Packet mode and QoS algorithms for buffered crossbar switches with FIFO queuing. Distributed Computing **23**(3), 163–175 (2010)
20. Kesselman, A., Kogan, K., Segal, M.: Best effort and priority queuing policies for buffered crossbar switches. Chicago Journal of Theoretical Computer Science **2012**(5), 1–14 (2012)
21. Kesselman, A., Kogan, K., Segal, M.: Improved competitive performance bounds for CIOQ switches. Algorithmica **63**(1-2), 411–424 (2012)
22. Kesselman, A., Lotker, Z., Mansour, Y., Patt-Shamir, B., Schieber, B., Sviridenko, M.: Buffer overflow management in QoS switches. SIAM Journal on Computing **33**(3), 563–583 (2004)
23. Kesselman, A., Lotker, Z., Mansour, Y., Patt-Shamir, B., Schieber, B., Sviridenko, M.: Buffer overflow management in QoS switches. SIAM J. Comput. **33**(3), 563–583 (2004)
24. Kesselman, A., Mansour, Y.: Harmonic buffer management policy for shared memory switches. Theor. Comput. Sci. **324**(2-3), 161–182 (2004)
25. Kogan, K., López-Ortiz, A., Nikolenko, S.I., Scalosub, G., Segal, M.: Large profits or fast gains: A dilemma in maximizing throughput with applications to network processors. J. Network and Computer Applications **74**, 31–43 (2016)
26. Kogan, K., López-Ortiz, A., Nikolenko, S.I., Sirotkin, A.V.: A taxonomy of semi-fifo policies. In: Proc. 31st IEEE International Performance Computing and Communications Conference, pp. 295–304. IEEE Press (2012)
27. Kogan, K., López-Ortiz, A., Nikolenko, S.I., Sirotkin, A.V., Tugaryov, D.: Fifo queueing policies for packets with heterogeneous processing. In: Proc. 1st Mediterranean Conference on Algorithms, Lecture Notes in Computer Science, vol. 7659, pp. 248–260. IEEE Press (2012)
28. Kogan, K., Nikolenko, S.I., Keshav, S., López-Ortiz, A.: Efficient demand assignment in multi-connected microgrids with a shared central grid. In: SustainIT, pp. 1–5 (2013)
29. Mansour, Y., Patt-Shamir, B., Lapid, O.: Optimal smoothing schedules for real-time streams. Distributed Computing **17**(1), 77–89 (2004)
30. Nikolenko, S.I., Kogan, K.: Single and multiple buffer processing. In: Encyclopedia of Algorithms. Springer (2015)
31. Pruhs, K.: Competitive online scheduling for server systems. SIGMETRICS Performance Evaluation Review **34**(4), 52–58 (2007)
32. ShahHeydari, S., LeNgoc, T.: Mmpp models for multimedia traffic. Telecommunication Systems **15**(3), 273–293 (2000). DOI 10.1023/A:1019199013546. URL `http://dx.doi.org/10.1023/A:1019199013546`
33. Sleator, D.D., Tarjan, R.E.: Amortized efficiency of list update and paging rules. Communications of the ACM **28**(2), 202–208 (1985)
34. Yang, H., Dasdan, A., Hsiao, R., Jr., D.S.P.: Map-reduce-merge: simplified relational data processing on large clusters. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007, pp. 1029–1040 (2007)
35. Yu, Y., Gunda, P.K., Isard, M.: Distributed aggregation for data-parallel computing: interfaces and implementations. In: SOSP, pp. 247–260 (2009)