LoRDMA: A New Low-Rate DoS Attack in RDMA Networks

Shicheng Wang*, Menghao Zhang$^{\dagger\diamond}$, Yuying Du$^{\circ}$, Ziteng Chen‡, Zhiliang Wang$^{\star\bullet}$, Mingwei Xu$^{\star\bullet}$, Renjie Xie*, Jiahai Yang$^{\star\bullet}$

*Institute for Network Sciences and Cyberspace, BNRist, Tsinghua University
†School of Software, Beihang University $^{\diamond}$Infrawaves $^{\bullet}$Zhongguancun Laboratory
$^{\circ}$Information Engineering University ‡Southeast University

***Abstract*—RDMA is being widely used from private data center applications to multi-tenant clouds, which makes RDMA security gain tremendous attention. However, existing RDMA security studies mainly focus on the security of RDMA systems, and the security of the coupled traffic control mechanisms (represented by PFC and DCQCN) in RDMA networks is largely overlooked. In this paper, through extensive experiments and analysis, we demonstrate that concurrent short-duration bursts can cause drastic performance loss on flows across multiple hops via the interaction between PFC and DCQCN. And we also summarize the vulnerabilities between the performance loss and the burst peak rate, as well as the duration. Based on these vulnerabilities, we propose the LoRDMA attack, a low-rate DoS attack against RDMA traffic control mechanisms. By monitoring RTT as the feedback signal, LoRDMA can adaptively 1) coordinate the bots to different target switch ports to cover more victim flows efficiently; 2) schedule the burst parameters to cause significant performance loss efficiently. We conduct and evaluate the LoRDMA attack at both ns-3 simulations and a cloud RDMA cluster. The results show that compared to existing attacks, the LoRDMA attack achieves higher victim flow coverage and performance loss with much lower attack traffic and detectability. And the communication performance of typical distributed machine learning training applications (`NCCL Tests`) in the cloud RDMA cluster can be degraded from 18.23% to 56.12% under the LoRDMA attack.**

I. Introduction

The advent of RoCEv2 (RDMA over Converged Ethernet Version 2) [34], [26] has led to a significant increase in the adoption of RDMA (Remote Direct Memory Access) in data centers. In RDMA, network clients can access the memory of network servers directly by using the "remote memory" abstraction. With RDMA-enabled network interface cards (RNICs), the remote memory access bypasses the CPU and the network stack in operating systems (OS) without any extra data copy, achieving a significant performance improvement. This performance benefit has been brought to a number of data center applications, especially distributed machine learning training tasks and distributed storage clusters [67], [16], [39], [101], [41], [10], [40], [60], [23], [8], [100], [56], [103], [11], [105], [37], [21], [7], [5]. And currently it is becoming a trend to expand RDMA from private high-performance computing clusters to public multi-tenant clouds [3], [4], [91].

Network and Distributed System Security (NDSS) Symposium 2024
26 February - 1 March 2024, San Diego, CA, USA
ISBN 1-891562-93-2
https://dx.doi.org/10.14722/ndss.2024.24025
www.ndss-symposium.org

However, the deployment of RDMA from private data centers to public multi-tenant clouds has put RDMA security under greater scrutiny. The OS-bypassing and protocol-offloading paradigm of RDMA offers substantial performance advantages, but it typically comes at the cost of security. On the one hand, OS bypassing disables the traditional softwarized security mechanism in operating systems and exposes server memory to remote clients without CPU mediation [83]. On the other hand, protocol offloading prefers a simplified transport protocol design due to the limited computation and memory resources in the RNIC, and has to rely on the hop-by-hop flow control (e.g., PFC (Priority-based Flow Control)) to eliminate packet loss and guarantee high performance. Therefore, RDMA faces a full range of security challenges, from confidentiality, integrity, to availability. While there are many recent studies [80], [96], [94], [90], [83], [38] that focus on the security of RDMA systems, the security of the underlying traffic control mechanisms, i.e., the hop-by-hop flow control (e.g., PFC [33]) and the end-to-end congestion control (e.g., DCQCN (Datacenter QCN) [116]) in RDMA networks, is largely overlooked.

In this paper, we investigate the security of traffic control mechanisms in RDMA networks with extensive experiments and theoretical analysis. Conventional wisdom usually uses the end-to-end congestion control (e.g., DCQCN) to alleviate problems of the hop-by-hop flow control (e.g., PFC) [32], [116], [68], [57], but we surprisingly find that the behaviors of the congestion control can also be misled by the flow control with well-crafted bursts, which derives significant security concerns. Specifically, by crafting concurrent bursts to overwhelm the egress queue of a switch port, the attackers can cause the backpressure of PFC PAUSE frames regardless of DCQCN deployment, resulting in congestion spreading to the upstream switches. Worse yet, the congestion detection of DCQCN is misled by the queues congested by back-spreading PFC frames, and therefore cuts the rate of innocent flows, which even share no link with the bursts. Moreover, due to the Additive-Increase/Multiplicative-Decrease (AIMD) property of DCQCN, the victim flow rate recovers very slowly, causing long-term average performance degradation. This phenomenon presents an unprecedented opportunity for attackers to conduct an *efficient* low-rate DoS attack: the attackers can widely cause high performance loss on flows across multiple hops, with low attack traffic volume and detectability.

In light of these observations, we present the LoRDMA attack, a new low-rate DoS (Denial-of-Service) attack in RDMA networks. Nevertheless, exploiting these vulnerabilities and conducting an efficient attack still face two substantial challenges. First, the attackers should coordinate the compromised devices (bots) to congest certain egress ports, in order to 1) cover more victim flows and 2) put higher performance degradation on each victim flow. However, selecting target ports to cover more flows is a generalized coverage problem, which is a classical NP-hard problem [15], [44]. Besides the coverage problem, deciding how many bots for each selected port to achieve a trade-off between the attack burst rate and the performance loss is also difficult. Although experiments have shown the rough relationship between them, the precise relationship is difficult for end-host attackers to acquire by experiments or offline simulations/mathematical analysis, due to the limited knowledge of flow rate and network configurations, and the complexity of the coupled traffic control mechanisms (PFC& DCQCN). Therefore, an efficient bot assignment is nontrivial to obtain. The second challenge is how to schedule the burst duration to cause high performance loss efficiently. Existing low-rate DoS attacks [62], [24], [25], [86] usually develop a theoretical model of the target service control system. They either construct a mathematical function between the attack impact and burst parameters, or identify some feedback signals to calculate the attack impact numerically, which helps make an efficient trade-off between the attack impact and the cost (e.g., average traffic volume). However, due to the complexity of the coupled traffic control systems in RDMA networks, it is also difficult to model this relationship mathematically.

Fortunately, we observe that RTT (Round-Trip Time), which has been proven to present a linear mapping with the queue length [68], [48], can well reflect the convergence of the burst impact. By monitoring RTT, the attackers can qualitatively infer whether the attack impact converges under the current attack pattern and adaptively adjust the burst parameters, though quantitatively evaluating the exact impact of a specific attack pattern is still difficult. Accordingly, we propose a two-step attack method consisting of a coordination and a schedule procedure. Guided by the RTT probing, the attackers first select the target ports and coordinate the bots to each target port to cover more target victim flows with rate degradation efficiently. Then the attackers schedule the burst duration to cause high performance loss on covered victim flows efficiently. We implement the attack tools including a line-rate burst generator and an RTT prober, and carry out evaluations on both ns-3 simulations and a real-world cloud RDMA cluster. All the codes are publicly available at GitHub [97]. Our evaluations demonstrate that the LoRDMA attack provides $\sim$10% higher average vicitm flow coverage and performance degradation in different topoligies, and only presents $\sim$50% of the direct flow contention, compared with existing attacks (e.g., link flood attacks [44]). And the LoRDMA attack also has several times higher attack efficiency compared to dumb burst parameter settings. Our simulations on real-world RDMA applications demonstrate that very few bots ($\sim$2% in the network) can cause performance loss on nearly 100% flows and a mean performance degradation ratio of 8.11% to 52.7% in different workloads and background traffic conditions. Experiments on the real-world cloud RDMA cluster also show that the communication performance of typical distributed machine learning training applications (`NCCL Tests`) is also significantly degraded by 18.23% to 56.12%. To the best of our knowledge, we are the first to identify PFC-misleading-congestion-control security vulnerabilities that may impact traffic hops away and propose a new attack correspondingly, which reminds researchers/engineers/operators to carefully revisit bursts when facing unideal network performance, and reconsider the security of traffic control in lossless RDMA networks.

Our contributions in this paper include:

- We conduct comprehensive experiments and provide an in-depth analysis of the performance loss by short-duration bursts in RDMA networks. We then reveal the vulnerabilities of the traffic control mechanisms in RDMA networks (§ III).
- We present the LoRDMA attack, a new low-rate DoS attack in RDMA networks. Taking RTT as the feedback, the attackers can efficiently tune the burst rate to the carefully selected target egress ports and the burst duration to achieve high attack efficiency (§ IV).
- We implement the attack tools including a line-rate burst generator and an RTT prober (§ IV-F), and conduct extensive evaluations to demonstrate the effectiveness and efficiency of the LoRDMA attack (§ V).

II. Background and Related work

A. RDMA Basics

RDMA was designed to be a high-throughput, low-latency, and low-CPU-occupation remote memory access technology. It eliminates the multiple data copies between application, kernel, and NICs, and allows direct memory-to-memory communication in user space, without CPU intervention or OS context switching between kernel and user space. It also offloads the transport protocol stack into dedicated hardware (i.e., RNIC), such as packet encapsulation, segmentation, reassembly, and traffic control mechanisms. Therefore, RDMA applications can directly access the memory of remote hosts with RNICs, achieving much lower latency and higher throughput without CPU mediation. Currently, RDMA has been widely used in a number of data center applications, especially distributed machine learning training tasks and distributed storage clusters [67], [16], [39], [101], [41], [10], [40], [60], [23], [8], [100], [56], [103], [11], [105], [37], [21], [7], [5].

RDMA applications can invoke the RDMA verbs API [35], [78] to directly operate RNIC hardware. In particular, the programmer can create a queue pair (QP) to set up an RDMA transport channel to a remote host. It can set different QP types, including Reliable Connection (RC), Unreliable Connection (UC), or Unreliable Datagram (UD). RDMA consists of two types of data transmission verbs. Two-sided verbs (SEND and RECV) are similar to traditional RPC messages and require CPU involvement. For example, the receiver CPU needs to issue RECV to its RNIC to declare readiness to receive data. And one-sided verbs (WRITE, READ, and ATOMIC) bypass CPU involvement completely. During the transmission, clients can directly access remote memory without remote CPU awareness and thus achieve ultra-high performance.

Although the OS-bypassing and protocol-offloading paradigm of RDMA enables substantial performance adavantages, it also imposes new constraints on the network traffic control mechanism. Mainstream implementations of RDMA usually require a lossless network to ensure high performance, because the loss handling on RNICs is expensive due to the limited hardware resources. In particular, the most widely used RDMA protocol, RoCEv2 [34], which encapsulates InfiniBand [35] over UDP, deploys PFC [33] to avoid packet loss. However, PFC brings several considerable performance problems, and congestion control (e.g., DCQCN [116]) is introduced subsequently to alleviate these performance problems. We give a detailed discussion on RDMA traffic control in § II-B.

B. Traffic Control Mechanisms in RDMA

RDMA requires a lossless network to achieve high transport performance in general. Specifically, in RoCEv2, the hop-by-hop flow control mechanism, PFC [33], is deployed to guarantee the loss-free property. When the ingress queue length exceeds a certain threshold (X-OFF) due to network congestion, the switch sends a "PAUSE" frame to its upstream entity (a switch or a RNIC) to stop packet transmission. And the transmission restarts with a "RESUME" frame when the queue drains below another threshold (X-ON). Despite zero packet loss, PFC results in many performance problems due to congestion spreading, such as head-of-line blocking (HLB), unfairness (victim flows), PFC storm, and even PFC deadlock [26], [31], [116], [89].

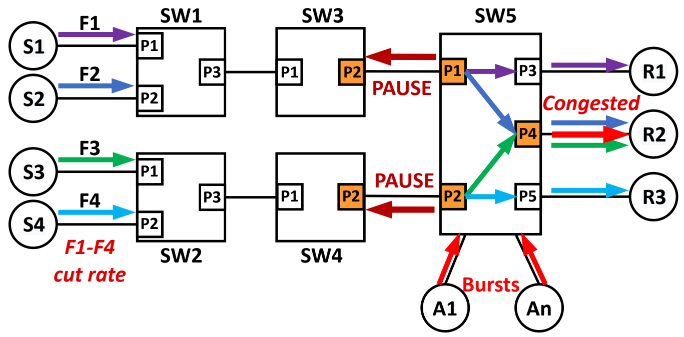
To alleviate the problems of PFC, conventional wisdom introduces the end-to-end congestion control. DCQCN, the default congestion control algorithm in RNICs from NVIDIA/Mellanox [66], is the most widely used one in leading industry companies [116], [26], [21], [5]. DCQCN detects congestion based on the egress queue length. The switch (Congestion Point) monitors egress queues and marks packets with ECN (Explicit Congestion Notification) [20] according to the RED (Random Early Detection) algorithm [19]. Note that in principle, ECN on switches should be set before PFC is triggered. The receiver (Notification Point) then responds to ECN-marked packets with congestion notification packets (CNPs). After receiving CNPs, to achieve fairness among different flows, the sender (Reaction Point) runs an AIMD rate adjustment mechanism, based on which the sending rate can be cut rapidly and recovered in a more moderate manner. Note that DCQCN has no slow start phase like TCP. When a flow starts, it sends at full line rate to accelerate the flow completion time.

There are many variants of congestion control algorithms in RoCEv2, e.g., QCN [32], TIMELY [68], HPCC [57]. However, QCN does not support layer-3 networks and limits the netowrk scale, TIMELY has been replaced with Swift by Google because of its convergence problems [48], and HPCC is still an academic paper without large-scale deployment [21]. To summarize, in current production RDMA networks, PFC and DCQCN are still the most widely used traffic control mechanisms. And thus in this paper we use PFC and DCQCN as the representatives of the hop-by-hop flow control and the end-to-end congestion control respectively.

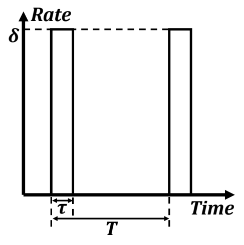
C. Related Work

RDMA security. The popularity of RDMA boosts the research on its security considerations [80], [96], [94], [90], [83], [38]. RFC 5042 [80] proposes a basic security analysis of RDMA system architectures and discusses various potential attacks and countermeasures. Pythia [96] proposes a side-channel attack on RNICs that allows attackers to learn the access pattern of victim clients by statistically analyzing the latency of well-crafted probe requests. sRDMA [94] highlights the vulnerability of weak authenticity and secrecy in RDMA systems, and also offloads symmetric cryptography onto SmartNICs to bridge the security gap. Simpson et al. [90] demonstrate the security issues in current RDMA systems, such as missing confidentiality, integrity, and authenticity, and also discuss guidelines on securing RDMA-based cloud storage systems. ReDMArk [83] provides a more systematic analysis of the security mechanisms in current RDMA architectures and RNIC implementations of various vendors. It also presents a comprehensive range of vulnerabilities with different adversary models, such as unauthorized access, resource exhaustion, and impersonation. Bedrock [38] offloads security functions, such as authenticity and access control, to programmable switches to avoid software overhead. Most of these works focus on the security of RDMA systems, and the research on transport layer security in RDMA networks remains lacking. Some existing studies [12], [109] identify the PFC-DCQCN interaction and improve the congestion control algorithm. However, they neither study the relationship between bursts and their impact on victim flow performance, nor how bursts can be exploited to conduct attacks. Worse yet, their solutions require substantial processing logic modifications on switches and RNICs, which have no support from current commercial production and face serious deployment obstacles in practical large-scale RDMA networks. Snyder et al. [92] identify that DCQCN can be exploited by attackers to gain an unfair advantage in bandwidth. However, they do not consider the impact of PFC, which is frequently triggered in realistic RDMA networks despite the deployment of DCQCN. Our work is the first study that reveals the relationship between bursts and victim flow performance via extensive experiments and theoretical analysis, and also presents a well-crafted new low-rate DoS attack to exploit the vulnerabilities efficiently.

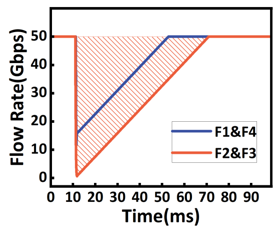
Low-rate DoS attacks. Low-rate DoS attacks are a classic type of DoS attacks [49], [62], [24], [25], [36], [86]. They focus on service provision systems, which can be modeled as a feedback-based control system. By crafting short-duration high-volume requests, they temporarily overwhelm the capacity of a service provision system and trigger "fake" feedback signals to degrade the service performance. TCP-based low-rate DoS attacks, such as Shrew [49] and Luo et al. [62], exploit the AIMD rate adjustment and RTO (Retransmission Time Out) mechanism, trigger packet loss by congesting switch port queues with periodic bursts, and make TCP decrease the performance of flows sharing the same link over a much longer term. The RoQ attacks [24], [25] analyze the Lyapunov stability [79] of the systems, such as TCP and load-balancers, and propose an optimization problem based on the relationship between the attack pattern and the impact. TCPwn [36] proposes an automated attack method by combining fuzzing and runtime analysis to generate specific attack methods for different implementations of TCP. The tail



(a) Illustrative topology.



(b) Burst parameters.



(c) Flow rate under bursts.

Fig. 1. LoRDMA vulnerability illustration.

attack [86] analyzes the attack impact on n-tier Web-App systems and constructs a feedback-based control framework to dynamically tune the attack parameters. As we can see, most existing low-rate DoS attacks require detailed theoretic analysis of the victim control system to guide an effective attack, such as queuing theory [86], [61], control theory [24], [25] or fuzzing [36]. However, RDMA networks consist of two coupled control systems: PFC and DCQCN, making mathematical analysis challenging. Our work is the first study to analyze the impact of low-rate DoS attacks on RDMA networks. Instead of deducing the mathematical relationship between attack pattern and impact, our work qualitatively reveals the convergence of the attack impact with burst parameters through extensive experiments and theoretical analysis, which inspires us to conduct the LoRDMA attack efficiently.

III. Motivating Measurement

In this section, we perform a systematic study on the impact of concurrent bursts on network performance. By adjusting different burst parameters, we investigate the variation of performance loss with each parameter. We then analyze the reasons for the performance degradation phenomenon, and summarize the vulnerabilities and principles for conducting an efficient low-rate DoS attack.

A. Experimental Setup

Like existing congestion control studies from top-tier conferences [116], [57], [12], [109], we develop our simulated testbed based on the open-source ns-3 DCQCN project [115]. We set up the topology as shown in Figure 1(a), a common unit from the CLOS network topology in data centers [1], [14], where 3 receivers under a ToR switch (SW5) are connected by 4 senders at other ToR switches (SW1&SW2), through different intermediate leaf switches (SW3&SW4). All links in the network have 100Gbps bandwidth with a propagation delay of 5μs. Specifically, S1-S4 start flows F1-F4 respectively to R1-R3. Each flow is allocated a fair share of the bandwidth bottleneck link, i.e., each flow converges to 50Gbps. Assuming some hosts are compromised by attackers as bots (A1-An), the attackers can manipulate the RNIC to craft line-rate bursts[1] to congest SW5.P4, and degrade the performance of legitimate flows. We set all parameters of PFC and DCQCN according to the default recommended values [33], [116], [115] (Table II in Appendix C).

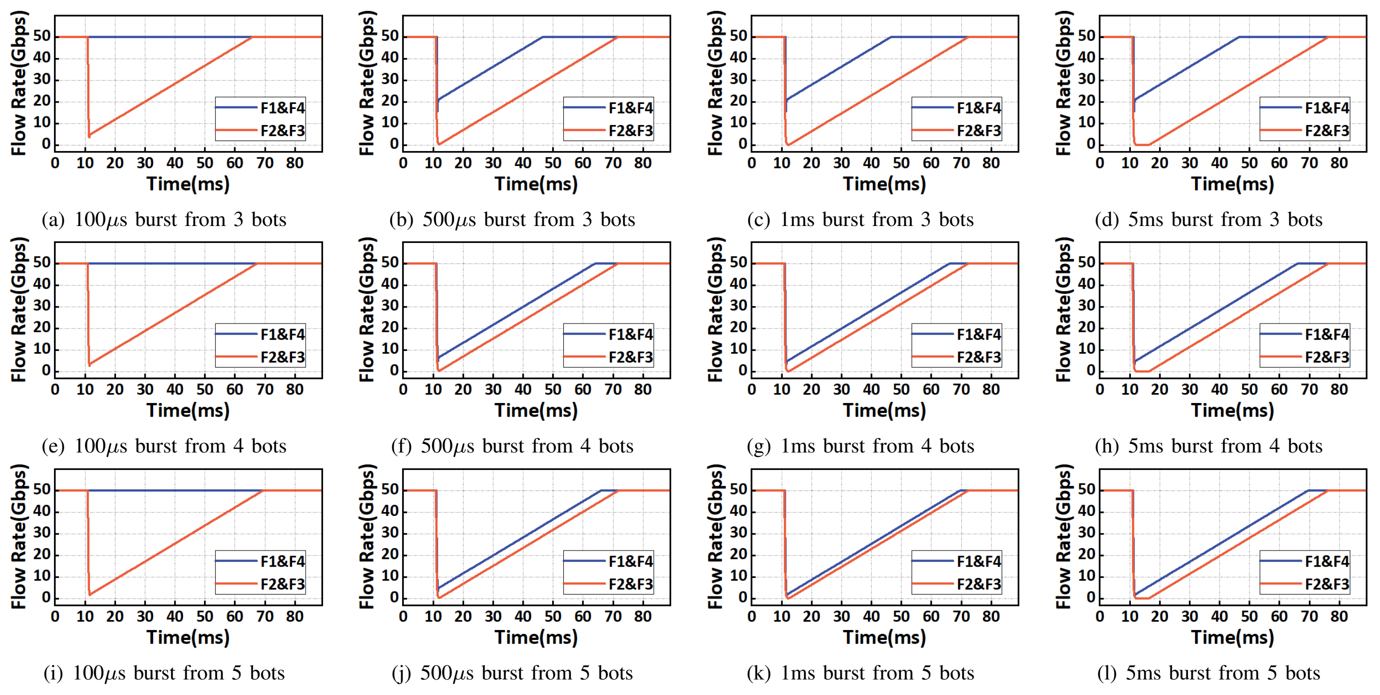
To study the impact of the bursts comprehensively, we first define the parameters to describe the bursts. As shown in Figure 1(b), we define the burst peak rate as δ, the burst duration as τ, and the period to restart the burst as T. In particular, we present δ as an integer multiple of the line rate, i.e., $\delta = n * 100Gbps, n \in \mathbb{N}_0$, where n stands for the number of bots sending the burst to the same destination egress port. To represent the impact on the performance of victim flows, we also define the *performance loss* of a flow caused by the bursts. As Figure 1(c) shows, for a flow f, the average bandwidth over time interval T can be computed as $\int_T R(t)\mathrm{d}t/T$, where $R(t)$ is the instantaneous rate of flow f at time t. The bandwidth loss can then be represented as the bandwidth difference: $\Delta Bw = R_0 - \int_T R(t)\mathrm{d}t/T = \int_T (R_0 - R(t))\mathrm{d}t/T$, where R_0 is the original rate of flow f. Note that the integral term as the numerator is exactly the area of the shadowed part in Figure 1(c) geometrically. Therefore, we define the performance loss of f as the shadowed area: $PL_f = \int_T (R_0 - R(t))\mathrm{d}t$. Based on these definitions, we conduct the simulations with various burst rate δ (i.e., the number of bots sending line-rate bursts to SW5.P4) and duration τ. We then monitor the flow rate over time, and calculate the performance loss for each flow correspondingly.

B. Experiment Results

Figure 2 shows the rate of different flows with different burst parameters. Specifically, we set the burst rate δ from 200Gbps to 800Gbps, and the burst duration τ from 100μs to 5ms. Due to space limitations, we only present a part of the results here. We also analyze the performance loss of victim flows with different parameters in Figure 3. Furthermore, we record the queue length and count PFC PAUSE frames under the 1ms burst in Figure 4.

Performance loss properties. As shown in Figure 2, the victim flow (F1-F4) rate decreases very fast in microseconds, and recovers nearly linearly because of the AIMD property in the congestion control. Actually, the recovery rate is determined by the parameters of DCQCN [116]. As we defined in § III-A, since the performance loss PL on a flow is exactly the shadowed part in Figure 1(c) geometrically, PL can be approximately regarded as the area of geometric shape whose height is the difference between the original flow rate R_0 and the lowest flow rate R_{lo} (denoted as $\Delta R = R_0 - R_{lo}$). Therefore, we refer to ΔR as the *performance loss factor*, which represents the performance loss on a flow roughly. We also define the *normalized performance loss factor* as $\text{Norm } \Delta R = \Delta R / R_0$, to characterize the severity of the performance loss experienced by different flows.

[1]In RDMA networks, line-rate burst can be generated in different ways, such as aggregating multiple mice flows (flow size $\leq$ BDP, Bandwidth-Dealy-Product) to avoid DCQCN rate cutting [12], [92], or even fully controlling the RNIC to craft traffic [83]. We give a more detailed discussion in § IV-A.



(a) 100μs burst from 3 bots (b) 500μs burst from 3 bots (c) 1ms burst from 3 bots (d) 5ms burst from 3 bots

(e) 100μs burst from 4 bots (f) 500μs burst from 4 bots (g) 1ms burst from 4 bots (h) 5ms burst from 4 bots

(i) 100μs burst from 5 bots (j) 500μs burst from 5 bots (k) 1ms burst from 5 bots (l) 5ms burst from 5 bots

Fig. 2. Flow rate with different burst duration τ and peak rate δ.

Different types of victim flows. Figure 2 shows two types of victim flows in Figure 1(a), which have different symptoms and mechanisms of performance degradation. (1) F2 and F3 are *directly* "cut off" by the bursts because of the egress queue congestion in SW5.P4. As Figure 4 shows, SW5.P4 has drastic queue buildup due to the bursts, and the packets of F2 and F3 get ECN marked, making DCQCN decrease their rate consequently. Worse yet, it takes a longer time for them to recover to the original rate because of the AIMD property in DCQCN, which is similar to existing TCP-targeted low rate DoS attacks [49], [62], [36]. (2) F1 and F4, surprisingly, also experience performance degradation. F1 and F4 have no link sharing, and thus no queue contention with the culprit bursts. But as shown in Figure 4, SW3.P2 and SW4.P2 also encounter significant queue buildup, resulting in the performance degradation on F1 and F4. In a word, they are *indirectly* cut by the bursts. We make further analysis of the performance loss on these two types of victim flows below.

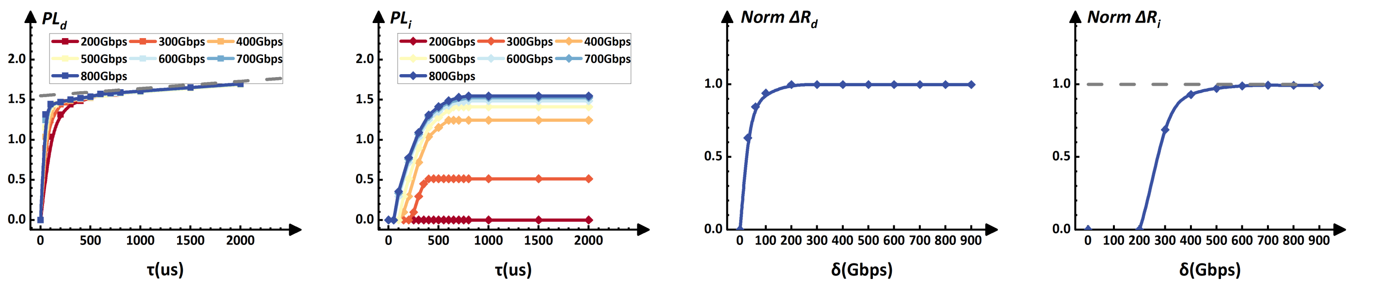
Performance loss on direct victim flows. As described above, direct victim flows (F2 and F3) are similar to those under traditional TCP low-rate DoS attacks. (1) Consider the performance loss factor of direct victim flows ΔR_d. On the one hand, given a bot number n (i.e., a given burst peak rate $\delta = n * 100Gbps$), ΔR_d gradually increases with τ and converges to R_{d0} (50Gbps) when τ is long enough, as shown in every row of Figure 2. On the other hand, given a τ long enough (e.g., 1ms), ΔR_d is relatively constant with increasing bot number (Figure 2(c), 2(g) and 2(k)), because very few bots (e.g. 2 bots) with line-rate bursts have already cut the rate of direct victim flows to nearly 0. We also illustrate the relationship between ΔR_d and δ in Figure 3(c) (shown by $\mathrm{Norm}\ \Delta R_d$). (2) Consider the performance loss on direct victim flows PL_d. PL_d is approximately an orthogonal triangle area when τ is short (e.g., Figure 2(a)). Its height is the performance loss factor (ΔR_d), and the slope of the hypotenuse k, is a constant determined by the DCQCN additive recovery parameters. Therefore, we have $PL_d = (\Delta R_d)^2/2k$, which increases super-linearly with τ because ΔR_d is also growing with τ^2. However, when τ is long enough and the flow rate decreases to 0 (ΔR_d converges), PL_d geometrically becomes a right trapezoid where its height is $\Delta R_d = R_{d0}$ (Figure 2(d), 2(h) and 2(l)). Accordingly, PL_d begins to increase linearly with τ, i.e., $PL_d = R_{d0} * \tau + (R_{d0})^2/2k$ and $\mathrm{d}PL_d/\mathrm{d}\tau = R_{d0}$. We also illustrate the variation of PL_d with τ in Figure 3(a).

Performance loss on indirect victim flows. (1) Consider the performance loss factor of indirect victim flows ΔR_i first. On the one hand, given a bot number n, i.e., $\delta = n * 100Gbps$, ΔR_i gradually increases with τ, and becomes constant when τ is long enough, as shown in every row of Figure 2. On the other hand, given a τ sufficiently long (e.g., 1ms), ΔR_i becomes higher and converges with δ (Figure 2(c), 2(g) and 2(k)), as also illustrated in Figure 3(d) (shown by $\mathrm{Norm}\ \Delta R_i$). (2) Then consider the performance loss on indirect victim flows PL_i. As Figure 3(b) shows, given a δ, PL_i increases with τ when τ is short. However, different from PL_d, when τ is long sufficiently, PL_i converges to the upper bound and stays constant. And the upper bound of PL_i increases with δ. Nevertheless, the growth rate of PL_i upper bound fast diminishes with δ increasing. For example, as shown in Figure 3(b), when $\delta \geq 500Gbps$, the upper bound of PL_i hardly grows, and therefore adding more bots sending bursts makes no more gain in PL_i.

C. Understanding the Performance Loss

In this subsection, we give an in-depth analysis of the performance degradation on these two types of victim flows.

[2]Actually, there exists a negligible error between real PL_d and $(\Delta R_d)^2/2k$ and in other equations. We use "=" in this paper for readability.



(a) Performance loss on direct victim flows PL_d with burst duration τ.
(b) Performance loss on indirect victim flows PL_i with burst duration τ.
(c) Normalized performance loss factor Norm ΔR_d on direct victim flows with burst peak rate δ ($\tau = 1ms$).
(d) Normalized performance loss factor Norm ΔR_i on indirect victim flows with burst peak rate δ ($\tau = 1ms$).

Fig. 3. Performance loss on different victim flows with burst duration and peak rate.

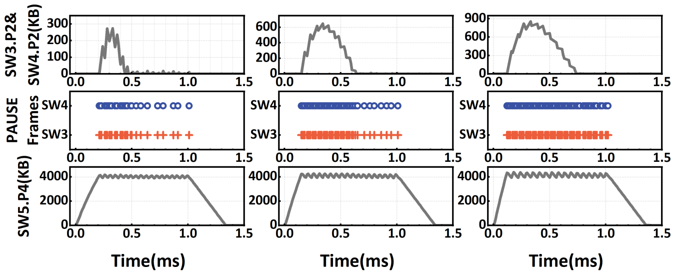


Fig. 4. Queue length and PFC PAUSE frame count during the burst (δ = 300Gbps, 400Gbps, 500Gbps, τ = 1ms).

The impact on direct victim flows is similar to those at TCP low-rate DoS [49], [62], [61], which is caused by direct queue contention between bursts and victim flows. As Figure 4 shows, the egress queue at SW.P4 is congested by the bursts, causing the packets of direct victim flows (F2 and F3) to get ECN marked and then get flow cut multiplicatively until 0 by DCQCN. As shown in Figure 3(c), the flow rate can easily be cut to 0 (Norm $\Delta R_d = 1$) with ~200Gbps (2 bots). And PL_d, which is dominated by ΔR_d, accordingly first grows super-linearly with τ before the flow rate decreases to nearly 0, and then grows linearly when the flow rate stays at nearly 0 (Figure 3(a)). Actually, when τ is long enough, direct victim flows will stay at nearly 0, which is equivalent to a link flood attack (LFA) [44], [93].

However, the performance degradation on indirect victim flows and the underlying causes are substantially different. The congestion points of F1 and F4, SW3.P2 and SW4.P2 (illustrated in Figure 4), are not passed by the culprit bursts. Actually, their queue buildup is caused by PFC PAUSE frames. When the bursts overwhelm SW5.P4, a large number of packets in F2 and F3 accumulate quickly in the ingress ports (SW5.P1 and SW5.P2). Their queue length rapidly exceeds the PFC X-OFF threshold, generates PFC PAUSE frames to the upstream switch ports (SW3.P2 and SW4.P2), and stops their packet transmission. Note that it takes a much longer time for the end-to-end DCQCN to reduce the rate of F2 and F3 compared to PFC. DCQCN reduces the rate by at most 50% in every update period, which is 10s of μs and comparable to the end-to-end latency, while the hop-by-hop PFC runs at μs level. Therefore, PFC dominates the control system transiently and spreads the congestion upstream. Consequently, the upstream egress queues quickly build up and results in an extra rate cut on F1 and F4 by DCQCN. In other words, the congestion spread by PFC misleads the congestion detection of DCQCN, causing the performance degradation on flows even without any direct queue contention with the attack bursts. As Figure 4 shows, higher δ results in higher PAUSE frame rate and thus more severe queue congestion at SW3.P2 and SW4.P2, finally leading to higher ΔR_i and PL_i. However, the congestion experienced by indirect victim flows is not persistent during bursts. Figure 3(b) shows that longer burst duration does not increase PL_i, and Figure 4 also proves that the queue length of SW3.P2 and SW4.P2 recovers even when the bursts exist and SW5.P4 remains congested. Actually, later in the burst duration, DCQCN finally dominates the traffic control after several update periods. It decreases the rate of direct victim flows F2 and F3 to 0, and mitigates the buildup at SW5.P1 and SW5.P2. Therefore, the PFC PAUSE frame rate significantly decreases, and the queue buildup at SW3.P2 and SW4.P2 also vanishes. Finally, F1 and F4 start to recover the rate regardless of bursts.

D. Vulnerabilities in RDMA Traffic Control

In this subsection, we summarize the vulnerabilities in RDMA traffic control for an efficient and stealthy attack.

First, by carefully congesting certain switch port queues (links) using bursts from multiple bots, the attackers can spread congestion broadly, and consequently mislead DCQCN to cut off victim flows *in an indirect way*. The attackers can directly congest fewer links by causing short-duration high link load using bursts (like traditional DoS attacks), and exploit PFC to indirectly shut down more links while keeping their load low. They can therefore cover more target flows with fewer high-load congested links and lower link correlation between bursts and victim flows, both of which are important footprints in security defense and performance anomaly detection. First, current DoS (especially link flood attack) defense studies [102], [111], [43] usually monitor the link load, especially the link with high flow density (more flows pass through) [110], [104], to detect the attack. They usually trigger mitigation mechanisms when the number of high-load congested links (queues) exceeds a threshold and identify the attack traffic by analysing (such as counting bytes) the traffic in the detected congested links. Second, current network monitoring systems [71], [112], [117], [95] also take queue contention between flows as one of the key signals in performance anomaly detection. Existing studies on performance anomaly diagnosis [99], [113], [54] usually collect the monitor information across networks (e.g., INT statistics), construct the queue contention relationship between flows, and find the major contributors to the queue contention (and hence the performance degradation) experienced by the victim flows. Therefore, as we can see from these two points, reducing the direct queue/link congestion between victim flows and attack traffic, can help reduce the detectability significantly.

Second, short-duration bursts can cause long-term flow degradation on victim flows with low detectability. A burst lasting for $<$1ms can significantly decrease the victim flow rate to nearly 0. However, due to the AIMD property of DCQCN rate adjustment, it takes 10s of milliseconds to recover to the original rate. Therefore, the average victim flow rate can be suppressed at a very low level with only $<$10Gbps average burst rate, by carefully tuning the duration τ and period T. For example, as our experiment shows, a 100Gbps line-rate flow can be reduced to 0 in 1ms but can only recover to the original rate in about 60ms. By setting τ as 1ms and T as 60ms, the average burst rate for each bot is only 1.6Gbps, while the average victim flow rate is 50Gbps, which is only 50% of the original rate. The average performance loss can also be further increased by adjusting the period, which is a trade-off between the impact and the cost. Due to the high performance of RNICs and line-rate start property of DCQCN, high volume traffic ($\sim$100Gbps) is common in RDMA networks. Besides, short-duration ($\sim$1ms) bursts with low average rate ($\sim$1Gbps) is difficult to capture for the prevalent second-level network monitoring tools currently. And the bursts are also difficult to be noticed at end-host, because these unexpected traffic is immediately dropped by RNICs without noticing upper-level applications at the destination hosts.

Principles to exploit the vulnerabilities. (1) The attackers should degrade more victim flows with higher impact and lower detectability, especially by exploiting the indirect victim flow effect. The bots should be carefully coordinated to attack different target egress queues, to cover more victim flows and put higher ΔR on them efficiently. For example, in Figure 3, 5 bots are sufficient to degrade both direct and indirect victim flows to about 0, and adding more bots is not cost-efficient. (2) The attackers should tune the burst duration τ and period T carefully to make a trade-off between the performance loss PL and the average burst rate, which is correlated with detectability and capital expenditure. For example, longer duration adds no damage gain on indirect victim flows and linear gain on direct victim flows, making diminishing returns on damage.

IV. The LoRDMA Attack

In this section, we present the LoRDMA attack to exploit the vulnerabilities in RDMA traffic control. First, we specify the threat model and define metrics to model attack impact and cost, aiding in guiding an efficient attack. We then identify the challenges to conduct such an attack, and finally propose our observation and attack methodology.

A. Threat Model

Attack scenarios. Our threat model is based upon the state-of-the-art studies on RDMA security that the adversary resides in RDMA networks [80], [96], [94], [90], [83], [38]. In particular, similar to Bedrock [38], we focus on an RDMA network where multiple users share the network infrastructure, which is common in both public cloud data centers and private RDMA clusters. We assume that the network infrastructure can be trusted, but there may be potential malicious hosts which seek to degrade the service performance of the entire network. Current RDMA clusters, even RDMA-capable cloud providers (e.g., Azure [4] and Alibaba [3]) have nearly no fabric-level isolation [26], [5], [21], and the network infrastructure is shared between hosts. Therefore, in current RDMA-enabled cloud instances (e.g., bare metal servers, VMs, and containers), network traffic can still impact each other.

Attacker capability. The attackers should be able to craft high-rate (even line-rate) burst traffic. This assumption can be achieved by RDMA hosts in multiple ways. First, as we mentioned in § II-B, RDMA networks allow line-rate start of each flow. And any network host can craft line-rate traffic by continuously generating line-rate mice flows shorter than 1 RTT (i.e., flow size $\leq$ BDP), which is uncontrollable by the end-to-end congestion control. This bandwidth exploitation method has been proposed and implemented by existing work [92], [12]. Second, mainstream RNICs have supported `IBV_QPT_RAW_PACKET`, a raw Ethernet programming feature enabling high-performance kernel-bypass traffic generation on the RNICs [83], [74]. Considering the attackers' control of RNICs and verbs API, they are hence fully capable of high-rate traffic generation without further compromising any network infrastructure. We have also validated the burst generating capability at Alibaba Cloud and a private cloud RDMA cluster at Kuaishou Technology (§ V-D).

Attacker prior knowledge. (1) We assume that the attackers know the network topology with the help of existing probing tools, such as traceroute [63] and ping [6], because RoCEv2 traffic is also a special kind of UDP traffic. However, these traditional probing tools can only help to collect network route information. It is difficult for them to probe network performance metrics (e.g., network latency [63] or bandwidth bottleneck [65], [30]), because RoCEv2 traffic and other TCP/UDP traffic are assigned in different queues with different schedule policies in general. (2) The attackers are also assumed to have prior knowledge of a specific set of network flows that are expected to be cut off (we refer to them as target flows). This can be specified in advance by the Attack-as-a-Service buyers [73], [85]. The attackers can also infer them from social engineering or probing techniques, which is orthogonal to this work. In fact, this assumption can be relaxed in practice, such as partial knowledge of flow routes or even only topology awareness, which is discussed in § VII.

Attack goal. The attackers aim to launch an *efficient* attack, i.e., cause *high impact* at *low cost*. The *impact* represents the coverage and performance loss on target flows. And we consider the *cost* to refer to not only capital expenditure, but also the risk of being detected. The attackers then carefully set the attack parameters δ, τ and T to achieve an efficient trade-off between the performance loss on target flows and the attack cost. Note that the relationship between T and the average performance loss is similar to traditional TCP-based low-rate DoS attacks, since DCQCN has similar AIMD rate adjustment mechanism. Therefore, our major focus in the rest of our paper is to investigate how to efficiently cause high and wide performance loss with δ and τ. We set T long enough (60 to 100ms based on our experience) to keep the average burst rate low and to make the network flows re-converge.

B. Challenges

Although we have shown the promising vulnerabilities for the LoRDMA attack, conducting an efficient attack still faces the following two major challenges.

The first challenge is how to coordinate the limited bots to cause high impact on more flows efficiently. First, congesting specific switch egress ports to cover most flow routes with limited bots is a generalized maximum coverage problem, which is a classical NP-hard problem. Traditional link flood attacks (LFA) propose target link selection algorithms based on heuristic or reconnaissance [44], [9], [93]. However, they do not exploit the indirect victim effect, and degrading every flow via directly contending the bandwidth would cause high burst rate and significant direct queue contention, i.e., high detectability [102], [111], [113], [99]. Furthermore, directly cutting all the flows also requires a huge number of bots, which requires high expense. Second, besides the coverage of victim flows, deciding the burst rate δ to each target port, in order to put high performance degradation on victim flows, is also difficult. An intuitive idea is to decide the burst rate δ on each target port according to the relationship about δ and ΔR caused on victim flows, as illustrated in Figure 3(c) and Figure 3(d). Unfortunately, obtaining the precise relationship between δ and ΔR, especially ΔR_i, requires comprehensive experiments on all topologies, which is unfeasible for the attackers. And the instantaneous flow rate of victim flows is also difficult for the end-host attackers to obtain. Another idea is offline simulating the network to derive this relationship. However, besides the network topology and flow distribution, offline simulating requires much more network configurations, such as DCQCN and PFC parameters, switch buffer and queue size, etc., which are also very difficult for the host-side attackers to acquire. Worse yet, mathematically deducing such a relationship is also impractical. Besides the lack of network configurations and conditions above, the coupled control systems in a large-scale topology make analyzing drastically complex. Especially, the congestion encountered by the indirect victim flows is caused by PFC in the downstream switches and DCQCN at the congestion point jointly, which is extremely difficult for the attackers to model and analyze.

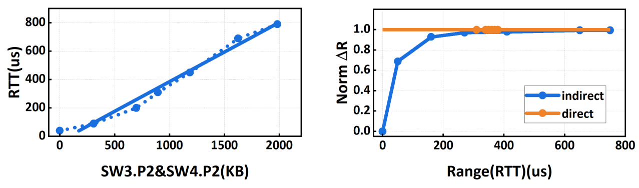
The second challenge is how to schedule the burst duration τ to cause high performance loss with low average burst rate, similar to what traditional low-rate DoS attacks do [49], [62], [61]. Existing low-rate DoS attacks have proposed fine-grained mechanisms to tune the attack burst parameters (burst peak rate, duration, and period). They model the adjustment behaviors of the target control system (such as TCP [49], [61], [36], switch queue management [24], or web application servers [86]), and theoretically infer the attack impact, such as fuzzing [36] or queuing theory [86]. They then abstract a mathematical model to estimate the attack impact (performance degradation) a specific set of burst parameters can cause [49], [24]; or find a feedback signal to compute the attack impact and guide the parameter tuning [86]. They can hence achieve an efficient attack with high impact and low costs. However, this method is difficult in RDMA networks. Although we obtain the rough relationship between τ and PL (Figure 3(a) and Figure 3(b)), it is difficult for the attackers to deduce this relationship precisely. The traffic control consists of two coupled systems (the hop-by-hop PFC and the end-to-end DCQCN), and the impact on victim flows may be caused by bursts hops away. For example, as we discussed in III-C, to calculate the PL_i an indirect flow f suffers, the attackers should analyse the queue buildup process at the target egress ports and deduce the back-spreading PFC PAUSE frame rate. Then they still have to calculate the egress queue length variation at upstream switches, and the corresponding ECN-marked packet rate to estimate when f stops decrease and recover its rate. The analysis above is far more complex, compared with the traditional attack impact analysis on a single queue with a single control system (e.g., TCP). Furthermore, as we mentioned in the previous paragraph, due to the limited information of attackers, experimental or simulation methods cannot be used to obtain precise relationships as well.

C. Key Observation & Attack Overview

Although it is challenging to construct an exact quantitative function of the attack impact, we observe that, RTT is a natural side-channel signal qualitatively reflecting whether the attack impact (ΔR and PL) converges with δ and τ, which can guide the attackers to set efficient burst parameters. Actually, RTT is a key signal reflecting queue length and congestion [68], [48], and has been proven to be linearly mapped to queue length [68], which is also validated by our experiment (Figure 5(a)). In particular, the queue length during the congestion decides how many packets are marked with ECN, i.e., how severely the victim flow rate will be cut. When the queue length resumes to zero, the congested flows will begin to recover their rate additively. As a result, RTT, which indicates the queue length, well reflects (1) the severity of congestion (i.e., ΔR) the victim flows are encountering, which is represented by the peak value of RTT; (2) when the congestion ends (the RTT resumes normal) and the victim flows begin to recover the rate, so that PL can be estimated.

However, RDMA RTT probing is not as trivial as in general networks. Traditional probing tools, such as ping or traceroute, fail to get the accurate RTT of RDMA communication due to the different QoS policies between TCP traffic and RoCEv2 traffic. Existing RDMA RTT measurement tools, such as rping [59], only support RTT probing between controlled hosts, resulting in a limited detection range. Luckily, we find a side channel to infer the RTT from bots to any host in the network. We notice that, when using RDMA CM (Connection Manager), a communication suite, to setup RDMA communications, a host replies with a `ConnectReject` packet upon receiving a `ConnectRequest` from an unknown host. Both packets are in RoCEv2 protocol and can reflect the RTT of the RDMA communication. Therefore, a bot can keep crafting connection requests to other hosts and monitor the long-term RTT of different target flows.

With the help of RTT probing, we can *adaptively* organize an efficient attack. We decouple the attack organization into two sub-problems: 1) coordinate the bots to congest the target switch ports, and 2) schedule the burst duration τ. In the coordination step, the attackers greedily select the "high-value" ports which can cut more flows, especially indirectly. Then they gradually add bots sending line-rate bursts to each target port, and monitor RTT_i and RTT_d, the RTT passing through



(a) Correlation between queue length and RTT. (b) Correlation between ΔR and range$\langle RTT\rangle$ of victim flows.

Fig. 5. RTT reflects queue buildup and performance loss by bursts (δ = 200Gbps∼800Gbps, τ = 1ms).

the congestion point of the indirect/direct victim flows. They stop adding the bot when ΔR_i, reflected by RTT_i, stops growing to achieve efficient δ on each target port. In the schedule step, similarly, they adaptively adjust τ based on the varying RTT and achieve efficient burst parameters. We give a more detailed description of our attack in § IV-D and § IV-E.

D. Coordination

In this subsection, we elaborate on the coordination procedure in the LoRDMA attack. The coordination procedure determines which egress port each bot should send burst traffic to congest, in order to bring high and wide performance loss on target flows. We first give a formal problem statement, then illustrate our insight on the relationship between RTT and ΔR, and finally give the detailed process.

Problem statement. The input of the coordination is the network topology, the target flow set F, the egress port set P of all switches, and the bot set B. The coordination outputs a mapping from bots to ports $\text{Coord}: B \rightarrow P \cup \{\phi\}$. For a bot $b \in B$, $\text{Coord}(b)$ describes whether or which switch egress port b should send traffic to congest[3]. To simplify the coordination, we specify that each bot sends all its line-rate burst to at most one destination, making an integer programming on burst rate δ (e.g., $\delta = n * 100Gbps, n \in \mathbb{N}_0$). To obtain higher ΔR on more flows, the attackers aim to 1) select the ports which can cover more flows directly and indirectly, and 2) carefully set the number of bots on each target port to cause sufficient ΔR and avoid deploying too many bots meanwhile. Note that since few bots (e.g., 2 bots) are sufficient to cut the direct victim flows of each port to about 0 (i.e., ΔR_d converges), we mainly focus on the rate degradation on indirect victim flows ΔR_i.

RTT range and ΔR. As we illustrate in § III, the rate degradation of indirect victim flows (F1 and F4) is caused by the queue buildup in their congested ports (SW3.P2 and SW4.P2). And higher queue length makes more severe rate cut ΔR_i (Figure 4). Therefore, the attackers can infer ΔR_i by monitoring the maximum queue length during the burst, which is exactly reflected by RTT_i. Since the queue length is linearly mapped with RTT (Figure 5(a)), we can obtain the maximum queue length by calculating the range of the RTT sequence $\langle RTT\rangle$ during the bursts, i.e., $\text{range}\langle RTT\rangle = \max\langle RTT\rangle - \min\langle RTT\rangle$. Note that we evaluated the max queue length using $\text{range}\langle RTT\rangle$ instead of $\max\langle RTT\rangle$, because the network topological complexity may introduce different $\min\langle RTT\rangle$, which is exactly the end-to-end propagation delay. However, for indirect victim flows, $\text{range}\langle RTT_i\rangle$ is not linear with ΔR_i. As Figure 5(b) shows, by increasing δ, ΔR_i converges after a specific $\text{range}\langle RTT_i\rangle$, although the latter continues to increase. And further adding more bots after ΔR_i converges is not efficient. To find the minimum bot number needed and the corresponding minimum $\text{range}\langle RTT_i\rangle$ where ΔR_i converges, we observe that, ΔR_d converges easily at lower burst rate δ (e.g., 200Gbps), and the corresponding $\text{range}\langle RTT_d\rangle$ is a good heuristic value implicating whether ΔR_i reaches peak and converges. For example, as shown in Figure 5(b), $\text{range}\langle RTT_i\rangle$ is about 270μs at a 500Gbps burst, and is approximately equal to $\text{range}\langle RTT_d\rangle$. In this case, both ΔR_i and ΔR_d nearly converge. Therefore, by monitoring how many bots can exactly make the $\text{range}\langle RTT_i\rangle \simeq \text{range}\langle RTT_d\rangle$, the attackers can find an efficient bot number congesting a selected port.

Coordination procedure. The overall coordination procedure is described in Algorithm 1. As analyzed in § IV-B, the target port selection is an NP-hard problem. Therefore, we propose a heuristic greedy algorithm to select M target ports. For each p in the port set P, we define a heuristic function $H_{(F,\alpha)}(p) = \alpha|F_i(p)| + (1-\alpha)|F_d(p)|$, which describes the weighted sum of the covered direct and indirect victim flow number ($F_d(p)$ and $F_i(p)$, respectively) in the target flow set F. The attackers greedily select the p with the highest H which heuristically cuts most flows. Then they start to deploy bots sending bursts to the port, and monitor the RTT sequence $\langle RTT_d\rangle$ and $\langle RTT_i\rangle$ as the feedback signal. To cut more flows indirectly, we prefer the bots whose ingress links to p have lower flow density [44], i.e., the number of flows in F. The attackers stop adding bots anymore when the range of RTT_i and RTT_d sequence is approximately equal, where ΔR_i is similar to ΔR_d, as shown in Figure 5(b).

E. Schedule

In this subsection, we elaborate on the schedule procedure in the LoRDMA attack. The schedule procedure aims to obtain an efficient burst duration τ with high performance loss.

Problem statement. In the coordination step, the attackers assign bots to send bursts to the target ports with sufficiently long duration τ (1ms in our motivating case), leading to higher volume and lower efficiency. However, naively decreasing the duration may reduce the performance loss. According to Figure 3, too short τ hurts the performance loss significantly. Therefore, we adaptively decrease the burst duration τ with the probed RTT values as the feedback signal to check whether the performance loss decreases.

RTT pattern and PL. The queue length at the congestion point of indirect victim flows during the burst (reflected by $\langle RTT_i\rangle$), shows variable patterns. As Figure 4 shows, it first presents a peak pattern, and then gradually falls back and oscillates at the original value. This 2-phase pattern is analyzed in § III-C, as DCQCN cuts down F2 and F3 after several control loops and alleviates the congestion on F1 and F4 spread by PFC PAUSE frames. Therefore, the lower RTT_i subsequence can be trimmed without reducing the attack impact, while the higher RTT_i sub-sequence not.

Schedule procedure. The overall schedule procedure is described in Algorithm 2. The attackers gradually reduce the

[3]After obtaining Coord, the attackers can further assign a bot to send bursts to different hosts so that its burst traffic can pass through the target port. Such assignment [44], [93] is not essentially difficult, and we omit the details due to space limitations.

Algorithm 1: Coordination Procedure

```
Data: $P$ - switch egress port set; $B$ - bot set; $F$ - target flow set;
      $H_{(F,\alpha)}(p) = \alpha|F_i(p)| + (1-\alpha)|F_d(p)|$ - heuristic objective function for port $p$: a
      weighted sum of the number of direct/indirect victim flows in $F$; $fd_F(l)$ - how many flows in
      $F$ pass through link $l$; Inglink$(b,p)$ - the ingress link where $b$ sends traffic to $p$
Input: $M$ - number of target ports to select; $\alpha$ - parameter of $H_{(F,\alpha)}(p)$; $\varepsilon$ - desired precision
       level of range$\langle RTT \rangle$;
Output: $T_r$- set of selected target ports,
        Coord: $B \rightarrow T_r \cup \{\phi\}$ - mapping describes whether and which target egress port a bot
        should attack
1  while $|T_r| < M$ do
2      # Find the port with the max heuristic objective function
3      $p = \arg\max_{p \in P} H_{(F,\alpha)}(p)$
4      ADD$(p, T_r)$
5      # Deploy bots and monitor $RTT_d$ and $RTT_i$ sequence
6      while $|\text{range}\langle RTT_d \rangle - \text{range}\langle RTT_i \rangle)| > \varepsilon$ do
7          # When range$\langle RTT_i \rangle \simeq$ range$\langle RTT_d \rangle$, $\Delta R_i$ converges
8          # Deploy the bot with the lowest footprint
9          $b = \arg\min_{b \in B} fd_F(\text{Inglink}(b,p))$
10         Coord$(b) = p$
11         # Remove the corresponding bot
12         REMOVE$(b, B)$
13     # Remove the corresponding victim flows and port
14     REMOVE$(F_d(p) \cup F_i(p), F)$
15     REMOVE$(p, P)$
16 return $T_r$, Coord
```

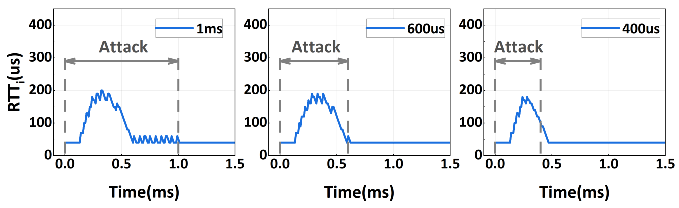


Fig. 6. RTT_i variation with different burst duration.

burst duration τ while maintaining the higher sub-sequence of the RTT_i intact. The attackers tune the duration in a bisection method. For every step, the attackers monitor the RTT_i sequence during the burst ($\langle RTT_i \rangle$), filter the higher RTT_i sub-sequence by a threshold (e.g., $100\mu s$), and check wether it remains intact. For example, as Figure 6 shows, when τ is decreased at 600μs, the peak pattern is still intact. But when at 400μs, the peak pattern gets cut, and the performance loss on indirect victim flows PL_i reduces (Figure 3(b)). Therefore, the 400μs duration is too short and 600μs is an efficient value.

F. Implementation Discussions

In this subsection, we discuss the implementation of the LoRDMA attack tools, including a line-rate burst generator and an RTT prober. Our prototype of attack tools is publicly available [97].

Algorithm 2: Schedule Procedure

```
Data: $\langle RTT_i \rangle_\tau$ - RTT sequence of indirect victim flows during $\tau$
Input: $\theta$ - threshold to filter high RTT sequence; $\tau_0$ - original burst duration; $\tau_{min}$ - minimum burst
       duration; $\epsilon$ - desired precision level of $\tau$;
Output: $\tau$ - final burst duration
1  # Filter out $RTT_i$ subsequence higher than $\theta$
2  $\langle RTT_i \rangle'_{\tau_0} = \text{filter}(\langle RTT_i \rangle_{\tau_0}, \theta)$
3  # Adaptively find convergence $\tau$ using bisection
4  $\tau = \tau_0$
5  $\tau_{lo} = \tau_{min}$
6  $\tau_{hi} = \tau_0$
7  while $|\tau_{hi} - \tau_{lo}| > \epsilon$ do
8      $\tau = (\tau_{hi} + \tau_{lo})/2$
9      $\langle RTT_i \rangle'_\tau = \text{filter}(\langle RTT_i \rangle_\tau, \theta)$
10     if len$(\langle RTT_i \rangle'_\tau) <$ len$(\langle RTT_i \rangle'_{\tau_0})$ then
11         $\tau_{lo} = \tau$
12     else
13         $\tau_{hi} = \tau$
14 return $\tau$
```

Burst generator. To make the LoRDMA attack practical, the burst generator should 1) craft line-rate burst consisting of attacker-specified packet bytes, and 2) control the burst duration (τ) and period (T). Fortunately, the capability of current RNICs fully supports these features. The RNIC hardware version newer than Mellanox CX-4 [76] and the RDMA Core driver version newer than MLNX_OFED v5.0 [77] have supported `IBV_QPT_RAW_PACKET`, a new QP type which enables high-performance kernel-bypass raw Ethernet programming on the RNIC [83], [74]. Therefore, the attackers can utilize the RNIC as a high-performance burst generator, and craft line-rate bursts to any host in the network by carefully setting the packet headers. For example, the attackers set the packet length as MTU for higher bandwidth usage, and set the destination IP address as the remote hosts to which the path passes through the target egress port. To avoid the packet being classified into normal TCP/IP traffic instead of RoCEv2 traffic, the DSCP field (the ToS bits in the IP header) should be set carefully. To control the duration and period of the burst, the burst generator should record the time during burst crafting. To avoid hurting the performance by wasting CPU cycles in time recording, we create a new thread in parallel as the timer.

RTT prober. As we mentioned in § IV-C, by using the RTT side-channel signal, a bot can estimate the RTT to any host in the network. We utilize the raw ethernet programming feature of RNIC and craft CM `ConnectRequest` packets [58]. Then the RNIC sniffs the corresponding reply packet, which is usually a `ConnectReject`, and estimates the RTT by subtracting their respective timestamps.

V. Evaluation

In this section, we conduct extensive evaluations on the LoRDMA attack. We first evaluate the effectiveness and efficiency of the coordination and schedule procedures in LoRDMA, then evaluate the impact of LoRDMA on RDMA applications, and finally validate LoRDMA at real testbeds.

A. Coordination Performance

In this subsection, we evaluate the effectiveness and cost of the coordination procedure with ns-3. We simulate networks with different topologies and traffic distributions, and compare the performance of different attack coordination methods.

Metrics. To evaluate the coordination process with different topologies, we define four metrics. First, we define the *coverage ratio* as the percentage of the target flows that the coordination methods can cut in the network, and define the *average of normalized performance loss factor* (Norm $\Delta R = \Delta R/R_0$) on flows to describe the overall impact on the target flows. Second, we also consider the attack footprint that affects the detectability. As described in § III-D, many DoS and LFA detection systems [102], [111], [43] monitor congested links and locate attack traffic from the traffic flowing through them. Therefore, we first record the *number of directly congested queues* required by different coordination methods to overwhelm using the burst traffic. Besides, existing performance anomaly diagnosis systems mainly analyse the flow contention with the victim flows in a queue [54], [99] to identify attack traffic. We accordingly define the *direct flow contention*, i.e., the percentage of target flows that pass through and experience the congestion in the directly congested queues.

Setup. To evaluate the robustness of the coordination process, we simulate 3 publicly available network topologies from topology zoo [47]: a small topology (Carnet), a medium topology (Switch), and a large topology (Cernet), as shown in Figure 12 in Appendix B. For each topology, we generate 10 target flow distribution cases, by randomly picking 2 hosts as the source and destination and setting a flow through the shortest path between them. Among these flows we set 20% as target flows to attack and the remaining flows as background traffic. We first set an ideal LoRDMA attack, where the attackers know all the flows in the network. The attackers then try to indirectly cut the target flows by exploiting PFC-spreading congestion to avoid direct flow contention. The second attack is an LoRDMA attack where the attackers only know target flows, and they try to cover more target flows with less direct congestion. Third, we also setup a baseline attack. To the best of our knowledge, there is no existing attack against RDMA traffic control (especially PFC), therefore, we set up the baseline inspired by the Crossfire [44], a classic link flood attack cutting links similarly. It keeps selecting the link with the highest flow density and deploying bots to congest it until no available bot remains. The key difference is that it does not utilize the indirect-cutting feature. It only deploys few (e.g. 2) bots for each target port and thus there are only direct victim flows by the queue contending between attack traffic and victim flows. Each coordination method generates a bot-to-target-port mapping to form congestion trees (LoRDMA) or links (Crossfire). We then count the target flows covered in the directly&indirectly cut links and their Norm ΔR, as well as the number of target congestion points and the target flows passing through these congestion points in each attack solution.

Figure 7 shows the performance of the attack solutions generated by different coordination methods in different topologies. Since different topologies have different bot number and switch node, we describe the bot number by a normalized factor, denoted as the ratio between the actual bot number and the switch number. First, compared with the Crossfire attack, the LoRDMA attack only aware of the target flows covers more victim flows ($\sim$10% higher on average). Besides the coverage, it also causes higher performance loss factor ΔR in each topology, demonstrating that indirectly cutting can cause similar performance loss with direct congestion. The LoRDMA attack aware of the background traffic, i.e., the ideal LoRDMA attack, which mainly indirectly cuts the target flows, causes a bit higher victim flow coverage and performance degradation. These results also show that when the number of bots is small, the indirectly cutting effect can be leveraged to achieve greater coverage and performance degradation. Furthermore, the LoRDMA attack also shows lower cost. Compared with the Crossfire attack, the LoRDMA attack has fewer congested ports directly overwhelmed by the bursts ($\sim$50% on average), and lower queue contention ($\sim$20% on average) between the burst traffic and target flows. The ideal LoRDMA attack which exploits the background traffic and indirectly cuts the target flows presents similar direct congestion point number. However, since it tries to select the link where fewer target flows go through, the direct flow contention is lower, also presenting lower attack footprint. To summarize, LoRDMA presents higher effectiveness and lower cost compared to the straightforward link-flood attacks in different topologies.

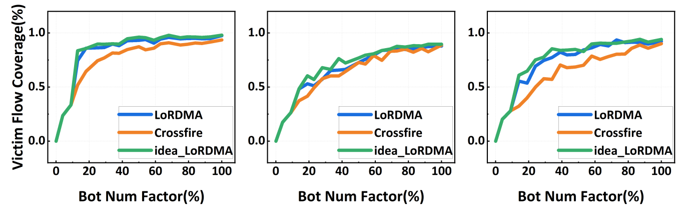
B. Schedule Performance

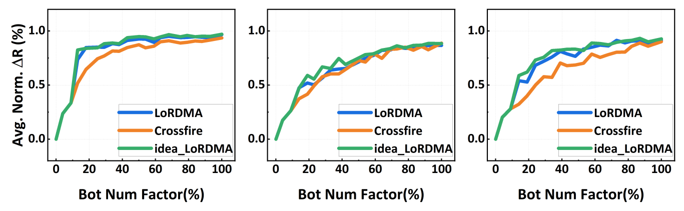
In this subsection, we evaluate the performance of the schedule procedure with ns-3. We first define the impact and efficiency of an attack with a specific parameter set (δ, τ), and then demonstrate the effectiveness of the schedule procedure under different attack parameters across various background traffic scenarios.

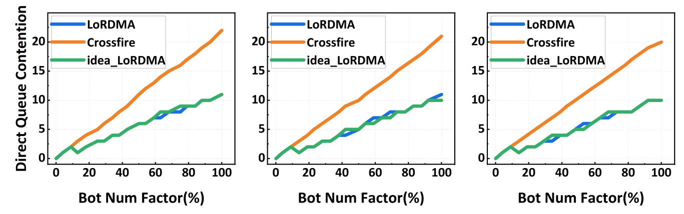
Metrics. We first define the impact and cost of an attack with a specific parameter set, and evaluate the efficiency correspondingly. We define the impact as the summed performance loss PL of each target flow (defined in § III-A). Specifically, the impact of an attack is defined as the weighted summed performance loss: $Impact = \Sigma(\alpha PL_i + (1-\alpha) PL_d)$, because an effective attack should always try to cause higher performance loss on as many indirect victim flows as possible, as discussed in § IV-D and § IV-E. And the cost is defined as the attack burst bandwidth $Cost = \delta * \tau$. We further define the efficiency of an attack as the ratio of impact and cost naturally [24]: $Efficiency = Impact/Cost$.

Setup. We set a multi-rack topology (Figure 13 in Appendix D) derived from Fat-tree topologies [1], [14] via abstracting the leaf and spine switches as one-big-switch (SW5), as Lao et al. do [52]. We set ≤ 1 bot under each ToR switch (SW1-4, SW6-9), and set up a target flow scenario similar to our motivating experiments (Figure 1(a)). We also add background traffic of different rate (0Gbps, 33.3Gbps, 50Gbps respectively) on the links of target flows. The attackers then craft bursts to congest SW5.P6 with different burst duration (τ).

Figure 8 shows the efficiency of the LoRDMA attack with different τ. First, the efficiency manifests as a pronounced


(a) Victim flow coverage at Carnet, Switch and Cernet, respectively.


(b) Average ΔR at Carnet, Switch and Cernet, respectively.


(c) Directly congested queue number at Carnet, Switch and Cernet, respectively.

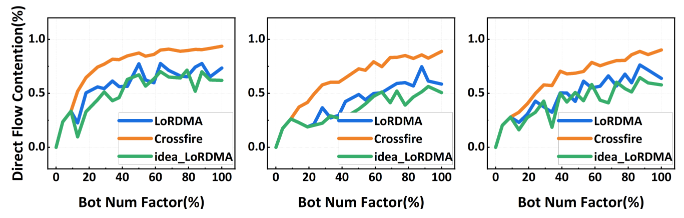
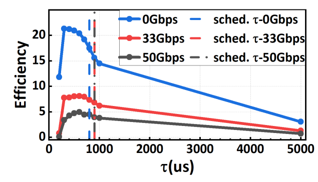

(d) Direct flow contention at Carnet, Switch and Cernet, respectively.

Fig. 7. Coordination performance with different bot number in different topologies.


(a) Attack efficiency as τ changes with different background traffic scenarios.

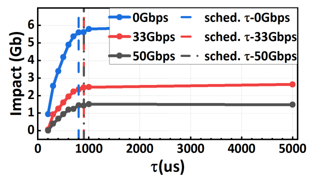
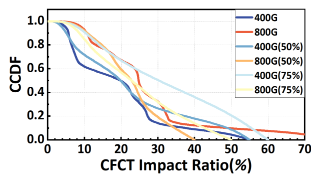
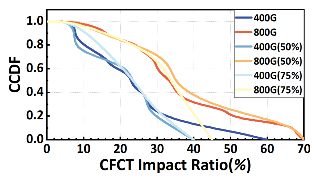
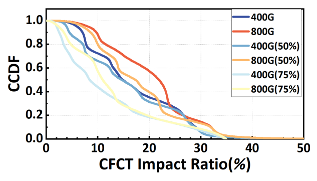

(b) Attack impact as τ changes with different background traffic scenarios.

Fig. 8. Schedule performance of the LoRDMA attack with different parameters.

unimodal function, indicating an optimal point exists. The LoRDMA attack with our schedule procedure approach, albeit not identifying the ideal parameter, consistently obtains a sufficiently efficient attack parameter across various background traffic conditions, compared with the preset adequately long fixed τ (e.g., 1ms or 2ms). Second, the LoRDMA attack with our schedule procedure can induce a sufficiently high impact under diverse background traffic conditions. Given that the impact is defined based on bandwidth loss, which is inherently associated with the original rate of the target flow, it can be lowered by bandwidth contention caused by background traffic. However, the LoRDMA attack can still lead to a considerable bandwidth degradation, approximately equivalent to the predetermined value such as 1ms and 2ms.


(a) Co-flow completion time impact ratio of distributed machine learning training with a low flow number.


(b) Co-flow completion time impact ratio of cloud storage with a low flow number.


(c) Co-flow completion time impact ratio of distributed machine learning training with a high flow number.

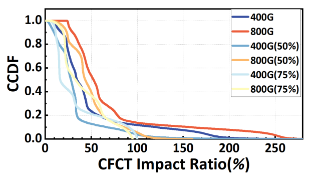

(d) Co-flow completion time impact ratio of cloud storage with a high flow number.

Fig. 9. Complementary Cumulative Distribution Function (CCDF) of the performance loss ratio.

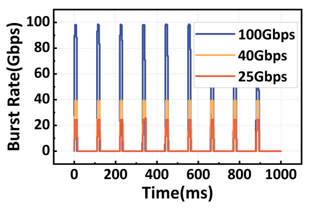
C. Impact on Real Applications

In this subsection, we evaluate the impact of the LoRDMA attack on different RDMA applications in a large-scale Fat-tree topology with ns-3.

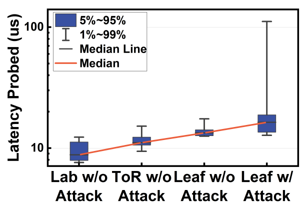
Metrics. We pick up 2 typical RDMA applications: distributed machine learning training [37], [52], and cloud storage [21], [90]. In the former application scenario, such as a data-parallel parameter-server (PS) architecture, multiple training nodes transport the batched data in the same size to the parameter server after a step of training. The latter has significantly different workload, and the flow size is quite diverse, showing a long-tailed distribution [84]. The communication pattern and performance of both application scenarios are closely correlated with the co-flow [13], [108]. Therefore, we accordingly evaluate the performance degradation on co-flows at different application classes.

Setup. We carry out evaluations in a Fat-tree (k = 8) topology [1], [14]. The link capacity is 100Gbps and the link delay is $5\mu s$. The network configuration is the same as Table II in Appendix C, and the ECMP routing scheme is deployed at each switch. We assume that every ToR switch has at most 1 bot for attackers. To simplify the coordination procedure, we specify that the source nodes of flows are all located on hosts under ToR1-ToR4 and the destination nodes are located on hosts under ToR5-ToR8. Therefore, the bots can select one of the egress queues under ToR5-ToR8 as the target symmetrically. We also omit the schedule process and set the burst duration as $500\mu s$.

Workload. We choose multiple workloads from two independent aspects: 1) the flow number and 2) the flow size. For the flow number, we set a different number of hosts within each ToR switch. Specifically, we set 9 or 25 hosts under each ToR switch respectively, and set the source and destination host of each flow arbitrarily with a random incast ratio varying from 1 to 4, or 1 to 16 correspondingly. For the flow size, we select different flow size distributions for each scenario. In distributed machine learning training, every flow has the same size of 12.8MB. And for cloud storage, we set a typical long-tailed traffic size distribution in operating data centers [84],

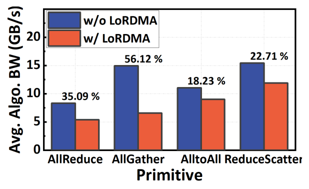


(a) Burst generator performance.

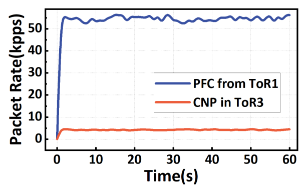


(b) Prober result.

Fig. 10. Validation of attack tools in real network.



(a) Performance of different communication primitives.



(b) PFC and CNP count over time.

Fig. 11. Attack impact in real network.

where $< 80\%$ of flows are smaller than 10MB, $< 90\%$ of flows are smaller than 100MB, and $\sim 10\%$ flows are larger than 200MB. We set various proportions of background traffic in these flows (0%, 50%, 75% respectively), and then observe the performance degradation experienced by the target flows under each background traffic condition.

We craft a $500\mu s$ burst from 4 and 8 bots respectively, and monitor the impact. We term the co-flow as the flows incasting to the same host. We further define the co-flow completion time (CFCT) impact ratio as the $\Delta CFCT/CFCT$ and evaluate the overall impact ratio in different applications. Our evaluation results show that a $500\mu s$ burst can cause significant performance loss. As shown in Figure 9, all co-flows in different scenarios have non-negligible performance loss. The minimum damage suffered by the coflows at different cases ranged from 1% to 24.5%, with a average value of 7.12%. And the median damage on coflows at each case varies from 8.11% to 52.7%, averaging at 25.2%. Furthermore, the maximum damage on coflows at each case varies from 29.1% to 251.6%, averaging at 65.47%. For a single burst period, short flows, which constitute the majority in data center traffic, suffer a higher impact ratio because the same performance loss counts more in terms of their total transmitted bytes. Furthermore, with varying proportions of background traffic, the degradation distribution is similar, which indicates that background traffic does not substantially influence the overall impact on coflow completion time.

D. Real Testbed and Cloud RDMA Cluster Evaluation

We validate the LoRDMA attack at a real testbed from our lab and a private cloud RDMA cluster from Kuaishou Technology with prior approval. The real testbed from our lab consists of a simple back-to-back topology with 2 DELL PowerEdge R730 servers, and each server is equipped with an NVIDIA Mellanox ConnectX-4 SmartNIC(40GbE), 2 Intel Xeon E5-2620 v3 CPUs (2.40GHz) and 128GB RAM. Each server runs on Ubuntu 16.04 and MLNX_OFED 5.4. This simple testbed is used to implement and evaluate our attack tools. The cloud RDMA cluster from Kuaishou Technology consists of 8 Inspur SA5280M6 servers, and each server is equipped with an NVIDIA Mellanox ConnectX-6 SmartNIC (100GbE), 2 Intel Xeon Platinum 8352Y CPUs (2.20GHz), 1024GB RAM, and 1 NVIDIA Tesla A10 GPUs. Each server runs on CentOS 7.4 and MLNX_OFED 5.4. All the devices are connected through a Fat-tree with 4 ToR switches and 2 leaf switches, as shown in Figure 14 in Appendix § E. The link capacity between leaf and ToR switches is 400Gbps, and the link capacity between ToR switches and severs is 100Gbps. All the attacker and victim applications are launched at the servers using host-model docker containers.

We first evaluate the performance of our burst generator. We run the burster at one of the RNICs in each real environment. And we also run the burster between 2 `scch5s` instances in Alibaba Cloud rented by ourselves. Since we do not control all the instances in the network, we set a 25Gbps rate limitation to reduce the impact on the network. The burst duration is 10ms and the period is 110ms. Figure 10(a) shows that the burst can reach line rate (40Gbps in our lab and 100Gbps in Kuaishou, respectively) or maximum rate (25Gbps in Alibaba) with precise duration and period control, validating the feasibility of crafting line-rate bursts with a single RNIC.

We then evaluate the performance of the prober. Figure 10(b) shows that it presents a stable correlation with the actual network delay in different end-to-end communication situations. Specifically, when not under attack, the probed latency for back-to-back communication (*Lab w/o Attack*), passing through one ToR switch (*ToR w/o Attack*) and a leaf switch (*Leaf w/o Attack*) is relatively stable at low levels. And the heavily long-tailed delays under attack are also well reflected (*Leaf w/ Attack*). All these evaluations show that our prober can accurately reflect the delay.

We also conduct a LoRDMA attack against real-world applications in the Kuaishou cloud RDMA cluster. We set up `NCCL Tests` [75], the test suite for the famous inter-GPU collective communication library NCCL, which is widely used in today's mainstream machine learning training frameworks. Specifically, we run different typical collective communication primitives, including AllReduce, AllGather, ReduceScatter and AlltoAll, as victims, and monitor their performance degradation under the LoRDMA attack. The algorithm bandwidth is defined as S/t, where S is the size of the data in the communication operation, and t is the time to the completion. As shown in Figure 14, we set up H1, H3 and H4 as the nodes running `NCCL Tests`. We then craft high-rate burst traffic to H2 from H4, H7, and H8, so that the attack traffic can only indirectly cut the target flows by spreading PFC. Due to the second-level switch monitoring granularity in the cloud RDMA cluster, we set a longer burst duration (10ms) to provide clearer network diagnostic data. We also set the period as 110 ms. Figure 11(a) illustrates the performance (algorithm bandwidth) with/without the LoRDMA attack. The percentages above the bars indicate the performance degradation ratio caused by LoRDMA. All the communication primitives suffers significant performance degradation, from 18.23% (AlltoAll) to 56.12% (AllGather). Figure 11(b) shows the network diagnostic information during the attack on AllReduce. ToR1 switch generates a large number of PFC PAUSE frames upstream, while ToR3 switch receives numerous CNP packets destined downstream to H5 and H6, validating our insight that PFC can be back-spread to mislead DCQCN severely.

VI. Defense Schemes

In this section, we discuss some potential solutions to detect and defend against the LoRDMA attack.

PFC-driven network anomaly detection and analysis. Numerous studies on network performance anomaly monitoring or DoS defense [113], [114], [99], [102], [111] consider the number of congested queues and the major queue contention contributor as key metrics to locate the culprit attacker. However, such philosophy may be less effective due to the congestion spreading caused by PFC, because the attack traffic may have a lower congested queue number and contention contribution, as analyzed in § III. We observe that the key to defend the LoRDMA attack is to take the PFC pause frames spreading into the causality dependency construction. By analyzing the causality of the spreading of PFC pause frames, the root cause of the congestion can be found, and the culprit traffic can then be located. We leave designing a precise and responsive RDMA network performance anomaly diagnosis system as our future work.

Fine-grained burst monitor. The millisecond- or even microsend-level burst duration requires that the monitoring granularity should be as fine as the corresponding level. For example, it is challenging to pinpoint the 1ms burst with a monitoring window of 50ms. Besides, the line-rate start property of DCQCN and PFC also makes the attack burst less outstanding in RDMA networks, rendering rate-based detection less effective, which requires a more responsive and precise mechanism. Moreover, the monitor mechanism should introduce low overhead to avoid violating the hardware kernel-bypass paradigm of RDMA. However, it is challenging for both in-network [113], [28], [111], [99] or host-based monitoring [27], [22], [69] to make a perfect trade-off between responsiveness, effectiveness and overhead, especially considering the harsh performance requirement in RDMA networks, which requires new ideas to resolve these problems.

Network fabric level isolation & virtualization. The feasibility of the LoRDMA attack lies in the sharing of the network infrastructure between hosts including target victims and attackers. Therefore, network fabric isolation can theoretically eliminate the infrastructure sharing and provides bounded bandwidth guarantees for each tenant, as the studies in traditional TCP data centers demonstrate [98], [81], [51]. However, currently fabric isolation mechanisms in RDMA network are still lacking, which needs more research effort. Besides fabric isolation, virtual RDMA overlay networking [18], [46], [29] may prevent the attackers from probing the real physical network environment and make the target selection difficult, which is promising to alleviate the damage of the LoRDMA attack. However, the underlay physical infrastructure can still be impacted, so the LoRDMA attack still takes effect to some extent. To conclude, more research effort on the fabric level isolation and virtualization in RDMA network is in urgent need to fully mitigate the LoRDMA attack.

VII. Discussion

Different degrees of network traffic knowledge. Our threat model assumes that the attackers know the target flow route set, which is a subset of the whole network traffic. However, the LoRDMA attack remains feasible with different knowledge of network traffic. An ideal scenario is that the attackers know not only the target flow but also a part of background traffic. To avoid the direct queue contention with target flows (we assume the attacker only cares about the performance loss and queue contention on the target flows), the attackers should try to coordinate carefully by putting more background flows at direct victim flows and covering more target flows. Our attack methods can suit such a scenario by assigning different weights for target and background flows, and we omit the details for space limitations. Furthermore, the LoRDMA attack can still work by relaxing the assumption of prior knowledge of attackers. Specifically, suppose that the attackers only know the network topology, i.e., only the network link information, but have no idea of the exact target flow set. In that case, the attackers choose some target links via some metrics or algorithms [44], [17], [50], [72]. Then, organizing attacks can be reduced into a coverage problem with the congestion tree caused by the bot traffic. Note that these attacks are similar to but still different from traditional link flood attacks [44], [9], [93], because the target link can be cut off using the PFC spread from downstream instead of high-load traffic.

Effect of background traffic. Background traffic can potentially affect the LoRDMA attack in multiple aspects. First, background traffic affects the attack impact since it shares links with target flows, which has been extensively evaluated and analyzed in § V. Second, background traffic may affect the RTT measurement. Stable background traffic only introduce a constant background noise and still enables reliable queue length estimations and RTT measurement. However, highly dynamic traffic may interfere with the measurement, leading to sub-optimal burst parameters potentially. Nevertheless, even sub-optimal burst parameters still cause significant performance degradation. Third, background traffic can affect burst synchronization to the target link due to varying path delays. Note that dynamic burst alignment is a well-studied topics which is orthogonal to our work [45], [82]. And even bursts that are not strictly synchronized can still cause severe congestion.

Attack feasibility with different queue disciplines. The congestion caused by bursts and the RTT measurement require queue sharing between the victim traffic and attack/probing traffic. Some ideal queue disciplines, such as fair queuing [70] which allocates a separate queue for each flow, can theoretically isolate the malicious traffic from the benign traffic. However, implementing these complex queue disciplines at line-rate (100Gbps) is impossible for current commercial switches due to resource constraints. Existing approximating fair queue scheduling methods (e.g., AFQ [87], PCQ [88], SP-PIFO [2], AIFO [107], and Cebinae [106]), usually require new hardware features such as programmable switching ASICs and more queuing resources, which are less supported in production environments. Current commercial switches usually set a static queue number and share buffers across ports, resulting in queue sharing between flows. Besides, most data flows (including attack/probing traffic) share the same priority and thus equal network resources. Therefore, attack traffic can still cause queue congestion and cut normal flows; and probing traffic can also estimate the congestion impact of normal flows.

Attack feasibility with different congestion control variants. Although in this paper we mainly focus DCQCN, the most widely used congestion control algorithm in production environments, LoRDMA can also be effective against other congestion control variants. Many end-to-end congestion control algorithms follow similar principles: 1) monitor network feedback as congestion signals of each flow (queue length or delay); 2) adjust flow rate in an AIMD way to achieve fairness among different flows. As a result, the hop-by-hop PFC can manipulate the flow-level signals by spreading the congestion to the flow paths (e.g., queue congestion or long delay), and mislead the congestion control to cut the rate of some innocent flows. We have also validated this vulnerability in a number of other congestion control variants, as shown in Appendix F.

ECMP and LoRDMA. ECMP is widely deployed in current data center networks for load balance, and may also influence LoRDMA if other equivalent links exist for the selected target link, e.g., the links between leaf and ToR switches (Figure 14 in Appendix E). First, the probing accuracy may be affected, because the probing traffic may go through different equivalent links. Second, since ECMP aims to distribute the traffic into different equal-cost paths, congesting the target link may require higher burst rate and thus higher bot number. However, for the target links without other equal-cost paths (the links between ToR switchs and hosts in Figure 14 in Appendix E), probing and congesting are similar to those in non-ECMP topologies.

Coverage and accuracy of RTT probing. The coordination and schedule procedures require that the attackers should probe the RTT of indirect and direct victim flows. Since the position of bots cannot be fully determined by the attackers, the coverage on the target flows is not always satisfied. We leave inferring the performance loss with partial RTT probing as our future work. Besides, the accuracy of congestion estimation by RTT probing can also be impacted by the noise, such as the probing packet processing latency on the remote host. It can be augmented by more statistical methods or control theory models [42], which is left as future work.

Extensibility of the LoRDMA attack. In this work we term the attack impact and cost as the bandwidth loss and average attack burst volume, respectively. However, we claim that the metrics can have various definition instances. For example, for an attacker aiming to maximize the delay, a natural definition of attack impact can be the difference between the communication delay before and after the attack. Furthermore, the attack cost can also consist of different metrics such as the attack duration (τ), the topological similarity of the target flow (e.g., Levenshtein distance [64], [55]), etc. Besides, the attack method can also be improved to achieve higher efficiency using more feedback control-theoretic frameworks, such as the Kalman filter [42], [86] or Reinforcement Learning [53]. We leave a more flexible attack framework as our future work.

VIII. Ethical Considerations

No ethical consideration is raised in our experiments. The experiments at the Kuiashou cloud RDMA cluster are conducted with prior approval. The experiments on ns-3 simulations and the testbed in our lab are conducted with all devices in our control. The experiments in Alibaba Cloud are conducted between our controlled instances with strict rate restrictions, and operated at the light traffic periods for a very short time.

IX. Conclusion

In this paper, we identify the security vulnerabilities of RDMA traffic control mechanisms through comprehensive experiments and analysis. We then propose the LoRDMA attack, an efficient low-rate DoS attack which causes significant performance degradation in RDMA networks. The coordination and schedule procedures provide high performance loss and victim coverage, and require low average burst rate and direct queue contention. Our evaluations validate the feasibility of the LoRDMA attack, and demonstrate it is highly effective and efficient in deteriorating the RDMA performance.

Acknowledgment

We thank our shepherd and the anonymous NDSS reviewers for their valuable feedbacks. This work is supported in part by the National Natural Science Foundation of China (No. 62221003 and No. 62172251), the National Key Research and Development Program of China under Grant No. 2022YFB3105001, the China Postdoctoral Science Foundation (2022M720202), and the Beijing Postdoctoral Research Foundation (2022-ZZ-078). Menghao Zhang and Zhiliang Wang are the corresponding authors.

References

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM CCR*, vol. 38, no. 4, 2008.

[2] A. G. Alcoz, A. Dietmüller, and L. Vanbever, "SP-PIFO: Approximating Push-In First-Out behaviors using Strict-Priority queues," in *USENIX NSDI*, 2020.

[3] Alibaba, "Alibaba builds high-speed RDMA network for AI and scientific computing." https://www.alibabacloud.com/blog/alibaba-builds-high-speed-rdma-network-for-ai-and-scientific-computing_594895, 2019.

[4] Azure, "High performance computing vm sizes." https://learn.microsoft.com/en-us/azure/virtual-machines/sizes-hpc/, 2023.

[5] 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 *USENIX NSDI*, 2023.

[6] R. T. Braden, "Requirements for Internet Hosts - Communication Layers," RFC 1122, 1989. [Online]. Available: https://www.rfc-editor.org/info/rfc1122

[7] M. Burke, S. Dharanipragada, S. Joyner, A. Szekeres, J. Nelson, I. Zhang, and D. R. Ports, "PRISM: Rethinking the rdma interface for distributed systems," in *ACM SIGOPS*, 2021.

[8] Q. Cai, W. Guo, H. Zhang, D. Agrawal, G. Chen, B. C. Ooi, K.-L. Tan, Y. M. Teo, and S. Wang, "Efficient distributed memory management with rdma and caching," *Proceedings of the VLDB Endowment*, vol. 11, no. 11, 2018.

[9] J. Cao, Q. Li, R. Xie, K. Sun, G. Gu, M. Xu, and Y. Yang, "The CrossPath attack: Disrupting the SDN control channel via shared links," in *USENIX Security*, 2019.

[10] Y. Chen, X. Wei, J. Shi, R. Chen, and H. Chen, "Fast and general distributed transactions using RDMA and HTM," in *EuroSys*, 2016.

[11] Y. Chen, Y. Lu, and J. Shu, "Scalable RDMA RPC on Reliable Connection with Efficient Resource Sharing," in *EuroSys*, 2019.

[12] W. Cheng, K. Qian, W. Jiang, T. Zhang, and F. Ren, "Re-architecting congestion management in lossless ethernet," in *USENIX NSDI*, 2020.

[13] M. Chowdhury and I. Stoica, "Coflow: A networking abstraction for cluster applications," in *ACM HotNets*, 2012.

[14] C. Clos, "A study of non-blocking switching networks," *The Bell System Technical Journal*, vol. 32, no. 2, 1953.

[15] R. Cohen and L. Katzir, "The generalized maximum coverage problem," *Information Processing Letters*, vol. 108, no. 1, 2008.

[16] A. Dragojević, D. Narayanan, M. Castro, and O. Hodson, "FaRM: Fast remote memory," in *USENIX NSDI*, 2014.

[17] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," *ACM SIGCOMM CCR*, vol. 29, no. 4, 1999.

[18] D. Firestone, A. Putnam, S. Mundkur, D. Chiou, A. Dabagh *et al.*, "Azure Accelerated Networking: SmartNICs in the Public Cloud," in *USENIX NSDI*, 2018.

[19] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, 1993.

[20] S. Floyd, D. K. K. Ramakrishnan, and D. L. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168, 2001. [Online]. Available: https://www.rfc-editor.org/info/rfc3168

[21] Y. Gao, Q. Li, L. Tang, Y. Xi, P. Zhang, W. Peng, B. Li, Y. Wu, S. Liu, L. Yan *et al.*, "When cloud storage meets RDMA," in *USENIX NSDI*, 2021.

[22] M. Ghasemi, T. Benson, and J. Rexford, "Dapper: Data plane performance diagnosis of tcp," in *ACM SOSR*, 2017.

[23] J. Gu, Y. Lee, Y. Zhang, M. Chowdhury, and K. G. Shin, "Efficient Memory Disaggregation with Infiniswap," in *USENIX NSDI*, 2017.

[24] M. Guirguis, A. Bestavros, and I. Matta, "Exploiting the transients of adaptation for RoQ attacks on Internet resources," in *IEEE ICNP*, 2004.

[25] M. Guirguis, A. Bestavros, I. Matta, and Y. Zhang, "Reduction of Quality (RoQ) Attacks on Dynamic Load Balancers: Vulnerability Assessment and Design Tradeoffs," in *IEEE INFOCOM*, 2007.

[26] 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.

[27] C. Guo, L. Yuan, D. Xiang, Y. Dang, R. Huang, D. Maltz, Z. Liu, V. Wang, B. Pang, H. Chen, Z.-W. Lin, and V. Kurien, "Pingmesh: A large-scale system for data center network latency measurement and analysis," in *ACM SIGCOMM*, 2015.

[28] A. Gupta, R. Harrison, M. Canini, N. Feamster, J. Rexford, and W. Willinger, "Sonata: Query-driven streaming network telemetry," in *ACM SIGCOMM*, 2018.

[29] Z. He, D. Wang, B. Fu, K. Tan, B. Hua, Z.-L. Zhang, and K. Zheng, "MasQ: RDMA for virtual private cloud," in *ACM SIGCOMM*, 2020.

[30] N. Hu, L. E. Li, Z. M. Mao, P. Steenkiste, and J. Wang, "Locating Internet bottlenecks: Algorithms, measurements, and implications," in *ACM SIGCOMM*, 2004.

[31] S. Hu, Y. Zhu, P. Cheng, C. Guo, K. Tan, J. Padhye, and K. Chen, "Deadlocks in datacenter networks: Why do they form, and how to avoid them," in *ACM HotNets*, 2016.

[32] IEEE, "IEEE 802.1 Qau - Congestion Notification." https://1.ieee802.org/dcb/802-1qau/, 2010.

[33] ——, "IEEE 802.1 Qbb - Priority-based Flow Control," https://1.ieee802.org/dcb/802-1qbb/, 2011.

[34] InfiniBand Trade Association, "InfiniBand Architecture Specification Release 1.2.1 Annex A17: RoCEv2," https://cw.infinibandta.org/document/dl/7781, 2014.

[35] ——, "InfiniBand Architecture Specification Volume 1 Release 1.4," https://cw.infinibandta.org/document/dl/8567, 2020.

[36] S. Jero, M. E. Hoque, D. R. Choffnes, A. Mislove, and C. Nita-Rotaru, "Automated attack discovery in TCP congestion control using a model-guided approach," in *NDSS*, 2018.

[37] Y. Jiang, Y. Zhu, C. Lan, B. Yi, Y. Cui, and C. Guo, "A Unified Architecture for Accelerating Distributed DNN Training in Heterogeneous GPU/CPU Clusters," in *USENIX OSDI*, 2020.

[38] Jiarong Xing, Kuo-Feng Hsu, Yiming Qiu, Ziyang Yang, Hongyi Liu, and Ang Chen, "Bedrock: Programmable network support for secure RDMA systems," in *USENIX Security*, 2022.

[39] A. Kalia, M. Kaminsky, and D. G. Andersen, "Using RDMA efficiently for key-value services," in *ACM SIGCOMM*, 2014.

[40] ——, "Design guidelines for high performance RDMA systems," in *USENIX ATC*, 2016.

[41] ——, "FaSST: Fast, scalable and simple distributed transactions with Two-Sided (RDMA) datagram RPCs," in *USENIX OSDI*, 2016.

[42] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, 1960.

[43] M. S. Kang, V. D. Gligor, and V. Sekar, "SPIFFY: inducing cost-detectability tradeoffs for persistent link-flooding attacks," in *NDSS*, 2016.

[44] M. S. Kang, S. B. Lee, and V. D. Gligor, "The crossfire attack," in *IEEE S&P*, 2013.

[45] Y.-M. Ke, C.-W. Chen, H.-C. Hsiao, A. Perrig, and V. Sekar, "CICADAS: Congesting the internet with coordinated and decentralized pulsating attacks," in *ACM Asia CCS*, 2016.

[46] D. Kim, T. Yu, H. H. Liu, Y. Zhu, J. Padhye, S. Raindel, C. Guo, V. Sekar, and S. Seshan, "FreeFlow: Software-based virtual RDMA networking for containerized clouds," in *USENIX NSDI*, 2019.

[47] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE JSAC*, vol. 29, no. 9, 2011.

[48] 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.

[49] A. Kuzmanovic and E. W. Knightly, "Low-Rate TCP-Targeted Denial of Service Attacks: The Shrew vs. the Mice and Elephants," in *ACM SIGCOMM*, 2003.

[50] A. Lakhina, J. Byers, M. Crovella, and P. Xie, "Sampling biases in ip topology measurements," in *IEEE INFOCOM*, 2003.

[51] V. T. Lam, S. Radhakrishnan, R. Pan, A. Vahdat, and G. Varghese, "Netshare and stochastic netshare: Predictable bandwidth allocation for data centers," *SIGCOMM CCR*, 2012.

[52] C. Lao, Y. Le, K. Mahajan, Y. Chen, W. Wu, A. Akella, and M. Swift, "ATP: In-network Aggregation for Multi-tenant Learning," in *USENIX NSDI*, 2021.

[53] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *NATURE*, vol. 521, 2015.

[54] Y. Lei, L. Yu, V. Liu, and M. Xu, "PrintQueue: performance diagnosis via queue measurement in the data plane," in *ACM SIGCOMM*, 2022.

[55] V. I. Levenshtein *et al.*, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8, 1966.

[56] B. Li, T. Cui, Z. Wang, W. Bai, and L. Zhang, "Socksdirect: Datacenter sockets can be fast and compatible," in *ACM SIGCOMM*, 2019.

[57] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh, and M. Yu, "HPCC: High precision congestion control," in *ACM SIGCOMM*, 2019.

[58] linux rdma, "RDMA Core Userspace Libraries and Daemons," https://github.com/linux-rdma/rdma-core/tree/master/librdmacm, 2023.

[59] ——, "rping," https://github.com/linux-rdma/rdma-core/blob/master/librdmacm/examples/rping.c, 2023.

[60] Y. Lu, J. Shu, Y. Chen, and T. Li, "Octopus: an RDMA-enabled distributed persistent memory file system," in *USENIX ATC*, 2017.

[61] J. Luo, X. Yang, J. Wang, J. Xu, J. Sun, and K. Long, "On a Mathematical Model for Low-Rate Shrew DDoS," *IEEE TIFS*, vol. 9, no. 7, 2014.

[62] X. Luo and R. K. C. Chang, "On a new class of pulsing denial-of-service attacks and the defense," in *NDSS*, 2005.

[63] G. S. Malkin, "Traceroute Using an IP Option," RFC 1393, 1993. [Online]. Available: https://www.rfc-editor.org/info/rfc1393

[64] R. Meier, P. Tsankov, V. Lenders, L. Vanbever, and M. Vechev, "NetHide: Secure and practical network topology obfuscation," in *USENIX Security*, 2018.

[65] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *IEEE Globecom*, 2000.

[66] Mellanox, "DCQCN parameters." https://support.mellanox.com/s/article/dcqcn-parameters, 2020.

[67] C. Mitchell, Y. Geng, and J. Li, "Using One-Sided RDMA reads to build a fast, CPU-Efficient Key-Value store," in *USENIX ATC*, 2013.

[68] 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," in *ACM SIGCOMM*, 2015.

[69] M. Moshref, M. Yu, R. Govindan, and A. Vahdat, "Trumpet: Timely and precise triggers in data centers," in *ACM SIGCOMM*, 2016.

[70] J. Nagle, "On packet switches with infinite storage," *IEEE transactions on communications*, vol. 35, no. 4, 1987.

[71] S. Narayana, A. Sivaraman, V. Nathan, P. Goyal, V. Arun, M. Alizadeh, V. Jeyakumar, and C. Kim, "Language-Directed Hardware Design for Network Performance Monitoring," in *ACM SIGCOMM*, 2017.

[72] M. J. Newman, "A measure of betweenness centrality based on random walks," *Social Networks*, vol. 27, no. 1, 2005.

[73] A. Noroozian, M. Korczyński, C. H. Gañan, D. Makita, K. Yoshioka, and M. Van Eeten, "Who gets the boot? analyzing victimization by DDoS-as-a-service," in *Springer RAID*, 2016.

[74] Nvidia, "Raw Ethernet Programming: Basic Introduction - Code Example." https://enterprise-support.nvidia.com/s/article/raw-ethernet-programming--basic-introduction---code-example, 2022.

[75] ——, "NCCL Tests," https://github.com/NVIDIA/nccl-tests, 2023.

[76] ——, "NVIDIA Mellanox ConnectX-4 Lx Ethernet Adapter." https://www.nvidia.com/en-us/networking/ethernet/connectx-4-lx/, 2023.

[77] ——, "Raw Ethernet - RDMA Core (MLNX_OFED v5.0-1.0.0.0) - NVIDIA Networking Docs." https://docs.nvidia.com/networking/display/rdmacore50/Raw+Ethernet, 2023.

[78] ——, "RDMA Aware Networks Programming User Manual," https://docs.nvidia.com/networking/display/rdmaawareprogrammingv17, 2023.

[79] K. Ogata, *Modern Control Engineering*, 4th ed. USA: Prentice Hall PTR, 2001.

[80] J. Pinkerton and E. Deleganes, "Direct data placement protocol (DDP) / remote direct memory access protocol (RDMAP) security," RFC 5042, 2007. [Online]. Available: https://www.rfc-editor.org/info/rfc5042

[81] L. Popa, P. Yalagandula, S. Banerjee, J. C. Mogul, Y. Turner, and J. R. Santos, "Elasticswitch: Practical work-conserving bandwidth guarantees for cloud computing," in *ACM SIGCOMM*, 2013.

[82] R. Rasti, M. Murthy, N. Weaver, and V. Paxson, "Temporal Lensing and Its Application in Pulsing Denial-of-Service Attacks," in *2015 IEEE Symposium on Security and Privacy*, 2015.

[83] B. Rothenberger, K. Taranov, A. Perrig, and T. Hoefler, "ReDMArk: Bypassing RDMA Security Mechanisms," in *USENIX Security*, 2021.

[84] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *ACM SIGCOMM*, 2015.

[85] J. J. Santanna, R. van Rijswijk-Deij, R. Hofstede, A. Sperotto, M. Wierbosch, L. Z. Granville, and A. Pras, "Booters—an analysis of ddos-as-a-service attacks," in *IFIP/IEEE IM*, 2015.

[86] H. Shan, Q. Wang, and C. Pu, "Tail attacks on web applications," in *ACM CCS*, 2017.

[87] N. K. Sharma, M. Liu, K. Atreya, and A. Krishnamurthy, "Approximating fair queueing on reconfigurable switches," in *USENIX NSDI*, 2018.

[88] N. K. Sharma, C. Zhao, M. Liu, P. G. Kannan, C. Kim, A. Krishnamurthy, and A. Sivaraman, "Programmable Calendar Queues for High-speed Packet Scheduling," in *USENIX NSDI*, 2020.

[89] A. Shpiner, E. Zahavi, V. Zdornov, T. Anker, and M. Kadosh, "Unlocking credit loop deadlocks," in *ACM HotNets*, 2016.

[90] A. K. Simpson, A. Szekeres, J. Nelson, and I. Zhang, "Securing RDMA for High-Performance Datacenter Storage Systems," in *USENIX HotCloud*, 2020.

[91] A. Singhvi, A. Akella, D. Gibson, T. F. Wenisch, M. Wong-Chan, S. Clark *et al.*, "1rma: Re-envisioning remote memory access for multi-tenant datacenters," in *ACM SGICOMM*, 2020.

[92] J. Snyder, A. R. Lebeck, and D. Zhuo, "RDMA Congestion Control: It's Only for the Compliant," in *IEEE CLOUD*, 2021.

[93] A. Studer and A. Perrig, "The coremelt attack," in *Springer ESORICS*, 2009.

[94] K. Taranov, B. Rothenberger, A. Perrig, and T. Hoefler, "sRDMA – Efficient NIC-based Authentication and Encryption for Remote Direct Memory Access," in *USENIX ATC*, 2020.

[95] R. Teixeira, R. Harrison, A. Gupta, and J. Rexford, "PacketScope: Monitoring the Packet Lifecycle Inside a Switch," in *ACM SOSR*, 2020.

[96] S.-Y. Tsai, M. Payer, and Y. Zhang, "Pythia: Remote Oracles for the Masses," in *USENIX Security*, 2019.

[97] S. Wang, "LoRDMA," https://github.com/wangshicheng1225/LoRDMA, 2023.

[98] S. Wang, K. Gao, K. Qian, D. Li, R. Miao *et al.*, "Predictable vfabric on informative data plane," in *ACM SIGCOMM*, 2022.

[99] W. Wang, X. C. Wu, P. Tammana, A. Chen, and T. S. E. Ng, "Closed-loop network performance monitoring and diagnosis with SpiderMon," in *USENIX NSDI*, 2022.

[100] X. Wei, Z. Dong, R. Chen, and H. Chen, "Deconstructing RDMA-enabled distributed transactions: Hybrid is better!" in *USENIX OSDI*, 2018.

[101] X. Wei, J. Shi, Y. Chen, R. Chen, and H. Chen, "Fast in-memory transaction processing using RDMA and HTM," in *ACM SOSP*, 2015.

[102] J. Xing, W. Wu, and A. Chen, "Ripple: A programmable, decentralized link-flooding defense against adaptive adversaries," in *USENIX Security*, 2021.

[103] J. Xue, Y. Miao, C. Chen, M. Wu, L. Zhang, and L. Zhou, "Fast Distributed Deep Learning over RDMA," in *EuroSys*, 2019.

[104] L. Xue, X. Ma, X. Luo, E. W. W. Chan, T. T. N. Miu, and G. Gu, "LinkScope: Toward Detecting Target Link Flooding Attacks," *IEEE TIFS*, vol. 13, no. 10, 2018.

[105] J. Yang, J. Izraelevitz, and S. Swanson, "Orion: A Distributed File System for Non-Volatile Main Memory and RDMA-Capable Networks," in *USENIX FAST*, 2019.

[106] L. Yu, J. Sonchack, and V. Liu, "Cebinae: Scalable in-network fairness augmentation," in *ACM SIGCOMM*, 2022.

[107] Z. Yu, C. Hu, J. Wu, X. Sun, V. Braverman, M. Chowdhury, Z. Liu, and X. Jin, "Programmable packet scheduling with a single queue," in *ACM SIGCOMM*, 2021.

[108] T. Zhang, R. Shu, Z. Shan, and F. Ren, "Distributed bottleneck-aware coflow scheduling in data centers," *IEEE TPDS*, vol. 30, no. 7, 2019.

[109] Y. Zhang, Y. Liu, Q. Meng, and F. Ren, "Congestion detection in lossless networks," in *ACM SIGCOMM*, 2021.

[110] J. Zheng, Q. Li, G. Gu, J. Cao, D. K. Y. Yau, and J. Wu, "Realtime DDoS Defense Using COTS SDN Switches via Adaptive Correlation Analysis," *IEEE TIFS*, vol. 13, no. 7, 2018.

[111] H. Zhou, S. Hong, Y. Liu, X. Luo, W. Li, and G. Gu, "Mew: Enabling large-scale and dynamic link-flooding defenses on programmable switches," in *IEEE S&P*, 2023.

[112] Y. Zhou, J. Bi, T. Yang, K. Gao, J. Cao, D. Zhang, Y. Wang, and C. Zhang, "Hypersight: Towards scalable, high-coverage, and dynamic network monitoring queries," *IEEE JSAC*, 2020.

[113] Y. Zhou, C. Sun, H. H. Liu, R. , S. Bai, B. Li, Z. Zheng, L. Zhu, Z. Shen, Y. Xi, P. Zhang, D. Cai, M. Zhang, and M. Xu, "Flow Event Telemetry on Programmable Data Plane," in *ACM SIGCOMM*, 2020.

[114] Y. Zhou, D. Zhang, K. Gao, C. Sun, J. Cao, Y. Wang, M. Xu, and J. Wu, "Newton: Intent-Driven Network Traffic Monitoring," in *ACM CoNEXT*, 2020.

[115] Y. Zhu, "NS-3 simulator for RDMA over Converged Ethernet v2 (RoCEv2)," https://github.com/bobzhuyb/ns3-rdma, 2016.

[116] 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," in *ACM SIGCOMM*, 2015.

[117] Y. Zhu, N. Kang, J. Cao, A. Greenberg, G. Lu, R. Mahajan, D. Maltz, L. Yuan, M. Zhang, B. Y. Zhao, and H. Zheng, "Packet-Level Telemetry in Large Datacenter Networks," in *ACM SIGCOMM*, 2015.

Appendix

A. Variable Table

Variables used in the analysis of performance loss caused by the LoRDMA attack are listed in Table I.

B. Coordination Evaluation Setup

Figure 12 shows the topologies for our experiments to evaluate the coordination procedure (§ V-A). These network

TABLE I. VARIABLE TABLE.

Variable	Description
δ	Burst peak rate
τ	Burst duration
T	Burst period
$R(t)$	Flow rate at time t
R_0	Original converged flow rate before bursts
R_{lo}	Lowest flow rate during bursts (usually 0)
PL	Performance loss on a flow, defined as $PL = \int(R_0 - R(t))\mathrm{d}t$
PL_d	Performance loss on a direct vicitm flow
PL_i	Performance loss on an indirect vicitm flow
ΔR	Performance loss factor, defined as $R_0 - R_{lo}$
Norm ΔR	Normalized performance loss factor, defined as $\Delta R / R_0$
(Norm) ΔR_d	(Normalized) performance loss factor on a direct flow
(Norm) ΔR_i	(Normalized) performance loss factor on an indirect flow

topologies are selected from topology zoo [47].

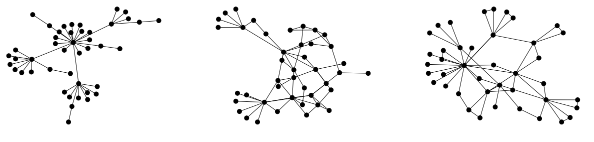

(a) Carnet topology. (b) Switch topology. (c) Cernet topology.

Fig. 12. Topologies used in coordination evaluation.

C. Network Parameters in Simulations

Table II illustrates the parameters of PFC and DCQCN in our ns-3 simulations. We set these parameters based on the values recommended by the state of the art [33], [116], [115].

TABLE II. PARAMETERS FOR DCQCN & PFC.

Parameter	Value
K_{max}	400KB
K_{min}	100KB
P_{max}	1.0
Switch buffer size	6MB
PFC dynamic threshold α	16

D. Topology in Schedule Evaluation

Figure 13 demonstrates the multi-rack topology in our schedule performance evaluation, which is derived from Fat-tree topologies [1], [14], [52]. We set at most 1 bot under each ToR switch (SW1-4, SW6-9), and set up a target flow distribution similar to our motivating experiments (Figure 1(a)), with different background traffic volumes.

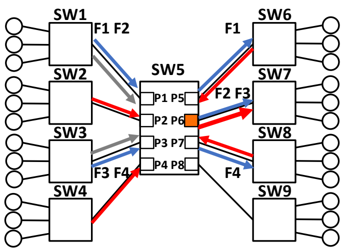


Fig. 13. Multi-rack topology.

E. Topology of the Kuaishou Cloud RDMA Cluster

Figure 14 shows the topology of the Kuaishou cloud RDMA cluster. We set `NCCL Tests` between H1, H5, and H6 as the victims, and craft burst traffic from H4, H7, and H8 to H2 to conduct the LoRDMA attack.

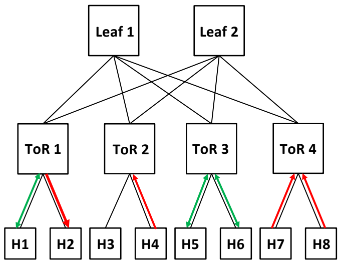
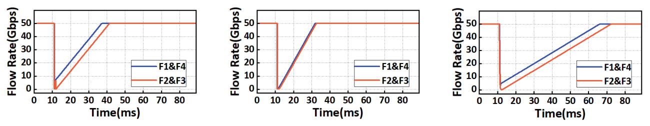


Fig. 14. Topology of the Kuaishou cloud RDMA cluster.

F. Validation on Congestion Control Variants

We carry out the motivating evaluation in § III with different congestion control variants to show the generality of the vulnerability inspiring the LoRDMA attacks. The experimental setup is similar to § III-A. The topology is the same with Figure 1(a), and we deploy 3 typical congestion control algorithms: QCN [32], TIMELY [68], and DCQCN. As shown in Figure 15, the rate of indirect victim flows (F1&F4) are degraded by the bursts in different congestion control algorithms, demonstrating that congestion control can be broadly misled by PFC.



(a) QCN. (b) TIMELY. (c) DCQCN.

Fig. 15. Validations in congestion control variants.