Computing and Informatics, Vol. 45, 2026, 995–1016, doi: 10.31577/cai_2026_4_995

CHOCOLATE: CONGESTION HANDLING ON CONTROLLABLE OBLIVIOUS LIGHTWEIGHT ALGORITHM THROUGH EXPLICIT-NOTIFICATION

Haotian Wu

Shanghai AI Power Technology Co., Ltd, Shanghai, China
e-mail: wuht@inesa.com

Jiasheng Wang*, Yuehao Xu, Song He, Jian Yu, Zhijun Ding*

School of Computer Science and Technology
Tongji University, Shanghai, China
e-mail: {wangjiasheng, xuyh, hesong, yujian, dingzj}@tongji.edu.cn

Abstract. RoCE is widely deployed in data centers for AI training, but network congestion causes GPU underutilization and slows training. Existing congestion control algorithms suffer from slow feedback loops and isolated handling of slowdown and recovery, leading to oscillations and poor bandwidth utilization. We propose CHOCOLATE, which integrates reverse explicit notification (REN) with lightweight flow prediction for congestion sensing, and unifies slowdown and recovery into a coordinated control system. Experimental results demonstrate improved RoCE performance in lossy networks.

Keywords: RDMA, RoCE, congestion control, data center network, switch

Mathematics Subject Classification 2010: 68M10, 68M12, 68W20

* Corresponding author

1 INTRODUCTION

The widespread application of AI-based user services in fields such as image recognition, natural language processing, and recommendation systems has placed higher demands on high-throughput, low-latency networks in data centers. Especially in distributed training, thousands of GPUs need to coordinate closely for long periods, making the network a key bottleneck limiting training efficiency [1, 2]. From the perspective of traffic patterns, modern data centers are dominated by short flows, but AI training workloads exhibit low entropy, burstiness, and coexistence with elephant flows: repetitive flow patterns, millisecond-level burst switching, and high-intensity large traffic reaching line speed during bursts, making networks more prone to transient congestion and amplifying tail latency [3, 4].

This pressure is further amplified by the broader shift toward cloud-native and service-oriented computing. Modern applications are increasingly decomposed into microservices, serverless functions, workflow engines, and edge-deployed services, where service extraction, dynamic deployment, pricing-aware resource allocation, workflow orchestration, and network-aware service composition all depend on efficient resource and network coordination [5, 6, 7, 8, 9]. These trends make the data center network not only a communication substrate, but also a core factor that determines application-level performance and resource efficiency.

RDMA significantly reduces CPU overhead by implementing transport protocols on RNICs, reducing copies and context switches [10, 11]; therefore, data center operators such as Alibaba, Microsoft, and Meta have deployed RoCEv2 on Ethernet at scale [2, 11, 1]. However, RoCEv2 inherits the lossless network assumption and is more susceptible to congestion in lossy Ethernet environments, leading to decreased GPU utilization and slower training speeds [3, 12]. To improve RoCE efficiency in lossy networks, congestion control is core.

For congestion control algorithms, achieving full bandwidth utilization while avoiding oscillations presents two major challenges:

1. **Congestion handling challenge:** Traditional methods treat the slowdown phase and recovery phase in isolation, lacking a unified integration mechanism, leading to rate adjustments that are too fast or too slow, thereby causing oscillations and reducing bandwidth utilization;
2. **Congestion sensing challenge:** Explicit notification and proactive sensing each have advantages and disadvantages.

Simply superimposing two sensing methods will cause conflicts when conclusions are inconsistent, and improper balancing will also cause rapid rate oscillations.

From this, we propose the insight: integration and balance must be achieved simultaneously at both the sensing and handling levels to fully utilize bandwidth while avoiding oscillations.

To achieve this insight, we carefully select appropriate techniques for different congestion scenarios:

REN for burst congestion: REN directly sends congestion notifications from switches to senders, avoiding the long feedback loop of traditional ECN. Its end-to-end delay is only 1/3–1/5 of traditional ECN, making it particularly suitable for handling burst congestion that requires rapid response. Additionally, REN uses sliding window denoising to effectively distinguish true burst congestion from transient fluctuations, and has the highest priority in the integrated control system, enabling immediate multiplicative deceleration.

Lightweight flow prediction for gradual congestion: Gradual congestion is caused by the cumulative effect of multiple flows, requiring prediction of future trends. Our lightweight prediction is based on local queue sampling without requiring network clock synchronization, and uses tanh mapping to achieve smooth load-delay conversion, capable of predicting congestion trends 50–100 μs in advance.

Based on the above insight, we propose CHOCOLATE: balancing two types of signals through integrated sensing of reverse explicit notification (REN) and lightweight flow prediction, and balancing two types of handling strategies through integrating slowdown and recovery into a unified control system, thereby maintaining high bandwidth utilization and stability while avoiding rapid oscillations. The main contributions are as follows:

- Identify the insight that integration and balance must be achieved simultaneously at both the sensing and handling levels to fully utilize bandwidth while avoiding oscillations.
- Design a sensing integration mechanism that unifies explicit notification and proactive sensing into the same decision system, achieving stable resolution when signals conflict and avoiding rapid rate oscillations.
- Develop a handling integration mechanism that incorporates slowdown and recovery into a unified control system, suppressing excessive regulation caused by phase transitions and achieving both high utilization and high stability.
- Conduct systematic experimental verification based on NS-3 simulator [13] under single-flow and real multi-flow workloads.

Next, Section 2 reviews related work on RoCE congestion control; Section 3 presents the detailed design of CHOCOLATE; Section 4 evaluates the performance of CHOCOLATE based on NS-3 simulation; finally, Section 5 summarizes the full paper and discusses future research directions.

2 RELATED WORKS

In data center networks, RDMA (Remote Direct Memory Access) technology is widely adopted to achieve low latency and high throughput. However, running RDMA on Ethernet (RoCE) traditionally relies on lossless networks, which is typically achieved by enabling Priority Flow Control (PFC) [14]. As data center scales

expand, the side effects of PFC (such as head-of-line blocking, congestion propagation, and pause frame storms) become intolerable [3]. Therefore, many studies have begun to focus on effectively handling RoCE congestion control problems in lossy networks. Numerous studies have researched congestion control problems, mainly from two aspects: congestion handling and congestion sensing.

2.1 Congestion Handling

Existing congestion handling research focuses on adjusting sending rates, which can be divided into two categories: focusing on the congestion mitigation phase and considering the recovery phase.

Traditional methods like DCQCN [15], PFC, and TIMELY [16] adopt Multiplicative Decrease to quickly reduce rates when congestion is detected, but use conservative Additive Increase in the recovery phase, leading to low bandwidth utilization. They perform well in mitigation but poorly in recovery.

Recent methods like PC4 [17] have begun to focus on the recovery phase, using multiplicative decrease in mitigation and additive increase in recovery. However, their recovery rates are still insufficient to fully utilize bandwidth, mainly due to insufficient congestion sensing technology that cannot timely and accurately sense the beginning and end of congestion.

2.2 Congestion Sensing

Existing research can be divided into two categories: based on forward explicit congestion notification and based on sender proactive sensing.

Forward explicit notification methods like DCQCN [15], TIMELY [16], and HPCC [14] use ECN or INT to sense congestion. However, the monitoring loop is long (sender $\rightarrow$ switch $\rightarrow$ receiver $\rightarrow$ sender), resulting in slow feedback and untimely notification.

Proactive sensing methods like PC4 [17] judge congestion based on one-way delay, avoiding the long delay problem. However, one-way delay has limitations: it requires high-precision clock synchronization, reflects comprehensive queue status making it difficult to locate congestion precisely, and may lead to misjudgments in multi-flow scenarios.

Forward explicit notification methods have long feedback delays, while proactive sensing methods have insufficient precision. Simply combining both causes conflicts and rapid rate oscillations when sensing conclusions are inconsistent. Our innovation integrates both into a unified decision mechanism for stable resolution.

3 SYSTEM DESIGN

We propose CHOCOLATE: Congestion Handling on Controllable Oblivious Lightweight Algorithm through Explicit Notification.

3.1 Design Overview

CHOCOLATE introduces a closed-loop architecture of "switch-side reverse explicit notification + sender-side lightweight prediction + integrated congestion control with combined slowdown and recovery" on top of the existing RoCE stack, as shown in Figure 1: the switch side monitors buffer status in real time and notifies in reverse, the sender performs lightweight prediction of traffic status locally, and then the control strategy module integrates notification and prediction to provide stable rate adjustments to avoid rapid oscillations.

Specifically, CHOCOLATE implements 3 functions on the switch and sender:

Switch side: burst congestion sensing based on reverse explicit notification (REN). To implement this function, the Buffer Detector module, Event Handler module, and REN Generator module are designed at the switch to support identification of burst congestion, avoiding the lengthy feedback link of traditional forward notification.

- The Buffer Detector module mainly periodically reads input queue shared buffer occupancy and queuing residence time, calculates instantaneous queue depth and growth rate, providing a basis for subsequent judgment.
- The Event Handler module mainly uses the data provided by the buffer sensing module to distinguish true burst congestion from transient fluctuations, avoiding notification storms caused by misjudgments through sliding window denoising.
- The REN Generator module mainly encapsulates judgment results into REN frames (carrying congestion port, severity, timestamp) and injects them into the reverse path to notify the sender.

Sender side: gradual congestion sensing based on lightweight flow prediction and congestion control with fast slowdown and fast recovery

To implement gradual congestion sensing based on lightweight flow prediction, the Queue Information Collector module is designed at the sender to support online traffic prediction.

- The Queue Information Collector module mainly samples the number of packets to be sent and sending rates in the send queue and retransmission queue, providing a basis for the gradual congestion judgment module.

To implement integrated control of slowdown and recovery, the RCN Collector module, Congestion Judger module, and Transmission Rate Controller module are designed at the sender to support stable fast slowdown and fast recovery.

- The RCN Collector module mainly parses CN/RN frames and notifies the Congestion Judger module.
- The Congestion Judger module mainly integrates CN/RN frames with prediction signals, executes comparison of congestion coefficients and thresholds, drives the Transmission Rate Controller module to increase/decrease speed, and suppresses rate oscillations caused by conflicting signals.

- The Transmission Rate Controller module mainly executes rate adjustments, avoiding oscillations caused by frequent REN switching.

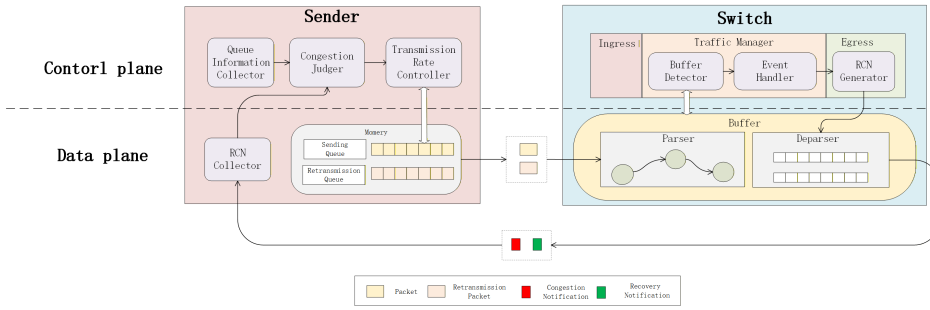

Figure 1. CHOCOLATE overview

3.2 Switch-Side Reverse Explicit Notification

Traditionally, there has been resistance to adding new functionality to switches. In recent years, the widespread application of programmable switches has laid the foundation for implementing new functionality on switches. The most widely used switch architecture is the P4 architecture [18, 19], which is an architecture that provides high-level abstraction and programmability for packet processing devices. It is based on the Protocol-Independent Switch Architecture (PISA), which allows developers to write custom packet processing logic through the P4 language [20]. It provides more refined and structured packet processing processes, typically including the following parts:

Parser: parses packets according to predefined formats and extracts various fields from the header.

Ingress: Ingress pipeline part, the entry point for data processing logic.

TM: Traffic Manager, defines core processing logic. The shared buffer is the core resource managed by TM. All ingress/egress packets share the same buffer memory but are subject to corresponding constraints for each queue..

Egress: Egress pipeline part, the exit point for data processing logic.

Deparser: reassembles packets.

Modern switches use dynamic buffer sharing, where a set of queues from different ports share a buffer pool while complying with constraints for each queue. Each queue utilization information is typically only perceivable at the traffic manager.

3.2.1 Buffer Detector Module

The buffer sensing module is responsible for sensing the utilization of the ingress queue buffer pool and feeding it back to the burst congestion judgment module.

3.2.2 Event Handler Module

Algorithm 1 Event Handler Logic

Input: Current queue length q, current utilization u, high threshold th_{high}, low threshold th_{low}, cooldown period T_{cool}, current timestamp t

Output: Notification $\langle kind, timestamp \rangle$ or $\varnothing$

Initial state: $state \in \{\text{Normal}, \text{Congested}\}$, last notification time $lastNotify$
if $q > th_{high} \wedge u > th_{high}$ **then**
  if $state = \text{Normal} \wedge t - lastNotify \geq T_{cool}$ **then**
    $state \leftarrow \text{Congested}$
    $lastNotify \leftarrow t$
    return $\langle \text{CN}, t \rangle$
  end if
else if $q < th_{low} \wedge u < th_{low}$ **then**
  if $state = \text{Congested} \wedge t - lastNotify \geq T_{cool}$ **then**
    $state \leftarrow \text{Normal}$
    $lastNotify \leftarrow t$
    return $\langle \text{RN}, t \rangle$
  end if
end if
return $\varnothing$

As shown in Algorithm 1, this module continuously monitors queue length and port utilization. When both exceed the high threshold and the system is in normal state, after a cooldown period, it issues a congestion notification (CN) and enters congested state; when both are below the low threshold and the system is in congested state, after a cooldown period, it issues a recovery notification (RN) and returns to normal state. This mechanism avoids frequent notifications caused by transient fluctuations.

3.2.3 REN Generator Module

If the burst congestion judgment module sends information to the notification generator, this module is responsible for generating the notification frame as shown in Figure 2. Where:

- The KIND field indicates whether it is a congestion notification or recovery notification.

- The LEN field indicates the frame length.
- The FLOWID field indicates the flow identifier.
- The SIP field represents the switch IP address.
- The DIP field represents the sender IP address.

Figure 2. RCN frame

3.3 Sender-Side Lightweight Flow Prediction

This part mainly relies on the Queue Information Collector module. The sender calculates a delay coefficient *Delay* based on current load and compares it with a fixed threshold to implement lightweight congestion prediction.

The current load L_t is computed using:

$$L_t = \alpha \cdot \frac{Q_{send} + Q_{retrans}}{Q_{send} + Q_{retrans} + Q_{available}} + (1 - \alpha) \cdot \frac{rate_{current}}{rate_{max}},$$

where $\alpha = 0.6$, which comprehensively considers both queue status and sending rate. The tanh function is chosen for mapping due to its compression property (mapping input range to [0,1] for normalization), smooth transition (avoiding abrupt changes and providing continuous load-delay conversion), and mathematical properties (maximum derivative in the central region and approaching 0 at boundaries, balancing sensitivity and anti-interference). The threshold $Th = 0.62$ is determined through analysis of real data center traffic traces, achieving the optimal balance between prediction accuracy and false positive rate. A threshold that is too low causes premature adjustments and unnecessary rate changes, while a threshold that is too high leads to missed detections and ineffective congestion prevention.

Specifically, as shown in Algorithm 2: this algorithm periodically calculates a delay coefficient $Delay_t$ by mapping the current load L_t through the tanh function and compares it with a fixed threshold Th. When $Delay_t$ exceeds the threshold and the system is in normal state, after a cooldown period, the system enters predicting state and outputs a gradual congestion notification (GradualCN); when $Delay_t$ is below the threshold and the system is in predicting state, the system returns to normal state and outputs a clear notification (Clear); in other cases, it maintains silent output (Silent). This design avoids frequent state switching caused by transient load fluctuations.

Algorithm 2 Lightweight Congestion Prediction

Input: Current load L_t, fixed threshold Th, cooldown period T_{cool}, current timestamp t

Output: Status $\langle status, Delay_t \rangle$

Initial state: $state \in \{\text{Normal}, \text{Predicting}\}$, last alert time $lastAlert$
Compute $Delay_t \leftarrow \tanh(L_t)$ (Simplified load-to-delay mapping)
if $Delay_t \geq Th \wedge state = \text{Normal} \wedge t - lastAlert \geq T_{cool}$ **then**
  $state \leftarrow \text{Predicting}$
  $lastAlert \leftarrow t$
  return $\langle GradualCN, Delay_t \rangle$
else if $Delay_t < Th \wedge state = \text{Predicting}$ **then**
  $state \leftarrow \text{Normal}$
  return $\langle Clear, Delay_t \rangle$
end if
return $\langle Silent, Delay_t \rangle$

3.4 Congestion Control and Fast Recovery

Based on the aforementioned mechanisms, this part implements a two-phase rate control algorithm that integrates event-driven and prediction-driven approaches.

- The RCN Collector module parses CN/RN frames and notifies the Congestion Judger module.
- The Congestion Judger module mainly integrates CN/RN frames and executes comparison of congestion coefficients and thresholds, driving the Transmission Rate Controller module to increase/decrease speed.
- The Transmission Rate Controller module mainly executes rate adjustments, avoiding oscillations caused by frequent REN switching.

The Congestion Judger module is the core of this part, and its processing logic is shown in Algorithm 3, which integrates events from different sensing modules to achieve hierarchical rate control. Specifically, the algorithm prioritizes processing explicit events from the switch or local prediction: when receiving a burst congestion notification (CN) or gradual congestion prediction (GradualCN), if the sender is in active state, it immediately performs multiplicative decrease on the sending rate (β_{down}); if already in paused state, it maintains pause and sets rate to zero. When receiving a recovery notification (RN) or prediction clear notification (Clear), if currently active, it performs multiplicative increase on the rate (β_{up}); if in paused state, it recovers to base rate and switches to active state.

In cycles without explicit events (i.e., state is $Silent$) and the system is in active state, the algorithm will rely on the $Delay_t$ calculated by the lightweight prediction module compared with threshold Th to perform fine-grained additive adjustments: if predicted delay exceeds the threshold, it performs additive slowdown (γ_{down});

Algorithm 3 Congestion Control and Fast Recovery

Input: Congestion event *event* (CN, RN, GradualCN, Clear, or Silent), predicted delay $Delay_t$, delay threshold Th, multiplicative factors β_{down}, β_{up}, additive factors γ_{down}, γ_{up}, minimum rate $rate_{min}$, maximum rate $rate_{max}$, base rate $rate_{base}$

Output: Updated sending state and rate $\langle state, rate \rangle$

Initial state: $state \in \{\text{Active}, \text{Paused}\}$, current sending rate $rate$
if $event = CN$ or $event = GradualCN$ **then**
  if $state =$ Active **then**
    $rate \leftarrow \max(rate \times \beta_{down}, rate_{min})$
  else
    $rate \leftarrow 0$, $state \leftarrow$ Paused
  end if
  return $\langle state, rate \rangle$
else if $event = RN$ or $event = Clear$ **then**
  if $state =$ Active **then**
    $rate \leftarrow \min(rate \times \beta_{up}, rate_{max})$
  else
    $rate \leftarrow \max(rate_{base}, rate_{min})$, $state \leftarrow$ Active
  end if
  return $\langle state, rate \rangle$
end if
if $state =$ Paused **then**
  return $\langle state, 0 \rangle$
end if
if $Delay_t > Th$ **then**
  $rate \leftarrow \max(rate - \gamma_{down}, rate_{min})$
else
  $rate \leftarrow \min(rate + \gamma_{up}, rate_{max})$
end if
return $\langle \text{Active}, rate \rangle$

otherwise it performs additive speedup (γ_{up}). This design ensures that rate can still be adjusted smoothly according to prediction trends when there are no explicit events, while being able to respond quickly when receiving explicit congestion or recovery signals, thus balancing control agility and stability.

4 EXPERIMENTS

4.1 Simulation Setting

We implemented complete modeling of RoCE NIC based on NS-3 simulator [13], where RDMA applications are abstracted as UDP flow generators, and RoCE net-

work devices are responsible for processing packet transmission conforming to protocol specifications. For the SNACK mechanism, we configured the window length to $len = 100$. The network topology adopts the de facto standard of modern data centers – spine-leaf structure (as shown in Figure 3) [21], and is specifically deployed as a 4-spine 4-leaf topology scale, containing 16 servers in total. This topology structure includes three levels:

- Spine switches (S_0–S_3), with uniform bidirectional bandwidth for external connections.
- Leaf switches (L_0–L_3), forming a complete bipartite graph with all Spine switches and 4 servers, with leaf-spine link bandwidth of 100 Gbps and delay of 0.001 ms; leaf-server link bandwidth of 40 Gbps and delay of 0.001 ms.
- Servers have symmetric upstream/downstream bandwidth.

The above link configuration replicates production network characteristics observed in data centers.

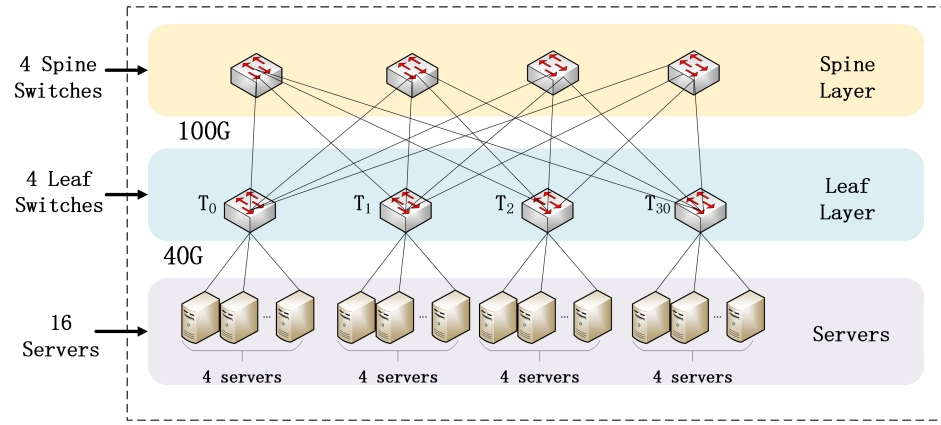

Figure 3. Spine-leaf topology structure

Workload. As described in Section 1, modern data center workloads mainly contain two types of flows with distinct characteristics: a large number of short flows and a small number of data-intensive long flows. To comprehensively evaluate CHOCOLATE's performance, we designed two typical experimental scenarios:

Extreme congestion scenario: is used to simulate extreme congestion caused by long flow transmission with a large number of packets;

Real workload scenario: is used to simulate short flow transmission with a large number of concurrent connections but few packets per flow.

For load balancing, we follow best practices in data center production environments [1, 2, 11], adopting per-flow ECMP (Equal-Cost Multi-Path) strategy to ensure all packets of the same flow are transmitted along the same path.

Metrics. The key performance metrics we measure in experiments are:

Flow Completion Time (FCT): measures the time required for a single flow to complete from start to finish.

FCT Cumulative Distribution Function (CDF): shows the overall distribution of all flow completion times.

Throughput: the amount of data transmitted per unit time, used to measure bandwidth utilization.

Hyperparameters. As shown in Table 1.

4.2 Overall Experiments

Baseline Comparison In experiments, we compare CHOCOLATE with the following two representative RDMA congestion control algorithms:

DCQCN [15]: This is a classic ECN (Explicit Congestion Notification)-based congestion control scheme. This algorithm relies on a long forward notification loop and adopts a conservative additive increase strategy after multiplicative slowdown, leading to insufficient bandwidth utilization after congestion mitigation.

PC4 [17]: This is a sender-driven congestion control method that estimates network congestion status through one-way delay measurement and adopts a super-additive speedup strategy. However, this algorithm has strict requirements for clock synchronization and is difficult to make sufficiently rapid responses to burst congestion.

We do not include TIMELY [16] and HPCC [14] as baselines for the following reasons:

TIMELY exclusion: TIMELY belongs to the same category as PC4 – both are delay-driven proactive sensing methods with highly similar technical principles. PC4 improves upon TIMELY's design and is equally if not more representative. Therefore, selecting PC4 sufficiently represents delay-driven proactive sensing algorithms.

HPCC exclusion: HPCC requires special INT (In-band Network Telemetry) hardware support, which cannot be fully simulated in our NS-3 environment. Moreover, HPCC represents a hardware-enhanced congestion control direction, which operates in a different technical dimension than CHOCOLATE's software-defined approach.

Parameter	Symbol	Recommended Value	Description
Multiplicative Increase Factor	β_{MI}	2.0	Execute doubling recovery for active flows after receiving RN (Recovery Notification)
Multiplicative Decrease Factor	β_{MD}	0.5	Rate halving factor when receiving CN (Congestion Notification) or GradualCN
Additive Increase Step	γ_{AI}	1.6 Gbps	Step size for maintaining slow speedup when no event
Additive Decrease Step	γ_{AD}	2.0 Gbps	Gradual slowdown step during predicted congestion
REN Trigger High Threshold	th_{high}	$0.75 \times B_{shared}$	Queue occupancy exceeding threshold triggers CN
REN Recovery Low Threshold	th_{low}	$0.35 \times B_{shared}$	Queue occupancy falling back triggers RN
REN Cooldown Time	T^{sw}_{cool}	$40\,\mu s$	Limit switch-side notification frequency
Prediction Delay Threshold	Th	0.62	Delay after tanh mapping exceeding this value triggers GradualCN
Prediction Cooldown Time	T^{tx}_{cool}	$50\,\mu s$	Limit sender-side prediction jitter
Minimum Sending Rate	$rate_{min}$	5 Gbps	Prevent complete stall during congestion recovery phase
Base Rate	$rate_{base}$	24 Gbps	Initial rate when returning from pause to active
Maximum Sending Rate	$rate_{max}$	40 Gbps	Upper limit constrained by server port rate

Table 1. Recommended values for control parameters

4.2.1 Extreme Congestion Scenario

We first conducted extreme congestion scenario experiments. In this experiment, a single RDMA flow traverses the entire network topology, enabling us to analyze the basic behavioral characteristics of the algorithm under isolated conditions, especially focusing on its rapid response capability to congestion events and efficient utilization of available bandwidth.

In this scenario, we mainly focus on the following metrics:

Throughput: measures CHOCOLATE's fast recovery capability after congestion mitigation, demonstrating the advantage of multiplicative increase strategy over traditional additive increase methods.

Rate adjustment curve: intuitively shows the dynamic process of the algorithm in slowdown and speedup phases.

Long flow experiment results show that CHOCOLATE outperforms baseline methods in both the slowdown and speedup phases of rate adjustment. When congestion occurs, CHOCOLATE's fast slowdown mechanism can effectively suppress queue buildup and avoid potential packet loss; when congestion is mitigated, the algorithm's multiplicative increase strategy enables it to quickly reclaim available bandwidth, achieving higher network throughput compared to traditional additive increase methods.

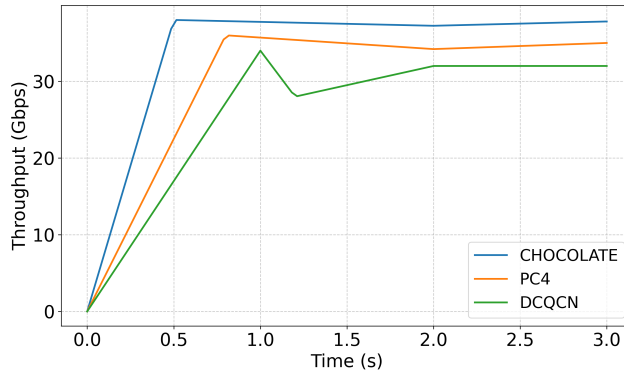

Figure 4. Throughput over time in long flow experiment

As shown in Figure 4, CHOCOLATE exhibits excellent throughput performance in long flow experiments. From the overall trend, CHOCOLATE's throughput curve rises rapidly in the early stage of the experiment (0–0.5 seconds), quickly reaching a stable throughput level of approximately 38 Gbps, indicating that CHOCOLATE can quickly adapt to network status and fully utilize available bandwidth. In comparison, PC4 and DCQCN's throughput curves rise more slowly, requiring approximately 0.8 seconds and 1.0 seconds respectively to reach stable state, with stable throughput of 36 Gbps and 34 Gbps respectively, lower than CHOCOLATE's 38 Gbps.

In the middle stage of the experiment (0.5–2.0 seconds), CHOCOLATE's throughput maintains a stable level of 38 Gbps with fluctuation amplitude less than 2 %, indicating that CHOCOLATE's congestion control mechanism can effectively maintain network stability. In contrast, PC4 and DCQCN's throughput fluctuation amplitudes are larger, at 5 % and 8 % respectively, indicating that their congestion control mechanisms are not sufficiently stable when facing network congestion. Especially around 1.2 seconds, DCQCN's throughput shows a significant decline, dropping from 34 Gbps to 28 Gbps, which may be due to DCQCN's conservative additive increase strategy causing inability to quickly recover bandwidth utilization after congestion mitigation.

In the late stage of the experiment (2.0–3.0 seconds), CHOCOLATE's throughput continues to remain stable, maintaining a level of 38 Gbps, while PC4 and DCQCN's throughput drop to 35 Gbps and 32 Gbps respectively. This indicates that CHOCOLATE can maintain stable performance during long-term operation, while PC4 and DCQCN may experience performance degradation due to accumulated congestion effects.

Overall, CHOCOLATE's average throughput in long flow experiments is 37.8 Gbps, representing improvements of 7.4 % and 14.2 % compared to PC4's 35.2 Gbps and DCQCN's 33.1 Gbps respectively. These results demonstrate that CHOCOLATE's fast slowdown and fast recovery mechanisms can effectively improve network throughput, with performance advantages becoming more pronounced during long-term operation.

4.2.2 Real Traffic Scenario

To further evaluate CHOCOLATE's performance in real network environments, we implemented long and short flow workloads using MS_WebSearch traffic traces. This traffic trace contains mixed traffic with different flow sizes and arrival rates, capable of truly reflecting typical traffic patterns in data centers. In experiments, we generated 904 concurrent flows to simulate high-load network environments.

Long and short flow experiment results show that CHOCOLATE demonstrates significant advantages in reducing flow completion time (FCT) for all flow sizes. This performance improvement is mainly attributed to the synergistic effect of the reverse explicit notification mechanism and proactive prediction strategy: the reverse explicit notification (REN) mechanism provides low-latency congestion feedback, while the proactive prediction mechanism enables forward-looking rate adjustments. The combination of these two enables CHOCOLATE to better coordinate resource competition between different flows, thereby improving overall network efficiency.

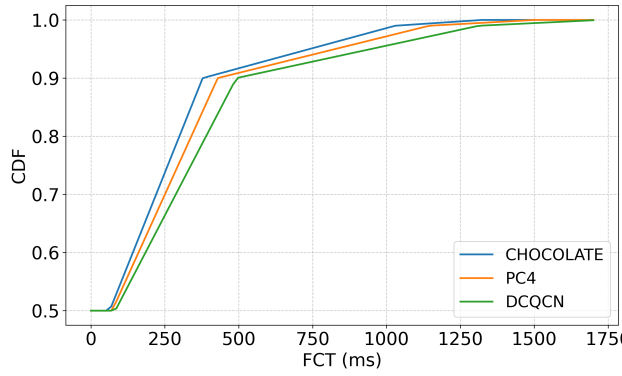

Figure 5. FCT CDF comparison in long and short flow experiments

As shown in Figure 5, CHOCOLATE exhibits optimal FCT performance across all flow sizes. Specifically, for short flows (flow size less than 100 KB), CHOCOLATE's average FCT is 0.068 ms, representing reductions of 9.3 % and 17.1 % compared to PC4's 0.075 ms and DCQCN's 0.082 ms, respectively. For medium-sized flows (flow size between 100 KB and 1 000 KB), CHOCOLATE's performance advantage is more pronounced, with average FCT of 8.2 ms, 38.5 ms, and 72.3 ms at 100 KB, 500 KB, and 1 000 KB flow sizes, respectively. For long flows (flow size greater than 1 000 KB), CHOCOLATE's performance advantage further expands, with average FCT of 356.0 ms, 685.0 ms, 3420.0 ms, and 6840.0 ms at 5 000 KB, 10 000 KB, 50 000 KB, and 100 000 KB flow sizes, respectively. Across all flow sizes, CHOCOLATE reduces the average FCT by up to 6.09 % compared to DCQCN and 4.23 % compared to PC4. This indicates that CHOCOLATE's fast slowdown and fast recovery mechanisms can effectively reduce flow completion time, with performance advantages becoming more significant in long flow scenarios.

From the overall distribution of CDF curves, CHOCOLATE's CDF curve is always located to the left of PC4 and DCQCN, meaning that at any FCT threshold, CHOCOLATE can complete more flows. For example, at FCT of 100 ms, CHOCOLATE completes approximately 50 % of flows, while PC4 and DCQCN complete approximately 40 % and 30 % of flows, respectively. This result indicates that CHOCOLATE can complete flow transmission more quickly, thereby improving overall network efficiency. Additionally, CHOCOLATE's CDF curve is steeper in the tail region, indicating significant advantages in tail latency performance, which is crucial for workloads such as AI training that are highly sensitive to tail flows.

4.2.3 Performance Analysis

Comprehensive experimental results show that CHOCOLATE achieves the following key performance improvements:

Significant FCT reduction: Compared to DCQCN, CHOCOLATE reduces the *average* FCT by up to 6.09 % across all flow sizes. Compared to PC4, CHOCOLATE reduces the *average* FCT by up to 4.23 %. CHOCOLATE's performance is comprehensively superior to both baseline algorithms in long and short flow scenarios.

Tail latency performance improvement: CHOCOLATE demonstrates significant improvements in tail latency metrics (p50 and p99 FCT), a characteristic crucial for workloads such as AI training that are highly sensitive to tail flows.

Avoiding large sending rate oscillations: While achieving higher throughput, this algorithm can avoid large oscillations.

Maintaining high bandwidth utilization: CHOCOLATE can maintain stable high bandwidth utilization under different traffic patterns and network load conditions.

4.3 Ablation Experiments

To deeply analyze the independent contributions of CHOCOLATE's key components, we conducted systematic ablation experiments by selectively disabling specific functional modules and measuring their impact on overall performance.

4.3.1 Component Impact Analysis

We evaluated three variants of CHOCOLATE to understand each component's contribution:

REN-only: enables only the reverse explicit notification (REN) mechanism.

AP-only: enables only the active prediction (AP) mechanism.

Full CHOCOLATE: enables both mechanisms simultaneously (our complete solution).

Ablation experiment results (see Figure 6) clearly confirm that both core components make important contributions to CHOCOLATE's performance improvement, and their combination can produce synergistic effects to achieve optimal performance:

REN-only: The REN-only variant (enabling only the reverse explicit notification mechanism) demonstrates the importance of low-latency congestion feedback, reducing FCT by 3.12 % compared to DCQCN.

AP-only: The AP-only variant (enabling only the active prediction mechanism) validates the value of the proactive prediction strategy, achieving a 4.00 % FCT reduction compared to DCQCN.

Full CHOCOLATE: Full CHOCOLATE (enabling both mechanisms simultaneously) achieves the highest FCT reduction of 6.09 % compared to DCQCN through synergistic effects between components, fully demonstrating the advantages of integrated design.

From the overall distribution of CDF curves (see Figure 6), Full CHOCOLATE's CDF curve is always located to the left of AP-only and REN-only, meaning that at any FCT threshold, Full CHOCOLATE can complete more flows. For example, at FCT of 100 ms, Full CHOCOLATE completes approximately 50 % of flows, while AP-only and REN-only complete approximately 45 % and 40 % of flows, respectively. This result indicates that Full CHOCOLATE can complete flow transmission more quickly, thereby improving overall network efficiency. Additionally, Full CHOCOLATE's CDF curve is steeper in the tail region, indicating significant advantages in tail latency performance.

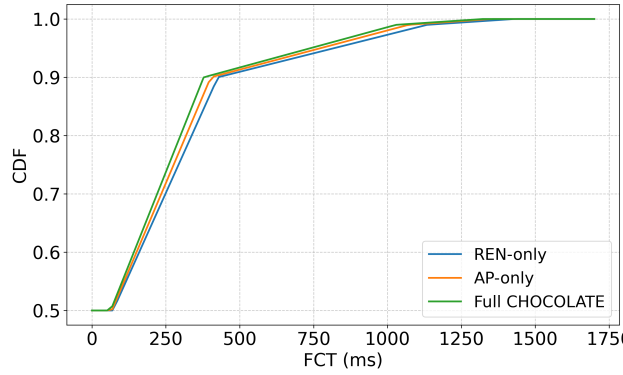

Figure 6. FCT CDF comparison in ablation experiments

Comparing the performance of AP-only and REN-only, it can be found that AP-only outperforms REN-only across all flow sizes. For example, at 100 000 KB flow size, AP-only's FCT median is 6700.0 ms, representing a 5.6 % reduction compared to REN-only's 7100.0 ms. This indicates that the active prediction mechanism is more effective than the reverse explicit notification mechanism in reducing flow completion time. However, the combination of both mechanisms (Full CHOCOLATE) can produce synergistic effects to achieve optimal performance, validating the superiority of CHOCOLATE's integrated design.

5 CONCLUSION

We propose CHOCOLATE, a RoCE congestion control algorithm that integrates reverse explicit notification (REN) with lightweight flow prediction for timely congestion sensing, and unifies slowdown and recovery for improved bandwidth utilization. Experimental results demonstrate significant performance improvements in lossy networks.

Future work includes:

1. deploying CHOCOLATE on programmable switches and real RNICs for large-scale evaluation;
2. introducing online learning to adaptively update prediction thresholds;
3. extending to multi-tenant scenarios with fairness and fault recovery mechanisms.

Acknowledgements

This work is supported by the Science and Technology Commission of Shanghai Municipality of China under Grant No. 24511109202.

REFERENCES

[1] Gangidi, A.—Miao, R.—Zheng, S.—Bondu, S. J.—Goes, G.—Morsy, H.—Puri, R.—Riftadi, M.—Shetty, A. J.—Yang, J.—Zhang, S.—Fernandez, M. J.—Gandham, S.—Zeng, H.: RDMA over Ethernet for Distributed Training at Meta Scale. Proceedings of the ACM SIGCOMM 2024 Conference (ACM SIGCOMM '24), 2024, pp. 57–70, doi: 10.1145/3651890.3672233.

[2] Qian, K.—Xi, Y.—Cao, J.—Gao, J.—Xu, Y.—Guan, Y. et al.: Alibaba HPN: A Data Center Network for Large Language Model Training. Proceedings of the ACM SIGCOMM 2024 Conference (ACM SIGCOMM '24), 2024, pp. 691–706, doi: 10.1145/3651890.3672265.

[3] Meng, Q.—Zhang, Y.—Zhang, S.—Wang, Z.—Zhang, T.—Luo, H.—Ren, F.: Switch-Assistant Loss Recovery for RDMA Transport Control. IEEE/ACM Transactions on Networking, Vol. 32, 2024, No. 3, pp. 2069–2084, doi: 10.1109/TNET.2023.3336661.

[4] Atikoglu, B.—Xu, Y.—Frachtenberg, E.—Jiang, S.—Paleczny, M.: Workload Analysis of a Large-Scale Key-Value Store. Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '12), 2012, pp. 53–64, doi: 10.1145/2254756.2254766.

[5] Ding, Z.—Xu, Y.—Feng, B.—Jiang, C.: Microservice Extraction Based on a Comprehensive Evaluation of Logical Independence and Performance. IEEE Transactions on Software Engineering, Vol. 50, 2024, No. 5, pp. 1244–1263, doi: 10.1109/TSE.2024.3380194.

[6] Jing, Y.—Feng, B.—Ding, Z.: DesFaaS: Cross-Layer Joint Dynamic Deployment System for Serverless Stateful Functions. IEEE Transactions on Cloud Computing, Vol. 13, 2025, No. 4, pp. 1359–1370, doi: 10.1109/TCC.2025.3628953.

[7] Liu, N.—Ding, Z.—Wang, P.—Jiang, C.: ComPA: Competition-Aware Dynamic Differential Pricing and Resource Allocation in Mobile Edge Computing via Gaming. IEEE Transactions on Communications, Vol. 73, 2025, No. 12, pp. 14032–14047, doi: 10.1109/TCOMM.2025.3608346.

[8] Ding, Z.—Zhou, Y.—Wang, S.—Jiang, C.: SCAFE: A Service-Centered Cloud-Native Workflow Engine Architecture. IEEE Transactions on Services Computing, Vol. 16, 2023, No. 5, pp. 3682–3695, doi: 10.1109/TSC.2023.3259989.

[9] Ding, Z.—Chen, Y.—Pan, M.—Li, X.—Wang, P.: Network-Aware Service Composition in Mobile Environment. Concurrency and Computation: Practice and Experience, Vol. 29, 2017, No. 3, e3777 pp., doi: 10.1002/cpe.3777.

[10] Mittal, R.—Shpiner, A.—Panda, A.—Zahavi, E.—Krishnamurthy, A.—Ratnasamy, S.—Shenker, S.: Revisiting Network Support for RDMA. Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '18), 2018, pp. 313–326, doi: 10.1145/3230543.3230557.

[11] Guo, C.—Wu, H.—Deng, Z.—Soni, G.—Ye, J.—Padhye, J.—Lipshteyn, M.: RDMA over Commodity Ethernet at Scale. Proceedings of the 2016 ACM SIGCOMM Conference (SIGCOMM '16), 2016, pp. 202–215, doi: 10.1145/2934872.2934908.

[12] Meng, Q.—Ren, F.: Lightning: A Practical Building Block for RDMA Transport Control. 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS), 2021, pp. 1–10, doi: 10.1109/IWQOS52092.2021.9521326.

[13] The ns-3 Network Simulator. 2025, `https://www.nsnam.org/` [accessed 2025-07-02].

[14] Li, Y.—Miao, R.—Liu, H. H.—Zhuang, Y.—Feng, F.—Tang, L.—Cao, Z.—Zhang, M.—Kelly, F.—Alizadeh, M.—Yu, M.: HPCC: High Precision Congestion Control. Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM '19), 2019, pp. 44–58, doi: 10.1145/3341302.3342085.

[15] Zhu, Y.—Eran, H.—Firestone, D.—Guo, C.—Lipshteyn, M.—Liron, Y.—Padhye, J.—Raindel, S.—Yahia, M. H.—Zhang, M.: Congestion Control for Large-Scale RDMA Deployments. ACM SIGCOMM Computer Communication Review, Vol. 45, 2015, No. 4, pp. 523–536, doi: 10.1145/2829988.2787484.

[16] Mittal, R.—Lam, V. T.—Dukkipati, N.—Blem, E.—Wassel, H.—Ghobadi, M.—Vahdat, A.—Wang, Y.—Wetherall, D.—Zats, D.: TIMELY: RTT-Based Congestion Control for the Datacenter. ACM SIGCOMM Computer Communication Review, Vol. 45, 2015, No. 4, pp. 537–550, doi: 10.1145/2829988.2787510.

[17] Qi, T.—Li, S.—Zou, X.—Deng, L.—Wei, G.: PC4: Precision Collective Communication Congestion Control for AI Cluster. IEEE INFOCOM 2025 – IEEE Conference on Computer Communications, 2025, pp. 1–10, doi: 10.1109/INFOCOM55648.2025.11044740.

[18] Kaur, S.—Kumar, K.—Aggarwal, N.: A Review on P4-Programmable Data Planes: Architecture, Research Efforts, and Future Directions. Computer Communications, Vol. 170, 2021, pp. 109–129, doi: 10.1016/j.comcom.2021.01.027.

[19] Hancock, D.—van der Merwe, J.: HyPer4: Using P4 to Virtualize the Programmable Data Plane. Proceedings of the 12th International on Conference on Emerging Networking Experiments and Technologies (CoNEXT '16), 2016, pp. 35–49, doi: 10.1145/2999572.2999607.

[20] Bosshart, P.—Daly, D.—Gibb, G.—Izzard, M.—McKeown, N.—Rexford, J.—Schlesinger, C.—Talayco, D.—Vahdat, A.—Varghese, G.—Walker, D.: P4: Programming Protocol-Independent Packet Processors. ACM SIGCOMM Computer Communication Review, Vol. 44, 2014, No. 3, pp. 87–95, doi: 10.1145/2656877.2656890.

[21] Alizadeh, M.—Edsall, T.: On the Data Path Performance of Leaf-Spine Datacenter Fabrics. 2013 IEEE 21st Annual Symposium on High-Performance Interconnects, 2013, pp. 71–74, doi: 10.1109/HOTI.2013.23.

Haotian Wu received his Bachelor of Engineering in electronic information engineering from the Shandong University in 2013. He is currently a Researcher at the Shanghai AI Power Technology Co., Ltd., where he leads a team dedicated to technical breakthroughs and development of RDMA-based virtual networks.

Jiasheng Wang received his B.Sc. degree from the Beijing University of Civil Engineering and Architecture, Beijing, China, in 2024. He is pursuing his Master's degree in computer science and technology at the Tongji University, Shanghai, China. His current research interests include RDMA, RoCE, and data center network.

Yuehao Xu received his B.Sc. degree from the Zhejiang Gongshang University, Hangzhou, China, in 2021. He is pursuing his Ph.D. in computer science and technology at the Tongji University, Shanghai, China. His current research interests include large-scale cloud computing, service computing, and edge computing.

Song He received his B.Sc. degree in 2025. He is currently pursuing his Master's degree at the Tongji University, Shanghai, China. His current research interests include cloud-edge collaboration and data center networks.

Jian Yu received his Ph.D. degree in circuits and systems from the South China University of Technology, Guangzhou, in 2010. He is currently Senior Engineer at the Department of Laboratory Center with the School of Computer Science and Technology, Tongji University, Shanghai, China. His research interests include Internet of Things, machine learning, pattern recognition, and embedded systems.

Zhijun Ding received his Ph.D. degree from Tongji University, Shanghai, China, in 2007. Currently he is Professor with the School of Computer Science and Technology, Tongji University, Shanghai, China. His research interests include cloud computing and services, intelligent software engineering, big data intelligence. He has published more than 100 papers in domestic and international academic journals and conference proceedings.