Essential Traffic Parameters for Shared Memory Switch Performance

Patrick Eugster[1,2], Alex Kesselman[3],
Kirill Kogan[4], Sergey Nikolenko[5,6], and Alexander Sirotkin[7,8]

[1] Purdue University, p@cs.purdue.edu
[2] Technical University of Darmstadt*
[3] Google Inc., alx@google.com
[4] IMDEA Networks Institute, kirill.kogan@imdea.org
[5] National Research University Higher School of Economics, St. Petersburg
[6] Steklov Institute of Mathematics at St.Petersburg, sergey@logic.pdmi.ras.ru **
[7] International laboratory for Applied Network Research
National Research University Higher School of Economics, Moscow
[8] St. Petersburg Institute for Informatics and Automation of the RAS, St. Petersburg,
alexander.sirotkin@gmail.com

Abstract. Cloud applications bring new challenges to the design of network elements, in particular accommodating for the burstiness of traffic workloads. Shared memory switches represent the best candidate architecture to exploit buffer capacity; 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.

1 Introduction

Cloud data centers are faced with workloads which evolve rapidly, driven by high volumes of end users, application types, cluster nodes, and overall data movement (e.g., big data processing [4, 10]). A primary design challenge in this context consists in selecting and deploying network switches that scale application performance, in a way which is robust and cost-effective. A network switch receives packets on ingress ports, applies

* P. Eugster was partially supported by the German Research Foundation (DFG) under project MAKI ("Multi-mechanism Adaptation for the Future Internet").

** The work of Sergey Nikolenko was partially supported by the Government of the Russian Federation grant 14.Z50.31.0030 and the Presidential Grant for Leading Scientific Schools, NSh-3856.2014.1.

specific policies to them, identifies destination ports, and sends them out through egress ports. When application-induced traffic bursts create an imbalance between incoming and outgoing packet rates for a given port, packets must be queued in the switch packet buffer. The available queue size on a port determines the port's ability to hold packets until the egress port can emit it. When buffer queue is full, packets are dropped. The allocation and availability of buffer resources to the ports, determined by the buffering architecture, affects burst absorption capabilities and performance characteristics of the network switch. Overprovisioning in terms of buffer capacity at each network node to absorb bursty behavior is not viable, as networks do not have unlimited resources; conversely, cloud data centers can only scale out as fast as the effective per-port cost and power consumption. These factors, in turn, are driven by the chosen buffering architecture. The *shared memory switch* allows to absorb traffic bursts in the best way since the whole buffer can be utilized by a same output port if needed. Since this is an actual choice in practice [13], here we focus our efforts on this type of buffer architecture.

The *buffer management policy* is a key element in meeting network design challenges. It directly impacts a switch's ability to transfer data at line rate and optimize desired objectives during congestion under various traffic conditions. Most existing buffer management policies are based on a simple characteristic such as buffer occupancy [9, 16], whereas traffic workloads have additional important characteristics such as processing requirements or value that are not explicitly taken into account. Efficient methods for buffer management incorporating new characteristics in admission decisions beyond *fairness* objective functions lead to new challenges in performance and implementation for traditional switch architectures. Inherited from the Internet, fairness is in fact a design choice which can conflict with other objectives in various economic models (e.g., utilization of network infrastructure or profit [7, 19, 20]).

We thus consider a shared memory switch where a buffer of size B is shared among all types of traffic. Each arriving packet is labeled with an output queue. Arrivals can be adversarial. During arrival, packets concurrently "access" the shared memory. Each input port decides if its arriving packets should be admitted based on information computed by output ports in a distributed manner. In this work we consider (possibly weighted) throughput optimization since relevant objectives such as better reuse of underlying infrastructure or profit maximization can be reduced to throughput optimization [19, 20]. However, 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 an intrinsic value ("worth") or processing requirement. Moreover, we remove a strong constraint from the recent work of Eugster et al. [15] by allowing packets for the same output port to have distinct processing requirements. We consider the paradigm of *competitive analysis* [8, 37]: an algorithm ALG is α-*competitive* for some $\alpha \geq 1$ if for any arrival sequence the number of packets transmitted by ALG is at least $1/\alpha$ times the number of packets transmitted by an optimal offline clairvoyant algorithm OPT. Worst case analysis shows whether additional workload characteristics should be taken into account in buffer management. Since all policies we consider are greedy (they accept and transmit all traffic if there is no congestion), we need to consider extreme cases during congestion periods that can actually happen in scenarios like

big data processing [12,38,39]. Worst case results help define buffer management "rules of thumb" that are independent of specific arrival distributions.

The goal of this paper is to offer designs with proven performance guarantees for the shared memory switch; we analyze the performance of buffer management policies and provide guarantees for their worst-case throughput. We consider two different traffic characteristics: (a) required processing and (b) values for packet transmission. Intuitively, they should have similar impacts on the desired objective. In the case of a single queue, both of them reach optima when all packets are ordered by required processing [21] or values with push-out[9]. However, generalizing them to a shared memory buffering architecture is challenging: the case of heterogeneous values was presented as an open question in SIGACT News [18, p. 22].

In the first part of this paper every incoming packet has unit value (and an output port label) but has processing requirement varying from 1 to k. In this case the objective comes down to maximizing the number of transmitted packets. We show that LQD is at least $(n/2-o(n))$-competitive for sufficiently large buffer size B and maximal required processing k; besides that we 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 policy (that pushes out a packet with maximal required processing from a queue with maximal average processing requirements in the case of congestion) that achieves the same lower bound as BPD. All lower bounds hold even for PQ (priority queueing) processing order, where packets are ordered according to processing requirements. The main result of this work is the 2-competitiveness of a semi-greedy variant of the Longest-Work-Drop (LWD) policy of [15] 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 $(\log_{B/n} k)(1-1/B)+1$-competitive. In the second part of this paper (b) 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. Intuitively, the model with values should be similar to the model with required processing. However, we show that 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 [15] that considers both queue occupancy and the average value in the same queue. MRD was conjectured in [15] to have constant competitiveness. We show that the model with values has a different nature as a generalization from the single-queue case (where both models have optimal online algorithms) to the shared memory switch, and it is not enough to simply consider the total value. In particular, we prove that MRD is at least V-competitive.

The paper is organized as follows. Section 2 discusses related prior art. Section 3 details the model underlying our work. Section 4 considers lower bounds of several algorithms for packets with heterogeneous processing requirements to understand properties of an "ideal" policy. The main result of this paper — 2-competitiveness of LWD

[9] In case of packets with values, the optimality of the greedy algorithm with pushout is trivial: order the queue by value.

policy for packets with heterogeneous requirements — is presented in Section 5. Section 6 considers a model with heterogeneous packet values. Section 7 concludes the paper.

2 Related Work

Aiello et al. [28] 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. Kesselman and Mansour [1] demonstrate that the LQD policy is at most 2- and at least $\sqrt{2}$-competitive. Both works consider homogeneous packet processing, i.e., each packet requires a single processing cycle. Eugster et al. [15] 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 it was shown there that LQD is at least $\left(\sqrt{k} - o(\sqrt{k})\right)$-competitive. Fortunately, in [15] 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 [15] that Biggest-Packet-Drop (BPD, a policy that pushes out packets with maximal processing requirement in case of congestion) is at least $\log k$ competitive for $B > \frac{k(k+1)}{2}$. Unlike the model in [15] the model we consider in this paper however allows for packets with heterogeneous processing requirements to be admitted to the same queue. This generalization over the model in [15] has a significant impact on the efficiency of considered policies and applicability to real-world scenarios. In particular, we can open a separate queue per processing requirement per output port but in this case the scalability of maximal number of supported queues can become a strong constraint once k and n are growing. Our current work can be viewed as part of a larger research effort concentrated on studying competitive algorithms for management of bounded buffers. Surveys by Goldwasser [18] and later by Nikolenko and Kogan [36] provide an excellent overview of this field. Initiated in [26, 35], this line of research has received tremendous attention over the past decade. Various models have been proposed and studied, including QoS-oriented models where packets have individual weights [2, 14, 26, 35]. A related field that has recently attracted much attention focuses on various switch architectures and aims to design competitive algorithms for various scenarios therein (cf. [3,5,6,22–25,34]). However, none of these models cover the case of packets with heterogeneous processing requirements, and our work extends and generalizes previous models to heterogeneous processing. The single queue case with heterogeneous processing requirements is considered in [21,32,33]. Kogan et al. considered the multiple separated queues case with heterogeneous processing requirements in [30]. The single queue case with packets containing a combination of heterogeneous processing with packet lengths or values has considered in [11, 31].

3 Model Description

We consider an $n \times n$ shared memory switch with n 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 $B \geq n$.

Each output port manages a single 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 $p(d, w)$ arriving at an 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 ordered in non-decreasing order of required processing, or (ii) first-in-first-out (FIFO) processing order, where packets are ordered in the 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 each input port that decides which ones should be admitted based on the state computed by each output port in the distributed manner. The arrivals are adversarial and do not assume any specific traffic distribution (more than n arrivals are allowed at the same time slot). An accepted packet can be later dropped from the buffer when another packet is accepted instead; 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*. For every $1 \leq i \leq |A|$, we will further refer to element $a_i \in A$ or to $A[i]$ as the i-th element in the set A, as induced by the order. Given two ordered sets A and B, we say $A \geq B$, 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 [29] (Lemma 1 and Corollary 2).

Lemma 1. *For any two ordered sets A and B satisfying $A \geq B$, 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 $A' \geq B'$.*

Corollary 1. *For any two ordered sets A, B satisfying $A \geq B$, 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 $A \geq B'$.*

4 The Quest for an Ideal 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, 28]. 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. Lower bounds on the competitive ratio represent specific sequences of

packets on which the optimal algorithm is much better than the one in question. they are easier to prove than upper bounds since it suffices to present a hard instance of an input sequence, but lower bounds can still provide important information regarding the comparative quality of online algorithms.

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 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 and $i \neq j^*$, push out last packet from Q_{j^*} and accept p into Q_i; else drop p.

Note that the proposed here version of LQD is not fully 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. Proofs of all theorems in this section are given in the Appendix.

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

Proof. Over the first burst, there arrive B packets of each of the following kinds: $\boxed{1 \mid 1}$, $\boxed{k \mid 2}$, $\boxed{k \mid 3}$, $\ldots$, $\boxed{k \mid n}$. 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 $1 \times \boxed{k \mid 2}$, $1 \times \boxed{k \mid 3}, \ldots, 1 \times \boxed{k \mid n}$, 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 estimate 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 $\frac{(B-n+1)+(n-1)\times\frac{B-n+1}{n(n-1)}}{\frac{B}{n}+(n-1)\times\frac{B-n+1}{n(n-1)}} = \frac{(B-n+1)+\frac{B-n+1}{n}}{\frac{B}{n}+\frac{B-n+1}{n}} \geq \frac{(B-n+1)}{2\frac{B}{n}} \approx n/2$.

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

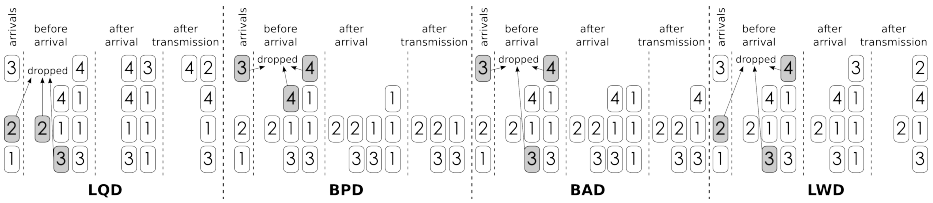

Fig. 1: A sample time slot of Longest Queue Drop (LQD), Biggest Packet Drop (BPD), Biggest Average Drop (BAD), and Largest Work Drop (LWD) policies with maximal processing $k = 4$, $n = 4$ output ports, and a shared buffer of size $B = 8$. Queues for each output port are shown horizontally. Shaded packets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

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 counterexample 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 $j^* = \arg\max_j\{W_j + \mathbb{1}_{i=j}w\}$ where $\mathbb{1}_{i=j} = 1$ 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_i, 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 all $(a-1)$ 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 t should be interpreted as the arrival of a single packet. If several packets arrive at the same time slot, we consider them independently, in the sequence in which they arrive. A time slot is divided into time units; arrival of each packet is a separate time unit (so the arrival phase takes up several time units), while processing and transmission phases both use only a single time unit (we do not separate them). 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 is congested and ends on the first time unit t_e when G has transmitted at least B packets since t_s. To simplify analysis, G 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 this section we show an upper bound, so weakening the algorithm only makes things worse for us.

In what follows we consider an artificially enhanced version of OPT: (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, the first packet q (in PQ order) is transmitted out of the i-th port of OPT (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. 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 B plus at most $n-1$ packets, dropping all buffered packets at time t_d adds at most 1 to the competitiveness of G. The general idea of our analysis here is similar to [27] 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 significantly simplifies analysis of online buffer management policies with various characteristics. The major question that we will soon answer is: is there a policy with a constant competitiveness in the model where each packet has both required processing and output port (admission of heterogeneous packets to the same queue is allowed)? Note that the processing order implemented in each queue has significant impact on the performance of a scheduling policy. 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. 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*. 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 (proven in the Appendix). Actually, we prove an even stronger result that 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 an 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) $S^{OPT}_{[t_b,t]} \leq S^G_{[t_b,t]}$; (2) for any $i \in [1, n]$, at time t $G_i \leq O_i$ and $|G_i| \geq |O_i|$.*

Proof. The proof proceeds by induction on the number of time units. *Base:* 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. *Hypothesis:* Assume that the lemma holds during $[t_b, t)$, $t \in [t_b, t_s - 1]$. *Step:* We are to show that the lemma holds during 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 Q_j in OPT or G, $1 \leq j \leq n$. If both O_j and G_j are nonempty just before t, by the induction hypothesis we have $O_j[1] \geq G_j[1]$. 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 $G_i \leq O_i$ and $|G_i| \geq |O_i|$. Thus, if OPT accepts (G accepts since G is greedy and

there is no congestion between two consecutive iterations), by Lemma 1(ii) condition (2) follows at time unit t. If OPT does not accept p to O_i, condition (2) follows by Corollary 1(ii).

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, so we have to prove (1) in Lemma 2. Part (2) of Lemma 2 will be used in the proof of Lemma 3.

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. For any time interval $I' = [t_s - 1, t]$, $t \in [t_s - 1, t_d)$, during an iteration we say that $B - S_{I'}^{LWD}$ packets with minimal latency in LWD buffer are colored in red; any other packet in LWD buffer is colored in white. Note that packets that ceased 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; due to space constraints, its proof is given in the Appendix.

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

Lemma 3. *For every OPT packet $p_j \in O_i$, $j \in [1, |O_i|]$ and $i \in [1, n]$, at time unit $t \in (t_s, t_e)$ either (1) there is a red packet $q_j \in L_i$, $r_t(p_j) \geq r_t(q_j)$ ($R_i \leq O_i$), or (2) for any red packet q at LWD buffer, $r_t(p_j) + W(R_i) \geq \mathrm{lat}^t(q)$.*

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**: suppose that the t-th time unit is devoted to processing all HOL packets and transmitting fully processed packets. In this case, either every nonempty queue L_j is active (in this case O_j is active too regardless of how many processing cycles remains in HOL packet of O_j by definition of OPT (property (3))) 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 induction 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 condition (1) follows by 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 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).

The main result of this section is the following theorem (see proof in Appendix).

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 [15] 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 [15] 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 $\frac{BV}{V^2+m-1} = V$.

Unfortunately, the MRD example shows that even both values and buffer occupancy together are not enough to achieve constant competitiveness. As a result, we are more pessimistic regarding the existence of a policy in this model with constant competitiveness (the open problem posed in SIGACT News [18, p. 22]).

7 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 tradeoffs inevitable on the path to a "perfect" policy in a shared memory switch, both analytically and with simulations. Recent research advocates smaller buffers in routers, aiming to reduce queueing delay in the presence of (mostly) TCP traffic; however, it sidesteps the issue that as buffers get smaller, the effect of processing delay becomes much more pronounced. The majority of currently deployed admission control policies do not take into account (at least explicitly) the importance of heterogeneous packet processing. 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 [15], as our generalized model requires different proof methods. In addition, we consider a model with heterogeneous packet values and provide preliminary results on whether a policy with constant competitiveness can exist. Simulations confirm our analytical findings and in particular demonstrate the relevance of worst-case analysis results for understanding overall (average) performance.

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. *J. Algorithms*, 55(2):113–141, 2005.
3. Susanne Albers and Markus Schmidt. On the performance of greedy algorithms in packet buffering. *SIAM Journal on Computing*, 35(2):278–304, 2005.
4. Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, and George Varghese. CONGA: distributed congestion-aware load balancing for datacenters. In *ACM SIGCOMM 2014 Conference*, pages 503–514, 2014.
5. Yossi Azar and Arik Litichevskey. Maximizing throughput in multi-queue switches. *Algorithmica*, 45(1):69–90, 2006.
6. Yossi Azar and Yossi Richter. An improved algorithm for CIOQ switches. *ACM Transactions on algorithms*, 2(2):282–295, 2006.
7. BBC News. US Watchdog to Propose New Net Neutrality Rules, 2014. `http://www.bbc.com/news/technology-27141121`.
8. Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
9. Wu chang Feng, Dilip D. Kandlur, Debanjan Saha, and Kang G. Shin. Stochastic fair blue: A queue management algorithm for enforcing fairness. In *INFOCOM*, pages 1520–1529, 2001.
10. Mosharaf Chowdhury, Yuan Zhong, and Ion Stoica. Efficient coflow scheduling with varys. In *SIGCOMM*, pages 443–454, 2014.
11. Pavel Chuprikov, Sergey I. Nikolenko, Kirill Kogan. Priority Queueing with Multiple Packet Characteristics. In *INFOCOM*, pages 1–9, 2015.
12. Paolo Costa, Austin Donnelly, Antony I. T. Rowstron, and Greg O'Shea. Camdoop: Exploiting in-network aggregation for big data applications. In *Proc. 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2012)*, pages 29–42, 2012.
13. Sujal Das and Rochan Sankar. Broadcom smart-buffer technology in data center switches for cost-effective performance scaling of cloud applications, 2012. `https://www.broadcom.com/collateral/etp/SBT-ETP100.pdf`.
14. Matthias Englert and Matthias Westermann. Lower and upper bounds on FIFO buffer management in QoS switches. *Algorithmica*, 53(4):523–548, 2009.
15. Patrick Eugster, Kirill Kogan, Sergey Nikolenko, and Alexander Sirotkin. Shared memory buffer management for heterogeneous packet processing. In *ICDCS*, 2014.
16. Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. pages 397–413, 1993.
17. CAIDA The Cooperative Association for Internet Data Analysis. [Online] `http://www.caida.org/`.
18. Michael Goldwasser. A survey of buffer management policies for packet switches. *SIGACT News*, 41(1):100–128, 2010.
19. Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving high utilization with software-driven WAN. In *ACM SIGCOMM 2013 Conference*, pages 15–26, 2013.
20. Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jon Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. B4: experience with a globally-deployed software defined wan. In *ACM SIGCOMM 2013 Conference*, pages 3–14, 2013.
21. 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.

22. 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.
23. Alexander Kesselman, Kirill Kogan, and Michael Segal. Improved competitive performance bounds for CIOQ switches. *Algorithmica*, 63(1-2):411–424, 2012.
24. Alexander Kesselman, Kirill Kogan, and Michael Segal. Best Effort and Priority Queuing Policies for Buffered Crossbar Switches. *Chicago Journal of Theoretical Computer Science*, 2012.
25. Alexander Kesselman, Kirill Kogan. Nonpreemptive Scheduling of Optical Switches. *IEEE Transactions on Communications*,55(6): 1212-1219, 2007.
26. 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.
27. Alexander Kesselman, Zvi Lotker, Yishay Mansour, Boaz Patt-Shamir, Baruch Schieber, and Maxim Sviridenko. Buffer overflow management in QoS switches. *SIAM J. Comput.*, 33(3):563–583, 2004.
28. Alexander Kesselman and Yishay Mansour. Harmonic buffer management policy for shared memory switches. *Theor. Comput. Sci.*, 324(2-3):161–182, 2004.
29. Kirill Kogan, Alejandro López-Ortiz, Sergey Nikolenko, Gabriel Scalosub, and Michael Segal. Large profits or fast gains: A dilemma in maximizing throughput with applications to network processors. *CoRR*, abs/1202.5755, 2013.
30. Kirill Kogan, Alejandro López-Ortiz, Sergey Nikolenko, Alexander Sirotkin. Multi-queued network processors for packets with heterogeneous processing requirements. In *COMSNETS*, pages 1–10, 2013.
31. Kirill Kogan, Alejandro López-Ortiz, Sergey Nikolenko, Gabriel Scalosub, and Michael Segal. Balancing work and size with bounded buffers. In *COMSNETS*, pages 1–8, 2014.
32. Kirill Kogan, Alejandro López-Ortiz, Sergey Nikolenko, Alexander Sirotkin, Denis Tugaryov. FIFO Queueing Policies for Packets with Heterogeneous Processing. In *MedAlg*, pages 248–260, 2012.
33. Kirill Kogan, Alejandro López-Ortiz, Sergey Nikolenko, Alexander Sirotkin. A taxonomy of Semi-FIFO policies. In *IPCCC*, pages 295–304, 2012.
34. Kirill Kogan, Sergey Nikolenko, Srinivasan Keshav, Alejandro López-Ortiz. Efficient demand assignment in multi-connected microgrids with a shared central grid. In *SustainIT*, pages 1–5, 2013.
35. Yishay Mansour, Boaz Patt-Shamir, and Ofer Lapid. Optimal smoothing schedules for real-time streams. *Distributed Computing*, 17(1):77–89, 2004.
36. Sergey I. Nikolenko, Kirill Kogan. *Single and Multiple Buffer Processing*. Encyclopedia of Algorithms, Springer 2015.
37. Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.
38. Hung-chih Yang, Ali Dasdan, Ruey-Lung Hsiao, and Douglas Stott Parker Jr. Map-reduce-merge: simplified relational data processing on large clusters. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 1029–1040, 2007.
39. Yuan Yu, Pradeep Kumar Gunda, and Michael Isard. Distributed aggregation for data-parallel computing: interfaces and implementations. In *SOSP*, pages 247–260, 2009.