# Rethinking Intra-host Congestion Control in RDMA Networks

Zirui Wan[1,2], Jiao Zhang[1,3,*], Yuxiang Wang[1], Kefei Liu[1], Haoyu Pan[1], Tao Huang[1,3]
State Key Laboratory of Networking and Switching Technology, BUPT, China[1]
China Mobile (Suzhou) Software Technology Co., Ltd[2]
Purple Mountain Laboratories[3]

## ABSTRACT

RDMA has been widely deployed in production datacenters. The conventional wisdom in system and networking believes that the intra-host network delivers high performance. However, intra-host resources witness a relative stagnation in technology trends compared to the evolving RDMA NIC (RNIC). Thus, the RNIC traffic may not get sufficient intra-host resources when it contends with intra-host traffic. A line of recent works from large-scale production datacenter operators demonstrates the emergence of intra-host congestion and associated performance collapse, which forces us to rethink the practice of intra-host congestion control. However, the ability to efficiently control RDMA intra-host networks is far less mature than inter-host networks, which brings challenges in congestion monitoring, intra-host resource allocation and RNIC traffic adjustment. In this paper, we propose RDMA intra-Host Congestion Control (RHCC), which combines sub-RTT granularity intra-host traffic congestion avoidance and proactive RNIC traffic adjustment. We implement RHCC on commodity servers and RNICs and conduct experiments to evaluate the performance. The results show that RHCC can increase/decrease the network throughput/latency by up to 2× and 1.4×, respectively.

## CCS CONCEPTS

• **Networks** → **Transport protocols**; **Data center networks**; • **Software and its engineering** → **Operating systems**.

## KEYWORDS

Congestion control, RDMA datacenter transport

**ACM Reference Format:**
Zirui Wan[1,2], Jiao Zhang[1,3,*], Yuxiang Wang[1], Kefei Liu[1], Haoyu Pan[1], Tao Huang[1,3]. 2024. Rethinking Intra-host Congestion Control in RDMA Networks. In *The 8th Asia-Pacific Workshop on Networking (APNet 2024), August 3–4, 2024, Sydney, Australia.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3663408.3663413

* Jiao Zhang is the corresponding author.


*APNet 2024, August 3–4, 2024, Sydney, Australia*

ACM ISBN 979-8-4007-1758-1/24/08
https://doi.org/10.1145/3663408.3663413

## 1 INTRODUCTION

In recent years, Remote Direct Memory Access (RDMA) has been widely deployed to provide high-speed networks for datacenter applications such as machine learning [17, 20], storage [16, 19, 32] and high performance computing [27, 28]. The above applications also have stringent performance requirements for more computational and storage resources. To meet these growing trends, commodity datacenter servers are equipped with multiple high-speed accelerators, including RNIC, GPU and NVMe SSD. These hardware accelerators, together with CPU cores, memory and PCIe interconnect, form a complicated intra-host network.

The conventional wisdom in system and networking believes that the intra-host network provides sufficient bandwidth and lossless interconnect fabric, and thus achieves consistent high performance. However, intra-host resources (e.g., memory bandwidth, PCIe bandwidth) witness a relative stagnation in technology trends compared to evolving hardware accelerators, especially RNICs. Specifically, the RNIC line rate increases rapidly from 25Gbps to 200Gbps. In contrast, the PCIe bandwidth increases from 63Gbps to 256Gbps. As a result, the RNIC traffic may not get sufficient intra-host resources when it contends with intra-host traffic and packets will be queued on the RNIC. RNIC constantly detects the watermark of the receiving buffer and will actively generate TX pauses (i.e., PFC pause frames) to the upstream switch to avoid buffer overflow in the PFC-enabled scenario, which may eventually lead to a PFC storm [19, 26].

A line of recent works from large-scale production datacenter operators demonstrates that the RNIC traffic suffers intra-host congestion and causes severe application-level performance collapse [14, 15, 19, 20, 22, 26]. For instance, Google [15] finds memory interconnect contention in their production clusters and leads to significant queueing at hosts. Alibaba [19] demonstrates that Tx pauses will be generated when RNIC PCIe congestion happens due to the contention of RDMA and TCP traffic. Bytedance [20, 22, 26] analyzes their performance metrics and diagnoses resource bottlenecks in the RNIC and intra-host fabric. We evaluate the intra-host congestion phenomenon in our testbed. We observe that intra-host congestion leads to up to 68% network throughput degradation and 2.6× average latency increase.

The emergence of intra-host congestion forces us to rethink the practice of intra-host congestion control for the RNIC traffic to avoid performance penalties. Inspired by the inter-host congestion control, we advocate that the new intra-host congestion control mechanism should achieve the following design goals: *(i)* fast converge with a sub-RTT granularity to alleviate congestion since the receiving buffer on the RNIC has limited size; *(ii)* easy to deploy on commodity servers and RNICs while adapting to different inter-host congestion control mechanisms and congestion signals; *(iii)*

flexible intra-host resource allocation policies between the RNIC traffic and intra-host traffic for different applications.

However, the ability to efficiently control RDMA intra-host networks is far less mature than inter-host networks, which brings challenges in realizing the design goals. First, since the intra-host network maintains a lossless fabric, classical congestion signals stay away from the actual congestion point. Second, the standard intra-host resource allocation tool does not support complex allocation functions, which makes the resource allocation for intra-host traffic challenging. Third, in the RDMA network, the receiver's kernel module can not directly modify data packets, which makes it difficult to adjust the RNIC traffic at the receiver.

In this paper, we propose RDMA intra-Host Congestion Control (RHCC) mechanism, which combines sub-RTT granularity intra-host traffic congestion avoidance and proactive RNIC traffic adjustment. Specifically, the receiver constantly monitors the intra-host congestion signal and allocates resources for intra-host traffic to provide guarantee. Meanwhile, the sender of RNIC traffic leverages the Programmable Congestion Control (PCC) [34] to periodically transmit probe packets and obtain the receiver processing latency. Then, the sender updates the sending rate to achieve max-min fairness among RNIC traffic. We implement RHCC in our testbed and conduct experiments to evaluate the performance. The results show that RHCC can increase/decrease the network throughput/latency by up to 2× and 1.4×, respectively.

**Contributions.** Our key contributions are:

• We **introduce the emerging RDMA intra-host congestion** in production datacenters and **analyze the associated impacts on performance degradation**.

• We **present RHCC, a novel RDMA intra-host congestion control mechanism** that combines sub-RTT granularity intra-host traffic congestion avoidance and proactive RNIC traffic adjustment.

• We **implement RHCC on commodity servers and RNICs and conduct experiments to evaluate the performance.** The results show that RHCC can improve application-level performance in terms of throughput and latency.

## 2 MOTIVATION

In this section, we introduce the background of RDMA intra-host networks and impacts of RDMA intra-host congestion. Then, we propose our design goals for RHCC.

### 2.1 Background: RDMA Intra-host Networks

**Intra-host network architecture.** Figure 1 illustrates an example of the intra-host network for commodity Intel servers. The Intel CPU socket contains the Core and the Uncore [1] sub-systems that are connected together through the mesh interconnect architecture [9]. Specifically, the Uncore sub-system consists of a variety of components that execute the traffic from the CPU core or I/O devices. For example, the memory controller manages access to the memory, and the Integrated I/O controller (IIO) manages traffic from the PCI

[1] "Uncore" roughly equates to logic outside the CPU processor cores but residing on the same die.

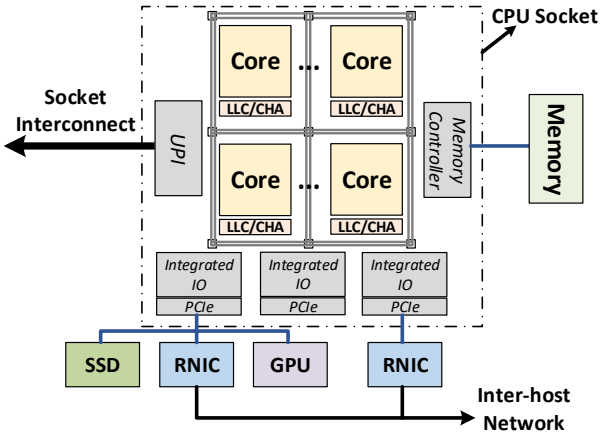



**Figure 1: Illustration of intra-host network for the commodity Intel server (AMD has a similar architecture).**

Express (PCIe) domain that consists of many hardware accelerators (e.g., RNIC, GPU). The Cache/Home Agent (CHA) manages the cache coherency, and the Ultra Path Interconnect (UPI) provides socket interconnect [3]. The above fabrics and the end-node devices together form the intra-host network.

**RNIC traffic intra-host data path.** RDMA has been widely deployed in commercial datacenters [16, 19, 37, 39]. Benefiting from the kernel bypass technology, the data path no longer passes through the CPU core in RDMA networks. Instead, the packet transmits from RNIC to the IIO, passing through the mesh architecture, and then enters memory by the memory controller.

Importantly, the RNIC traffic intra-host data path traverses lossless RNIC-to-IIO and IIO-to-memory controller interconnects, both of which use credit-based flow control schemes. For example, when a data packet enters the RNIC, it will check if the PCIe has sufficient credits through PCIe transactions [15, 29]. When credits are available, the data packet will enter the IIO. Otherwise, the data packet will be queued in the receiving buffer on the RNIC. Similarly, if the memory controller has sufficient credits, packets in the IIO will be directly written to the memory through the memory controller. Otherwise, packets will be queued in the IIO buffer.

### 2.2 Understanding Impacts of RDMA Intra-host Congestion

**Emergence of intra-host congestion.** Datacenter applications are increasingly driving demands for more computation, communication and storage resources. To meet these growing demands, commodity servers are equipped with many high-speed accelerators. For example, NVIDIA DGX [7] can be equipped with up to eight high-speed Infiniband RNICs and eight GPUs to achieve outstanding performance. The chunkservers in distributed storage system are equipped with multiple NVMe SSDs to provide high throughput and IOPS [19, 25]. However, intra-host resources witness a relative stagnation in technology trends compared to

evolving accelerators, especially RNICs [2]. Specifically, limited by physical properties and market's preference for DRAM and PCIe, the bandwidth of memory and PCIe lags behind the RNIC [36]. As a result, the RNIC traffic may not get sufficient intra-host resources when it contends with intra-host traffic.

In this paper, *we primarily focus on the intra-host congestion caused by the receiver-side RNIC traffic* along its data path, i.e., RNIC PCIe congestion and memory interconnect congestion, since intra-host congestion predominantly occurs at the receiver [14, 15, 23, 24].

**Impacts analysis of RDMA intra-host congestion.** When congestion happens, the intra-host network latency increases and PCIe credits are replenished with a larger delay and eventually exhausted, which results in packet queueing on the RNIC. RNIC constantly detects the watermark of the receiving buffer. In the PFC-enabled scenario, to avoid packet loss caused by the buffer overflow, RNIC actively generates Tx pauses to the upstream switch, which may eventually lead to a PFC storm [19, 26]. In the PFC-free scenario, RNIC actively generates CNPs as congestion signals back to senders.

A line of recent works from large-scale production datacenter operators demonstrates that intra-host congestion has happened and caused severe application-level performance collapse [14, 15, 19, 20, 22, 26]. Google [14, 15] finds that intra-host congestion happens when there is memory bandwidth contention between CPU-to-memory traffic and RNIC traffic, and results in 35-55% throughput degradation. Alibaba [19] finds that Tx pauses are generated when RNIC PCIe congestion happens due to the contention of RDMA and TCP traffic. Bytedance demonstrates the RNIC PCIe congestion caused by the loopback traffic [20] and diagnoses resource bottlenecks in the RNIC and intra-host fabric [22, 26].

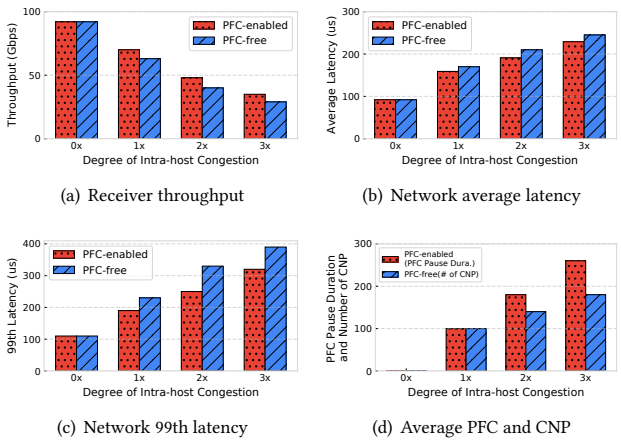
To alleviate memory bandwidth bottleneck, modern servers support Intel Data Direct I/O (DDIO) [1] that enables RNIC to directly access data in the Last Level Cache (LLC). However, its benefit is small due to the leaky DMA problem (i.e., frequent cache eviction triggered by new incoming messages) and the congestion situation could worsen [18, 24, 38].

In the following, we conduct experiments to measure the impacts of RDMA intra-host congestion, including pause duration, throughput degradation and network latency, and provide insights on the root causes.

**Experiment setup.** We build a testbed with two Dell R740 servers connected via a single switch. Each server is equipped with dual Intel Xeon 6240 CPUs, one Mellanox ConnectX-6 DX 100 Gbps RNIC connected over 128 Gbps PCIe 3.0 ×16, two memory channels DDR4 DIMMs and enables DDIO with 4 MB cache ways. Thus, our servers achieve the same set of intra-host network resources with the latest commercial server but with scaling (e.g., a server with 256 Gbps PCIe 4.0 will have 2× more RNIC bandwidth) [22].

We use the standard benchmark tool Mellanox Perftest [2] to transmit data from one server to another and measure the network throughput and latency. We vary the message size (from 64KB to 1MB) and QP number (from 1 to 8) to change the intra-host resource utilization of RNIC traffic. We use Intel Memory Latency Checker (MLC) [8] to generate memory access (i.e., CPU-to-memory) traffic with a 1:1 read/write ratio workload. We increase the memory bandwidth consumption from 1× to 3× by increasing the number of active CPU cores to fulfill more memory requests, which results in a total observed bandwidth consumption of 19.0 GBps, 25.3 GBps and 30.3 GBps, respectively.

[2]NVLink and NVSwitch [11] in modern servers enables dedicated, fully-connected GPU-to-GPU communication bypassing the intra-host network.




(a) Receiver throughput (b) Network average latency

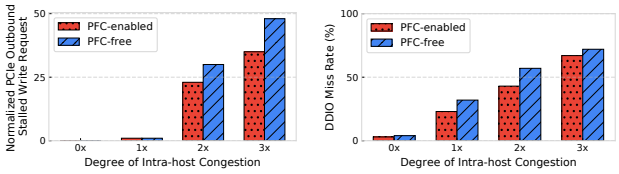
(c) Network 99th latency (d) Average PFC and CNP

**Figure 2: Network performance under different level intra-host congestion.**

We use Intel Performance Counter Monitor (PCM) [5] to measure the host-level metrics (e.g., memory bandwidth utilization) and count the PCIe write hit/miss number, which indicates the usefulness of DDIO. More PCIe writes miss reflects that cache eviction happens frequently and DDIO fails. We use NVIDIA Neohost tool [6] to measure the number of PCIe outbound stalled write requests that reflects the backpressure from PCIe to the RNIC.

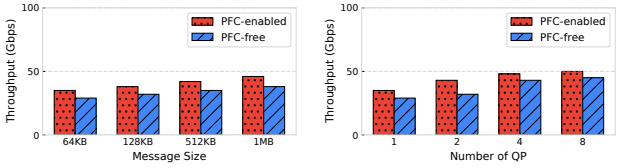
**[Figure 2] Intra-host congestion leads to significant receiver throughput degradation and network latency increase.** Figure 2 shows the receiver throughput, latency and PFC pause/CNP under different degrees of intra-host congestion. We can see that the case of 0× shows the best network performance and the receiver throughput/latency decreases/increases with increasing degrees of intra-host congestion. In the 3× case, in the PFC-enabled and PFC-free network, the throughput decreases by 62% and 68%, respectively, compared with 0× case. Meanwhile, the average latency increases by 2.4× and 2.6×, respectively. In Figure 2(d), we measure the impacts of congestion control mechanism on the inter-host network and find that both the PFC pause duration and the number of CNPs significantly increases as the degree of intra-host congestion increases.

**[Figure 3] Intra-host congestion leads to severe PCIe backpressure to the RNIC and a high DDIO miss rate.** Figure 3(a) shows the normalized PCIe outbound stalled write requests over 1× case. We can see that the PCIe backpressure significantly increases as the degree of intra-host congestion increases. This indicates that the root cause of observed performance degradation in Figure 2 is the high memory bandwidth contention, which prevents RNIC from acquiring sufficient intra-host resources (i.e., memory bandwidth). As a result, packets are queued and eventually network congestion control is triggered. Figure 3(b) shows the DDIO miss rate. We can




(a) PCIe backpressure (b) DDIO miss rate

**Figure 3: PCIe backpressure and DDIO miss rate under different level intra-host congestion.**




(a) Different message size (b) Different QP number

**Figure 4: Receiver throughput under different message size and QP number. (with 3× degree of intra-host congestion)**

see that the DDIO miss rate increases with increasing degrees of intra-host congestion. This is because cache line eviction requires sufficient memory bandwidth for memory write-back operations and DDIO write allocation will be decelerated under high-level intra-host congestion.

**[Figure 4] Impacts of intra-host congestion slightly ameliorate with increasing message sizes and number of QPs.** Figure 4 shows the throughput under different message sizes and QP numbers with 3× degree of intra-host congestion and we note that the throughput slightly increases. This is because larger message size and QP number lead to more in-flight packets, which results in higher memory bandwidth contention and imposes a higher bandwidth consumption for RNIC traffic. However, the benefit is marginal and larger message size requires larger space in the DDIO-allocated cache, which aggravates the leaky DMA problem.

## 2.3 RHCC: Design Goals and Challenges

We need to design a new RDMA intra-Host Congestion Control (RHCC) mechanism to fundamentally resolve the intra-host resource contention problem. Inspired by the inter-host network congestion control, we advocate that RHCC should achieve the following design goals:

*(i) Fast converge.* The intra-host network can quickly converge to avoid congestion with a sub-RTT granularity since the receiving buffer on the RNIC has limited size and the intra-host latency is within hundreds of nanosecond [21].

*(ii) Easy to deploy.* RHCC should be deployable on modern commodity servers and RNICs. Specifically, RHCC should be adaptive to different inter-host congestion control mechanisms and congestion signals (e.g., ECN, RTT or INT).

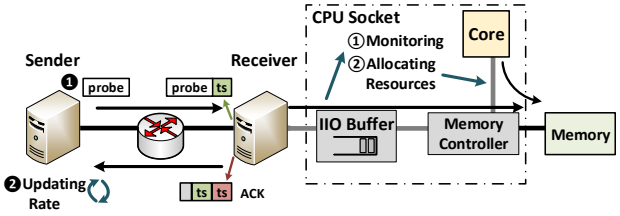



**Figure 5: The overview of RHCC framework.**

*(iii) Flexible resource allocation.* RHCC should support different allocation policies for intra-host resources (e.g., memory bandwidth) between intra-host traffic and RNIC traffic.

There are three major challenges in realizing design goals:

**Challenge #1:** The intricacies of intra-host networks make it difficult to obtain appropriate congestion signals. In addition, classical congestion signals stay away from the actual congestion point since the intra-host network maintains a lossless fabric.

**Challenge #2:** The standard intra-host resource allocation tool does not support complex allocation functions, which makes resource allocation for intra-host traffic challenging.

**Challenge #3:** It is difficult to adjust the RNIC traffic by the receiver. This is because, in the RDMA network, receiver's kernel module can not directly modify data packets. For instance, existing mechanism hostCC [15] tries to mark ECN-bit at the receiver and thus can not be deployed.

Nowadays, Mellanox introduces Programmable Congestion Control (PCC) [34] programmability on the ConnectX-6 DX RNIC and extends it to the DPUs (e.g., BlueField-2 [30], BlueField-3 [31]). PCC provides a proactive network latency probe mechanism that obtains hardware timestamps of when receiver receives packets and generates ACKs. PCC supports timestamp collection at the sender and self-defined control logic. This new feature helps us design and implement RHCC.

# 3 RHCC DESIGN

In this section, we propose RHCC to handle impacts of RDMA intra-host congestion and present our implementation.

## 3.1 RHCC Framework

Figure 5 shows the framework of RHCC. RHCC combines sub-RTT granularity intra-host traffic congestion avoidance and proactive RNIC traffic adjustment. Specifically, the receiver constantly monitors the intra-host congestion signal (i.e., IIO buffer occupancy) and allocates resources for intra-host traffic. Meanwhile, the sender of RNIC traffic periodically transmits probe packets to obtain the timestamp (ts) of when the receiver receives probe packets and transmits back ACKs. Then, the sender calculates the receiver processing latency and updates the sending rate.

### 3.2 Intra-host Traffic Congestion Response

The intra-host traffic uses IIO buffer occupancy as the congestion signal to address **Challenge #1** for three reasons. First, the IIO buffer occupancy increases immediately *when, and only when* memory interconnect congestion happens. This is because memory interconnect congestion causes a greater delay in memory controller credit replenishment and will further cause IIO buffer queueing. Thus, the IIO buffer occupancy provides precise and timely indication of location and scale for intra-host congestion. Second, the IIO buffer occupancy can be monitored using the register provided by commodity Intel CPU hardware at sub-RTT granularity without any modifications. In contrast, obtaining other congestion signals (e.g., RNIC buffer occupancy) is more difficult and requires vendor's assistance. Third, the monitor process lies outside the RNIC traffic data path and thus does not affect intra-host congestion.

RHCC leverages the Intel Model Specific Registers (MSRs) [13] to collect congestion signals. The MSRs maintain the cumulative value of IIO occupancy that incremented at IIO clock frequency. RHCC measures the instant IIO occupancy, $I_{cur}$, by subtracting two cumulative values and calculates a target threshold, $I_{thr}$, where $I_{cur} > I_{thr}$ means intra-host congestion happens. Due to the limitations of the intra-host resource allocation tool, RHCC uses a coarse-grained allocation policy that sets multi-level restrictions on available memory bandwidth for intra-host traffic and releases remaining memory bandwidth to RNIC traffic, which addresses **Challenge #2**. We use Intel MLC tool to measure the total memory bandwidth of our servers to be 33.2GBps. Thus, we set five levels [3] to equalize the memory bandwidth, where the third level (i.e., 20GBps) provides sufficient remaining memory bandwidth for our RNIC traffic. When congestion happens, RHCC levels down the available memory bandwidth to avoid bandwidth wastage.

RHCC uses the Intel Memory Bandwidth Allocation (MBA) tool of Intel Resource Director Technology (RDT) [4, 10] to allocate intra-host resources. MBA changes the memory traffic bandwidth available to the CPU cores by introducing latency for memory requests. MBA enables arbitrary resources to be allocated for any number of CPU cores by setting specific MSR registers, which increases the flexibility of the resource allocation policy.

### 3.3 RNIC Traffic Congestion Response

The RNIC traffic uses the probe mechanism to make adjustments to address **Challenge #3** for three benefits. First, the probe mechanism can reflect the scale of intra-host backpressure across the entire receiver RNIC data path, which helps RHCC handle different types of congestion, including RNIC PCIe congestion and memory interconnect congestion. Second, the probe mechanism does not modify data packets, and thus provides transparent support for upper-layer applications. Third, PCC provides flexible self-defined congestion control logic that can be adapted to different deployed congestion control mechanisms in commercial datacenters. In addition, the new RNIC (e.g., CX6, CX7) and DPU (e.g., BlueField-2, BlueField-3) have been widely deployed in modern datacenters, which supports the deployment of RHCC.

Each newly received probe ACK triggers the sender measures the receiver processing delay, $D_{cur}$. Similar to the intra-host congestion response, RNIC traffic also calculates a target delay, $D_{thr}$, where $D_{cur} > D_{thr}$ means congestion happens. Then, the sender calculates the applicable rate for the receiving process, $R_{rev}$, as:

$$R_{rev} = R_{rev} \times [1 - \alpha \times (D_{cur} - D_{thr}) - \beta \times (D_{cur} - D_{old})] \quad (1)$$

where $D_{old}$ is the receiving delay in the last round and we ensure that $R_{rev} \geq 0$. Thus, RHCC can dynamically adjust the rate to fully utilize the receiver intra-host resources. RHCC uses the Proportional Integral Derivative (PID) based adjustment to govern the dynamics of the rate. Compared with the parameter-fixed AIMD adjustment scheme, the error-based calibration and differential regulation terms enable RHCC to adaptively realize dynamic adjustment according to the real-time state of the receiver and efficiently utilize intra-host resources.

Then, RHCC combines the receiving process rate with the deployed inter-host methods by determining the transmitting rate„ $R_{trans}$, as:

$$R_{trans} = min(R_{rev}, R_{cc}) \quad (2)$$

where $R_{cc}$ is the rate calculated by the deployed inter-host congestion control algorithm. In this way, RHCC handles inter-host network congestion and can be deployed with different inter-host control methods. Meanwhile, RHCC achieves max-min fairness among RNIC traffic when allocating inter-host and intra-host resources.

## 4 RHCC EVALUATION

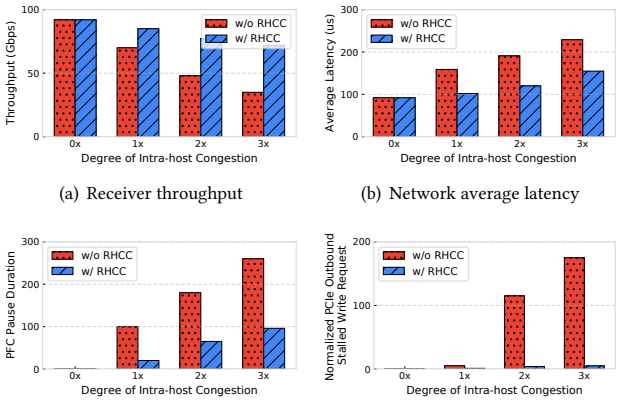
In this section, we implement RHCC on two servers and conduct experiments to evaluate the performance. The receiver-side RHCC is a loadable Linux kernel module with approximately 500 lines of C code. The sender-side RHCC is implemented using the PCC framework.

**Testbed setup.** We use the same testbed with two servers connected via a single switch as in §2.2. We implement RHCC on these two servers based on the previous analysis. The receiver-side RHCC is a loadable Linux kernel module with approximately 500 lines of C code. The sender-side RHCC is implemented using the PCC framework and is combined with DCQCN.

**System parameters.** RHCC generates a probe packet for every 32KB data packets to make a trade-off between network overhead and control granularity. We measure the receiver processing latency of 120$ns$ using the probe mechanism in the absence of intra-host congestion. The latency reaches approximately 900$ns$ congestion scenario. Thus, we simply set $D_{thr} = 200ns$, $\alpha = 0.0005$ and $\beta = 0.0005$ where the measured latency is in nanoseconds. We use the default DCQCN as the inter-host congestion control mechanism.

**Traffic loads.** We repeat the experiments in §2.2 and analyze the performance results.

[3]The five levels available memory bandwidth for intra-host traffic are 33.2GBps, 26.5GBps, 20.0GBps, 15.0GBps and 10.0GBps, respectively.




(a) Receiver throughput (b) Network average latency

(c) Average PFC (d) PCIe backpressure

**Figure 6: RHCC reduces PCIe backpressure and improves the throughput in the PFC-enabled cases. The results in PFC-free cases are similar and we omit them.**

**Performance metrics.** We have four major metrics: *(i)* inter-host traffic throughput; *(ii)* inter-host traffic latency; *(iii)* normalized PFC pause duration; *(iv)* normalized PCIe backpressure.

**[Figure 6] RHCC reduces PCIe backpressure by orders of magnitude and further improves the throughput even with high degree of intra-host congestion.** Figure 6 illustrates RHCC results using the same experiments as in Figure 2 with PFC-enabled scenario. We observe that RHCC can mitigate intra-host congestion and achieve better application-level performance under different degrees of congestion. Benefits from the intra-host traffic congestion response, as shown in Figure 6(d), RHCC significantly reduces the PCIe stalled write requests by 80%, 93% and 97% in the 1×, 2× and 3× congestion case, respectively. Therefore, the RNIC traffic can acquire sufficient intra-host resources and mitigate the triggering of inter-host congestion control mechanism. In Figure 6(c), we can see that the PFC pause duration has been reduced by up to 64% in the 3× congestion case, which improves utilization of inter-host network and avoids the potential PFC storm. As a result, the network throughput and average latency increases/decreases by up to 2× and 1.4×, respectively, in the 3× congestion case. We further conducted experiments in the PFC-free scenario and found that RHCC can also achieve better performance. The results are similar to the PFC-enabled scenario and we omit them.

## 5 DISCUSSION AND FUTURE WORK

**New intra-host architecture.** Compute Express Link (CXL) [33] is an emerging industry interconnect standard. CXL enables high-performance connectivity between CPU cores and CXL devices by maintaining a unified, coherent memory space. However, CXL requires extensive incremental modifications to existing commodity servers and the associated benefits of alleviating intra-host congestion merit future research. In addition, datacenter operators have observed other types of emerging intra-host congestion. For instance, commercial RNICs have limited on-chip SRAM and thus can cache only a small amount of QPs. When the number of QP exceeds the threshold, Interconnect Context Memory (ICM) cache miss and extra PCIe bandwidth increases, resulting in performance collapse [22]. New technology SRNIC [37] provides a cache-free QP scheduler to mitigate this congestion, which is orthogonal and complementary to RHCC.

**New resource allocation policy.** The intra-host resource allocation mechanism should support different underlying policies (e.g., weighted max-min fairness and priority schedule) between the intra-host traffic and the RNIC traffic. RHCC now supports max-min fairness among RNIC traffic and provides guarantee for intra-host traffic. We plan to incorporate other policies in the future.

**New DDIO optimization mechanism.** DDIO [1] leverages LLC for fast packet processing. Its benefit in RDMA networks is small due to the leaky DMA problem [18, 24, 38]. Some researchers propose dynamic LLC allocation mechanisms to improve the performance [18, 35, 38]. However, their benefits in the case of memory interconnect congestion merit future research.

**New programmable intra-host network.** In contrast to the inter-host network, the intra-host network lacks sufficient programmability. For instance, it is difficult to implement complex adjustment functions when using the intra-host resource allocation tool (e.g., Intel MBA). In addition, commercial RNIC only provides simple transmitted/received packet counters (e.g., Port/HW Counters [12]). Using diagnostic counters for debugging typically requires vendor's assistance and the number of counters is limited. For instance, it is difficult to obtain the RNIC buffer occupancy. We hope the next-generation intra-host network could provide opportunities to resolve these problems.

## 6 CONCLUSION

This paper rethinks intra-host congestion control for RDMA networks and presents RHCC, which combines intra-host traffic congestion avoidance with sub-RTT granularity and proactive RNIC traffic adjustment. RHCC ensures fast congestion avoidance and can work with different inter-host congestion controls. We implement RHCC on commodity servers and RNICs and conduct experiments to evaluate the performance. The extensive evaluations confirm that RHCC is a promising solution for strengthening the performance of intra-host RDMA networks. We believe RHCC is a start towards intra-host congestion control for future high-speed RDMA networks. *This work does not raise any ethical issues.*

## ACKNOWLEDGEMENT

This work is partly supported by the National Key R&D Program of China under Grant No. 2022YFB2901700, the Shandong Provincial Natural Science Foundation under Grant No. ZR2023LZH011, and the National Natural Science Foundation of China (NSFC) under Grant No. 62132022 and 62372053.

# REFERENCES


[1] 2012. Intel® Data Direct I/O Technology (Intel® DDIO): A Primer. https://www.intel.com/content/dam/www/public/us/en/documents/technology-briefs/data-direct-i-o-technology-brief.pdf. (2012).

[2] 2018. Mellanox perftest package. https://community.mellanox.com/docs/DOC-2802.. (2018).

[3] 2019. Intel Xeon Processor Scalable Family Datasheet. https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/2nd-gen-xeon-scalable-datasheet-vol-1.pdf. (2019).

[4] 2019. Introduction to Memory Bandwidth Allocation. https://www.intel.com/content/www/us/en/developer/articles/technical/introduction-to-memory-bandwidth-allocation.html. (2019).

[5] 2022. Intel® performance counter monitor. https://www.intel.com/software/pcm. (2022).

[6] 2022. Mellanox NEO-Host. https://support.mellanox.com/s/productdetails/a2v50000000N2OlAAK/mellanox-neohost. (2022).

[7] 2022. Nvidia dgx a100. https://www.nvidia.com/en-us/data-center/dgx-a100/. (2022).

[8] 2023. Intel. 2023. Intel® Memory Latency Checker. (2023). https://www.intel.com/content/www/us/en/developer/articles/tool/intelr-memory-latency-checker.html. (2023).

[9] 2023. Intel® 64 and IA-32 Architectures Software Developer Manuals. https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html. (2023).

[10] 2023. Intel® Resource Director Technology (Intel® RDT). https://www.intel.com/content/www/us/en/architecture-and-technology/resource-director-technology.html. (2023).

[11] 2023. NVLink and NVSwitch:Fastest HPC Data Center Platform. https://www.nvidia.com/en-us/data-center/nvlink/. (2023).

[12] 2023. Understanding MLX5 Linux counters and status parameters. https://enterprise-support.nvidia.com/s/article/understanding-mlx5-linux-counters-and-status-parameters. (2023).

[13] 2024. Intel® 64 and IA-32 Architectures Software Developer's Manual. https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html. (2024).

[14] Saksham Agarwal, Rachit Agarwal, Behnam Montazeri, Masoud Moshref, Khaled Elmeleegy, Luigi Rizzo, Marc Asher de Kruijf, Gautam Kumar, Sylvia Ratnasamy, David Culler, et al. 2022. Understanding host interconnect congestion. In *HotNets*.

[15] Saksham Agarwal, Arvind Krishnamurthy, and Rachit Agarwal. 2023. Host Congestion Control. In *SIGCOMM*.

[16] Wei Bai, Shanim Sainul Abdeen, Ankit Agrawal, Krishan Kumar Attre, Paramvir Bahl, Ameya Bhagat, Gowri Bhaskara, Tanya Brokhman, Lei Cao, Ahmad Cheema, et al. 2023. Empowering azure storage with RDMA. In *NSDI*.

[17] Jianbo Dong, Zheng Cao, Tao Zhang, Jianxi Ye, Shaochuang Wang, Fei Feng, Li Zhao, Xiaoyong Liu, Liuyihan Song, Liwei Peng, et al. 2020. Eflops: Algorithm and system co-design for a high performance distributed training platform. In *HPCA*.

[18] Alireza Farshin, Amir Roozbeh, Gerald Q Maguire Jr, and Dejan Kostić. 2020. Reexamining Direct Cache Access to Optimize {I/O} Intensive Applications for Multi-hundred-gigabit Networks. In *ATC*.

[19] Yixiao Gao, Qiang Li, Lingbo Tang, Yongqing Xi, Pengcheng Zhang, Wenwen Peng, Bo Li, Yaohui Wu, Shaozong Liu, Lei Yan, et al. 2021. When cloud storage meets rdma. In *NSDI*.

[20] Yimin Jiang, Yibo Zhu, Chang Lan, Bairen Yi, Yong Cui, and Chuanxiong Guo. 2020. A unified architecture for accelerating distributed {DNN} training in heterogeneous {GPU/CPU} clusters. In *OSDI*.

[21] Xinhao Kong, Jiaqi Lou, Wei Bai, Nam Sung Kim, and Danyang Zhuo. 2023. Towards a Manageable Intra-Host Network. In *Proceedings of the 19th Workshop on Hot Topics in Operating Systems*. 206–213.

[22] Xinhao Kong, Yibo Zhu, Huaping Zhou, Zhuo Jiang, Jianxi Ye, Chuanxiong Guo, and Danyang Zhuo. 2022. Collie: Finding Performance Anomalies in {RDMA} Subsystems. In *NSDI*.

[23] Gautam Kumar, Nandita Dukkipati, Keon Jang, Hassan MG Wassel, Xian Wu, Behnam Montazeri, Yaogong Wang, Kevin Springborn, Christopher Alfeld, Michael Ryan, et al. 2020. Swift: Delay is simple and effective for congestion control in the datacenter. In *SIGCOMM*.

[24] Qiang Li, Qiao Xiang, Derui Liu, Yuxin Wang, Haonan Qiu, Xiaoliang Wang, Jie Zhang, Ridi Wen, Haohao Song, Gexiao Tian, Chenyang Huang, Lulu Chen, Shaozong Liu, Yaohui Wu, Zhiwu Wu, Zicheng Luo, Yuchao Shao, Chao Han, Zhongjie Wu, Jianbo Dong, Zheng Cao, Jinbo Wu, Jiwu Shu, and Jiesheng Wu. 2023. From RDMA to RDCA: Toward High-Speed Last Mile of Data Center Networks Using Remote Direct Cache Access. (2023).

[25] Qiang Li, Qiao Xiang, Yuxin Wang, Haohao Song, Ridi Wen, Wenhui Yao, Yuanyuan Dong, Shuqi Zhao, Shuo Huang, Zhaosheng Zhu, et al. 2023. More than capacity: performance-oriented evolution of Pangu in Alibaba. In *FAST*.

[26] Kefei Liu, Zhuo Jiang, Jiao Zhang, Haoran Wei, Xiaolong Zhong, Lizhuang Tan, Tian Pan, and Tao Huang. 2023. Hostping: Diagnosing intra-host network bottlenecks in {RDMA} servers. In *NSDI*.

[27] Xiaoyi Lu, Nusrat S Islam, Md Wasi-Ur-Rahman, Jithin Jose, Hari Subramoni, Hao Wang, and Dhabaleswar K Panda. 2013. High-performance design of hadoop rpc with rdma over infiniband. In *IPCC*. IEEE.

[28] Xiaoyi Lu, Md Wasi Ur Rahman, Nusrat Islam, Dipti Shankar, and Dhabaleswar K Panda. 2014. Accelerating spark with RDMA for big data processing: Early experiences. In *2014 IEEE 22nd Annual Symposium on High-Performance Interconnects*. IEEE.

[29] Rolf Neugebauer, Gianni Antichi, José Fernando Zazo, Yury Audzevich, Sergio Lpez-Buedo, and Andrew W Moore. 2018. Understanding PCIe performance for end host networking. In *SIGCOMM*.

[30] NVIDIA. 2021. BlueField-2. https://resources.nvidia.com/en-us-accelerated-networking-resource-library/bluefield-2-dpu-datasheet?lx=LbHvpR&topic=networking-cloud. (2021).

[31] NVIDIA. 2022. BlueField-3. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/documents/datasheet-nvidia-bluefield-3-dpu.pdf. (2022).

[32] Satadru Pan, Theano Stavrinos, Yunqiao Zhang, Atul Sikaria, Pavel Zakharov, Abhinav Sharma, Mike Shuey, Richard Wareing, Monika Gangapuram, Guanglei Cao, et al. 2021. Facebook's tectonic filesystem: Efficiency from exascale. In *FAST*.

[33] Debendra Das Sharma. 2022. Compute Express Link (CXL): Enabling heterogeneous data-centric computing with heterogeneous memory hierarchy. *IEEE Micro* (2022).

[34] Yuval Shpigelman, Idan Burstein, Noam Bloch, Reut Zuck, and Roee Moyal. 2021. Programmable Congestion Control Communication Scheme. (May 20 2021). US Patent App. 16/986,428.

[35] Amin Tootoonchian, Aurojit Panda, Chang Lan, Melvin Walls, Katerina Argyraki, Sylvia Ratnasamy, and Scott Shenker. 2018. ResQ: Enabling {SLOs} in Network Function Virtualization. In *NSDI*.

[36] Minhu Wang, Mingwei Xu, and Jianping Wu. 2022. Understanding I/O direct cache access performance for end host networking. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* (2022).

[37] Zilong Wang, Layong Luo, Qingsong Ning, Chaoliang Zeng, Wenxue Li, Xinchen Wan, Peng Xie, Tao Feng, Ke Cheng, Xiongfei Geng, et al. 2023. SRNIC: A Scalable Architecture for {RDMA}{NICs}. In *NSDI*.

[38] Yifan Yuan, Mohammad Alian, Yipeng Wang, Ren Wang, Ilia Kurakin, Charlie Tai, and Nam Sung Kim. 2021. Don't forget the I/O when allocating your LLC. In *ISCA*. IEEE.

[39] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. 2015. Congestion control for large-scale RDMA deployments. *SIGCOMM* (2015).