GRC: Extending RDMA to Wide-Area Networks via Gateway-Mediated Rate Control

Jue Zhang*†, Menghao Zhang*†, Zihan Niu*, Bo Peng*, Shucan Yang*, Xiaohe Hu‡
*School of Software, Beihang University †State Key Laboratory of Internet Architecture, Tsinghua University ‡Infrawaves

***Abstract*—Remote Direct Memory Access (RDMA) has been widely adopted in data centers (DCs) due to its low latency and high throughput. Recently, there is growing interest in extending RDMA to support cross-DC services. However, existing long-haul RDMA studies mainly focus on dedicated optical fiber, and extending RDMA to multi-hop wide-area network (WAN) infrastructure is largely overlooked. WANs lack RDMA-specific in-network support and have high latency, challenging existing RDMA transports designed for lossless fabrics and rapid feedback loops. In this paper, we propose GRC, a Gateway-Mediated Rate Control mechanism designed to enable RDMA over managed or service-guaranteed WANs with minimal changes to existing infrastructure. Our key insight is to delegate WAN rate control to the DC gateway, which monitors WAN links, calculates a shared reference rate for each destination DC, and enforces it by sending Congestion Notification Packets (CNPs) back to source RNICs. This enables RNICs to adjust rates within one RTT and significantly reduces congestion-induced packet loss. GRC requires only a gateway upgrade and commonly available WAN-device configurations, while remaining compatible with existing DCQCN-based intra-DC congestion control, making it a practical solution for RDMA over managed WANs. Evaluation shows GRC achieves near-zero packet loss without PFC in our evaluated settings. Compared with representative RDMA transport mechanisms, GRC reduces inter-DC flow FCT by at least 55.8%.**

I. Introduction

Remote Direct Memory Access (RDMA) has gained widespread adoption due to its ability to bypass the kernel and offload the protocol stack to the Network Interface Card (NIC), offering high throughput and low latency with low host CPU overhead. Many data centers (DCs) deploy RDMA over Converged Ethernet Version 2 (RoCEv2) for communication within DC applications [1], [2]. Recent demand for cross-DC services has motivated studies on extending RDMA to inter-DC communication [3], [4], [5], [6], [7].

However, existing RDMA mechanisms are deeply tied to the assumptions of low-latency and RDMA-aware fabrics. RoCEv2 is highly sensitive to packet loss and commonly relies on Priority Flow Control (PFC) to provide a lossless environment, which requires switch headroom proportional to the bandwidth-delay product (BDP). RDMA congestion control schemes are also primarily designed for controlled DC fabrics, where congestion signals can be generated by switches and returned within short feedback loops [8], [9], [10], [11]. Recent long-haul RDMA systems reduce PFC headroom [4], [5] or improve congestion control [3], [6], [12], [7], but mainly focus on engineered inter-DC connectivity, which becomes increasingly restrictive for geographically dispersed DCs. In this paper, we target a different setting: interconnecting multiple geographically distributed DCs over existing multi-hop wide-area network (WAN) infrastructure, while requiring minimal changes to existing RDMA NICs (RNICs), DC networks, and WAN devices.

Migrating RDMA to such WAN environments is especially challenging. First, WAN paths lack the RDMA-specific fabric support assumed by existing RDMA transports. WAN devices may provide only generic forwarding and queue-management functions, rather than RDMA-specific support such as PFC, packet trimming [13], or In-band Network Telemetry (INT) [9]. Second, WAN-scale Round-Trip Time (RTT) and BDPs amplify the cost of delayed feedback. RDMA senders usually inject traffic aggressively to achieve high throughput, but over long feedback loops they may send a large amount of data before receiving the first congestion signal, causing packet loss and triggering costly Go-Back-N retransmissions. Recent lossy RDMA designs relax the lossless-fabric assumption by adding extra recovery mechanisms to the RNIC or the end-host stack [14], [15], [13], but they are primarily designed for controlled DC fabrics rather than WAN paths. Conversely, adopting TCP-like conservative startup and mature loss recovery can reduce burstiness, but sacrifices the aggressive, low-overhead, high-throughput data movement that motivates RDMA. Finally, unlike software transport stacks, RDMA transport logic is largely implemented in RNIC hardware, making protocol changes more costly.

Against this backdrop, we ask how far RDMA can be pushed over WANs while preserving today's RNICs and network hardware as much as possible. Because RDMA loss recovery is largely embedded in RNICs, we focus on reducing WAN-induced losses through rate control rather than fully redesigning loss recovery. Based on the preceding analysis, end-host-based approaches require each sender to independently infer WAN congestion and suffer from long feedback delays. We observe that outgoing inter-DC RDMA traffic naturally traverses the gateway switches that connect DCs to the WAN. This makes the gateway a natural control point for maintaining shared congestion state across flows toward the same destination DC (dst-DC), avoiding repeated congestion discovery. Moreover, using the gateway as the control point cleanly separates the heterogeneous intra-DC and inter-DC control domains. The gateway can handle WAN-facing congestion

979-8-3195-0662-7/26/$31.00 ©2026 IEEE

using aggregate dst-DC state, and translate its decisions into standard RoCEv2 congestion feedback that existing RNICs already understand. As a result, WAN-mediated rate control can evolve independently from intra-DC RDMA mechanisms, without requiring hardware-level modifications to RNICs or WAN devices.

Based on these observations, we propose GRC, a gateway-mediated rate control mechanism for RDMA over WANs that mitigates excessive WAN queue buildup and congestion-induced packet loss. GRC requires only a customized gateway and does not rely on future RNIC capabilities or RDMA-specific WAN support. As a deployment trade-off, GRC targets managed or Service-Level Agreement (SLA)-backed WANs where generic traffic-management mechanisms provide RDMA traffic with protection against loss-based transports. GRC comprises three key modules: Delay Monitor, Rate Calculation, and Rate Control. The first two estimate the WAN delay and derive an aggregate reference rate for each dst-DC, while Rate Control drives the aggregate sending rate toward this reference through DCQCN-compatible Congestion Notification Packets (CNPs). We implement a prototype of GRC and make its source code publicly available on GitHub [16]. Evaluation shows GRC achieves near-zero packet loss without PFC on WANs, remains stable under extremely high load with an average normalized Flow Completion Time (FCT) of only $1.7\times$ for inter-DC flows. In contrast, other line-rate-start-based approaches, such as DCQCN [8], selective retransmission, and UnoCC [17], incur substantial packet loss, retransmissions, and even congestion collapse. Compared with slow-start-based approaches such as GEMINI [18], GRC reduces the FCT of inter-DC flows by 55.8%–69.2%.

Our key contributions are:

- We present GRC, a gateway-mediated rate-control architecture that maintains destination-level WAN congestion state at the DC gateway, enabling congestion information to be shared across flows toward the same dst-DC.
- We design a gateway rate-control mechanism that converts WAN delay into an aggregate reference rate and actuates unmodified DCQCN RNICs through standard CNPs, without shaping or buffering data packets at the gateway.
- We implement GRC in NS-3 to evaluate its performance, fairness, and convergence, and prototype it on a P4-programmable switch to demonstrate its practical feasibility and quantify its resource overhead.

Finally, we provide a discussion in §V, survey related work in §VI, and conclude the paper in §VII.

II. Background and Motivation

A. Cross-DC Communication and the Limits of TCP

Modern cloud and AI workloads are increasingly deployed across geo-distributed DCs to overcome the resource limits of a single DC as well as improve reliability and flexibility. This shift makes cross-DC communication a critical performance bottleneck: traffic across regions is no longer limited to background backup or bulk transfer, but now carries latency- and throughput-sensitive workloads such as distributed storage, model training, cross-region inference, and federated learning [19], [20], [21], [22], [23]. Therefore, efficient cross-DC transport is essential for application performance and resource utilization.

Traditional cross-DC communication still largely relies on TCP-based software stacks due to their maturity and deployability. However, with WAN links reaching 100 Gbps and beyond, TCP is increasingly strained by large-BDP paths and CPU-intensive packet processing and memory operations [24], [25], [23], [26]. RDMA addresses these bottlenecks by offloading transport processing to RNICs and bypassing the kernel, reducing host CPU overhead while improving throughput efficiency [1], [2]. This makes RDMA a promising candidate for future cross-DC communication if it can be extended beyond tightly controlled single-DC environments.

Problem Scope. We target cross-DC RDMA over managed or SLA-backed multi-hop WANs designed to provide performant multi-DC connectivity. The operator controls each DC and its gateway, but requires no RDMA-specific support from the WAN. We assume differentiated treatment for RDMA through separate traffic-class queues, protected bandwidth, or fair scheduling against loss-based traffic. These capabilities are supported by carrier-grade devices [27], [28], [29] and exposed in managed/service-provider WANs [30], [31].

B. Existing RDMA Protocols

Using hardware offload, RDMA trades flexibility for performance. Consequently, it is generally deployed in relatively controlled environments within DCs. More specifically, many RDMA mechanisms rely on the following two assumptions: (**A1**): controlled or RDMA-aware fabric support; (**A2**): short RTTs or low BDPs.

Loss recovery. Traditional RoCEv2 uses Go-Back-N retransmission, making it highly loss-sensitive. Production deployments therefore rely on PFC for lossless Ethernet, which requires switches to provision headroom of approximately $2\times$BDP (**A1**, **A2**). Recent lossy RDMA designs relax the lossless-network assumption, but still rely on DCN-specific assumptions. IRN [14] tracks missing packets using a bitmap whose size scales with BDP, which strains the scarce RNIC SRAM in large BDP scenarios (**A2**) [23]. DCP [13] relies on switch-side packet trimming, and therefore assumes fabric support (**A1**).

Congestion control. Congestion control is crucial for controlling queue buildup, but existing RDMA congestion control schemes are largely tailored to DC environments, such as DCQCN [8], Timely [10], and HPCC [9]. They commonly assume that: (i) congestion feedback returns quickly enough to correct aggressive sending before excessive queues or losses develop, as in DCQCN and HPCC (**A2**); (ii) fabric-controlled congestion signals are available, such as Explicit Congestion Notification (ECN) marking in DCQCN and INT telemetry in HPCC (**A1**); and (iii) RTT references remain stable, such as Timely's RTT threshold and HPCC's baseRTT (**A2**).

Recent long-haul RDMA systems have extended RDMA beyond a single DC, but mainly over-engineered inter-DC con-

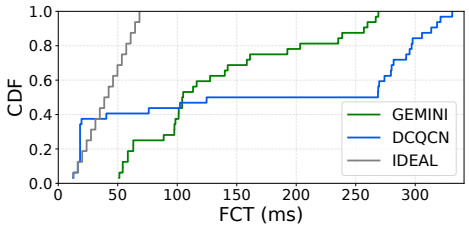


Fig. 1: FCT distribution under a cross-DC burst.

nectivity. Azure [3] first demonstrated regional-scale RDMA deployment in a carefully managed cloud infrastructure, while Bifrost [4] and Swing [5] reduce the buffer pressure in long-distance lossless RDMA. Orthogonally, BiCC [6], Themis [7], and Uno [17] improve congestion control fairness between inter-DC and intra-DC RDMA flows. However, these systems still assume dedicated paths and lossless or RDMA-aware fabric support, which limits their applicability to shared multi-hop WANs where such support cannot be assumed.

C. Challenges of Extending RDMA to WANs

Cross-DC networks exhibit fundamental differences from DCNs due to their inherent heterogeneity. In terms of RTT, flows with widely different RTTs (from μs to ms) may coexist in the same deployment, and the BDP can easily exceed 100 MB. From the fabric perspective, DCN switches are typically shallow-buffered and under the operator's direct control. In contrast, WAN devices often have deeper buffers and are managed across broader administrative domains. These differences break the assumptions of existing RDMA congestion-control mechanisms (§II-B): INT-like telemetry may be unavailable, ECN marking policies may not be controlled by the RDMA operator, and fixed-threshold RTT schemes cannot handle vastly different RTT baselines. Most importantly, RDMA's aggressive startup may inject excessive packets before delayed WAN feedback returns, causing severe packet loss. Under RoCEv2's Go-Back-N recovery, such losses can further amplify the performance impact of congestion-control failure.

A natural alternative is to borrow ideas from TCP-based WAN transports. However, many TCP schemes either use loss as a congestion signal (e.g., CUBIC [32]) or tolerate occasional loss during bandwidth probing (e.g., BBR [33]), which can trigger expensive loss recovery under RoCEv2. Delay-based control seems more appealing. GEMINI [18] follows this direction by using ECN for intra-DC congestion and RTT for inter-DC congestion, explicitly addressing DC/WAN heterogeneity. Nevertheless, its TCP-style conservative startup becomes costly at 100 Gbps and beyond. When the WAN BDP reaches hundreds of megabytes, even transfers of comparable size may spend most of their lifetime ramping up rather than sending at the available line rate. As a result, a mechanism that was acceptable at lower link speeds can severely underutilize ultra-high-speed WAN paths.

To make this failure mode concrete, we conduct an NS-3 experiment with three DCs connected by a shared WAN switch. All links run at 100 Gbps with 1 ms one-hop delay, and the WAN switch has a 120 MB buffer. Each DC contains four hosts with 100 Gbps RNICs. During an 80 ms burst, DC0 and DC1 each start one 50 MB RDMA flow to DC2 every 5 ms. We compare DCQCN (with ECN enabled in the WAN) and GEMINI, and also plot an *Ideal* baseline, which represents line-rate transmission with infinite switch buffering.

Figure 1 shows that DCQCN exhibits a clear bifurcation in FCT. Some flows finish quickly because RNICs inject packets at line rate before congestion feedback arrives, but once flows accumulate at the WAN bottleneck, severe loss triggers Go-Back-N retransmissions. These retransmissions overlap with newly arriving flows, creating a vicious cycle and a pronounced gap in the FCT distribution. The result would be even worse if ECN is not available on the WAN switch. In contrast, GEMINI gradually ramps up its sending rate, which smooths traffic bursts and reduces packet loss. As a result, it achieves lower overall FCT than DCQCN, but its fastest flows are much slower than DCQCN's early-completing flows, and its average FCT remains far above the *Ideal* baseline.

D. Problem Summary and Design Goals

We distill the previous observations into two fundamental challenges that jointly impede the extension of existing frameworks:

Challenge 1: RDMA offload limits protocol adaptability to heterogeneous WAN environments. Because RDMA transport logic is largely implemented in RNIC hardware, modifying loss recovery, congestion feedback processing, or per-packet scheduling is substantially harder than doing so in software transports. Moreover, a scheme must handle congestion from both the WAN and within the DC, and such high heterogeneity further complicates protocol design. Therefore, adapting RNIC-side mechanisms to heterogeneous WAN environments can introduce substantial design and deployment complexity.

Challenge 2: delayed feedback creates a throughput–reliability tradeoff. In high-latency WANs, injecting data aggressively to fill the pipe before any congestion signal returns may cause serious packet loss and degrade reliability. A conservative ramp-up avoids loss but sacrifices the throughput advantage that motivates RDMA. The continued growth of WAN link bandwidth further amplifies these problems by increasing the BDP and the cost of delayed feedback.

An intuitive approach is to wait for future generations of more capable switch/RNIC hardware and for WAN environments to provide richer congestion signaling. However, such an approach is expensive and subject to considerable uncertainty. We therefore seek a principled solution with two design goals:

- **Low-cost deployability.** Reuse existing RNICs, RDMA stacks, and WAN infrastructure to minimize migration cost and deployment changes.
- **WAN-efficient RDMA.** Preserve RDMA's low-CPU, high-throughput benefits over WAN RTTs while limiting packet loss and RoCEv2 recovery penalties.

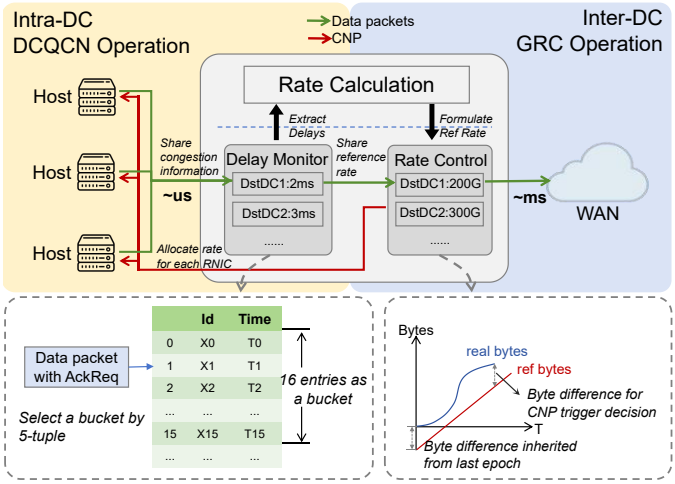


Fig. 2: GRC overview.

III. Design

A. GRC *Overview*

GRC approaches cross-DC RDMA from the perspective of rate control rather than loss recovery. This choice follows our deployability goal: loss recovery is deeply tied to RNIC hardware, and redesigning it would require intrusive changes to existing RDMA stacks. In contrast, rate control can reduce WAN-induced losses before recovery is invoked, allowing GRC to preserve existing RNIC recovery mechanisms while keeping loss rare in the common case. This focus remains valuable even with future loss-recovery improvements, because retransmissions still consume WAN bandwidth, increase completion latency, and interfere with concurrent traffic.

The challenges in §II-D make it difficult for a single end-host control loop to efficiently handle heterogeneous intra-DC and inter-DC conditions. In particular, a single RNIC must accommodate substantially different congestion environments (Challenge 1), while WAN-scale latency further delays its response to congestion (Challenge 2). GRC therefore moves WAN-facing rate control decisions to the gateway, decoupling inter-DC rate control from the intra-DC RDMA control loop. The workflow of GRC is shown in Fig. 2. Specifically, the gateway is responsible for centrally monitoring the status of wide-area network links to different DCs (*Delay Monitor* module), determining the aggregate sending rate toward each DC (*Rate Calculation* module), and allocating bandwidth among different flows destined for the same DC (*Rate Control* module). This architecture helps mitigate the two challenges as follows:

Concentrating complexity at the gateway. GRC requires replacing or upgrading only the DC gateway (e.g., a programmable switch), without hardware-level modifications to RNICs or WAN switches. Positioned at the DC–WAN boundary, GRC bridges the intra-DC and inter-DC control loops. We assume that DCQCN continues to be used as the default congestion control protocol within the DC.

Mitigating WAN feedback delay through shared congestion state. This placement of GRC is not merely a compromise imposed by limited RNIC programmability. It is also a cleaner system decomposition. With conventional end-host control, each new flow must discover congestion independently and wait for at least one WAN RTT before receiving its first congestion signal. GRC instead maintains destination-level congestion state at the gateway by aggregating signals from all flows heading to the same dst-DC. When a new flow reaches the gateway, GRC can immediately apply CNP feedback based on congestion information learned from earlier or concurrent flows, so each flow no longer needs to independently spend one or more WAN RTTs exploring the congestion state before reacting.

B. Delay Monitor

GRC operates on an epoch-by-epoch basis, where each control epoch lasts $epochDur$, typically on the order of milliseconds. The Delay Monitor is responsible for producing an epoch-level congestion signal for each dst-DC, which is later consumed by the Rate Calculation module. Therefore, it should provide a timely and representative estimate of the aggregate WAN delay toward each dst-DC.

However, existing mechanisms are not designed to provide millisecond-level delay estimates at low switch overhead. RouteScout [34] estimates RTT by monitoring the TCP handshake, which limits its sampling frequency to the order of seconds. DART [35], in contrast, needs to track flow sequence numbers, introducing complex state while achieving only tens-of-milliseconds monitoring granularity. Our key insight is that the Reliable Connection (RC) transport of RoCEv2 provides a special mechanism that makes this feasible: RoCEv2 data packets include an Acknowledgment Request (AckReq) bit in the IB header. When this bit is set, the receiver responds with an acknowledgment (ACK) packet. As a result, this provides a lightweight way to match data packets with their corresponding ACKs in the common case. For instance, assuming a packet payload of 1 KB and that 2% of packets have the AckReq bit set, a 100 Gbps traffic stream would generate approximately 250 AckReq-flagged packets per millisecond. This high frequency allows us to measure RTT by selecting a portion of these packets, recording their departure times, and waiting for the corresponding ACKs.

To minimize resource consumption, we store sampled AckReq packets from all traffic in a shared fixed-size hash table. Each sampled packet is hashed to an entry containing a timestamp and a compact identifier derived from its 5-tuple and sequence number, which distinguishes different packets mapped to the same entry. Upon ACK arrival, the monitor retrieves the corresponding entry to compute RTT and accumulates the RTT sum and sample count for the corresponding dst-DC. The control plane computes the average RTT for each dst-DC in every epoch. However, this naive design still leaves two key concerns unresolved: (1) whether the samples represent the aggregate dst-DC condition or are biased toward a few high-rate flows; (2) whether retransmissions can cause an ACK to be matched with the wrong transmission.

Flow-balanced sampling via two-level hashing. A flat hash table may be dominated by high-rate flows, causing the

Algorithm 1 Update Delay Entry

```
1: index ← hash(newPkt.5Tuple) × 16
          + hash(newPkt.seq) mod 16
2: if now − table[index].ts > Timeout then
3:    table[index].ts ← now
4:    table[index].identifier ← hash(newPkt)
5: else if table[index].identifier = hash(newPkt)
   then
6:    table[index].ts ← now
7: end if
```

measured RTT to reflect flow-specific conditions (e.g., local congestion within dst-DC) rather than the aggregate condition of a dst-DC. To limit this bias, GRC uses a two-level hash structure. It first maps a packet to a bucket using its 5-tuple, and then maps it to one of the entries inside the bucket using its sequence number, as shown in line 1, Algorithm 1. In this way, each flow can occupy only a bounded number of outstanding RTT entries, while different flows toward the same dst-DC still contribute samples to the epoch-level average.

Retransmission-aware ACK matching. Retransmissions create a more serious correctness issue. If the original transmission of an AckReq packet does not produce a valid returning ACK, but a retransmitted copy does, matching the ACK to the timestamp of the original transmission would incorrectly include the recovery time and overestimate RTT. Therefore, when an AckReq packet hits an occupied entry with the same packet identifier, GRC refreshes the timestamp instead of ignoring the packet, so the entry records the latest observed transmission (lines 5-6, Algorithm 1). We set the entry Timeout slightly below the RNIC's retransmission timeout (RTO). A rarer case is spurious retransmission, where the ACK for the original transmission is still in flight when the retransmitted copy refreshes the timestamp. This may produce an unrealistically small RTT sample. To avoid this, GRC discards RTT samples that are more than one RTO below the current RTT estimate.

C. Rate Calculation

The Rate Calculation module runs at the boundary of every control epoch. The control plane periodically collects delays from the Delay Monitor and uses this information to adjust the reference sending rate ($refRate$) for traffic destined for different dst-DCs. Our design should meet the following two conditions:

• **Queue-length control.** RDMA is highly sensitive to packet loss, so we bias control decisions toward avoiding loss (and thus limiting queue buildup). We exclude schemes that may incur loss during probing (e.g., BBR [33]).

• **Provide weighted fairness (optional)**. Consider the following scenario: DC1 and DC2 send data to DC3 simultaneously, where DC1 has one flow and DC2 has three flows sharing the same bottleneck. With traditional fairness in common congestion control mechanisms, the sum of the rates of the three flows from DC2 might end up being the same as that of the single flow from DC1. We aim to provide DC administrators with appropriate interfaces that enable them to automatically adjust the weights to a certain extent based on the DC's transmission intent (the number of flows).

Algorithm 2 Update $refRate$

```
1: epochsPerRTT ← baseRTT / epochDur
2: queueDelay ← newRTT − baseRTT
3: targetRate ← w / (δ · queueDelay)
4: step ← (v · w) / (δ · epochsPerRTT · baseRTT)
5: if targetRate > refRate then
6:    refRate ← refRate + step
7: else
8:    β_e ← 1 − (1 − β)^(1/epochsPerRTT)
9:    step ← max(step, (refRate − targetRate) · β_e)
10:   refRate ← refRate − step
11: end if
12: upperRate ← max(baseRate, realRate × 1.2)
13: if refRate > upperRate then
14:   refRate ← (refRate − upperRate) ∗ 0.8 + upperRate
15: end if
```

We draw inspiration from COPA [36], primarily because it delivers *predictable queue lengths* and *weighted fairness*. COPA can be viewed through a Network Utility Maximization (NUM) lens: each sender i is associated with a utility function that trades throughput for delay: $U_i(\lambda_i, d_s) = \log \lambda_i - \delta_i \log d_s$, where λ_i is the (long-term) sending rate and d_s denotes the (queueing-dominated) *switch delay*. The parameter δ_i controls the throughput-delay trade-off. To maximize U_i, it is required that $\lambda_i = \frac{1}{\delta \cdot d_s}$. Operationally, each sender periodically measures d_s and computes a target rate $\lambda_{t,i} = \frac{1}{\delta \cdot d_{s\_measured}}$, then moves its current rate λ_i toward this target based on the sign of $\lambda_{t,i} - \lambda_i$. At steady state, COPA has proved that (1) the standing queue length is proportional to $\sum \frac{1}{\delta_i}$, and (2) when multiple senders share the same bottleneck but use different δ_i values, their steady-state bandwidth shares are proportional to $1/\delta_i$.

However, COPA cannot be directly used in our gateway-based design. Due to the limited control capabilities of switches, it is difficult to implement a window-based congestion control algorithm at the gateway. Therefore, it must be reformulated as a rate-based algorithm. This change makes queue control harder. A window bounds the amount of in-flight data, while a rate target only constrains sending over time and cannot immediately limit transient backlog. COPA's AIAD update further aggravates this issue: its additive increase may continue building queues before feedback is reflected, and its additive decrease may drain queues too slowly after congestion appears. For RDMA over WAN, such transient queue peaks can cause packet loss. GRC therefore keeps COPA's inverse-delay target, but redesigns the update rule as a rate-based control loop with faster queue draining.

Algorithm 2 shows how GRC updates the destination-level reference rate once per epoch. The control plane first estimates the base RTT as the minimum delay ($baseRTT$) observed over a long time window, allowing GRC to track long-

term WAN RTT changes such as routing variations. Given a new RTT sample, GRC computes the queuing delay and derives a COPA-style target rate as $w/(\delta \cdot queueDelay)$ (lines 2–3). Here, w explicitly controls weighted fairness: instead of using different δ_i values as in COPA, all participating endpoints share the same δ while using different weights w_i. Section III-E explains how GRC estimates and updates w. GRC then computes a bounded per-epoch update step (line 4). The step is normalized by $epochsPerRTT$, so the rate can change only gradually over an RTT-scale time window. This bounded update reduces the impact of transient WAN jitter or occasional inaccurate RTT samples. Moreover, the velocity parameter v is increased only when the rate evolves in the same direction for multiple consecutive RTTs, preventing sporadic jitter-induced fluctuations from being amplified. By default, $v = 1$; it is increased to 2 or 4 only after the update direction remains unchanged for 5 or 10 consecutive RTTs, respectively. If the target rate is higher than the current $refRate$, GRC performs additive increase by one bounded step (lines 5–6). Otherwise, GRC decreases $refRate$ toward the target rate (lines 8–10). To avoid slow queue draining under severe congestion, GRC enforces a minimum decrease proportional to the gap between $refRate$ and $targetRate$, making the gap shrink by approximately a β fraction over one RTT when this minimum-decrease rule dominates. Therefore, this mechanism mainly takes effect when $refRate$ is far above $targetRate$, accelerating convergence and queue draining. Near equilibrium, the gap is small, and the additive step dominates, so β barely affects local behavior.

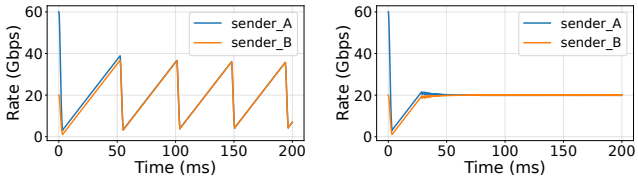
Another issue arises when the destination-level demand is below the reference rate. In this unsaturated regime, the gateway may maintain a large $refRate$, but the active RNICs do not have enough data to consume the corresponding rate budget. We introduce a single lightweight restriction on the reference rate: it is capped at 1.2× the $realRate$ observed in the previous epoch. If the $refRate$ exceeds this bound, it is forced to decay exponentially back toward the real rate and further bounded by $baseRate$ (lines 12-14). This simple but effective restriction ensures that the $refRate$ does not deviate excessively from the achievable network throughput. When a flow starts in isolation, GRC initializes it at $baseRate$, which is typically much faster than the conventional TCP-style slow start.

D. Rate Control

Rate Control is the actuation layer that translates $refRate$ into feedback understood by unmodified RNICs. It does not implement a proxy, shaper, or packet buffer at the gateway; all data packets are forwarded at line rate. Instead, GRC converts aggregate traffic that exceeds $refRate$ into DCQCN-compatible CNPs, so that RNICs react through their native congestion-control logic and drive the sending rate toward $refRate$. During this process, it should ensure that the resulting rate allocation is *fair* and *smooth* across flows, while maintaining no per-flow state at the gateway.

Algorithm 3 CNP Generation Logic

```
1: realBytes ← realBytes + pkt.size
2: bytesDiff ← realBytes − refBytes
3: pktCounter ← pktCounter + 1
4: if RED_CHECK(bytesDiff) ∨ pkt.ecn then
5:     sendCNP(pkt)
6:     cnpCounter ← cnpCounter + 1
7:     pkt.ecn ← 0
8: end if
```



(a) Signed carry-over. (b) Non-negative carry-over.

Fig. 3: Convergence behavior under different settings.

Our key insight is to expose excess aggregate traffic as a virtual-queue signal, so that GRC can drive the aggregate sending rate toward $refRate$ without enforcing rates at the gateway. Algorithm 3 is executed for each arriving packet. For each dst-DC, GRC maintains $realBytes$, the actually forwarded bytes, and $refBytes$, the byte budget accumulated according to $refRate$. It computes $bytesDiff = realBytes - refBytes$, and feeds this virtual excess into a RED-style CNP generation function (lines 1–7). Since high-rate flows send more packets, they are naturally more likely to receive CNPs, allowing fairness to emerge from RNICs' native DCQCN response rather than from per-flow scheduling at the gateway. The RED parameters $K_{\min}$, $K_{\max}$, and $P_{\max}$ are local to GRC and need not match the switch settings inside the DC. If a packet is already ECN-marked, GRC immediately returns a CNP and clears the mark, following the approach in Themis [7]. Finally, GRC maintains $pktCounter$ and $cnpCounter$ to estimate the CNP probability used by Rate Calculation.

Across epoch boundaries, a natural choice is to inherit the signed virtual difference from the previous epoch. Concretely, GRC can initialize the next epoch's $refBytes$ with the remaining gap between $refBytes$ and $realBytes$, so that both excess traffic and unused budget are carried over. Although this signed carry-over preserves long-term accounting and can eventually equalize competing flows, we find that it prevents the rates from converging. Figure 3 shows this effect using a fluid model of DCQCN. With signed carry-over, senders with different initial rates can become fair, but their rates keep oscillating instead of converging to the fair share. The reason is that $bytesDiff$ may become deeply negative when the aggregate rate is below $refRate$. During rate recovery, this negative credit delays feedback and causes the aggregate rate to overshoot $refRate$. The delayed CNPs then push the rates down again, leading to persistent oscillation.

To remove this stale-credit effect, GRC only carries over positive virtual backlog $\max(0, realBytes - refBytes)$ at epoch boundaries. Thus, excess traffic remains visible to the

next epoch, while unused budget is discarded. We could also clamp $bytesDiff$ on every packet, but doing so may waste transient unused budget under bursty mice-dominated traffic. Epoch-boundary clamping preserves short-term burst tolerance within an epoch while preventing negative credit from delaying future CNPs. As shown in Figure 3, this simple change enables both senders to converge smoothly to the fair shares.

E. GRC*'s Weighted Fairness*

Equal aggregate sharing may not fully reflect flow-level demand: a source DC with many active flows and a source DC with only one active flow would receive the same aggregate share. To address this, GRC supports weighted fairness as an optional policy mechanism. §III-C introduced the weight w, but determining w is non-trivial. Ideally, w should be proportional to the current number of concurrent flows in the aggregate. Since maintaining per-flow state consumes a large amount of switch memory, obtaining this number directly is difficult.

Our key insight is that, in a congestion control mechanism, the aggregate congestion signal contains indirect information about flow concurrency: with more active flows sharing a bottleneck, the gateway must generate stronger feedback to regulate the aggregate rate. Therefore, GRC infers flow concurrency from the intensity of aggregate congestion feedback. Suppose there are N flows at the bottleneck, and the bottleneck bandwidth is C. Previous work [37] proved that the steady-state CNP triggering probability can be approximated as:

$$p^* \approx \sqrt[3]{\frac{R_{AI}N^2}{\tau' C^2}\left(\frac{1}{B}+\frac{N}{T\cdot C}\right)^2}$$

Here, R_{AI}, τ', B, and T are DCQCN parameters, denoting the additive-increase step, the α update interval, the byte-count threshold for rate increase, and the timer-based rate-increase interval, respectively. Using the CNP and packet counters maintained by the Rate Control module, we estimate the triggering probability as $\hat{p} = \frac{N_{\text{CNP}}}{N_{\text{pkt}}}$, and replace C with $refRate$. In the regime where the $\frac{N}{C}$ term dominates, the model implies the compact scaling $N \propto \hat{p}^{\frac{3}{4}} \cdot refRate$.

Given the flow-concurrency surrogate $\hat{N} \triangleq p^{\frac{3}{4}} \cdot refRate$, we next map it to a concrete weight for each source DC. Given a maximum weight w_{max}, we adopt a monotone linear mapping with clipping $w = \text{clip}\left(k \cdot \hat{N}, 1, w_{max}\right)$. The parameters w_{max} and k are chosen to keep weights both effective and safe. The upper bound w_{max} is necessary for queue control: since the steady-state queue scale is proportional to $\sum_i w_i/\delta$, unbounded weights could lead to excessive queueing. In practice, w_{max} is configured according to operator policy and hardware limits, e.g., the ratio between the maximum egress rate of a DC and the maximum desired per-flow rate.

Given w_{max}, GRC determines k from operational samples. Each DC records the estimated concurrency surrogate $\hat{N}$ in every epoch, and DCs periodically synchronize these samples. GRC then selects a global k such that as many samples of $k \cdot \hat{N}$ as possible fall within $[1, w_{max}]$, so that most operating points avoid being clipped to either boundary.

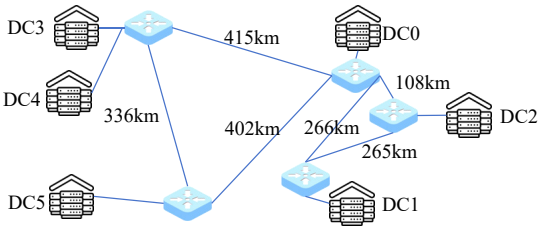


Fig. 4: Simulation topology.

IV. Evaluation

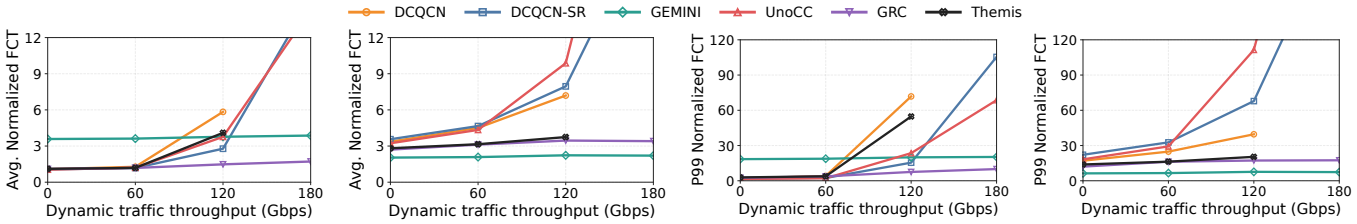
In this section, we implement and evaluate GRC to answer the following questions: (1) Can GRC perform efficiently under various and complex circumstances (§IV-B)? (2) How do different parameters and components affect GRC's performance (§IV-C)? (3) Can GRC be implemented on current programmable switches with acceptable resource overhead (§IV-B)?

A. Experimental Setup

Implementation. We implement GRC in the NS-3 simulator and prototype it on a P4-programmable Tofino switch. The source code is publicly available on GitHub [16]. For the Tofino prototype, the data plane part of GRC is implemented with ∼800 lines of P4-16 code. It measures delay and records the results in registers, while monitoring the traffic rate of each DC to generate CNPs upon congestion. We set the size of the register used to store packet arrival timestamps to 1024. The control plane part is written in ∼600 lines of C code. It periodically reads RTT and traffic statistics from the data plane registers. Subsequently, it adjusts the $refRate$ accordingly and writes the updated rate limits back to the data plane.

Topology. Figure 4 shows the NS-3 simulation topology, which includes 5 WAN switches and 6 DCs. The geographical distribution of WAN switches is extracted from a subset of CERNET [38]. We convert every 100 km into a delay of 500 μs and set the bandwidth of WAN links to 400 Gbps. Then, we connect 6 DCs to WAN switches, and the delay between the gateway switch and the WAN switch is 300 us. Each DC employs a Fat-Tree topology (k = 4) with an oversubscription ratio of 1:1. The gateway is connected to four switches of the core layer with 200 Gbps bandwidth and 1 μs latency, while other links within DC are 100 Gbps with 1 μs latency. The switch buffers of DCN, gateway, and WAN are set to 9 MB [39], 160 MB, and 320 MB respectively.

Traffic workload. We use WebSearch [9] workloads to determine the flow size distributions. We first generate intra-DC traffic at 30% load in each DC. Next, we generate inter-DC traffic and set the average throughput between each pair of DCs to 30 Gbps as the background. Finally, we inject some "dynamic traffic". For a given average dynamic throughput for each DC, we divide these flows into several groups. Each group of flows lasts for 30 ms and is destined to the same DC (groups may overlap in time). This creates unpredictable hotspots, increasing the likelihood of WAN incast. We evaluate



(a) Inter Avg. FCT. (b) Intra Avg. FCT. (c) Inter P99 FCT. (d) Intra P99 FCT.

Fig. 5: Normalized FCT under WebSearch traffic pattern.

GRC under different loads by regulating the amount of dynamic traffic. All traffic workloads last for 100 ms.

Baselines and Metrics. We compare GRC with native RDMA and representative baselines. *DCQCN* represents native RoCEv2 transport [8]. *DCQCN-SR* augments native DCQCN with ideal selective retransmission. This diagnostic baseline isolates whether improving loss recovery alone is sufficient under WAN congestion, while keeping the same end-host congestion-control behavior. *GEMINI* represents a cross-DC congestion-control design for heterogeneous intra- and inter-DC traffic [18]. *Themis* is a recent cross-DC RDMA scheme that addresses congestion-induced unfairness between intra- and inter-DC traffic [7]. *UnoCC* denotes the congestion-control component of Uno, without its erasure-coding-based reliability component or sub-flow load balancing [17]. Unless otherwise stated, GRC uses $epochDur = 1$ ms, $1/\delta = 6$ MB, $\beta = 0.3$, $baseRate = 0.25 \times link\ bandwidth$, $(K_{\min}, K_{\max}, P_{\max}) = (100\text{ KB}, 8\text{ MB}, 0.5)$, and $w_{\max} = 4$. We use the normalized FCT, i.e., a flow's actual FCT normalized by its base FCT when no other flows compete with it as the primary metric. We exclude PFC deployment over WAN—not only for device support issues, but also as the PFC configuration headroom in our scenario hits 600 MB, exceeding the buffer limit. Except for DCQCN-SR, all baselines use Go-Back-N retransmission.

B. Overall Effectiveness

FCT reduction. The overall FCT under WebSearch workload is shown in Figure 5. We observe that the baseline RDMA exhibits a bimodal behavior. Under low load, RDMA achieves low FCT because the switch buffers can absorb occasional bursts. As the load increases, however, burst packets overflow the switch buffers, resulting in packet loss. The subsequent Go-Back-N retransmissions cause retransmitted traffic to overlap with new flows, further amplifying congestion. In some high-load settings, this feedback loop leads to congestion collapse, and some simulations fail to complete within a reasonable time. We therefore terminate runs that exceed 10 hours (most experiments complete within about 3 hours). Even ideal selective retransmission does not address this issue. For congestion-control-level failures, selective retransmission alone cannot fully avoid the same vicious cycle. Since the original UnoCC design does not consider congestion within the WAN and preserves line-rate startup, it remains unable to prevent packet losses in the WAN. Themis maintains low intra-DC FCT, but cannot effectively handle WAN congestion under high load. GEMINI's slow-start-like behavior inherently penalizes inter-DC flows and achieves only 57.4% of GRC's WAN-link utilization in the first 100 ms under the highest load. In contrast, RDMA with GRC remains stable across the entire load range. The FCT under high load increases only slightly compared to low load, indicating that GRC can effectively handle burst traffic without sacrificing overall performance. Under high load, GRC reduces the average FCT of inter-DC flows by at least 55.8%. This stability comes from GRC sharing congestion information across flows, compensating for missing congestion signals and delivering feedback within a few microseconds, enabling RNICs to react promptly and accurately.

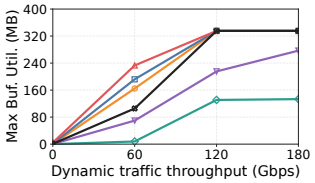


Fig. 6: Max buffer utilization.

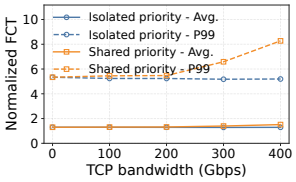


Fig. 7: TCP Coexistence.

WAN switch buffer utilization. To further understand the root causes of the performance gap, Figure 6 reports the max buffer utilization on WAN switches. DCQCN, Themis, DCQCN-SR, and UnoCC all reach the maximum buffer capacity of 320 MB under high load, thereby triggering substantial packet loss. Compared with GRC, GEMINI's overly conservative sending strategy reduces WAN switch buffer occupancy, but significantly increases the FCT of inter-DC flows. To determine whether GRC's benefit comes solely from avoiding buffer overflow, we repeat the 180 Gbps dynamic WebSearch experiment with WAN/gateway buffers scaled from 320/160 MB to $6\times$ their default sizes. At $6\times$, forwarding loss disappears for Themis, DCQCN, and UnoCC, while GEMINI remains loss-free. GRC still reduces average inter-DC FCT by 16.8%, 28.8%, 43.0%, and 53.8% over Themis, DCQCN, UnoCC, and GEMINI, respectively, showing that its benefit is not merely due to avoiding buffer overflow.

Coexistence with TCP traffic. We use queue isolation as a representative case of our intended deployment model and run RDMA and TCP traffic concurrently on the WAN. Protected bandwidth and fair scheduling are also within our intended deployment scope, but we do not evaluate them separately. Based on the topology in Figure 4, we inject continuous TCP traffic between WAN switches. We use CUBIC [32] as the TCP congestion control algorithm. We consider two

TABLE I: Resource overhead of GRC on switch.

Computational		Memory	
ALUs	**Gateways**	**SRAM**	**TCAM**
15.91%	9.09%	3.30%	0.38%

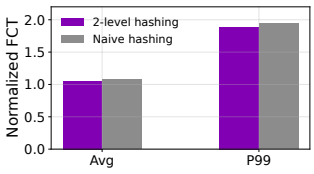


Fig. 8: Influence of 2-level hashing.

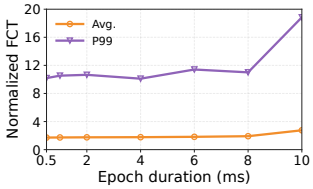


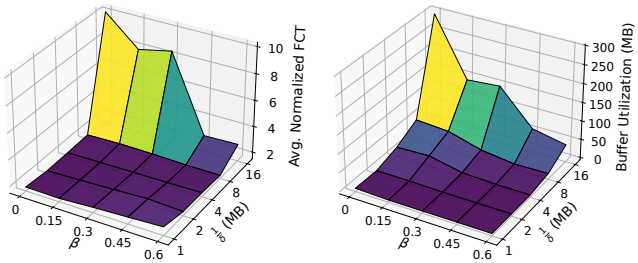
Fig. 9: Influence of epoch duration.

configurations: isolated priority, where RDMA and TCP use separate priority queues scheduled with round-robin and a dedicated 160 MB TCP buffer on the WAN switch, and shared priority, where they share the same queue and buffer under a first-in-first-out (FIFO) scheduling policy. We use the same workload as the 60 Gbps dynamic-traffic setting in Figure 5. As shown in Figure 7, the average and P99 FCTs of inter-DC RDMA traffic show negligible changes under isolated priority, compared with increases of up to 16% and 55% under shared priority. Although almost no packet loss occurs, shared priority increases the average WAN-switch buffer occupancy by up to 16% and reduces the average RDMA throughput by up to 34% compared with isolated priority, demonstrating the impact of contention with buffer-filling TCP traffic. Loss-based TCP inherently disadvantages delay-based GRC in shared queues by filling buffers and degrading performance. In contrast, GRC's performance remains stable with queue isolation.

Hardware resource overhead. We run GRC on a Tofino switch and report its resource utilization in Table I. The timestamp table stores outstanding sampled AckReq packets rather than per-flow state, so its required capacity mainly depends on the sampling rate and WAN RTT. It need not capture all AckReq packets, but only enough samples for reliable delay estimation. In our 6-DC setting, 1,024 entries provide tens of RTT samples per dst-DC per epoch, which is sufficient for this purpose. While entry collisions can be frequent, packet identifiers prevent erroneous matches. With the 24-bit identifier used in our simulation, the probability of an erroneous match is below 0.03% per epoch under the evaluated setting. Their impact is further mitigated by averaging RTT samples within each epoch. On Tofino, the table can scale to 65,536 entries while consuming only 6.48% of SRAM and 8.33% of Map RAM on average.

C. GRC *Deep Dive*

Impact of localized congestion interference. A concern is that localized congestion within the dst-DC may bias GRC's delay estimate and unnecessarily throttle unaffected flows. We generate 100 Gbps WebSearch background traffic from DC1 to DC0 and inject two large flows from DC1 and DC2 to the same DC0 host, creating localized congestion while leaving most background flows unaffected. Figure 8 compares two-level hashing with naive hashing. Although the congested flow accounts for 50% of DC1-to-DC0 packets, two-level hashing confines the flow to a single bucket, substantially reducing its impact on delay estimation and improving isolation and performance.



(a) Avg. Inter FCT. (b) P99 buffer utilization.

Fig. 10: Parameter sensitivity of β and δ.

TABLE II: Impact of dynamic weighting on rate allocation.

Dynamic w	Source DC	Rate (Gbps)	Avg. $\hat{N}$	Avg. w
Enabled	DC0	210.25	2.15×10^8	3.00
Enabled	DC1	130.88	1.07×10^8	1.99
Disabled	DC0	173.72	2.22×10^8	2.50
Disabled	DC1	165.95	9.79×10^7	2.50

GRC's weighted fairness. We evaluate dynamic weighting with DC0 and DC1 simultaneously sending eight and four 1 GB flows, respectively, to the same dst-DC. When dynamic weighting is disabled, we fix the weight of each source DC to $(1 + w_{\max})/2$. As shown in Table II, fixed weights give DC0 and DC1 similar rates despite their different flow counts. Enabling dynamic weights increases DC0's rate and reduces DC1's rate, substantially improving weighted fairness. Moreover, the ratio of the estimated flow-count metric $\hat{N}$ closely matches the expected 2:1 ratio when dynamic weighting is enabled. The measured weights and final rates do not exactly reach a 2:1 ratio because rate convergence takes time and the weight clipping mechanism introduces bounded distortion. The aggregate rate is slightly below the 400 Gbps bottleneck capacity because the COPA-style controller periodically drains the queue, allowing new senders to obtain reliable $baseRTT$ measurements.

Ablation study on the Rate Calculation module. We study the impact of two other components in the Rate Calculation module—dynamic w and explicit handling of *unsaturated sending* (Lines 12–15 of Algorithm 2)—both of which are absent in COPA. For the remaining experiments in this subsection, we use the 180 Gbps dynamic-traffic setting from §IV-B. The experimental results are presented in Table III. Compared with the original GRC, performance degrades to varying degrees across all cases, especially in P99 inter-FCT. Both of the above components help GRC to control the injection of traffic into the WAN more prudently and reduce the burstiness of traffic.

Parameter sensitivity. GRC mainly depends on δ and β, which control the steady-state queue target and the pace of MD, respectively. Figure 10(a) shows the FCT of inter-DC

TABLE III: Ablation results of Rate Calculation.

Variant	Avg. inter-FCT	P99 inter-FCT
Full	1.71	9.94
w/o saturation-rate check	1.73	10.25
w/o dynamic w	1.74	11.14

flows under different values of δ and β. Overall, the FCT exhibits a U-shaped curve as a function of $\frac{1}{\delta}$. When $\frac{1}{\delta}$ is excessively small, GRC makes more conservative decisions, which can lead to underutilization of throughput. In contrast, when $\frac{1}{\delta}$ is overly large, it behaves too aggressively, resulting in worse FCT performance. Furthermore, we observed packet loss in simulations when $\frac{1}{\delta}$ reaches 16 MB, which accounts for the sharp degradation in performance. Figure 10(b) further shows that P99 buffer utilization grows approximately linearly with $\frac{1}{\delta}$, providing a useful guideline for parameter tuning. For reasonable $\frac{1}{\delta}$, β reduces peak buffer utilization by accelerating rate decrease when *refRate* substantially exceeds *targetRate*. Its effect diminishes at large $\frac{1}{\delta}$ because the target rate itself becomes overly high.

Figure 9 shows the impact of $epochDur$. GRC's performance varies moderately with $epochDur$ from 0.5 to 8 ms, while at 10 ms, the average and P99 normalized FCT increase to 2.76 and 18.84, respectively, accompanied by a 0.73% packet loss rate. Considering the control-plane–data-plane interaction overhead, we set $epochDur$ to 1 ms by default; prior Tofino-based systems demonstrate that millisecond-scale control intervals are practical [40], [41].

V. Discussion

Extending GRC to different scenarios. The decoupled design of GRC allows the intra-DC and inter-DC control loops to evolve independently to some extent. On the inter-DC side, GRC currently derives the $refRate$ from the Delay Monitor; more explicit WAN telemetry (e.g., INT) or a centralized controller could instead augment or replace this process. On the intra-DC side, the current design relies on DCQCN-compatible CNP actuation. Supporting a non-CNP-based transport would require a different actuation interface and re-deriving the weighted-fairness model, which is left for future work.

Coexistence with buffer-filling transports. Coexistence with loss-based, buffer-filling transports is a known limitation of delay-based and queue-controlling congestion-control schemes [18]. COPA addresses this issue by detecting buffer-fillers or elastic cross traffic and switching to a TCP-competitive mode. GRC could adopt a similar strategy, e.g., by relaxing its delay target or adjusting the effective δ. However, competing with buffer-filling transports incurs more packet loss and unstable queueing delay, which not only places greater demands on RNIC loss recovery but also undermines the high-performance rationale for deploying RDMA. As a deployment trade-off, we therefore assume that RDMA traffic is protected through queue isolation, protected bandwidth, or fair scheduling, and treat strict shared-queue TCP/RDMA coexistence as out of scope.

GRC in multi-gateway and multipath scenarios. A DC may use multiple egress gateways, where ACKs may return via a different gateway, affecting the Delay Monitor. A minimal-change extension is to run an independent GRC instance on each gateway, while using packet mirroring to recover RTT samples across gateways [42]. However, independent $refRate$ decisions may introduce inter-gateway competition and affect stability, potentially requiring further tuning and validation. Separately, the WAN may provide multiple paths, which can reduce the effectiveness of GRC's dst-DC-level decisions. GRC can instead be combined with techniques such as segment routing (SR) to make decisions at the <dst-DC, path> granularity [43], [44], and can also be integrated with traffic engineering for more holistic optimization, which we leave to future work.

VI. Related Work

Congestion control. In recent years, congestion control mechanisms have been extensively studied. The most common approach leverages delay and packet loss [45], [36], [46], [18], [33]. Other works introduce proactive congestion control [47], [48], [49], [50], [51], [52], [53]. However, due to packet loss, RNIC modifications, or additional assumptions about network infrastructure, these approaches are difficult to apply to our scenario. There are also some works exploring automatic parameter tuning [54], [55], which is orthogonal to our work.

Lossy RDMA. Another line of research focuses on improving the robustness of RDMA in lossy environments [14], [15], [23], [56], [57], [13] by modifying RNICs. IRN [14] introduces selective retransmission in RNICs, but requires 5× BDP-sized bitmaps. SRNIC [15] onloads bitmaps into host memory, which also harms performance if packet loss is too frequent. DCP [13] is a switch–RNIC co-designed RDMA transport for high-speed lossy fabrics that provides a lightweight lossless control plane (via header-only packet trimming). LoWAR [57] is specifically designed for RDMA over WAN and introduces FEC to combat packet loss. These works can be integrated with GRC to achieve better transmission performance.

VII. Conclusion

To extend RDMA over WANs with limited RDMA-specific support and long RTTs, we propose GRC, a gateway-based rate control mechanism. GRC monitors WAN links at the gateway to calculate and enforce a shared, congestion-aware sending rate for all traffic destined for the same dst-DC. NS-3 evaluations demonstrate GRC's performance benefits, while the P4 prototype validates its implementation feasibility.

Acknowledgment

We thank our shepherd and the anonymous ICNP reviewers for their valuable comments. This work is supported in part by the National Natural Science Foundation of China (No. 62402025), the Open Research Fund of State Key Laboratory of Internet Architecture (HLW2025ZD01), and the Fundamental Research Funds for the Central Universities. Menghao Zhang is the corresponding author.

REFERENCES

[1] J. Hu, H. Shen, X. Liu, and J. Wang, "RDMA transports in datacenter networks: survey," *IEEE Network*, vol. 38, no. 6, pp. 380–387, 2024.

[2] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn, "RDMA over commodity Ethernet at scale," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 202–215.

[3] W. Bai *et al.*, "Empowering azure storage with RDMA," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 2023, pp. 49–67.

[4] P. Yu *et al.*, "Bifrost: Extending RoCE for long distance inter-DC links," in *2023 IEEE 31st International Conference on Network Protocols (ICNP)*. IEEE, 2023, pp. 1–12.

[5] Y. Chen *et al.*, "Swing: Providing long-range lossless RDMA via PFC-relay," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 1, pp. 63–75, 2022.

[6] Z. Wan *et al.*, "BiCC: Bilateral congestion control in cross-datacenter RDMA networks," in *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 2024, pp. 1381–1390.

[7] Z. Niu, M. Zhang, J. Zhang, R. Xie, Y. Yang, and X. Hu, "THEMIS: Addressing congestion-induced unfairness in long-haul RDMA networks," in *2025 IEEE 33rd International Conference on Network Protocols (ICNP)*. IEEE, 2025, pp. 1–13.

[8] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, "Congestion control for large-scale RDMA deployments," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 523–536, 2015.

[9] Y. Li *et al.*, "HPCC: High precision congestion control," in *Proceedings of the ACM special interest group on data communication*, 2019, pp. 44–58.

[10] R. Mittal *et al.*, "TIMELY: RTT-based congestion control for the datacenter," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 537–550, 2015.

[11] Y. Zhang, Q. Meng, C. Hu, and F. Ren, "Revisiting congestion control for lossless Ethernet," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, 2024, pp. 131–148.

[12] M. Long, J. Han, W. Wang, J. Yang, and K. Xue, "LSCC: Link-segmented congestion control for RDMA in cross-datacenter networks," in *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*. IEEE, 2024, pp. 1–10.

[13] W. Li, X. Liu, Y. Zhang, Z. Wang, W. Gu, T. Qian, G. Zeng, S. Ren, X. Huang, Z. Ren *et al.*, "Revisiting RDMA reliability for lossy fabrics," in *Proceedings of the ACM SIGCOMM 2025 Conference*, 2025, pp. 85–98.

[14] R. Mittal *et al.*, "Revisiting network support for RDMA," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 313–326.

[15] Z. Wang, L. Luo, Q. Ning, C. Zeng, W. Li, X. Wan, P. Xie, T. Feng, K. Cheng, X. Geng *et al.*, "SRNIC: A scalable architecture for RDMA NICs," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 2023, pp. 1–14.

[16] GRC, "GRC-NS3," https://github.com/Networked-System-and-Security-Group/GRC, 2026.

[17] T. Bonato, S. Abdous, A. Kabbani, A. Ghalayini, N. Gebara, T. Lam, A. Agarwal, T. Chen, Z. Yu, K. Taranov *et al.*, "Uno: A one-stop solution for inter-and intra-data center congestion control and reliable connectivity," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2025, pp. 1195–1210.

[18] G. Zeng, W. Bai, G. Chen, K. Chen, D. Han, Y. Zhu, and L. Cui, "Congestion control for cross-datacenter networks," *IEEE/ACM Transactions on Networking*, vol. 30, no. 5, pp. 2074–2089, 2022.

[19] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[20] K. Hsieh *et al.*, "Gaia: Geo-Distributed machine learning approaching LAN speeds," in *14th USENIX symposium on networked systems design and implementation (NSDI 17)*, 2017, pp. 629–647.

[21] A. Choudhury, Y. Wang, T. Pelkonen, K. Srinivasan, A. Jain, S. Lin, D. David, S. Soleimanifard, M. Chen, A. Yadav *et al.*, "MAST: Global scheduling of ML training across Geo-Distributed datacenters at hyperscale," in *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, 2024, pp. 563–580.

[22] H. Lim, J. Ye, S. Abdu Jyothi, and D. Han, "Accelerating model training in multi-cluster environments with consumer-grade GPUs," in *Proceedings of the ACM SIGCOMM 2024 Conference*, 2024, pp. 707–720.

[23] K. Lv, J. Li, P. Zhang, H. Pan, L. Li, S. Hu, Z. Li, G. Xie, J. Zhou, and K. Tan, "OmniDMA: Scalable RDMA transport over WAN," in *Proceedings of the 9th Asia-Pacific Workshop on Networking*, 2025, pp. 135–141.

[24] R. Singh, N. Bjorner, S. Shoham, Y. Yin, J. Arnold, and J. Gaudette, "Cost-effective capacity provisioning in wide area networks with SHOOFLY," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 534–546.

[25] ESnet, "Host Tuning: Background Information," https://fasterdata.es.net/host-tuning/background/, 2024, last edited: 2024-01-07. Accessed: 2025-02-11.

[26] H. N. Schuh, A. Krishnamurthy, D. Culler, H. M. Levy, L. Rizzo, S. Khan, and B. E. Stephens, "CC-NIC: a cache-coherent interface to the NIC," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, 2024, pp. 52–68.

[27] Cisco Systems, Inc., *Modular QoS Configuration Guide for Cisco NCS 540 Series Routers, Cisco IOS XR Release 26.1.x, 26.2.x*, Cisco Systems, Inc., 2026, accessed: 2026-08-10. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/iosxr/ncs5xx/qos/26xx/b-qos-cg-26xx-ncs540/config-mod-qos-congestion-management.html

[28] Juniper Networks, *Understanding CoS Output Queue Schedulers*, Juniper Networks, junos OS Documentation, accessed: 2026-08-10. [Online]. Available: https://www.juniper.net/documentation/us/en/software/junos/traffic-mgmt-qfx/topics/concept/cos-qfx-series-schedulers-understanding.html

[29] Arista Networks, *EOS 4.36.1F: Quality of Service*, Arista Networks, arista EOS User Manual, accessed: 2026-08-10. [Online]. Available: https://www.arista.com/en/um-eos/eos-quality-of-service

[30] Google Cloud, "Configure traffic differentiation for cloud interconnect," Google Cloud Documentation, accessed: 2026-08-11. [Online]. Available: https://docs.cloud.google.com/network-connectivity/docs/interconnect/how-to/cci/configure-traffic-differentiation

[31] F. Le Faucheur and W. S. Lai, "Requirements for support of differentiated services-aware MPLS traffic engineering," RFC Editor, RFC 3564, Jul. 2003. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3564

[32] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.

[33] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time," *Queue*, vol. 14, no. 5, pp. 20–53, 2016.

[34] M. Apostolaki, A. Singla, and L. Vanbever, "Performance-driven internet path selection," in *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR)*, 2021, pp. 41–53.

[35] S. Sengupta, H. Kim, and J. Rexford, "Continuous in-network round-trip time monitoring," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 473–485.

[36] V. Arun and H. Balakrishnan, "Copa: Practical Delay-Based congestion control for the internet," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, 2018, pp. 329–342.

[37] Y. Zhu, M. Ghobadi, V. Misra, and J. Padhye, "ECN or delay: Lessons learnt from analysis of DCQCN and TIMELY," in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, 2016, pp. 313–327.

[38] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[39] C. H. Song, X. Z. Khooi, R. Joshi, I. Choi, J. Li, and M. C. Chan, "Network load balancing with in-network reordering support for RDMA," in *Proceedings of the ACM SIGCOMM 2023 Conference*, 2023, pp. 816–831.

[40] L. Yu, J. Sonchack, and V. Liu, "Mantis: Reactive programmable switches," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '20. Association for Computing Machinery, 2020, pp. 296–309.

[41] C. Zeng, L. Luo, T. Zhang, Z. Wang, L. Li, W. Han, N. Chen, L. Wan, L. Liu, Z. Ding, X. Geng, T. Feng, F. Ning, K. Chen, and C. Guo, "Tiara: A scalable and efficient hardware acceleration architecture for stateful layer-4 load balancing," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. USENIX Association, Apr. 2022, pp. 1345–1358.

[42] The P4.org Architecture Working Group, "$P4_{16}$ Portable Switch Architecture (PSA)," P4 Language Consortium, Tech. Rep., Dec. 2022, version 1.2. [Online]. Available: https://p4.org/p4-spec/docs/PSA-v1.2.html

[43] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, "Segment Routing Architecture," Internet Engineering Task Force, Tech. Rep. RFC 8402, Jul. 2018. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8402

[44] S. Gay, R. Hartert, and S. Vissicchio, "Expect the Unexpected: Sub-Second Optimization for Segment Routing," in *Proceedings of IEEE INFOCOM 2017*, 2017, pp. 1–9.

[45] L. S. Brakmo *et al.*, "TCP vegas: New techniques for congestion detection and avoidance," in *Proceedings of the conference on Communications architectures, protocols and applications*, 1994, pp. 24–35.

[46] K. Tan *et al.*, "A compound TCP approach for high-speed and long distance networks," in *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*. IEEE, 2006, pp. 1–12.

[47] S. Hu *et al.*, "Aeolus: A building block for proactive transport in datacenters," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 422–434.

[48] I. Cho, K. Jang, and D. Han, "Credit-scheduled delay-bounded congestion control for datacenters," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 239–252.

[49] Y. Liu *et al.*, "Fork: A dual congestion control loop for small and large flows in datacenters," in *Proceedings of the Twentieth European Conference on Computer Systems*, 2025, pp. 446–459.

[50] B. Montazeri *et al.*, "Homa: A receiver-driven low-latency transport protocol using network priorities," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 221–235.

[51] M. Handley *et al.*, "Re-architecting datacenter networks and stacks for low latency and high performance," in *Proceedings of the conference of the ACM special interest group on data communication*, 2017, pp. 29–42.

[52] G. Zeng *et al.*, "FlashPass: Proactive congestion control for shallow-buffered WAN," in *2021 IEEE 29th International Conference on Network Protocols (ICNP)*. IEEE, 2021, pp. 1–12.

[53] S. Zou, J. Huang, J. Liu, T. Zhang, N. Jiang, and J. Wang, "GTCP: Hybrid congestion control for cross-datacenter networks," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 932–942.

[54] Y. Gao *et al.*, "DCQCN+: Taming large-scale incast congestion in RDMA over Ethernet networks," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. IEEE, 2018, pp. 110–120.

[55] Z. Chen *et al.*, "Paraleon: Automatic and adaptive tuning for DCQCN parameters in RDMA networks," in *2024 IEEE 32nd International Conference on Network Protocols (ICNP)*. IEEE, 2024, pp. 1–12.

[56] S. Li, L. Li, W. Dou, Y. Zhang, C. Wang, X. Cui, and L. Li, "SDLoRe: A loss recovery algorithm based on segment detection in lossy RDMA networks," *Computer Networks*, vol. 258, p. 111019, 2025.

[57] T. Zuo, T. Sun, S. Zhu, W. Li, L. Lu, Z. Du, and Y. Zhang, "LoWAR: Enhancing RDMA over lossy WANs with transparent error correction," in *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*. IEEE, 2024, pp. 1–10.