Computer Networks 143 (2018) 49–61

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Host-based scheduling: Achieving near-optimal transport for datacenter networks

Weibin Xie, Fang Wang*, Dan Feng, Lingling Zhang, Tingwei Zhu, Qingyu Shi

Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System (School of Computer Science and Technology, Huazhong University of Science and Technology), Ministry of Education of China, Wuhan, China

ARTICLE INFO

Article history:
Received 22 November 2017
Revised 16 June 2018
Accepted 25 June 2018
Available online 28 June 2018

Keywords:
Flow scheduling
Datacenter network
Low latency
Tail latency
Near-optimal transport

ABSTRACT

Datacenters use limited network resources to host complex and diverse applications, which requires transport schemes to treat diverse applications as a black box and provide low latency for latency-sensitive applications. Many schemes need to beforehand obtain flow information (e.g., flow size, deadline or traffic distribution) or require new hardware design or modification of applications, which leads to difficult use and inefficiency in practice. To solve the dilemma, we present Strict Priority Queuing (SPQ), an information-agnostic and readily deployable flow scheduling scheme, which provides near-optimal flow completion times (FCT) for latency-sensitive applications and effectively harnesses the long-tail behaviors of flows. Unlike the existing in-network priority schemes, SPQ enables host-based, fine-grained flow scheduling, leaving the in-network queuing mechanism simple. SPQ does not make any assumptions about the availability of any flow information and hence, can be applied to any types of datacenter applications. Moreover, SPQ approximates the Least Attained Service (LAS) scheduling discipline and hence is a near-optimal solution. Meanwhile, SPQ utilizes two novel feedback adjustment mechanisms to alleviate the possible negative impact of long flows on short flows. Our simulation results demonstrate that SPQ effectively addresses some major limitations of the in-network priority schemes, resulting in the near-optimal performance in reducing the average and tail latency. For example, the average FCT of short flows for SPQ only has a 0–3.5% gap with respect to the ideal information-aware scheme under a Hybrid workload.

1. Introduction

Today's datacenter applications are diverse, such as web searching, social networking, recommendation systems, and advertisement systems [1–3]. These applications generate a massive number of latency-sensitive short flows mixed with latency-insensitive long background flows, with variable flow sizes and diverse deadline requirements. Meanwhile, short flows need low latency to satisfy user demands. Even a very small delay increase can significantly impact the user experience and hence, reduce revenue [4–9]. Moreover, for many applications, the flow size is simply unknown in advance [7,10]. As a result, transport schemes cannot make assumptions about the availability of detailed flow information. In addition, packet transfer pacing is generally based on the feedbacks which incur delays of a round-trip-time (RTT) and hence, congestion can significantly affect the flow performance. Flows competing for shared network resources, such as a shared link, commonly cause packet losses and retransmissions due to timeouts [11]. As a result, FCT generally exhibits long-tail behaviors, with the tail-length up to two orders of magnitude larger than the mean [12].

Current solutions take two extreme approaches to address these problems. On one hand, the information-aware schemes [13–18] focus on achieving good performance, while largely overlooking the flexibility and complexity of design. On the other hand, information-agnostic schemes [1,2,19–21], while making no assumptions about the availability of detailed flow information and hence applicable to wide range of applications, offer limited performance.

The information-aware schemes need to beforehand obtain flow information, e.g., flow sizes, traffic distribution, and may require new hardware design or modification of applications [7,10,13,15]. These schemes usually use some mechanisms to definitely distinguish different flows based on the flow information, and achieve good performance. However, the flow information for datacenter applications may not always be available [4–6,22]. Moreover, con-

* Corresponding author.
E-mail addresses: hustxwb@hust.edu.cn, hustxwb@qq.com (W. Xie), wangfang@hust.edu.cn (F. Wang).

https://doi.org/10.1016/j.comnet.2018.06.020
1389-1286/

veying flow information to the transport layer can also be costly, especially when such information must be dynamically updated, as it may require customization of switch hardware or modification of applications [7,10,23].

Current information-agnostic schemes make no assumptions about the availability of flow information and hence, are applicable to a wide range of datacenter workloads and are generally easy to deploy, at the expense of offering limited performance. They improve FCT for short flows by adjusting flow rates in response to network condition changes [1,2,7,13,15]. For example, L2DCT imitates LAS to adjust the window sizes of flows per RTT. However, without supporting preemption or explicitly distinguishing latency-sensitive short flows from latency-insensitive long flows, the short flows may frequently wait behind the long flows, resulting in excessively large FCT for latency-sensitive short flows [12,15].

PIAS [7] is a good attempt to strike a balance between the complexity of design and performance gain. PIAS applies the Shortest Job First (SJF) policy to improve FCT for short flows, without requiring the modification of switches. However, PIAS does not completely address the problem, as it is a semi-information-agnostic scheme, which still assumes the availability of flow information, hence, limiting its scope of applicability, and offers moderate performance gain. More specifically, PIAS must acquire the traffic distribution to calculate multiple demotion thresholds. It is very difficult to accurately predict the demotion thresholds without precise flow information and hence, PIAS cannot provide good performance for applications whose flow sizes cannot be easily predicted. Second, PIAS uses the static demotion thresholds which cannot effectively fit workloads that change over time.

In view of the above status quo, we identify four important design goals to be achieved to fully tackle the cost or complexity of design versus performance challenge:

1. *Applicability to a wide range of datacenter applications:* The solution must be applicable to different datacenter applications with or without the availability of flow information.
2. *Near-optimal FCT for short flows:* The solution must strive to lower FCT for latency-sensitive short flows, comparable to the ideal information-aware schemes, e.g., pFabric [13].
3. *Dealing with long-tailed distribution effectively:* The solution must be able to harness the tail latency effectively, e.g., effectively lowering 99th and 99.9th percentile FCT for short and medium flows [12,18,24].
4. *Low implementation cost:* The solution should not require new hardware design or modification of datacenter applications, and should be compatible with the existing datacenter transport protocols.

In this paper, we present Strict Priority Queue (SPQ), aiming at achieving the above design goals. More specifically, SPQ is a host-based, information-agnostic and readily deployable flow scheduling scheme to provide near-optimal FCT for latency-sensitive short flows and deal with long-tail behaviors effectively. Unlike in-network priority scheduling schemes, such as pFabric [13] and PIAS [7] which use the same flow scheduling method at both host side and switch side, SPQ decouples host-side flow scheduling from switch-side flow scheduling to approximate the LAS scheduling discipline, and utilizes the L2DCT [1] protocol and Explicit Congestion Notification (ECN) [25] to provide rate control and loss recovery. Meanwhile, SPQ uses two novel feedback adjustment mechanisms to alleviate the possible negative impact of long flows on short flows.

Compared with pFabric [13] and PIAS [7] that generally divide all flows into 8 different priorities, SPQ is able to provide virtually unlimited number of priority levels, in terms of the number of bytes of a flow that has been sent to the fabric. This, according to the LAS algorithm [1,7], allows SPQ to achieve near-optimal scheduling with host-based flow scheduling. At switches, SPQ utilizes the simplistic, single First-In-First-Out (FIFO) queue with a shallow buffer. As a result, in line with the Internet design principle by pushing the complexity towards to edge of the Internet, SPQ can make full use of the host-side flexibility of dealing with flows to achieve the near-optimal scheduling order, while keeping the network core simple. This is in stark contrast to the existing in-network priority scheduling schemes [13,15] that require significant changes to the switches or applications, incurring much higher costs and less flexibility to adapt to the new application requirements.

In summary, this paper makes the following key contributions:

- We analyze the limitations of in-network prioritization and their inflexibility of adapting to diverse applications and workloads, and we identify the root cause of these limitations.
- We design SPQ, which achieves high performance and is applicable to any datacenter applications by two key innovations: decoupling host-side flow scheduling from switch-side flow scheduling and two novel feedback adjustment mechanisms.
- We evaluate the performance of SPQ using the NS2 simulator [26] in both non-oversubscribed [13,27,28] and oversubscribed fabrics [7], compared with two information-agnostic schemes (DCTCP and L2DCT), a semi-information-agnostic scheme (PIAS), and the ideal information-aware scheme (pFabric).

Our simulation results demonstrate that SPQ is able to achi-eve the four design goals. For example, under the Hybrid workload, the average FCT of short flows for SPQ only has a 0–3.5% gap to the ideal information-aware scheme (pFabric), and this gap is within 0–1.1% under the Data Mining workload. Furthermore, SPQ significantly outperforms DCTCP, L2DCT, and PIAS for all flow sizes, loads for all three workloads in two fabrics. For instance: it reduces the 99.9th percentile FCT of short flows by up to 91%, 90%, and 88% for the three workloads, respectively. Meanwhile, the results also show that SPQ is effective in avoiding some major limitations of in-network prioritization. For example, SPQ completely avoids the packet loss caused by in-network prioritization.

The rest of this paper is organized as follows. Section 2 presents the motivations, and analyzes some major limitations for in-network priority schemes. Section 3 presents the design of SPQ. Section 4 describes the implementation details on NS2. Section 5 describes the experimental evaluation for both non-oversubscribed and oversubscribed fabrics. Section 6 presents the related works. Finally, Section 7 concludes the paper.

2. Motivation and problem exploration

The performance of PIAS is restricted by the static demotion thresholds and the mismatch between demotion thresholds and traffic. With the variation of applications, the static demotion thresholds cannot provide stable performance for different applications, which can cause severe performance degradation and packet loss. Meanwhile, the inherent limitations of in-network prioritization cause the situation that PIAS cannot provide fine-grained performance, especially for the tail latency, which is explored and analyzed in Section 5.2.3. On the other hand, PIAS [7] claims that the mismatch only incurs an impact about 10% for the average FCT of short flows. Because the average FCT of short flows is more dependent on the scheduling algorithm. The significant advantage of in-network prioritization is that it is able to lower the average latency (average FCT) for short flows over other schemes. However, due to the static demotion thresholds and inherent limita-

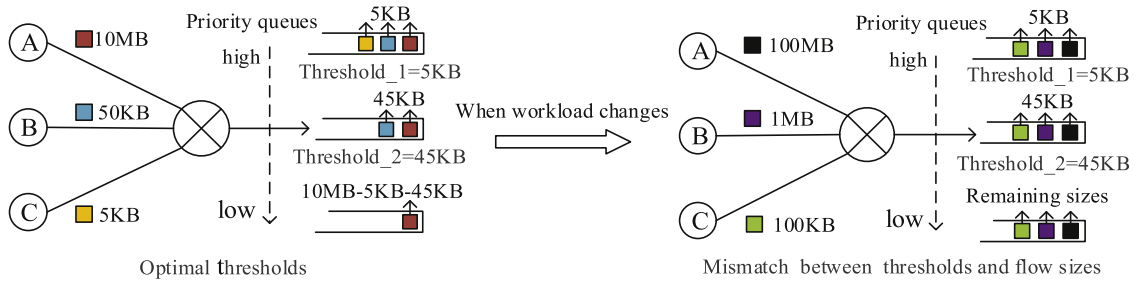


Fig. 1. A simple example showing that the mismatch between thresholds and workloads leads to a performance decline of PIAS. To simplify the description, we only use three priority queues.

tions of in-network prioritization, the applicable range of PIAS is restricted, and PIAS fails to provide good performance on reducing the tail latency (99th and 99.9th percentile FCT), mitigating upstream bandwidth loss (Fig. 6) and delivering fine-grained FCT (Fig. 4).

In this section, we explore the inherent limitations of in-network priority schemes, such as pFabric and PIAS. Such sche-mes generally assigns all flows to 8 priority levels based on specific flow information (e.g., flow size, deadline or traffic distribution) [7,13,15,18]. In particular, three major limitations are identified and their performance impacts are highlighted based on NS2 simulations for a leaf-spine datacenter topology with the Web Search workload which will be described in detail in Section 5.1. The topology is a widely used datacenter topology, and the workload is also a widely-used realistic traffic based on the productions of datacenters. Then, we summarize the benefits of decoupling host-side flow scheduling from switches in addressing these limitations and more.

2.1. Static demotion thresholds

The first limitation of in-network prioritization is that they try to divide diverse and variable flows with several static demotion thresholds, which limits their flexibility and range of applicability, making them hard to provide steady performance for practical applications. Flow patterns in datacenter networks may change from time to time due to diverse and complex datacenter workloads, dynamic tuning of these thresholds is impractical as the flow information may not be available [6,13,29]. Fig. 1 gives an example, demonstrating that the performance of PIAS may decline as the workload pattern changes. First, three hosts (*A*, *B*, *C*), respectively send different number of bytes ($f_A = 10\,\text{MB}$, $f_B = 50\,\text{KB}$, $f_C = 5\,\text{KB}$) to the same switch. To deliver low latency for short flows, PIAS sets optimal thresholds (5 KB, 45 KB) for the three priority queues in the left figure. However, when the workload changes ($f_A = 100\,\text{MB}$, $f_B = 1\,\text{MB}$, $f_C = 100\,\text{KB}$), the static thresholds do not match the new workload. The right figure shows that latency-sensitive short flows ($f_C = 100\,\text{KB}$) fall into the lowest priority queue, and may wait behind long flows. This situation is also very common in other in-network priority schemes.

To quantify the impact, we implement a simple PIAS experiment on NS2 [26]. In this experiment, we use two series of different demotion thresholds for PIAS: one is the optimal thresholds of the Web Search workload, and the other is the optimal thresholds for the Data Mining workload. Then we apply both sets of thresholds to the Web Search workload. With the improper thresholds, i.e., the optimal thresholds for Data Mining workload, Fig. 2 shows that the number of packet losses rises sharply. A large number of packet losses would inevitably cause serious bandwidth loss and timeouts thus retransmissions, causing severe performance degradation.

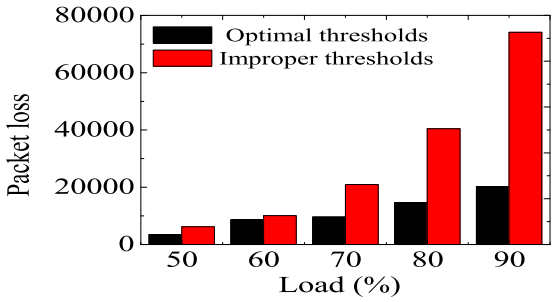


Fig. 2. Under Web Search workload, the number of packet losses for PIAS with different demotion thresholds.

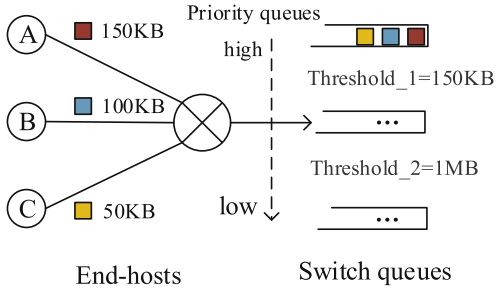


Fig. 3. A simple example showing that PIAS cannot strictly distinguish different flows due to the limited number of queues. To simplify the description, we only use three priority queues.

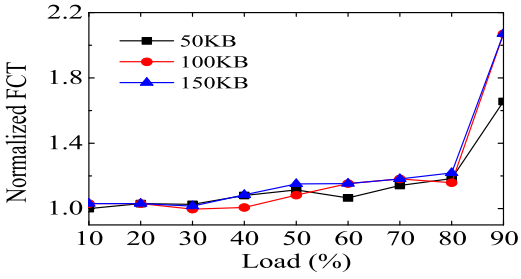


Fig. 4. PIAS: The normalized FCT of three different short flows for Web Search workload at various loads.

2.2. Limited number of queues

The second well-known limitation is that the number of available queues in existing commodity switches is limited, usually 4–10 queues [7,10,13,15]. The in-network priority schemes generally mark all flows with 8 different priorities at end-hosts, and allows service differentiation among up to 8 flow groups of distinct ranges of flow sizes [7,13,15,18]. However, this method is coarse-grained, and cannot strictly distinguish all flows. For example, Fig. 3 shows that PIAS sets two demotion thresholds, i.e., $\tau_1 = 150\,\text{KB}$ and

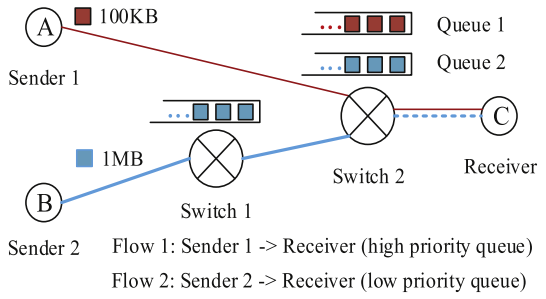


Fig. 5. A simple example illustrating that in-network priority schemes lead to upstream bandwidth loss.

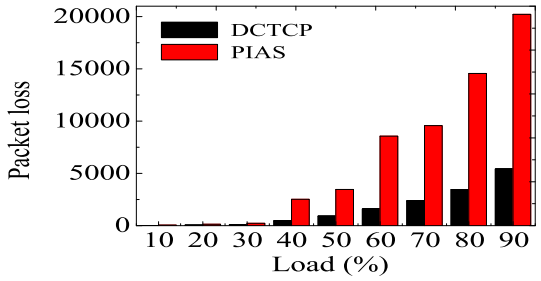


Fig. 6. Under Web Search workload, the number of packet losses for PIAS and DCTCP.

$\tau_2 = 1$ MB, to map flow groups of three size ranges into three priority queues. Then three different flows ($f_A = 150$ KB, $f_B = 100$ KB, $f_C = 50$ KB) are mapped in the same highest priority queue. In this case, in-network priority schemes cannot distinguish the three flows. In other words, the packets of flow C (50 KB) may wait behind the packets of flow B (100 KB) and flow A (150 KB). However, the optimal priority order should be A → B → C. This situation is very common in all queues of in-network priority schemes.

To quantify the impact, we run a simulation experiment to count the FCT for three different short flows (50 KB, 100 KB and 150 KB), which are mapped into the same highest priority queue. We employ the FCT of the shortest flow (50 KB) as the standard to normalize the other two short flows. Fig. 4 shows that the three different short flows achieve almost the same FCT at various loads. Especially, at 30%, 40%, and 50% loads, the shortest flow (50 KB) has larger FCT than the other two larger short flows (100 KB and 150 KB). The root cause is the lack of service differentiation among these flows due to coarse-grained priority queuing.

2.3. Upstream bandwidth loss

The last well-known limitation is that the in-network prioritization means that switches make local decisions for flow scheduling, which could easily lead to upstream bandwidth loss, especially for multi-link topologies [10,13,15]. For example, for the scenario depicted in Fig. 5, host B sends 1 MB data flows to receiver C, and host A sends 100 KB data flows to receiver C at the same time. The two flows must share the same output port at switch 2 and assume that flow A is mapped to a higher priority queue than flow B. As a result, packets from flow B cannot be sent until all the packets from flow A are sent. This, however, may lead to a large number of backlogged packets or even packet losses for flow B and hence severe upstream bandwidth loss.

To quantify the performance impact, we run the Web Search workload on NS2 to count the number of packet losses of PIAS [7] and DCTCP [2], respectively. Fig. 6 shows that the number of packet losses for PIAS is $2X - 7X$ those for DCTCP at various loads. According to above analysis, a large number of packet losses would inevitably bring severe bandwidth loss.

The root cause of these limitations for in-network priority schemes is that they want to provide the same flow scheduling scheme at both host-side and switch-side. However, the flexibility of handling flows has a huge difference between end-hosts and switches. We can implement many fine and complicated flow scheduling operations at end-hosts without any limitations, but cannot do this work at existing commodity switches. In fact, in-network priority schemes sacrifice the flexibility and performance of end-hosts, in exchange for priority scheduling at switches. This method limits their performance and applicable range in return. However, with decoupling host-side flow scheduling from switches, SPQ can avoid these restrictions to obtain the near-optimal performance and a widely applicable range.

2.4. Why SPQ works?

To maximize the datacenter resource utilization, today's clouds generally allow multiple applications to share a physical machine, e.g., using virtual machines, generating a large number of network flows [30,31]. These flows have to compete together for limited resources of the same Network Interface Card (NIC). As a result, the NICs at the hosts become the first and most important contention point of datacenter network fabric [7,13,32–34]. The ability to effective schedule latency-sensitive short flows versus latency-insensitive long flows at individual hosts can directly decide the performance of the entire datacenter network, because end-hosts directly decide the order and rates that all flows are sent to the fabric.

Moreover, host-based flow scheduling is free from the three limitations that plague the in-network priority schemes. First, since a host only needs to schedule flows emitted from the host itself, it can provide much finer grained scheduling than a switch, i.e., per-flow-based, rather than per-flow-group-based scheduling. Second, since the lower priority flows are throttled before they emitted into the fabric, host-based scheduling does not suffer from the upstream bandwidth loss. Third, since the number of priority levels for host-based flow scheduling is virtually unlimited, host-based flow scheduling does not need demotion thresholds and hence, is free from the drawbacks introduced by demotion thresholds.

Why can SPQ minimize the average and tail latency of flows, and can be applied to any types of datacenter applications with or without the availability of flow information? There are three key points that must be explained. *First, since we provide near-optimal scheduling only at the end-hosts, but not at switches, how to relieve the possible negative impact of long flows on short flows in the fabric?* In short, the problem is how to maintain low latency for short flows in the fabric. SPQ provides two feedback adjustment mechanisms to solve this problem: near-optimal scheduling at end-hosts and rate control based on the feedbacks of ECN and timeouts. SPQ utilizes the feedbacks of ECN and timeouts to infer fabric conditions, then decides how much effort must be used to adjust long flows' rates at end-hosts. In the premise of minimally sacrificing the rates of long flows, SPQ maintains low latency for latency-sensitive short flows.

Second, why is SPQ a real information-agnostic flow scheduling scheme with widely applicable ranges? PIAS not only needs to obtain the traffic distribution in advance, but also requires the datacenter applications unvarying. However, SPQ uses the host-based scheduling mechanism to avoid these problems. SPQ does not make any assumptions about the availability of any detailed flow information and hence, can be applied to any types of datacenter applications. Therefore, SPQ is a real information-agnostic flow scheduling scheme with widely applicable ranges.

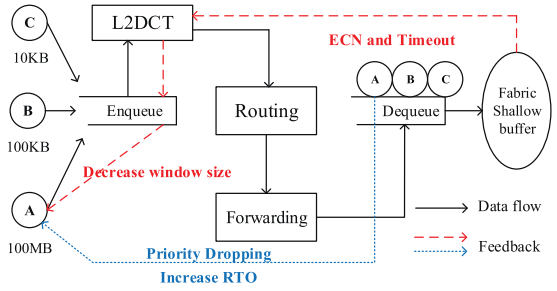


Fig. 7. The schematic diagram of SPQ.

Third, why can SPQ provide the near-optimal performance in reducing the average and tail latency? About information-agnostic scheduling algorithms, LAS is one of the best algorithms that minimize the average and tail latency [2,27,35]. By decoupling the host-side flow scheduling from switches, SPQ can imitate LAS to achieve the near-optimal scheduling at end-hosts. Meanwhile, SPQ uses L2DCT [1] to implement rate control and loss recovery, which also approximates LAS to adjust the window sizes of flows per RTT. Hence, SPQ is able to provide the near-optimal performance in reducing the average and tail latency.

3. SPQ design

The purpose of SPQ is to minimize the average and tail latency of flows and to achieve widely applicable ranges. The key problem of existing schemes is that they cannot simultaneously achieve the two goals. In this section, we present SPQ, a host-based flow scheduling scheme. Section 3.1 gives a overview and Section 3.2 presents the design details of SPQ.

3.1. Overview

SPQ decouples host-side flow scheduling from switches to approximate the LAS scheduling discipline, and utilizes L2DCT and ECN to provide rate control and loss recovery (Fig. 7). At end-hosts, SPQ marks all flows with different priorities based on the number of bytes that each flow has sent. Unlike in-network priority schemes that usually divide all flows into 8 different priority levels, SPQ can provide unlimited number of priority levels. In this way, SPQ can more effectively distinguish different flows at the finest possible granularity. Due to the high flexibility of handling flows at end-hosts and the effective scheduling algorithm, SPQ can achieve the near-optimal scheduling order at end-hosts.

At switches, SPQ uses a shallow buffer and the First-In-First-Out (FIFO) policy to avoid these well-known limitations of in-network priority schemes. In this way, SPQ can be applied to any types of datacenter applications with or without the availability of flow information. In addition, in order to alleviate the negative impact of long flows on latency-sensitive short flows, many datacenter transports [1,2,7,10] employ the shallow buffer which has a small buffer size. SPQ also utilizes the shallow buffer to accelerate the rotation of packets, alleviating the possible negative impact of long flows on short flows [2,15]. Meanwhile, SPQ utilizes the feedbacks of ECN and timeouts to infer the degree of network congestion, then decides how much flow rates should be adjusted for long flows. SPQ provides two feedback adjustment mechanisms to adjust the rates of long flows: near-optimal priority scheduling at end-hosts (the blue dotted line in Fig. 7), and rate control based on the feedbacks of ECN and timeouts (the red dotted line in Fig. 7).

In summary, SPQ always ensures that the smaller the number of bytes that a flow has sent into the fabric, the higher priority it has. If long flows limit the rates of latency-sensitive short flows in the fabric, inferred by ECN and timeout signals, SPQ adjusts the rates of long flows.

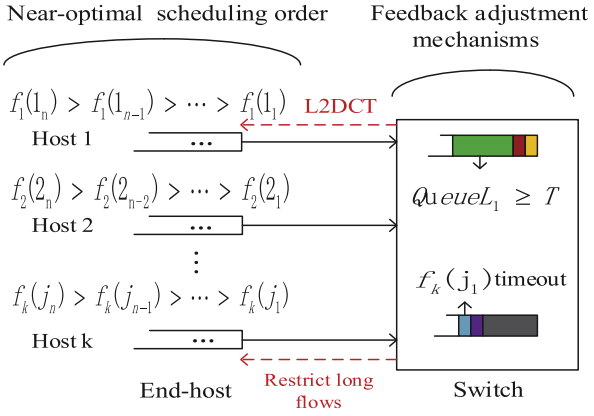


Fig. 8. Two different types of congestion that cause high latency for short flows.

3.2. The detailed design of SPQ

We use $f_i(q_m)$ to denote the flow q from host i, and use q_m $(1 \leq m \leq n)$ to represent the number of bytes that flow $f_i(q_m)$ has sent. Without loss of generality, we set:

$$q_m > q_{m-1} > \cdots > q_1 \tag{1}$$

In what follows, we use Fig. 8 to illustrate how SPQ can achieve near-optimal scheduling at end-hosts, and why it can be applied to any types of datacenters applications with or without detailed flow information.

3.2.1. Packet scheduling at end-hosts

Fig. 8 shows that SPQ achieves the near-optimal scheduling order for all hosts, namely:

$$f_i(q_m) \rightarrow f_i(q_{m-1}) \cdots \rightarrow f_i(q_1), \tag{2}$$

Where $1 \leq i \leq k$, $1 \leq q \leq j$, $1 \leq m \leq n$. In the information-agnostic case, according to the LAS algorithm, the number of bytes that each flow has sent can approximate the real flow size [1,7]. In order to minimize short flows' FCT, according to the order of bytes (inequality (1)), the near-optimal flow scheduling order for all individual hosts should follow formula (2). Hence, SPQ can provide much finer grained scheduling (per-flow-based) to effectively distinguish all different flows at end-hosts, and is able to avoid the limitation of having a limited number of queues.

More specifically, SPQ can classify all flows with virtually unlimited number of priorities which are used to determine the scheduling and dropping order of flows at individual hosts. When a packet arrives to the NIC, if it is identified as a new flow, its priority would be set as the size of the packet. If not, its priority is set at the number of bytes that the flow has accumulatively sent into fabric. Upon the arrival of a packet at the NIC, there are two cases when the packets from the lowest priority flow will be dropped. In the first case, when the NIC's queues are fully occupied, SPQ would discard the lowest priority packets. This case is normal priority scheduling at NIC's queues, regardless of network congestion. In the second case, fabric congestion worsens and long flows seriously restrict short flows' rates, which leads to the need to retransmit short flows' packets. In this case, SPQ discards the lowest priority packets regardless of whether NIC's queues are fully occupied or not.

The above host-based flow scheduling ensures that SPQ achieves higher bandwidth utilization than in-network priority

sche-duling schemes, because SPQ shifts the packet loss from fabric to the end-hosts, and the packet loss at end-hosts does not waste any bandwidth in fabric [13].

3.2.2. Packet scheduling at switches

SPQ's switches only need a single FIFO queue with a shallow buffer and Explicit Congestion Notification (ECN) [25]. ECN is a basic built-in function available for existing commodity switches [1,2,15]. SPQ does not need priority scheduling at switches. Since SPQ uses two feedback adjustment mechanisms to maintain low latency for latency-sensitive short flows, SPQ can also minimize the average and tail latency of flows without priority scheduling at switches. Meanwhile, SPQ makes full use of the shallow buffer to maintain the low latency for short flows in the fabric, as short flows require low queue occupancy to keep low latency while a large queue has been built up by long flows [29]. With a shallow buffer, the latency-sensitive short flows have more opportunities to occupy queues, and the feedback adjustments can achieve better effects to mitigate the network congestion and packet loss.

3.2.3. Rate control

Before describing the principle, we first discuss under which conditions congestion and packet loss take place. Fig. 8 shows two different types of congestion that cause high latency for packets of short flows. The first case: there are too many long flows occupying the queues of switches [15]. The second case: the link is fully utilized. We should utilize different mechanisms to cope with the two different cases, as they imply different fabric conditions. SPQ uses the feedbacks of ECN and timeouts to infer the fabric conditions.

When the end-host receives an ACK with ECN marking, the conclusion is that short flows' packets do not need to be retransmitted. It implies that the fabric is not fully utilized, but there are too many packets of long flows competing bandwidth with short flows' packets. In Fig. 8, when $QueueL_1 \geq T$ ($QueueL_1$ denotes the queue length of one switch port, T denotes the ECN marking threshold), it indicates that there are too many packets of long flows here. In this case, SPQ utilizes L2DCT [1] to adjust long flows' window sizes:

$$cwnd = cwnd * (1 - b/2) \tag{3}$$

When the end-host does not receive ECN marking, long flows' window sizes are set as:

$$cwnd = cwnd + k, \tag{4}$$

Where b is the backoff penalty, k is an increment in the congestion window per RTT. L2DCT imitates LAS to adjust the window sizes of flows per RTT, which is an implicit rate control. Based on the implicit rate control mechanism, L2DCT defines k and b, which can be used to adjust the congestion window size per RTT [1].

The other case is that the link is almost fully utilized, causing that the end-host detects a timeout of a short flow's packet. In Fig. 8, we use $f_k(j_1)$ to denote one short flow of host k (j_1 denotes the number of bytes that $f_k(j_1)$ has sent into fabric). When the end-host detects a timeout of $f_k(j_1)$, it reveals that the link is almost fully utilized.

In this case, we should adopt a more strict method to restrict long flows' rates. However, in the extreme case ($b = 1$), L2DCT only reduces the window sizes of long flows by half [1]. It is not enough to alleviate the negative impact that long flows limit the rates of short flows in fabric. On the basis of L2DCT (long flows' window sizes are still set as: $cwnd = cwnd * (1 - b/2)$), SPQ uses the other feedback mechanism to adjust long flows' rates: near-optimal priority scheduling at end-hosts. In this case, SPQ discards the lowest priority packets which belong to long flows at the NIC's queues, until the degree of congestion relieves to the first case. It is worth noting that SPQ does not discard the lowest priority packets in the queues of switches, because it would cause serious upstream bandwidth loss. When the degree of congestion falls, SPQ only uses L2DCT to adjust long flows' rates. Moreover, in the entire process, SPQ uses L2DCT to make the short flows keep normal rates.

In summary, SPQ uses the feedbacks of ECN and timeouts to infer the degree of network congestion, and then decides how much flow rates should be adjusted for long flows.

3.2.4. Loss recovery

SPQ uses different retransmission timeouts (RTO) based on the packet priorities and fabric conditions. At the beginning, all flows have the same value of RTO. When an ACK packet with ECN marking is received, and there are timeouts of packets, SPQ increases the long flows' value of RTO.

4. Implementation

We implement SPQ using NS2 [26] simulations. In this section, we present each implementation component of SPQ in detail.

4.1. End-host implementation

We implemented the L2DCT transport protocol and the near-optimal scheduling mechanism on NS2. the near-optimal priority scheduling mechanism built on the top of L2DCT (Fig. 7) is similar to that of a Linux kernel module. Meanwhile, SPQ encodes all packets' priorities in the IP headers based on the number of bytes that each flow has sent into fabric.

4.2. Switch implementation

There are two key implementations in switches: ECN and shallow buffer.

ECN: There are two most important parameters for ECN: the ECN marking threshold T and the weight g of the new sample. The ECN marking threshold T has an important effect on the rates of flows [36]. When T is set to a high value, there are too many long flows' packets competing with short flows' packets for the link in fabric. When T is set to a low value, long flows cannot obtain the high throughout performance. Hence, the ECN marking threshold should simultaneously deliver low latency for short flows and high throughout for long flows [1,2,8]. In addition, the weight g implicitly defines the congestion window size of a flow [1]. Referring to DCTCP, we set $g = 1/16$, and set $T = 65$ packets for 10 Gbps link [2].

Shallow buffer: The appropriate buffer size should greatly alleviate the negative impact of long flows on short flows without resulting in too many timeouts. In our simulations, we set the buffer size of each port to 360 KB.

5. Evaluation

In this section, by using the NS2 simulator [26], we evaluate the performance of SPQ compared with four different schemes. We divide the simulations into two parts. We first present the basic simulation settings. Then we evaluate five schemes in two topologies. We mainly focus on the non-oversubscribed fabric.

5.1. Simulation illustration

Fabric topology: We use two fabric topologies to evaluate the five schemes: a non-oversubscribed network [7,13,27,28] and an oversubscribed network [7]. Both of them use ECMP to implement load balancing [37–39]. As for the non-oversubscribed network, we utilize the leaf-spine topology including 4 spine (core) switches, 9

leaf (top-of-rack) switches, and 144 hosts. It is a widely used datacenter topology [13,27,28]. Each leaf switch uses 16 10 Gbps downlinks to connect 16 hosts, respectively, and uses 4 40 Gbps uplinks to connect 4 core switches respectively.

The other topology is a 3:1 oversubscribed fabric [7], including 4 spine switches, 3 leaf switches, and 144 hosts. Each leaf switch is connected to 48 hosts using 10 Gbps links and 4 spine switches using 40 Gbps links, thus forming an oversubscribed network.

Traffic workloads: We use three widely-used realistic datacenter workloads: a Web Search workload [12], a Data Mining workload [27], and a Hybrid workload [7] that consists of the former two workloads. In our simulations, all flows arrive according to a Poisson process, and the source and destination of each flow are selected randomly [7,13]. The three traffic workloads have a diverse mix of short and long flows with long-tailed characteristics [40]. In the Web Search workload, over 70% of all flows are less than 1 MB, however, over 95% of all bytes are from these flows of which the flow sizes are larger than 1 MB. The Data Mining workload is more extremely skewed: over 90% of flows are under 250 KB, but the 95% of all bytes comes from the 3.6% flows of which the flow sizes are over 35 MB [13]. Meanwhile, the sizes of short flows in Web Search workload are larger than those in Data Mining workload. Hence, The Data Mining workload is very easy to handle, but the Web Search workload is difficult to handle as it is likely that multiple flows are concurrently active in the same link [2,13]. Among most of our simulations and analysis, we focus on more challenging Web Search workload and Hybrid workload.

Benchmarks: We break down all flows across short flows (0, 100 KB], medium flows (100KB, 10MB], long flows (10 MB, ∞) [7,13,15,40]. The main objective of this paper is to provide the near-optimal FCT for latency-sensitive short flows and to effectively harness long-tail behaviors of flows. Hence, the main performance metrics of our simulations is the FCT. Meanwhile, for clear comparison, we normalize the FCT for all compared schemes. We divide the performance metrics into 4 parts:

- To show the differences of bandwidth loss, we count the number of packet losses for DCTCP, PIAS, and SPQ, respectively.
- To reveal the impact of the limited number of queues, we count the average FCT of three different short flows (Web Search workload) for PIAS and SPQ respectively.
- And we use the average FCT to reveal the differences that the five schemes deliver low latency for latency-sensitive short flows.
- Finally, to show the differences handling long-tailed distribution, we calculate the 99th percentile and 99.9th percentile FCT for short flows and medium flows [2,12,24].

Compared schemes: We compare SPQ with four datacenter transport schemes: two information-agnostic schemes including DCTCP [2] and L2DCT [1], a semi-information-agnostic scheme (PIAS [7]), and the ideal information-aware and state-of-the-art scheme (pFabric [13]). Meanwhile, we set some main parameters referring to the best settings of the four schemes with a few modifications which is more convenient for our simulations. The specific values of these parameters can be found in Table 1. Especially, we choose the optimal parameters for PIAS and pFabric based on the static workloads. However, they cannot get these optimal parameters for dynamic datacenter applications, it means that the performance of PIAS and pFabric would fall for dynamic datacenter applications.

Simulation purposes: Since the five schemes have different applicable ranges or restrictions, it is unconvincing if we assess them abandoning their applicable ranges. Hence, what we care about can be divided into the following three parts:

1. Compared with information-agnostic schemes (DCTCP and L2DCT), SPQ should achieve significant improvements in reducing the average (average FCT) and tail latency (99th and 99.9th percentile FCT), and should have the same applicable range with or without the availability of flow information. Our simulation results demonstrate these advantages of SPQ over DCTCP and L2DCT.
2. Compared with the semi-information-agnostic (PIAS), SPQ should achieve better performance in lowering the average and tail latency, and could be applied to any datacenter applications without the availability of the traffic distribution. According to above explorations (Section 2), we can find that the applicable range and performance of PIAS are restricted by the static demotion thresholds and the inherent limitations of in-network prioritization. These inherent limitations incur serious packet loss, and PIAS cannot strictly differentiate these different flows which are mapped into the same priority queue. Meanwhile, due to the static demotion thresholds, PIAS cannot provide stable performance for dynamic workloads, and PIAS fails to effectively handle the tail latency (99th and 99.9th percentile FCT). However, with decoupling host-side flow scheduling from switches, SPQ does not need demotion thresholds and priority scheduling in switches, and thus is able to fully avoid these major limitations of in-network priority schemes. As a result, SPQ achieves significant improvements over PIAS in lowering the average and tail latency and alleviating packet loss.
3. Compared with the ideal information-aware scheme (pFabirc), SPQ should achieve the analogous performance on reducing the average and tail latency of latency-sensitive short flows. The simulation results demonstrate that SPQ achieves the near-optimal performance.

Table 1
Main simulation parameter settings .

Scheme	Parameter
DCTCP	$qsize = 240$ pkts
L2DCT	$ECN\_threshold = 65$ pkts
PIAS	$min\_rto = 2$ ms
pFabric	$qsize = 120$ pkts
	$min\_rto = 250\ \mu s$
SPQ	$qsize = 240$ pkts
	$ECN\_threshold = 65$ pkts
	$min\_rto(short\ \ flows) = 2$ ms
	$min\_rto(long\ \ flows) = 200$ ms

5.2. Simulations in the non-oversubscribed fabric

In this section, we evaluate the five schemes in the non-oversubscribed fabric. We first show the number of packet losses with two tables. Then, we compare the average FCT of three different flows under the Web Search workload to reveal whether SPQ is restricted by the number of available queues in existing commodity switches. Finally, we focus on the average and tail latency.

5.2.1. Packet loss

In-network priority schemes are easy to cause serious upstream bandwidth loss, resulting in a large number of packet losses. Can SPQ avoid this drawback? To explain this problem, we count the number of packet losses for DCTCP, PIAS, and SPQ for Web Search workload and Data Mining workload respectively. In order to use the number of packet losses to denote the impact of upstream bandwidth loss, we only count the packet losses in the fabric. We do not add up the packet losses at end-hosts, as packet loss at end-hosts do not waste any fabric bandwidth [13,15]. In our simulations, we set some very moderate parameters (e.g., $min\_rto = 2$ ms) for PIAS, which is beneficial for in-network priority schemes to alleviate packet loss.

Table 2
The number of packet losses for Web Search workload .

Load (%)	10	20	30	40	50	60	70	80	90
SPQ	8	60	107	216	541	891	1906	3307	6782
DCTCP	55	139	229	468	921	1590	2381	3453	5415
PIAS	185	768	1653	2527	3466	8566	9574	14,630	20,217

Table 3
The number of packet losses for Data Mining workload.

Load (%)	10	20	30	40	50	60	70	80	90
SPQ	0	1	1	3	2	1	10	12	4
DCTCP	10	32	36	63	36	55	76	87	111
PIAS	3823	6130	9190	12,434	11,251	15,675	18,025	22,339	26,714

Tables 2 and 3 show the number of packet losses under the Web Search and Data Mining workloads, respectively. We can get three conclusions from the two tables:

1. First, the two tables demonstrate that SPQ can greatly alleviate upstream bandwidth loss and that the performance gap between SPQ and PIAS is very large. This is because the host-based flow scheduling mechanism is able to effectively alleviate the packet loss in the fabric.
2. Second, the number of packet losses for SPQ is slightly lower than those for DCTCP at both workloads (special case: the 90% load under Web Search workload). The reason is that the near-optimal scheduling mechanism and two feedback adjustment mechanisms can effectively ease packet loss for long flows. Furthermore, we can find that the special case is consistent with the latter simulation results in Section 5.2.3 (Figs. 10(a), 11(a), and 12(a)): L2DCT gets slightly worse performance than DCTCP at 90% load under the Web Search workload. And SPQ uses the L2DCT protocol to provide rate control and loss recovery.
3. Last, the packet losses' number of SPQ and DCTCP are less at Data Mining workload than those at Web Search workload, respectively, which is consistent with the characteristics of the two workloads. As the above description (Section 5.1), Data Mining workload is easier to handle than Web Search workload. However, this situation is reversed for PIAS: the number of packet losses for PIAS at Data Mining workload is larger than those at Web Search workload. The reason is that long flows' sizes are larger at Data Mining workload than those at Web Search workload. And long flows are always mapped into several lower priority queues, which causes that more long flows' packets are dropped under Data Mining workload.

5.2.2. The average FCT of the same priority short flows

Limited by the number of available queues in existing commodity switches, in-network priority schemes cannot strictly differentiate these different flows which are mapped into the same priority queue. Can SPQ avoid this limitation? To explain this problem, we count the average FCT of three different short flows (10 KB, 50 KB, and 100 KB) which are mapped into the same highest priority queue under Web Search workload. Fig. 9 demonstrates that SPQ can completely avoid the well-known limitation, because SPQ can provide much finer grained scheduling (per-flow-based scheduling) to effectively differentiate these flows at all individual hosts. In detail, the average FCT of three different short flows are divided into three different levels across all loads. As for PIAS (Fig. 4), however, the three different short flows (50 KB, 100 KB, and 150 KB) obtain almost the same FCT at various loads.

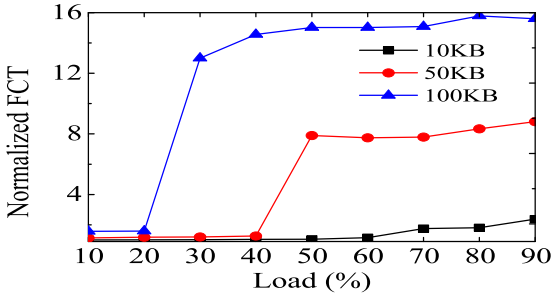


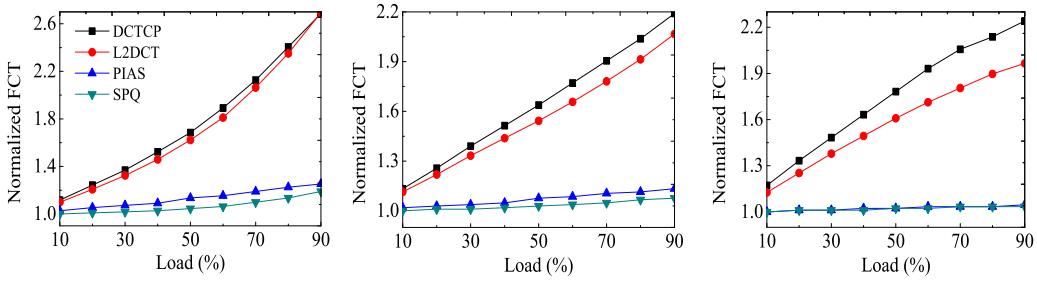
Fig. 9. SPQ: The normalized FCT of three different short flows for Web Search workload at various loads.

5.2.3. Comparison with information-agnostic and semi-information-agnostic schemes

Here, we compare SPQ with two information-agnostic schemes (DCTCP and L2DCT) and a semi-information-agnostic scheme (PIAS). And we employ the FCT of SPQ as the standard to normalize the FCT of other schemes. The results reveal that SPQ outperforms DCTCP, L2DCT, and PIAS for the three workloads in the non-oversubscribed network. We mainly make these observations: the average, 99th, and 99.9th percentile FCT of latency-sensitive short flows and medium flows. Meanwhile, we also count the average FCT of overall flows.

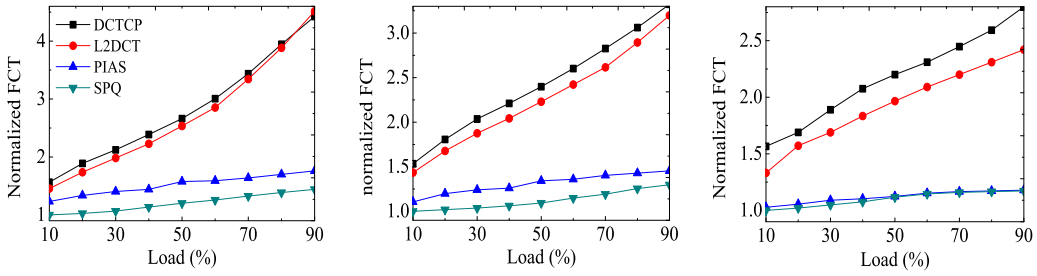
Latency-sensitive short flows: Figs. 10–12 show that SPQ can effectively minimize the average and tail latency for latency-sensitive short flows. Here, we summarize the following conclusions:

1. Minimizing average latency: SPQ significantly outperforms both DCTCP and L2DCT for the three workloads, and improves the average FCT of short flows by up to 56% and 55% under Web Search workload respectively. Meanwhile, SPQ also achieves better performance than PIAS for the three workloads. For example, SPQ reduces the average FCT of short flows by up to 8%. The reason is that SPQ achieves the near-optimal scheduling with decoupling host-side flow scheduling from switch-side.
2. Minimizing tail latency: SPQ significantly outperforms the three schemes in reducing the tail latency of short flows. In detail, compared with DCTCP, L2DCT, and PIAS, SPQ reduces the 99th percentile FCT by up to 67%, 69%, and 24%, respectively, and improves the 99.9th percentile FCT by up to 91%, 90%, and 88% respectively. We make two more detailed observations. First, PIAS obtains the poor performance for the 99.9th percentile FCT (Fig. 12(a)). Although in-network prioritization can effectively lower the average latency for short flows, the ability of handling tail latency is restricted by the inherent limitations of in-network prioritization. Second, SPQ can effectively harness the long-tail behaviors of flows, because SPQ can provide much finer grained scheduling than PIAS (i.e., per-flow-based, rather



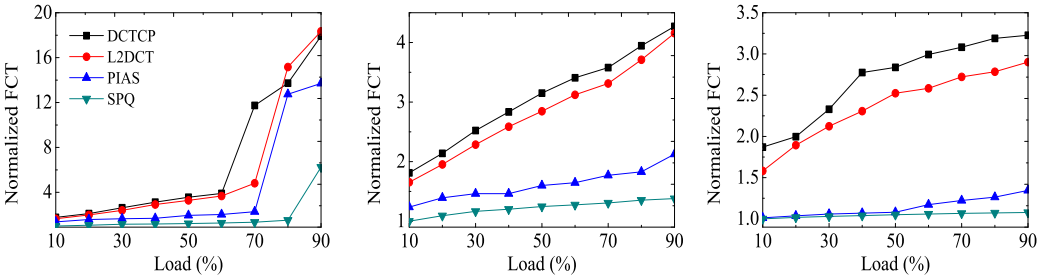
(a) Web Search workload (b) Hybrid workload (c) Data Mining workload

Fig. 10. (0,100 KB]: The normalized average FCT of short flows for three workloads at various loads.



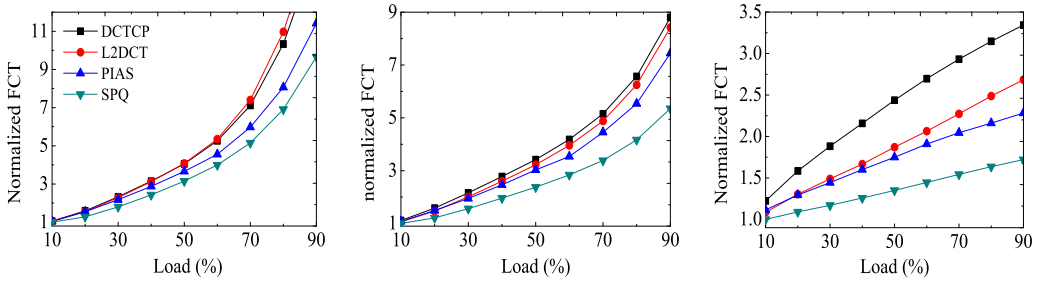
(a) Web Search workload (b) Hybrid workload (c) Data Mining workload

Fig. 11. (0,100 KB]: The normalized 99th percentile FCT of short flows for three workloads at various loads.



(a) Web Search workload (b) Hybrid workload (c) Data Mining workload

Fig. 12. (0,100 KB]: The normalized 99.9th percentile FCT of short flows for three workloads at various loads.



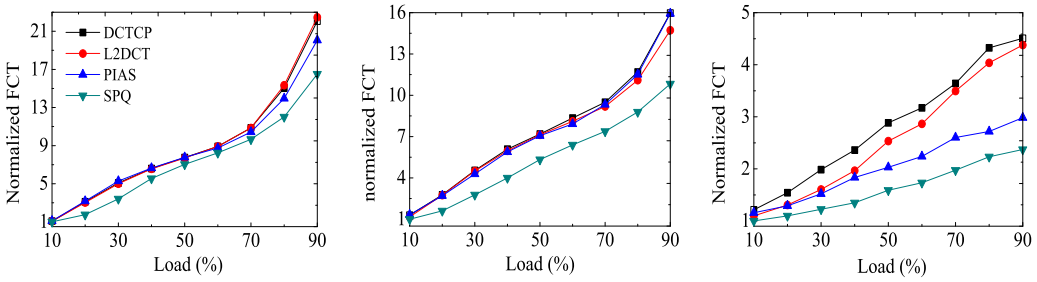
(a) Web Search workload (b) Hybrid workload (c) Data Mining workload

Fig. 13. (100 KB,10 MB]: The normalized average FCT of medium flows for three workloads at various loads.

than per-flow-group-based scheduling), which is good for reducing the tail latency.

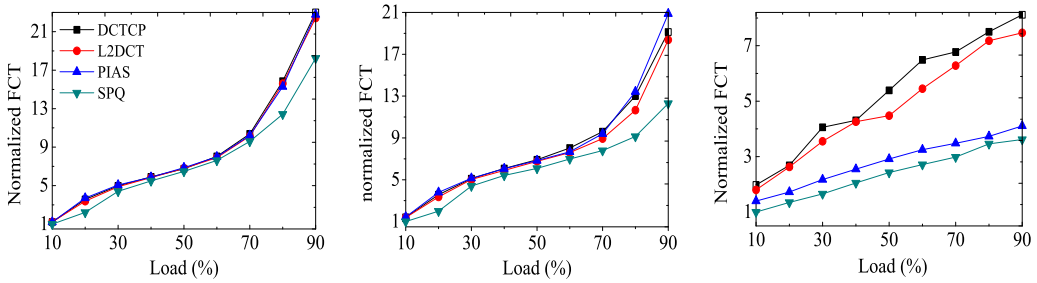
Medium flows: For medium flows in (100 KB,10 MB], Figs. 13–15 show that SPQ also significantly outperforms the three schemes for the three workloads. Under Web Search workload, compared with DCTCP, L2DCT, and PIAS, SPQ reduces the average FCT of medium flows by up to 37%, 40%, and 15% respectively. Especially, the average FCT of medium flows for SPQ is 28% lower than that for PIAS at 90% load under Hybrid workload. As for Data Mining workload, SPQ also achieves a great improvement of the average FCT over PIAS (Fig. 13(c)): SPQ improves the average FCT by up to 25%. To deliver low latency for short flows, in-network priority schemes frequently discard the packets of lower priority flows, leading to the moderate performance gain for medium flows.

How about the tail latency of medium flows? We make two detailed observations. First, SPQ achieves the very obvious advantage over DCTPC, L2DCT, and PIAS. For example, compared with DCTCP, L2DCT, and PIAS, under Hybrid workload, SPQ reduces the 99th percentile FCT by up to 32%, 26%, and 32% respectively, and re-



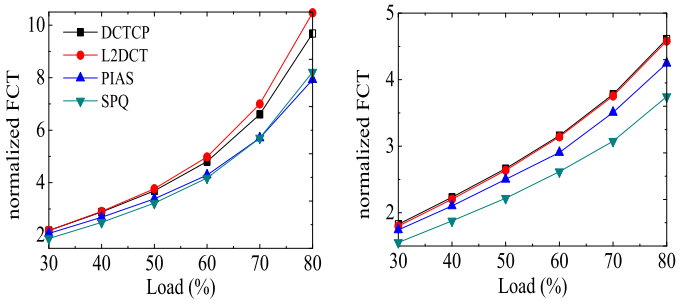
(a) Web Search workload (b) Hybrid workload (c) Data Mining workload

Fig. 14. (100 KB,10 MB]: The normalized 99th percentile FCT of medium flows for three workloads at various loads.



(a) Web Search workload (b) Hybrid workload (c) Data Mining workload

Fig. 15. (100 KB,10 MB]: The normalized 99.9th percentile FCT of medium flows for three workloads at various loads.



(a) Web Search workload (b) Hybrid workload

Fig. 16. Overall average: The normalized average FCT of overall flows for Web Search workload and Hybrid workload.

duces the 99.9th percentile FCT by up to 36%, 33%, and 41% respectively. Because SPQ is able to utilize the network resource more effectively, and can avoid hurting the lower priority flows. Second, PIAS even has worse performance than DCTCP and L2DCT for Web Search workload and Hybrid workload (Figs. 14(a), 14(b), 15(a), and 15(b)). PIAS always discards the packets of lower priority flows to maintain low latency for short flows, resulting in further enlarging the tail latency for medium flows. As for Data Mining workload, the performance gap is more obvious: SPQ improves the 99.9th percentile FCT by up to 55%, 52%, and 17%, respectively.

Overall flows: Fig. 16 shows that SPQ still achieves an obvious improvement for the average FCT of overall flows. Under Hybrid workload, compared with DCTCP, L2DCT, and PIAS, SPQ improves the average FCT of overall flows by up to 19%, 18%, and 12% respectively. This is because SPQ can effectively improve the bandwidth utilization for overall flows. However, under Web Search workload, SPQ gets the analogous performance to PIAS at high loads (Fig. 16(a)). The reason is that there are too many short flows restricting the long flows' rates at Web Search workload (more than half flows). At high loads, the packets of short flows more likely need to be retransmitted due to timeouts. As a result, in order to maintain low latency for short flows, SPQ employs a more rigorous method to restrict long flows' rates. In return, the average FCT of long flows would increase, causing that the average FCT of overall flows for SPQ increases at high loads. However, with the high flexibility and low complexity of design, SPQ effectively addresses some major limitations of the in-network priority schemes, and achieves the near-optimal performance in lowering the average and tail latency.

5.2.4. Comparison with the ideal information-aware scheme

How about the performance gap between SPQ and the ideal information-aware scheme (pFabric)? We utilize the FCT of pFabric as the standard to normalize SPQ. Fig. 17 indicates that SPQ achieves the analogous performance to pFabric. The performance gap between SPQ and pFabric is very small. At low loads, the average and 99th percentile FCT of short flows for SPQ are nearly the same with pFabric. Specifically, under Hybrid workload, the gap of the average FCT is within 0–3.5%, and the gap of the 99th percentile FCT is within 0–7.7%. Under Data Mining workload, the gap of the average FCT is within 0–1.1%, and the gap of the 99th percentile FCT is within 0–1.8%. However, SPQ can be applied to any datacenter applications with or without the availability of any flow information. It is worthwhile to sacrifice a little performance in exchange for the widely applicable range without any restrictions, which is also the main purpose of this paper.

5.3. Simulations in the oversubscribed fabric

In this section, we run the Web Search workload to evaluate SPQ compared with DCTCP, L2DCT, and PIAS in an oversubscribed fabric. We make the three observations. First, SPQ achieves significant improvements over the three schemes in lowering the average and tail latency for short flows (Fig. 18). For example, SPQ improves the average FCT by up to 28%, 26%, and 5% respectively, and improves the 99.9th percentile FCT by up to 57%, 53%, and 32%, respectively. Second, SPQ also significantly outperforms the three schemes for medium flows (Fig. 19). For example, SPQ improves the 99th percentile FCT by up to 26%, 25%, and 44%, respectively,

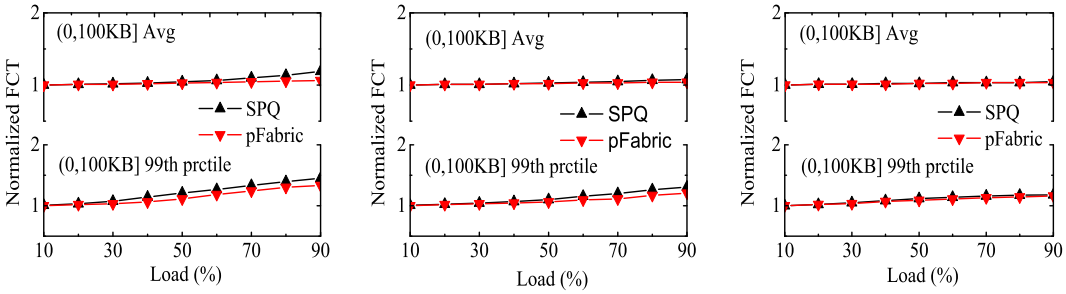


Fig. 17. The normalized FCT of short flows for three workloads at various loads.

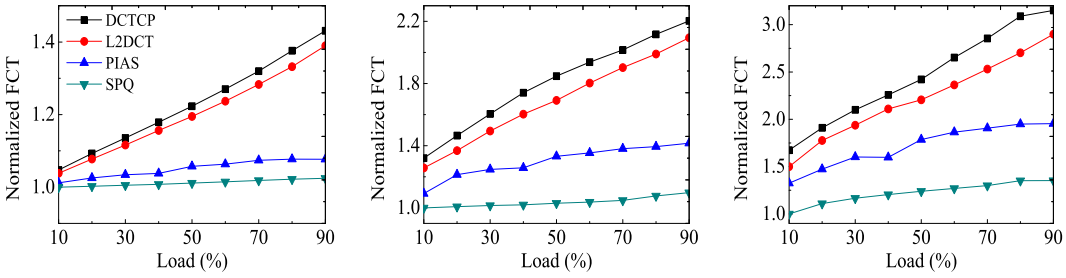


Fig. 18. Web Search workload: The normalized FCT for short flows in an oversubscribed network.

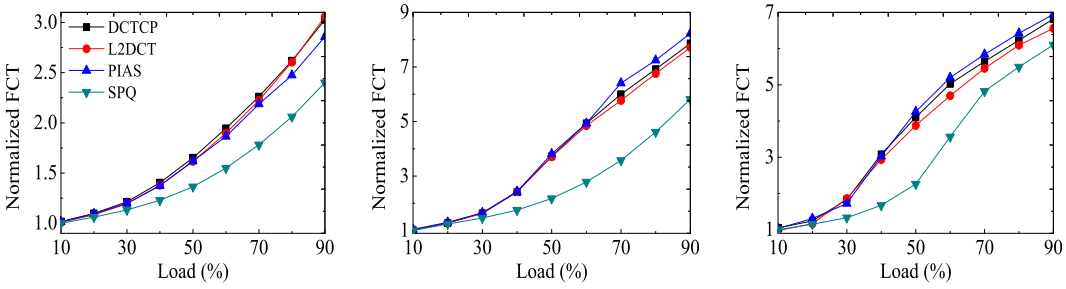


Fig. 19. Web Search workload: The normalized FCT for medium flows in an oversubscribed network.

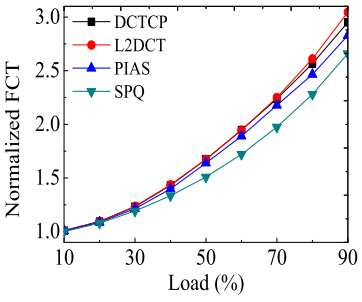


Fig. 20. Web Search workload: The normalized average FCT for overall flows in an oversubscribed fabric.

and improves the 99.9th percentile FCT by up to 45%, 42%, and 47%, respectively. Meanwhile, we can find that PIAS gets the poor performance for medium flows, especially for the tail latency (Fig. 19 (b) and (c)). Because PIAS frequently discards the packets of lower priority flows to maintain low latency for short flows in the fabric, which is easier to deteriorate the performance of lower priority flows (including medium flows) in the oversubscribed fabric. Third, SPQ also achieves the best performance for overall flows (Fig. 20): SPQ improves the average FCT by up to 10%, 13%, and 6%, respectively.

6. Related work

Existing flow scheduling schemes can be divided into three categories: information-agnostic scheduling schemes (e.g., [1,2,19–21]), information-aware scheduling schemes(e.g., [13–18]), and the semi-information-agnostic scheduling scheme (e.g., [7]).

Information-agnostic scheduling schemes: These schemes make no assumptions about the availability of flow information and hence, are applicable to a wide range of datacenter workloads and are generally easy to deploy, at the expense of offering limited performance. For example, DCTCP [2] improves the FCT of short flows by the shallow buffer and ECN [25]. On the basis of DCTCP, L2DCT [1], D2TCP [21], MCP [20], and HULL [19] add new mechanisms to lower latency for short flows or satisfy the deadlines for deadline-constrained traffic respectively. L2DCT mimics LAS to add size-awareness, and D2TCP [21] adds deadline-awareness. MCP [20] makes use of ECN to provide a more precise rate control mechanism. However, without supporting preemption or definitely distinguishing latency-sensitive short flows from latency-

insensitive long flows, these schemes achieve the inferior performance [15].

Information-aware scheduling schemes: These schemes focus on achieving good performance, while largely overlooking the flexibility and complexity of design. For example, they need to beforehand obtain flow information (e.g., flow sizes, traffic distribution), and may require new hardware design or modification of applications, which leads to difficult use and inefficiency in practice. These schemes generally use prior knowledge to definitely distinguish different flows. For example, to imitate the Shortest Remaining Processing Time (SRPT) algorithm, pFabric [13] assumes that all flow sizes and deadlines are known in advance. Meanwhile, pFabric uses the flow priorities to decide which packet to schedule or drop. On the basis of prior knowledge, PDQ [14] and D3 [18] use the switch arbitration and explicit rate control to implement flow scheduling [7], and PASE [15] synthesizes the strengths of some solutions to provide outstanding performance [15]. LSTF [16] simulates the Least Slack Time First (LSTF) policy.

Semi-information-agnostic scheduling schemes: The scheme (PIAS [7]) attempts to strike a balance between the complexity of design and performance gain. However, PIAS does not completely address the problem, as it still assumes the availability of flow information (i.e., traffic distribution), hence, limiting its scope of applicability, and offers moderate performance gain. Moreover, PIAS must use the traffic distribution to calculate multiple demotion thresholds, which makes these demotion thresholds difficult to fit the workloads that change over time.

7. Conclusion

In this paper, we present SPQ aiming at achieving near-optimal transport for datacenter networks while keeping the low complexity of design. SPQ enables host-based, fine-grained flow scheduling, leaving the in-network queuing mechanism simple. Due to two key innovations: decoupling host-side flow scheduling from switches and two novel feedback adjustment mechanisms, SPQ achieves the near-optimal performance in lowering the average and tail latency, and achieves a widely applicable range without making any assumptions about the availability of any flow information, new hardware design or modification of applications. The simulation results demonstrate that SPQ achieves all of our design goals, and that it is effective in avoiding some major limitations of in-network prioritization.

Future work: We intend to explore an efficient solution for the deadline-constrained traffic with the high flexibility and low complexity of design. To lower the average latency of flows, many schemes only need to consider the order how to schedule the packets of flows. However, to meet the deadlines of deadline-constrained flows with lowering the latency of flows, transport schemes must address two problems: in what order to schedule packets, and when to schedule them.

Acknowledgment

This work is supported in part by the National High Technology Research and Development Program (863 Program) of China under Grant No.2013AA013203, No.2015AA016701, No.2015AA015301; Hubei Province Technical Innovation Special Project (2017AAA129), Wuhan Application Basic Research Project (2017010201010103), Fundamental Research Funds for the Central Universities. This work is also supported by NSFC 61772216, 61502190, and CERNET Innovation Project NGII20170120.

References

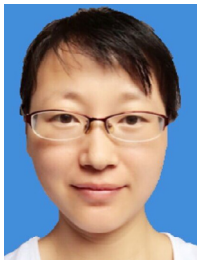
[1] A. Munir, I.A. Qazi, Z.A. Uzmi, A. Mushtaq, S.N. Ismail, M.S. Iqbal, B. Khan, Minimizing flow completion times in data centers, in: Proceedings of the IEEE INFOCOM, 2013, pp. 2157–2165, doi:10.1109/INFCOM.2013.6567018.
[2] M. Alizadeh, A. Greenberg, D.A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, M. Sridharan, Data center TCP (DCTCP), in: Proceedings of the ACM SIGCOMM, ACM, 2010, pp. 63–74, doi:10.1145/1851182.1851192.
[3] W. Guo, V. Mahendran, S. Radhakrishnan, Join and spilt TCP for SDN networks: architecture, implementation, and evaluation, Comput. Netw. 137 (2018) 160–172, doi:10.1016/j.comnet.2018.03.022.
[4] R. Mittal, V.T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, D. Zats, Timely: Rtt-based congestion control for the datacenter, in: Proceedings of the ACM SIGCOMM, 2015, pp. 537–550, doi:10.1145/2785956.2787510.
[5] C. Lee, C. Park, K. Jang, S. Moon, D. Han, Accurate latency-based congestion feedback for datacenters, in: Proceedings of the USENIX ATC, USENIX Association, 2015, pp. 403–415.
[6] T. Benson, A. Akella, D.A. Maltz, Network traffic characteristics of data centers in the wild, in: Proceedings of the ACM IMC, 2010, pp. 267–280, doi:10.1145/1879141.1879175.
[7] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, H. Wang, Information-agnostic flow scheduling for commodity data centers, in: Proceedings of the USENIX NSDI, 2015, pp. 455–468.
[8] W. Bai, L. Chen, K. Chen, H. Wu, Enabling ECN in multi-service multi-queue data centers, in: Proceedings of the USENIX NSDI, 2016, pp. 537–549.
[9] I. Cho, K. Jang, D. Han, Credit-scheduled delay-bounded congestion control for datacenters, in: Proceedings of the ACM SIGCOMM, 2017, pp. 239–252, doi:10.1145/3098822.3098840.
[10] L. Chen, K. Chen, W. Bai, M. Alizadeh, Scheduling mix-flows in commodity datacenters with Karuna, in: Proceedings of the ACM SIGCOMM, 2016, pp. 174–187, doi:10.1145/2934872.2934888.
[11] S. Lee, D. Lee, M. Lee, H. Jung, B.-S. Lee, Randomizing TCP payload size for TCP fairness in data center networks, Comput. Netw. 129 (2017) 79–92, doi:10.1016/j.comnet.2017.09.007.
[12] D. Zats, T. Das, P. Mohan, D. Borthakur, R. Katz, Detail: reducing the flow completion time tail in datacenter networks, in: Proceedings of the ACM SIGCOMM, 2012, pp. 139–150, doi:10.1145/2377677.2377711.
[13] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, S. Shenker, pFabric: minimal near-optimal datacenter transport, in: Proceedings of the ACM SIGCOMM, 2013, pp. 435–446, doi:10.1145/2486001.2486031.
[14] C. Hong, M. Caesar, P.B. Godfrey, Finishing flows quickly with preemptive scheduling, in: Proceedings of the ACM SIGCOMM, 2012, pp. 127–138, doi:10.1145/2342356.2342389.
[15] A. Munir, G. Baig, S.M. Irteza, I.A. Qazi, A. Liu, F. Dogar., Friends, not foes: synthesizing existing transport strategies for data center networks, in: Proceedings of the ACM SIGCOMM, 2014, pp. 491–502, doi:10.1145/2619239.2626305.
[16] M. Radhika, A. Rachit, R. Sylvia, S. Scott, Universal packet scheduling, in: Proceedings of the USENIX NSDI, 2016, pp. 501–521.
[17] A. Sivaraman, S. Subramanian, A. Agrawal, S. Chole, S. Chuang, T. Edsall, M. Alizadeh, S. Katti, N. McKeown, H. Balakrishnan, Towards programmable packet scheduling, in: Proceedings of the ACM HotNets-XIV, 2015, pp. 23:1–23:7, doi:10.1145/2834050.2834106.
[18] C. Wilson, H. Ballani, T. Karagiannis, A. Rowtron, Better never than late: Meeting deadlines in datacenter networks, in: Proceedings of the ACM SIGCOMM, 2011, pp. 50–61, doi:10.1145/2018436.2018443.
[19] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, M. Yasuda, Less is more: Trading a little bandwidth for ultra-low latency in the data center, in: Proceedings of the USENIX NSDI, 2012, pp. 253–266.
[20] L. Chen, S. Hu, K. Chen, H. Wu, D.H.K. Tsang, Towards minimal-delay deadline-driven data center TCP, in: Proceedings of the ACM HotNets-XII, 2013, pp. 21:1–21:7, doi:10.1145/2535771.2535788.
[21] B. Vamanan, J. Hasan, T.N. Vijaykumar, Deadline-aware datacenter TCP (D2TCP), in: Proceedings of the ACM SIGCOMM, 2012, pp. 115–126, doi:10.1145/2342356.2342388.
[22] N.K. Sharma, A. Kaufmann, T. Anderson, A. Krishnamurthy, J. Nelson, S. Peter, Evaluating the power of flexible packet processing for network resource allocation, in: Proceedings of the USENIX NSDI, 2017, pp. 67–82.
[23] J. Perry, H. Balakrishnan, D. Shah, Flowtune: flowlet control for datacenter networks, in: Proceedings of the USENIX NSDI, 2017, pp. 421–435.
[24] A.B. Downey, Evidence for long-tailed distributions in the internet, in: Proceedings of the ACM SIGCOMM Workshop on Internet Measurement, 2001, pp. 229–241, doi:10.1145/505202.505230.
[25] K.R.S. Floyd, D. Black, Rfc 3168: the addition of explicit congestion notification (ECN) to IP, 2001, http://www.rfc-editor.org/info/rfc3168.
[26] The network simulator ns-2, (http://www.isi.edu/nsnam/ns/).
[27] A. Greenberg, J.R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D.A. Maltz., P. Patel, S. Sengupta, Vl2: a scalable and flexible data center network, in: Proceedings of the ACM SIGCOMM, 2009, pp. 51–62, doi:10.1145/1592568.1592576.
[28] M. Al-Fares, A. Loukissas, A. Vahdat, A scalable, commodity data center network architecture, in: Proceedings of the ACM SIGCOMM, 2008, pp. 63–74, doi:10.1145/1402958.1402967.
[29] H. Wu, J. Ju, G. Lu, C. Guo, Y. Xiong, Y. Zhang, Tuning ECN for data center networks, in: Proceedings of the ACM CoNEXT, 2012, pp. 25–36, doi:10.1145/2413176.2413181.

[30] M. Chowdhury, Y. Zhong, I. Stoica, Efficient coflow scheduling with varys, in: Proceedings of the ACM SIGCOMM, 2014, pp. 443–454, doi:10.1145/2619239.2626315.
[31] Z. Zheng, N.B. Shroff, Online multi-resource allocation for deadline sensitive jobs with partial values in the cloud, in: Proceedings of the IEEE INFOCOM, 2016, pp. 1–9, doi:10.1109/INFOCOM.2016.7524430.
[32] K. Jang, J. Sherry, H. Ballani, T. Moncaster, Silo: predictable message latency in the cloud, in: Proceedings of the ACM SIGCOMM, 2015, pp. 435–448, doi:10.1145/2785956.2787479.
[33] V. Jeyakumar, M. Alizadeh, D. Mazi, B. Prabhakar, C. Kim, A. Greenberg, Eyeq: practical network performance isolation at the edge, in: Proceedings of the USENIX NSDI, 2013, pp. 297–311.
[34] Y. Xu, M. Bailey, B. Noble, F. Jahanian, Small is better: avoiding latency traps in virtualized data centers, in: Proceedings of the ACM SOCC, 2013, pp. 7:1–7:16, doi:10.1145/2523616.2523620.
[35] I.A. Rai, G. Urvoy-Keller, M.K. Vernon, E.W. Biersack, Performance analysis of las-based scheduling disciplines in a packet switched network, in: Proceedings of the ACM SIGMETRICS, 2004, pp. 106–117, doi:10.1145/1005686.1005702.
[36] N.K. Sharma, M. Liu, K. Atreya, A. Krishnamurthy, Approximating fair queueing on reconfigurable switches, in: Proceedings of the USENIX NSDI, 2018, pp. 1–16.
[37] L.M. Surhone, M.T. Tennoe, S.F. Henssonow, Equal-Cost Multi-Path Routing, Betascript Publishing, 2010.
[38] S. Ghorbani, Z. Yang, P.B. Godfrey, Y. Ganjali, A. Firoozshahian, Drill: Micro load balancing for low-latency data center networks, in: Proceedings of the ACM SIGCOMM, 2017, pp. 225–238, doi:10.1145/3098822.3098839.
[39] H. Zhang, J. Zhang, W. Bai, K. Chen, M. Chowdhury, Resilient datacenter load balancing in the wild, in: Proceedings of the ACM SIGCOMM, 2017, pp. 253–266, doi:10.1145/3098822.3098841.
[40] E. Vanini, R. Pan, M. Alizadeh, P. Taheri, T. Edsall, Let it flow: resilient asymmetric load balancing with flowlet switching, in: Proceedings of the USENIX NSDI, 2017, pp. 407–420.

Weibin Xie received the BE degree in Energy and Power Engineering from the China University of Mining and Technology (CUMT), Xuzhou, China, in 2011. He is currently a Ph.D. student majoring in Computer Architecture in HUST. His current research interests include Computer networks and protocols and distributed storage systems. He has a publication in an international conference IWQoS.

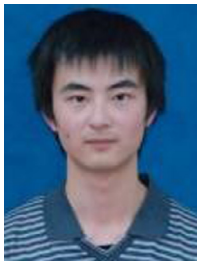
Fang Wang received her BE degree and Master degree in computer science in 1994, 1997, and Ph.D. degree in computer architecture in 2001 from Huazhong University of Science and Technology (HUST), China. She is a professor of computer science and engineering at HUST. Her interests include distribute file systems, parallel I/O storage systems and graph processing systems. She has more than 50 publications in major journals and international conferences, including FGCS, ACM TACO, SCIENCE CHINA Information Sciences, Chinese Journal of Computers and HiPC, ICDCS, HPDC, ICPP.

Dan Feng received the BE, ME, and Ph.D. degrees in Computer Science and Technology in 1991, 1994, and 1997, respectively, from Huazhong University of Science and Technology (HUST), China. She is a professor and vice dean of the School of Computer Science and Technology, HUST. Her research interests include computer architecture, massive storage systems, and parallel file systems. She has more than 100 publications in major journals and international conferences, including IEEE-TC, IEEE-TPDS, ACM-TOS, JCST, FAST, USENIX ATC, ICDCS, HPDC, SC, ICS, IPDPS, and ICPP. She serves on the program committees of multiple international conferences, including SC 2011, 2013 and MSST 2012. She is a member of IEEE and a member of ACM.



Lingling Zhang is currently a Ph.D. student majoring in Computer Architecture in Huazhong University of Science and Technology (HUST), Wuhan, China. His current research interests include computer architecture, big data and distributed storage systems.

Tingwei Zhu received the BE degree in computer science and technology from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2012. He is currently a Ph.D. student majoring in Computer Architecture in HUST. His current research interests include software-defined networking and distributed storage systems. He has several publications in international conferences, including IWQoS and ICPP.



Qingyu Shi received the BE degree in computer science and technology from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2014. He is currently a Ph.D. student majoring in Computer Architecture in Wuhan National Laboratory for Optoelectronics (WNLO). His current research interests include software-defined networking and network storage system.