Reflex: Enhancing Delay-Based RDMA Congestion Control for Cross-Datacenter Networks

Jiaxue Liu, Xingjian Zhang, Yiran Zhang and Shangguang Wang
Beijing University of Posts and Telecommunications

***Abstract*—Deploying RDMA over cloud wide-area networks (WANs) poses significant challenges for delay-based congestion control (e.g., TIMELY, Swift). Through experimental investigations, we find that most intra-DC flows suffer performance degradation due to mismatched control loops, while long inter-DC flows also experience excessive throttling. To address this, we propose Reflex, an enhanced framework compatible with existing delay-based CC schemes. Reflex comprises two modules: Near-Source Feedback (NSF), which generates pseudo-ACKs to decouple the control loop for inter-DC flows, and Near-Destination Throttling (NDT), which actively isolates and throttles congested inter-DC flows. NS-3 simulations demonstrate that Reflex reduces overall average normalized FCT by 32.9% for TIMELY and 49.2% for Swift. Specifically, for TIMELY, it reduces intra-DC and inter-DC FCTs by 30.3% and 52.8%, respectively. For Swift, these reductions are 54.0% and 20.8%.**

***Index Terms*—RDMA, Congestion Control, Cross-Datacenter**

I. Introduction

Remote Direct Memory Access (RDMA) has become the de facto standard for high-performance datacenter networks due to its ultra-low latency and zero-copy features [1], [2]. In recent years, driven by the training of large language models (LLMs) [3]–[6] and the demand for geographically distributed applications such as disaster recovery [7], the trend of deploying RDMA across datacenter networks has become increasingly prominent [8], [9].

However, deploying RDMA in cross-DC environments poses unique challenges for congestion control (CC). Typical delay-based CC algorithms like TIMELY [10] and Swift [11] rely on precise round-trip time (RTT) for congestion detection and control. Yet, due to geographical constraints, inter-DC flows exhibit millisecond-level RTT, whereas intra-DC flows maintain microsecond-level RTT. When both flow types coexist in cross-DC scenarios, this significant RTT disparity causes CC algorithms to become inefficient.

Although there is research on ECN-based congestion control for cross-datacenter networks [9], [12]–[14], few studies focus on delay-based congestion control. We conduct in-depth experiments to study the performance of typical delay-based congestion controls. Our key observations are: 1) Inter-DC flows outperform intra-DC flows in general: When congestion occurs, inter-DC flows typically complete transmission before the millisecond-level RTT feedback arrives, whereas intra-DC flows swiftly reduce speed or window size in response to microsecond-scale feedback. 2) Inter-DC long flows suffer severe tail FCT: The high WAN latency is misinterpreted by delay-based CCs as severe congestion, triggering aggressive rate/window collapse. Furthermore, the initial uncontrolled bursts of these flows frequently trigger Priority Flow Control (PFC), exacerbating tail latency. 3) Tuning parameters of delay-based CCs does not help mitigate the FCT degradation: Adjusting control thresholds fails to resolve these structural issues. Both issues stem from a fundamental control loop mismatch: intra-DC flows adapt at microsecond timescales, whereas inter-DC flows operate on millisecond timescales.

To overcome these problems, we propose **Reflex**, a lightweight and transparent enhancement framework for delay-based RDMA CCs. It primarily comprises two components:

1) Near-Source Feedback (NSF) module at the external switch (ESW) of the source datacenter. NSF selectively generates pseudo ACKs to feed the sender with near-source RTTs based on a flow state machine. This mechanism effectively decouples the control loop from the WAN and balances the overhead of generating pseudo ACKs.

2) Near-Destination Throttling (NDT) module at the ESW of the destination datacenter. NDT segregates inter-DC flows into a normal or a controlled queue based on a state table derived from near-destination delay. To effectively throttle congested flows, NDT implements adaptive weighted round robin scheduling, out-of-order packet prevention, and a flow suspension mechanism.

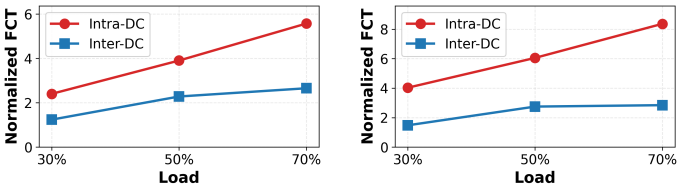
The main contributions of this paper are as follows:

- We have identified the fundamental issues with delay-based CC algorithms in cross-DC networks, pinpointing control loop mismatch as the root cause.
- We design Reflex, a framework that integrates NSF and NDT modules to enhance intra-DC performance and stabilize inter-DC flows, without modifying existing RDMA CC algorithms.
- We evaluate Reflex via NS-3 simulations [15]. The results show that Reflex improves the overall average FCT performance by 32.9% for TIMELY and 49.2% for Swift. Reflex also boosts intra-DC performance by 54% for Swift and lowers the tail latency of inter-DC large flows by 77.3% for TIMELY.

II. Background and Motivation

A. Cross-Datacenter RDMA Networks

The rapid expansion of geographically distributed applications has driven the demand for cross-datacenter services. Cloud service providers now distribute data and computation across multiple regional datacenters to enhance fault tolerance,



(a) TIMELY (b) Swift

Fig. 1: Normalized FCT of intra-DC and inter-DC flows under different loads

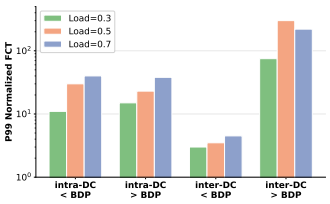


Fig. 2: P99 normalized FCT of TIMELY

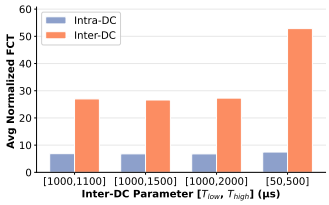


Fig. 3: Parameter tuning for TIMELY

reduce user access latency, and meet data sovereignty requirements. Given these applications demands for high throughput and low latency, RDMA has emerged as a compelling transport technology for cross-datacenter networking due to its highly efficient performance.

Cross-datacenter RDMA communication typically involves multiple datacenters interconnected via a wide area network. Datacenters connect to the WAN via external switches (ESW) positioned at the network edge. Cross-datacenter links comprise dedicated and public links, with dedicated links offering bandwidths up to 1.6 Tbps, while public links provide only hundreds of Mbps or several Gbps. Given that distances between datacenters often span hundreds or even thousands of kilometers, cross-datacenter communication RTT inevitably reaches millisecond levels.

B. Delay-Based RDMA Congestion Control

Although ECN-based RDMA congestion controls (e.g., DCQCN [1]) remain dominant, the industry—including hyperscale cloud providers like Google—is gradually shifting toward delay-based congestion control [10], [11]. These algorithms leverage RTT signals to overcome ECN's dependence on infrastructure, offering a switch-agnostic, easier-to-deploy solution across heterogeneous networks. Compared with ECN marks, RTT measurements provide continuous, fine-grained feedback. This enables senders to detect the onset of congestion early and adjust transmission rates.

TIMELY [10] and Swift [11] are two representative delay-based congestion control in the RDMA-enabled datacenters. TIMELY is a **rate-based** congestion control that directly regulates sending rates by inferring changes in queue length from *RTT gradients*. It adjusts the sending rate proportionally to RTT gradients when the RTT is between T_{low} and T_{high} thresholds. Swift is a **window-based** congestion control that adjusts the congestion window according to the *RTT values*. Besides, Swift introduces a decomposition mechanism that distinguishes network delay from host processing delay. By scaling $Target\_Delay$ based on hop_delay and $base\_delay$, it achieves fairer and more precise regulation. Both algorithms enable low-latency, high-bandwidth communication through only end-to-end RTT measurement.

C. Experimental Observations

Although there is research on ECN-based congestion control on cross-datacenter networks [12], [14], few of them study delay-based congestion control. To investigate the performance of typical delay-based congestion control in cross-datacenter networks, we conduct experiments examining the performance of TIMELY and Swift under varying workloads.

Experimental Setup: We set up a network topology comprising two datacenters connected by an inter-DC WAN link in NS-3 simulations. Each datacenter adopts a Leaf-Spine topology equipped with 16 servers and 9 switches, including one External Switch (ESW). The intra-DC links are 100 Gbps with $1\mu s$ propagation delay, and the inter-DC WAN link is 1.6 Tbps with $500\mu s$ propagation delay. Consequently, the base Round-Trip Time (RTT) is $8\mu s$ for intra-DC flows and $1012\mu s$ for inter-DC flows. For parameter settings, TIMELY uses the default parameter settings [10], while Swift uses a $base\_delay$ of $200ns$ and a hop_delay of $1\mu s$. We evaluate the Normalized Flow Completion Time (FCT)[1] of both algorithms under the typical WebSearch workload [16].

Observation 1: Inter-DC flows outperform intra-DC flows in general. As illustrated in Figure 1, under all load conditions, for both TIMELY and Swift, the average normalized FCT of inter-DC flows consistently outperforms that of intra-DC flows. This disparity stems from the fact that the vast majority (over 95%) of inter-DC flows are smaller than the Bandwidth-Delay Product (BDP) of the WAN ($\approx$ 12.5 MB). Consequently, these flows complete transmission before the first ACK arrives, and transmitting at line rate. In contrast, intra-DC flows receive feedback and detect congestion almost instantaneously. Consequently, they swiftly trigger rate or window reductions, thereby shouldering the primary responsibility for congestion control and unconditionally yielding bandwidth to inter-DC flows.

Observation 2: Inter-DC long flows suffer severe tail FCT. As illustrated in Fig 2, while short flows evade penalty mechanisms, inter-DC flows exceeding the BDP suffer severe tail latency degradation. On one hand, the high-latency feedback is perceived by the sender as severe congestion, triggering aggressive corrective mechanisms based on algorithmic formulas. For TIMELY, the inter-DC RTT (new_rtt) far exceeds T_{high}, forcing the rate update logic $rate \leftarrow rate \cdot (1 - \beta \cdot (1 - \frac{T_{high}}{new\_rtt}))$ to its extreme: as the term $\frac{T_{high}}{new\_rtt}$ approaches zero, the rate is slashed by the full multiplicative factor $(1-\beta)$. Similarly, Swift collapses the congestion window via $cwnd \leftarrow cwnd \cdot \max(1 - \beta \cdot \frac{delay - target}{delay}, 1 - max\_mdf)$; given that the inter-DC delay dwarfs the target ($delay \gg target$), the reduction ratio $\frac{delay - target}{delay}$ approaches 1, driving the window

[1]The Normalized FCT is the ratio of the actual FCT to the ideal FCT.

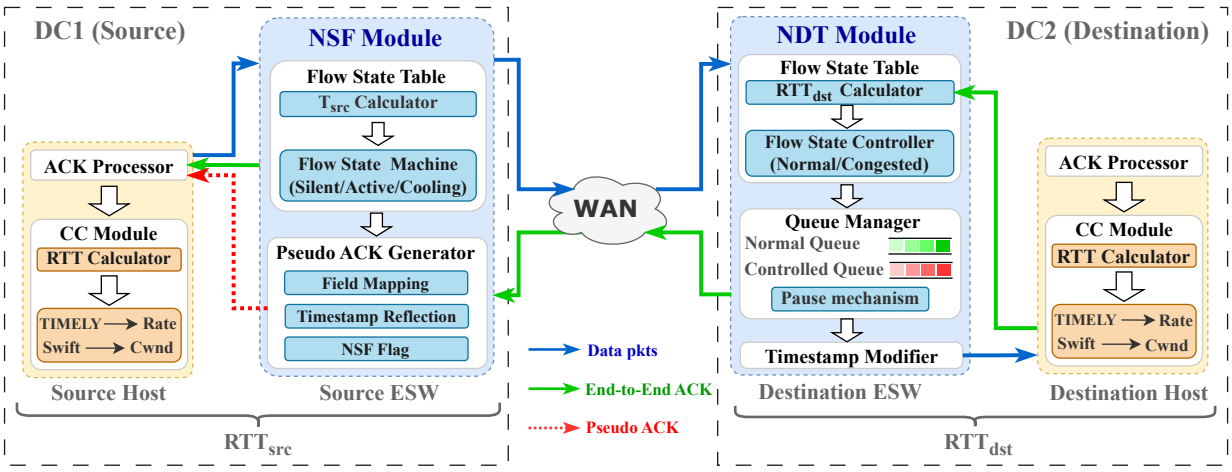


Fig. 4: Overview of Reflex

reduction to hit the maximum decrement limit (max_mdf). On the other hand, the initial uncontrolled bursts of these flows often trigger Priority Flow Control (PFC) before control loops can intervene. Empirical observations confirm that PFC consistently occurs on paths carrying long flows. For instance, at 70% load, a leaf switch traversed by three long inter-DC flows generated 566 PFC events. These two factors collectively cause a sharp deterioration in the tail FCT performance of long inter-DC flows.

Observation 3: Tuning parameters of delay-based CCs does not help mitigate the FCT degradation. We further investigated whether the performance degradation of inter-DC long flows could be mitigated through parameter tuning. For TIMELY, we calibrate the thresholds $[T_{low}, T_{high}]$ to accommodate the WAN RTT for inter-DC long flows. However, as shown in Figure 3, while this yields modest improvements, inter-DC long flows still lag behind intra-DC flows by approximately $4\times$. Moreover, expanding the threshold range from $[1000, 1100]\mu s$ to $[1000, 2000]\mu s$ provides negligible marginal gains. For Swift, we have also made corresponding adjustments to the relevant parameters, yet the results remain unsatisfactory. In conclusion, parameter tuning could not fundamentally resolve the issue.

Root Cause — Control loop mismatch. The root cause of the performance issues described above lies in the mismatch between the control loop frequency and the RTT of different flow types. Both TIMELY and Swift rely on ACK arrival as the "clock" to drive their control logic. Specifically, TIMELY updates the rate once per RTT, whereas Swift performs finer-grained window adjustments on each ACK. For intra-DC flows, this results in a high-frequency control loop operating every few microseconds, which ensures flow strictly adheres to congestion signals and remains highly sensitive to queue pressure. Conversely, the millisecond-level RTT of inter-DC flows creates a significant "control vacuum." This allows the majority of short flows to evade congestion control, while the few long flows suffer from delayed and coarse-grained corrections, leading to a highly unbalanced performance gap.

III. Design

A. Design Goal

1) Optimizing intra-DC FCT performance while balancing inter-DC flows. Our primary goal is to address the FCT performance imbalance between intra-DC and inter-DC flows caused by the control loop mismatch. Inter-DC flows tend to monopolize shared bandwidth, compromising the performance of latency-sensitive intra-DC flows. Therefore, we aim to design a mechanism that regulates inter-DC flows effectively to enhance intra-DC flows FCT performance (Observation 1). Furthermore, this regulation should not become a zero-sum game; it should minimize PFC triggering while safeguarding the performance of inter-DC long flows (Observation 2).

2) Transparent to existing delay-based RDMA CCs: To facilitate practical deployment in production environments, we aim to propose an enhancement framework on top of existing RDMA CCs rather than replace them. The enhancement framework should remain transparent to delay-based CCs, i.e., TIMELY and Swift. Thus, it requires no modifications to the core congestion control logic on RDMA NICs at end hosts.

B. Overview

Based on the design goals above, we propose **Reflex**, whose core design philosophy is to decouple the sluggish end-to-end control loop of inter-DC flows into responsive local segments. This design leverages the fundamental mechanisms of existing delay-based congestion control algorithms (i.e., TIMELY and Swift), which strictly adjust transmission rates or windows based on RTT signals carried by returning ACKs. While preserving the integrity of host logic, this framework transparently guides the sender's congestion response by strategically manipulating these input signals at the network edge.

Figure 4 presents an overview of Reflex, where ESWs are deployed at datacenter boundaries to serve as transparent control intermediaries. This architecture comprises two key modules: the Near-Source Feedback (NSF) module (Section III-C) at the source-side ESW, which generates proactive feedback—Pseudo ACKs—allowing the sender to perceive network conditions instantaneously and preempt near-source

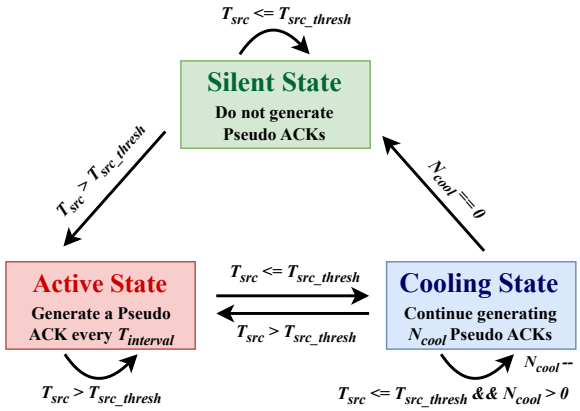


Fig. 5: Flow state machine in the NSF module.

congestion; and the Near-Destination Throttling (NDT) module (Section III-D) at the destination-side ESW, which isolates congested inter-DC flows to mitigate congestion within the destination DC.

C. Near-Source Feedback Module

The goal of the NSF module is to decouple the source-side control loop from the high-latency WAN control loop.

Overall workflow. The NSF module operates at the egress of the source ESW. For identified inter-DC flow, the module computes the near-source delay T_{src} by subtracting the packet's transmission timestamp from the current arrival time. This metric serves as the input to the Flow State Machine, which evaluates the current congestion state to determine whether to trigger the generation of pseudo ACKs.

Flow state machine. To balance responsiveness with minimal signaling overhead, the module employs an event-driven Flow State Machine. As shown in Figure 5, the frequency of feedback generation is determined by the relationship between measured delay T_{src} and the congestion threshold T_{src_thresh}.

• Silent State: When $T_{src} \leq T_{src\_thresh}$, the flow remains in the Silent State. The module generates no Pseudo ACKs, ensuring zero signaling overhead for non-congested flows.

• Active State: When $T_{src} > T_{src\_thresh}$, the inter-DC flow transits to the *Active State*, triggering the generation of a Pseudo ACK every $T_{interval}$. This interval not only mitigates reverse-path overhead but also synchronizes feedback frequency with the sender's control granularity. For instance, it supports TIMELY's per-RTT rate adjustments or satisfying Swift's per-ACK window updates. This mechanism effectively accommodates both rate- and window-based control loops.

• Cooling State: A key design consideration is how to transit from Active State to Silent State. Abruptly halting feedback immediately after congestion disappears prevents the sender from acquiring new RTT samples. Sender algorithms that rely on continuous gradient updates (e.g., TIMELY) may fail to compute the negative gradient required for acceleration, leaving the sender stuck at low speeds. To address this, when the corresponding inter-DC flow is in the Active State and detects $T_{src} \leq T_{src\_thresh}$, it transits to the *Cooling State* instead of directly transiting to Silent State. It initializes a counter N_{cool} and continues generating Pseudo ACKs. For each subsequent packet, if $T_{src} \leq T_{src\_thresh}$ and $N_{cool} > 0$, N_{cool} is decremented and the inter-DC flow stays in this state. Once N_{cool} reaches zero, it reverts to *Silent State*. This ensures the sender observes the declining RTT trend, correctly infers congestion resolution, and safely resumes rate increase.

Pseudo ACK generator. When triggered by the Flow State Machine, the generator constructs the feedback packet through three steps. First, it performs field mapping by extracting flow tuples from the forward data packet to synthesize a valid ACK header. Second, it implements *Timestamp Reflection* by directly echoing the current timestamp carried in the data packet into the ACK payload. Finally, it tags the packet with a unique NSF Flag, which distinguishes the feedback from the standard end-to-end ACKs for senders.

Rewrite RTT input for CCs at the sender. Upon receiving a pseudo ACK generated by the NSF, the sender decouples the congestion control loop from the wide-area network. Specifically, it computes the near-source segment RTT (RTT_{src}) using the timestamp carried by the pseudo ACK and adopts it as the input for TIMELY or Swift. Meanwhile, the standard end-to-end ACKs are processed for delivery confirmation, their arrival does not trigger any updates of RTT input, thereby insulating the control logic from high-latency WAN feedback.

D. Near-Destination Throttling Module

While the NSF module effectively manages source-side bottlenecks, it remains blind to congestion occurring within the remote destination DC, particularly where inter-DC and intra-DC flows converge. To address this limitation, we introduce the NDT module. NDT decouples the destination-side control loop from the high-latency WAN by monitoring local segment delays. Upon detecting congestion within the remote destination DC, it actively isolates aggressive inter-DC flows, thereby preventing queue buildup and PFC storms while safeguarding the performance of local intra-DC flows.

Overall workflow. The NDT module maintains two distinct queues at the egress of the destination ESW: a Normal Queue for forwarding non-congested packets with low latency, and a Controlled Queue for throttling congested flows. The processing workflow proceeds as follows: upon inter-DC flow arrival, the NDT module queries the Flow State Table to determine its current congestion status at the destination DC. Based on this status, the packet is dispatched into the appropriate queue.

Flow state controller. To determine whether the inter-DC flow experiences within the destination DC, NDT overwrites the timestamp of outbound inter-DC packets with their egress time as they depart the destination ESW. Consequently, when the corresponding ACKs return to the ESW, the intra-DC delay can be accurately measured. Since the NSF module is active, the sender no longer relies on the original end-to-end timestamp for rate control; thus, we can repurpose this header field to carry local telemetry without loss of critical information. Then the congestion status of each flow can be evaluated by comparing the calculated RTT_{dst} against T_{dst_thresh}. Specifically, if RTT_{dst} exceeds this threshold,

the flow is marked as *Congested* in the Flow State Table, and subsequent packets are allocated to the Controlled Queue. Conversely, if RTT_{dst} falls below the threshold, the flow status is updated to *Normal*, and packets are dispatched to the Normal Queue. As a result, this status is refreshed in real-time upon every ACK arrival.

Queue manager. Upon packet arrival, the Queue Manager queries the Flow State Table to direct the packet into either the Normal Queue or the Controlled Queue. To realize the throttling of congested inter-DC flows, NDT leverages the following mechanisms:

1) Adaptive weighted round robin scheduling. To throttle congested flows, we employ a WRR scheduler with a weight ratio of $N_{throttle}$: 1 between the Normal and Controlled Queues. When the Normal Queue is occupied, the scheduler prioritizes it, serving $N_{throttle}$ packets from the Normal Queue for every single packet from the Controlled Queue. Conversely, when the Normal Queue is empty, the Controlled Queue is automatically permitted to utilize the full available bandwidth to maximize link efficiency.

2) Prevention of OoO packets. A critical challenge arises when a flow transitions from *Congested* back to *Normal* state. Immediate redirection to the Normal Queue could cause out-of-order delivery if legacy packets remain in the Controlled Queue. To preserve sequence integrity, new packets from a recovered flow enter the Normal Queue only if the Controlled Queue contains no preceding packets from that flow; otherwise, they continue to be enqueued in the Controlled Queue until it drains.

3) Pause mechanism. We identify an extreme congestion scenario in which the ratio of flows mapped to the Controlled Queue to total active flows exceeds a critical threshold α. In such cases, we introduce a protective pause mechanism where the Controlled Queue suspends transmission entirely. This suspension persists until either congestion resolution is detected via returning ACKs or a maximum pause timeout $T_{maxpause}$ is reached.

IV. EVALUATION

A. Experimental Setup

We evaluate Reflex in NS-3. We adopt the topology detailed in Section II-C, employing ECMP for load balancing. As our design operates as an enhancement to delay-based RDMA congestion control, we utilize TIMELY and Swift as the underlying algorithms and evaluate performance by comparing scenarios with Reflex enabled versus disabled. We primarily evaluate performance based on normalized FCT and PFC counts. Key parameter settings are listed in Table I.

TABLE I: Parameter settings of Reflex

Parameter	Value
$T_{\text{src\_thresh}}$	5 μs
$T_{\text{dst\_thresh}}$	10 μs
T_{interval}	5 μs (TIMELY) / 3 μs (Swift)
N_{cool}	5
α	0.7
N_{throttle}	8
T_{maxpause}	500 μs

B. Overall Performance

We compared Reflex against TIMELY and Swift, two representative delay-based congestion control algorithms, evaluating their overall performance under varying network loads. The results are presented in Figure 6 and Figure 7.

Average normalized FCT performance. As shown in Figure 6, across varying loads and flow types, performance improves significantly with Reflex. For instance, at 70% load, Reflex reduces TIMELY's overall average normalized FCT by 32.9%; specifically, intra-DC and inter-DC flows improve by 30.3% and 52.8%, respectively. The gains are even more pronounced with Swift: at 70% load, Reflex reduces the overall average normalized FCT by 49.2%, with intra-DC improving by 54.0% and inter-DC by 20.8%. These results confirm that Reflex effectively mitigates intra-DC congestion while safeguarding inter-DC throughput. However, a minor exception is observed for TIMELY at 30% load. Since fewer inter-DC flows encounter congestion at this low load, enabling Reflex restricts the sending rate to a more conservative level, resulting in a slight performance degradation.

P99 normalized FCT performance. Figure 6 also displays the P99 normalized FCT. A key observation is the effective protection of intra-DC flows. At 70% load, Reflex reduces P99 normalized FCT of intra-DC flows by 42.9% for TIMELY and 69.3% for Swift. While inter-DC P99 latency slightly degrades in specific scenarios, significant gains are achieved in most cases, particularly under high load. These results demonstrate Reflex's robustness in handling latency-sensitive flows.

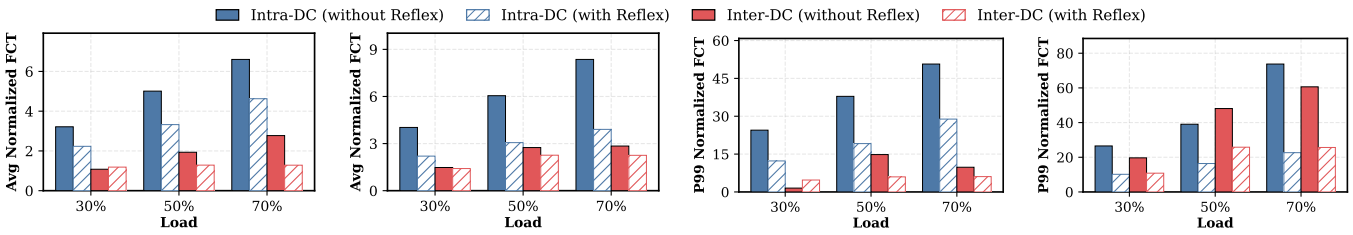
Performance of inter-DC large flows. Figure 7 highlights the performance of inter-DC large flows, which often suffer severe degradation due to slow sender reaction and PFC interference. By decoupling control loops, Reflex achieves substantial improvements. Notably, for TIMELY at 70% load, the P99 normalized FCT drops remarkably by 77.3%. We also observed a significant reduction in PFC generation. In TIMELY, PFC counts dropped from hundreds to zero at 30% and 50% loads, and from 2095 to 152 at 70% load. Swift exhibits similar trends.

C. Ablation Study

To evaluate the performance of the NSF and NDT modules individually, we conduct ablation experiments under 70% load. The results are shown in Figure 8.

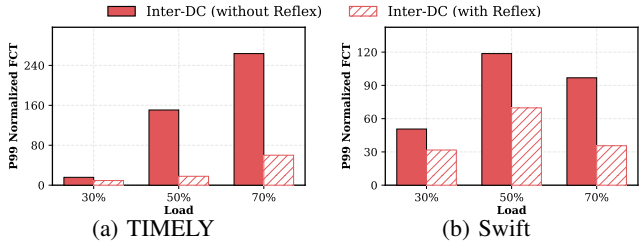
NSF module is the key to enhancing intra-DC flow performance. The NSF module promptly regulates inter-DC transmission rates via feedback pseudo ACKs, thereby securing more bandwidth for intra-DC flows. For instance, under Swift, it improved the average normalized FCT for intra-DC flow by 52.6%.

NDT module further enhances overall performance. The NDT module isolates congested inter-DC flows, thereby indirectly safeguarding intra-DC flow. It prevents performance degradation by mitigating the head-of-line blocking of long



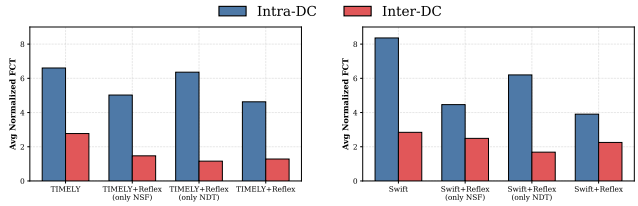
(a) TIMELY - Avg (b) Swift - Avg (c) TIMELY - P99 (d) Swift - P99

Fig. 6: Performance comparison under different loads.



(a) TIMELY (b) Swift

Fig. 7: Performance of large inter-DC flows (over 10MB).



(a) TIMELY - 70% load (b) Swift - 70% load

Fig. 8: Ablation experiments.

flows that would otherwise trigger excessive PFC. For example, in TIMELY, NDT effectively isolated 130 congested flows, accounting for approximately 12.9% of inter-DC flows. This isolation significantly mitigated buffer contention, reducing triggered PFC frames by 21.3% , thereby indirectly safeguarding intra-DC flows.

V. Conclusion

This paper exposes the limitations of delay-based RDMA CCs in cross-datacenter networks: most intra-DC flows suffer performance degradation due to mismatched control loops, while long inter-DC flows experience excessive throttling. To address this, we propose Reflex, a lightweight enhancement framework that enables hosts to react rapidly to congestion by deploying NSF and NDT modules on ESWs. Results demonstrate significant improvements in the average normalized FCT for TIMELY and Swift, offering a robust approach for deploying wide-area RDMA.

Acknowledgment

This work is supported in part by the National Key Research and Development Program of China (No. 2024YFB2907000), and by National Science Foundation of China (NSFC) under Grant No. 62302055, and in part by the Fundamental Research Funds for the Central Universities, and in part by the Shandong Provincial Natural Science Foundation under Grant No.ZR2024LZH006. Corresponding author: Yiran Zhang.

References

[1] 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*, vol. 45, no. 4, pp. 523–536, 2015.

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

[3] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-lm: Training multi-billion parameter language models using model parallelism," *arXiv preprint arXiv:1909.08053*, 2019.

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[5] J. Wang, X. Liao, X. Liu, J. Suo, Z. Huo, C. Zhang, X. Xu, R. Shen, X. Xie, and L. Xiao, "Deepcee: Efficient cross-region model distributed training system under heterogeneous gpus and networks," 2025. [Online]. Available: https://arxiv.org/abs/2505.15536

[6] A. Gangidi, R. Miao, S. Zheng, S. J. Bondu, G. Goes, H. Morsy, R. Puri, M. Riftadi, A. J. Shetty, J. Yang *et al.*, "Rdma over ethernet for distributed training at meta scale," in *ACM SIGCOMM*, 2024, pp. 57–70.

[7] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM*, vol. 43, no. 4, pp. 3–14, 2013.

[8] W. Bai, S. S. Abdeen, A. Agrawal, K. K. Attre, P. Bahl, A. Bhagat, G. Bhaskara, T. Brokhman, L. Cao, A. Cheema *et al.*, "Empowering azure storage with {RDMA}," in *NSDI*, 2023, pp. 49–67.

[9] 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.

[10] R. Mittal, V. T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats, "Timely: Rtt-based congestion control for the datacenter," *ACM SIGCOMM*, vol. 45, no. 4, pp. 537–550, 2015.

[11] G. Kumar, N. Dukkipati, K. Jang, H. M. Wassel, X. Wu, B. Montazeri, Y. Wang, K. Springborn, C. Alfeld, M. Ryan *et al.*, "Swift: Delay is simple and effective for congestion control in the datacenter," in *ACM SIGCOMM*, 2020, pp. 514–528.

[12] A. Saeed, V. Gupta, P. Goyal, M. Sharif, R. Pan, M. Ammar, E. Zegura, K. Jang, M. Alizadeh, A. Kabbani *et al.*, "Annulus: A dual congestion control loop for datacenter and wan traffic aggregates," in *ACM SIGCOMM*, 2020, pp. 735–749.

[13] 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.

[14] Z. Wan, J. Zhang, M. Yu, J. Liu, J. Yao, X. Zhao, and T. Huang, "Bicc: Bilateral congestion control in cross-datacenter rdma networks," in *IEEE INFOCOM*. IEEE, 2024, pp. 1381–1390.

[15] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and tools for network simulation*. Springer, 2010, pp. 15–34.

[16] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh *et al.*, "Hpcc: High precision congestion control," in *ACM SIGCOMM*, 2019, pp. 44–58.