# ByteTuning: Watermark Tuning for RoCEv2

Lizhuang Tan, Zhuo Jiang, Kefei Liu, Haoran Wei, Pengfei Huo, Huiling Shi, Wei Zhang and Wei Su

**Abstract**—RDMA over Converged Ethernet v2 (RoCEv2) is one of the most popular high-speed datacenter networking solutions. Watermark is the general term for various trigger and release thresholds of RoCEv2 flow control protocols, and its reasonable configuration is an important factor affecting RoCEv2 performance. In this paper, we propose ByteTuning, a centralized watermark tuning system for RoCEv2. First, three real cases of network performance degradation caused by non-optimal or improper watermark configuration are reported, and the network performance results of different watermark configurations in three typical scenarios are traversed, indicating the necessity of watermark tuning. Then, based on the RDMA Fluid model, the influence of watermark on the RoCEv2 performance is modeled and evaluated. Next, the design of the ByteTuning is introduced, which includes three mechanisms. They are (1) using simulated annealing algorithm to make the real-time watermark converge to the near-optimal configuration, (2) using network telemetry to optimize the feedback overhead, (3) compressing the search space to improve the tuning efficiency. Finally, We validate the performance of ByteTuning in multiple real datacenter networking environments, and the results show that ByteTuning outperforms existing solutions.

**Index Terms**—Remote Direct Memory Access, RDMA over Converged Ethernet v2, Explicit Congestion Notification, Priority-based Flow Control, Datacenter Networking

## 1 Introduction

Remote Direct Memory Access (RDMA) is one of the most important and effective solutions for high-speed datacenter networking (DCN) [1]. It offers low latency and high throughput with unique features including protocol offloading and memory semantics [2]. Protocol offloading refers to offloading the network protocol onto the RDMA NICs (RNICs), which greatly reduces CPU overhead. Memory semantics provides the capabilities to access remote memory directly bypassing kernels of both sides, without remote CPU involvement. RDMA has been widely used in cloud computing and high-performance computing, such as distributed storage [3] and machine learning [4], by Microsoft [5], [6], Google [7], Alibaba [8], [9], [10], Huawei [11], and ByteDance [12], etc.

RDMA has higher requirements for network performance, such as high speed, low latency, and zero packet loss [13]. The two major characteristics of traditional Ethernet, best-effort forwarding and congestion-loss tolerance, are difficult to meet the performance requirements of RDMA. Currently, RDMA over Converged Ethernet v2 (RoCEv2) is the most popular Ethernet solution carrying RDMA [14]. It uses two flow control technologies, Explicit Congestion Notification (ECN) and Priority-based Flow Control (PFC), to achieve network lossless [15]. In addition, DCQCN [16] is a popular RoCEv2-oriented ECN/PFC-based congestion control (CC) algorithm. The operation of ECN/PFC depends on several parameters, which are collectively referred to as watermark in the industry. Proper watermark configuration for ECN/PFC is the crux to ensuring lossless and low-latency of RDMA network [17].

Most RoCEv2 switches have opened the ECN/PFC watermark configuration interfaces to data center operators, who can apply their own watermark configuration solutions according to network scale, interconnection topology, traffic patterns, etc. The existing watermark configuration solutions can be divided into the following four categories: (i) is the default solution recommended by the switch equipment manufacturer, (ii) is the solution [18] derived from the DCTCP/DCQCN theoretical model or engineering experience, (iii) is the solution represented by AI-ECN that searches for the best watermark configuration by analyzing the taffic model and building a test environment for manual or automated watermark tuning, (iv) is the solution represented by ACC [19] that searches for the near-optimal watermark configuration through distributed reinforcement learning. These solutions have their own shortcomings. (i) tends to pursue reliability and conservatism, resulting in the inability to achieve near-optimal network performance. The theoretical model of (ii) usually assumes fixed network properties (e.g., number of flows, RTT, hop count), which does not conform to the time-varying situation of the real datacenter networking. The cost of (iii) is extremely


- *L. Tan, H. Shi and W. Zhang are with Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Ji'nan 250014, P. R. China. They are also with Shandong Provincial Key Laboratory of Computing Power Internet and Service Computing, Shandong Fundamental Research Center for Computer Science, Ji'nan 250014, P. R. China. (e-mail: tanlzh; shihl; wzhang@sdas.org)*
- *Z. Jiang, H. Wei and P. Huo are with Douyin Vision Co., Ltd., Beijing 100086, P. R. China. (e-mail: jiangzhuo.cs; weihaoran.antony; huopengfei@bytedance.com)*
- *K. Liu is with State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, P. R. China. (e-mail: lkf365@bupt.edu.cn)*
- *W. Su is with National Engineering Research Center of Advanced Network Technologies, Beijing Jiaotong University, Beijing 100044, P. R. China. (e-mail: wsu@bjtu.edu.cn)*


*This work was done by L. Tan during his research internship at ByteDance and Ph.D. at BJTU. This work was supported in part by the National Natural Science Foundation of China under Grant No.62402257, the Shandong Provincial Natural Science Foundation under Grant No.ZR2022LZH015, No.ZR2023QF025 and No.ZR2023LZH017, the Integrated Innovation of Science, Education and Industry of Qilu University of Technology (Shandong Academy of Sciences) under Grant No.2023PX057, the Talent Project of Qilu University of Technology (Shandong Academy of Sciences) under Grant No.2023RCKY141. (Corresponding Authors: Z. Jiang and W. Su)*

expensive, and the tuning success rate and efficiency are low. (iv) is the best watermark tuning solution at present, but the asynchronous tuning strategy can easily cause the mutual interference between switches, thereby affecting the tuning efficiency [20]. All switches lack the rhythm of action for synchronous tuning, and make their own decisions but share feedback affected by other switches, which is also an important problem that distributed reinforcement learning is difficult to converge.

***How to implement RoCEv2 watermark tuning automatically and optimally is the goal of this paper***. The work and innovations of this paper are as follows:

(1) This paper reports three typical cases of network performance degradation caused by non-optimal or improper watermark configuration, and traverses the RoCEv2 watermark configuration space in three scenarios to demonstrate the necessity of watermark tuning.

(2) This paper proposes ByteTuning, a lossless network watermark tuning mechanism based on centralized feedback control. ByteTuning collects real-time network status information through network telemetry, uses simulated annealing algorithm to make the watermark configuration parameters approach near-optimal, and supports full-link network performance tuning from server to switch. In addition, ByteTuning simplifies the watermark configuration search space and improves tuning efficiency by minimizing interface coverage sets, data aggregation, and configuration language translation.

(3) This paper uses RDMA Fluid to model the relationship of watermark configuration to the network, and evaluates the impact of ECN and PFC watermark on network throughput, packet loss, and convergence.

(4) This paper evaluates the tuning performance of ByteTuning in the real data center of ByteDance through standard tests, Redis storage, Multi-host RNICs and other experiment scenarios. The results show that ByteTuning outperforms all existing solutions.

The rest of this paper is organized as follows: Sec. 2 introduces the relevant background and real cases, illustrating the necessity of watermark tuning. Sec. 3 investigates the configuration principles and representative solutions of ECN and PFC. Sec. 4 establishes the ECN/PFC watermark model based on Fluid, and evaluated the impact of watermark parameters on queues and packet loss. Sec. 5 elaborates on the design details of ByteTuning. Sec. 6 evaluates the performance of ByteTuning in several scenarios. Sec. 8 discusses the pros and cons of distributed and centralized tuning.

## 2 Background

In this section, first, we briefly describe the development of the RoCEv2 network and several influencing factors of its watermark configuration. Then, we report on three typical cases of misconfiguration of watermarks that occurred in real data centers. Finally, we traverse the watermark configuration space and analyze the results in three scenarios, proving the necessity of watermark tuning.

### 2.1 RoCEv2 and Watermark Configuration

Originally, RDMA was carried over the InfiniBand [21], which realizes high-throughput and low-latency lossless networking with credit-based flow control and simplified transmission protocols. In 2010, IBTA released RoCE, also known as InfiniBand over Ethernet (IBoE), which aims to replace the TCP/IP Network Layer with the InfiniBand Network Layer over the Ethernet Link Layer. In 2014, IBTA proposed RoCEv2 [22], using UDP/IP for transmission to solve the scalability problem of RoCE. In 2015, the Data Center Bridging (DCB) protocol suite was established, including IEEE 802.1Qbb Priority-based Flow Control (PFC), IEEE 802.1Qaz Enhanced Transmission Selection (ETS) and Data Center Bridging eXchange (DCBX), and IEEE 802.1Qau Congestion Notification. DCB is an enhancement to traditional Ethernet, also known as lossless Ethernet, that makes RoCE performance comparable to InfiniBand [23].

In order to achieve lossless transmission, as shown in Fig. 1(a-b), RoCEv2 introduces multiple flow control mechanisms such as ECN [24] and PFC [16]. ECN relies on switch to mark congested flows to achieve precise rate decrease at the sender. PFC extends the standard IEEE 802.3x PAUSE frame to include IEEE 802.1p Class of Service values. It can create 8 virtual channels on an Ethernet link, allowing to pause and restart any one of the virtual channels independently [25]. Both ECN and PFC take effect according to the current queue length, containing multiple trigger thresholds, which are collectively called watermark.

From the perspective of switch chips, in order to coordinate the rate between different network devices and obtain the best forwarding throughput and switching latency [26], as shown in Fig. 1(d), most of the switch chips adopt the shared-memory architecture, in which all input and output ports share the one memory. All exchanged data is stored and forwarded in the memory. The minimum unit of memory is Cell, e.g., the Cell size of the Broadcom Trident 3 is 256 B, and the total memory size is 32 MB. The memory management unit (MMU) is the core of the switch chip, and uses registers to count the logical queue length on the memory [27]. According to the purpose, the memory is divided into three parts, namely Guaranteed Buffer, Shared Buffer and Headroom Buffer [28].

(1) **Guaranteed Buffer**: A fixed buffer that provides a minimum memory guarantees for all physical ports.

(2) **Shared Buffer**: A shared buffer that can be preempted by all flows of any physical interface and queue priority.

(3) **Headroom Buffer**: A fixed buffer that stores in-transit traffic during PFC has been triggered but not yet in effect.

*How to properly configure ECN/PFC watermark is the crucial issue to efficiently utilize memory resources and ensure RoCEv2 network performance*. Intuitively, when ECN/PFC watermark is higher, the switch queue length is longer, and the CC is triggered later, the network throughput is larger, the switching latency is larger, and the packet loss probability is higher. However, there are many factors that affect the actual effect of watermark, resulting in an unsatisfactory relationship between RoCEv2 network performance and watermark configuration. In general, watermark configuration usually refer to network topology, traffic patterns, QoS requirements, and chip design.

(1) **Network Topology**: The interconnection topology, interface speed, and communication distance are the basic considerations for watermark configuration. With a fixed end-to-end latency, setting an excessively large ECN thresh-

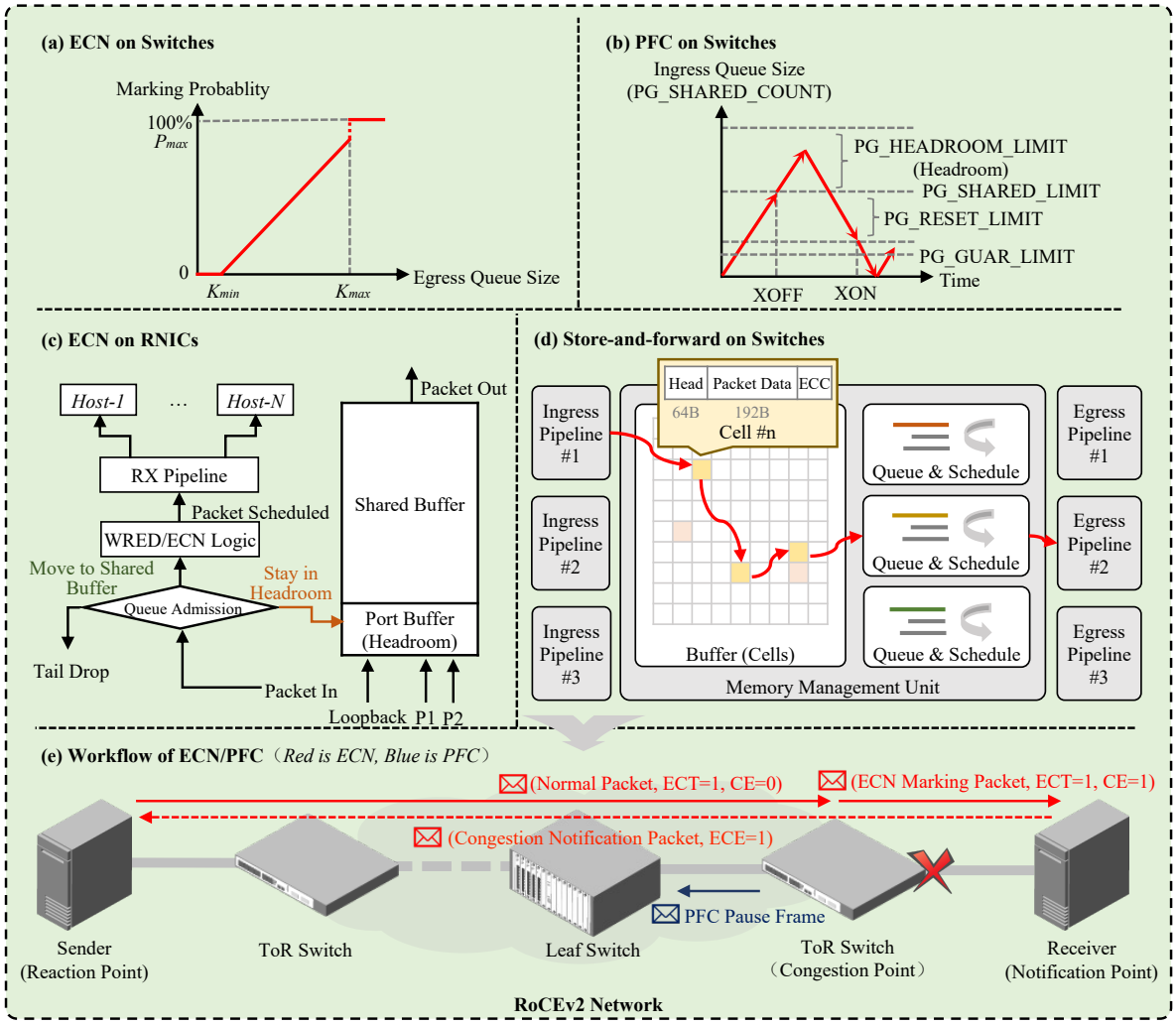



Fig. 1. Summary of mechanisms and watermarks in RoCEv2 network

old will increase the number of in-flight packets and the probability of congestion loss.

(2) **Traffic Pattern**: Many-to-one[1] Incast traffic pattern is common in cloud computing and high-performance computing [29]. In such scenarios, whether ECN can be quickly marked, whether PFC can take effect reasonably, and whether the DCQCN can respond quickly to avoid burst traffic to blow up the switch memory are important considerations for watermark configuration. In the burst scenario of multi-flow line-speed startup, the time $T$ required for the buffer of the switch connected to the receiver to fill up is

$$T = B/(N \times R). \tag{1}$$

Where $B$ is the memory size of the switch, and the ToR switch is usually 32 MB. $N$ is the number of flows, and $R$ is the RNIC rate. Taking a 240-to-1 Incast as an example, when the $R$ is 25 Gbps, it only needs 42.66 us to blow up the ToR switch. If the DCQCN/ECN/PFC does not take effect enough within this period, packet loss is very likely to occur.

1. We use $m$-to-$n$ to represent the pattern in which $m$ senders send traffic to $n$ receivers.

(3) **QoS Requirement**: In general, a high watermark seems to imply high bandwidth and high switching latency. Therefore, it is also an important consideration to meet the differentiated demands of the QoS on throughput and latency [30].

(4) **Chip Design**: Taking ECN as an example, its watermark parameters $K_{min}$ and $K_{max}$ can usually be configured by referring to the network bandwidth-delay product (BDP). E.g., $K_{min}$ is less than the expected BDP, and $K_{max}$ is less than or equal to the tolerable BDP. However, the switch chip usually measures the current queue length by software polling, and calculates the current ECN marking probability $p(t)$ by exponentially averaging the current queue length and the historical queue length. This results in $p(t)$ being non-continuous, and thus, the actual marking probability changes lag the current queue length changes, which is a deviation from the theoretical model of DCQCN. In addition, different switch equipment manufacturers have different understandings of switch chips, resulting in inconsistent performance of the same watermark configuration in data centers built with switches of different brands and switch chips of different types.

TABLE 1
Comparison of three cases

| Case # | Type | Performance | Root Case | Solution |
|---|---|---|---|---|
| **I** | Non-Optimal | Persistent Loss | $K_{max} > XOFF$ | Reduce the $K_{max}$ of Leaf Switch |
| **II** | Improper | Occasional Loss | RDMA Incast | Reduce the $XOFF$ of ToR Switch |
| **III** | Non-Optimal | Unfair Slowdown | TCP/RDMA mixed running | Increase ECN/PFC watermark |

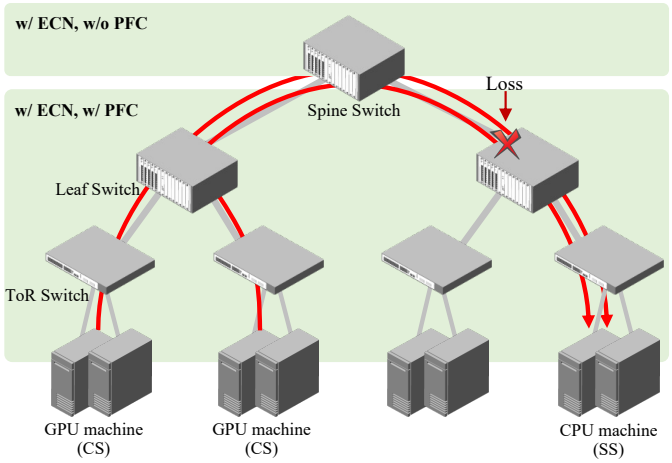



Fig. 2. Distribution of machines running BytePS and the traffic characteristics

### 2.2 Analysis of Three Cases of Non-Optimal or Improper ECN/PFC Watermark Configuration

We report and analyze three ECN/PFC watermark configuration cases, which are common in real DCNs, indicating that incorrect or inappropriate ECN/PFC watermark configuration has a great impact on network performance, as shown in Tab. 1.

**Case I: Improper watermark configuration leads to serious packet loss**

BytePS [31] is an improved distributed training framework for the traditional Parameter Server algorithm, including GPU Computation, Summation Service (SS) and Communication Service (CS). Inter-machine communication takes place between CS and SS. Due to uneven distribution of machines, RDMA line-speed startup, huge model parameters, and n-to-n communication model, network congestion is very difficult to avoid. At this time, improper watermark configuration is more likely to cause network crashes. As shown in Fig. 2, in a BytePS cluster, we observe that the uneven machine distribution resulted in a large amount of packet loss, with the peak rate of 300 K/s. Furthermore, the CPU machine has extremely low throughput and receives many PFC frames, which seriously affects the training speed. We investigate that the improper configuration ($K_{max}$ is too large) of ECN watermark caused PFC to be triggered when the queue length did not exceed the $K_{max}$. Since PFC is not enabled on the Spine switch, packet loss occurs in the inbound direction of the upstream interface of the Leaf switch on the CPU machine (SS) side.

**Case II: Specific traffic patterns leads to occasional Egress packet loss**

We observe a small amount of packet loss in the RDMA queue of a ToR switch when the Egress queue is not full. The reason is that in the traffic pattern such as n-to-1 RDMA Incast, when there are enough senders or PFC watermark is high, the traffic arriving from the first few interfaces of the switch cannot trigger PFC, and the traffic arriving later will not be buffered in the Headroom Buffer, but Shared Buffer. If the Shared Buffer and the Guaranteed Buffer of the Egress interface are filled at the same time, the Egress interface will lose packets. We mitigate packet loss by lowering PFC watermark $XOFF$ of the Ingress RDMA queue.

**Case III: TCP/RDMA mixed running leads to unfair slowdown**

Although RDMA and TCP are differentiated by different priorities on the same interface, they share the same exchange buffer. We observe that bursty TCP will occupy more buffers and affect RDMA throughput [32]. In a cluster with 8 servers running Allreduce training, (i) when there is no TCP, the RDMA throughput is about 135 Gbps; (ii) when there is TCP (∼10 Gbps), the RDMA throughput drops to 102 Gbps, with 40 K/s PFC frames; (iii) when the RDMA-TCP WRR weight is 99:1, the actual throughput ratio of RDMA-TCP is 89:11. We alleviate this unfair slowdown issue by excessively reducing the WRR weight and WRED threshold of TCP queues or Increasing ECN/PFC watermark. The root cause of this issue is that TCP and RDMA use different CC algorithms. The unfairness between flows also occurs when different brands of RNICs or different RDMA CC algorithms coexist.

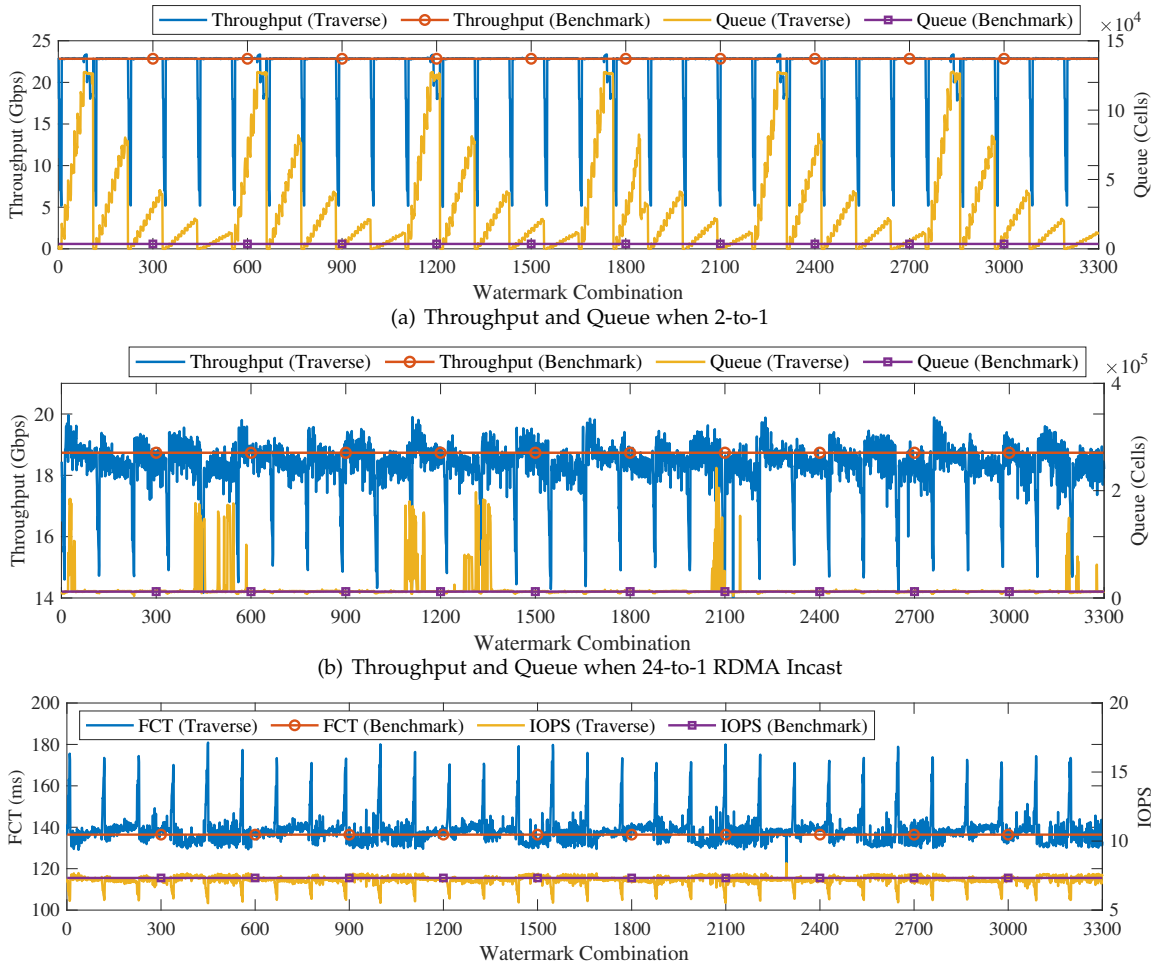
The above three cases demonstrate the importance of ECN/PFC watermark configuration to network performance. Therefore, given the network topology, traffic pattern and QoS requirement, how to find the optimal watermark configuration scheme in the current scenario is vital.

### 2.3 Analysis of Three Traversal Results of ECN/PFC Watermark Configuration

We select five crucial ECN/PFC parameters presented in Fig. 1., including $K_{min}$, $K_{max}$, $P_{max}$, $headroom$, and $\alpha$[2], and traversed their different combinations to observe the impact on throughput, queue length, and flow completion time (FCT) under the three scenarios.

The experiment environment includes 24 servers with Mellonax CX5 25GE RNIC, four H3C S6850 25GE ToR switches and two H3C S9820 100GE Leaf switches. The OFED version is 5.2. The CC algorithm is DCQCN and its parameters are recommended by Mellanox. The traffic generation tool is Perftest. The benchmark is an experience-based watermark solution deployed massively by ByteDance. The traversal space of ECN watermark is as

2. $\alpha$ is a parameter provided by the Broadcom switch chip to dynamically manage the PFC. Readers can refer to the design description of the Broadcom chip. It can simply be said that the smaller the $\alpha$, the earlier the switch triggers the PFC. It should be noted that $\alpha$ here is different from $\alpha$ in the DCQCN algorithm.




(a) Throughput and Queue when 2-to-1

(b) Throughput and Queue when 24-to-1 RDMA Incast

(c) FCT and IOPS when 24-to-1 RDMA Incast

Fig. 3. The traversal result on the many-to-one scenario

follows: $K_{min} \in \{0, 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000\}$, $K_{max} = K_{min} + 500$, $P_{max} \in \{1, 10, 20, 30, 40, 50, 60, 70, 80, 90\}$. The traversal space of PFC watermark is as follows: $\alpha \in \{1, 3, 5, 11, 20\}$, $headroom \in \{100, 200, 300, 400, 500, 600\}$. The traversal space size is 3300.

**Scene I: 2-to-1**

The number of QPs for each sender is 10, and the message size is 1024 KB. The traversal results are shown in Fig. 3(a), and the key conclusions are as follows.

(1) The benchmark solution is not the optimal. For throughput performance, 11.09% (366/3300) of the combinations surpass the benchmark, with a maximum improvement of 2.11%. For queue performance, 16.36% (540/3300) of the combinations outperform the benchmark, and the near-optimal queue length is 0. 7.00% (231/3300) of the combinations outperform the benchmark in both metrics.

(2) High ECN watermark does not exactly equal high throughput. When $\alpha = 1$, the throughput corresponding to $K_{min} = 4500$ is only 81.02% (∼22.22 Gbps) of the highest throughput, and $K_{min} = 5000$ is only 75.78% (∼19.48 Gbps). We believe that too large $K_{min}$ and too small $\alpha$ will lead to premature triggering of PFC, resulting in decreased throughput.

(3) Certain special ECN/PFC watermarks can aggravate unfair slowdowns and victims between flows. When $\alpha = 1$ and $K_{min} = 5000$, the throughputs of the two senders always converge to 14.21 Gbps and 9.43 Gbps at some moments. We guess that this is related to the DCQCN/ECN/PFC mechanism, i.e., there are some extreme cases, and the throughput of different flows cannot jump out of injustice.

**Scene II: 24-to-1 RDMA Incast**

The number of QPs for each sender is 50. The burst interval is 1000 us, and each burst is 64 KB. Each burst FCT and IOPS are counted, as shown in Fig. 3(b) and Fig. 3(c), and the key conclusions are as follows.

(1) The benchmark solution is not the optimal. For throughput, 69.87% (2306/3300) of the combinations surpass the benchmark, with a maximum improvement of 10.39%. For queue, 10.33% (341/3300) of the combinations outperform the benchmark, and the near-optimal queue length is 0. 1.57% (231/3300) of the combinations outperform the benchmark in both metrics. For burst FCT, 14.48% (478/3300) of the combinations surpass the benchmark. The benchmark time is 136.4ms (i.e., IOPS = 7.32), and the shortest is 117.9 ms (i.e., IOPS = 8.4) with a 13.53% improvement.

(2) The proper watermark solution can optimize FCT. On the Incast scenario, although the low watermark can achieve a shorter queue length and thus reduce the switching latency, it can easily lead to low throughput. This usually doesn't trigger PFC, but the low throughput results in

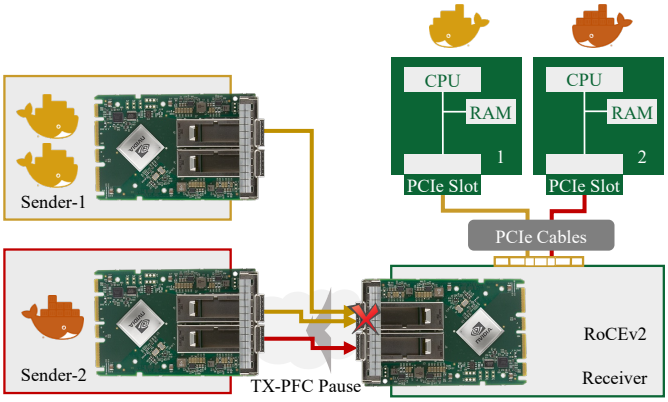


Fig. 4. Experiment topology on the Multi-host scenario

longer FCT.

(3) For RDMA Incast, the importance of the watermark of PFC is higher than that of ECN. Based on the data in Fig. 3(c), the Beta Coefficient obtained by Stepwise Regression on $headroom$, $\alpha$, $K_{min}$, $P_{max}$, and $FCT$ is [-802.29, -3858.92, 0, 0]. This means that ECN alone cannot avoid packet loss caused by DCQCN line-speed startup, and PFC plays a decisive role at this time.

**Scene III: Multi-host RNIC**

As the network speed reaches 100/200/400+ GE, which is close to the bus bandwidth of PCIe, QPI, xGMI, etc., the RNIC [33] becomes a potential congestion point [34]. Meanwhile, Servers typically deploy two or more PCIe slots. The Multi-host technology separates the PCIe bus of the RNIC into multiple independent interfaces, allowing multiple computing or storage hosts to use multiple PCIe channels and one RNIC to access the network [35]. E.g., the Mellanox CX-6 Dx SmartNIC implements multi-channel bandwidth division through Virtual Output Queue [36] and Host Management, and uses a shared Ingress buffer to accommodate burst traffic when the incoming rate exceeds what the host can receive. Although this can reduce CPU access latency and save costs, the performance of inter-flow isolation and fairness is poor. The symptom is that the many-QP flow triggers TX-PFC causing the less-QP flow to become the victim flow [37].

TX-PFC is the PFC PAUSE frame that actively requests the upstream switch to suspend sending traffic after the receiving RNIC detects that the current receiving buffer exceeds the threshold. In most TX-PFC cases, the PCIe bus is congested, and the root cause is cross-NUMA access, long PCIe path, cache miss, etc. TX-PFC can easily lead to PFC Storm and PFC Deadlock [38], so avoiding triggering TX-PFC is an important goal of Multi-host RNIC. In order to achieve this goal, Mellanox CX-6 Dx introduces Multi-host NIC-ECN, which extends the ECN function from the switch to the receiving RNIC, to actively mark and avoid triggering TX-PFC.

We verify the impact of different Multi-host NIC-ECN watermarks on throughput fairness. The experiment topology is shown in Fig. 4, where the two yellow flows are 500 QPs RDMA traffic, the red flow is 1 QP RDMA traffic, and the receiving RNIC is a 100 GE Mellanox CX-6 Dx RNIC with two Physical Functions (PFs). The traversal results are shown in Fig. 5, and the key conclusions are as follows.

(1) The total throughput of the receiving RNIC is much lower than the expected throughput of 100 Gbps, with a minimum of 54.91 Gbps. Since the yellow flow can easily trigger TX-PFC, the red flow is unreasonably blocked on the ToR switch. The total throughput of all watermarks is less than 75.34 Gbps (i.e., the total throughput of yellow flow is less than 48.28 Gbps, and that of red flow is less than 27.06 Gbps).

(2) As the $K_{min}$ increases, the network throughput decreases slightly. The highest throughput of $K_{min} = 256/512/1024$ is reduced by 0.95/1.27/1.90% than that of $K_{min}=64$.

In summary, optimizing the ECN/PFC watermark is an effective way to improve the performance of RoCEv2 networks.

## 3 Related Work

In this section, we introduce the watermark configuration principles in industry and the typical solutions in academia about ECN and PFC respectively.

### 3.1 Existing ECN watermark configuration principles

ECN reuses the judgment and processing of Weighted Random Early Detection (WRED) [39]. When the Egress is congested, the ECN-enabled packets will be marked with ECN=11 according to the marking probability at the Ingress. ECN parameters include low marking threshold $K_{min}$, high marking threshold $K_{max}$, and marking probability $P_{max}$. The existing ECN watermark configuration usually refers to the following strategies:

(1) Watermark configurations are usually static and symmetrical. Switches at the same level use the same watermark configuration.

(2) It uses fixed topology, fiber length, hop count and traffic pattern to calculate watermark parameters under extreme congestion.

(3) The actual watermark is usually obtained through manual fine-tuning of the calculation results.

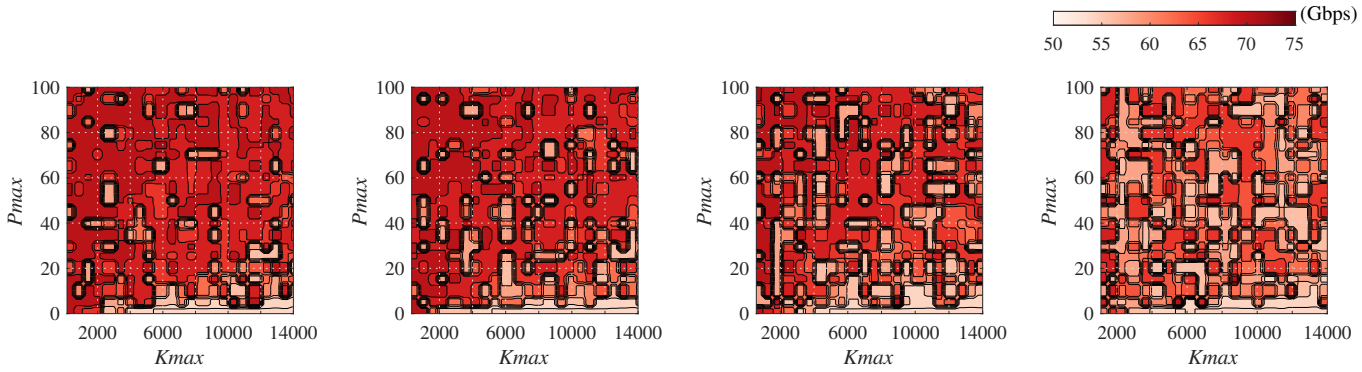
### 3.2 Typical ECN watermark configuration solutions

In 2010, Alizadeh *et al.* [40] proposed DCTCP, which is an early use of ECN for datacenter networking. DCTCP gives the lower bound of the marking threshold $K$:

$$K > (C \times RTT)/7. \quad (2)$$

Where C is the bandwidth of the bottleneck link. But this is calculated under the coexistence of $N$ long flows with the same RTT. Considering the micro-burst and the feedback delay, DCTCP adopts fixed thresholds of $K = 20$ and $K = 50(\sim 58.5$ KB) for 1 GE and 10 GE respectively in the experiment. DCTCP also recommends to determine the real RTT through large-scale network measurement, and then calculate appropriate $K$.

In 2012, Wu *et al.* [41] observed that a higher ECN threshold would increase long-flow throughput, and a lower threshold would ease TCP Incast, so they proposed a threshold calculation solution, ECN$*$ based on the transient length of the egress queue. ECN$*$ believes that the marking threshold should be set to




(a) $K_{min}=64$ (b) $K_{min}=256$ (c) $K_{min}=512$ (d) $K_{min}=1024$

Fig. 5. The traversal result at different $K_{min}$ on the Multi-host scenario

TABLE 2
Comparison of ECN watermark solutions

| Solution (Year) | Principle | Recommended Value | | |
|---|---|---|---|---|
| | | $K_{min}$ | $K_{max}$ | $P_{max}$ |
| DCTCP (2010) | $K_{max}=K_{min}>(C\times RTT)/7$. | 58.5 KB (10 GE) | | 100% |
| ECN* (2012) | $K_{max}=K_{min}=BDP/MTU=T\times C/MTU$. | —— | —— | 100% |
| DCQCN (2015) | Based on experience. | 5 KB (40 GE) | 200 KB (40 GE) | 1% |
| MQ-ECN (2016) | $K_{max}=K_{min}=min(Quantum/T_{round},C)\times RTT\times\lambda$. | —— | —— | —— |
| DCTCP-DEMT (2018) | If $N\leq C\times D/2$, $K_{max}=K_{min}=C\times D$; if $N\geq C\times D$, $K_{max}=K_{min}=0$; Otherwise, $K_{max}=K_{min}=2C\times D-2N$. Where $N$ is the number of flows, $D$ is the end-to-end latency. | —— | 10.8 KB (10 GE) | 100% |
| DCTCP-PMSB (2018) | $K_i=\frac{w_i}{\sum_{q=1}^{n_i}w_q}\times C_i\times RTT\times\lambda$, $n_i$ is the number of switch interfaces, $w_q$ is the queue weight. | —— | —— | —— |
| HPCC (2019) | Refer to DCTCP. | 30 KB (10 GE) | | 100% |
| | Refer to DCQCN. | 100 KB (25 GE) | 400 KB (25 GE) | —— |
| P-PFC (2020) | Refer to DCQCN. | 40 KB | 200 KB | 100% |
| ACC (2021) | Dynamic tuning via distributed deep reinforcement learning. | —— | —— | —— |

$$K=\frac{BDP}{MTU}=\frac{T\times C}{MTU}. \tag{3}$$

The above fixed threshold solutions are all for a single queue. In 2016, Bai *et al.* [42] observed that under the fixed threshold strategy, the queuing latency increased significantly with the number of queues (e.g., the average RTT of 8-queue network is ∼7X that of single-queue network), so they proposed MQ-ECN for multi-queue multi-priority network. The marking threshold $K_i$ for each queue is

$$K_i=min(\frac{Quantum}{Tround},C)\times RTT\times\lambda, \tag{4}$$

where $Quantum$ is the total number of bytes sent in a round, $T_{round}$ is the total time spent polling all queues.

In 2019, Zhang *et al.* [43] found that RTT is not fixed, and the maximum fluctuation reaches +268%. Using a large RTT to calculate the threshold will cause the queue to accumulate, which will significantly increase the switching latency (e.g., for Web Search, adopting P90 RTT (∼250 KB) as the ECN threshold results in a 119.2% increase in short-flow P99 FCT, and adopting the average RTT (∼100 KB) results in an 8% decrease in throughput.). They proposed ECN#, which inherits the marking strategy based on the transient queue length of ECN∗, and actively marks to eliminate unnecessary queuing latency as the queue accumulate.

In 2021, Wang *et al.* [19] proposed an ECN watermark tuning algorithm ACC based on multi-agent reinforcement learning, in which each switch acts as an agent and dynamically adjusts its own watermark to maximize the expected throughput reward.

In addition, there are some research works related to ECN watermark tuning. They reconstruct the ECN marking strategy from the perspective of resource allocation and fairness guarantee when different interfaces, queues or flows share the switching buffer. In 2017, Shan *et al.* [44], [45] found that microbursts cause ECN over-marking, thereby resulting in sender over-reaction and throughput collapse. They proposed CEDM to distinguish between transient and long-term queue increases by tracking the change rate of queue length. In 2019, Majidi *et al.* [46] proposed the Deep-

TABLE 3
Comparison of PFC watermark solutions

| Solution (Year) | Principle | Recommended Value | | | |
|---|---|---|---|---|---|
| | | $K_{pfc}$ | $P\_duration$ | $headroom$ | $\alpha$ |
| DCQCN (2015) | PFC trigger threshold $K_{pfc} = \beta(B - 8n \times Tflight–s)/8n$, where $\beta$ is the dynamic threshold for the Trident II chip, equivalent to $\alpha$. $s$ is the instantaneous buffer occupancy, $n$ is the number of interfaces, $Tflight$ is equivalent to $headroom$. Assume that the MTU is 1500 B and the network speed is 40 GE. | 24.47 KB | —— | 22.4 KB | $\beta = 8$ |
| HPCC (2019) | Based on recommendations from switch equipment vendors. | —— | —— | —— | 1/8 |
| P-PFC (2020) | $P\_duration = B \times P/V$, where $B$ is the buffer, which is 9 MB; $P$ is the number of switch ports, which is 16; $V$ is the network speed, which is 40 GE. | 200 KB | 112.5 us | —— | —— |

RL-a, which achieves an optimal per-interface ECN marking strategy based on deep reinforcement learning (DRL). They [47] further employed machine learning to optimize the marking thresholds for elephant and mouse flows, respectively. In 2021, Huang *et al.* [48] proposed a buffer-aware fair marking algorithm BFEM for the interface starvation problem caused by per-interface ECN, which punishes the attacker flow that occupies Share Buffer excessively.

### 3.3 Existing PFC watermark configuration principles

PFC is an enhancement to traditional PAUSE flow control [49]. Its watermark parameters include Headroom Threshold, PFC Guaranteed Threshold, Back-Pressure Trigger Threshold, Offset Threshold, etc.

(1) **Headroom**: A threshold that determines the buffer used to store in-flight packets for the period of time after PFC is triggered and before PFC takes effect. The industry usually refers to the following equation to determine the Headroom.

$$Headroom = MTU_R + 2 \times Link_{delay} + MTU_S + RESP. \quad (5)$$

Where $MTU_R$ is the transmission latency of the switch chip waiting for the large packet to be sent before sending the PFC PAUSE frame. $Link_{delay}$ is the one-way propagation latency (e.g., the latency for the upstream switch to receive the PAUSE frame). $MTU_S$ is the transmission latency required by the upstream switch to send a large packet. $RESP$ is the latency required by the upstream switch to process the PAUSE frame. This equation is known to be conservative and often leads to idle waste of buffer.

(2) **PFC Guaranteed**: A threshold that determines the per-interface exclusive buffer used to guarantee basic forwarding (e.g., PG_GUAR_LIMIT in Fig. 1(b)). If the PFC Guaranteed Threshold is too small, packets will be lost when PFC takes effect.

(3) **Back-Pressure Trigger**: A threshold that determines the upper limit of Shared Buffer usage for RDMA traffic (e.g., XOFF in Fig. 1(b)). This threshold is divided into two types: dynamic and static. Usually, the dynamic threshold $\alpha$ is used to allocate the proportion of available Shared Buffer for each interface or queue.

The existing PFC watermark configuration usually refers to the following strategies:

(1) PFC is usually restricted inside the Pod [50], that is, the Spine switch does not enable the PFC function to avoid PFC Storm and PFC Deadlock, as shown in Fig. 2.

(2) PFC watermark is usually higher than ECN, to avoid PFC taking effect before ECN.

### 3.4 Typical PFC watermark configuration solutions

In 2020, Tian *et al.* [51] proposed Predictive PFC (P-PFC), similar to CEDM, which actively triggers PFC PAUSE by monitoring the severity of buffer growth. Therefore, P-PFC can quickly respond to transient congestion, keep buffer occupancy low and control tail FCT. In addition, P-PFC adopts a conservative buffer growth prediction algorithm, which has a low false alarm rate. That is to say, unless the traditional PFC may trigger PAUSE with a high probability, P-PFC will not trigger PAUSE to damage throughput and latency. In addition, Cui *et al.* [52] designed a priority-aware flow control mechanism G-PFC, to prevent high-priority flow from head-of-line blocking when low-priority flow causes congestion. Shan*et al.* [53] propose dynamic and shared headroom allocation scheme, which dynamically allocates headroom to congested queues and enables the allocated headroom to be shared among different queues.

In general, As summarized in Tab. 2 and 3, these solutions either need to change the software and hardware to reconstruct ECN/PFC, or lack flexibility to deal with all scenarios.

## 4 MODEL

In this section, we use RDMA Fluid to model the packet loss probability and quantity during the convergence process of line-speed startup for $N$ RDMA flows, and analyze the impact of the watermark combination of ECN/PFC on throughput and queues. All parameters are shown in Tab. 4. In particular, we explore the cause of early-transmission packet loss during RDMA Incast due to the mismatch between DCQCN/ECN/PFC[3].

3. In this section, only $K_{pfc}$ is used to represent the PFC triggering threshold to simplify the model.

TABLE 4
Simulation parameters of the RoCEv2 Fluid model

| Parameters | Value | Explanation |
|---|---|---|
| $S$ | 1 KB | Packet size. |
| $C$ | 40 Gbps | Bandwidth of bottleneck link. |
| $N$ | 2, 4, 8 | Number of RDMA flows at bottleneck. |
| $B$ | 1000 KB | Buffer size, corresponding to 1000 packets. |
| $\alpha_D$ | 1 | Rate reduction factor, updating according to DCQCN algorithm. |
| $R_{AI}$ | 40 Mbps | Rate increase step, an additional increase of the target rate in the DCQCN fastrecovery phase. |
| $ByteCounter$ | 100 MB | Byte counter for rate increase, corresponding to 10000 packets. |
| $Timer$ | 55 us | Time counter for rate increase, fixed value. |
| $F$ | 5 | Fast recovery steps, fixed value. |
| $g$ | 1/256 | $Adjustmentparameterfor\alpha_D$, fixed value. |
| $\tau$ | 50 us | Receiver response period, within which Receiver generates at most 1 CNP. |
| $\tau^*$ | 85 us | Feedback latency, which is the sum of RTT and receiver response period. |
| $\tau^{'}$ | 55 us | Acceleration interval, within which if no CNP is received, Sender decreases $\alpha_D$. |
| $K_{min}$ | 5 KB | ECN low marking threshold, corresponding to 5 packets. |
| $K_{max}$ | 200 KB | ECN high marking threshold, corresponding to 200 packets. |
| $P_{max}$ | 10% | ECN marking probability. |
| $K_{pfc}$ | 300 KB | PFC trigger threshold, corresponding to 300 packets. |
| $T_{pfc}$ | 838 us | PFC pause duration. The default value of the switch is the time required to forward 65535×512 bits, which corresponds to 838 us for 40 GE. |

Three Incast scenarios with different traffic scales are considered:

(1) For a small traffic scale, $K_{min} < q(t) = N \times R_C - C < K_{pfc}$ is satisfied. At this time, the congested flow only triggers ECN and does not trigger PFC.

(2) For a medium traffic scale, $K_{pfc} < q(t) = N \times R_C - C < B$ or $\frac{dq}{dt} \times RTT < B$ is satisfied, where $B$ is the maximum buffer size of the bottleneck switch. At this time, the congested flow slowly triggers ECN and PFC, both of which take effect, and there is no packet loss.

(3) For a massive traffic scale, which is discussed later in this section, $K_{pfc} < B < q(t) = N \times R_C - C$ or $B < \frac{dq}{dt} \times RTT$ is satisfied. At this time, ECN has been marked but CNP has not taken effect, and the congested flow triggers PFC.

At the bottleneck switch, the ECN trigger probability $p_{ecn}$ is

$$p_{ecn} = \begin{cases} 0, & q(t) < K_{min} \\ \dfrac{q(t) - K_{min}}{K_{max} - K_{min}} P_{max}, & K_{min} < q(t) < K_{max} \\ 1, & q(t) > K_{max} \end{cases} \quad (6)$$

The PFC trigger probability $p_{pfc}$ is

$$p_{pfc} = \begin{cases} 1, & q(t) > K_{pfc} \\ 0, & q(t) < K_{pfc}. \end{cases} \quad (7)$$

We consider that the ToR and Leaf switch enable the ECN/PFC function, and the Spine switch only enables the ECN function (see Fig. 2). Packet loss occurs only on the Leaf switch on the receiving side. In particular, the condition for packet loss is that the queue occupancy exceeds the total available buffer space. The queue change of the bottleneck switch (Leaf switch) is

$$\frac{dq}{dt} = N \times R_C(t) - C(1 - p_{pfc}(t)). \quad (8)$$

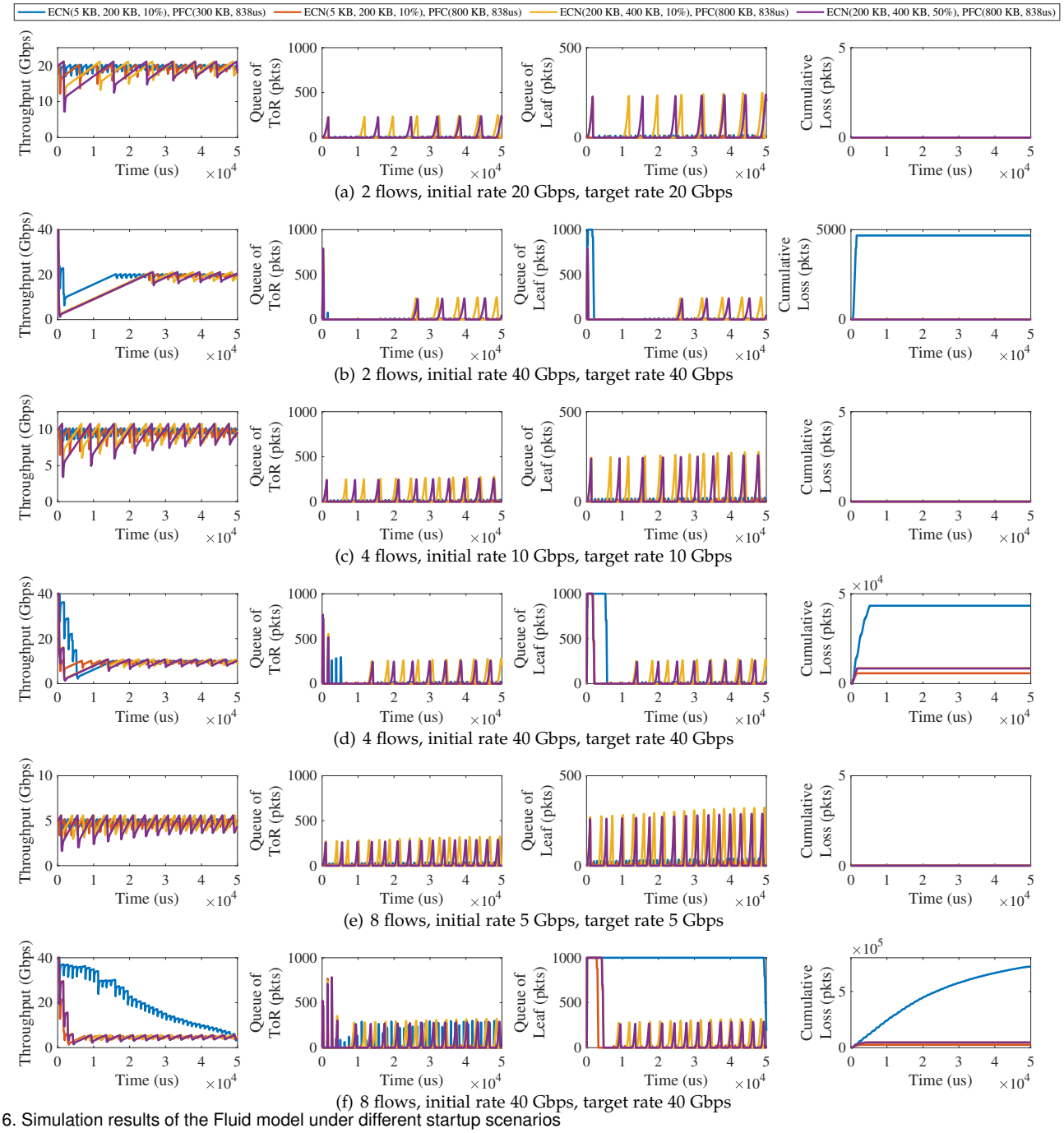
And the queue change of the ToR switch on the receiving side is

$$\frac{dq}{dt} = N \times R_C(t)(1 - p_{pfc}(t)) - C. \quad (9)$$

Since the parameter $\alpha_D$ in the DCQCN includes two update strategies, they are $\alpha_D = (1-g)\alpha_D + g$ and $\alpha_D = (1-g)\alpha_D$. Wherein, the update condition of the former is that the sender receives a valid CNP, and the update condition of the latter is that the sender has not received the CNP continuously. Therefore, the update trend of $\alpha_D$ is

$$\frac{d\alpha_D}{dt} = \frac{g}{\tau^{'}}((1 - (1 - p(t - \tau^*))^{\tau^{'} R_C(t - \tau^*)}) - \alpha_D(t)). \quad (10)$$

In addition, the update of the per-flow target rate $R_T$ includes two cases, corresponding to the deceleration phase and acceleration phase of DCQCN, in which the acceleration phase is divided into Fastrecovery and Additiveincrease according to different acceleration strategies. Among them, when deceleration, the variation of $R_T$ is $R_T - R_C$; when Fastrecovery, the variation of $R_T$ is 0; when Additiveincrease, the variation of $R_T$ is $R_{AI}$. The trigger condition of the Additiveincrease stage is that $Timer$ and $ByteCounter$ are not all greater than $F$ or not all less than $F$. Therefore, the update trend of the $R_T$ is




(a) 2 flows, initial rate 20 Gbps, target rate 20 Gbps

(b) 2 flows, initial rate 40 Gbps, target rate 40 Gbps

(c) 4 flows, initial rate 10 Gbps, target rate 10 Gbps

(d) 4 flows, initial rate 40 Gbps, target rate 40 Gbps

(e) 8 flows, initial rate 5 Gbps, target rate 5 Gbps

(f) 8 flows, initial rate 40 Gbps, target rate 40 Gbps

Fig. 6. Simulation results of the Fluid model under different startup scenarios

$$\begin{aligned}\frac{dR_T}{dt} =& -\frac{R_T(t)-R_C(t)}{\tau}\left(1-(1-p(t-\tau^*))^{\tau R_C(t-\tau^*)}\right)\\ &+R_{AI}R_C(t-\tau^*)\frac{1-p(t-\tau^*)^{FB}p(t-\tau^*)}{(1-p(t-\tau^*))^{-B}-1}\\ &+R_{AI}R_C(t-\tau^*)\frac{1-p(t-\tau^*)^{FTR_C(t-\tau^*)}p(t-\tau^*)}{(1-p(t-\tau^*))^{-TR_C(t-\tau^*)}-1}.\end{aligned} \tag{11}$$

The update trend of the per-flow actual rate $R_C$ is

$$\begin{aligned}\frac{dR_C}{dt} =& -\frac{R_C(t)\alpha_D(t)}{2\tau}\left(1-(1-p(t-\tau^*))^{\tau R_C(t-\tau^*)}\right)\\ &+\frac{R_T(t)-R_C(t)}{2}\frac{R_C(t-\tau^*)p(t-\tau^*)}{(1-p(t-\tau^*))^{-B}-1}\\ &+\frac{R_T(t)-R_C(t)}{2}\frac{R_C(t-\tau^*)p(t-\tau^*)}{(1-p(t-\tau^*))^{-TR_C(t-\tau^*)}-1}.\end{aligned} \tag{12}$$

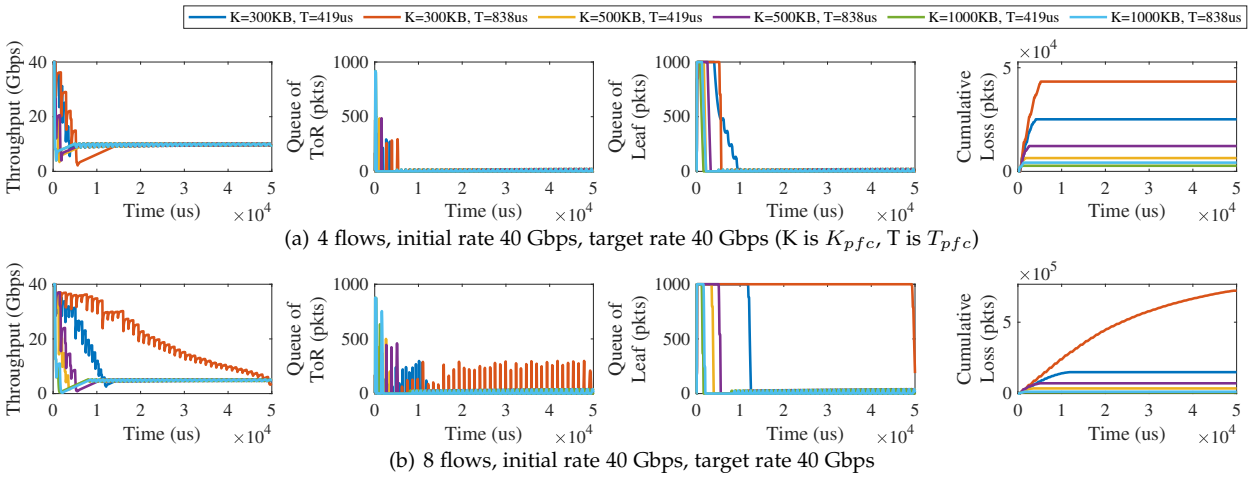
Therefore, during the period when PFC is triggered, the accumulated packet loss $L$ of the bottleneck switch is

$$L=\int_0^T \max\{0, N\times R_C(t)-C(1-p_{pfc}(t))-B\}\,dt. \tag{13}$$

We simulate the cumulative packet loss for different flow numbers $N$ under different watermark parameters. The simulation code is open source [54]. The simulation topology is shown in Fig. 2, and the results are shown in Fig. 6:

(1) The larger the number of flows or the larger the initial rate, the slower the convergence and the more packet loss.

(2) Reducing the initial rate and initial target rate can alleviate congestion. In particular, too high of these two rates




(a) 4 flows, initial rate 40 Gbps, target rate 40 Gbps (K is $K_{pfc}$, T is $T_{pfc}$)

(b) 8 flows, initial rate 40 Gbps, target rate 40 Gbps

Fig. 7. Simulation results of the Fluid model under different PFC parameters ($K_{min} = 5KB$, $K_{max} = 200KB$, $P_{max} = 10\%$)

must cause excessive deceleration.

(3) Appropriate ECN/PFC watermark parameters can improve packet loss and convergence. Under the same PFC parameters, higher ECN marking threshold and probability performance is worse (i.e., the combination of ECN (5 KB, 200 KB, 10%) and PFC (800 KB, 838 us) always perform best).

The exception is that the low watermark (blue line) does not achieve good convergence. We believe this is due to the low marking probability and PFC threshold. Therefore, we further verify the impact of different PFC trigger and duration thresholds on throughput and queuing, and the results are shown in Fig. 7. Convergence performance, queue length at congestion points, and packet loss are negatively correlated with the PFC threshold and positively correlated with the PFC duration. Correspondingly, the combination of $K_{pfc}$=$300KB$ and $T_{pfc}$=$838us$ performs the worst.

## 5 Design

In this section, we introduce the design goals, overall architecture, optimization methods, and tuning algorithm of ByteTuning, an automated watermark tuning system.

### 5.1 Design Goals

ByteTuning should have the following characteristics:

(1) Simplicity: The feedback control of ByteTuning should be easy to deploy, especially the network status measurement and watermark configuration delivery should be lightweight.

(2) Efficiency: ByteTuning should avoid using a stateless blind search algorithm to avoid invalid repeated searches, due to the huge watermark configuration space. In addition, ByteTuning should have strong adaptability to reduce control lag.

(3) Scalability: ByteTuning should adapt to servers, topologies and traffic patterns in different situations, especially to adapt to different software and hardware switches.

### 5.2 Architecture

Based on the above design goals, as shown in Fig. 8, the core modules of ByteTuning include network telemetry, traffic generation engine and tuning engine. Among them,

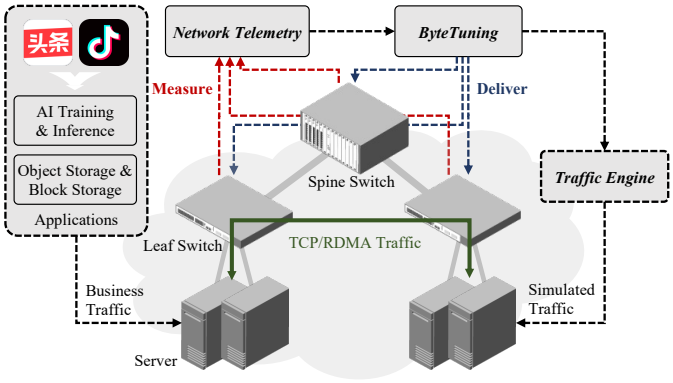



Fig. 8. Key modules of ByteTuning

the network telemetry module collects network status on demand, which is the input and feedback signal of Bytetuning. The traffic generation engine constructs network traffic, avoiding excessive reliance on typical real services such as machine learning training and inference. The tuning module executes the tuning algorithm and delivers the near-optimal watermark configuration.

### 5.3 Tuning Algorithm

The essence of watermark tuning is a black-box optimization, i.e., only the correspondence between input and output is known, but its internal structure cannot be expressed in a specific form. Therefore, the goal of tuning is to optimize the black-box objective function within a limited evaluation cost, and find a watermark configuration solution as quickly as possible, under which the value of the black-box objective function is close to the global optimum. For this problem, common solving algorithms include traversal, grid search, random search, Bayesian optimization, Monte Carlo tree search, heuristic search, and reinforcement learning. Considering the solution efficiency and deployment difficulty, ByteTuning adopts simulated annealing (SA) algorithm.

Since DCN usually focuses on real-time throughput and latency, and considering the difficulty of collecting network performance indicators, ByteTuning uses dynamic throughput $T$ and dynamic latency $Q$ to evaluate network performance, so the feedback parameters of ByteTuning are

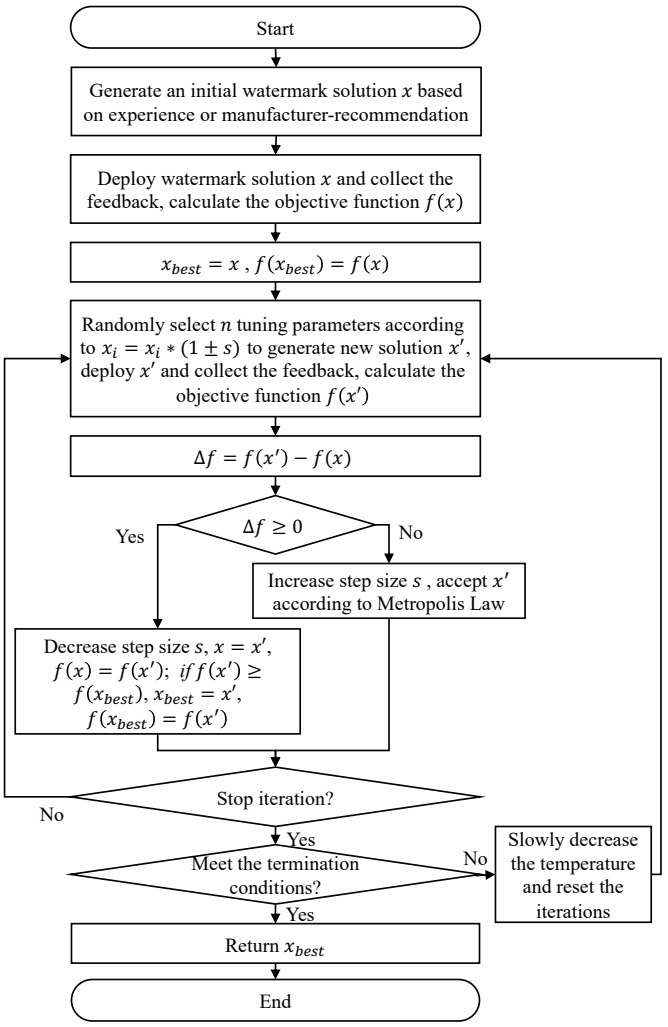



Fig. 9. Search steps for ByteTuning

**Algorithm 1:** Watermark Tuning Algorithm

**Input:** Initial watermark: $x$, Iterations: $I$, Initial temperature: $tem$, Target temperature: $tem_{goal}$, Cooling coefficient: $C$, Search step: $s$.
**Output:** Near-optimal watermark: $x_{best}$, Optimal objective function: $f(x_{best})$.

1 **Function** *bytetuning()*
2 $f(x)$ = compute_Energy($x$);
3 $x_{best} = x$;
4 $f(x_{best}) = f(x)$;
5 **while** $tem > tem_{goal}$ **do**
6 **for** $i = 1 : I$ **do**
7 $f(x)$ = compute_Energy($x$);
8 $x'$ = perturb($x$);
9 $f(x')$ = compute_Energy($x'$);
10 **if** $f(x')$ - $f(x) \geq 0$ **then**
11 $x = x'$ and $f(x) = f(x')$;
12 $s = s/2$;
13 **if** $f(x')$ - $f(x_{best}) \geq 0$ **then**
14 $x_{best} = x'$ and $f(x_{best}) = f(x')$;
15 **end**
16 **else**
17 $s = s \times 2$;
18 **if** $e^{\frac{f(x')-f(x)}{tem}} > rand()$ **then**
19 $x = x'$ and $f(x) = f(x')$;
20 **end**
21 **end**
22 **end**
23 $tem = tem \times C$;
24 **end**
25 **end**

the weight coefficient $\beta$ of throughput and latency and the fairness coefficient $\gamma$ between flows [55]. The optimization objectives $f$ include maximizing throughput, minimizing latency, and maximizing inter-flow fairness.

So for throughput- or delay-sensitive traffic, the watermark tuning objective function is the weighted sum of throughput and queues, that is

$$f{=}\text{maximize}\sum_{i=1}^{n}[\beta\frac{T_{now}(switch_i)}{T_{max}(switch_i)}+(1{-}\beta)\frac{Q_{min}(switch_i)}{Q_{now}(switch_i)}], \quad (14)$$

where $T_{now}$ and $Q_{now}$ are the throughput and queue (latency) of the current tuning round, and $T_{max}$ and $Q_{min}$ are the historical optimal values.

For fairness-sensitive traffic, the objective function is to maximize the per-flow expectation and minimize its variance, that is

$$f{=}\text{maximize}\{\frac{\sum_{i=1}^{n}[\beta\frac{T_{now}(flow_i)}{T_{max}(flow_i)}+(1-\beta)\frac{Q_{min}(flow_i)}{Q_{now}(flow_i)}]}{\gamma\sum_{i=1}^{n}[T_{now}(flow_i)-T_{now}(\bar{flow}_i)]^2}\}. \quad (15)$$

As shown in Algo. 1[4], the optimization strategy of the tuning algorithm based on simulated annealing includes:

(1) ByteTuning becomes more conservative as the iteration increases, corresponding to the slow decrease in temperature in SA.

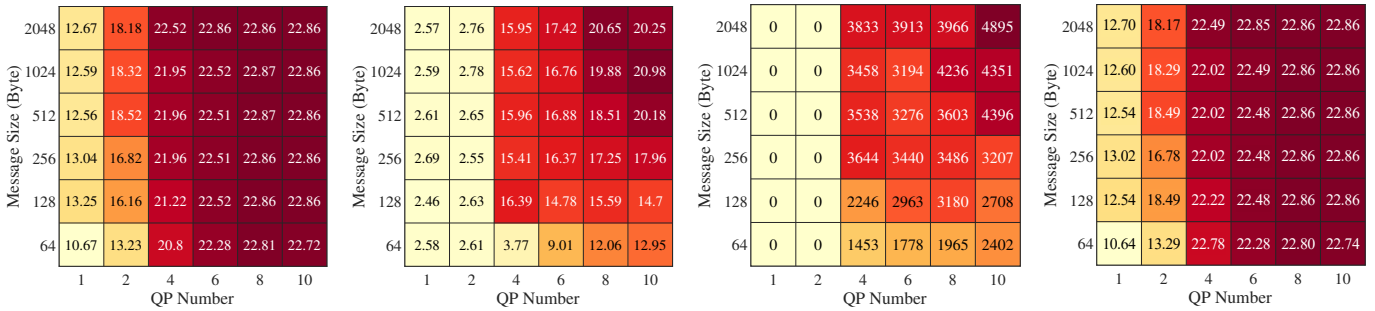
(2) In order to take into account both the breadth and depth of the search, ByteTuning adopts a step size adjustment strategy of *"if a better watermark is found, the step size will be halved; if a poorer watermark is found, the step size will be doubled"*. Among them, the purpose of halving the step size is to enable ByteTuning to deepen the search in the better solution area, so as to search for the near-optimal watermark solution more efficiently. The doubling of the step size enables ByteTuning to jump out of the poor solution area as much as possible to avoid invalid search. The overall process is shown in Fig. 9.

Among them, when the new solution is inferior to the old solution, the tuning algorithm adopts the Metropolis criterion to accept the new solution (see lines 18-20). The number of iterations and the cooling process determine the total search rounds. We recommend that the number of iterations $I$ be 10, initial temperature $tem$ be 100, the target temperature $tem_{goal}$ be 0.01, the cooling coefficient $C$ be 0.99 and the search step $s$ be 0.5. The algorithm has been desensitized and open source [56].

### 5.4 Optimization Strategy

ByteTuning is optimized in the following three ways.

4. The function $compute_Energy()$delivers watermark configuration and collects network status, which returns the objective function value according to the Eq. 14 and 15; The function $perturb()$ returns the new solution $x'$ generated by perturbation, as described in optimization strategy (2).




(a) Result of throughput (Gbps) (b) Result of latency (us) (c) Result of queue (Cells) (d) INT result of throughput (Gbps)

Fig. 10. Measurement results of throughput, latency, and queue when 2-to-1 with different traffic patterns

(1) Optimizing for Telemetry: Modern data centers deploy a variety of network measurement tools, most of which are out-of-band, supporting the measurement of queue throughput, packet forwarding rate, packet loss rate, buffer utilization ECN/PFC statistics, switch hardware utilization, etc. But their reporting interval is at least ms level (e.g., 30s to 600s for SNMP, 100ms to 30s for out-of-band telemetry), which is difficult to meet the requirements of ByteTuning.

Firstly, we recommend using queue lengths to evaluate network latency. As shown in Fig. 10 (a-c), we observe that network latency was positively correlated with switch queue using out-of-band telemetry (ONT). For example, the average no-load latency of the RDMA network in a Pod is 2.61 us, the light-load latency is 16.37 us, and the heavy-load latency is 21.33 us. The corresponding real-time queue lengths are 0, 3516, and 4667 Cells.

Additionally, we recommend using in-band network telemetry (INT) [57] to obtain end-to-end fine-grained network statistics (e.g., Broadcom Trident3, Tomahawk2 and newer chips already support this feature). In order to reduce the overhead, we directly obtain the network latency by telemetering the queue, and indirectly infer the throughput by the number of telemetry reports. As shown in Fig. 10 (a) and (d), the throughput results obtained by INT and ONT are almost the same.Throughput is related to the number of telemetry reports per unit time, the average packet size, and the telemetry sampling rate, that is,

$$Throughput = \frac{Num_{TelemetryReports} \times S}{SamplingRate}. \quad (16)$$

Finally, considering the constraints of the processing performance of switches and network telemetry servers, ByteTuning processes the original network telemetry information through data aggregation and feeds it to the tuning module. Available aggregation operations include $Min()$, $Max()$, $Sum()$, $P99()$, etc.

(2) Compress the Search Space: Assuming that the entire network contains $k$ switches, and each switch is enabled with ECN and PFC, the size of the watermark tuning search space is $(ECN_{option} \times PFC_{option})^m$, e.g., if each switch is tuned separately in Sec.2.3, the search space is $3300^6$. Since the data center has a hierarchical symmetric topology and top-down symmetric traffic, ByteTuning can simplify the control overhead by picking out the switches through which the traffic passes. Moreover, we recommend two strategies to simplify this process, one is to only collect the state of the switch at the receiving side and the other is to configure the same watermark for all ports of the switch at the same layer.

(3) Translation for Configuration Language: ByteTuning needs to shield the differences in watermark configuration commands and chip features of software/hardware switches. We design a configuration language translator for more than 60 kinds of switches including Cisco, Arista, Huawei, H3C and other commercial switches and whitebox switches. It acts between the tuning module and the switch to realize the differentiated distribution of watermark configuration intentions.

## 6 Result

In this seciton, we verify the performance of ByteTuning and various benchmark tuning schemes in various scenarios, proving the superiority of ByteTuning.

### 6.1 Setup

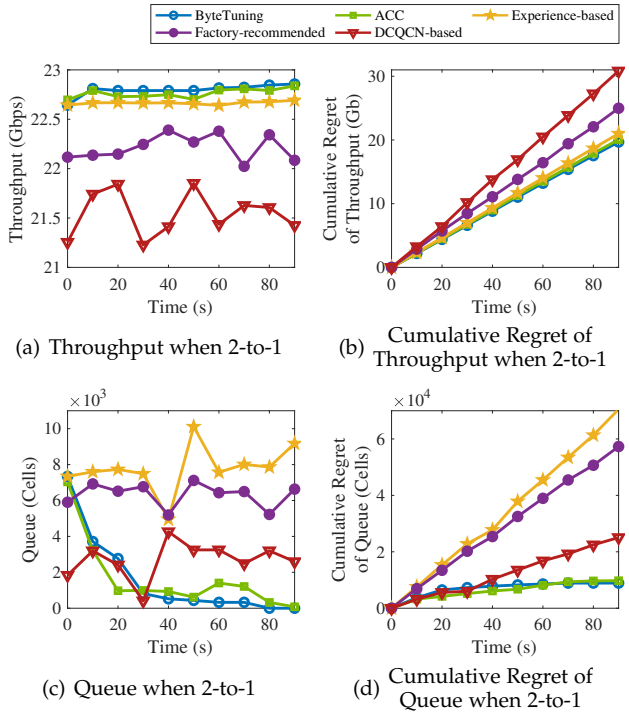
We build two test clusters in the real data center. The first cluster contains 40 servers, and they are interconnected as shown in Fig. 2. The bandwidth of the access/aggregation/core links is 25/100/200 Gbps. The second cluster contains 3 Multi-host servers, which are respectively connected to 3 ToR switches. The bandwidth of the access/aggregation/core links is 100/200/400 Gbps. The server is equipped with Intel Xeon Platinum 8260 CPU@2.40GHz 96 cores, 25 GE Single-host Mellanox CX-5 or 200 GE Multi-host Mellanox CX-6 RNIC, 376 GB DDR4 memory, Debian v9.13 (Linux Kernel v4.14) and MLX OFED v5. 0.

We compare the performance of ByteTuning, ACC, experience-based watermark, manufacturer-recommended watermark and DCQCN-based watermark. Unless otherwise specified, the network telemetry period is 1 s, and the weight $\beta = 0.5$. Tuning algorithm parameters are described in Sec. 5.3. The tuning parameters for ECN/PFC are $K_{min}$, $K_{max}$, $P_{max}$, $headroom$, and $\alpha$.

### 6.2 Scene I: Standard Test

On the first cluster, we verify the tuning performance of ByteTuning when 2-to-1 and 39-to-1 RDMA Incast.

**2-to-1**: The number of QPs for each sender is 100, and the message size is 1024 KB. The results are shown in Figure 1. Compared with ACC/experience-based/Manufacturer-recommended/DCQCN-based, the average throughput of




(a) Throughput when 2-to-1 (b) Cumulative Regret of Throughput when 2-to-1

(c) Queue when 2-to-1 (d) Cumulative Regret of Queue when 2-to-1

Fig. 11. The tuning result when 2-to-1

ByteTuning increased by 0.14%/0.57%/2.62%/5.81% (see Fig. 11(a)), and the average queue length of switches decreased by 3.21%/ 79.12%/74.28%/39.56% (see Fig. 11(c)). In particular, since the 30s, the switch queue of ByteTuning has always been lower than 1000 Cells, and since the 80 s, it has dropped to 0. ACC is always below 1000 Cell after 20 s, except 60 s and 70 s.

In addition, as shown in Fig. 11(b) and 11(d), the cumulative throughput regret is the accumulation of the difference between the actual throughput and the theoretical maximum throughput, and the cumulative queue regret is the sum of the cumulative queues. Both ByteTuning and ACC are better than other solutions, indicating that they can effectively optimize the network performance of RoCEv2, and ByteTuning is slightly better than ACC.

**39-to-1 RDMA Incast**: The number of QPs for each sender is 10 or 50, the burst interval is 1000 us, and each burst is 10 KB or 64 KB. We evaluate P50 and P99 FCT of different solution. As shown in Fig. 12, for P50 FCT, ByteTuning performs best. For P99 FCT, ByteTuning is only about 2.62% worse than ACC when QP number = 50 and message size = 10 KB. Compared with the experience-based watermark, ByteTuning optimizes P50 FCT by 14.65%/8.11%/7.19%/9.19% and P99 FCT by 22.71%/6.52%/8.87%/8.20% on the four cases respectively.

### 6.3 Scene II: Redis Storage

As a high-speed in-memory key-value database sensitive to throughput and latency, when the message size exceeds 400 KB, the performance benefit of Redis over RDMA [58] is significantly better than that of Redis over TCP. We verify the performance optimization of ByteTuning for the communication between Redis Server and Proxy. The Redis system is ByteDance be4redis, and the benchmark tool is Redis-benchmark. Redis Client sends Set to Redis Server, the message size is 64 KB, the number of clients is 20, and the number of threads for each client is 10. The evaluation metrics include IOPS, FCT, queues and number of ECN-marked packets.

The results are shown in Tab. 5. The performance of Redis over RDMA is better than Redis over DCTCP, with a 43.7% increase in throughput and a 99.2% reduction in switch queue. Compared with the other two watermark solutions, the throughput of ByteTuning increased by 9.1% and 5.4%, and the FCT decreased by 18.2% and 7.1% respectively. In particular, the number of ECN-marked packets of ByteTuning is close to that of DCTCP.

### 6.4 Scene III: Multi-host RNIC

On the second cluster, we evaluate the 2-to-1 performance of ByteTuning and the ECN watermark parameters of the Multi-host RNIC recommended by the manufacturer under different DCQCN speed adjustment intervals, to verify the adaptability of ByteTuning to the CC algorithm.

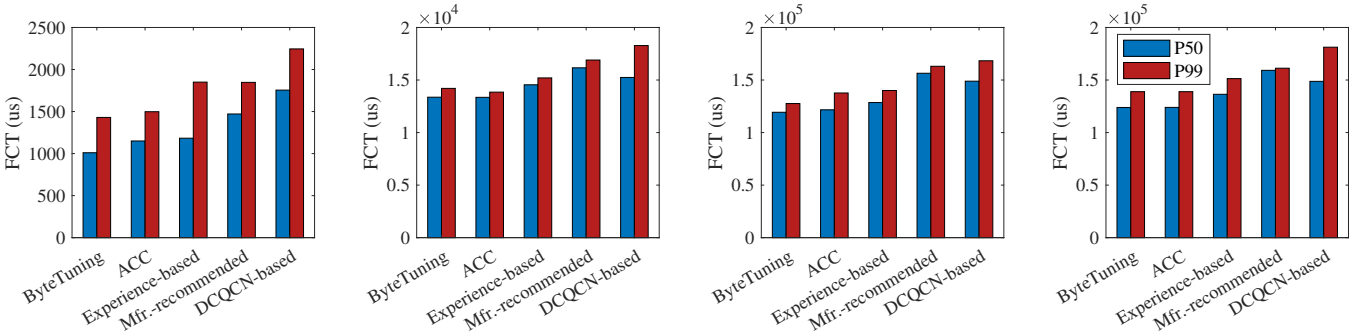
The result is shown in Fig. 13. (1) When Rate_Reduce_Monitor_Period=0/1 and Min_Time_Between_CNPs=0, the number of ECN-marked packets of ByteTuning reduces by 27.52% and 11.70%, while the throughput reduces by 0.8% and 1.9%. The reason may be that the increase of DCQCN speed adjustment sensitivity leads to the slowdown of congestion. In other cases, the number of ECN-marked packets of ByteTuning increases by 5.03%-44.59%. (2) In all cases, the number of TX-PFC Pause of ByteTuning reduces by 4.3%-91.16%, and the TX-PFC Pause Duration reduces by 17.59%-97.79%. (3) When 3-to-1, where Host_C is Multi-host Server with two PFs, Host C_1 and Host C_2, and the two DCQCN adjustment parameters take 4, the 1QP throughput of ByteTuning increases by 1071.42% and 81.41% (see Tab. 6), compared with no-VoQ and VoQ (manufacturer-recommended watermark).

### 6.5 Scene IV: Telemetry Overhead

As shown in Tab. 7, we count the bandwidth overhead generated by ByteTuning using different network telemetry tools on the first cluster. Among them, in-band network telemetry (INT) achieves extremely high telemetry frequency while having the smallest overhead, followed by out-of-band network telemetry (ONT). According to the optimization described in Sec. 5.4, we measure the telemetry performance of the Broadcom TD3/TH3 chip to reach 200 KPPS. In addition, Broadcom TD4/TH4 already supports per-packet in-band network telemetry, which is adopted by ByteTuning with extremely low overhead.

## 7 Discussion

In this section, we focus on discussing and comparing ByteTuning with the existing state-of-the-art working ACC. They are representatives of centralized watermark tuning and distributed watermark tuning respectively. In general, ACC and ByteTuning are the few works that optimize RoCEv2 network performance via tuning. Some previous works have slightly discussed watermark configuration (see Sec. 3), but they don't go deep.

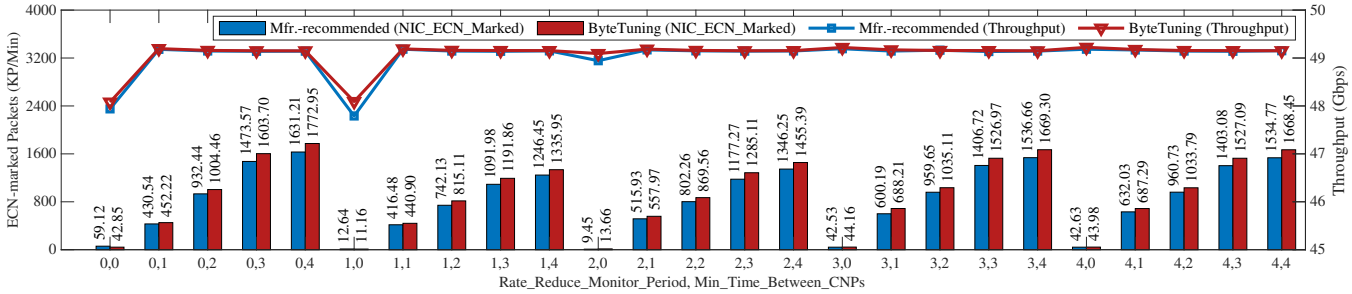



(a) QP=10, Message Size=10 KB (b) QP=50, Message Size=10 KB (c) QP=10, Message Size=64 KB (d) QP=50, Message Size=64 KB

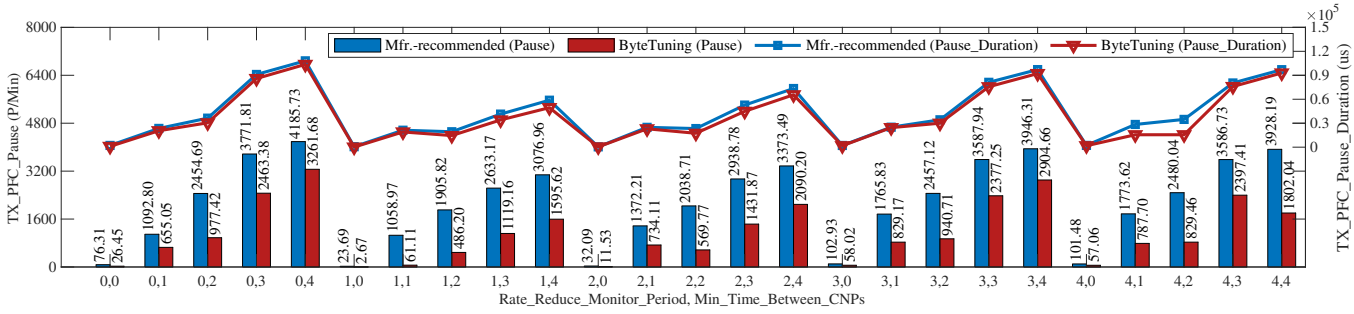
Fig. 12. The tuning result when 39-to-1 RDMA Incast

TABLE 5
The tuning result of Redis storage

| | Redis over DCTCP | Redis over RDMA (Experience-based) | Redis over RDMA (Manufacturer-recommended) | Redis over RDMA (ByteTuning) |
|---|---|---|---|---|
| Throughput (Gbps)/IOPS (K) | 14.41 / 25.51 | 20.73 / 39.11 | 21.41 / 41.03 | **22.59 / 43.20** |
| FCT (ms) | 15.60 | 11.23 | 9.86 | **9.16** |
| Queue (Cells) | 1.16 M | 8.23 K | 4.81 K | **2.53 K** |
| ECN-marked packets | **3.79 K** | 12.80 K | 6.72 K | 4.97 K |




(a) ECN-marked packets and throughput on the Multi-host RNIC




(b) TX-PFC Pause and Duration on the Multi-host RNIC

Fig. 13. The tuning result on the Multi-host RNIC

TABLE 6
The tuning result of throughput and fairness on the Multi-host RNIC

| | No-VoQ | VoQ (Mfr.-recommended) | ByteTuning |
|---|---|---|---|
| A→C_1, 500 QP | 25.1 Gbps | 25.2 Gbps | 25.2 Gbps |
| B→C_1, 500 QP | 24.8 Gbps | 25.0 Gbps | 25.0 Gbps |
| B→C_2, 1 QP | 3.5 Gbps | 22.6 Gbps | **41.0 Gbps** |

TABLE 7
The overhead from different network telemetry tools of ByteTuning

| | SNMP | sFlow | ONT | INT |
|---|---|---|---|---|
| Bandwidth | 11.2 Mbps | 16.6 Mbps | 13.78 Mbps | 10.2 Mbps |
| Frequency | 0.03 | 1/16384 (Sampling ratio) | 1 | 1000 |

It is an interesting solution to optimize the watermark configuration of the RoCEv2 switch cluster through deep reinforcement learning, but we find a important problem when deploying ACC. To sum it up, it is very easy to disrupt

the rhythm of watermark tuning by trying to reduce the communication overhead between switches. Due to the lack of synchronous tuning beats such as centralized ones, there is a lack of interaction between ACC switches, resulting in fast tuning but slow convergence. In other words, the newly configured watermark of switch #1 will affect the throughput and queue of switch #2 connected in series, causing switch #2 to mistakenly think that its own watermark is inappropriate, although it may be the near-optimal watermark. Therefore, the core difference between distributed and centralized is the trade-off of communication overhead (whether synchronization between switches is required). At this stage, RoCEv2 network telemetry has become common, and ByteTuning should be the most appropriate engineering practice.

In fact, any centralized system should consider its scalability. Key performance bottlenecks for ByteTuning include the Telemetry / Measurement Server and ByteTuning Server. For the former, the industry usually uses distributed technology to build, which has strong scalability; for the latter, the number of tuning tasks is matched with the computing tasks, and the tuning overhead is very small, so the scalability is also very strong.

TABLE 8
Comparison of distributed and centralized watermark tuning

| | **Distributed** | **Centralized** |
|---|---|---|
| Solution | ACC | ByteTuning |
| Algorithm | Deep reinforcement learning | Simulated annealing |
| Tuning object | ECN | ECN&PFC |
| Min tuning interval | 1 RTT | 1 RTT |
| Deployability | For incremental switches, retrofitting the existing switch to have sufficient computing power. | For incremental and existing switches, avoiding to retrofit switch and reducing deployment costs. |
| Pros&Cons | (1) Fast tuning and low overhead,<br>(2) Inter-switch interference and slow convergence,<br>(3) High computation overhead. | (1) Introducing in-band telemetry to reduce feedback overhead,<br>(2)High tuning latency and cost. |

## 8 Conclusion and Future Work

In this paper, we illustrate the necessity of RoCEv2 watermark tuning including but not limited to case analysis and model simulation, and introduce the detailed design of the centralized tuning system ByteTuning. Experiment results show that it is sufficiently comprehensive, efficient and reliable. In the future, we will combine the advantages of existing solutions and try to apply collaborative multi-objective multi-agent reinforcement learning to the ByteTuning algorithm to solve the trade-off problem of distributed and centralized concerns. In addition, a direction worthy of follow-up is to further improve the flexibility and adaptability of ByteTuning, and improve network performance for more complex RDMA applications.

## Acknowledgments

The authors are grateful to the editor and anonymous reviewers for helpful comments and suggestions that helped us improve the technical and presentation quality of the paper significantly.

## References

[1] J. Hu, H. Shen, X. Liu, and J. Wang, "RDMA Transports in Datacenter Networks: Survey," *IEEE Network*, pp. 1–8, 2024.

[2] A. Kalia, M. Kaminsky, and D. G. Andersen, "Design Guidelines for High Performance RDMA Systems," in *Proceedings of ATC'16*. Denver, CO: USENIX, Jun. 2016, pp. 437–450.

[3] S. Ma, T. Ma, K. Chen, and Y. Wu, "A Survey of Storage Systems in the RDMA Era," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4395–4409, 2022.

[4] J. Xue, Y. Miao, C. Chen, M. Wu, L. Zhang, and L. Zhou, "Fast Distributed Deep Learning over RDMA," in *Proceedings of EuroSys'19*. Dresden, Germany: ACM, Mar. 2019, pp. 1–14.

[5] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn, "RDMA over Commodity Ethernet at Scale," in *Proceedings of SIGCOMM'16*. Florianopolis, Brazil: ACM, Aug. 2016, pp. 202–215.

[6] W. Bai, S. S. Abdeen, A. Agrawal, K. K. Attre, P. Bahl, A. Bhagat, G. Bhaskara, T. Brokhman, L. Cao, A. Cheema, R. Chow, J. Cohen, M. Elhaddad, V. Ette, I. Figlin, D. Firestone, M. George, I. German, L. Ghai, E. Green, A. Greenberg, M. Gupta, R. Haagens, M. Hendel, R. Howlader, N. John, J. Johnstone, T. Jolly, G. Kramer, D. Kruse, A. Kumar, E. Lan, I. Lee, A. Levy, M. Lipshteyn, X. Liu, C. Liu, G. Lu, Y. Lu, X. Lu, V. Makhervaks, U. Malashanka, D. A. Maltz, I. Marinos, R. Mehta, S. Murthi, A. Namdhari, A. Ogus, J. Padhye, M. Pandya, D. Phillips, A. Power, S. Puri, S. Raindel, J. Rhee, A. Russo, M. Sah, A. Sheriff, C. Sparacino, A. Srivastava, W. Sun, N. Swanson, F. Tian, L. Tomczyk, V. Vadlamuri, A. Wolman, Y. Xie, J. Yom, L. Yuan, Y. Zhang, and B. Zill, "Empowering azure storage with RDMA," in *Proceedings of NSDI'23*. Boston, MA: USENIX, Apr. 2023, pp. 49–67.

[7] D. Gibson, H. Hariharan, E. Lance, M. McLaren, B. Montazeri, A. Singh, S. Wang, H. M. G. Wassel, Z. Wu, S. Yoo, R. Balasubramanian, P. Chandra, M. Cutforth, P. Cuy, D. Decotigny, R. Gautam, A. Iriza, M. M. K. Martin, R. Roy, Z. Shen, M. Tan, Y. Tang, M. Wong-Chan, J. Zbiciak, and A. Vahdat, "Aquila: A unified, low-latency fabric for datacenter networks," in *Proceedings of NSDI'22*. Renton, Washington, USA: USENIX, Apr. 2022, pp. 1249–1266.

[8] Y. Gao, Q. Li, L. Tang, Y. Xi, P. Zhang, W. Peng, B. Li, Y. Wu, S. Liu, L. Yan, F. Feng, Y. Zhuang, F. Liu, P. Liu, X. Liu, Z. Wu, J. Wu, Z. Cao, C. Tian, J. Wu, J. Zhu, H. Wang, D. Cai, and J. Wu, "When Cloud Storage Meets RDMA," in *Proceedings of NSDI'21*. Virtual Event, USA: USENIX, Apr. 2021, pp. 519–533.

[9] Z. Wang, T. Ma, L. Kong, Z. Wen, J. Li, Z. Song, Y. Lu, G. Chen, and W. Cao, "Zero Overhead Monitoring for Cloud-native Infrastructure using RDMA," in *Proceedings of ATC'22*. Carlsbad, CA: USENIX, Jul. 2022, pp. 639–654.

[10] K. Gao, C. Sun, S. Wang, D. Li, Y. Zhou, H. H. Liu, L. Zhu, and M. Zhang, "Buffer-based End-to-end Request Event Monitoring in the Cloud," in *Proceedings of NSDI'22*. Renton, WA: USENIX, Apr. 2022, pp. 829–843.

[11] Z. He, D. Wang, B. Fu, K. Tan, B. Hua, Z.-L. Zhang, and K. Zheng, "MASQ: RDMA for virtual private cloud," in *Proceedings of SIGCOMM'20*. Virtual Event, USA: ACM, Jul. 2020, pp. 1–14.

[12] X. Kong, Y. Zhu, H. Zhou, Z. Jiang, J. Ye, C. Guo, and D. Zhuo, "Collie: Finding Performance Anomalies in RDMA Subsystems," in *Proceedings of NSDI'22*. Renton, Washington, USA: USENIX, Apr. 2022, pp. 287–305.

[13] W. Li, X. Liu, Y. Li, Y. Jin, H. Tian, Z. Zhong, G. Liu, Y. Zhang, and K. Chen, "Understanding communication characteristics of distributed training," in *Proceedings of APNet'24*. Sydney, Australia: ACM, Aug. 2024, pp. 1–8.

[14] Z. Guo, S. Liu, and Z. Zhang, "Traffic control for RDMA-enabled data center networks: A survey," *IEEE Systems Journal*, vol. 14, no. 1, pp. 677–688, 2019.

[15] IEEE, "IEEE 802 Nendica Report: The Lossless Network for Data Centers," pp. 1–29, 2018.

[16] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, "Congestion Control for Large-Scale RDMA Deployments," in *Proceedings of SIGCOMM'15*. London, United Kingdom: ACM, Aug. 2015, pp. 523–536.

[17] J. Huang, J. Liu, N. Jiang, S. Liu, J. Hu, and J. Wang, "Achieving fast convergence and high efficiency using differential explicit feedback in data center," *IEEE Transactions on Cloud Computing*, vol. 11, no. 3, pp. 2312–2324, 2022.

[18] Z. Chen, M. Zhang, G. Li, and M. Xu, "Poster: Chameleon: Automatic and Adaptive Tuning for DCQCN Parameters in RDMA Networks," in *Proceedings of SIGCOMM'23*. New York, NY, USA: ACM, 2023, p. 1091–1093.

[19] 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 SIGCOMM'21*, Virtual Event, USA, Aug. 2021, pp. 384–397.

[20] J. Hu, J. Huang, W. Lyu, W. Li, Z. Li, W. Jiang, J. Wang, and T. He, "Adjusting switching granularity of load balancing for heterogeneous datacenter traffic," *IEEE/ACM Transactions on Networking*, vol. 29, no. 5, pp. 2367–2384, 2021.

[21] P. MacArthur, Q. Liu, R. D. Russell, F. Mizero, M. Veeraraghavan, and J. M. Dennis, "An integrated tutorial on InfiniBand, verbs, and MPI," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2894–2926, 2017.

[22] InfiniBand Trade Association, "Supplement to InfiniBand architecture specification volume 1 release 1.2.2 annex A 17: RoCEv2 (IP routable RoCE)," 2014.

[23] J. Hu, Y. He, W. Luo, J. Huang, and J. Wang, "Enhancing load balancing with in-network recirculation to prevent packet reordering in lossless data centers," *IEEE/ACM Transactions on Networking*, pp. 1–14, 2024.

[24] K. Ramakrishnan, S. Floyd, and D. Black, "RFC 3168: The addition of explicit congestion notification (ECN) to IP," 2001.

[25] J. Hu, C. Zeng, Z. Wang, J. Zhang, K. Guo, H. Xu, J. Huang, and K. Chen, "Load balancing with multi-level signals for lossless datacenter networks," *IEEE/ACM Transactions on Networking*, vol. 32, no. 3, pp. 2736–2748, 2024.

[26] K. Liu, C. Tian, Q. Wang, Y. Chen, B. Tian, W. Sun, K. Meng, L. Yan, L. Han, J. Fu, W. Dou, and G. Chen, "PayDebt: Reduce Buffer Occupancy Under Bursty Traffic on Large Clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4707–4722, 2022.

[27] J. Ye, Y. Peng, Y. Li, and J. Huang, "Proactive buffer management of shared-memory switches for distributed deep learning," in *Proceedings of APNet'24*. Sydney, Australia: ACM, Aug. 2024, pp. 183–184.

[28] D. Shan, G. Peng, S. Ren, J. Ma, S. Long, and Y. Tang, "Adaptive approximate fair queueing for shared-memory programmable switches," *IEEE Transactions on Network Science and Engineering*, vol. 11, no. 4, pp. 3563–3576, 2024.

[29] Z. Li, J. Huang, S. Wang, and J. Wang, "Achieving Low Latency for Multipath Transmission in RDMA Based Data Center Network," *IEEE Transactions on Cloud Computing*, vol. 12, no. 1, pp. 337–346, 2024.

[30] Y. Ren, X. Sun, K. Li, J. Lin, S. Feng, Z. Ren, J. Yin, and Z. Qi, "Dissecting the Workload of Cloud Storage System," in *Proceedings of ICDCS'22*. Bologna, Italy: IEEE, Jul. 2022, pp. 647–657.

[31] Y. Jiang, Y. Zhu, C. Lan, B. Yi, Y. Cui, and C. Guo, "A Unified Architecture for Accelerating Distributed DNN Training in Heterogeneous GPU/CPU Clusters," in *Proceedings of OSDI'20*. Virtual Event, USA: USENIX, Nov. 2020, pp. 463–479.

[32] V. Addanki, W. Bai, S. Schmid, and M. Apostolaki, "Reverie: Low pass Filter-Based switch buffer sharing for datacenters with RDMA and TCP traffic," in *Proceedings of NSDI'24*. Santa Clara, CA: USENIX, Apr. 2024, pp. 651–668.

[33] X. Kong, J. Chen, W. Bai, Y. Xu, M. Elhaddad, S. Raindel, J. Padhye, A. R. Lebeck, and D. Zhuo, "Understanding {RDMA} microarchitecture resources for performance isolation," in *Proceedings of NSDI'23*. Boston, MA: USENIX, Apr. 2023, pp. 31–48.

[34] S. Agarwal, A. Krishnamurthy, and R. Agarwal, "Host congestion control," in *Proceedings of SIGCOMM'23*. New York, NY, USA: ACM, 2023, pp. 275–287.

[35] K. Liu, J. Zhang, Z. Jiang, H. Wei, X. Zhong, L. Tan, T. Pan, and T. Huang, "Diagnosing End-Host Network Bottlenecks in RDMA Servers," *IEEE/ACM Transactions on Networking*, pp. 1–15, 2024.

[36] P. Yébenes, G. Maglione-Mathey, J. Escudero-Sahuquillo, P. J. García, and F. J. Quiles, "Modeling a switch architecture with virtual output queues and virtual channels in hpc-systems simulators," in *Proceedings of HPCS'16*. Innsbruck, Austria: IEEE, July 2016, pp. 380–386.

[37] J. Lou, X. Kong, J. Huang, W. Bai, N. S. Kim, and D. Zhuo, "Harmonic: Hardware-assisted RDMA performance isolation for public clouds," in *Proceedings of NSDI'24*. Santa Clara, CA: USENIX, Apr. 2024, pp. 1479–1496.

[38] S. Hu, Y. Zhu, P. Cheng, C. Guo, K. Tan, J. Padhye, and K. Chen, "Deadlocks in Datacenter Networks: Why Do They Form, and How to Avoid Them," in *Proceedings of HotNets'16*. Atlanta, GA, USA: ACM, Nov. 2016, pp. 92–98.

[39] J. Hu, J. Huang, Z. Li, J. Wang, and T. He, "A receiver-driven transport protocol with high link utilization using anti-ecn marking in data center networks," *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 1898–1912, 2022.

[40] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center TCP (DCTCP)," in *Proceedings of SIGCOMM'10*. New Delhi, India: ACM, Aug. 2010, pp. 63–74.

[41] H. Wu, J. Ju, G. Lu, C. Guo, Y. Xiong, and Y. Zhang, "Tuning ECN for Data Center Networks," in *Proceedings of CoNEXT'12*. Nice, France: ACM, Dec. 2012, pp. 25–36.

[42] W. Bai, L. Chen, K. Chen, and H. Wu, "Enabling ECN in multi-service multi-queue data centers," in *Proceedings of NSDI'16*. Santa Clara, CA: USENIX Association, Mar. 2016, pp. 537–549.

[43] J. Zhang, W. Bai, and K. Chen, "Enabling ecn for datacenter networks with rtt variations," in *Proceedings of CoNEXT'19*. Orlando, Florida: ACM, Dec. 2019, pp. 233–245.

[44] 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.

[45] D. Shan and F. Ren, "ECN marking with micro-burst traffic: Problem, analysis, and improvement," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1533–1546, 2018.

[46] A. Majidi, X. Gao, N. Jahanbakhsh, S. Jamali, J. Zheng, and G. Chen, "Deep-RL: Deep Reinforcement Learning for Marking-Aware via per-Port in Data Centers," in *Proceedings of ICPADS'19*. Tianjin, China: IEEE, Dec. 2019, pp. 392–395.

[47] A. Majidi, N. Jahanbakhsh, X. Gao, J. Zheng, and G. Chen, "DC-ECN: A machine-learning based dynamic threshold control scheme for ECN marking in DCN," *Computer Communications*, vol. 150, pp. 334–345, 2020.

[48] W. Lyu, J. Huang, J. Liu, Z. Li, S. Zou, W. Li, J. Wang, and D. Zhang, "Mitigating port starvation for shallow-buffered switches in datacenter networks," in *Proceedings of ICDCS'21*, DC, USA, Jul. 2021, pp. 921–931.

[49] J. Huang, S. Zhou, Z. Li, Y. Li, Z. Chen, X. Zhu, J. Shao, S. Li, W. Jiang, J. Wang *et al.*, "Coupling congestion control and flow pausing in data center network," in *Proceedings of ICPP'24*. Gotland, Sweden: ACM, Aug. 2024, pp. 1247—-1256.

[50] Y. Chen, C. Tian, J. Dong, S. Feng, X. Zhang, C. Liu, P. Yu, N. Xia, W. Dou, and G. Chen, "Swing: Providing long-range lossless RDMA via pfc-relay," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 1, pp. 63–75, 2022.

[51] C. Tian, B. Li, L. Qin, J. Zheng, J. Yang, W. Wang, G. Chen, and W. Dou, "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.

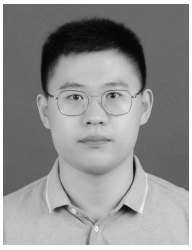
[52] Z. Cui and S. Y. Rim, "G-PFC: A Packet-Priority Aware PFC Scheme for the Datacenter," in *Proceedings of APNOMS'20*, Daegu, South Korea, Sep. 2020, pp. 385–388.

[53] D. Shan, Y. Liu, T. Zhang, Y. Liu, Y. Tang, H. Li, and P. Zhang, "Less is more: Dynamic and shared headroom allocation in pfc-enabled datacenter networks," in *Proceedings of ICDCS'23*. Hong Kong, China: IEEE, July 2023, pp. 591–602.

[54] L. Tan, "ByteTuning simulation code," 2022. [Online]. Available: https://github.com/lzhtan/Fluid-Model-for-DCQCN-ECN-PFC

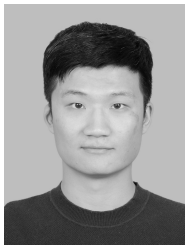
[55] D. Shan, P. Zhang, W. Jiang, H. Li, and F. Ren, "Towards the fairness of traffic policer," in *Proceedings of INFOCOM'21*. Vancouver, BC, Canada: IEEE, May 2021, pp. 1–10.

[56] L. Tan, "ByteTuning code," 2022. [Online]. Available: https://github.com/lzhtan/ByteTuning
[57] L. Tan, W. Su, W. Zhang, J. Lv, Z. Zhang, J. Miao, X. Liu, and N. Li, "In-band network telemetry: A survey," *Computer Networks*, vol. 186, p. 107763, 2021.
[58] B. Cassell, T. Szepesi, B. Wong, T. Brecht, J. Ma, and X. Liu, "Nessie: A Decoupled, Client-Driven Key-Value Store Using RDMA," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 12, pp. 3537–3552, 2017.



**Lizhuang Tan** (S'18-M'22) received his Ph.D. degree from School of Electronic and Information Engineering, Beijing Jiaotong University, P. R. China in 2022. He is currently an Associate Professor with Shandong Provincial Key Laboratory of Computer Networks, Shandong Computer Science Center (National Supercomputer Center in Ji'nan), Qilu University of Technology (Shandong Academy of Sciences). His research interests include network measurement, management and optimization, especially software defined networking and datacenter networking.

**Zhuo Jiang** received his B.E. degree from the Beijing University of Posts and Telecommunications, P. R. China in 2012, and Ph.D. degree from Tsinghua University, P. R. China in 2018. He is currently working at Douyin Vision Co., Ltd. He is interested in building networked system with high performance and high reliability.



**Kefei Liu** received his B.E. degree from Beijing University of Posts and Telecommunications (BUPT), P. R. China in 2019. He is currently a Ph.D. student in BUPT and Research Intern in Douyin Vision Co., Ltd. He has published papers in ACM SIGCOMM, USENIX NSDI, IEEE GLOBECOM and IEEE/ACM ToN. His research interests include datacenter networking and RDMA.

**Haoran Wei** received his B.S. and M.S. degrees from Beijing University of Posts and Telecommunications, P. R. China in 2019 and 2022. He is currently working at Douyin Vision Co., Ltd. as a software engineer. His research interests include datacenter networking, low-latency networking, and RDMA-based network systems.

**Pengfei Huo** received his B.E.degree in communication engineering from Xinjiang University, P. R. China in 2015. He successively served as senior network engineer in Juniper, Baidu. He is currently working at Douyin Vision Co., Ltd. as a software engineer. He focuses on the architecture design of data center and backbone network, In particular, he has rich experience in the optimization and deployment of high-performance networks and supercomputing services.

**Huiling Shi** received her M.S. degree from Shandong University, P. R. China in 2004. She is currently an Associate Researcher with Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences). Her research interests include network architectures and supercomputing network.

**Wei Zhang** (M'17) received his B.E. degree from Zhejiang University, P. R. China in 2004, M.S. degree from Liaoning University, P. R. China in 2008, and Ph.D. degree from Shandong University of Science and Technology, P. R. China in 2018. He is currently a Professor with Shandong Provincial Key Laboratory of Computer Networks, Shandong Computer Science Center (National Supercomputer Center in Ji'nan), Qilu University of Technology (Shandong Academy of Sciences). His research interests include future generation network architecture, edge computing and edge intelligence.

**Wei Su** received his B.S., M.S., and Ph.D. degrees in communication and information systems from Beijing Jiaotong University (BJTU), P. R. China in 2001, 2004 and 2008. He is a Full Professor at School of Electronic and Information Engineering, BJTU. He has studied Internet for more than 20 years. He was selected for the Beijing Young Talents Program in 2013. He won the second prize of National Technology Invention Award in 2014. He was a visiting scholar with Future University, Hakodate, Japan in 2015. He has published more than 50 research papers and 4 monographs in areas of communications and computer networks. His research interests are future generation network architecture and mobile Internet.