# Barre:Empowering Simplified and Versatile Programmable Congestion Control in High-Speed AI Clusters

Yajuan Peng*, Haoran Wei*, Xiaolong Zhong, Junkai Huang, Haohan Xu, Zicheng Wang, Yang Bai, Zhuo Jiang, and Jianxi Ye, Xiaoliang Wang, Xiaoming Fu ✉, Huichen Dai ✉

Contact : Huichen Dai (ByteDance) <daihuichen@bytedance.com>

# Background

- **Dilemma of Advanced CC Algorithms**

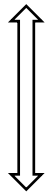
Observation:

HPC and AI data centers continue to predominantly employ **DCQCN**[1],
but hardware and workloads update

*Difficult to fine-tune for AI clusters*

Limitation for advanced CC algorithms:

Deployment requirements & overhead



*Mismatch*

Commercial hardware capabilities

[1] Congestion control for large-scale RDMA deployments. Yibo Zhu. et. al. SIGCOMM'15

# Background

- **Demand for Effective CC in Modern HPC/AI Clusters**

Large-scale distributed training jobs introduce multi-dimension and stringent performance requirements

| Collectives | Parallel Patterns | Traffic pattern | Network congestion | Incast Scale | Key Requirement |
|---|---|---|---|---|---|
| AllReduce | Data Parallel | Tree or Ring | 1 to 1, or many to 1 | <20 | Full bisection bandwidth |
| AllGather | | Ring | 1 to 1 | =1 | Bandwidth |
| AlltoAll | Expert Parallel | Fullmesh | N to N, N is very large | >500 | |
| SendRecv | Pipeline Parallel | Point to Point | 1 to 1, or many to 1 | <20 | Strict latency |

Establishing universally optimal parameter settings is challenging.

Transitions between traffic patterns frequently trigger severe congestion events.

# Motivation

- **Design Rationale of CC in High-speed HPC/AI Cluster**

Expensive RDMA hardware & scalability & emerging performance requirements

*The design of algorithms must be compatible with available hardware, necessitating careful consideration of the costs and efficiency associated with exploring various combinations of hardware, software, and algorithmic approaches.*

– Sara Hooker in the hardware lottery[1]

Fundamental requirements for CC algorithms:

- **Broad Applicability**: adapt rapidly to traffic bursts
- **Simplified Logic**: easy to set parameters and threshold
- **Implementation Practicality**: support by commercial hardware

[1] The hardware lottery. Sara Hooker. ACM Communications'21

# Motivation

- **Re-examine the CC Design**

Hardware-informed congestion signal selection criteria

- **ECN** : simple, efficient, extensively adopted
- **INT** : requires path-wide switch support, leading to challenging deployment
- **RTT** : end-to-end nature and high sensitivity; however, baseline determination is difficult

Rate adaptation mechanisms optimized for high-bandwidth environments

- **Credit-based control** : high performance, expensive, but need to modify RoCEv2 protocol
- **Window-based control**: precise, incompatible with RDMA and RNIC, coupled with the transport protocol stack
- **Rate-based control**: deployment-friendly, unable to bound inflight bytes

Co-design methodologies that exploit hardware capabilities while mitigating intrinsic limitations

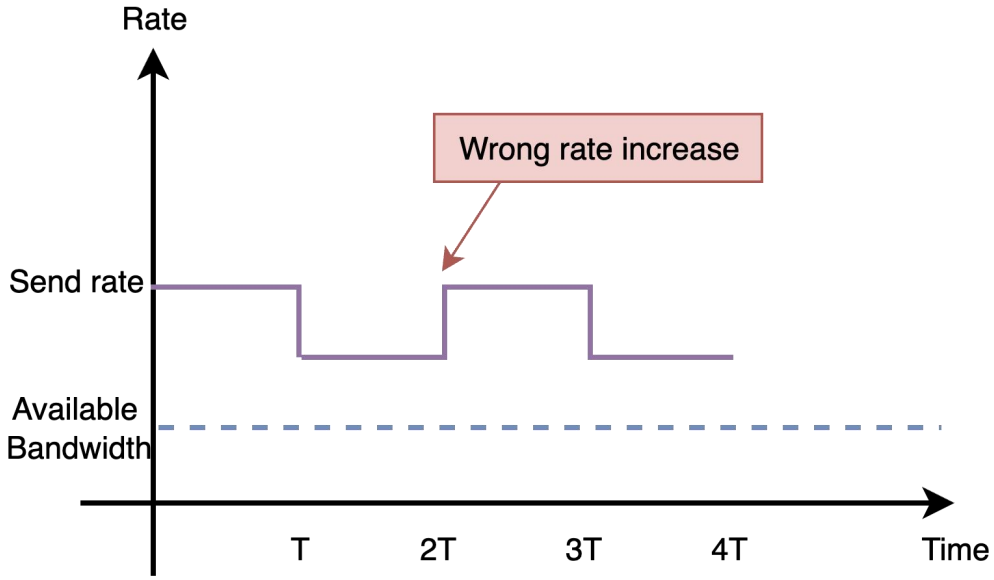
- **Programmable Congestion Control (PCC) provided by SmartNIC**
- **Limitations**: RoCEv2 rate limiter only supports rate-based control, without CWND
- Handling 400Gbps traffic CC events with limited computing resources is challenging

# Motivation

- **ECN Base Problem(Wrong rate increase)**

IF RTT * Rate < MTU

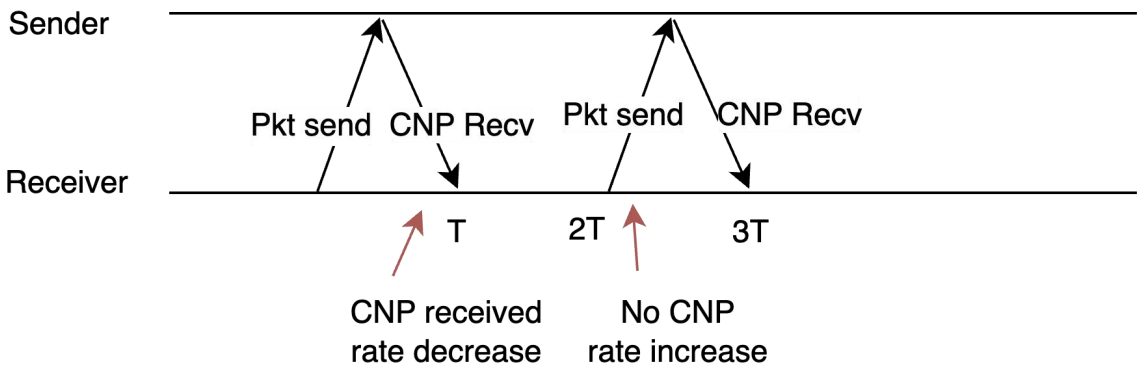
Rate will be wrong growth.




For example:

T=20us, MTU = 4KB

minimum rate = 4KB/20us = 1.6Gbps



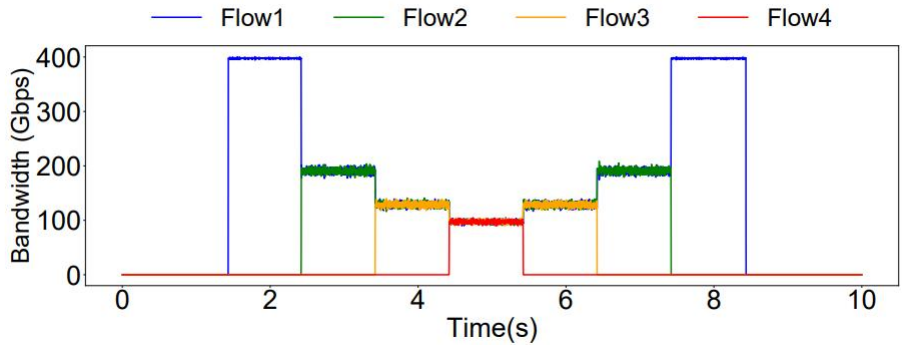
# Barre Design

## Based on the characteristics and inherent limitations of RNICs

- leverage the PCC capabilities and event interfaces provided by RNICs
- practical, scalable, simple, compatible, easy to deploy

## Basic idea: adaptive adjustment interval (AIMD)

- Delay-based increase: $R = R + \alpha$, if no CNP in last real-time RTT
- Per-CNP-based decrease: $R = R \times \beta$ ($0.95 < \beta < 0.99$) each time receives a CNP (~1us)
- Dynamic self-constraint fairness




Throughput of 4 flows shows Barre fairness

# Barre Design

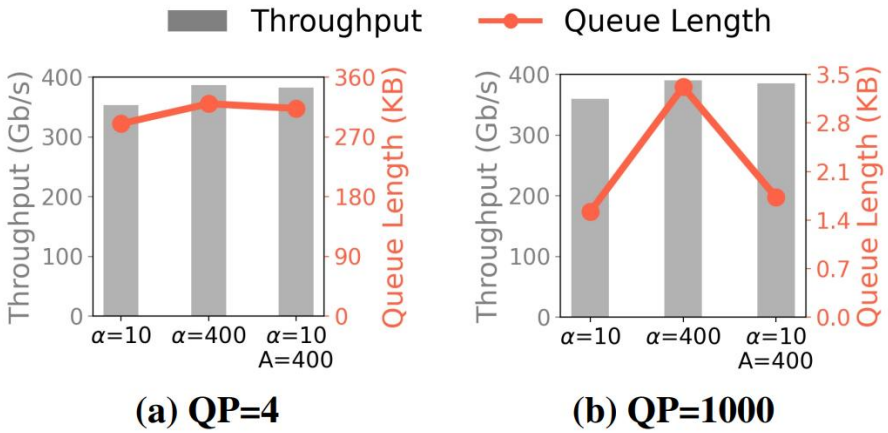
## Functional Components : **Fast Increase**, Dual-lock, Inflight Monitor

Fixed increase factor ***α*** :

- faces dilemma between **inadequate bandwidth utilization** and **burst transient congestion**
- Unable to deal with both mice flows and elephant flows simultaneously

**Fast Increase** :

- Conservative increase ***α*** & aggressive increase ***A*** (no CNP in a continuous period)
- Adaptively adjust to network state between idle and saturate




(a) QP=4 (b) QP=1000

Fast Increase achieves **optimal bandwidth utilization** while **keeping low queue length** in both **mice flow** and **elephant flow**.

# Barre Design

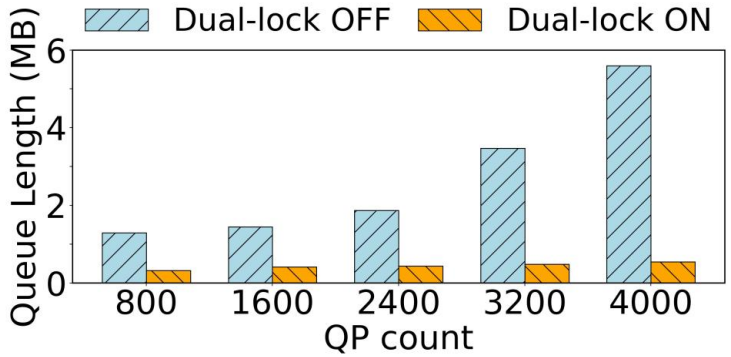
## **Functional Components :** Fast Increase, **Dual-lock**, Inflight Monitor

While **BDP (Rate*RTT) < MTU** :

- Multiple RTTs are required to send a single data packet
- The quantity of CNP cannot precisely reflect congestion

**Dual-lock** :

- Rate increase condition : “Timer (real-time RTT)” **AND** “ByteCounter”
- Fairness: high-speed flows: “Timer” (RTT), low-speed flows: “ByteCounter”.  Flexibly response to congestion.




Dual-lock maintains **extremely low switch queue length** in large-scale Incast.

# Barre Design

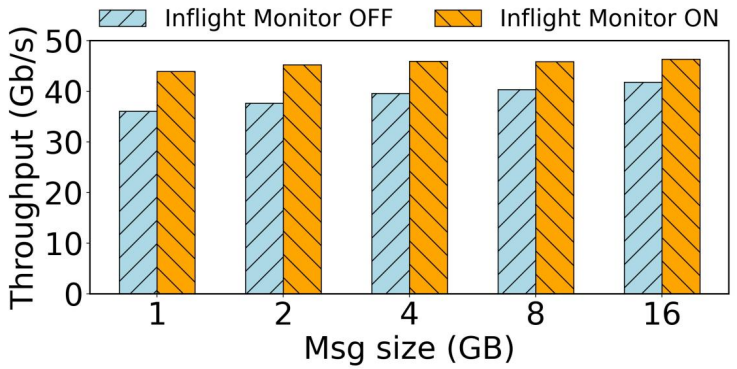
## **Functional Components :** Fast Increase, Dual-lock, **Inflight Monitor**

RNIC lacks **per-packet ACK** and **payload size header** :

- Unable to track and limit the amount of **inflight bytes** that have been sent but have not been confirmed.
- Causing data packets to continuously be injected into the link during congestion.

### **Inflight Monitor** :

- Accumulate sending bytes within current RTT : $\mathbf{S_{now}}$ (TX events from PCC)
- Temporary restriction : if $\boldsymbol{S_{now} > R \times}$ ***real-time RTT***, $\boldsymbol{R' = R \times 1/4}$ keep until CNP is received

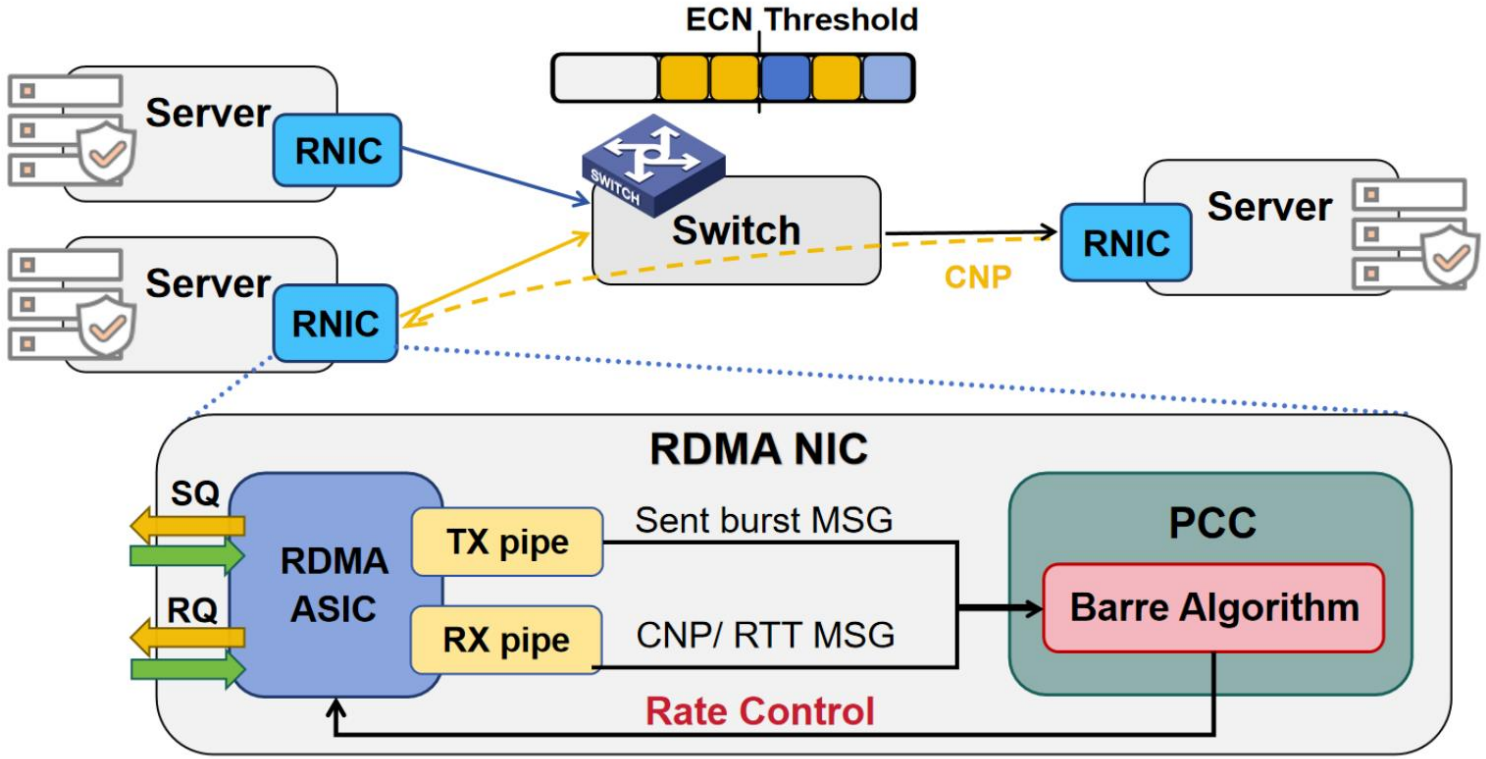



Inflight Monitor significantly improves the throughput in large-scale congestion in NCCL AlltoALL test.

# Implementation Details

## System architecture

- ➢ Implement on BlueField-3 SuperNICs
- Leverage the maximal potential of the event interface provided by PCC
- Achieve integration of software, hardware, and algorithm.



# Implementation Details

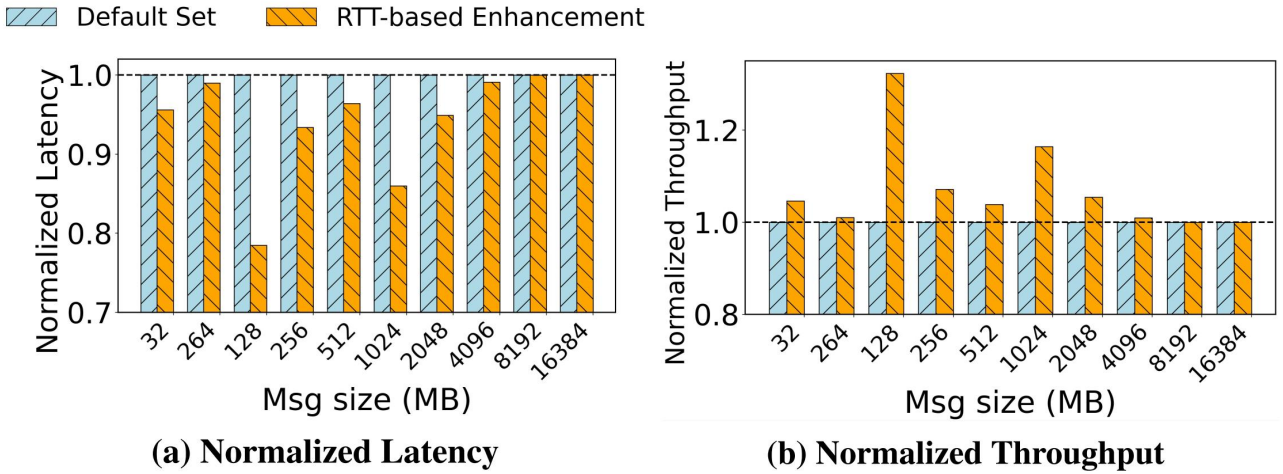
## RTT-based Enhancement

### RTT Probe Modification :

- RTT probe packets may be lost (8.9%) due to hardware NIC limitation
- Make **RTT_Rsp packet** carry both the sending timestamp of RTT_Req and the receiving timestamp of RTT_Rsp.
- Update the last ***real-time RTT*** whenever an RTT probe packet returns.

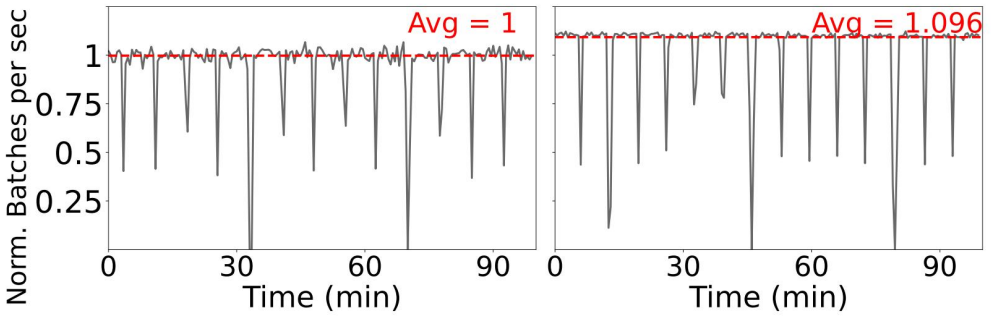
### RTT-based per-flow Increase Factor $\alpha_k = RTT_k \cdot \alpha/C$ :

- Achieve fairness among flows with different path lengths for RTT-based increase.




(a) Normalized Latency (b) Normalized Throughput

The performance of RTT-based enhancement in 128-GPU NCCL AlltoAll test.




(a) Default set (b) RTT-based enhancement

The throughput log from real training tasks.

# Evaluation Results

## Network Topology

- ➢ Barre has been deployed in a AI production cluster for over one year.
- A three-layer switch CLOS topology, connects over 10K GPUs equipped with 400 Gbps BlueField-3 SuperNIC

## Baselines :

- **DCQCN** : The most commonly used general CC algorithm in data centers.
- **InfiniBand (IB)** : Specialized network infrastructure solutions, efficient yet costly.

## Evaluation :

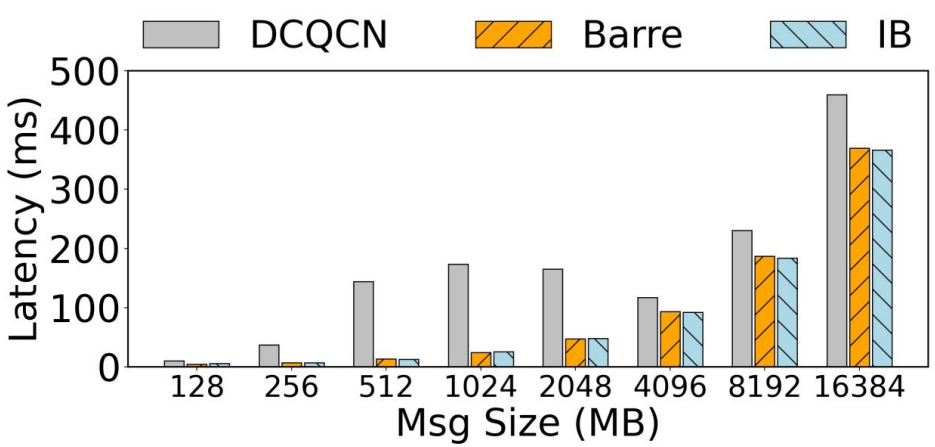
- NCCL AlltoAll test [1] : standard benchmark tests for collective communication

[1] NVIDIA NCCL Test. https://github.com/NVIDIA/nccl-tests/tree/master

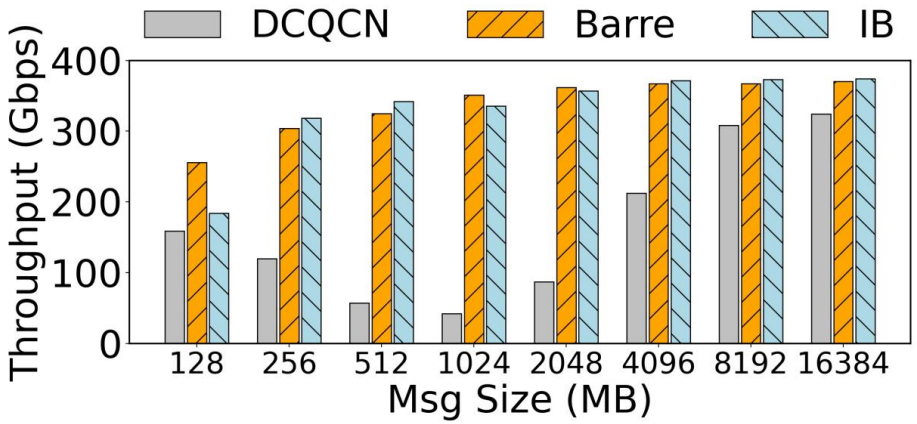
# Evaluation Results

## Large-scale Collective Communication

- NCCL AlltoAll test in 400 Gbps 256-GPU cluster




(a) Latency

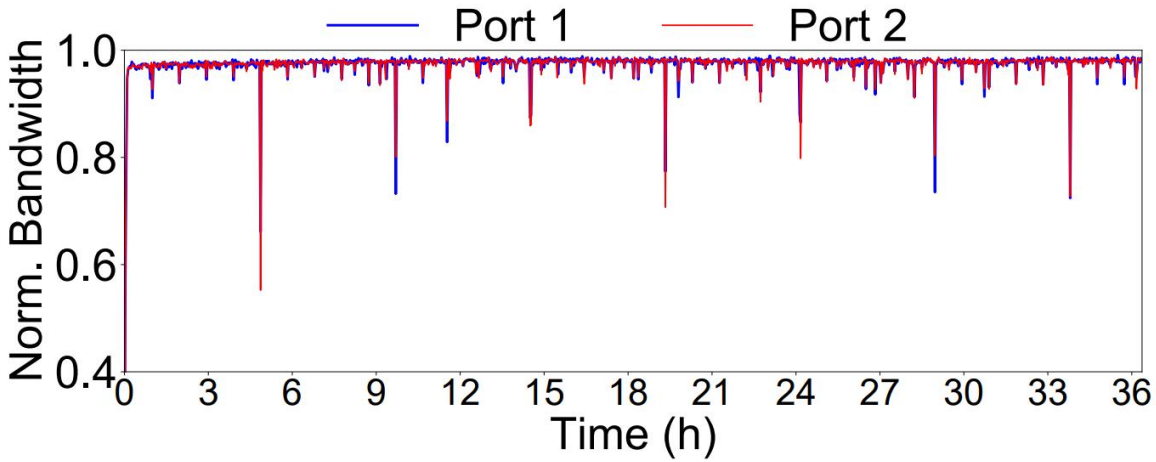



(b) Bus Bandwidth

- ✓ Barre achieves superior latency and throughput **comparable to IB** while significantly outperforming DCQCN.
- ✓ Compared to IB, Barre exhibits substantially lower hardware **deployment costs** and implementation **complexity**, along with enhanced **scalability** in large-scale clusters.

# Evaluation Results

## Switch traffic logs collected from the production environment




- ✓ Barre achieves achieves near-optimal utilization without triggering PFC event.
- ✓ Barre exhibits robust convergence and fairness under varying traffic loads.
- ✓ Barre has been deployed on RoCEv2 clusters with 30k+ 400Gbps NICs over one year.

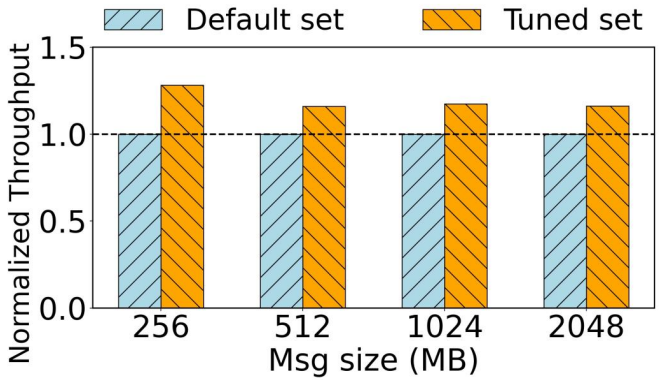
# Evaluation Results

## Using Barre’s key idea to optimize DCQCN

- Fine-tune DCQCN parameters based on Barre techniques and the capability of current RNIC

DCQCN Parameter Tuning Settings

| Parameter | Default Value | Tuned Value | Explanation |
|---|---|---|---|
| min_time_between_cnps | 4 | 0~2 | Interval for Receiver Responding with CNP (us) |
| rpg_min_dec_fac | 50 | 80~98 | The minimum reduction at each rate decrease |
| rate_reduce_monitor_period | 4 | 1~4 | Minimum deceleration interval (us) |
| rpg_alpha_rate | 5 | 8~20 | Rate increase magnitude |
| rpg_time_reset | 300 | 20~100 | Rate increase interval (us) |

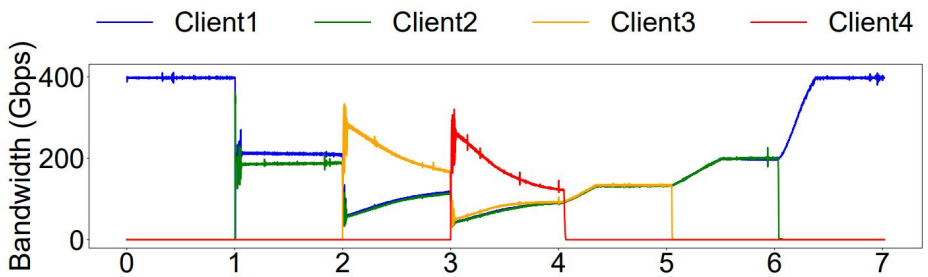



NCCL AllReduce Test on 1024-GPU cluster

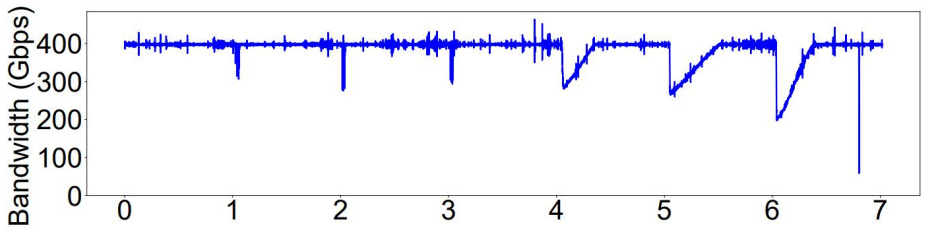
- The normalized throughput is increased by 19.54% in average.

# Evaluation Results

## Using Barre’s key idea to optimize DCQCN

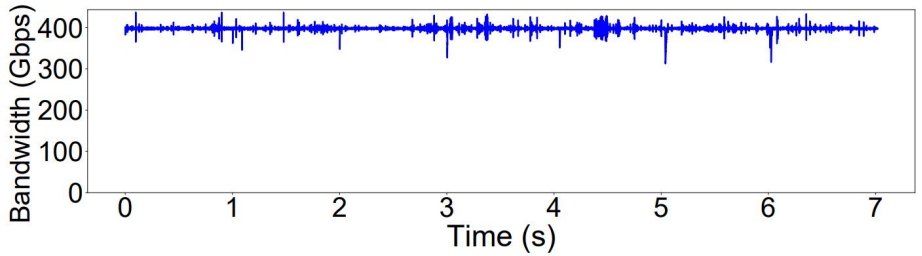



(a) Client Bandwidth (Default Parameter)




(b) Server Bandwidth (Default Parameter)


(c) Client Bandwidth (Tuned Parameter)




(d) Sever Bandwidth (Tuned Parameter)

- ➢ Default DCQCN faces problems of oscillations in network bandwidth and low utilization.
- ✓ Tuned DCQCN improves stability, fairness and rapid convergence.

# Conclusion

## Barre:Empowering Simplified and Versatile Programmable Congestion Control in High-Speed AI Clusters

- Leverage the PCC capabilities and event interfaces provided by RNICs
- Practical, scalable, simple, compatible, easy to deploy
  - ☑ Barre has been deployed on RoCEv2 clusters with 30k+ 400Gbps NICs over one year.
- Achieves superior latency and throughput **comparable to IB** using commercial RNICs
- Flexibly optimize other CC algorithms

# THANK YOU

Yajuan Peng*, Haoran Wei*, Xiaolong Zhong, Junkai Huang, Haohan Xu, Zicheng Wang, Yang Bai, Zhuo Jiang, and Jianxi Ye, Xiaoliang Wang, Xiaoming Fu ✉, Huichen Dai ✉

Welcome to contact us: Huichen Dai (ByteDance) <daihuichen@bytedance.com>