L2BM: Switch Buffer Management for Hybrid Traffic in Data Center Networks

Yi Liu*, Jiangping Han*†, Kaiping Xue*†, Ruidong Li‡, Jian Li*
*School of Cyber Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027 China
‡National Institute of Information and Communications Technology, Kanazawa University, Tokyo 184-0015, Japan
†Corresponding author: J. Han (jphan@ustc.edu.cn), K. Xue (kpxue@ustc.edu.cn)

2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS) | 979-8-3503-3986-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDCS57875.2023.00076

***Abstract*—With Remote Direct Memory Access (RDMA) extended to commercial Ethernet, modern Data Center Networks (DCNs) carry both traditional TCP and RDMA, to support diversified application requirements. RDMA flows are guaranteed lossless transmission through Priority-based Flow Control (PFC), while TCP flows are generally lossy traffic with packet loss. However, TCP is prone to excessively occupy the shared buffer, frequently triggering PFC pause frames and overflows at switches, damaging the performance of RDMA, which expose the vulnerability of existing buffer management policies. In this paper, we propose L2BM, a buffer management algorithm for shared-memory switches to support dynamic hybrid traffic. L2BM utilizes the average occupying time of packets in each ingress queues, to perceive the congestion states timely at ingress ports, allocating the ingress pool fairly and flexibly. Based on the perception, L2BM allocates more buffer for ingress queues with faster drain and lower congestion degrees to absorb micro-burst and reduce pause frames, less buffer for long-occupied queues to prevent excessive injection. As a result, L2BM achieves low tail latency, high burst traffic absorption capacity and low buffer occupancy. Evaluations show that L2BM enable to cut the tail latency of RDMA traffic by 50% at high workloads, reduce the buffer occupancy by 40% and decrease average query delay by 57%, while ensuring few PFC pause frames and maintaining good performance of TCP flows.**

***Index Terms*—DCN, TCP, RDMA, buffer management, PFC**

I. Introduction

In recent years, with the rapid development of high-performance computing and online services, Data Center Networks (DCNs) have increasingly high requirements for high bandwidth and low latency. In large-scale commercial DCNs, the link bandwidth has reached 100Gbps, while switches are gradually tending to shallow buffers to reduce latency [1]. In addition to queuing delay at switches, high CPU overhead and processing latency cause traditional TCP fails to meet services highly sensitive to delay and packet loss. Therefore, the deployment of such services usually adopts Remote Direct Memory Access (RDMA) technology, to support some new generation DCN workloads [2]. RDMA directly transmits data to storage space of target receivers, thus rapidly completing the exchange of massages between the memory of computing nodes without the participation of kernels, greatly reducing the processing delay at end hosts. In order to adapt to Ethernet switches and network interface cards in DCNs, RDMA over commodity Ethernet (RoCEv2) [3] is designed and extended to the IP layer. Meanwhile, many traditional applications require steady throughput and high reliability. Therefore, applications in modern DCNs are extensive, with complex and diverse transmission requirements [4], [5].

To meet diverse application demands, RDMA is often used for distributed high-performance applications like artificial intelligence, scientific computing [3], while TCP and its related extensions [6]–[8] are still dominant for communication between some traditional applications and inter data centers. Unfortunately, distributed high-performance applications are characterized by many-to-one communication pattern, resulting in Incast traffic [9], [10]. Under Incast traffic, synchronized bursts exceed the egress port capacity of switches, causing queue to build up and may lead to congestive packet loss, which is a great blow to delay-sensitive applications. To cope with packet loss and mitigate timeout, TCP (lossy traffic) utilizes fast retransmission and fast recovery mechanism, while RDMA (lossless traffic) lacks a complete packet loss protection mechanism. To ensure lossless transmission of RDMA in Ethernet, switches are configured with Priority-based Flow Control (PFC) mechanism. Once an ingress port detects that its buffer occupancy exceeds a specified threshold, it prevents upstream ports from sending packets through specific PFC frames. TCP and RDMA flows have different requirements for switch buffer, but they inevitably share resource on switches while coexisting. When the switch buffer is limited, the coexisted hybrid traffic will encroach resource and affect each other.

The key technologies on switches to realize that TCP and RDMA share buffer resources are mainly summarized as two parts: 1) Switch port queues are divided into lossy queues and lossless queues. TCP flows and RDMA flows are isolated to different queues through flow priority with bandwidth reserved. RDMA traffic enters lossless queues through Differentiated Services Code Point (DSCP) mark to reduce the interaction between two traffic classes; 2) In order to make better use of shallow buffer, Dynamic Threshold (DT) algorithm is adopted at both ingress and egress ports. Each queue is allocated memory from a shared buffer pool. The shared buffer threshold of each port and each traffic class is proportional to the size of unallocated shared buffer size controlled by a specific factor. If the occupancy of one queue exceeds the threshold, PFC or packet loss will be triggered.

However, different transmission requirements and PFC mechanism bring following problems and challenges for

2575-8411/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDCS57875.2023.00076

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on December 01,2023 at 04:01:02 UTC from IEEE Xplore. Restrictions apply.

hybrid traffic to share memory:

- **Congestion degree of packets is unaware for ingress queues**. Different from the First In First Out (FIFO) rules of egress queues, ingress queues cannot sense the congestion degree of packets in output queues. Some queues can be discharged quickly though there are more packets, while others queues are occupied for longer time as related output queues are congested. Using traditional DT [11] to allocate buffer or PFC threshold may result in low buffer utilization, as packets are still lost or PFC is triggered though enough space is remained. Packet loss of concurrent flows will cause tail latency. And frequent PFC pause frames inevitably damage the performance of RDMA, leading to problems such as head-of-the-line blocking, deadlock, unfairness and PFC storms [3], [12]–[14].
- **Excessive occupation of lossy traffic**. Lossy traffic like TCP often sends messages according to the window and has longer congestion control loop, leading to excessive buffer occupancy and unfair resource allocation, which may cause packet loss and even starve lossless traffic like RDMA flows.

To solve problems above caused by the hybrid protocol traffic sharing underlying network, existing solutions focus on optimizing the division of egress buffer pool to improve buffer occupancy, such as traditional DT algorithm and other optimization algorithms based on the DT [15]–[18]. In particular, ABM [19] can isolate traffic of different protocols with priority queues at switches and performs well, but only supports lossy traffic like TCP. Unfortunately, all of these schemes do not take PFC thresholds at ingress and resource allocation of lossless traffic into consideration.

In this paper, based on traffic patterns, congestion features and buffer occupation of TCP and RDMA, we propose L2BM, a dynamic algorithm for hybrid buffer management in shared-memory switches. L2BM utilizes the occupying time of packets in ingress pools to weigh the congestion states of queues, so as to allocate buffer threshold flexibly for flow control. Specifically, we design a packet sojourn time module to record the average occupying time of packets in each ingress queue. Through this module, not blindly determining PFC thresholds for ingress queues with consistent control factors any more, L2BM can predict the emptying rate of ingress queues and adjust the control factor of DT timely with a weight inversely proportional to the occupying time, so as to adapt to traffic patterns with different congestion levels and sojourn time. As a result, L2BM promises larger buffer (with high PFC threshold) to queues emptied faster, by setting a larger weight to prevent PFC pause frames and absorb bursts, while allocating smaller weights to queues occupied longer, avoiding excessive injection and occupancy. While improving burst absorption of switches, L2BM can alleviate the interference between lossless traffic and lossy traffic by limiting buffer occupancy and decreasing pause frames.

In summary, we make the following contributions:

- We reveal the problem that lossy traffic like TCP in hybrid protocol DCNs increases the latency of RDMA traffic and triggers frequent PFC pause frames in shallow-buffer switches. And we analyze the reasons including that different protocols are not distinguished and the emptying rate of ingress queues cannot be perceived.
- We design a buffer management algorithm L2BM, to protect lossless RDMA traffic at Ethernet switches in hybrid DCNs. Innovatively, L2BM utilizes the average occupying time of packets in ingress queues to perceive congestion states of hybrid protocols, adjusting the PFC threshold allocation factor in real time.
- We implement a sojourn time recording module in switch Memory Management Unit (MMU) with C++. Through large-scale ns-3 simulation, we verify that L2BM decreases the tail latency of RDMA by 50%, reduces the buffer occupancy by 40% and decreases average query delay by 57%, while ensuring few PFC pause frames and maintaining good performance of TCP flows.

The rest of this paper is structured as follows. Section II outlines the background knowledge and explains the motivation of our work. The proposed L2BM is detailed in Section III. Thereafter, the performance evaluation and analysis are shown in Section IV. Then Section V briefly survey the related works and conclusions are drawn in Section VI.

II. Problem Analysis

In this section, we first introduce shared memory switches for lossless traffic, including buffer management, PFC mechanism and traffic isolation. Then we discuss drawbacks of existing buffer management policies for hybrid traffic through analysis and simulation experiments.

A. Shared buffer management in DCNs

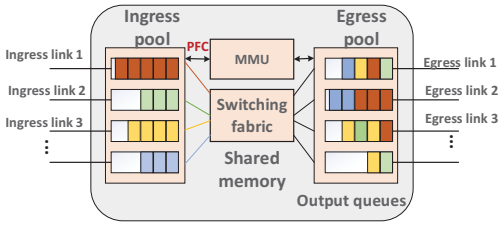


Fig. 1: The architecture of a shared-memory switch.

Shared memory switch architecture [20]: Fig. 1 shows the framework of a output-queued shared-memory switch. Packets arriving at different ingress ports are sent and stored in egress queues in negligible time, assuming that the switch fabric can process incoming packets with routing algorithms fast enough. For high buffer utilization, each queue in all egress ports are assigned a static reserved buffer to ensure the basic forwarding function, and shares the public memory to tolerate burst traffic on the basis of specific buffer allocation policies in Memory Management Unit (MMU). In order to implement the PFC

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on December 01,2023 at 04:01:02 UTC from IEEE Xplore. Restrictions apply.

mechanism at ingress, unlike traditional switches, the MMU of switches designed for lossless networks needs to maintain both the ingress and egress pool allocation, separately for PFC and egress queue share.

Isolation of lossless traffic and lossy traffic: TCP responds to congestion at switches much slower than RDMA owing to its long Round Trip Time (RTT) (TCP $25.4\mu s$ v.s. RDMA $1.7\mu s$) [21]. Assuming that lossy packets and lossless packets queue in the same physical queue, RDMA end nodes can quickly perceive the queue accumulation and reduce sending rates, while TCP is still sending packets in a large window, which will cause RDMA traffic starvation. Therefore, lossy traffic and lossless traffic are isolated in different priority queues for separate congestion control. For example, two priorities are for RDMA while others for TCP. Accordingly, egress ports schedule 8 priority queue packets through Round Robin algorithm, reserving link bandwidth for diverse queues.

Switch buffer configuration for hybrid traffic: Attention, packets are actually written in the central memory and queue at egress queues, while egress pools and ingress pools can be regarded as virtual counters, used to set the thresholds of ingress queue and egress queue respectively. Each arriving packet can be enqueued only if it is admitted by both the ingress pool and related egress pool at the same time, otherwise dropped. Meanwhile, this packet is repeatedly recorded by the ingress pool queue counter and egress pool queue counter. After the packet depart from the switch, corresponding byte size is removed from both pools simultaneously.

PFC is a hop-by-hop two-layer priority flow control mechanism, which creates eight virtual channels on an Ethernet link. Each virtual channel is assigned a corresponding priority of IEEE 802.1P, permitting the suspension and restart of any virtual channel separately, so that flows of other virtual channels can pass through without interruption. PFC of ingress ports is triggered by the occupancy of ingress buffer. Ingress pool adopts dynamic buffer sharing. The maximum buffer space for the priority of each port is controlled by a configurable parameter. The buffer size allocated for traffic priority p of ingress port i is B_i^p:

$$B_i^p(t) = \alpha \times \left[B - \sum_{i=1}^{N} \sum_{p=1}^{8} Q_i^p(t) \right], \tag{1}$$

where α is the configurable control parameter default set to $\frac{1}{16}$, B is the total buffer size, $Q_i^p(t)$ is the buffer occupancy size of ingress port i and priority p at time t, and N is the number of active ports. Each ingress port uses the reserved memory first. When the reserved buffer is used up, shared buffer starts to come in handy.

Once PFC threshold B_i^p is exceeded, for lossless traffic, the port generates a pause frame to the given upstream port/priority, and there is a headroom reserved to absorb on-flight packets on the link before the pause frame reaches upstream ports and takes effect. The size of the headroom is decided by the Maximum Transmit Unit (MTU) size, PFC reaction time of the egress port, and the propagation delay between the sender and the receiver. After the ingress queue length gradually drop to another low threshold, a resume frame is sent to recover the transmission. In particular, for lossy traffic, packets out of thresholds are just dropped at the ingress. Fig. 2 shows how PFC works.

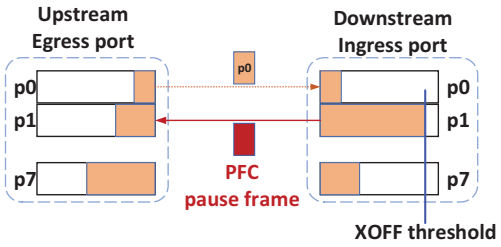


Fig. 2: How PFC works.

Similarly, the dynamic threshold algorithm is also adopted at egress ports to divide buffer size for different priority queues, and all exceeding packets are discarded here. To prevent lossless traffic starvation, two independent buffer pools are used at the egress to manage lossless and lossy traffic respectively. In recent years, many buffer management strategies based on switch egress queues have been proposed, like EDT [16], TDT [17], ABM [19].

On the whole, buffer space for lossless traffic is divided into three parts: static buffer, service pool and headroom pool.

- Static buffer: a buffer size separately divided from the buffer on the chip, which is used to divide the port guaranteed buffer. The static buffer shared by a port is exclusive and cannot be used by other ports even when it is empty.
- Service pool: a shared memory for all ports, divided by specific buffer management policies.
- Headroom pool: reserved for on-flight packets while triggering PFC pause frames for lossless traffic. Buffer space for lossy traffic has no headroom pool.

B. Drawbacks for hybrid traffic

The differences between these two traffic classes in hybrid DCNs are summarized as follows: 1) RTT of TCP is much larger than RDMA, which makes TCP flows occupying buffer more aggressively; 2) RDMA needs PFC to guarantee no packet loss; 3) Their traffic patterns are different. To protect lossless traffic and achieve optimal buffer allocation at switches in hybrid DCNs, buffer management policies needs to satisfy: 1) Reduce the chance of PFC pause frames, to protect lossless traffic performance and avoid diverse problems caused by PFC; 2) High burst flow absorption capacity and fairness; 3) Enable to flexibly distinguish traffic pattern and congestion degree, to reduce the waste of buffer resources.

1) When buffer allocation meets flow control: Specifically, the buffer allocation in lossy networks only works on destination output ports, that is, whether an arriving packet is admitted depends on whether the destination output port queue

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on December 01,2023 at 04:01:02 UTC from IEEE Xplore. Restrictions apply.

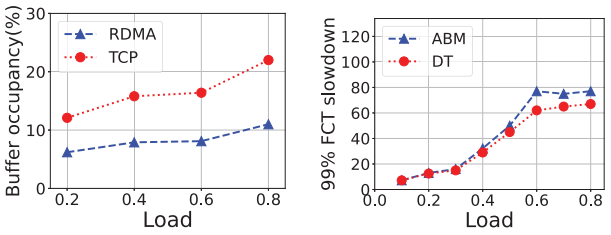
exceeds its queue threshold allocated by buffer management algorithms, and the packet may be discarded. However, in lossless networks, flow control is introduced to ensure zero packet loss, which works on switch ingress queues to prevent packet loss caused by excessive traffic injection into the switch from a single port. Therefore, the switch needs to maintain two thresholds, the buffer allocation threshold at the egress and the PFC threshold at the ingress, for a dual guarantee of high burst absorption capacity and zero packet loss. There is no doubt that this brings about a problem, that how does PFC threshold distinguish traffic of different protocols and different congestion levels, so as to ensure both the expected performance of buffer allocation and a good flow control effect.

2) Defects of classical policies: To protect lossless queue forwarding without packet loss, some switches can divide egress service pools into lossy queue service pools and lossless queue service pools automatically or manually. However, rigid division makes resource allocation inflexible, and the service pool at ingress is still shared. Therefore, switches often use general dynamic thresholds to pursue fairness, but cannot identify the congestion degree of each queue and the traffic with different service rate protocols. Specifically, the ingress pool only counts received packets, and does not know whether packets are congested at output queues, nor does it know the discharge rate of the ingress queue at this moment. Unfortunately, an more advanced buffer management algorithm ABM proposed recently, which claims to combine buffer management with active queue management to isolate traffic of different protocols, does not perform well in lossy and lossless hybrid networks, as it only considers TCP traffic and buffer division based on switch output priority queue, not considering flow control at ingress.

3) The influence of lossy traffic on lossless networks: We use some simulation experiments to demonstrate the different modes and interactions of TCP traffic and RDMA traffic. We use a three-layer fat-tree topology with 128 severs to generate both TCP traffic and RDMA traffic for server applications. The switch is configured with PFC and uses the dynamic threshold algorithm mentioned in the background.

First of all, we do not consider hybrid traffic, but separately test the switch buffer occupancy of TCP and RDMA traffic under the same web search workload. Servers under each leaf switch randomly send data to servers under other leaf switches. We use DCTCP [22] and DCQCN [12] for congestion control. Fig. 3(a) shows that under the same workload, TCP traffic occupies much more than RDMA traffic on the switch, because TCP has long RTT and RDMA has high requirements for lossless transmission. When these two traffic classes share switch resources, how to set the flow control threshold/buffer allocation is a huge challenge.

Fig. 3(b) shows the tail latency of RDMA flows under TCP/RDMA hybrid traffic. The load of RDMA flows is kept unchanged, while the load of TCP applications increases from 0.1 to 0.8. Unfortunately, we find that even if lossy traffic and lossless traffic are isolated to different priority queues,



(a) Buffer occupancy of RDMA and TCP respectively. (b) Tail latency of RDMA flows with different BM algorithms.

Fig. 3: Buffer occupancy of TCP/RDMA and tail latency of RDMA flows with the load of TCP flows increasing.

the increase of TCP load still greatly pulls up the tail latency of RDMA flows. ABM performs even worse under high loads. Meanwhile we have observed a large number of PFC pause frames, which are not expected to happen. The reason is that traditional Ethernet switch buffer management algorithms lack consideration of flow control. When lossy traffic and lossless traffic coexist, their congestion levels are different, as shown in Fig. 4. Flow control thresholds cannot correctly sense the congestion level at egress queues. Therefore, we design a buffer management algorithm that can flexibly adjust the allocation of ingress pools by perceiving congestion degree through the average waiting time of ingress packets.

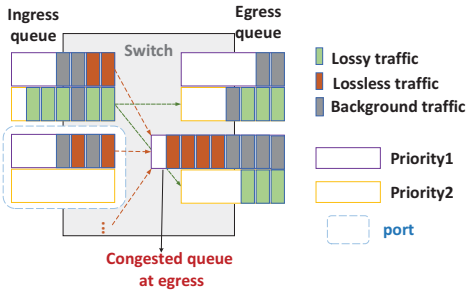


Fig. 4: Priority queues at ingress and egress ports under different congestion states of hybrid traffic.

III. Design

Motivated by defects of classical policies and experimental observations, we propose L2BM, a congestion-aware buffer management algorithm at ingress for PFC thresholds. In this section, we first explain the framework of L2BM, then discuss details of two important components of L2BM, a congestion perception module and buffer allocation strategy at ingress for flow control. Finally, we theoretically analyze the performance of L2BM in all aspects.

A. *Framework of L2BM*

The architecture of L2BM is showed in Fig. 5. At shared-memory switches in hybrid DCNs, the egress queue threshold, allocated by egress pool, is set to limit the length of the output queues, which is related to congestion control, preventing

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on December 01,2023 at 04:01:02 UTC from IEEE Xplore. Restrictions apply.

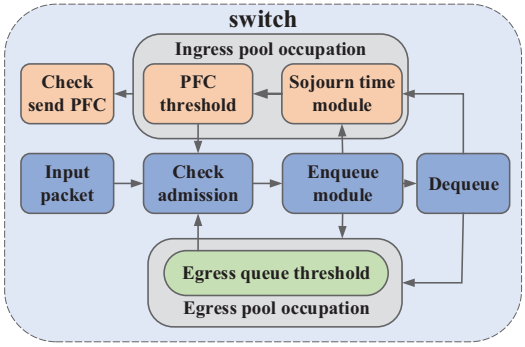


Fig. 5: L2BM architecture.

excessive occupancy or starvation. However, the ingress pool is shared by hybrid traffic. PFC threshold, allocated by L2BM at the ingress pool, is set to check sending pause frames or not, achieving flow control and preventing packet loss due to excessive injection. An arriving packet can be allocated memory and sent to the destination device only if it is admitted by both the ingress port and destination egress port, that is occupation of the ingress and the egress queue it belongs to neither exceeds the corresponding thresholds. Once this packet enqueues with admission or dequeues to downstream devices, occupancy of buffer pools is updated or removed, meanwhile the congestion detect module records and updates the average occupying time of related ingress queues. The implementation of L2BM consists the congestion detection module based on packet sojourn time at ingress ports and the concrete buffer management algorithm, which are detailed as following two parts.

B. Congestion detection module

Different from the FIFO queuing mode of output-queued packets, the waiting time of a packet in the ingress queue depends on its queue location in the destination output queue. This leads to a problem, that two input queues with the same queue length may need to be emptied in different time due to the different congestion levels of the output ports corresponding to the packets. However, TCP packets tend to pile up in output queues because of their slow response to congestion. Even if a smaller Explicit Congestion Notification (ECN) [13] threshold is set to control congestion, RDMA can react faster and packets can be emptied faster. Naturally, we should allocate a larger PFC threshold for queues that can be emptied as soon as possible to prevent unnecessary pause frames, and allocate smaller space for congested queues to prevent them from over occupying and causing packet loss. When the occupation of a queue reaches a certain threshold, the sojourn time of a single packet does not represent the emptying rate of the queue. Therefore, our congestion detection module utilizes the average sojourn time of all packets in an ingress queue , which are insidious to different output queues, to measure the discharging rate. At the input port, the congestion sensing module establishes a residence time recorder and a packet counter for each input queue. Update the sojourn time and counter every time a packet is queued or queued. In this way, the congestion sensing module can know the average occupied time of packets in the ingress queue at any time. The sojourn time of packets, namely the queuing time, can be calculated by the length of the destination output queue and related link capacity.

TABLE I: Parameters of shared buffer policy

Parameter	Description
B	Total shared buffer size of switch
t	Time
$Q(t)$	Total buffer occupation at time t
$Q^p_{out,i}$	Length of output queue at egress port i/priority p
$Q^p_{in,i}$	Length of input queue at ingress port i/priority p
N^p_i	The number of packets at ingress port i/priority p
μ^p_i	Drain rate of egress port i/priority p
τ^p_i	Average sojourn time of packets at ingress queue
α	Parameter for shared buffer allocation
$T^p_i(t)$	PFC threshold of queue at port i/priority p at t

We detail the sojourn time module in Algorithm 1. Parameters of shared buffer policy are listed in TABLE I. $Q^p_{out,i}$ and $Q^p_{in,i}$ are actually buffer size occupied by specific queues, similar to accouters at ingress ports and egress ports. Once a packet is admitted and enqueues, $Q^p_{out,i}$ and $Q^p_{in,i}$ are updated and recorder the packet size. In order to measure the average residence time of all packets in ingress queues, theoretically, the switch needs to establish a counter for each non discharged packet to update how long the packet can leave the queue in real time. However, the memory capacity of the switch is limited. In order to ensure an efficient exchange rate, the switch only maintains a total latency recorder for each input queue, and updates it when there are packets in and out of the queue. Specifically, once a packet received by port i destined for port j/priority p enqueues, the length of the output queue will be updated immediately. The length of the output queue $Q^p_{out,j}$ at this time and the port rate μ^p_j of the priority queue can be used to calculate the sojourn time of this packet, which will be included in the total time, and the number of packets N^p_i will be updated at the same time. As long as one packet is not drained, its remaining waiting time will continue to decrease until it is discharged. Therefore, when updating the total time, in addition to adding the waiting time of new packets, the time interval $t_{interval}$ that undischarged packets has been waiting for from last update time t_{prev} to now should be removed, finally forming Δ, as line 8 and line 9 in Algorithm 1 shows. Once a packet departs from the queue, the total time needs to remove the remaining sojourn time and update the number of packets, as line 14 shows.

Eventually, the average sojourn time of packets at ingress port i/priority p can be expressed as:

$$\tau^p_i = \frac{t_{total}}{N^p_i}. \tag{2}$$

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on December 01,2023 at 04:01:02 UTC from IEEE Xplore. Restrictions apply.

When there are enough buffered packets in an ingress queue, the remaining discharge time of each packet forms a distribution. For queues with short average latency, they are at low congestion degrees. These queues are allocated larger PFC thresholds as they can be emptied faster. For queues with long average latency, most packets will occupy the buffer for a long time and the discharge rate is slow, hence there is congestion detected and queues need to be limited in avoid of excessive occupation.

Algorithm 1 Sojourn time updating algorithm

Initialize total sojourn time of packets at ingress port i/priority p;
Once a packet received by port i destined for port j/priority p enqueues:
if $N_i^p > 0$ **then**
    $t_{interval} = t_{now} - t_{prev}$;
else
    $t_{interval} = 0$;
end if
$\Delta = \frac{Q_j^p}{\mu_j^p} - N_i^p \times t_{interval}$;
$t_{total} = t_{total} + \Delta$;
$N_i^p = N_i^p + 1$;
$t_{prev} = t_{now}$;
Once a packet received by port i destined for port j/priority p dequeues:
$N_i^p = N_i^p - 1$;
$t_{total} = t_{total} - (t_{now} - t_{prev})$;

C. *Buffer management algorithm*

L2BM is a buffer management scheme for the switch ingress pool, which combines flow control and congestion control. The threshold is the upper limit of the amount of data that can be received by the ingress port and is used for PFC. PFC threshold in L2BM not only considers the remaining buffer space, but also considers the congestion degree of packets queuing at output queues, which is measured by the average residence time.

$$T_i^p(t) = \frac{C}{\tau_i^p} \times \alpha \times [B - Q(t)]. \tag{3}$$

Control factor α. Based on the traditional dynamic threshold algorithm used by RoCEv2, the shared buffer allocation per port per traffic class is controlled by a parameter α. A large α can help reduce the chance of PFC pause frame generation, but a large α may lead to unfair buffer allocation and long queuing delay. In most real switches, α is usually set to 0.125. Of course, α can be configured according to the urgency and quality of service of traffic with different priorities.

Congestion perception factor. L2BM adds another control factor, which is the sojourn time of packets at ingress ports, to revise α. τ_i^p is maintained by the congestion detection module. C is a normalization constant, which can be adjusted and configured in different switches. In our experiment, we normalize C as the sum of the average sojourn time of packets in all ingress queues. This revise factor actually assigns the weight to different priority queues (traffic classes) according to their congestion degree, which is inversely proportional to the average sojourn time. Of course, the normalization method can be customized under different optimization objectives and application priorities.

D. *Analysis*

For a queue at ingress port i/priority p, the buffer allocation factor weighted by the occupied time is normalized and defined as $w_i^p(t)$, which is an adaptive control parameter:

$$w_i^p(t) = \frac{C}{\tau_i^p(t)} \times \alpha. \tag{4}$$

When all active ingress queues are gradually occupied and reach the allocated PFC thresholds by L2BM, finally in a steady state, that is, arrival rate of packets is basically equal to the queue draining rate, total buffer occupation of the switch is

$$Q(t) = \sum_{i=1}^{N} \sum_{p=1}^{8} Q_{in,i}^p(t), \tag{5}$$

$$Q(t) = \sum_{i=1}^{N} \sum_{p=1}^{8} T_i^p(t), \tag{6}$$

$$Q(t) = \sum_{i=1}^{N} \sum_{p=1}^{8} w_i^p(t) \times (B - Q(t)). \tag{7}$$

It can be solved from Eq. (7) that

$$Q(t) = \frac{B \sum_{i=1}^{N} \sum_{p=1}^{8} w_i^p(t)}{1 + \sum_{i=1}^{N} \sum_{p=1}^{8} w_i^p(t)}, \tag{8}$$

$$T_i^p(t) = \frac{B w_i^p(t)}{1 + \sum_{i=1}^{N} \sum_{p=1}^{8} w_i^p(t)}. \tag{9}$$

No burst flows: As described in the motivation, buffer occupancy of TCP flows is usually higher than that of RDMA, as TCP flows perceive congestion at switches slowly. L2BM can help switch ingress ports quickly sense the occupation time and congestion degree of TCP flows, and limit the allocated weight by Eq. (4), restricting the flow control threshold as Eq. (9) shows. Accordingly, RDMA traffic will be assigned a larger PFC threshold to tolerate unpredictable burst traffic.

Burst flow arrival: Burst traffic generated by sudden requests is injected into the switch from several ingress ports and queued at the same egress port queue, which makes the average waiting time of RDMA packets longer. After the module detects congestion, it will appropriately reduce the threshold to prevent excessive injection and overflow. It is worth noting that draining rates of ingress queues depend on the location and congestion degree of the destination output queues, rather than the port discharging rate, which is the reason why L2BM utilizes the buffer occupying time to adjust α.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on December 01,2023 at 04:01:02 UTC from IEEE Xplore. Restrictions apply.

Mitigate PFC diffusion: In Algorithm 1, the extra waiting time of packets due to ports being paused by PFC is not included in the average waiting time, that is, the module only calculates the queuing delay and ignores the suspended time, so as to avoid misjudging the suspension caused by PFC as excessive injection of the ingress ports, and reduce the unnecessary spreading of PFC to upstream devices.

IV. Performance Evaluation

In this section, we evaluate L2BM through large-scale ns-3 simulations. Our experiment is dedicated to highlighting the following results:

- L2BM can effectively protect the performance of RDMA in the hybrid protocol DCNs. When the load of lossy traffic increases, it can maintain the Flow Completion Time (FCT) of RDMA without damaging TCP flows. In particular, under high load, L2BM can reduce the 99% FCT slowdown of RDMA flows by 50%, reduce the total buffer occupation of ToR switches by 40%, and greatly reduce the number of PFC pause frames during a certain period.
- L2BM has good burst traffic absorption capacity. Under the same background TCP traffic load, L2BM can reduce the tail latency of incast flows by 23% and average request latency by 57%, effectively improving the tolerance of Ethernet switches to burst RDMA traffic.

Setup. The DCN topology shown in Fig. 6, a three-layer clos [23] architecture has 2 core switches, 4 aggregation switches and 4 Top of Rack (ToR) switches. Each ToR switch connects 32 servers with 25Gbps links , 128 servers and 10 switches in total. Link bandwidth between switches is 100Gbps. Propagation delay between aggregation switches and core switches is $5\mu s$, and the remaining links are $1\mu s$. All switches are configured with PFC. The total shared buffer size of the switch is 4MB. All servers are installed with RDMA Network Interface Cards (NICs), which are compatible with RDMA and TCP traffic.

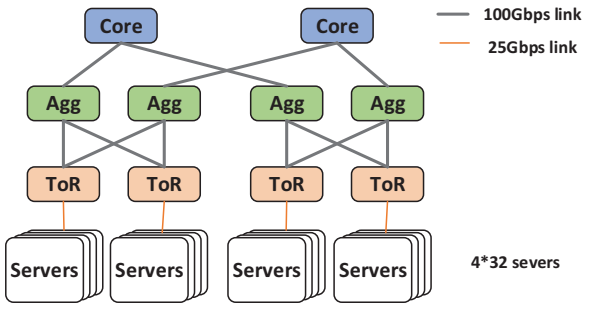


Fig. 6: Topology of the DCN in our simulations.

Workload. We generate traffic based on incast traffic and Web Search, a realistic workload heavy tailed. For Web Search workload, we generate flows with the Cumulative distribution function (CDF) of flow size under realistic workload. For incast traffic, a target server requests a file of a certain size, which is distributed and stored in other random N servers. For example, if the request size is x MB, these N servers will respond almost simultaneously, that is, each of them sends $\frac{x}{N}$ MB data to the target server at the same time. Incast traffic is also called fan-in bursts [10], characterized by high burst, high synchronization, and only if all N flows have been finished can the request be completed. We can define the severity of bursts by adjusting the value of N.

Comparison schemes. We use the traditional dynamic threshold algorithm DT and ABM as the comparison policies. The basic idea of these two schemes is briefly described as follows:

- DT is a dynamic algorithm for PFC threshold and output queue threshold used in Ethernet switches by default. The ingress and egress ports can be designed with different α. Here we mainly consider ingress pool. The larger α is, the harder it is to trigger PFC, but queue delay increases and unfairness is caused. The smaller α is, the easier it is to trigger PFC and bring a series of problems. In order to discuss the actual impact of α on lossless networks, we choose $\alpha = 0.125$ and $\alpha = 0.5$. The former refers to the PFC used in RoCEv2 network of Microsoft, and the latter is a default common parameter.
- ABM is designed based on switch output queues. Considering the number of congested queues and bandwidth allocation of each priority at egress ports, ABM can reduce the interaction of flows with different congestion control protocols.

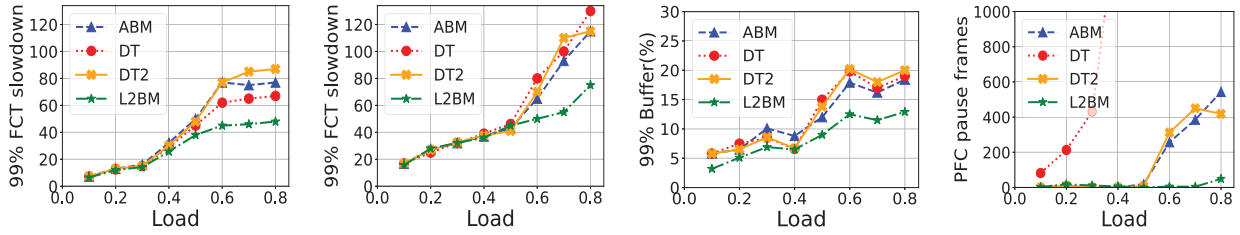
The congestion control algorithm adopted by TCP flows is DCTCP, and DCQCN is adopted by RDMA, both of which are based on ECN feedback. We compare the switch total buffer occupancy, the number of PFC pause frames and FCT slowdown (normalized FCT) i.e., the actual FCT divided by the ideal FCT with no other traffic in the network, under different TCP loads and different burst concurrency. 99% FCT is also called tail latency.

A. Isolation effect of lossy traffic and lossless traffic

We adopt the topology shown in Fig. 6. In order to realize a hybrid network where RDMA and TCP flows share switch resources, 16 servers under each ToR switch generate RDMA traffic by Poisson arriving according to the flow size CDF of web search application. The average arrival rate is related to traffic load, and data is randomly sent to all other servers. The other 16 servers generate TCP flows in the same way. Each port of the switch has eight priority queues in total. We use two of them to isolate lossless traffic and lossy traffic. In experiments, RDMA traffic maintains a load of 0.4, and TCP traffic load increases from 0.1 to 0.8. We evaluate the isolation effect of various buffer management algorithms on lossless traffic, 99% FCT slowdown, buffer occupancy, and the number of PFC pause frames.

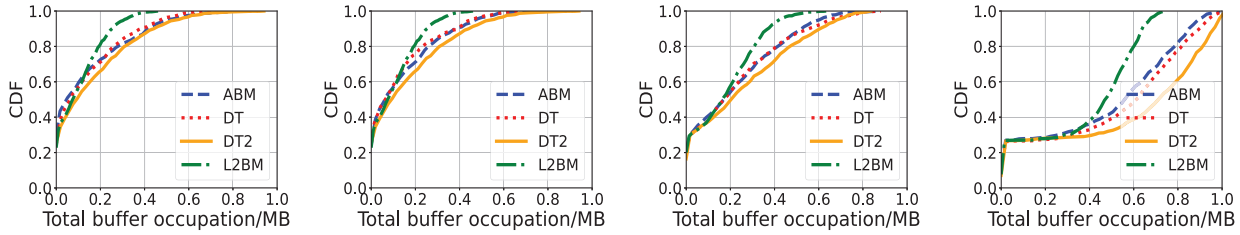
The experimental results are shown in Fig. 7. As can be seen in Fig. 7(a), with the increase of TCP traffic load, RDMA traffic is gradually encroached, resulting in an increase of 99%

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on December 01,2023 at 04:01:02 UTC from IEEE Xplore. Restrictions apply.



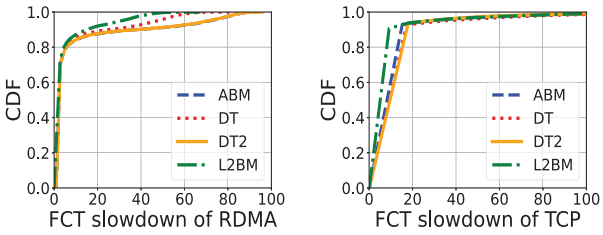
(a) RDMA flows. (b) TCP flows. (c) Buffer occupancy. (d) Number of PFC pause frames.

Fig. 7: **Performance comparison as the load of TCP traffic increases. L2BM outperforms by achieving lower FCT slowdown for both RDMA and TCP flows, especially at high loads. L2BM can reduce the overall buffer occupation of ToR switches and effectively decrease the number of PFC pause frames.**



(a) ToR switch 1. (b) ToR switch 2. (c) ToR switch 3. (d) ToR switch 4.

Fig. 8: **CDF of four ToR switches. On the whole, L2BM optimizes the total occupancy.**



(a) RDMA flows. (b) TCP flows.

Fig. 9: **L2BM significantly reduces the FCT of lossless traffic, while maintaining and optimizing the FCT of lossy traffic.**

FCT slowdown, which affects the effect and experience of web query applications. Under heavy load of lossy traffic, compared with the default DT algorithm, L2BM can reduce the 99% FCT slowdown of RDMA by nearly 50%, while not sacrificing TCP flows, reducing the 99% FCT slowdown of TCP flows by about 30% in Fig. 7(b). Compared with DT, which can alleviate the encroachment on the FCT of RDMA, ABM has a poor effect and can not isolate lossy traffic and lossless traffic. Fig. 7(c) shows the total occupancy of a ToR switch during the entire traffic transmission period. L2BM can limit 99% buffer occupancy to less than 13%. Even if the TCP load reaches 0.8, it reduces about 40% compared with other algorithms, greatly reducing the queuing delay of small flows and burst tolerance of switches.

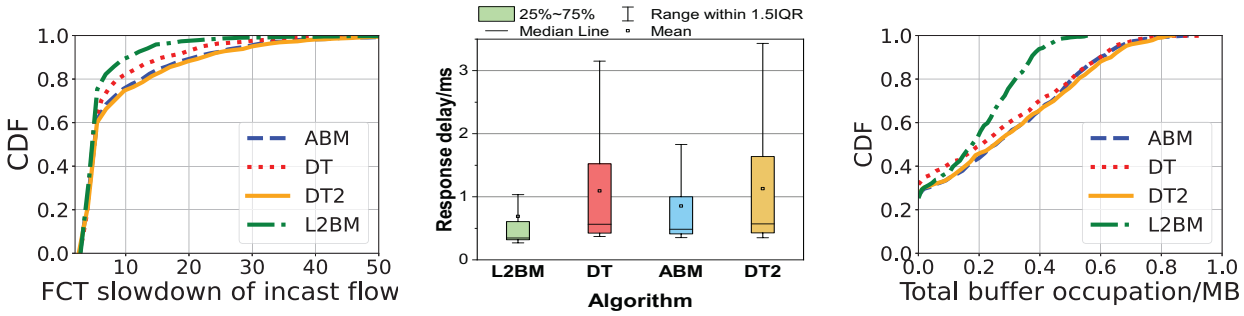
Especially, Fig. 7(d) shows the number of all PFC pause frames generated in the whole simulation process when using various algorithms under the same traffic pattern. It is obvious that L2BM can guarantee very few or even no pause frames even under extremely high traffic loads, avoiding PFC storms, deadlocks, throughput degradation and other problems from the root. Because the α of DT is too small, a large number of pause frames are generated, which can reach more than 10000 under high load. PFC has high trigger probability and serious propagation. ABM and DT2 have similar effect, but there are still about 500 pause frames under high load. In order to display the results more accurately and highlight the performance of L2BM, TABLE II records the number of PFC pause frames tracked in the whole process under different loads in detail.

TABLE II: The number of PFC pause frames

load	0.4	0.5	0.6	0.7	0.8
ABM	1	18	257	385	543
DT	1548	2597	7316	9083	10775
DT2	0	8	312	450	418
L2BM	6	0	5	4	49

Unlike traditional algorithms, which set the PFC threshold rigidly, L2BM can sense the congestion degree of output queues at the switch entrance and adjust the PFC threshold more flexibly. Therefore, it can effectively reduce the probability of triggering PFC, protect RDMA traffic and avoid harmful problems, without excessive buffer occupancy. Instead, it improves buffer utilization and reduces the overall FCT, finally the coexistence of lossy traffic and lossless traffic

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on December 01,2023 at 04:01:02 UTC from IEEE Xplore. Restrictions apply.



(a) Incast flows. (b) Error bar of all query delay. (c) CDF of buffer occupancy.

Fig. 10: **Performance comparison and error analysis in incast traffic with background flows.**

is well realized.

Fig. 8 shows CDF of the total occupancy of four ToR switches when the TCP traffic load is 0.8. The CDF is a trace of the total occupancy from traffic generation to completion, which is recorded every 0.001s. L2BM significantly reduces the overall buffer occupation of all ToR switches, as L2BM can allocate buffer more equitably and achieve better flow control. Fig. 9 shows the CDF of TCP flows and RDMA flows under high load. L2BM optimizes the completion time of RDMA flows without damaging TCP flows.

B. Burst Deep Dive

The main purpose of buffer management algorithms is to absorb burst traffic, so the performance of the algorithm under incast traffic is an crucial indicator to measure its effectiveness. In this experiment, we still make half of the servers generate the traffic under the web search workload, using DCTCP. And we set the web search traffic load up to 0.8 to verify the burst tolerance of each algorithm under high-load background traffic.

As explained above, we first make $x = 1MB$ (25% of total buffer size) and $N = 5$, that is, each time the target server initiates a file request, five other servers send 200KB of data to it. Each server may initiate requests, and the occurrence of each request follows Poisson distribution. In our actual experiment, there are 376 requests in 0.5 seconds, and 1880 flows with a size of 200KB are generated totally. We evaluated the FCT distribution of all incast flows, the actual response latency of all requests, and buffer occupation of the switch under different buffer management policies.

Normalized FCT distribution: Fig. 10(a) shows the CDF of FCT slowdown of all incast flows. Compared with other algorithms, L2BM significantly reduces the normalized FCT and limits 90% of them under 10. Suppose 10 is taken as the evaluation criterion of normalized completion time, L2BM saves at least 90% of the incast flows, 10% more than DT, and 10% more than ABM and DT2.

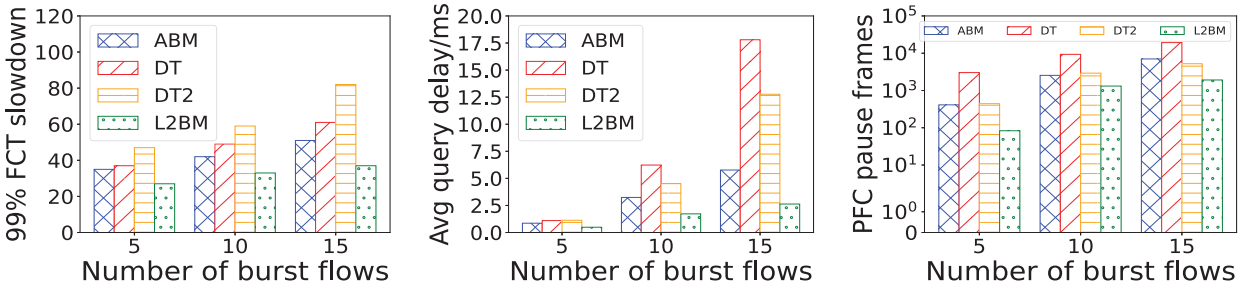
Average request latency: The response time is the maximum FCT of the five flows generated by each request. The average response time under L2BM is less than 0.48ms, which is 55% lower than DT, 43% lower than ABM, and 57% lower than DT2, the request delay greatly optimized. In addition, standard deviation and fluctuation range of response latency under L2BM are smaller. The minimum delay is compressed to 0.34ms and the maximum delay is limited to 2.9ms, which is 47% lower than DT. More error bar analysis are detailed in Fig. 10(b).

Buffer occupancy: Fig. 10(c) shows the CDF of buffer occupancy of a ToR switch under incast traffic. L2BM limits the buffer occupancy to less than 400KB for more than 90% of time, about 35% more than other algorithms. Specially, L2BM ensures that the buffer utilization does not exceed 600KB in 99% of time, while other algorithms can only maintain 99% of the utilization does not exceed 900KB or even 1MB.

Impact of the number of concurrent flows: Similarly, with the TCP web search workload as the background traffic, we change the number of concurrent flows of each request from $N = 5$ to $N = 10$ and $N = 15$, also requesting 1MB file. That is, more packets will arrive at the switch at the same time. Therefore, as the number of concurrent flows increases, the performance of the algorithm will weaken. We analyze L2BM's ability to process concurrent flows from the following three dimensions:

- **Tail latency**: Fig. 11(a) shows the 99% FCT slowdown of incast flows at different concurrency. The increase of concurrency has no significant influence on L2BM. Even if there are 15 concurrent flows, L2BM can limit the normalized tail latency to less than 40, which is 50% lower than DT2 and 23% lower than ABM.
- **Average latency**: Fig. 11(b) shows the average response latency for all requests. When there are 10 concurrent flows, the average latency of L2BM does not exceed 2.5ms. When there are 15 concurrent flows, the average latency brought by L2BM does not exceed 3.7ms, which is 78.8% lower than DT. DT has the worst performance, because its α is too small, easy to trigger PFC when burst traffic arrives. Even though the switch has free buffers, these high-frequency PFC pause frames may be fatal to the maximum completion time of concurrent flows. ABM also performed poorly as it does not consider flow control at ingress.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on December 01,2023 at 04:01:02 UTC from IEEE Xplore. Restrictions apply.



(a) FCT of all incast flows.

(b) Avg. response time of all queries

(c) PFC pause frames.

Fig. 11: **Performance comparison in incast traffic with background flows under variable incast degree.**

- **PFC pause frames**: Fig. 11(c) shows the number of PFC pause frames caused with various algorithms. The pause frames triggered by L2BM are basically one order of magnitude lower than other algorithms. Even under high concurrency, L2BM brings no more than 2000 PFC pauses frames. In theory, fewer PFC pause frames may cause excessive occupancy and unfairness. However, L2BM only flexibly allocates available buffers to burst flows according to the congestion status at output queues, not encroaching on other flows, but avoids unnecessary pauses.

C. Deployability

L2BM only needs a sojourn time recording module in the switch MMU, which is deployable with existing commodity switches through simple commands, and also easy to be implemented in hardware programmable switches or Field Programmable Gate Array (FPGA). We consider to conduct more experiments in real testbed environment in our future work.

V. Related work

Buffer management policies for lossy networks. Most switches today use the on-chip shared buffering pool to increase the buffer utilization, instead of the private memory that is exclusive for a single port [20]. The shared buffer is dynamically allocated to different ports according to specific buffer management policies. In recent years, buffer management strategies based on DT have been proposed almost every year. DT is usually the management algorithm used by the switch by default. Some variants of DT are devised: 1) Specific algorithms designed for different traffic patterns [15]–[18]: EDT [16] alleviates packet dropping caused by micro-burst traffic through fully utilizing the switch buffer and temporarily relaxing the fairness constraint; TDT [17] can sense different traffic patterns; 2) Combining active queue management with buffer partition, such as ABM [19]; 3) Using machine learning algorithm to optimize buffer partition [24], [25]: NDT uses deep reinforcement learning to learn buffer management policies without human instructions; 4) Considering different priority queues [26], [27], such as FAB, ABM. These schemes hardly consider lossless traffic and flow control at ingress ports.

Active queue management and congestion feedback. There are many ECN schemes designed to mitigate micro-burst traffic [1], [28]–[30]. BCC [1] is proposed for marking management at shared buffer switches. BCC selectively combines the congestion signals from the per-port and shared-buffer ECN marking schemes to reduce the packet loss rate and improve the overall network performance. Besides, TCD [31] is designed for congestion detection in lossless networks. TCD can detect congestion ports accurately and identify flows contributing to congestion as well as flows only affected by hop-by-hop flow controls.

PFC and avoidance. PFC is developed to enable RoCEv2 for zero packet loss, but it causes collateral damages, including head-of-line blocking, unfairness and even deadlock. Therefore, lots of research work focuses on avoiding PFC or alleviate above problems, including end-to-end congestion control [12], [14] and modification of PFC at switches [32]–[34]. DCQCN reduces the probability of triggering PFC by limiting the ECN threshold. GFC [34] is proposed to manipulate the port rate at a fine granularity, so that all ports can keep packets flowing even cyclic buffer dependency exists and deadlock is eliminated. P-PFC [32] monitors the derivative of buffer occupation, predicts the happening of PFC trigger in the future, and proactively triggers PFC pause in advance.

VI. Conclusion

In this paper, we revealed that when RDMA flows share switch buffer resources with lossy traffic such as TCP in hybrid DCNs, as the flow control at switch ingress ports cannot distinguish the congestion states of different traffic classes, unnecessary PFC pause frames are generated and the tail latency of RDMA flows are vulnerable to TCP traffic. In order to solve this problem, we proposed L2BM, a buffer management algorithm that can sense emptying time of ingress queues and congestion degree of output queues by estimating the average occupying time of packets for each ingress queue. L2BM sets more appropriate flow control threshold and ingress pool division, only need for a residence time recording module in the MMU. Large scale experiments show

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on December 01,2023 at 04:01:02 UTC from IEEE Xplore. Restrictions apply.

that L2BM outperforms previous buffer management policies, achieving flexible buffer allocation for TCP/RDMA traffic and better burst absorption.

ACKNOWLEDGMENT

This work is supported in part by the National Natural Science Foundation of China under Grant No. 61972371, Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. Y202093, JSPS KAKENHI under Grant No. JP19H04105, and the Fundamental Research Funds for the Central Universities.

REFERENCES

[1] 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," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 489–502, 2021.

[2] M. Zaharia, M. Chowdhury, T. Das, A. Dave *et al.*, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, 2012, pp. 15–28.

[3] C. Guo, H. Wu, Z. Deng *et al.*, "RDMA over commodity ethernet at scale," in *Proceedings of the 2016 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*. ACM, 2016, pp. 202–215.

[4] W. Xia, P. Zhao, Y. Wen, and H. Xie, "A survey on data center networking (DCN): Infrastructure and operations," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 640–656, 2017.

[5] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. H. Katz, "DeTail: reducing the flow completion time tail in datacenter networks," in *Proceedings of the 2012 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*. ACM, 2012, pp. 139–150.

[6] J. Han, K. Xue, Y. Xing, J. Li *et al.*, "Leveraging Coupled BBR and Adaptive Packet Scheduling to Boost MPTCP," *IEEE Transactions on Wireless Communnications*, vol. 20, no. 11, pp. 7555–7567, 2021.

[7] J. Han, K. Xue, W. Wei, Y. Xing, J. Liu, and P. Hong, "Transparent multipath: Using double MPTCP proxies to enhance transport performance for traditional TCP," *IEEE Network*, vol. 35, no. 5, pp. 181–187, 2021.

[8] R. Zhuang, J. Han, K. Xue, J. Li *et al.*, "Achieving flexible and lightweight multipath congestion control through online learning," *IEEE Transactions on Network and Service Management*, vol. 20, no. 1, pp. 46–59, 2022.

[9] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph, "Understanding TCP incast throughput collapse in datacenter networks," in *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*. ACM, 2009, pp. 73–82.

[10] G. Zeng, L. Chen, B. Yi, and K. Chen, "Cutting tail latency in commodity datacenters with cloudburst," in *Proceedings of the 2022 IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2022, pp. 600–609.

[11] 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.

[12] Y. Zhu, H. Eran, D. Firestone *et al.*, "Congestion control for large-scale RDMA deployments," in *Proceedings of the 2015 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*. ACM, 2015, pp. 523–536.

[13] Y. Zhu, M. Ghobadi, V. Misra, and J. Padhye, "ECN or Delay: Lessons Learnt from Analysis of DCQCN and TIMELY," in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies (CoNEXT)*. ACM, 2016, pp. 313–327.

[14] R. Mittal, V. T. Lam, N. Dukkipati, E. R. Blem *et al.*, "TIMELY: RTT-based congestion control for the datacenter," in *Proceedings of the 2015 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*. ACM, 2015, pp. 537–550.

[15] H. Yousefi'zadeh and E. A. Jonckheere, "Dynamic neural-based buffer management for queuing systems with self-similar characteristics," *IEEE Transactions on Neural Networks*, vol. 16, no. 5, pp. 1163–1173, 2005.

[16] D. Shan, W. Jiang, and F. Ren, "Absorbing micro-burst traffic by enhancing dynamic threshold policy of data center switches," in *Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 118–126.

[17] S. Huang, M. Wang, and Y. Cui, "Traffic-aware buffer management in shared memory switches," *IEEE/ACM Transactions on Networking*, vol. 30, no. 6, pp. 2559–2573, 2022.

[18] D. Shan, W. Jiang, and F. Ren, "Analyzing and enhancing dynamic threshold policy of data center switches," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 9, pp. 2454–2470, 2017.

[19] V. Addanki, M. Apostolaki, M. Ghobadi *et al.*, "ABM: active buffer management in datacenters," in *Proceedings of the 2022 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*. ACM, 2022, pp. 36–52.

[20] U. Cummings, A. Lines, P. Pelletier, and R. Southworth, "Shared-memory switch fabric architecture," 2010.

[21] S. Yan, X. Wang, X. Zheng, Y. Xia, D. Liu, and W. Deng, "ACC: automatic ECN tuning for high-speed datacenter networks," in *Proceedings of the 2021 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*. ACM, 2021, pp. 384–397.

[22] M. Alizadeh, A. G. Greenberg, D. A. Maltz *et al.*, "Data center TCP (DCTCP)," in *Proceedings of the 2010 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*. ACM, 2010, pp. 63–74.

[23] A. Andreyev, "Introducing data center fabric, the next-generation facebook data center network," 2014, accessed on May., 2023. [Online]. Available: https://engineering.fb.com/2014/11/14/production-engineering/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/

[24] M. Wang, S. Huang, Y. Cui, W. Wang, and Z. Li, "Learning buffer management policies for shared memory switches," in *Proceedings of the 2022 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2022, pp. 730–739.

[25] H. Rezaei, H. Almasi, and B. Vamanan, "Smartbuf: An agile memory management for shared-memory switches in datacenters," in *Proceedings of the 29th IEEE/ACM International Symposium on Quality of Service (IWQoS)*. IEEE, 2021, pp. 1–7.

[26] M. Apostolaki, L. Vanbever, and M. Ghobadi, "FAB: Toward flow-aware buffer sharing on programmable switches," in *Proceedings of the 2019 Workshop on Buffer Sizing (BS)*. ACM, 2019, pp. 1–6.

[27] E. L. Hahne and A. K. Choudhury, "Dynamic queue length thresholds for multiple loss priorities," *IEEE/ACM Transactions on Networking*, vol. 10, no. 3, pp. 368–380, 2002.

[28] W. Lyu, J. Huang, J. Liu, Z. Li *et al.*, "Mitigating port starvation for shallow-buffered switches in datacenter networks," in *Proceedings of the 41st IEEE International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 921–931.

[29] D. Shan, F. Ren, P. Cheng, R. Shu, and C. Guo, "Micro-burst in data centers: Observations, analysis, and mitigations," in *Proceedings of the 26th IEEE International Conference on Network Protocols (ICNP)*. IEEE, 2018, pp. 88–98.

[30] W. Bai, K. Chen, L. Chen, C. Kim, and H. Wu, "Enabling ECN over generic packet scheduling," in *Proceedings of the 2016 ACM Conference on emerging Networking Experiments and Technologies (CoNEXT)*. ACM, 2016, pp. 191–204.

[31] Y. Zhang, Y. Liu, Q. Meng, and F. Ren, "Congestion detection in lossless networks," in *Proceedings of the 2021 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*. ACM, 2021, pp. 370–383.

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

[33] S. Hu, Y. Zhu, P. Cheng, C. Guo *et al.*, "Tagger: Practical PFC deadlock prevention in data center networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 889–902, 2019.

[34] K. Qian, W. Cheng, T. Zhang, and F. Ren, "Gentle flow control: avoiding deadlock in lossless networks," in *Proceedings of the 2019 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*. ACM, 2019, pp. 75–89.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on December 01,2023 at 04:01:02 UTC from IEEE Xplore. Restrictions apply.