Less is More: Dynamic and Shared Headroom Allocation in PFC-enabled Datacenter Networks

Danfeng Shan[1], Yuqi Liu[1], Tong Zhang[2], Yifan Liu[3], Yazhe Tang[1], Hao Li[1], Peng Zhang[1]
[1]*Xi'an Jiaotong University* [2]*Nanjing University of Aeronautics and Astronautics* [3]*Tsinghua University*

***Abstract*—In datacenters, lossless network is very attractive as it can achieve ultra-low latency. In commodity Ethernet, lossless forwarding is achieved by hop-by-hop Priority-based Flow Control (PFC). To avoid buffer overflow, PFC-enabled switches need to reserve some buffer as *headroom*, which is for absorbing in-flight packets during the delay for backpressure messages to take effect. However, with the growing link speed in production networks, the buffer becomes increasingly insufficient, and the headroom can occupy a considerable fraction of buffer. As a result, the remaining buffer for absorbing normal traffic bursts is significantly squeezed, leading to frequent PFC messages that degrade the network performance.**

However, the current static and queue-independent headroom allocation scheme is inherently inefficient in solving this problem. In light of this, we propose Dynamic and Shared Headroom allocation scheme (DSH), which dynamically allocates headroom to congested queues and enables the allocated headroom to be shared among different queues. By statistical multiplexing, DSH needs much less headroom to ensure lossless forwarding. Furthermore, DSH can be implemented on switching chips with moderate modifications. Extensive simulations show that DSH can absorb 4× more bursts without triggering PFC messages and reduce the flow completion time by up to ∼31%.

***Index Terms*—Priority-based Flow Control, Bursty Traffic, Buffer Management**

I. Introduction

Lossless network is increasingly attractive in datacenters as it can provide ultra-low latency for applications [1]–[5]. In commodity Ethernet, lossless transmission is achieved by the hop-by-hop Priority-based Flow Control (PFC) mechanism. To avoid packet dropping, a PFC-enabled switch sends a PAUSE frame to its upstream device when its buffer is about to overflow. Receiving the PAUSE frame, the upstream device holds back the packet transmission. To prevent buffer overflow, a fraction of buffer should be reserved as *headroom* to absorb in-flight packets before the PAUSE frame takes effect. However, in large-scale networks, PFC messages can result in serious performance issues, such as head-of-line blocking, congestion spreading, collateral damage, and even deadlocks [3], [4], [6]–[12]. Therefore, it is a common belief that PFC should be triggered as few as possible. Ideally, PFC should only serve as a backup method to ensure lossless packet transmissions.

However, due to the recent industrial trends, it is more and more likely that datacenter networks (DCNs) suffer from frequent PFC messages. Specifically, the link speed of DCN has rapidly grown from 1Gbps to 40Gbps/100Gbps and will continuously increase to 400Gbps in the near future [3], [13]. The amount of required headroom is increasingly large as it is positively related to the link speed. Unfortunately, the memory size in the switching chip cannot keep pace with the link capacity [14]–[16]. This is because datacenter switches usually employ on-chip memory for high-speed and low-latency memory access, and the memory size is limited by the chip area and cost. Specifically, the buffer size (related to switching capacity) has decreased by 4× over the past decade (§III-A). With these trends, a considerable amount of buffer should be reserved as headroom, significantly squeezing the buffer space for accommodating normal traffic. As a result, the queue length can easily hit the PFC pause threshold and PFC messages will be frequently generated. Although recent advances in end-to-end congestion control (CC) algorithms [1]–[4] can keep persistent buffer occupancy low, they cannot completely tackle this issue. This is because end-to-end CC takes at least 1 round-trip time (RTT) to react to traffic changes, and has no control over short-term congestion events, which are very common in DCNs. Specifically, studies have shown that most flows will be finished within 1 RTT in future DCNs [17], [18] and most congestion events will be caused by sub-RTT traffic bursts [19]. Within 1 RTT, it is the buffer management scheme that determines whether PFC messages can be avoided.

In face of the considerable headroom requirement, we find that the current headroom allocation scheme (which is called SIH in this paper) is quite inefficient in headroom utilization. Our experiments show that 75% of headroom keeps unused 99% of the time even when the network load is up to 90% (§III-B). This is because SIH reserves a *static* amount of worst-case headroom *independently* for every ingress queue in every port, which wastes a substantial amount of memory due to the following two reasons:

(1) Not all ingress queues need to occupy the headroom. An ingress queue takes up the headroom only when it becomes congested and it is very unlikely that all ports/queues are congested at the same time. Thus in most cases, most headroom keeps unused and is wasted.

(2) Every ingress queue is allocated with the worst-case headroom, while the worst case rarely occurs for all ingress queues simultaneously. The worst-case headroom requirement assumes that traffic arrives at the highest rate. In reality, queues of the same ingress port share the common uplink capacity, and thus it is impossible that all ingress queues are receiving traffic at line rate.

Note that SIH's inefficiency is inherent, lying in its queue-independent and static headroom allocation method. Thus, the problem cannot be simply addressed by adjusting the amount of reserved headroom, which may lead to unacceptable packet

2575-8411/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDCS57875.2023.00019

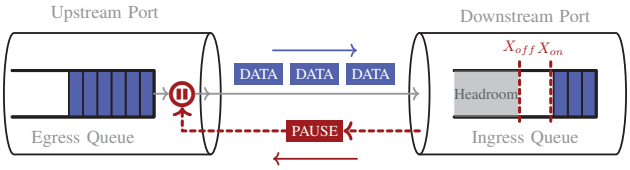


Fig. 1: Hop-by-hop priority-based flow control

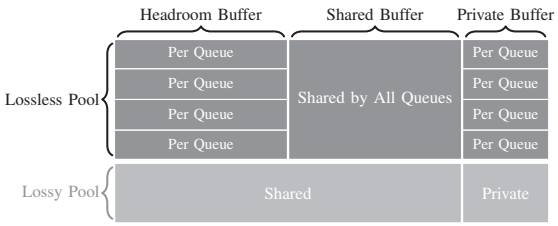


Fig. 2: Buffer Partition in a PFC-enabled Switch

loss for lossless networks. In this case, a brand-new headroom allocation scheme is needed to radically resolve this issue.

In light of this, we propose *Dynamic and Shared Headroom* allocation scheme (DSH) (§IV) to significantly reduce the headroom allocation without risking packet loss. To this end, DSH combines *port-level* flow control and *queue-level* flow control. The *port-level* flow control performs flow control operations on port-wise, which enables DSH to guarantee lossless transmission with a small amount of reserved headroom. It is based on the observation that different ingress queues in the same port naturally share the common uplink capacity. Thus, to avoid packet overflow, there is no need to independently reserve headroom for each ingress queue. Instead, DSH only needs to reserve headroom for each port and make the ingress queues in the same port share the reserved headroom. However, the headroom reduction comes at a cost. Occupying the headroom reserved for each port will pause the entire traffic from the upstream link, which is harmful to performance as performance isolation is violated. Thus, DSH also utilizes queue-level flow control most of the time and makes port-level flow control merely an insurance method against packet loss.

The *queue-level* flow control performs flow control actions on queue-wise, which is the same as PFC. It is based on the observation that not all queues need to occupy headroom at the same time. Thus, DSH *dynamically* allocates headroom to each queue only when the queue gets congested. In this way, the headroom is allocated only when truly needed, and thus will not be wasted. Furthermore, DSH enables the headroom to be *shared* among different ingress queues. In this way, DSH can take advantage of statistical multiplexing to efficiently utilize the allocated headroom.

We evaluate DSH with extensive ns-3 simulations (§V). Our microbenchmarks show that DSH can absorb over 4× more bursty traffic without triggering PFC messages, and can effectively mitigate the impairments (including collateral damage and deadlock) induced by PFC (§V-A). Our large-scale simulations show that DSH reduces the flow completion time by up to ∼57.7% for short fan-in flows and up to ∼31.1% for other flows (§V-B).

The rest of the paper is organized as follows. §II introduces the background of PFC and switch buffer. §III discusses the problem of the current headroom allocation scheme. Next, §IV describes the design of DSH. In §V, we present the evaluations of DSH. Finally, §VII concludes the paper.

II. Background

In this section, we introduce the background of PFC and switch buffer.

A. Priority-based Flow Control

Ethernet-based datacenter networks rely on Priority-based Flow Control (PFC) [20] to guarantee lossless packet forwarding. PFC is a hop-by-hop flow control mechanism (as shown in Fig. 1). In a PFC-enabled switch, once the length of an ingress queue exceeds a preset threshold (i.e., X_{off}), the switch sends a PAUSE frame to the upstream device. Receiving the PAUSE frame, the upstream device suspends the packet transmission for some duration specified by the PAUSE frame. When the ingress queue length falls below another threshold (i.e., X_{on}), the device sends a PAUSE frame with zero duration (we refer to it as *RESUME frame* in this paper) to the upstream device, which recovers the packet transmission.

To prevent packet dropping, X_{off} should be set conservatively. This is because it takes time for the PAUSE frame to arrive at the upstream device and take effect. To prevent buffer overflow, enough buffer beyond X_{off} should be reserved to accommodate in-flight packets during this time. The reserved buffer beyond X_{off} is called *buffer headroom*.

In the PFC standard [20], traffic can be classified into 8 priority classes. Each priority class is mapped to a separate queue and packets belonging to different priority classes are placed into different queues. The PFC messages carry the priority information and can only pause/resume a specific traffic class.

B. Buffer Architecture in PFC-enabled Switches

On a switching chip, the packets waiting to be transmitted are stored in a packet buffer. Today's commodity high-speed switching chips usually employ on-chip shared memory for high-bandwidth and low-latency packet access [3], [13], [21]–[28].

Basically, in a PFC-enabled switch, the buffer is partitioned into two pools [25], [29], [30] (as shown in Fig. 2). One pool is dedicated to lossless traffic that relies on PFC to avoid congestion loss. The other pool is dedicated to lossy traffic that is allowed to be dropped due to buffer overflow. Buffer is hard partitioned in such way to ensure performance isolation between two kinds of traffic. In this paper, we mainly focus on the lossless pool.

In the lossless pool, the buffer is further partitioned into three segments:

- **Private buffer:** buffer space reserved for each queue, which guarantees each queue's minimum buffer resource.
- **Shared buffer:** buffer space shared among all queues.
- **Headroom buffer:** buffer space reserved for each queue, which absorbs in-flight packets after sending PAUSE frames.

C. Buffer Allocation and Management in the Lossless Pool

The switching chip utilizes a Memory Management Unit (MMU) to allocate the buffer to arriving packets. For lossless traffic, MMU manages the buffer from the ingress perspective [29]–[31] (i.e., allocating buffer to ingress ports/queues). The sizes of private buffer and headroom buffer are explicitly configured. The remaining buffer serves as shared buffer.

Private buffer. There is no explicit rule specifying how to configure the private buffer size. Nevertheless, the amount of private buffer is relatively small (e.g., 16% in Arista 7050X3 switches [26]).

Headroom buffer. Different from the private buffer, the size of the headroom buffer should be carefully configured to prevent packet loss. This is because it takes some delay for a PAUSE frame to take effect, and the MMU needs to reserve enough headroom to absorb the arriving traffic during this delay. According to [1], [3], [32], [33], the headroom size for each ingress queue (denoted by η) is given by

$$\eta = 2\left(C \cdot D_{prop} + L_{MTU}\right) + 3840\text{B} \tag{1}$$

where C is the capacity of the upstream link, D_{prop} is the propagation delay of the upstream link, and L_{MTU} is the length of an MTU-sized packet. The rationale of such a setting is as follows. The delay for PFC pause to take effect comprises the following five parts:

① Waiting delay: A port may be busy transmitting another packet when a PAUSE frame is generated. The PAUSE frame needs to wait for the transmission to be finished. In the worst case, the port just begins to transmit the first bit of an MTU-sized packet, and thus the PAUSE frame needs to wait for L_{MTU}/C time.

② Propagation delay (of PAUSE frame): It takes D_{prop} time for the PAUSE frame to arrive at the upstream device. D_{prop} depends on the cable length and propagation speed of signals. In datacenters, the distance between two connected switches can be as large as 300 meters [3]. For single-mode fibers, the speed of light is 65% of that in a vacuum. As a result, the propagation delay is $\sim$1.5μs.

③ Processing delay: It takes some time for the switch to process the PAUSE frame and stop the transmission. The PFC definition has capped this time to 3840B/C [32].

④ Response delay: When the upstream port decides to execute the pause action, it might be sending another packet. In the worst case, the switch just begins to transmit the first bit of an MTU-sized packet. Thus, it takes L_{MTU}/C for the pause action to truly take effect.

⑤ Propagation delay (of the last packet): When the upstream device stops sending packets, there are still some in-flight packets on the link, which should also be absorbed by the headroom. It takes another D_{prop} time for the last sent packet to arrive at the downstream switch.

Combining the above five parts results in Eq. (1).

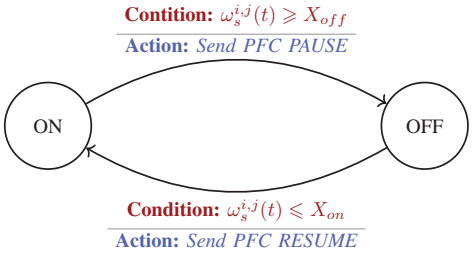


Fig. 3: State transition diagram of PFC

Shared buffer. The shared buffer is available to all queues. MMU utilizes a buffer management scheme to ensure fair and efficient allocation of the shared buffer among all queues. Among various buffer management schemes, Dynamic Threshold (DT) is the most common one on commodity switching chips [3], [4], [21]–[23], [26], [29], [34]–[36].

DT uses a threshold to restrict the length of each queue. The threshold is dynamically adjusted according to the remaining buffer size. Specifically, let $T(t)$ denote the threshold at time t, $\omega_s^{i,j}(t)$ denote the amount of shared buffer occupation for queue j in port i at time t, and B_s denote the buffer size of the shared segment. The threshold is given by

$$T(t) = \alpha \cdot \left(B_s - \sum_i \sum_j \omega_s^{i,j}(t)\right) \tag{2}$$

where α is a control parameter. The rationale behind DT is as follows. When the network is less congested, the amount of buffer occupancy is low and thus the remaining buffer size is high. DT adjusts the threshold to a higher value, which enables each queue to occupy more buffer, making the buffer efficiently used. On the contrary, when the network is more congested, the amount of buffer occupancy is high and thus the remaining buffer size is low. DT adjusts the threshold to a lower value, which restricts the buffer occupations of each queue, making the buffer fairly shared among different queues.

MMU workflow with PFC enabled. With PFC enabled, MMU monitors the ingress queue lengths (denoted by $q^{i,j}(t)$) and decides where to place each arriving packet. Furthermore, MMU generates PFC PAUSE messages to upstream devices based on the amount of shared buffer occupancy and X_{off}/X_{on} thresholds[1]. The PFC mechanism can be described as the state transition diagram in Fig. 3[2].

[1]With private buffer, the X_{off}/X_{on} thresholds are compared with the amount of shared buffer occupancy, rather than queue length.

[2]Note that the PFC standard uses pause timers to suspend the traffic transmission rather than ON/OFF states. Nonetheless, it is logically identical to the ON/OFF signals as the pause duration specified by a PAUSE frame is usually longer than the duration for the queue length to fall bellow the X_{on} threshold. Thus, we use ON/OFF states to describe the PFC mechanism for the ease of understanding.

Specifically, for each arriving packet, there are four cases:

① $q^{i,j}(t) < \phi$ (ϕ is the amount of private buffer reserved for each ingress queue): MMU puts the packet into the private buffer.

② $\phi \leqslant q^{i,j}(t) < \phi + T(t)$: MMU puts the packet into the shared buffer. Furthermore, if the ingress queue is in OFF state and $\omega_s^{i,j}(t) < X_{on}$, MMU triggers a RESUME frame (i.e., PFC PAUSE frame with zero pause duration) to the upstream device and makes the ingress queue turn into ON state.

③ $\phi + T(t) \leqslant q^{i,j}(t) < \phi + T(t) + \eta$: MMU puts the packet into the headroom buffer. Furthermore, if the ingress queue is in ON state, MMU triggers a PAUSE frame to the upstream device and makes the ingress queue turn into OFF state.

④ $q^{i,j}(t) \geqslant \phi + T(t) + \eta$: MMU drops the packet.

III. Motivation

In this section, we present the problem of the current headroom allocation scheme.

A. Headroom Occupies Considerable Memory

It is expected that most of the memory should serve as shared buffer to absorb bursty traffic without triggering PFC messages. However, with the current buffer allocation scheme, the headroom buffer occupies considerable memory, which can significantly squeeze "footroom" buffer[3] and result in frequent PFC messages.

Specifically, the current buffer allocation scheme *independently* reserves a *static* headroom for every ingress queue [1], [3]. Assume that each ingress queue requires η headroom. The total headroom size (denoted by h) is given by

$$h = N_p \times N_q \times \eta \tag{3}$$

where N_p is the number of ingress ports, N_q is the number of queues per port, and η is given by Eq. (1).

With such method, MMU has to allocate worst-case headroom for every ingress queue, and *headroom can occupy a large fraction of memory*. For example, Broadcom Trident2 switching chip contains 12MB memory. It has 32 40GbE ports (i.e., $N_p = 32$ and $C = 40$Gbps). For each port, the PFC standard supports 8 queues (i.e., $N_q = 8$). Assume that the MTU is 1500B (i.e., $L_{MTU} = 1500$B) and $D_{prop} = 1.5\mu$s, MMU needs to allocate ∼5.33MB memory for headroom buffer in total, which occupies 44.4% of total memory.

With the growing link capacity, this situation gets worse. The link speed of production DCN has grown from 1Gbps to 40Gbps and 100Gbps in the past decade [4], [13] and is continuously growing. With higher link speed, MMU needs to allocate more headroom to avoid buffer overflow. However, the buffer size is limited by the chip area and cost, and thus cannot scale with the switching capacity [14]–[16]. As a result, *the fraction of required headroom becomes increasingly large*, significantly squeezing the footroom buffer. Fig. 4 depicts the trend of buffer size and the fraction of required headroom in Broadcom's switching chips. The switch buffer size per unit of capacity has decreased by 4× in the last decade (from 157μs to 37μs), while the fraction of required headroom has increased by 56% (from 43% to 67%).

[3]For convenience, we define footroom as the buffer other than headroom.

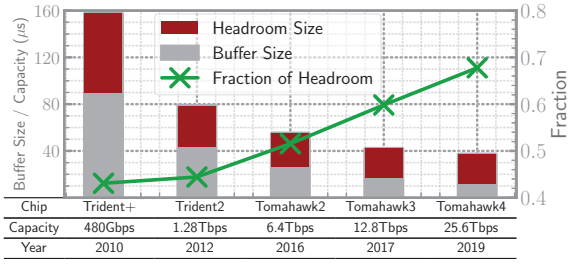


Fig. 4: Trends of buffer in Broadcom's switching chips

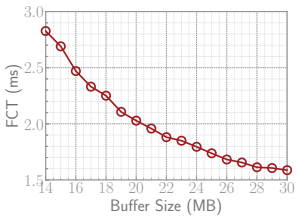


Fig. 5: FCT vs. buffer size

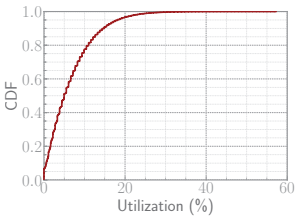


Fig. 6: Headroom utilization

Without enough "footroom" buffer, PFC messages can be frequently triggered, which may result in serious performance impairments (e.g., head-of-line blocking, congestion spreading, collateral damage) and even network deadlocks.

To quantitatively demonstrate the performance degradation brought by inadequate buffer, we perform a large-scale ns-3 simulation on a 256-server leaf-spine topology (more details in §V-B). The congestion control algorithm is PowerTCP [37], which can effectively keep persistent queue length low. We use the widely-used web search workload [27] to generate realistic DCN traffic. The total network load is 90%. Fig. 5 shows the average flow completion time (FCT) with different buffer sizes. The FCT with 14MB buffer is 78.1% worse than that with 30MB buffer.

To alleviate this problem, current network operators have to restrict the number of priority queues [3]. However, this ad hoc approach can aggravate the head-of-line blocking problem, as different services cannot be isolated and the congestion of one point can spread to the entire network. Furthermore, lots of studies [17], [38]–[41] have shown that multiple service queues can greatly improve the network performance. Restricting the number of queues prevents the network applications from benefiting from them.

B. Current Headroom Allocation Scheme is Inefficient

Despite the increasingly large fraction of headroom, the current *static and independent* headroom allocation scheme (referred to as SIH) is still quite inefficient and wasteful. This is due to three reasons:

(1) Not all queues need to occupy headroom. An ingress queue needs to occupy headroom only when it is congested and its queue length is higher than the X_{off} threshold. In reality, it is unlikely that all queues are congested at the same time [42]. Despite this, SIH conservatively allocates worst-case headroom size for all ingress queues. As a result, most headroom buffer keeps unused.

(2) Traffic heading for different ingress queues at the same port shares the capacity of uplink. In a port, all ingress queues are connected to the same uplink and thus the traffic heading for them shares the link capacity. When a traffic class of a port needs to be paused, its traffic arriving rate should be lower than C as long as other traffic classes also have in-flight packets on the uplink. Thus, the actual required headroom size can be lower than η.

(3) The upstream device is not always sending traffic at full rate. When allocating headroom buffer for an ingress queue, SIH assumes that the upstream device will always send packets at line rate before PAUSE frame takes effect. However, the upstream queue can be empty. As a result, the sending rate of upstream device can be lower than link capacity and the headroom can be over-allocated.

To quantitatively demonstrate the inefficiency, we perform a large-scale ns-3 simulation with the same settings as the previous one except that the congestion control algorithm is DCQCN [1], which has higher buffer occupancy. To examine the efficiency of headroom, we extract the local maximum values of headroom occupancy, which indicates the actual required headroom size. Fig. 6 shows the distribution of headroom utilization for a port at the local maximum point. The headroom utilization is only 4.96% at the median and 25.33% at the 99th percentile.

Note that *the inefficiency of SIH is inherent* and cannot be avoided by simply adjusting the headroom buffer size. This is because SIH has to allocate headroom buffer in the worst case to ensure that buffer never overflows, even though the worst case rarely occurs. Thus, to efficiently utilize the headroom buffer, we need to seek another headroom allocation scheme.

IV. DSH Design

To address the inefficiency problem of SIH, we propose *Dynamic and Shared Headroom (DSH)* allocation, which aims to efficiently allocate headroom while ensuring no congestion loss. In this section, we first explain the key ideas behind DSH (§IV-A). Then we present the details of our design (§IV-B), and theoretically analyze DSH's ability of burst absorption (§IV-C). Finally, we show that DSH is easy to be implemented on switching chips (§IV-D).

A. Key Ideas

DSH utilizes two ideas to achieve efficient headroom allocation while ensuring no congestion loss.

(1) DSH proactively reserves a small amount of buffer as insurance headroom to avoid packet loss. Different ingress queues in the same port share the uplink capacity. Thus, to avoid buffer overflow under any circumstances, there is no need to allocate η headroom for each ingress queue. Rather, we only need to reserve η buffer for each port. When any ingress queue in a port starts to occupy the insurance buffer, the port pauses the entire upstream port. In this way, the amount of reserved headroom is significantly reduced. Of course, this can violate performance isolation among different traffic classes. Thus, we also have the following mechanism to make the port-wise pause rarely triggered.

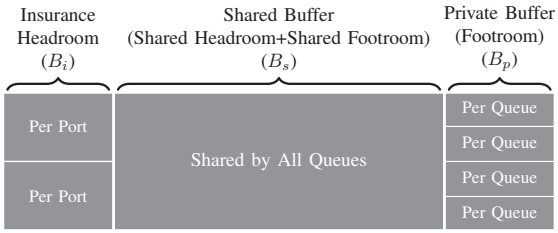


Fig. 7: Buffer partition with DSH

(2) DSH dynamically allocates the headroom to congested queues and makes the allocated headroom shared among different ingress queues. An ingress queue needs to occupy headroom only when it is congested. Thus, rather than wasting headroom for uncongested queues, DSH tries to allocate headroom to a queue only when it becomes congested. In this way, when few queues are congested, most buffer can serve as "footroom" to absorb bursty traffic without triggering PFC pauses. Furthermore, as the allocated headroom buffer is not necessarily used up, DSH enables the headroom buffer to be shared among ingress queues. In this way, DSH can take advantage of statistical multiplexing to improve the buffer efficiency.

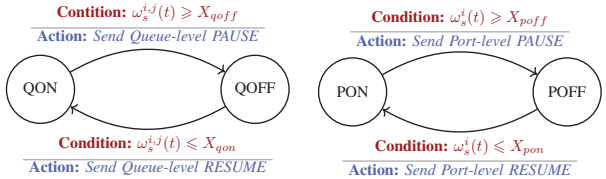
B. DSH Mechanisms

Next, we show how to realize the above key ideas with simple mechanisms.

Buffer organization. Fig. 7 shows the buffer organization with DSH. In addition to the traditional buffer partitions, DSH further divides the headroom into two parts: shared headroom and insurance headroom. The insurance headroom is statically reserved for each port to guarantee no congestion loss. The shared headroom is dynamically allocated as needed and shared among different ingress queues.

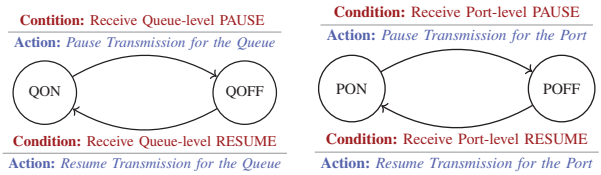
Furthermore, as both shared headroom and shared buffer are dynamically shared and allocated, DSH integrates them into a single segment, collectively called shared buffer. Such a design brings two benefits: (1) It facilitates the implementation of DSH on switching chips, as the buffer partition is the same as the existing one on commodity switching chips. (2) It improves the buffer utilization. By integrating two kinds of buffer, both headroom and footroom share the same piece of buffer, increasing the degree of statistical multiplexing. As a result, the buffer can be more efficiently utilized.

Buffer allocation and management. The management of the private buffer keeps unchanged. For insurance headroom, DSH



(a) Queue-level flow control (b) Port-level flow control

Fig. 8: State transition diagrams at ingress side (receiver)



(a) Queue-level flow control (b) Port-level flow control

Fig. 9: State transition diagrams at egress side (sender)

statically reserves some memory for each ingress port. Assume that there are N_p ports, the insurance headroom size (denoted by B_i) is given by

$$B_i = N_p \times \eta \tag{4}$$

where η is given by Eq. (1).

The remaining memory is for shared buffer, which is dynamically allocated to ingress queues. DSH uses a threshold $T(t)$ to restrict the buffer occupancy (including both shared headroom and shared footroom) for each ingress queue.

Furthermore, DSH adopts DT to dynamically adjust the threshold $T(t)$ based on the remaining buffer. DT has been widely used in commodity switching chips for decades. It can make DSH adaptive and efficient while simple to be implemented.

The calculation of threshold $T(t)$ is the same as that in existing switching chip (i.e., Eq. (2)), except that the amount of shared buffer occupancy (i.e., $\omega_s^{i,j}(t)$) should include the buffer occupancy of both shared headroom and shared footroom. Thus, DSH does not need to modify the existing hardware logic of threshold maintenance, facilitating its implementation.

Flow control. DSH has two types of flow control mechanisms to guarantee against packet loss: queue-level flow control and port-level flow control.

The *queue-level flow control* is similar to the PFC mechanism. Specifically, a PFC PAUSE frame is sent to the upstream port when the amount of shared buffer occupancy for an ingress queue becomes higher than the pause threshold (denoted by X_{qoff}). Arriving at the upstream port, the PFC PAUSE frame will suspend the packet transmission of the corresponding traffic class.

The only difference is the setting of X_{qoff} threshold. With DSH, the $X_{qoff}(t)$ threshold is set as

$$X_{qoff}(t) = T(t) - \eta \tag{5}$$

The rationale is two-fold. (1) When an ingress queue becomes congested, DSH tries to reserve enough headroom (i.e., η) for it. (2) When an ingress queue is less congested, it contributes the unused buffer to other congested queues. Specifically, the buffer occupancy of an uncongested queue is much lower than $T(t)$. In other words, it occupies less buffer and leaves much room as free buffer, which raises the value of $T(t)$ (recall that $T(t)$ is proportional to the free buffer size). Accordingly, this raises the X_{qoff} threshold and enables other congested queues to occupy more buffer before triggering PFC messages. In summary, *DSH tries to reserve enough headroom for an ingress queue if and only if it becomes congested.*

Of course, DSH cannot guarantee that every ingress queue can get η headroom when becoming congested, as the shared headroom is dynamically allocated as needed rather than statically reserved beforehand. Thus, DSH also contains port-level flow control to avoid buffer overflow under any circumstances.

The *port-level flow control* is triggered when the total occupancy of shared footroom and headroom of *all* queues in a port becomes higher than a pause threshold (denoted by X_{poff}). When triggered, the ingress port sends a port-level PAUSE frame (i.e., a PFC PAUSE frame with all pause timers of priorities set) to the upstream port. Arriving at the upstream port, the port-level PAUSE frame (i.e., a PFC PAUSE frame with all pause timers of priorities unset) will suspend the packet transmissions of *all* traffic classes.

The $X_{poff}(t)$ threshold is set as

$$X_{poff}(t) = N_q \times T(t) \tag{6}$$

The intuition is simple. DSH allocates $T(t)$ buffer as footroom and headroom for each ingress queue. Thus, for all ingress queues in a port, the total allocated buffer is $N_q \times T(t)$. The rationale behind the intuition is that DSH allows the ingress queues in the same port to share the allocated buffer, especially headroom. Specifically, by restricting the port-level buffer occupancy (rather than queue-level), a congested queue can occupy the headroom allocated to other queues (in the same port) if it has used up its allocated headroom. As the traffic heading for the ingress queues in the same port naturally shares the capacity of uplink, port-level buffer share is both efficient and fair. In this way, not only can DSH utilize the shared buffer more efficiently, but also the port-level flow control can be less triggered.

MMU workflow with DSH. Putting all together, the workflow of DSH can be described as state transition diagrams depicted in Fig. 8 (at ingress side) and Fig. 9 (at egress side).

At ingress side, there are two queue-level states for each ingress queue:

- **QON**: The ingress queue is not congested. The upstream port is allowed to send packets of the corresponding traffic class to the ingress queue. Arriving packets are put into either private buffer (if $q^{i,j} \leqslant \phi$) or shared buffer (if $q^{i,j} > \phi$).

- **QOFF**: The ingress queue is congested. The corresponding traffic class in the upstream port is being paused. Arriving packets are put into the shared buffer.

Without congestion, an ingress queue is in QON state. It turns into the QOFF state when its shared buffer occupancy becomes higher than $X_{qoff}(t)$. During the transition, a PFC PAUSE frame is sent to the upstream port to suspend the packet transmission of the corresponding traffic class. If the congestion is mitigated and the buffer occupancy of the ingress queue becomes lower than $X_{qon}(t)$, the ingress queue turns into the QON state. During the transition, a queue-level RESUME frame (i.e., PFC PAUSE frame with zero pause duration) is sent to the upstream port to recover the packet transmission of the corresponding traffic class.

Furthermore, for each ingress port, there are two port-level states:

- **PON**: The ingress port is not congested. The upstream port is allowed to send packets if the corresponding traffic class is not paused by queue-level flow control. Arriving packets are put into either private buffer (if $q^{i,j} \leqslant \phi$) or shared buffer (if $q^{i,j} > \phi$).
- **POFF**: The ingress port is congested. All traffic classes in the upstream port are being paused. Arriving packets are put into the insurance headroom.

Without congestion, the port is in PON state. If the total buffer occupancy of an ingress port becomes higher than X_{poff}, the ingress port turns into POFF state. During the transition, a port-level PAUSE frame is sent to the upstream port to suspend the packet transmission of *all* traffic classes. When the congestion of the port is mitigated and its buffer occupancy becomes lower than X_{pon}, the ingress port turns into PON State. During the transition, a port-level RESUME frame (i.e., PFC PAUSE frame with all pause timers unset) is sent to cease the previous (port-level) pause action.

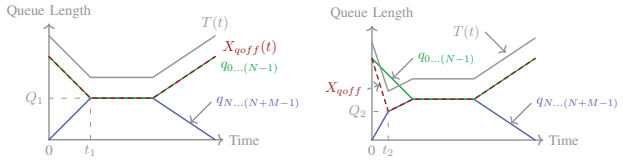
The egress-side workflow is similar except for the conditions/actions of state transitions, which is omitted due to space limitations.

C. DSH Analysis

In this part, we theoretically analyze the ability to absorb bursty traffic of DSH.

We consider the same scenario as that in [43]. Specifically, N ingress queues (i.e., queue 0, ..., queue $N-1$) become congested at time t_0 ($t_0 \ll 0$). At $t = 0$, another M empty ingress queues (i.e., queue N, ..., queue $N+M-1$) begin to transmit bursty traffic simultaneously. Traffic arriving at $N+M$ queues has an offered load of R ($R > 1$), which has normalized to the traffic departure rate. For ease of analysis, we further make the following assumptions:

1) There is no private buffer, namely $B_p = 0$.
2) The resume threshold is infinitely close to but lower than the pause threshold, namely $0 < X_{qon} - X_{qoff} < \epsilon$ for every $\epsilon > 0$.
3) The delay for a PFC PAUSE frame to take effect is infinitely close to 0.



(a) $R \leqslant \frac{1-\alpha N}{\alpha M} + 1$ (b) $R > \frac{1-\alpha N}{\alpha M} + 1$

Fig. 10: Evolutions of queue length and threshold

Under these assumptions, the ability of burst absorption for DSH and SIH can be given by Theorem 1 and Theorem 2, respectively.

Theorem 1. *For ingress queue N, ..., ingress queue $(N+M-1)$, DSH can avoid PFC PAUSEs if and only if*

$$d < \begin{cases} \frac{\alpha(B-N_p\cdot\eta)-\eta}{[1+\alpha(N+M)](R-1)}, & if\, R \leqslant \frac{1-\alpha N}{\alpha M} + 1 \\ \frac{\alpha(B-N_p\cdot\eta)-\eta}{(1+\alpha N)[(1+\alpha M)(R-1)-\alpha N]}, & if\, R > \frac{1-\alpha N}{\alpha M} + 1 \end{cases} \tag{7}$$

where d is the duration of bursty traffic.

Proof. At $t = 0$, the length of ingress queue 0, ..., ingress queue $(N-1)$ (denoted by $q_0(t)$, ..., $q_{N-1}(t)$) is the same as the X_{qoff} threshold, namely

$$q_i(0) = X_{qoff}(0) = T(0) - \eta, \quad 0 \leqslant i < N \tag{8}$$

where threshold $T(t)$ is given by

$$T(t) = \alpha \cdot (B_s - \sum_i q_i(t)) \tag{9}$$

Substituting Eq. (8) into Eq. (9) yields

$$\begin{cases} T(0) = \frac{\alpha(B_s+N\eta)}{1+\alpha N} \\ q_i(0) = T(0) - \eta = \frac{\alpha B_s - \eta}{1+\alpha N} \end{cases} \tag{10}$$

At $t = 0^+$, M ingress queues become active. Their queue lengths begin to increase and $T(t)$ begins to decrease. Accordingly, the X_{qoff} threshold will also decrease, which in turn makes $q_0(t), \ldots, q_{N-1}(t)$ decrease. Let $T'(t)$ denote the derivative of $T(t)$ and $q_i'(t)$ denote the derivative of $q(t)$, then we have

$$T'(0^+) = -\alpha \cdot \sum_i q_i'(0^+) \tag{11}$$

$$q_i'(0^+) = \begin{cases} \max(T'(0), -1), & 0 \leqslant i < N \\ R - 1, & N \leqslant i < N+M \end{cases} \tag{12}$$

Substituting Eq. (12) into Eq. (11) yields

$$T'(0^+) = -\alpha \cdot N \cdot \max(T'(0), -1) - \alpha \cdot M \cdot (R-1) \tag{13}$$

There are two cases:

1) $R \leqslant \frac{1-\alpha N}{\alpha M} + 1$

In this case, $T'(0^+) \geqslant -1$. Thus, $q_i(t)\,(0 \leqslant i < N)$ is decreasing at the same rate of X_{qoff} threshold. The queue length evolution is depicted in Fig. 10a. $q_i(t)\,(N \leqslant i < N+$

$M-1)$ will keep increasing. During this time, $T(t)$ and $q_i(t)$ can be given by

$$T(t) = \frac{\alpha}{1+\alpha N}\left[B_s + N\eta - M(R-1)t\right] \tag{14}$$

$$q_i(t) = \begin{cases} \frac{\alpha B_s - \eta - \alpha M(R-1)t}{1+\alpha N}, & 0 \leqslant i < N \\ (R-1)t, & N \leqslant i < N+M-1 \end{cases} \tag{15}$$

This situation continues until $q_i(t)\,(N \leqslant i < N+M-1)$ reaches the X_{qoff} threshold at $t = t_1$, namely $q_i(t_1) = T(t_1) - \eta$. Solving t_1 from it, we get

$$t_1 = \frac{\alpha B_s - \eta}{[1+\alpha(N+M)](R-1)} \tag{16}$$

At $t = t_1$, queue N, ...queue $(N+M-1)$ begin to pause upstream devices. Thus, DSH can avoid PFC PAUSEs if and only if $d < t_1$, where d is the duration of bursty traffic arriving at queue N, ...queue $(N+M-1)$.

2) $R > \frac{1-\alpha N}{\alpha M} + 1$

In this case, $T'(0^+) < -1$. Thus, $q_i(t)\,(0 \leqslant i < N)$ is decreasing at a rate lower than X_{qoff} threshold. The queue length evolution is depicted in Fig. 10b. During $[0, t_2]$, $T(t)$ and $q_i(t)$ can be given by

$$T(t) = \frac{\alpha(B_s + N\eta)}{1+\alpha N} - \left[\alpha M(R-1) - \alpha N\right]t \tag{17}$$

$$q_i(t) = \begin{cases} \frac{\alpha B_s - \eta}{1+\alpha N} - t, & 0 \leqslant i < N \\ (R-1)t, & N \leqslant i < N+M-1 \end{cases} \tag{18}$$

This situation continues until $q_i(t)\,(N \leqslant i < N+M-1)$ reaches the X_{qoff} threshold at $t = t_2$, namely $q_i(t_2) = T(t_2) - \eta$. Solving t_2 from it, we get

$$t_2 = \frac{\alpha B_s - \eta}{(1+\alpha N)\left[(1+\alpha M)(R-1) - \alpha N\right]} \tag{19}$$

At $t = t_2$, queue N, ...queue $(N+M-1)$ begin to pause upstream devices. Thus, DSH can avoid PFC PAUSEs if and only if $d < t_2$, where d is the duration of bursty traffic arriving at queue N, ...queue $(N+M-1)$.

□

Theorem 2. *For ingress queue N, ..., ingress queue $(N+M-1)$, SIH can avoid PFC PAUSEs if and only if*

$$d < \begin{cases} \frac{\alpha(B - N_q \cdot N_p \cdot \eta)}{[1+\alpha(N+M)](R-1)}, & \text{if } R \leqslant \frac{1-\alpha N}{\alpha M} + 1 \\ \frac{\alpha(B - N_q \cdot N_p \cdot \eta)}{(1+\alpha N)[(1+\alpha M)(R-1) - \alpha N]}, & \text{if } R > \frac{1-\alpha N}{\alpha M} + 1 \end{cases} \tag{20}$$

where d is the duration of bursty traffic.

Proof. With SIH, the queue length evolution is the same as Fig. 10, except that $X_{off} = T(t)$. Thus, we can solve t_1 and t_2 by simply let $\eta = 0$, which yields

$$\begin{cases} t_1 = \frac{\alpha B_s}{[1+\alpha(N+M)](R-1)} \\ t_2 = \frac{\alpha B_s}{(1+\alpha N)[(1+\alpha M)(R-1) - \alpha N]} \end{cases} \tag{21}$$

Thus, SIH can avoid PFC PAUSEs if and only if

$$d < \begin{cases} \frac{\alpha B_s}{[1+\alpha(N+M)](R-1)}, & \text{if } R \leqslant \frac{1-\alpha N}{\alpha M} + 1 \\ \frac{\alpha B_s}{(1+\alpha N)[(1+\alpha M)(R-1) - \alpha N]}, & \text{if } R > \frac{1-\alpha N}{\alpha M} + 1 \end{cases} \tag{22}$$

On the other hand, for SIH, $B_s = B - N_q \cdot N_q \cdot \eta$. Substituting it into Eq. (22) yields (20). □

Remarks. Theorem 1 and Theorem 2 prove that DSH has better scalability to the number of queues than SIH. Specifically, DSH's ability of burst absorption is irrespective of the number of queues per port (i.e., N_q). In comparison, SIH's ability of burst absorption is negatively related to the number of queues per port. This implies that DSH can support as many queues as possible, which can improve performance isolation between different services and enable the deployment of many advanced traffic optimization systems (e.g., PIAS [38], Homa [17]).

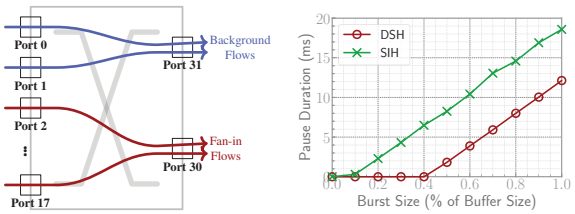
D. DSH Implementation

In this part, we discuss the feasibility of implementing DSH on switching chips. We show that DSH is very easy to be implemented, as it does not need to change the existing buffer architecture and only needs some moderate changes to existing flow control mechanisms.

Queue-level flow control. As shown in §II-B, existing PFC mechanism triggers PFC PAUSE messages when the queue length reaches $T(t)$ and PFC RESUME message when the queue length reaches $T(t) - \delta$, where δ is a configurable parameter. DSH's queue-level flow control mechanism is the same except that the threshold to trigger a PFC PAUSE message (i.e., X_{qoff}) is $T(t) - \eta$, and the threshold to trigger a PFC RESUME message (i.e., X_{qon}) is $T(t) - \eta - \delta_q$. Thus, we only need to change the conditions of generating PFC message. Specifically, the switching chip needs two extra subtractors. One is for calculating the X_{qoff} threshold, whose inputs are $T(t)$ and η. The other is for calculating the X_{qon} threshold, whose inputs are X_{qoff} and δ_q.

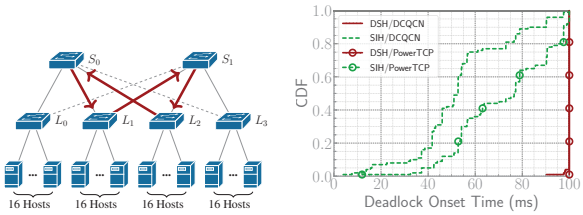
Port-level flow control. Lots of commodity switching chips have already supported port-level flow control based on port-wise buffer occupancy [29], [35]. Thus, to achieve port-level flow control, we only need to change the conditions of generating port-level PAUSE/RESUME messages. The pause threshold (i.e., X_{poff}) is $N_q \cdot T(t)$, where N_q is the number of queues per port. The switching chip needs a multiplier, whose inputs are N_q and $T(t)$, to calculate the pause threshold. In particular, if N_q is a power of two, only a shift register is needed for the calculation. The resume threshold (i.e., X_{pon}) is $X_{poff} - \delta_p$, where δ_p is a configurable parameter. The switching chip needs a subtractor, whose inputs are X_{poff} and δ_p, to calculate the resume threshold.

Consolidating two kinds of flow controls. For the downstream port, two kinds of flow control work independently, and thus no further modifications are required to consolidate them. For the upstream port, we only need to change the condition



(a) Scenario (b) Total pause duration

Fig. 11: PFC avoidance



(a) Scenario (b) CDF of deadlock onset time

Fig. 12: Deadlock avoidance

of suspending packet transmissions. Specifically, each egress queue suspends packet transmissions if the egress queue is in QOFF state *or* the egress port is in POFF state. To realize this, the switching chip needs to maintain a queue-level state per egress queue and a port-level state per egress port as Fig. 9 shows. Then the pause condition can be easily generated by an OR gate.

Overall resource increments to the switch ASIC. Based on the above analysis, we argue that DSH brings acceptable resource increments to switch ASIC due to the following reasons. (1) DSH does not require additional counters/registers. DSH only requires the buffer occupancy of each queue/port, which is already available in existing switch ASIC. (2) DSH does not touch the memory allocation and management mechanisms. DSH's physical memory architecture (Fig. 7) is the same as the existing one in commodity switches (Fig. 2). Furthermore, DSH does not modify the the memory allocation and management mechanisms of footroom buffer, while the allocation of headroom buffer can be realized by additional flow control mechanisms. Thus, implementing DSH does not need to modify the existing memory allocation and management mechanisms. (3) Simple and cheap comparison/arithmetic operations are required to realize the conditions of triggering PFC messages. DSH can be implemented by modifying the conditions of triggering PFC messages, while all conditions are based on comparison between buffer occupancy (and its derivatives) and thresholds. Thus, DSH only requires comparison and simple arithmetic operations (i.e., subtraction).

V. Evaluation

In this section, we evaluate DSH's performance with extensive packet-level simulations on the ns-3 platform [44]. We summarize the results below.

- DSH can absorb 4× more bursty traffic without triggering PFC messages.
- DSH can effectively mitigate the performance impairments (i.e., collateral damage and deadlock) brought by PFC messages.
- DSH can reduce the FCT by up to 57.7% for short fan-in flows and up to 31.1% for one-to-one background flows in large-scale DCN topology.

A. Microbenchmarks

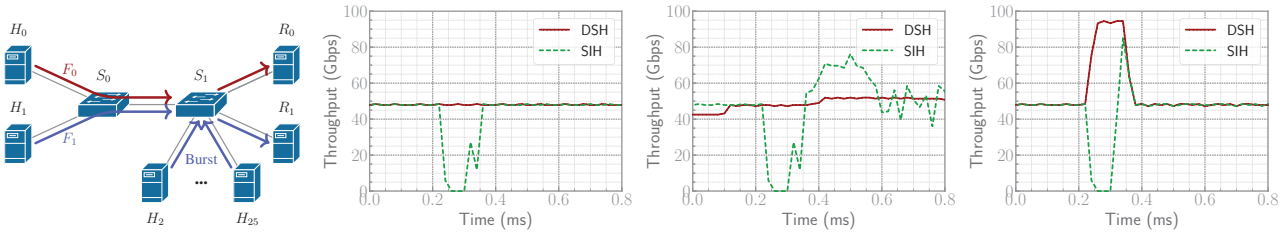
In this part, we evaluate DSH's basic performance with carefully constructed microbenchmarks. We emulate the Broadcom Tomahawk switching chip, which has 32 100Gbps ports and 16MB shared memory. Each port has 8 queues. One queue is reserved for ACK packets and PAUSE/RESUME messages, and has the highest priority. Other seven queues are scheduled by the DWRR algorithm with a quantum of 1600B. The link delay is 2μs and thus $\eta = 56840$B. The total headroom size for SIH is $56840\text{B} \times 32 \times 7 = 12$MB. The private buffer size is 672KB (3KB for each DWRR-scheduled queue). For DT, α is set to 1/16 according to [3]. The X_{qon}/X_{pon} threshold is the same as the X_{qoff}/X_{poff} threshold.

PFC avoidance. First, we evaluate the ability of DSH to avoid PFC messages. As shown in Fig. 11a, we start two background flows at the beginning, which are from ingress port 0 and ingress port 1, respectively, and head for egress port 31. At $t = 0.1$s, we start 16 fan-in bursty flows, which are from ingress port 2-17 to egress port 30. Fig. 11b shows the total pause duration of all fan-in flows. DSH can avoid PAUSE messages when the burst size is no larger than 40% of buffer size, which is over 4× better than SIH.

Deadlock avoidance. One impairment brought by PFC messages is deadlock [3], [6]–[8], [11], which is a serious problem as it can make a large part of network unusable. In this part, we evaluate the ability of DSH to avoid deadlock.

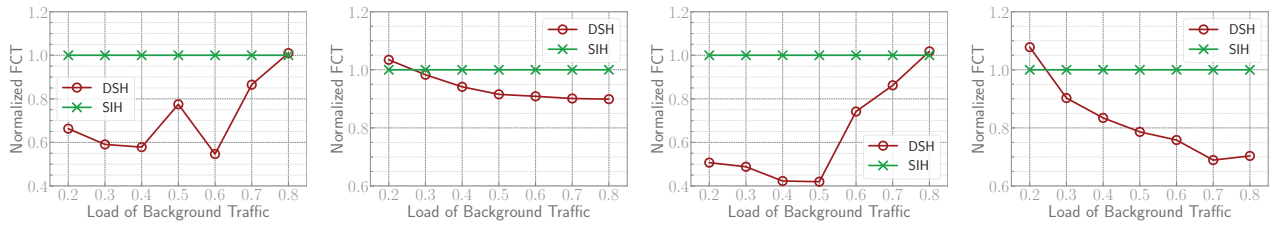
We consider a topology shown in Fig. 12a, which is a leaf-spine topology with two link failures marked with dashed lines (i.e., S_0-L_3 and S_1-L_0). In the topology, there are two spine switches (S_0 and S_1) and four leaf switches ($L_0 - L_3$). Each leaf switch is connected to 16 hosts with 100Gbps downlinks, and connected to two spine switches with 400Gbps uplinks. The link delay is 2μs. We generate fan-in flows, which are from L_0 to L_3, from L_3 to L_0, from L_1 to L_2, and from L_2 to L_1, respectively. As a result, there is a cyclic buffer dependency marked as red lines: $S_0 \rightarrow L_1 \rightarrow S_1 \rightarrow L_2 \rightarrow S_0$. The fan-in ratio (i.e., the number of flows) ranges from 1 to 15. The flow size is randomly chosen based on the Hadoop workload [28], and flow arrivals follow a Poisson process. The network load is 0.5 at the downlinks of each leaf switch. Each scheme is tested 100 times and each simulation lasts for 100ms.

Fig. 12b shows the CDF of deadlock onset time. With SIH, deadlock occurs for all simulations either with DCQCN [1] or PowerTCP [37]. In comparison, DSH can avoid 96% deadlocks with DCQCN and all deadlocks with PowerTCP. This is because DSH can leave more "footroom" to absorb



(a) Scenario (b) Throughput of F_0 (w/o CC) (c) Throughput of F_0 (DCQCN) (d) Throughput of F_0 (PowerTCP)

Fig. 13: Mitigation of collateral damage



(a) [DCQCN] Fan-in traffic (b) [DCQCN] Background traffic (c) [PowerTCP] Fan-in traffic (d) [PowerTCP] Background traffic

Fig. 14: Average FCTs with different loads of background traffic

bursty traffic, avoiding PFC messages.

Collateral damage mitigation. Another impairment brought by PFC messages is that it can lead to performance degradation of innocent flows, which is known as collateral damage [3].

In this part, we show that DSH can mitigate this problem by avoiding PFC PAUSEs. We consider a typical scenario shown in Fig. 13a, which is a common unit in datacenters and widely adopted by other work [9], [45], [46]. All links are 100Gbps and the propagation delay is 2μs. Two long-lived flows, F_0 and F_1, are sending traffic from H_0 and H_1 to R_0 and R_1, respectively. After their throughputs reach 50Gbps, H_2-H_{25} generate 24 concurrent fan-in flows to R_1. Each flow has a size of 64KB, which is smaller than a BDP and thus the fan-in traffic is uncontrolled by congestion control algorithms. At this time, the congestion point is S_1 and both F_1 and fan-in flows contribute to the congestion. On the other hand, F_0 is an innocent flow that does not contribute to the congestion. Ideally, the throughput of F_0 should not be reduced.

Fig. 13 shows the throughput of F_0. With SIH, the throughput of F_0 is significantly reduced. This is because the pause threshold X_{poff} is very low and thus PFC PAUSEs can be easily triggered. As a result, switch S_0 is paused and the packet transmission of F_0 is suspended. In comparison, DSH can effectively avoid performance degradation for F_0. This is because it can efficiently utilize the headroom and leave enough "footroom" for PFC avoidance. Besides, we observe that the state-of-the-art congestion control algorithms (Fig. 13c and Fig. 13d) are not able to avoid the collateral damage. This is because end-to-end congestion control requires at least 1 RTT to react to traffic changes. Within 1 RTT, it is the buffer management scheme that determines whether PFC messages can be avoided.

B. Benchmark Traffic

In this part, we evaluate DSH in a large-scale DCN topology.

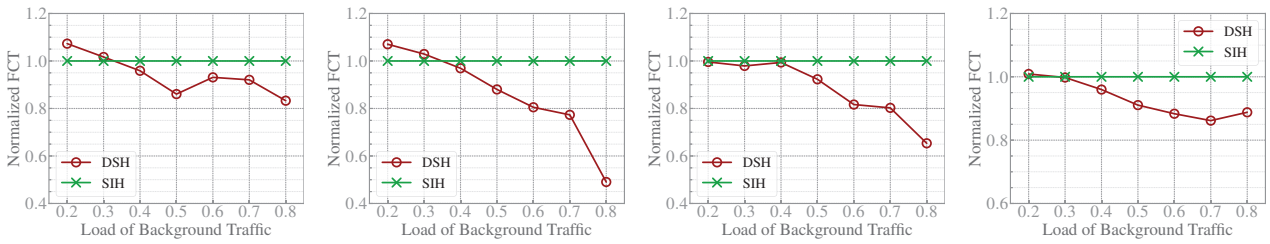
Topology. We build a leaf-spine topology with 16 leaf switches, 16 spine switches, and 256 servers. Each leaf switch is connected to 16 servers with 100Gbps downlinks and 16 spine switches with 100Gbps uplinks, forming a full-bisection network. The link delay is 2μs and thus the base RTT is 16μs across the spine. We employ ECMP for load balancing.

Switch. We emulate the Broadcom Tomahawk switching chip. The settings are the same as those in previous simulations.

Transport. We consider two end-to-end congestion control algorithms: DCQCN [1] and PowerTCP [37]. We use the default parameter settings in their open-source simulations.

Workload. We generate two kinds of traffic: background traffic and bursty fan-in traffic. The background traffic follows one-to-one pattern. The sender and receiver are randomly chosen. The flow sizes are generated according to a web search workload [27]. The fan-in traffic follows many-to-one pattern, where 16 senders simultaneously transmit 64KB data to the same receiver. The senders are randomly chosen and are in different racks from the receiver. Fan-in flows are classified into the same traffic class. Background flows are randomly classified into other traffic classes. Flow arrivals follow a Poisson process. The total network load is 0.9.

Results. Fig. 14 shows the flow completion time (FCT) under different loads of background traffic. For clear comparison, we normalize each FCT to the value achieved by SIH. The results show that both fan-in flows and background flows can benefit from DSH. With DCQCN, DSH can reduce the average FCTs of background traffic and fan-in traffic by up to 10.1% and 43.3%, respectively. With PowerTCP, DSH can reduce the



(a) Leaf-Spine + Data mining (b) Leaf-Spine + Cache (c) Leaf-Spine + Hadoop (d) Fat-Tree (k=16) + Web Search

Fig. 15: Average FCTs across different workloads and topologies (transport: DCQCN)

average FCTs of background traffic and fan-in traffic by up to 31.1% and 57.7%, respectively. This is because DSH can provide more footroom to absorb transient bursty traffic and avoid PFC messages.

Furthermore, we evaluate DSH across different DCN applications and network architectures. Besides the web search workload, we also consider other three realistic workloads: a data mining workload [47], a cache workload [28], and a Hadoop workload [28]. Besides leaf-spine topology, we also consider fat-tree topology (k=16) [48]. Other settings keep unchanged. Fig. 15 shows the FCT of background traffic across different workloads and topologies with DCQCN. The results confirm that DSH can improve FCT with different DCN workloads and network topologies.

VI. Related Work

To reduce PFC messages, lots of end-to-end congestion control mechanisms [1], [2], [4], [6], [9], [37], [46], [49]–[52] are proposed in recent years. They aim to maintain persistent queue length low, which avoids triggering PFC messages. However, it usually requires 1 RTT for the sources to receive the congestion feedback. Within 1 RTT, it is the switch's local mechanisms (e.g., buffer management and flow control) that determine whether PFC messages are generated. Thus, DSH is complementary to them.

Besides the congestion control mechanisms, lots of studies focus on mitigating the impairments of PFC in other ways. PLB [12] leverages load balancing to alleviate PFC's head-of-line problem. Several approaches (i.e., TCP-Bolt [6], Tagger [7], GFC [8], ITSY [11]) focus on detecting, avoiding, and recovering from PFC deadlocks. To reduce the queue buildup caused by incast traffic, P-PFC [10] uses the derivative of buffer change to predict the buffer occupancy, and proactively generates PFC messages to avoid queue buildup. In comparison, DSH focuses on efficient headroom allocation and thus is orthogonal to them.

VII. Conclusion

In datacenter networks, PFC-enabled switches need to reserve some buffer as *headroom* to avoid buffer overflow. However, with the growing link speed, the buffer becomes increasingly inadequate, and the headroom occupies a considerable fraction of buffer, significantly squeezing the buffer space for accommodating normal traffic. As a result, PFC messages can be frequently generated, bringing about serious performance impairments. In this paper, we argue that the current static and queue-independent headroom allocation scheme is inherently inefficient. We propose Dynamic and Shared Headroom (DSH) allocation scheme, which dynamically allocates headroom to congested queues and enables allocated headroom to be shared among different queues. Extensive simulations show that DSH can significantly reduce the PFC messages and improve the network performance.

Acknowledgment

We thank the anonymous reviewers for their constructive comments. This work was supported in part by the National Key R&D Program of China (No. 2022YFB2901700), in part by the National Natural Science Foundation of China (No. 62002165, No. 62172323, and No. 62272382), and in part by the Natural Science Foundation of Jiangsu Province (No. BK20200445).

References

[1] Y. Zhu, H. Eran, D. Firestone, *et al.*, "Congestion Control for Large-Scale RDMA Deployments," in *ACM SIGCOMM*, 2015.

[2] R. Mittal, V. T. Lam, N. Dukkipati, *et al.*, "TIMELY: RTT-based Congestion Control for the Datacenter," in *ACM SIGCOMM*, 2015.

[3] C. Guo, H. Wu, Z. Deng, *et al.*, "RDMA over Commodity Ethernet at Scale," in *ACM SIGCOMM*, 2016.

[4] Y. Li, R. Miao, H. H. Liu, *et al.*, "HPCC: High Precision Congestion Control," in *ACM SIGCOMM*, 2019.

[5] Z. He, D. Wang, B. Fu, *et al.*, "MasQ: RDMA for Virtual Private Cloud," in *ACM SIGCOMM*, 2020.

[6] B. Stephens, A. L. Cox, A. Singla, J. Carter, C. Dixon, and W. Felter, "Practical DCB for Improved Data Center Networks," in *IEEE INFOCOM*, 2014.

[7] S. Hu, Y. Zhu, P. Cheng, *et al.*, "Tagger: Practical PFC Deadlock Prevention in Data Center Networks," in *ACM CoNEXT*, 2017.

[8] K. Qian, W. Cheng, T. Zhang, and F. Ren, "Gentle Flow Control: Avoiding Deadlock in Lossless Networks," in *ACM SIGCOMM*, 2019.

[9] W. Cheng, K. Qian, W. Jiang, T. Zhang, and F. Ren, "Re-architecting Congestion Management in Lossless Ethernet," in *USENIX NSDI*, 2020.

[10] C. Tian, B. Li, L. Qin, *et al.*, "P-PFC: Reducing Tail Latency with Predictive PFC in Lossless Data Center Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1447–1459, 2020.

[11] X. C. Wu and T. S. Eugene Ng, "Detecting and Resolving PFC Deadlocks with ITSY Entirely in the Data Plane," in *IEEE INFOCOM*, 2022.

[12] J. Hu, C. Zeng, Z. Wang, H. Xu, J. Huang, and K. Chen, "Load Balancing in PFC-Enabled Datacenter Networks," in *APNet*, 2022.

[13] A. Singh, J. Ong, A. Agarwal, *et al.*, "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network," in *ACM SIGCOMM*, 2015.

[14] W. Bai, S. Hu, K. Chen, K. Tan, and Y. Xiong, "One More Config is Enough: Saving (DC)TCP for High-speed Extremely Shallow-buffered Datacenters," in *IEEE INFOCOM*, 2020.

[15] G. Zeng, J. Qiu, Y. Yuan, H. Liu, and K. Chen, "FlashPass: Proactive Congestion Control for Shallow-buffered WAN," in *IEEE ICNP*, 2021.

[16] P. Goyal, P. Shah, K. Zhao, G. Nikolaidis, M. Alizadeh, and T. E. Anderson, "Backpressure Flow Control," in *USENIX NSDI*, 2022.

[17] B. Montazeri, Y. Li, M. Alizadeh, and J. Ousterhout, "Homa: A Receiver-driven Low-latency Transport Protocol Using Network Priorities," in *ACM SIGCOMM*, 2018.

[18] S. Hu, W. Bai, G. Zeng, *et al.*, "Aeolus: A Building Block for Proactive Transport in Datacenters," in *ACM SIGCOMM*, 2020.

[19] Q. Zhang, V. Liu, H. Zeng, and A. Krishnamurthy, "High-resolution Measurement of Data Center Microbursts," in *ACM IMC*, 2017.

[20] IEEE DCB. "802.1Qbb – Priority-based Flow Control." (Jul. 16, 2011), [Online]. Available: https://1.ieee802.org/dcb/802-1qbb.

[21] S. Das and R. Sankar, "Broadcom Smart-Buffer Technology in Data Center Switches for Cost-Effective Performance Scaling of Cloud Applications," Broadcom, White Paper, Apr. 2012.

[22] "Congestion Management and Buffering in Data Center Networks," Extreme Networks, White Paper, 2014. [Online]. Available: http://learn.extremenetworks.com/rs/extreme/images/Congestion-Management-and-Buffering-wp.pdf.

[23] A. Arcilla and T. Palmer, "Broadcom Trident 3 Platform Performance Analysis," Broadcom, White Paper, May 2019. [Online]. Available: https://docs.broadcom.com/doc/12395356.

[24] B. Wheeler, "Tomahawk 4 Switch First to 25.6Tbps, Broadcom Doubles 400Gbps Ports With Unprecedented 512 Serdes," The Linley Group, Microprocessor Report, Dec. 2019. [Online]. Available: https://www.linleygroup.com/mpr/article.php?id=12237.

[25] "Cisco Nexus 9300 Platform Buffer and Queuing Architecture," Cisco, White Paper, Nov. 2014. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/white-paper-c11-732452.pdf.

[26] "Arista 7050X3 Series Switch Architecture," Arista, White Paper, Oct. 2013. [Online]. Available: https://www.arista.com/assets/data/pdf/Whitepapers/7050X3_Architecture_WP.pdf.

[27] M. Alizadeh, A. Greenberg, D. A. Maltz, *et al.*, "Data Center TCP (DCTCP)," in *ACM SIGCOMM*, 2010.

[28] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the Social Network's (Datacenter) Network," in *ACM SIGCOMM*, 2015.

[29] *BCM88800 Traffic Management Architecture*, Design Guide, Broadcom, Feb. 2021. [Online]. Available: https://docs.broadcom.com/doc/88800-DG1-PUB.

[30] "How To Configure Mellanox Spectrum Switch for Lossless RoCE," Mellanox. (Dec. 2018), [Online]. Available: https://support.mellanox.com/s/article/howto-configure-mellanox-spectrum-switch-for-lossless-roce.

[31] "Cisco Nexus 9300-EX Platform Switches Architecture," Cisco, White Paper, 2017. [Online]. Available: https://www.cisco.com/c/dam/global/nl_nl/solutions/data-center-virtualization/pdfs/Cisco_Nexus_9300_EX_Platform_Switches_white_paper_NL.pdf.

[32] "Priority Flow Control: Build Reliable Layer 2 Infrastructure," Cisco, White Paper, Sep. 2015. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/switches/nexus-7000-series-switches/white_paper_c11-542809.pdf.

[33] IEEE DCB. "Proposal for Priority Based Flow Control." (May 8, 2008), [Online]. Available: https://www.ieee802.org/1/files/public/docs2008/bb-pelissier-pfc-proposal-0508.pdf.

[34] "Understanding the Alpha Parameter in the Buffer Configuration of Mellanox Spectrum Switches," Mellanox. (Dec. 2018), [Online]. Available: https://support.mellanox.com/s/article/howto-configure-mellanox-spectrum-switch-for-lossless-roce.

[35] *Cisco Nexus 9000 Series NX-OS Quality of Service Configuration Guide, Release 6.x*, Cisco, Apr. 22, 2020. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/6-x/qos/configuration/guide/b_Cisco_Nexus_9000_Series_NX-OS_Quality_of_Service_Configuration_Guide.pdf.

[36] Y. He, N. Batta, and I. Gashinsky, "Understanding Switch Buffer Utilization in CLOS Data Center Fabric," in *Workshop on Buffer Sizing*, 2019.

[37] V. Addanki, O. Michel, and S. Schmid, "PowerTCP: Pushing the Performance Limits of Datacenter Networks," in *USENIX NSDI*, 2022.

[38] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang, "Information-Agnostic Flow Scheduling for Commodity Data Centers," in *USENIX NSDI*, 2015.

[39] M. P. Grosvenor, M. Schwarzkopf, I. Gog, *et al.*, "Queues Don't Matter When You Can JUMP Them!" In *USENIX NSDI*, 2015.

[40] N. K. Sharma, M. Liu, K. Atreya, and A. Krishnamurthy, "Approximating Fair Queueing on Reconfigurable Switches," in *USENIX NSDI*, 2018.

[41] L. Chen, J. Lingys, K. Chen, and F. Liu, "AuTO: Scaling Deep Reinforcement Learning for Datacenter-Scale Automatic Traffic Optimization," in *ACM SIGCOMM*, 2018.

[42] W. Bai, S. S. Abdeen, A. Agrawal, *et al.*, "Empowering Azure Storage with RDMA," in *USENIX NSDI*, 2023.

[43] A. K. Choudhury and E. L. Hahne, "Dynamic Queue Length Thresholds For Shared-memory Packet Switches," *IEEE/ACM Transactions on Networking*, vol. 6, no. 2, pp. 130–140, 1998.

[44] "ns-3." (), [Online]. Available: https://www.nsnam.org/.

[45] M. Handley, C. Raiciu, A. Agache, *et al.*, "Re-architecting Datacenter Networks and Stacks for Low Latency and High Performance," in *ACM SIGCOMM*, 2017.

[46] Y. Zhang, Y. Liu, Q. Meng, and F. Ren, "Congestion Detection in Lossless Networks," in *ACM SIGCOMM*, 2021.

[47] A. Greenberg, J. R. Hamilton, N. Jain, *et al.*, "VL2: A Scalable and Flexible Data Center Network," in *ACM SIGCOMM*, 2009.

[48] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," in *ACM SIGCOMM*, 2008.

[49] P. Taheri, D. Menikkumbura, E. Vanini, S. Fahmy, P. Eugster, and T. Edsall, "RoCC: Robust Congestion Control for RDMA," in *ACM CoNEXT*, 2020.

[50] J. Zhang, J. Shi, X. Zhong, *et al.*, "Receiver-Driven RDMA Congestion Control by Differentiating Congestion Types in Datacenter Networks," in *IEEE ICNP*, 2021.

[51] J. Zhang, Y. Zhang, Z. Guan, *et al.*, "HierCC: Hierarchical RDMA Congestion Control," in *APNet*, 2022.

[52] X. Zhong, J. Zhang, Y. Zhang, Z. Guan, and Z. Wan, "PACC: Proactive and Accurate Congestion Feedback for RDMA Congestion Control," in *IEEE INFOCOM*, 2022.