Exploring Token-Oriented In-Network Prioritization in Datacenter Networks

Kexin Liu, Bingchuan Tian, Chen Tian, Bo Li, Qingyue Wang, Jiaqi Zheng, Jiajun Sun, Yixiao Gao, Student Member, IEEE, Wei Wang, Guihai Chen, Wanchun Dou, Yanan Jiang, Huaping Zhou, Jingjie Jiang, Fan Zhang, and Gong Zhang

Abstract—In memory computing and high-end distributed storage demand low latency, high throughput, and zero data loss simultaneously from datacenter networks. Existing reactive congestion control approaches cannot both minimize queuing latency and ensure zero data loss. A token-oriented proactive approach can achieve them together by controlling congestion even before sending data packets. However, state-of-the-art token-oriented approaches only strive to optimize network-level metrics: maximizing throughput while achieving flow-level fairness. This article answers the question of how to support objective-aware traffic scheduling in token-oriented approaches. The novelty of Token-Oriented in-network Prioritization (TOP) is that it prioritizes tokens instead of data packets. We make three contributions. Via simulations over a hypothetical TOP system, our first contribution is demonstrating the potential performance gain that can be brought by TOP. Second, we investigate the applicability of TOP. Although the overhead of enabling necessary TOP features in switches is trivial, we find that mainstream commodity datacenter switches do not support them. We hence propose a readily-deployable remedy to achieve in-network prioritization by pushing both switch and end-host hardware capacity to an extreme end. Lastly, we implement a running TOP system with Linux hosts and commodity switches, and evaluate TOP in testbeds and with large-scale simulations for various scenarios.

Index Terms—Token-oriented proactive congestion control, datacenters

1 Introduction

Motivation. Datacenters have evolved rapidly over the past few years. In-memory computing [1], [2] and high-end distributed storage [3], [4] demand low latency, high throughput, and zero data loss simultaneously from datacenter networks. Catering to this trend, new kernel-bypassing technologies such as Data Plane Development Kit (DPDK) and Remote Direct Memory Access (RDMA) become promising [5], [6], [7], [8], [9], [10], [11], [12].

End-to-end latency consists of network delay and host protocol stack's processing time. Network delay consists of transmission delay, propagation delay and queuing delay. By implementing the whole protocol stack in user-space or host NICs, DPDK / RDMA etc. provide both lower host latency and higher per-connection throughput with reduced CPU consumption [13], [14], [15], [16], [17], [18]. Under this circumstance, the end-to-end latency is dominated by network delay. In datacenter networks, transmission and propagation delays are almost negligible. However, how to reduce in-network queuing delay remains a problem, which requires a clever congestion control mechanism in end-host, together with switches' coordination. Most datacenter networks are oversubscribed [19] and congestion is not uncommon: packet drops due to congestion can be observed when the whole network utilization is around only 25 percent [20].

Existing large body of reactive congestion control approaches cannot both minimize queuing latency and ensure zero data loss. To control data queue length, the mainstream approach is to react to congestion signals such as ECN [15], [21], [22] or network delay [17], [23]. This approach alone is not lossless when there are bursty flow arrivals or incast traffic [24]. Thus, the Priority Flow Control (PFC) feature of Ethernet switches / NICs needs to be enabled to prevent packet loss. Consequently, when congestion happens, data queues can build up and PFC pause frames can be propagated. They both significantly increase network latency [14].

A token-oriented proactive approach can achieve low queuing latency and zero data loss together by controlling congestion even before sending data packets [24]. We illustrate the contribution chains from mechanisms to features, and from features to benefits in Fig. 1. With receiver proactive congestion control, a receiver sends per-flow tokens to each sender in an end-to-end fashion. These tokens are carefully paced by the receiver to interleave the arrival time of incoming data packets on the last downlink. With switch proactive congestion control, a switch rate-limits and paces all passing tokens on each link, so that the aggregated bandwidth of reverse direction data packets is no more than the available

- K.Liu, B.Tian, C.Tian, B.Li, Q.Wang, J.Zheng, J.Sun, Y.Gao, W.Wang, G.Chen, W.Dou, Y.Jiang, and H.Zhou are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu 210008, China. E-mail: {mg1733038, bctian, mrgaoxx}@smail.nju.edu.cn, {tianchen, jzheng, ww, gchen, douwc}@nju.edu.cn, btlcmr0702@hotmail.com, {2748508366, jiangyanan52239088, 610450964}@qq.com, sunjane001@live.com.
- J. Jiang, F. Zhang, and G. Zhang are with the Huawei Technologies Company, Ltd, Shenzhen 518129, China. E-mail: {jiang.jingjie, zhang.fan2, nicholas.zhang}@huawei.com.

Manuscript received 19 Mar. 2019; revised 17 Nov. 2019; accepted 6 Dec. 2019. Date of publication 10 Dec. 2019; date of current version 20 Jan. 2020. (Corresponding author: Chen Tian.)
Recommended for acceptance by J. Zhai.
Digital Object Identifier no. 10.1109/TPDS.2019.2958899

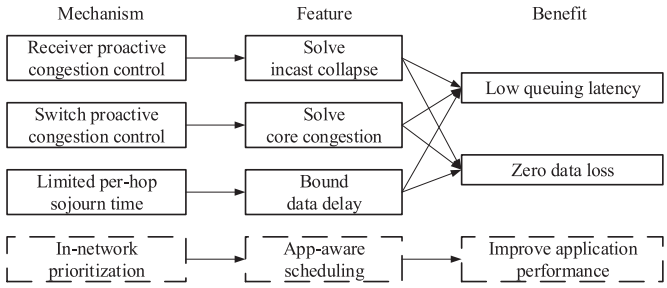


Fig. 1. From mechanisms to features then to benefits.

capacity. A sender strictly follows the token allocation and sends one full-length data packet when receiving a new valid token. Incast problem is elegantly solved. And ideally, there should be nearly zero queuing latency and data loss. While, a queue can build up when passing flows have different round-trip time (RTT). Network topologies, host characteristics and the maximum sojourn time of tokens in each queue all affect RTT difference. By capping the length of token queues, per-queue sojourn time can be limited. Then per-hop data delay can be bounded. This bound can be used to calculate per-hop buffer requirement to ensure zero data loss.

State-of-the-art token-oriented approaches (e.g., ExpressPass [24]) only optimize network metrics: maximizing throughput while achieving flow-level fairness. Meanwhile, various objective-aware data packet scheduling approaches have been proposed. Some reduce Flow Completion Time (FCT) [17], [21], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34]; some add deadline awareness [28], [35], [36], [37]; some even schedule at the granularity of coflow [38], [39], [40].

This paper answers the question of how to support objective-aware traffic scheduling in token-oriented approaches. The novelty of Token-Oriented in-network Prioritization (TOP) is that it prioritizes tokens instead of data packets. Each switch can perform scheduling over all passing tokens so that the system can determine the priorities of reverse direction flows on each link, hence improving application performance (Fig. 1).

Our Contributions. Existing commodity switches usually support 8 physical priority queues. To support objective-aware traffic scheduling, TOP should classify tokens into different queues and perform preemptive scheduling among all queuing tokens through the Strict Priority (SP) mode. Note that without in-network prioritization, proactive congestion control and limit per-hop sojourn time are feasible with commodity hardware (as demonstrated by ExpressPass [24]). While to support TOP, there are new challenges. First, TOP needs to rate-limit all output tokens, in a certain group of queues of a port, as a whole. Otherwise, proactive congestion control is compromised. Second, TOP needs to bound token latency caused by prioritization. Without care, a naive design may block a token in a low priority queue for an arbitrarily long time in SP mode. This situation pushes token delay unbounded and in turn compromises the zero data loss goal (Section 2).

Via simulations over a hypothetical TOP system, our first contribution is demonstrating the potential performance gain that can be brought by TOP. We assume a hypothetical switch that can meet the two requirements mentioned above. Compared with state-of-the-art approaches, TOP can reduce the average FCT by 44.24 percent, the deadline missing ratio by 79.18 percent, and the average coflow completion time by 58.88 percent. It is close to the upper bound of performance, where an ideal switch supports an infinite number of priorities (similar to pFabric [31]) (Section 3).

Second, we investigate the applicability of TOP. An interesting observation is that although the overhead of enabling necessary TOP features in switches is trivial, mainstream commodity datacenter switches do not support them. We thus propose a readily-deployable remedy solution to achieve in-network prioritization by pushing both switch and end-host hardware capacity to an extreme end. With a novel combinatorial configuration of existing switch capabilities, TOP can accurately rate-limit all tokens from multiple priority queues in a port, with a single rate-limiter. By translating multi-hop queuing latency to an end-to-end one-way latency threshold, a sender can drop tokens if their arrival time exceed a given threshold to bound token latency. Design details such as packet loss and optional unscheduled packets are also discussed to complete the whole picture (Section 4).

Lastly, we implement a running TOP system with Linux hosts and commodity switches, and evaluate TOP in testbeds and with large-scale simulations. TOP works well in a wide-range, such as symmetric-routing and multi-path topologies, incast scenarios, high-fan in networks etc. (Section 5).

2 Background and Motivation

We start with a review of state-of-the-art objective-aware packet scheduling designs (Section 2.1). We then describe the key aspects and problems of token-oriented approaches (Section 2.2).

2.1 Objective-Aware Packet Scheduling

Instead of traditional network-level metrics such as maximizing throughput and achieving flow-level fairness, datacenter networks can be optimized for different objectives to meet applications' requirements better. Note that this paper prefers distributed scheduling approaches. The applicability of a centralized scheduling approach to nowadays large and high-speed networks is yet to be proven [33].

FCT-Oriented. For many applications, we need to minimize the average Flow Completion Time (FCT) to maximize application throughput [21]. For tiny flows (i.e., less than tens of packets), one-way network latency is critical. Some works strive to keep switch queue length at a low level to reduce packet latency [21], [25], [26], [27]. For other flows, their network throughput dominate their FCT. To mimic the Shortest-Job-First (SJF) or Shortest-Remaining-Time-First (SRTF) principals, existing approaches usually prioritize small flows over large flows to reduce the average FCT [28], [30], [31], [32], [34].

Deadline-Oriented. For a partition-aggregate workflow, flow deadline missing ratio directly impacts the quality of data processing results [35]. D^3 pioneers the idea of incorporating deadline awareness into network scheduling [35]. To mimic the Earliest-Deadline-First (EDF) principal in a single bottleneck, existing approaches usually prioritize flows with earlier deadlines over flows with later deadlines.

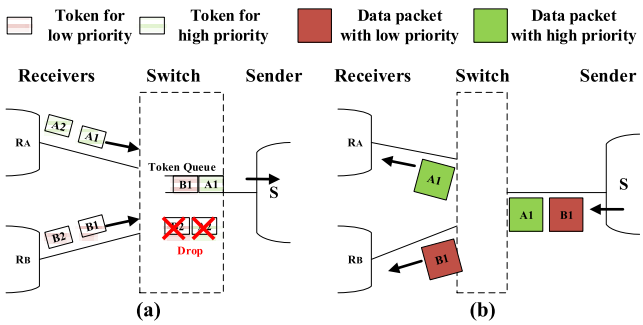


Fig. 2. ExpressPass has no in-network prioritization.

Coflow-Oriented. The coflow abstraction considers the all-or-nothing application semantics for all flows in a communication stage of a given job [41]. Following the SJF / SRTF or EDF principals, we can also minimize the average Coflow Completion Time (CCT) [38], [39], [42] or minimize the deadline missing ratio of coflows [38].

2.2 Token-Oriented Work

Token-oriented approaches [24], [43], [44], [45] are promising because of their proactive congestion control. A token (or, credit in ExpressPass, grant in Homa and PULL packet in NDP) is a special control packet sent from a receiver to a sender. A typical token-oriented flow proceeds as follows (e.g., pHost [43] and ExpressPass [24]). On the arrival of a new flow, a dedicated signal packet is sent from the sender to the receiver as a notification together with flow information such as flow size and / or deadline. Or, new flow information can be embedded in the first several kick-start data packets (i.e., NDP [44] and Homa [46]). After this hand-shaking procedure, all remaining packets are scheduled by the receiver. Upon reception of a token, the sender can correspondingly send a full-length data packet to the receiver.

There are three available scheduling dimensions along a specific token's processing procedure. Receiver: when there exist multiple incoming flows, a receiver can choose which flow to allocate the next token. Switches: when there exist multiple token packets in the queue, a switch can choose which token to be paced out first. Sender: when there exist multiple tokens for different outbound flows, a sender can choose which token to be consumed for the next sending slot. Note that sender scheduling can be invalidated if switches already rate-limit the tokens for the last downlink to a sender: the sender just needs to response to every incoming token.

ExpressPass. To provide proactive congestion control, a switch rate-limits and paces all passing tokens on each link, so that the aggregated bandwidth of reverse direction data packets is no more than the available capacity [24]. Symmetric routing ensures the corresponding token flow is on the exact reverse direction of each data flow. The token queue rate is limited to around 5 percent of the link capacity, calculated by the length of a token versus a full length data packet. Each switch queue builds up when passing flows have different round-trip time (RTT). Three factors can contribute to flow RTT differences: 1) the difference in path lengths, 2) the variance in token processing time at the host and 3) the sojourn time difference of token queuing in switches. The first two factors are bounded by network topologies and host characteristics. ExpressPass limits the length of each token queue to 8, which is a trade-off point among utilization, convergence time, and data queue occupancy. Thus, the third factor is also bounded. Zero data loss can be ensured by allocating an amount of packet buffer corresponding to the bounded delay.

TABLE 1
Feature Comparison

	pHost	NDP	ExpressPass	Homa	TOP
Solve incast	N	Y	Y	N	Y
Solve core congestion	N	N	Y	N	Y
Bound data delay	N	N	Y	N	Y
In-network prioritization	N	N	N	Y	Y
Readily-deployable	Y	N	Y	Y	Y

Fig. 2 illustrates the problem of lacking support to in-network prioritization. There are two flows $R_A \rightarrow S$ and $R_B \rightarrow S$. All links have equal capacity. R_A's flow has a higher application priority than the flow of R_B. Assuming in each time window, two packets can be transmitted by the sender to receivers. R_A and R_B each generates two tokens. Switch rate-limiter drops tokens A_2 and B_2 to match the capacity of the reverse link capacity (shown in Fig. 2a). Then the sender generates exactly one data packet for each receiver that can go through the bottleneck link during the time window (shown in Fig. 2b). If in-network token prioritization exists, we can drop token B_1 instead of A_2. Thus, without changing the completion time of flow $R_B \rightarrow S$, flow $R_A \rightarrow S$'s completion time can be significantly reduced. Hence the application performance can be improved.

In this paper we focus on lossless token-oriented approaches such as ExpressPass. There are other approaches. We list the comparison of previous approaches in Table 1.

pHost and Homa. A non-blocking network core is assumed to provide full bisection bandwidth. Thus, the receiver and the sender are two scheduling points for each flow [43]. To improve short flows' performance, each flow also has a few "free tokens" assigned so that the first several data packets can be sent without waiting for scheduling from the receiver. A token expires if the sender has not used it within 1.5× MTU-sized packet transmission time after its reception. pHost does consider optimization for FCT and deadline scenarios by also following SJF / SRTF / EDF principals.

In a practical oversubscribed network, pHost generates tokens without considering core congestion. Under medium to heavy load scenarios, packets accumulate in core switch queues. Eventually, it leads to packet loss. Also, large queue length effectively negates the latency benefit brought to short flows by free tokens. More specifically, pHost faces the incast problem as free tokens belong to many concurrent flows can cause collision at a common receiver.

Built on pHost, Homa targets RPC-like scenarios, where most bytes are from extremely small flows with several KBs. The major difference is that Homa uses priorities for unscheduled / scheduled packets. Still, it requires non-blocking topologies and it does not handle incast and core congestion.

NDP. NDP [44] builds on several existing technologies. The first is Cut-Payload [47], which trims the payload of a dropped packet and uses the header to precisely inform the receiver about the loss event. The second is per-packet load balancing over a non-blocking network core. NDP allows each flow to start sending a Bandwidth-Delay-Production

(BDP) worthy of packets at full link rate without tokens. As a result, NDP can save one RTT delay for tiny flows. By limiting the length of data packet queues, NDP can constrain the one-way delay for either a data packet or its header (after loss). If dropped, a data packet needs to be pulled by the receiver again or wait for retransmission timeout. All control packets, including ACK / NACK / PULL packets, are sent with a high packet priority. NDP sets the length of a control packet queue larger than 1,000.

As a clean-slate solution, NDP significantly changes switch behavior hence it is not readily-deployable. The unscheduled first RTT packets from new flows may repetitively cause scheduled data packet drops of a victim flow for a (theoretically) arbitrary long time. NDP's performance severely degrades in an oversubscribed network. The reason is that NDP can generate overwhelming amount of control packets including trimmed headers in fan-in networks. This large number of trimmed headers can still overflow large control packet queues. The lost of ACK / NACK packets causes control deadlock which significantly affects the throughput.

Other Works. There are other works aimed at utilizing the bandwidth [48], offloading packet processing [49] and ensuring fast interactive response time [50]. These approaches are valuable complementary works to our token-oriented design.

2.3 Reactive Works

Reactive congestion control approaches react to congestion signals such as ECN [15], [21], [22] or network delay [17], [23]. In other words, these approaches begin taking into effect after congestion happens, which usually causes long queueing delay and even packet loss. Thus the Priority Flow Control (PFC) needs to be enabled to prevent packet loss. Unfortunately, PFC has some essential limitations like head-of-line blocking and deadlock. Long queueing delay and PFC both introduce network latency thus affecting the flow completion time.

HPCC. HPCC is a state-of-art reactive congestion control approach leveraging in-network telemetry (INT) to obtain more precise link load information. Like other reactive approaches, HPCC allows each flow to send one RTT at first. During the propagation of the data from the sender to the receiver, each switch along the path leverages INT to insert link information to the data. The information is sent back by ACK. After the sender receives ACK, it uses the network information to adapt the flow rate.

HPCC does not prioritize small flows. And any returned measurement information takes no effect on such small flows. Overreaction is also found in HPCC thus wasting bandwidth and hurting the performance of large flows.

3 Potential Gain of TOP

The first part presents a hypothetical TOP system that meets our requirements (Section 3.1). Details of performance evaluation methodology are given in the next part (Section 3.2). Via simulations over the hypothetical system, we demonstrate the potential performance gain (Section 3.3).

3.1 A Hypothetical TOP System

Overview. TOP is built on top of ExpressPass by prioritizing tokens of different flows to enforce objective-aware packet scheduling. We first introduce some terms in Table 2. Volume of signal packets is usually negligible. Network traffic is mainly composed by two kinds of packets: token packets, and data packets. Each full-length data packet is a ONE-TO-ONE response to a token packet. To make fully use of link, the rate of token packets must be at least $\frac{S_{token}}{S_{token}+S_{data}} \approx 5\%$ of link bandwidth, where S_{token} and S_{data} represent the average sizes of token packets and data packets, respectively. On the other hand, the rate of token packets cannot exceed 5 percent, otherwise certain switches must be congested by returned data packets.

TABLE 2
Term Definition

Signal packet	e.g., Ready-To-Send sent from sender to receiver with flow information when establishes a connection.
Token/Credit/ Grant/PULL	a special packet from receiver to sender to control the sending rate of data, used in token-oriented protocols.
Sojourn time	time duration of which a packet enters and leaves a queue.
Obsolete/Expired token	a token whose sojourn time exceeds a predefined threshold.

TOP prioritizes tokens instead of data to ensure zero data loss. If data is prioritized as that in HOMA, lower priority data can be blocked by higher priority data. The unbounded queuing delay eventually leads to packet drop. Because of strict priority queuing, fast retransmission can not be used. It requires a large timeout (usually a few milliseconds) to start retransmission. The completion time of lower priority flows can be largely delayed.

A Hypothetical Switch. Shown in Fig. 3 is such a switch. We use the highest priority queue Q7 for signal packets. We keep one priority (e.g., Q6 in Fig. 3a or Q0 in Fig. 3b) for data packets, and 6 priorities can be used by applications to differentiate tokens for different flows. We briefly discuss it later in Section 3.2. There are two features required for this switch.

R1: queue group rate-limiter. For each port, there are two rate-limiters, one for tokens, the other for data packets (the volume of signal packets is negligible). Similar to ExpressPass, 5 percent / 95 percent bandwidth are reserved for tokens / data respectively. Once every packet transmission time, a highest priority token is selected from all queuing tokens and paced to the link.

Because not all priorities are used at the same time, if we split the bandwidth to each token priority separately, the returning data packet will not use up 95 percent bandwidth. This will waste the bandwidth seriously. So credit of different priority should share the total 5 percent bandwidth.

R2: Discard expired tokens. In order to bound sojourn time, expired tokens must be dropped. Otherwise, in the worst case, the end-to-end latency of a packet can be arbitrarily large, thus many data packets can return a switch at the same time, which makes queuing delay unbounded (see Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2019.2958899). As a result, data packet loss is a much severe event than token packet loss.

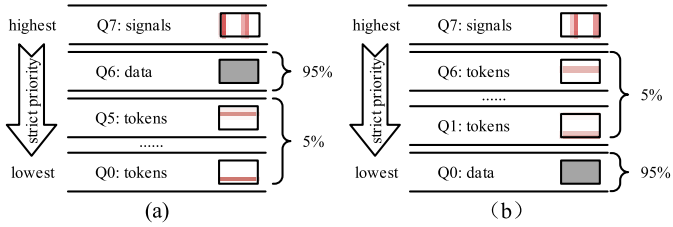


Fig. 3. Two work modes of a switch.

We set the sojourn time threshold to be a fixed number (e.g., 8) of packet transmission time. As a hypothetical switch, a queuing token is automatically dropped if its queuing time exceeds the threshold. Intuitively, this model meets all expected features for TOP listed in Table 1.

Transport Protocol. TOP uses similar token feedback control as that in ExpressPass. The target token loss rate is also 10 percent for each flow.

3.2 Methodology

We evaluate TOP by large-scale NS-3 simulations. The described methodology is used throughout the whole paper.

Topologies. We use dumbbell/leaf-spine topologies for single-path/multi-path evaluations. The dumbbell topology has 2 rack switches connected to each other, and each is connected with 40 servers. The leaf-spine topology has 8 rack switches connected by 4 core switches, and each is connected with 16 servers. For both topologies, each server is connected via a 10 Gbps link, and the default oversubscription ratio is 4:1. We use the single-path topology as the default topology. And the non-blocking dumbell topology with no oversubscription is also used to conduct simulations.

Workloads. For FCT-oriented and deadline-oriented objectives, we use the following production traces to evaluate our algorithms: web search, data mining and IMC10. Their cumulative distribution functions and descriptions can be found in [43]. We generate flows from these traces using Poisson process with load r similar to previous works [31], [43]. We set $r = 0.6$ by default and evaluate our algorithms with $0.3 \le r \le 0.8$. For coflow-oriented objectives, we use Fackbook logs [38] as our workloads, which has been widely accepted as a benchmark for coflow scheduling [38], [40], [51]. We scale down the size and arrival time of coflows to match our topologies when necessary.

Metrics. For FCT-oriented objectives, we choose average/tail FCT and average slowdown [31], [43] as main metrics. Slowdown is defined as the ratio of observed FCT to optimal FCT (when there is only one flow in networks). Although both metrics are frequently used to evaluate the performance of scheduling algorithms, they are quite different. For heavy-tailed flow distribution, the average FCT is not sensitive to small flows. As a comparison, slowdown is not sensitive to large flows, and throughput loss cannot be revealed by slowdown. Besides them, the bottleneck throughput, the average queuing delay for data packets, the maximum data queue occupancy in switches, and the average data packet loss ratio are shown for certain evaluations. For deadline-oriented and coflow-oriented objectives, we choose deadline missing ratio and the average coflow completion time (CCT) as the metrics, respectively.

Flow Priority. Now we discuss how to assign priority to flows. Suppose we are minimizing average FCT. When there are K physical queues for tokens and a flow may contain at most p packets, we need to decide $K-1$ thresholds to partition the interval $[1; p]$ into K segments. Flows in each priority should have the same capacity in total. If a priority is too idle, this priority is likely to be wasted, because few flows benefit from this priority; if a priority is too busy, a large number of flows will share this priority, which can also lead to a bad result. So we calculate the queue threshold value so that their total size are equally split to queues.

Denote the flow size distribution in a datacenter traffic as $f(x); x > 0$. For a heavy-tailed distribution, we use an inverse proportional function, e.g., $\frac{1}{x \ln p}$ $(x \in [1; p])$, to approximate the probability density function, where $\ln p$ is the normalization factor. To guarantee balanced capacities among all physical queues, for the ith threshold, we have $\int_1^{t_i} \frac{dx}{x \ln p} = \frac{i}{K}$, which indicates that $t_i = p^{\frac{i}{K}}$. Thus given a flow with size x, its priority can be calculated as $\lfloor K \log_p x \rfloor$. Practically, we will use $\lfloor K \log_p (ax + b) \rfloor$ for a better performance, where a and b are adjustable parameters. For a light-tailed distribution, we may use an exponentially decreasing function to approximate it, and its thresholds can be obtained in a similar way.

Therefore, a large flow has a lower priority, while a short flow has a higher priority.

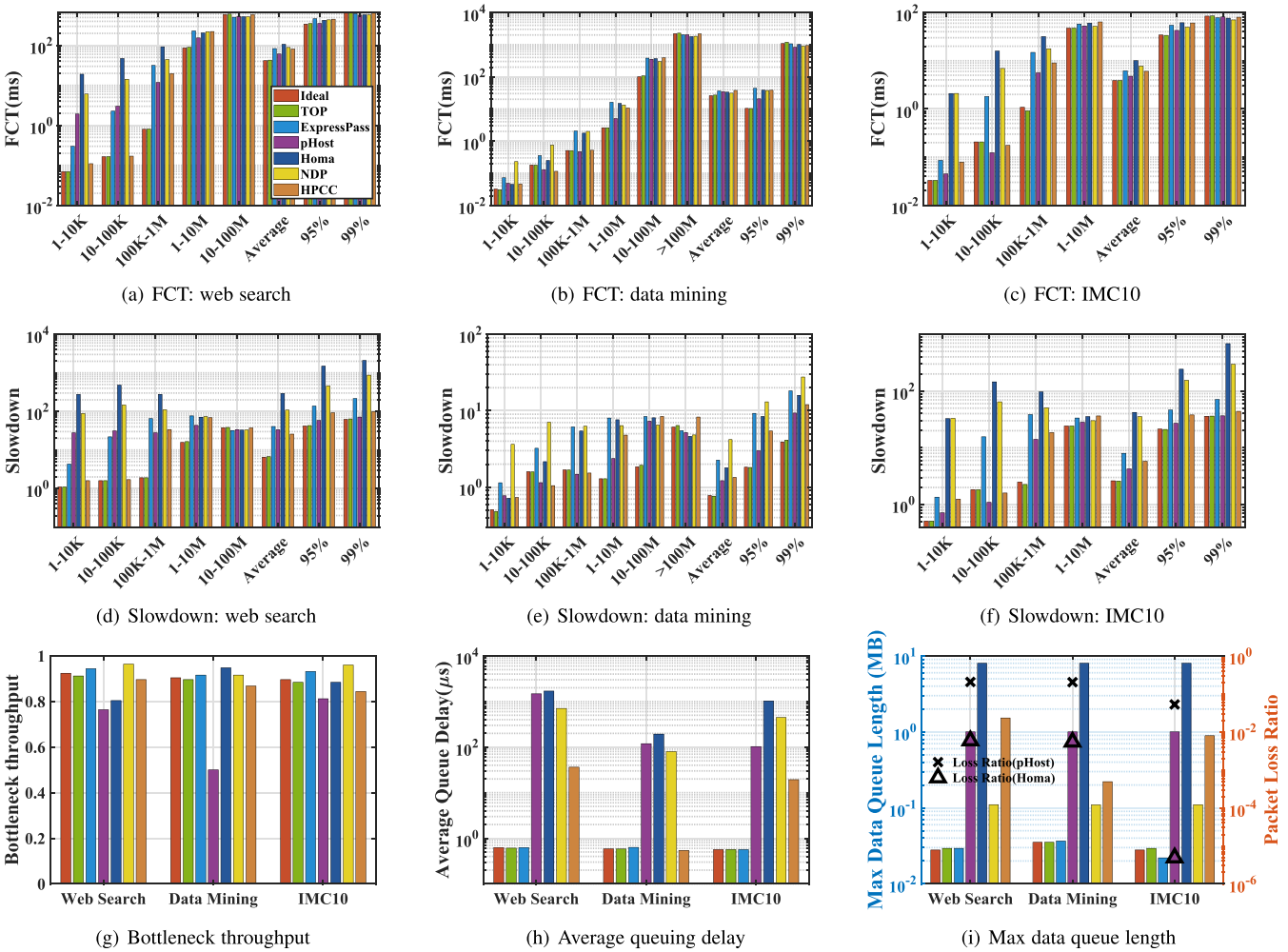
In non-clairvoyant scenarios where the size of each flow is not known until the connection is closed, a flow is assigned with a priority according to the number of received data packets. This approach is an approximation of the Least-Attained-Service (LAS) principal, which is widely used in non-clairvoyant scheduling. For deadline-oriented and coflow-oriented objectives, priority thresholds are chosen in similar ways as that of the FCT-oriented objective.

Compared with. The performance of an ideal switch with an infinite number of priorities are presented as the upper bound. Besides, we also implement ExpressPass, pHost, NDP and Homa in the simulations.

3.3 Comparison Results

For each experiment, 10,000 flows are generated using the Poisson process. Unless otherwise specified, we use default settings. Results are listed and analyzed as follows.

FCT. We first evaluate both ideal and TOP under default settings and topology, together with ExpressPass, pHost, NDP, Homa and HPCC for comparison. Results are partitioned into small intervals for the average and tail FCT and slowdown, as shown in Fig. 4. Note that in some figures, the y-axis is plotted in log scale. The performance of NDP and Homa are the worst. As analyzed in Section 2.2, the main reason of NDP's performance is that under severe congestion, an overwhelming number of control packets are generated. These packets overflow signal packet queues and cause control deadlocks. Homa suffers from blind packets' transmission with high priority. In fan-in networks (e.g., our default topology), it causes buffer overflow, and in turn causes retransmission. Note that Homa performs better for small flows, benefiting from blind packets transmission in the first RTT round. Shown in Fig. 4g, Homa also suffers from throughput problems. This is because packet loss occurs in Homa, as shown in Fig. 4i.



(a) FCT: web search (b) FCT: data mining (c) FCT: IMC10

(d) Slowdown: web search (e) Slowdown: data mining (f) Slowdown: IMC10

(g) Bottleneck throughput (h) Average queuing delay (i) Max data queue length

Fig. 4. Overall performance comparison.

Generally speaking, TOP performs much better than ExpressPass and pHost. Compared with ExpressPass, TOP has a smaller FCT and slowdown for most flows. Clearly, the goal of priority scheduling is effectively achieved with only 6 priority levels. However, for some large flows in the last interval, the FCT of TOP is slightly larger (shown in Figs. 4a and 4b). The tail 95 percent latency of TOP is smaller than ExpressPass and pHost. However, the tail 99 percent latency is slightly larger. These are caused by the conflict between priority scheduling and high throughput [52]. Shown in Fig. 4g, there exists throughput gap. Compared with pHost, TOP has better performance for medium and large flows. For small flows pHost is better, as it is in Homa. Note that pHost uses free tokens to transmit data packets in the first RTT before hand-shaking, thus the performance of small flows can be significantly improved. For example, in data mining workload, about 75 percent of the flows has less than 10 KB data to transmit, which leads to an extremely small slowdown,[1] as shown in Fig. 4e.

Shown in Fig. 4g, pHost suffers from throughput problems. pHost has no in-network limits for tokens. A large percentage of bandwidth can be occupied by tokens, instead of data packets. For data packets, $\mathcal{O}(100\ \mathrm{ms})$ average queuing delay (Fig. 4h), unbounded buffer growth and packet losses (Fig. 4i) are observed. On the contrary, TOP suffers none of these problems. It is a good approximation of the ideal switch. The FCT/slowdown differences between them are only 3.54 percent/9.7 percent on average.

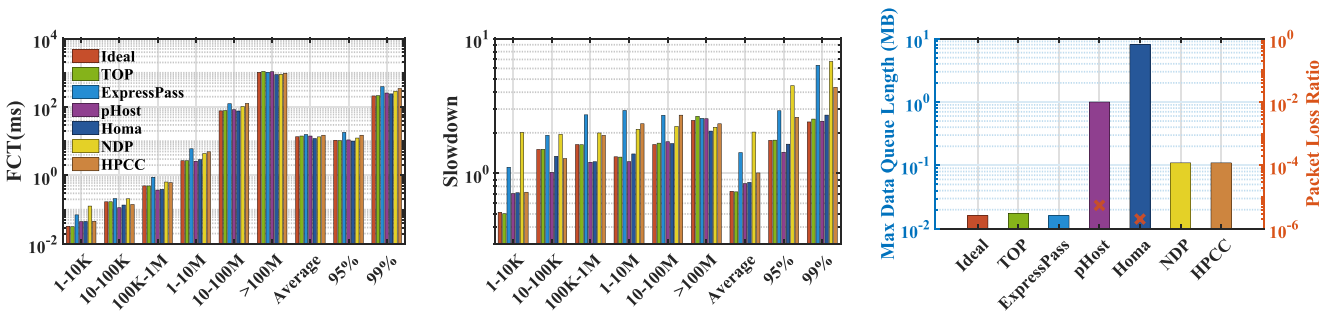
We also compare TOP with HPCC, a state-of-art proactive congestion control protocol. TOP also has a smaller FCT and slowdown for most flows. In HPCC, the meta-data carried by the ACK takes no effect on small flows, especially which size is smaller than one RTT. And the overreaction of the micro burst of small flows hurts the bandwidth utilization, as shown in Fig. 4g. The throughput of HPCC is slightly smaller than TOP.

To sum up, compared with ExpressPass, pHost, NDP, Homa and HPCC, the average FCT is reduced by 44.24, 29.39, 37.62, 59.8 and 47.1 percent respectively. The average slowdown is reduced by 81.40, 49.28, 81.01, 86 and 74.03 percent respectively.

In our evaluations, the maximum occupied queue length is quite small for TOP. It is only slightly larger than that of ExpressPass, as is shown in Fig. 4i. These results suggest that the theoretical bound (see Appendix A, available online.

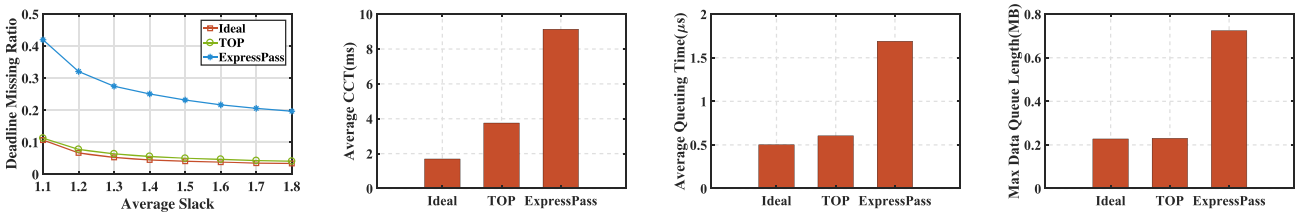
For generality, simulations under a non-blocking network are also conducted. The results are shown in Fig. 5. The

1. Slowdown can be less than 1 if all data packets are successfully received before hand-shaking finished.



(a) FCT: data mining (b) Slowdown: data mining (c) Queue length: data mining

Fig. 5. Non-blocking results.



(a) Deadline missing ratio (flow) (b) The average CCT (coflow) (c) The avg. queuing delay (coflow) (d) Max buffer size (coflow)

Fig. 6. Two objectives: deadline and coflow.

average and tail FCT of TOP are comparable with other approaches. The FCT of small flows with size between 1 and 10 K is largely reduced by TOP. The average slowdown of TOP are greatly improved. This benefits from the in-network prioritization. Queue length is shown in Fig. 5c. It indicates that pHost and Homa suffer from a long queueing delay and packet loss.

Deadline & Coflow. Finally, we demonstrate the results of other scheduling objectives: deadline-aware scheduling and coflow-aware scheduling. We choose ExpressPass as the baseline, and results are shown in Fig. 6. For deadline-aware scheduling, the deadline missing ratio is reduced by 79.18 percent, calculated as $\frac{0:197-0:041}{0:197}$.

Coflow scheduling is much challenging for transport protocols, because hundreds or thousands of flows may start almost at the same time when a large coflow arrives. Compared with the baseline, the average CCT is reduced by 58.88 percent, while the queuing delay and maximum data queue occupancy is only about $\frac{1}{3}$ of ExpressPass.

4 From Requirements to Reality

4.1 Investigating Requirements

Here comes a natural question: is TOP readily-deployable? We perform a comprehensive investigation of user manuals of main vendors' switches. We discuss the requirement discarding expired tokens waiting in queues (i.e., R2 in Section 3.1) first. To support it, each token should be attached with a timestamp metadata when the token enters a switch. The switch needs to periodically scan all tokens to remove expired ones, so that buffer is available to new incoming tokens. To our best knowledge, no existing switch supports such complex operation. Our ongoing prototyping on NetFPGA suggests that its cost is non-trivial.

The requirement rate-limiter for a group of queues (i.e., R1 in Section 3.1) is more complicated. The answer can be Yes, and No.

Yes for High-End Enterprise and Core Switches. There are several high-end switches do support this. Examples are Huawei CE6800 [53] and CE12800 [54] series. They support 802.1Qaz Enhanced Transmission Selection (ETS) [55]. With priority grouping feature, several queues in a port, instead of a single queue or the whole port, can be regarded as a group, and a single rate-limiter can be applied on such group. Thus we could treat all token queues as a group and rate-limit it.

The main problem is that such switches are not cost-effective. They are not designed for datacenters and mostly target enterprise and telecom scenarios. They implement abundant other features hence they are usually very expensive. The quotation price of a 48-port 10 Gbps CE6855 switch is tens of thousands dollars, excluding optical transceivers. As a comparison, a mainstream 32-port 100 Gbps datacenter switches (e.g., Arista 7060X) has a quotation of only half of them.

No for Mainstream Datacenter Switches. We analyze capabilities of widely-deployed datacenter switches by digging into their manuals (e.g., Arista 7060X [56] and Mellanox SN2700 [57]). They claim to support ETS. However, their rate-limiters can be only applied to a queue or the whole port, instead of several queues (of a port) together. We verified this with real Arista and Mellanox switches.

We have tried several walk-around ideas to rate-limit all tokens to 5 percent of link capacity, while none of them works. Here we show some bad examples.

Shown in Fig. 3a, we may let data packets occupy the second highest priority queue, e.g., Q6. If the data load is not high enough to use up all 95 percent of link capacity, the available bandwidth to tokens can be larger than 5 percent. This is unacceptable since reverse flows can be larger than 95 percent of link capacity, which in turn renders data loss. It is the same if we let data packets occupying the lowest priority queue Q0, shown in Fig. 3b. Residing in higher priorities, tokens can occupy bandwidth larger than 5 percent unless we can limit the capacity of all 6 queues.

One option is to rate-limit each token queue such that their capacities sum up to 5 percent. For example, we may rate-limit each of 6 token queues to $\frac{5\%}{6} \approx 0{:}83\%$ of the link capacity. However, it's a fact that at most time, some token queues can be idle (i.e., have no token to transmit) due to imbalanced traffic pattern. We cannot automatically adjust hardware configurations timely to match the fast-changing flows, thus the actual bandwidth used by tokens can be less than 5 percent, and the throughput of returning data packets is hurt.

One may think of combining egress prioritization and ingress rate-limiter together. Imaging tokens with mixed priorities arrive at an ingress port with a large rate. The ingress rate-limiter would drop token packets randomly, regardless of their priorities, hence compromise prioritization.

Overhead of Adding Group Rate-Limiters. From the point of hardware cost, the overhead of implementing a group rate-limiter is trivial. Our ongoing prototyping on NetFPGA suggests that several counters can meet the requirements. We have a private discussion with an architect from a market-leading switch chip vendor. The comment is that: the company's obsolete generations of chips actually have this function of group rate-limiters. However, during their life-cycles there exist almost no application scenarios for this feature. Let's cite his original words: "...if you show me how to use this feature, I am happy to bring it back."

We are currently trying to persuade switch chip vendors to add this feature. As a temporary remedy solution, we also propose a readily-deployable approach to achieve in-network prioritization by pushing existing datacenter switch and end-host hardware capacity to an extreme end.

4.2 Rate-Limiting All Token Queues

The key intuition is to separate the priority grouping feature in ETS into two steps: grouping and prioritizing. Forwarding rules are ingeniously configured to make each token enter a switch twice thus the two steps can be done one by one. We use Fig. 7 as an example to illustrate an innovative design which combines per-port rate-limiter and breakout cables [58] together.

With breakout cables, a single physical Quad-lane Small Form-factor Pluggable (QSFP) port can be split into four single-lane sub-ports and each sub-port now functions as an independent port. In this example, we use a 40 Gbps leaf switch for rate-limiting and prioritization, and split port 2 into four sub-ports (i.e., port 2/1, 2/2, 2/3, 2/4), each with 10 Gbps bandwidth. An additional low-end 10 Gbps switch (assistant switch) is used to route tokens back. A 40 Gbps port in leaf switch is connected to four 10 Gbps ports in assistant switch via a breakout cable.

TOP uses different VLAN IDs for data packets and token packets, thus even from the same ingress port, the two kinds of packets can be forwarded to different egress ports. Assuming some packets enter the leaf switch at port 1 and finally leave at port 3. Simply speaking, data packets will be forwarded to port 3 directly via VLAN A, while token packets will be forwarded to port 2/2 via VLAN B and finally forwarded to port 3 via VLAN C. The trace of token packets are marked as (1)-(4) in Fig. 7.

Rate-Limiting Configuration. In this example, we integrate one 40 Gbps physical port and two logical sub-ports (i.e., port 3 and sub-port 2/1, 2/2 in leaf switch) together to emulate the function of a single TOP virtual port. In the leaf switch, all tokens enter port 1 are forwarded to a sub-port 2/2 first. Exploiting the per-port rate-limiting capability, TOP limits outbound traffic of port 2/2 to 2 Gbps (i.e., 5 percent of 40 Gbps physical port capacity). In port 2/2, tokens are mapped to 6 priority queues and compete for the outbound bandwidth. Then the survived token packets enter port a1 in assistant switch. Following the forwarding table in assistant switch, the token packets are then forwarded to port a1′. Further, all token packets go back to sub-port 2/1 in leaf switch via the break-out cable again. Finally, the leaf switch forwards the tokens (distinguished by VLAN) to port 3. In port 3, all token packets are mapped to a single queue directly since all tokens are already prioritized and rate-limited by port 2/2. Again, outbound data packets enters one queue using the remaining 95 percent of link capacity.

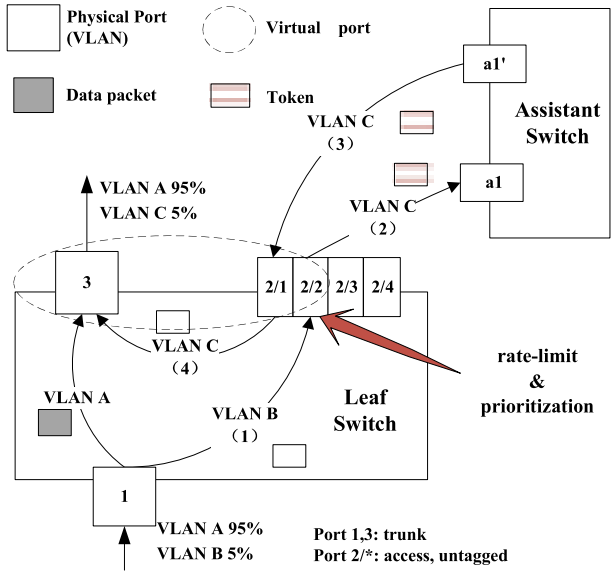


Fig. 7. Using commodity switch to emulate TOP port.

To sum up, we trade-off the number of ports for TOP token rate-limiting and prioritization. For a 40/100 Gbp switch, each virtual port in TOP consums 1+1/2 physical ports in leaf or spine switches and 2 ports in the assistant 10 Gbps switch.

Forwarding Configuration. A switch has a forwarding table in the form of $(\mathsf{VLAN}; \mathsf{destMAC}) \rightarrow \mathsf{Port}$. Packets with the same VLAN ID and destination MAC address are forwarded to the same port. For our example, a token should be forwarded twice in leaf switch: $\mathsf{port}\ 1 \rightarrow \mathsf{port}\ 2{=}2$ and $\mathsf{port}\ 2{=}1 \rightarrow \mathsf{port}\ 3$.

TOP uses VLANs globally to distinguish data and token packets. Three VLANs are configured globally in all leaf and spine switches. VLAN A is always for data packets, VLAN B and C are always for token packets. We use two VLAN IDs for token packets to support multi-hop networks. From bottom up, we use $(\mathsf{B}; \mathsf{MAC}) \rightarrow \mathsf{subport}\ 2{=}*$ and $(\mathsf{C}; \mathsf{MAC}) \rightarrow \mathsf{port}\ 3$ for switches in odd-numbered levels (e.g., leaf level). While in even-numbered levels (e.g., spine level), We use $(\mathsf{C}; \mathsf{MAC}) \rightarrow \mathsf{subport}\ 2{=}*$ and $(\mathsf{B}; \mathsf{MAC}) \rightarrow \mathsf{port}\ 3$ for switches.

In addition, to forward tokens correctly in the assistant switch, we need additional VLAN IDs. We need to configure the VLAN mode of sub-ports to VLAN access mode

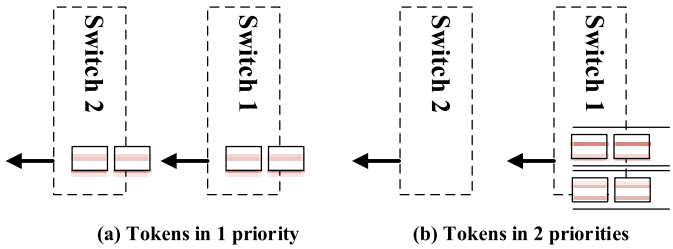


Fig. 8. Bound token delay.

and other ports to VLAN trunk mode. Note that hosts can process data / token packets regardless of their VLAN IDs.

For each MAC destination, 3 forwarding entries are statically configured. A typical switch such as Mellanox SN2700 has 256 K L2 entries. It can support over 85 K servers. For a 48 ports assistant switch, it need to consume 48 / 2 = 24 VLAN IDs in total. We can reuse these 24 VLAN IDs if we have several assistant switches. To sum up, TOP needs 27 VLAN IDs in total: 3 for leaf and spine switches and 24 for assistant switches. For example, if we have a 32 ports 40 Gbps switch, we can use it to emulate about 32*2 / 3 = 21 virtual TOP ports. We need to consume extra 42 ports in an assistant switch and 3 + 21 = 24 VLAN IDs. Notice that we just need a very basic packet relay function for the assistant switch, thus we can even replace the assistant switch with a SFP media converter to do such reroute, which usually costs only $20 per port.

In conclusion, TOP uses different VLAN IDs to rate-limit all token queues in mainstream datacenter switches. The overhead of it will be discussed later in Section 4.4.

4.3 Bounding Per-Token Delay

The key idea is to move the responsibility of bounding token delay from switches to end-hosts. Our observation is that precise time synchronization is achievable in datacenter networks. The Precision Time Protocol (PTP) is an IEEE protocol used to synchronize clocks throughout a network [59]. On a local datacenter network, PTP can achieve clock accuracy in the sub-microsecond range. Whenever a token reaches a sender, the sender can calculate its sojourn time along the whole path. If a token has been blocked for an overly long time, the sender can drop this obsolete token to bound valid token latency. We follow the token queue length setting in ExpressPass, i.e., 8 tokens per queue. TOP can set the threshold of maximum path sojourn time as $\mathsf{Number\_Of\_Hops} \times 8$ token pacing time. Thus, TOP tokens observe exactly the same maximum latency as that of ExpressPass.

Note that due to the existence of multiple priority queues, maximum per-hop token sojourn time can be as large as the path sojourn time. A two-hop example is shown in Fig. 8a. There are only tokens of 1 priority in the network, and queue length per-hop is set to 2 tokens. The maximum token delay is 4 units (1 unit is one token's pacing inteval). In the case of Fig. 8b, there are tokens of two priorities. A low priority token can have 4 units delay and is valid. It has been blocked in switch 1 by tokens in both the higher queue and the same queue. In this case, the observed data RTT difference in switch 1 can be doubled. This affects the calculation of maximum data queue occupancy for zero data loss. We leave detailed analysis and discussion to appendix (Section A), available in the online supplemental material.

4.4 Putting Together

Now we put the above two techniques together and complete the whole picture of the readily-deployable solution. We assume a flow's information such as size and deadline are known by the sender. With TOP, a flow is treated as a sequence of full-length data packets (except the last one). Each data packet is assigned a sequence ID. The transport protocol works as follows:

- When a new flow arrives, the sender sends a SYNC signal packet with flow information to the receiver.
- In each pacing slot, the receiver chooses a flow's token, attaches it with the corresponding priority, timestamps it, and paces it out to the link.
- The sender receives the token and checks the timestamp.
- For every token that has a sojourn time larger than the threshold, drops it.
- For every token that is not timed out, a data packet with ID from the corresponding flow is sent back.
- Receiver checks ID of the received data packet, closes the connection if all data received.

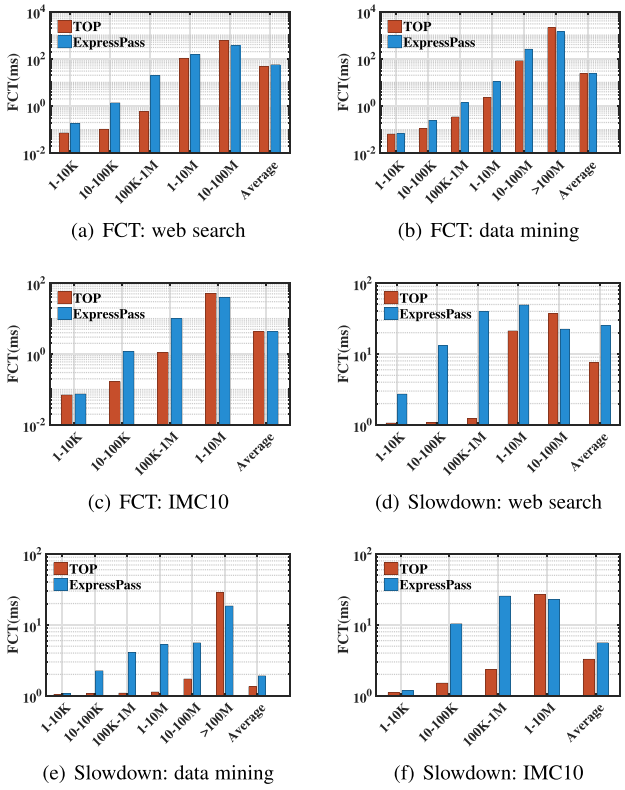
There are other design details.

Packet Loss. Normally, zero data packet loss is guaranteed by our design. In quite occasional cases such as a switch failure, a packet loss can still occur. When the receiver realizes that a packet is lost (if received packets have much larger IDs), the missing ID is sent to the sender by a dedicated signal packet. The sender then retransmits this packet as soon as it receives a token from this flow. Note that the sender only retransmits a specific packet at most once in each RTT.

Routing and Reorder. In multi-path leaf-spine / Clos networks, we use symmetric ECMP by default to ensure tokens and data packets of a flow follow exactly reverse paths. If per-packet multi-path load-balancing is supported (e.g., Drill [60]), hot-spots can be mostly eliminated. With flow size information, handling flow packet-reorder is trivial. Under this circumstance, switch proactive congestion control is pacing tokens for all switches in the same tier of its aggregation / spine block [20]. Our evaluation suggests that per-packet multi-path does improve performance, at the same time does not noticeably increase maximum queue occupancy (Section 5).

Low-Priority Unscheduled Packets (Optional). The free tokens mechanism in pHost, NDP and Homa can save one RTT and significantly reduce the FCT for tiny flows when network load is low. However, it also causes the incast problem at receivers (in pHost and Homa) or hurts scheduled packets (in NDP). The lesson we learnt is that: unscheduled packets and scheduled packets should not equally compete. We propose a novel unscheduled packets approach to solve the problem at the cost of reducing a physical queue for token packets.

In Fig. 3, we can reserve the lowest priority queue Q0 for unscheduled packets, and Q1 for scheduled packets. Now tokens can only use Q2-Q6. We limit the queue length of Q0 to exactly 1 full-length packet. With 1 packet queue length, TOP enables unscheduled packets to be forwarded. While they cannot be queued unless their sizes are less than full-length, which are very rare cases. Now all of them are transmitted opportunistically. Easy to see that the end-to-end data delay is still bounded. The SYNC signal should inform



(a) FCT: web search (b) FCT: data mining

(c) FCT: IMC10 (d) Slowdown: web search

(e) Slowdown: data mining (f) Slowdown: IMC10

Fig. 9. Remedy TOP solution versus extra bandwidth.

the receiver about the existence of unscheduled packets in a flow. Not all flows are allowed to send unscheduled packets. Unscheduled packets are designed for latency-sensitive short flows to save the first RTT, not for large flows. Only short flows with no more than m packets can send at most m unscheduled packets, where m is an adjustable parameter (e.g., Bandwidth-Delay-Production is around 6 packets in a 40 Gbps network).

Low-priority unscheduled packets are optional in our design. Only when latency-sensitive flows compose the majority, should we consider trading a physical queue for unscheduled packets. Otherwise, reducing token queues from 6 to 5 may hurt the performance, since smaller number of queues shrink the gap among different flows. Also, unscheduled packets compromise the zero data loss goal. Certain application scenarios such as distributed storage may not prefer this option.

Non-Clairvoyant Cases. To prioritize non-clairvoyant flows, the protocol workflow should be revised. When a flow ends, the sender should send a FIN signal packet with the final flow size. This FIN signal usually arrives before the last data packet. The receiver should wait until all data are received before close the connection. Note that unscheduled packets should not be used in non-clairvoyant scenarios, since TOP cannot judge whether a flow is small or large.

Overhead of Remedy. As mentioned above, our remedy solution integrates 1 + 1/2 physical ports to a virtual TOP port. One may concern that trading physical ports for token priority wastes throughput, and such waste may be worthless. We conduct an extra simulation. For ExpressPass, we add 50 percent extra bandwidth by setting the capacity of server to 15 Gbps and the capacity between switches to 150 Gbps. Results are shown in Fig. 9. Because of our priority based design, small flows' FCT of TOP is better than that of ExpressPass, and the average FCT of TOP is comparable to that of ExpressPass. Large flows' (> 10M) FCT is a little larger than that of ExpressPass because large flows benefit from extra bandwidth. The average slowdown of TOP is also better than that of ExpressPass.

To sum up, the remedy solution brings some benefits at some cost. It could be a reasonable trade-off in necessary scenarios to benefit small flows.

5 TOP Prototyping and Deep Dive

5.1 Implementation

Testbeds. We build two small-scale testbeds for evaluating TOP versus pHost, ExpressPass and our ideal approach. We haven't implemented NDP and Homa yet. Instead they are evaluated in our NS-3 simulations. All testbeds use the same dumbbell topology shown in Fig. 10. All hosts are Dell PowerEdge R730 with two 8-core Intel E5-2630 2.40 GHz CPU, 128G memory, a 4 TB hard disk. Each server has a 4-port Intel X710 10 Gbps NIC and a dual-port CX-5 100 Gbps NIC. Server OS is Ubuntu 14.04.3 LTS. For our hypothetic and ideal approaches in testbed 1, we use two servers to emulate 4-port token-preemptive switches. For all other approaches in testbed 2, we use Mellanox SN2700 and a $400 assistant switch.

Sender/Receiver Prototype. We implement ExpressPass, pHost and TOP as three independent modules inside SoftNIC. All senders / receivers use the CX-5 NIC. SoftNIC is a framework based on DPDK technology which can bypass the kernel and achieve high performance. For all experiments, we use 10 Gbps mode. This is the maximum bandwidth our end-host prototype can support.

Hypothetic/Ideal Switch Prototypes. SoftNIC is also used to emulate 4-port hypothetic / ideal switch. The hypothetic / ideal switch has functions such as finding a token with the highest priority, drop timeout tokens, etc. To achieve high performance, we use the rte_llring queue in DPDK as the basic queue type. We pin each SoftNIC worker thread to a distinct CPU core, which processes one port each. Using Tcpdump tool to capture the packets, we verify that the software switch can schedule packets with microsecond level accuracy in 10 Gbps mode.

TOP Switch Configuration. We enable the bandwidth shaping function on corresponding ethernet ports on Mellanox switches. The CX-5 NIC of each server enables PTP support by linuxptp. One server is chosen as the PTP master clock and switches forward PTP messages among servers. The measured synchronization error is around 100 ns. Mellanox SN2700 has 32 ports. We can use them to emulate about 21 virtual ports to support TOP operations.

The topology shown in Fig. 10 has a baseline one-way delay of about 10 μs. The maximum token queue length is set as 6 packets, and the maximum per-hop queuing delay can be calculated as $\frac{6\times 84\text{Byte}}{10\text{Gbps}\times 5\%} \approx 8\ \mu\text{s}$. Therefore, the token timeout threshold $T = 10\ \mu\text{s} + 2 \times 8\ \mu\text{s} = 26\ \mu\text{s}$.

We summarize our experimental results as answers to the following questions.

- How does TOP perform in real testbeds? The performance of TOP is close to the ideal approach. The average FCT

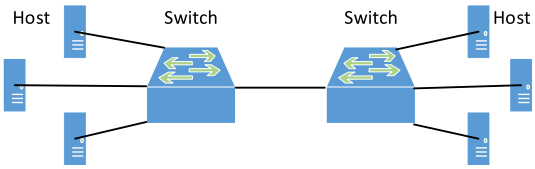


Fig. 10. Dumbbell topology of testbeds.

is reduced by 35.58 and 48.96 percent, and the average slowdown is reduced by 83.35 and 91.56 percent, compared with ExpressPass and pHost, respectively. No data packet loss is observed. In addition, we find that it is hard to implement pHost with real system constraints.

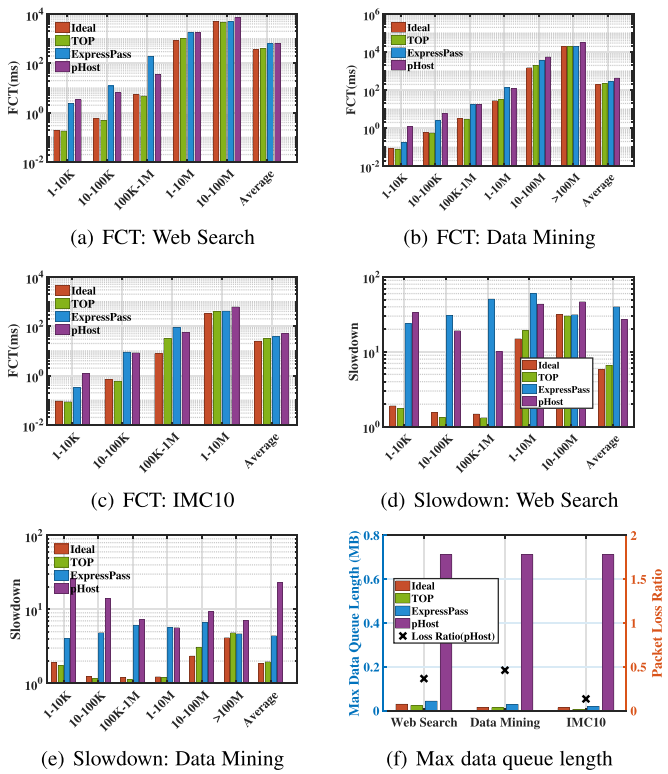
- Does TOP perform well in different scenarios?Yes. We evaluate TOP with large-scale simulations over all kinds of scenarios, including different workloads, network loads, topologies and fan-in factors. We also evaluate TOP in non-clairvoyant scenarios. The performance of TOP is close to ideal in most cases.

Besides, we also explore the impact of parameter settings.

5.2 Testbed Results

We use two small-scale testbeds to evaluate our algorithms. Unscheduled packets are disabled thus 6 physical queues are used for token packets. For each experiment, 1,000 flows are generated using Poisson process with load of 0.6. Results are shown in Fig. 11.

The performance of our TOP design is quite close with Ideal in most cases. On average, the differences of FCT / slowdown between two designs are only 14.79 percent / 13.66 percent, respectively. Besides, compared with ExpressPass and pHost, the average FCT is reduced by 35.58 and 48.96 percent, and the average slowdown is reduced by 83.35 and 91.56 percent, respectively. Meanwhile, we observed that the maximum data queue occupancy of all evaluated algorithms, except pHost, is quite small, thus zero data packet loss is guaranteed.

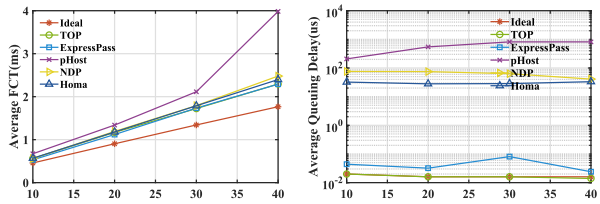


(a) FCT: Web Search (b) FCT: Data Mining

(c) FCT: IMC10 (d) Slowdown: Web Search

(e) Slowdown: Data Mining (f) Max data queue length

Fig. 11. Testbed results.



(a) FCT (b) Average queuing delay

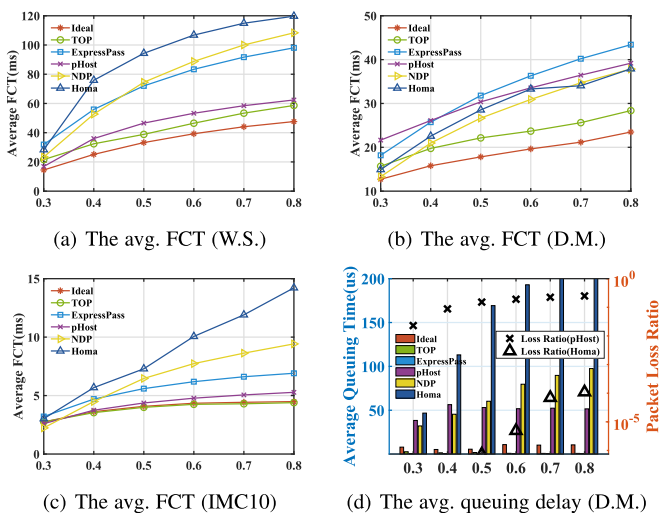
Fig. 12. The incast performance.

Note that the performance of pHost in our implementation is not as good as it in simulations. In pHost, a sender can only use unexpired tokens. It is hard to implement pHost with real system constraints. First of all, Linux has CPU interrupts. Tokens are accumulated in NICs during interrupts. After an interrupt, a batch of tokens arrived together while most of them are dropped by the sender due to timeout of 1.5× MTU transmission time. This leads to a significant throughput loss. Second, tokens can occupy the switch buffers and consumes most of the bandwidth due to the lack of in-network rate-limiting. Again, the bandwidth available to data packets is hurt.

5.3 Miscellaneous Scenarios

The Incast Problem. Due to uncontrolled free tokens in the first RTT, pHost and Homa still has the incast problem. We use a simple topology, where n servers directly connects to a single big switch, to evaluate the incast performance. The first $n-1$ servers start at the same time to send flows to the last server and each sender host sends exactly one flow with 50 packets. Results are shown in Fig. 12, which indicates that pHost suffers a serious incast problem as the number of concurrent flows growing, while other algorithms suffers no obvious incast problem.

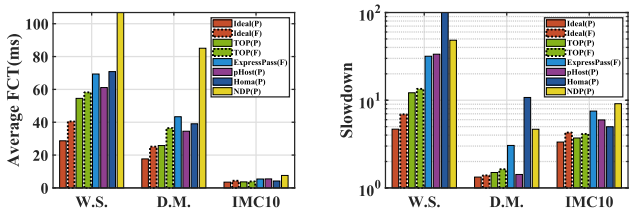
Traffic Load. We evaluate our algorithms under various network loads from 0.3 to 0.8. TOP performs well under both light and heavy load. Results are shown in Fig. 13.



(a) The avg. FCT (W.S.) (b) The avg. FCT (D.M.)

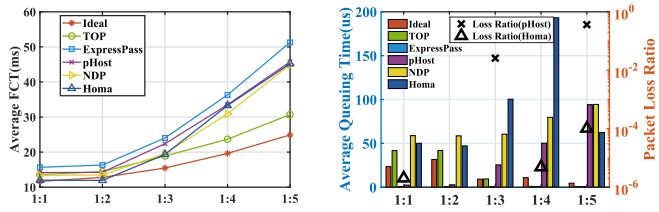
(c) The avg. FCT (IMC10) (d) The avg. queuing delay (D.M.)

Fig. 13. The impacts of traffic loads.



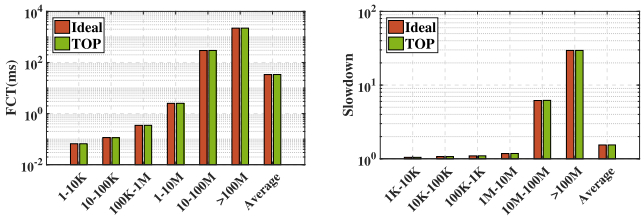
(a) The average FCT (b) The average slowdown

Fig. 14. Multi-path scenarios.



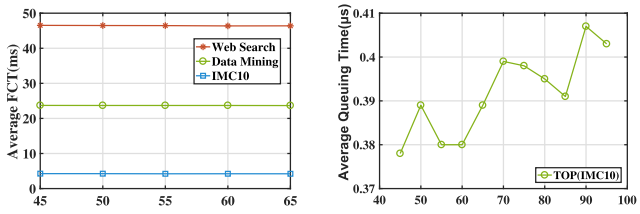
(a) The average FCT (b) The average queuing delay

Fig. 15. The impact of fan-in factors.



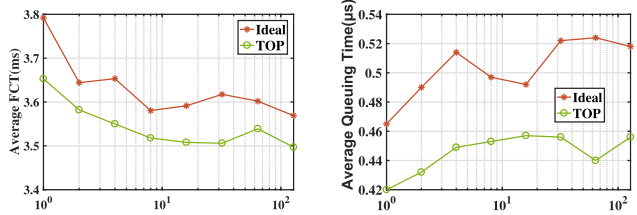
(a) FCT (b) Slowdown

Fig. 16. Non-clairvoyant scheduling.



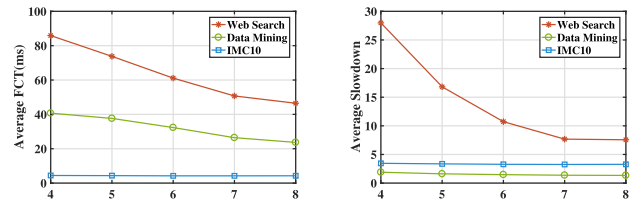
(a) The average FCT (b) The average queuing delay

Fig. 17. The impacts of token timeout threshold.



(a) The average FCT (b) The average queuing delay

Fig. 18. The impacts of unscheduled packets.



(a) The average FCT (b) The average slowdown

Fig. 19. The impacts of # physical queues.

Note that for some light workloads, pHost performs slightly better than TOP, e.g., 0.3, as shown in Fig. 13a. This is the benefit of free tokens. Under light loads, the network is not so busy, thus data packets sent in the first RTT have a high probability of being received successfully. Besides, the incast problem is not likely to occur and these packets are not likely to disturb the transmission of scheduled data packets. However, when load $r \geq 0.4$, TOP performs better than all other algorithms for all workloads.

To sum up, the average FCT is reduced by 46.07 percent compared with ExpressPass, and by 29.72 percent compared with pHost. The average queuing time for data packets is reduced by 98.42 percent compared with pHost, and zero data packet loss is guaranteed.

Multi-Path Topologies. To verify the performance in a multi-path environment, we evaluate our algorithms with both symmetric per-flow routing and asymmetric per-packet routing. We use the suffix F and P to distinguish the two routing approaches in our results, respectively. Besides, we use symmetric per-flow routing for ExpressPass and per-packet routing for pHost, Homa and NDP, as clarified in their papers. Results are shown in Fig. 14.

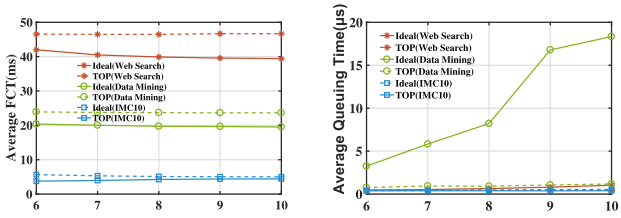
In all workloads, TOP has better performance in per-packet routing, because packets are load-balanced to each core switch pseudo-uniformly even in a small time scope. However, per-flow routing does not have this advantage. Among different workloads, the larger the average flow size is, the bad the per-flow routing performs. The difference of TOP's performance between two routing approaches are 6, 6 and 29 percent for IMC10, web search and data mining workloads, respectably. As expected, data mining workload has the largest flow size on average. To sum up, compared with ExpressPass, pHost, NDP and Homa, the average FCT is reduced by 40.47, 33.08, 56.77 and 33.9 percent, and the average slowdown is reduced by around 61.71, 63.70, 67.71 and 89.4 percent, respectively.

Fan-in Factors. We evaluate our algorithms in oversubscribed networks with fan-in factors from 1:1 to 5:1, and results are shown in Fig. 15. TOP performs well over in all cases, and the performance improvement increases as the fan-in factor increases. Clearly, priority scheduling plays an important role in highly-oversubscribed networks. Due to unlimited tokens, pHost and Homa become worse as the fan-in factor increases. With 5:1 fan-in factor, the average FCT is reduced by 40.12, 32.87, 31.71 and 32 percent, compared with ExpressPass, pHost, NDP and Homa, respectively.

Non-Clairvoyant. Non-clairvoyant scenarios are shown in Fig. 16. Too many priority levels lead to fair sharing for LAS [34], [40], thus we only use 8 priority levels for ideal case. Recalling the results in Fig. 4b and 4e, the performance of TOP in non-clairvoyant scenarios is still comparable even with pHost in clairvoyant scenarios, and is better than ExpressPass.

5.4 Algorithm Parameters

Token Timeout Threshold. Senders drop tokens if they timeout. The drop threshold T can be chosen as $d + |P|t_h$ (about



(a) The average FCT (b) The average queuing delay

Fig. 20. The impacts of token-queue length.

45 ms in evaluated topologies), as illustrated in Section A, available in the online supplemental material. In this part, we want to evaluate what will happen if clocks on hosts are not well-synchronized, i.e., the scenarios that synchronization precision $D > 0$. We scan the token timeout from 45 ms to 95 ms, which corresponds to a 0-50 ms synchronization precision. Shown in Fig. 17a, the drop threshold has no significant influence on the average FCT. As expected, shown in Fig. 17b, a larger threshold leads to a slightly larger queuing delay, which matches our theoretical results.

Unscheduled Packets. As analyzed before, only when small flows are majority of the whole workload should we consider trading a priority queue for unscheduled packets. Therefore, we only evaluate the impacts of unscheduled packets in data mining workloads. Results in Fig. 18 indicate that, only 4-8 unscheduled packets for small flows are enough, and we cannot benefit more with a larger number of unscheduled packets.

Physical Queues. Finally, we discuss the impact of physical queues, including the number of physical queues (Fig. 19) and the length of each token-queue (Fig. 20). The average FCT decreases as the number of physical queues increasing, benefiting from fine-grained priority assignments. As the length of each token-queue growing, the average FCT has a decreasing tendency, while the average queuing delay slightly increases in most cases. Due to bounded end-to-end delay, the impact of the token-queue length is negligible. However, it should be limited to prevent token packets from occupying too much switch buffer, especially for shared-buffer switches.

6 Conclusion

Token-oriented approaches are promising. They can provide lossless network service without requiring the Priority Flow Control feature. In this paper, we explore the gain and applicability of TOP, i.e., Token-Oriented in-network Prioritization, which provides in-network prioritization in a readily-deployable way. With a maximum queue occupancy comparable to ExpressPass, TOP can achieve superior performance across different optimization objectives and scenarios. We are currently in the process of supporting TOP in real switches.

Acknowledgments

The authors would like to thank anonymous reviewers for their valuable comments. This research is supported by the National Key R&D Program of China 2018YFB1003505, National Natural Science Foundation of China under Grant Numbers 61772265 and 61802172, Collaborative Innovation Center of Novel Software Technology and Industrialization, and Jiangsu Innovation and Entrepreneurship (Shuangchuang) Program.

References

[1] M. Zaharia et al., "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in Proc. 9th USENIX Conf. Netw. Syst. Design Implementation, 2012, p. 2.
[2] H. Zhang, G. Chen, B. C. Ooi, K.-L. Tan, and M. Zhang, "In-memory big data management and processing: A survey," IEEE Trans. Knowl. Data Eng., vol. 27, no. 7, pp. 1920–1948, Jul. 2015.
[3] C. DeSanti and J. Jiang, "FCoE in perspective," in Proc. Int. Conf. Adv. Infocomm Technol., 2008, Art. no. 138.
[4] D. Minturn and J. Metz, "Under the hood with NVMe over fabrics," in Ethernet Storage Forum, 2015. [Online]. Available: http://https://www.snia.org/sites/default/files/ESF/NVMe_Under_Hood_12_15_Final2.pdf
[5] Intel, "Data plane development kit," 2011. [Online]. Available: https://www.dpdk.org/
[6] C. Mitchell, Y. Geng, and J. Li, "Using one-sided RDMA reads to build a fast, CPU-efficient key-value store," in Proc. USENIX Annu. Tech. Conf., 2013, pp. 103–114.
[7] A. Dragojević, D. Narayanan, O. Hodson, and M. Castro, "FaRM: Fast remote memory," in Proc. 11th USENIX Conf. Netw. Syst. Design Implementation, 2014, pp. 401–414.
[8] A. Kalia, M. Kaminsky, and D. G. Andersen, "Using RDMA efficiently for key-value services," ACM SIGCOMM Comput. Commun. Rev., vol. 44, pp. 295–306, 2014.
[9] H. Li, A. Kadav, E. Kruus, and C. Ungureanu, "MALT: Distributed data-parallelism for existing ML applications," in Proc. 10th Eur. Conf. Comput. Syst., 2015, Art. no. 3.
[10] X. Wei, J. Shi, Y. Chen, R. Chen, and H. Chen, "Fast in-memory transaction processing using RDMA and HTM," in Proc. 25th Symp. Operating Syst. Princ., 2015, pp. 87–104.
[11] Y. Chen, X. Wei, J. Shi, R. Chen, and H. Chen, "Fast and general distributed transactions using RDMA and HTM," in Proc. 11th Eur. Conf. Comput. Syst., 2016, Art. no. 26.
[12] J. Shi, Y. Yao, R. Chen, H. Chen, and F. Li, "Fast and concurrent RDF queries with RDMA-based distributed graph exploration," in Proc. 12th USENIX Symp. Operating Syst. Design Implementation, 2016, pp. 317–332.
[13] D. Cohen et al., "Remote direct memory access over the converged enhanced ethernet fabric: Evaluating the options," in Proc. 17th IEEE Symp. High Perform. Interconnects, 2009, pp. 123–130.
[14] C. Guo et al., "RDMA over commodity ethernet at scale," in Proc. ACM SIGCOMM Conf., 2016, pp. 202–215.
[15] Y. Zhu et al., "Congestion control for large-scale RDMA deployments," ACM SIGCOMM Comput. Commun. Rev., vol. 45, pp. 523–536, 2015.
[16] M. Alizadeh et al., "Data center transport mechanisms: Congestion control theory and ieee standardization," in Proc. 46th Annu. Allerton Conf. Commun., Control, Comput., 2008, pp. 1270–1277.
[17] R. Mittal et al., "TIMELY: RTT-based congestion control for the datacenter," ACM SIGCOMM Comput. Commun. Rev., vol. 45, pp. 537–550, 2015.
[18] Y. Zhu, M. Ghobadi, V. Misra, and J. Padhye, "ECN or delay: Lessons learnt from analysis of DCQCN and TIMELY," in Proc. 12th Int. Conf. Emerg. Netw. Experiments Technol., 2016, pp. 313–327.
[19] N. Farrington and A. Andreyev, "Facebook data center network architecture," in Proc. IEEE Opt. Interconnects Conf., 2013, pp. 49–50.
[20] A. Singh et al., "Jupiter rising: A decade of Clos topologies and centralized control in Google's datacenter network," in Proc. ACM Conf. Special Interest Group Data Commun., 2015, pp. 183–197.
[21] M. Alizadeh et al., "Data center TCP (DCTCP)," in Proc. ACM SIGCOMM Conf., 2011, pp. 63–74.
[22] H. Wu, J. Ju, G. Lu, C. Guo, Y. Xiong, and Y. Zhang, "Tuning ECN for data center networks," in Proc. 8th Int. Conf. Emerg. Netw. Experiments Technol., 2012, pp. 25–36.
[23] C. Lee, C. Park, K. Jang, S. B. Moon, and D. Han, "Accurate latency-based congestion feedback for datacenters," in Proc. USENIX Annu. Tech. Conf., 2015, pp. 403–415.
[24] I. Cho, K. Jang, and D. Han, "Credit-scheduled delay-bounded congestion control for datacenters," in Proc. Conf. ACM Special Interest Group Data Commun., 2017, pp. 239–252.

[25] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, "Less is more: Trading a little bandwidth for ultra-low latency in the data center," in Proc. 9th USENIX Conf. Netw. Syst. Design Implementation, 2012, p. 19.
[26] K. Jang, J. Sherry, H. Ballani, and T. Moncaster, "Silo: Predictable message latency in the cloud," ACM SIGCOMM Comput. Commun. Rev., vol. 45, no. 4, pp. 435–448, 2015.
[27] M. P. Grosvenor et al., "Queues don't matter when you can JUMP them!" in Proc. 12th USENIX Conf. Netw. Syst. Design Implementation, 2015, pp. 1–14.
[28] C.-Y. Hong, M. Caesar, and P. Godfrey, "Finishing flows quickly with preemptive scheduling," in Proc. ACM SIGCOMM Conf. Appl. Technol. Archit. Protocols Comput. Commun., 2012, pp. 127–138.
[29] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz, "DeTail: Reducing the flow completion time tail in datacenter networks," in Proc. ACM SIGCOMM Conf. Appl. Technol. Archit. Protocols Comput. Commun., 2012, pp. 139–150.
[30] A. Munir et al., "Minimizing flow completion times in data centers," in Proc. IEEE INFOCOM, 2013, pp. 2157–2165.
[31] M. Alizadeh et al., "pFabric: Minimal near-optimal datacenter transport," in Proc. ACM SIGCOMM Conf. SIGCOMM, 2013, pp. 435–446.
[32] A. Munir, G. Baig, S. M. Irteza, I. A. Qazi, A. X. Liu, and F. R. Dogar, "Friends, not foes: Synthesizing existing transport strategies for data center networks," in Proc. ACM Conf. SIGCOMM, 2014, pp. 491–502.
[33] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, "Fastpass: A centralized zero-queue datacenter network," in Proc. ACM Conf. SIGCOMM, 2014, pp. 307–318.
[34] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang, "Information-agnostic flow scheduling for commodity data centers," in Proc. 12th USENIX Conf. Netw. Syst. Design Implementation, 2015, pp. 455–468.
[35] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," in Proc. ACM SIGCOMM Conf., 2011, pp. 50–61.
[36] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter TCP (D2TCP)," in Proc. ACM SIGCOMM Conf. Appl. Technol. Archit. Protocols Comput. Commun., 2012, pp. 115–126.
[37] H. Zhang et al., "Guaranteeing deadlines for inter-datacenter transfers," in Proc. ACM 10th Eur. Conf. Comput. Syst., 2015, Art. no. 20.
[38] M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with varys," in Proc. ACM Conf. SIGCOMM, 2014, pp. 443–454.
[39] F. R. Dogar, T. Karagiannis, H. Ballani, and A. Rowstron, "Decentralized task-aware scheduling for data center networks," ACM SIGCOMM Comput. Commun. Rev., vol. 44, pp. 431–442, 2014.
[40] M. Chowdhury and I. Stoica, "Efficient coflow scheduling without prior knowledge," ACM SIGCOMM Comput. Commun. Rev., vol. 45, pp. 393–406, 2015.
[41] M. Chowdhury and I. Stoica, "Coflow: A networking abstraction for cluster applications," in Proc. 11th ACM Workshop Hot Topics Netw., 2012, pp. 31–36.
[42] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing data transfers in computer clusters with orchestra," ACM SIGCOMM Comput. Commun. Rev., vol. 41, pp. 98–109, 2011.
[43] P. X. Gao, A. Narayan, G. Kumar, R. Agarwal, S. Ratnasamy, and S. Shenker, "pHost: Distributed near-optimal datacenter transport over commodity network fabric," in Proc. 11th ACM Conf. Emerg. Netw. Experiments Technol., 2015, Art. no. 1.
[44] M. Handley et al., "Re-architecting datacenter networks and stacks for low latency and high performance," in Proc. Conf. ACM Special Interest Group Data Commun., 2017, pp. 29–42.
[45] J. Zhang, F. Ren, R. Shu, and P. Cheng, "TFC: Token flow control in data center networks," in Proc. 11th Eur. Conf. Comput. Syst., 2016, Art. no. 23.
[46] B. Montazeri, Y. Li, M. Alizadeh, and J. Ousterhout, "Homa: A receiver-driven low-latency transport protocol using network priorities," in Proc. Conf. ACM Special Interest Group Data Commun., 2018, pp. 221–235.
[47] P. Cheng, F. Ren, R. Shu, and C. Lin, "Catch the whole lot in an action: Rapid precise packet loss notification in data center," in Proc. 11th USENIX Conf. Netw. Syst. Design Implementation, 2014, pp. 17–28.
[48] F. Liu, J. Guo, X. Huang, and J. C. S. Lui, "eBA: Efficient bandwidth guarantee under traffic variability in datacenters," IEEE/ACM Trans. Netw., vol. 25, no. 1, pp. 506–519, Feb. 2017.
[49] P. Jin et al., "PostMan: Rapidly mitigating bursty traffic by offloading packet processing," in Proc. USENIX Annu. Tech. Conf., 2019, pp. 849–862.
[50] Y. Niu, F. Liu, X. Fei, and B. Q. Li, "Handling flash deals with soft guarantee in hybrid cloud," in Proc. IEEE Conf. Comput. Commun., 2017, pp. 1–9.
[51] Y. Li et al., "Efficient online coflow routing and scheduling," in Proc. 17th ACM Int. Symp. Mobile Ad Hoc Netw. Comput., 2016, pp. 161–170.
[52] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker, "High-speed switch scheduling for local-area networks," ACM Trans. Comput. Syst., vol. 11, no. 4, pp. 319–352, 1993.
[53] Huawei, "Huawei cloudengine 6800 user manual," 2018. [Online]. Available: https://support.huawei.com/enterprise/en/switches/cloudengine-6800-pid-7597815
[54] Huawei, "Huawei cloudengine 12800 user manual," 2018. [Online]. Available: https://support.huawei.com/enterprise/en/switch/cloudengine-12800-pid-7542409
[55] IEEE, "802.1Qaz enhanced transmission selection," 2011. [Online]. Available: https://1.ieee802.org/dcb/802-1qaz/
[56] Arista, "User manual: Arista EOS arista networks," 2017. [Online]. Available: https://www.arista.com/assets/data/docs/Manuals/EOS-4.17.1F-Manual.pdf
[57] Mellanox, "Mellanox MLNX-OS user manual for ethernet," 2017. [Online]. Available: http://www.mellanox.com/related-docs/prod_switch_software/MLNX-OS_ETH_v3_6_5000_UM.pdf
[58] Mellanox, "Mellanox breakout cables 40G to 4x10G and 100G to 4x25G," 2011. [Online]. Available: https://community.mellanox.com/docs/DOC-1450
[59] J. Eidson and K. Lee, "IEEE 1588 standard for a precision clock synchronization protocol for networked measurement and control systems," in Proc. Sensors for Ind. Conf., 2002, pp. 98–105.
[60] S. Ghorbani, Z. Yang, P. Godfrey, Y. Ganjali, and A. Firoozshahian, "DRILL: Micro load balancing for low-latency data center networks," in Proc. Conf. ACM Special Interest Group Data Commun., 2017, pp. 225–238.

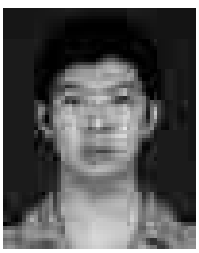
Kexin Liu received the BS degree from the Department of Software Engineering, Sun Yat-sen University, China, in 2017. She is currently working toward the PhD degree in the Department of Computer Science and Technology, Nanjing University, China. Her research interests include datacenter networks and network architecture.

Bingchuan Tian received the BE degree from the Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China, in 2016. He is currently working toward the PhD degree in the Department of Computer Science and Technology, Nanjing University, China. His research interests include intent-based networking, congestion control, and network scheduling.

Chen Tian received the BS, MS, and PhD degrees from the Department of Electronics and Information Engineering, Huazhong University of Science and Technology, China, in 2000, 2003, and 2008, respectively. He is an associate professor with the State Key Laboratory for Novel Software Technology, Nanjing University, China. He was previously an associate professor with the School of Electronics Information and Communications, Huazhong University of Science and Technology, China. From 2012 to 2013, he was a postdoctoral researcher with the Department of Computer Science, Yale University. His research interests include data center networks, network function virtualization, distributed systems, Internet streaming, and urban computing.

Bo Li received the BS degree from the Department of Computer Science and Engineering, Nanjing University of Science and Technology, China, in 2016. He is currently working toward the MS degree at Nanjing University, China. His research interests include distributed networks and systems.

Qingyue Wang received the BS degree from the Department of Computer Science and Technology, Wuhan University, China, in 2019. She is currently working toward the MS degree in the Department of Computer Science and Technology, Nanjing University, China. Her main research interest include datacenter networks.



Jiaqi Zheng received the PhD degree from Nanjing University, China, in 2017. He was a research assistant with the City University of Hong Kong, in 2015, and a visiting scholar with Temple University, in 2016. He is currently a research assistant professor with the Department of Computer Science and Technology, Nanjing University. His research interests include computer networking, particularly, data center networks, SDN/NFV, and machine learning systems. He is a member of the ACM. He received the Best Paper Award from IEEE ICNP 2015, Doctoral Dissertation Award from ACM SIGCOMM China 2018, and First Prize in the Jiangsu Science and Technology Award in 2018.

Jiajun Sun received the BE degree from the School of Computer Science and Communication Engineering, Jiangsu University, in 2015, and the ME degree from the Department of Computer Science and Technology, Nanjing University, in 2018. His research interests include distributed systems and networking. He is currently a R&D engineer with Alibaba Cloud.

Yixiao Gao received the bachelor's degree in information management and information system from Nanjing University, China, in 2017. He is working toward the PhD degree in the Department of Computer Science and Technology, Nanjing University, China. His research interests include some fields of datacenter networking, such as congestion control and application-specific optimization. He is a student member of the IEEE and ACM.

Wei Wang received the MS degree from the ESE Department, Nanjing University, in 2000, and the PhD degree from the ECE Department, National University of Singapore, in 2008. He is currently an associate professor with the CS Department, Nanjing University. His research interests include the area of wireless networks, including device-free sensing, cellular network measurements, and software defined radio systems.

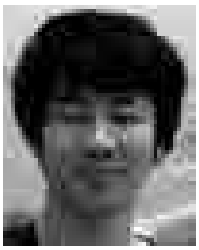
Guihai Chen received the BS degree in computer software from Nanjing University, in 1984, the ME degree in computer applications from Southeast University, in 1987, and the PhD degree in computer science from The University of Hong Kong, in 1997. He is a distinguished professor of Nanjing University. He had been invited as a visiting professor by the Kyushu Institute of Technology in Japan, University of Queensland in Australia and Wayne State University. He has a wide range of research interests with focus on parallel computing, wireless networks, data centers, peer-to-peer computing, high-performance computer architecture, and data engineering. He has published more than 350 peer-reviewed papers, and more than 200 of them are in well-archived international journals, such as the IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Computers, IEEE Transactions on Knowledge and Data Engineering, IEEE/ACM Transactions on Networking, and ACM Transactions on Sensor Networks, and also in well-known conference proceedings, such as HPCA, MOBIHOC, INFOCOM, ICNP, ICDCS, CoNext, and AAAI. He has won nine paper awards including ICNP 2015 Best Paper Award and DASFAA 2017 Best Paper Award.

Wanchun Dou received the PhD degree in mechanical and electronic engineering from the Nanjing University of Science and Technology, China, in 2001. He is currently a full professor of the State Key Laboratory for Novel Software Technology, Nanjing University. From April 2005 to June 2005 and from November 2008 to February 2009, he respectively visited the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, as a visiting scholar. Up to now, he has chaired three National Natural Science Foundation of China projects and published more than 60 research papers in international journals and international conferences. His research interests include workflow, cloud computing, and service computing.

Yanan Jiang received the BS degree from the Department of Software Engineering, Jilin University, China, in 2017. She is currently working toward the MS degree in the Department of Computer Science and Technology, Nanjing University, China. Her research interests include datacenter networks and network measurement.

Huaping Zhou received the BS degree from the School of Computer Science and Engineering, Beihang University, China. He is currently working toward the master's degree from the Department of Computer Science and Technology, Nanjing University, China. His research interests include datacenter networks and distributed systems.

Jingjie Jiang received the BEng degree from the Department of Automation, Tsinghua University, China, in 2012, and the PhD degree from the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, in 2017. She visited the Department of Electrical and Computer Engineering, University of Toronto, in 2015. She was a researcher with Theory Lab, Huawei Hong Kong research center from 2018 to 2019. She is currently with Google Shanghai. Her research interests include datacenter networking and scheduling, congestion control, block chain, and distributed systems.



Fan Zhang received the BEng (first class hons) degree from Chu Kochen Honors College, Zhejiang University (ZJU), in 2010, and the PhD degree from the Hong Kong University of Science and Technology (HKUST), in 2015. After graduation, he joined Future Network Theory Lab, Huawei Technologies as a senior engineer from 2015 to 2017. He was a senior engineer with the Hong Kong Applied Science and Technology Research Institute (ASTRI) from 2017 to 2018. He is currently with Theory Lab, Huawei Hong Kong Research Institute as a senior researcher. His research interests include mathematical modeling of computer networks, neural network in artificial intelligence, cross-layer QoE-aware resource allocation for 5G wireless systems, low complexity dynamic programming and control, networked control systems, stochastic optimization, convex optimization, and control theory.

Gong Zhang's major research directions are network architecture and large-scale distributed systems. He has abundant R&D experience on system architect in networks, distributed system, and communication system for more than 20 years. He has more than 90 global patents in which some play significant roles in the company. In 2000, he acted as a system engineer for L3+ switch product and became the PDT (Product development team) leader for smart device development, pioneering a new consumer business for the company since 2002. Since 2005, he was a senior researcher, leading future internet research and cooperative communication. In 2009, he was in charge of the Advance Network Technology Research Department, leading researches of future network, distributed computing, database system, and data analysis. In 2012, he became the principal researcher and led the system group in data mining and machine learning.