Noisy Neighbor: Exploiting RDMA for Resource Exhaustion Attacks in Containerized Clouds

Gunwoo Kim[1], Taejune Park[2], and Jinwoo Kim[1]*

[1] Kwangwoon University, Seoul, Republic of Korea
[2] Chonnam National University, Gwangju, Republic of Korea

Abstract. In modern containerized cloud environments, the adoption of RDMA (Remote Direct Memory Access) has expanded to reduce CPU overhead and enable high-performance data exchange. Achieving this requires strong performance isolation to ensure that one container's RDMA workload does not degrade the performance of others, thereby maintaining critical security assurances. However, existing isolation techniques are difficult to apply effectively due to the complexity of microarchitectural resource management within RDMA NICs (RNICs). This paper experimentally analyzes two types of resource exhaustion attacks on NVIDIA BlueField-3: (i) state saturation attacks and (ii) pipeline saturation attacks. Our results show that state saturation attacks can cause up to a 93.9% loss in bandwidth, a 1,117× increase in latency, and a 115% rise in cache misses for victim containers, while pipeline saturation attacks lead to severe link-level congestion and significant amplification, where small verb requests result in disproportionately high resource consumption. To mitigate these threats and restore predictable security assurances, we propose HT-Verbs, a threshold-driven framework based on real-time per-container RDMA verb telemetry and adaptive resource classification that partitions RNIC resources into hot, warm, and cold tiers and throttles abusive workloads without requiring hardware modifications.

Keywords: RDMA (Remote Direct Memory Access) · Cloud Security · Containerized Clouds · Resource Exhaustion Attacks

1 Introduction

RDMA (Remote Direct Memory Access) is a high-performance technology that enables direct memory access between systems without involving the host CPU or operating system kernel, significantly reducing latency and CPU overhead while delivering high throughput [11]. RDMA communicates through RDMA NICs (RNICs), such as the NVIDIA BlueField-3, which are designed with programmable cores integrated into the host NIC system on chip. These cores are directly connected to data center Ethernet or InfiniBand links, allowing RNICs to process packets on-path and respond without involving the host OS networking stack, thus reducing latency for certain applications [12,17]. Based on these

* Corresponding author

advantages, RNICs are widely adopted in recent cloud environments, including distributed machine learning frameworks and high performance storage systems, where low-latency and high throughput communication are critical [29,20].

Despite these advantages, RNICs face a significant challenge in cloud environments: the *performance isolation* problem. This issue arises when a malicious tenant exhausts RNIC microarchitecture resources (e.g., NIC caches, processing units) by generating abusive RDMA workloads. Specifically, an attacker can overload the RNIC through carefully crafted RDMA operations, resulting in increased latency, elevated cache miss rates, and, in severe cases, denial of service (DoS) due to internal resource starvation. Such resource exhaustion disrupts critical NIC components and degrades the performance of co-located tenants by creating system-wide bottlenecks, *thus undermining the security assurance guarantees that multi-tenant clouds depend on to ensure availability and predictable performance.*

The performance isolation issue in RDMA has been explored in several prior studies [15,22,18,13,10]. However, most of these works focus on virtual machine or bare-metal environments, limiting their applicability to containerized architectures. With the shift in cloud infrastructure toward cloud-native architectures based on containers, there is growing interest in adapting RDMA to these environments [14,19,21,27]. For example, NVIDIA, a leading RNIC vendor, provides an RDMA plugin for Kubernetes [2], enabling seamless integration of RDMA capabilities within containers. Furthermore, emerging applications such as distributed deep learning frameworks and data-intensive analytics pipelines increasingly leverage RDMA to achieve low-latency, high-throughput communication in containerized setups [17,12]. Despite these advancements, the impact of resource exhaustion in container environments remains largely unexplored.

In this paper, we systematically investigate the impact of resource exhaustion attacks on the performance of legitimate containers in a containerized cloud environment. We focus on how a malicious container can drain microarchitectural resources of a shared RNIC in RDMA-enabled settings, significantly degrading the performance of co-resident victim containers. To this end, we propose two classes of attacks: (i) *state saturation attacks*, in which the attacker floods RNIC resources such as queue pairs and caches; and (ii) *pipeline saturation attacks*, in which the attacker abuses RDMA verbs to overload the communication pipeline or induce amplified load on the RNIC.

We conduct experiments on NVIDIA BlueField-3, a widely used RNIC, within RDMA-enabled container environments built using Docker and SR-IOV. Our results demonstrate that excessive RDMA operations from an attacker container can cause severe performance degradation in co-located containers. In state saturation attacks, the victim container experiences a bandwidth drop of approximately 93.9%, a latency increase of up to 1,117×, and a 115% rise in cache miss rates. Pipeline saturation attacks generate over 20,000 PAUSE frames, indicating link-level congestion, and result in substantial amplification—e.g., 23.1 bytes received by the RNIC per 8-byte request—showing that the attacker consumes disproportionately more resources than the traffic it

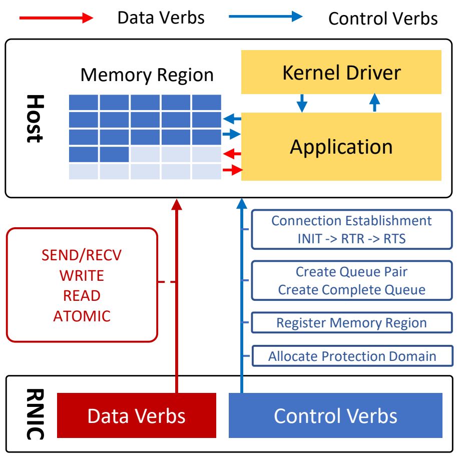

Fig. 1: Overview of RDMA workflow.

injects. To mitigate these issues, we propose HT-Verbs, a defense system that enforces fair resource usage and ensures predictable performance in containerized RDMA-enabled environments.

Contributions. We summarize our main contributions as follows:

- We present novel resource exhaustion attacks targeting RDMA-enabled container environments, demonstrating how microarchitectural resource contention within RNICs can violate performance isolation guarantees and undermine critical security assurances in multi-tenant clouds.
- Through extensive experiments on NVIDIA BlueField-3, we show that our attacks can severely degrade victim container performance, resulting in up to a 93.9% loss in bandwidth, a 1,117× increase in latency, and a 115% rise in cache miss rates, thereby quantitatively exposing vulnerabilities that threaten runtime assurance.
- We propose HT-Verbs, a conceptual RNIC-based defense mechanism that mitigates these attacks by dynamically monitoring and classifying per-container RDMA resource usage. While a full implementation is left to future work, our design demonstrates the feasibility of enforcing hardware-level resource isolation to restore predictable performance and strengthen security assurances in multi-tenant container environments.

2 Background and Motivation

2.1 Remote Direct Memory Access (RDMA)

Fig. 1 illustrates the overall RDMA workflow. RDMA communication is facilitated through an API called *verbs*, which are categorized into *control verbs*

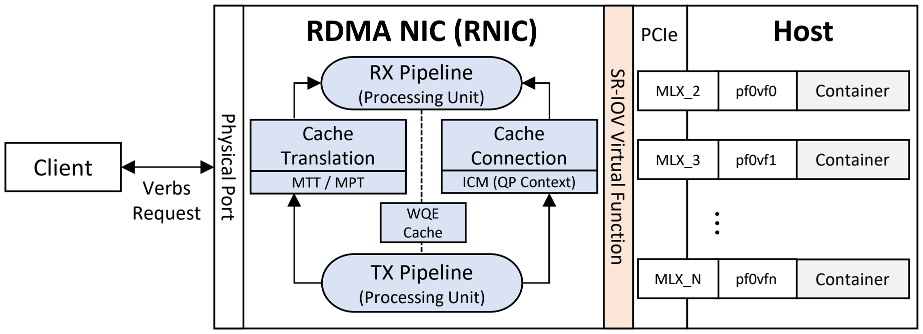

Fig. 2: Architecture of an RNIC and an example of SR-IOV-based virtualization to serve multiple containers.

and *data verbs*. Control verbs manage the initialization process by creating and configuring key resources, such as queue pairs (QPs) and completion queues (CQs). Subsequently, the application allocates memory in the host's DRAM, maps virtual addresses to physical ones, and registers these memory regions with the RNIC (RDMA NIC). This registration enables the RNIC to directly access memory without CPU intervention, thereby reducing latency and improving throughput.

Data verbs, such as WRITE and SEND, are used to actually move bytes between endpoints. Once initialization is complete, an application can leverage data verbs to transfer data between local and remote memory. Data verbs fall into two categories: (i) two-sided verbs and (ii) one-sided verbs. In the former, both the sender and the receiver must post work requests. For example, a SEND operation places the payload into a receive buffer that the remote CPU has already advertised. In the latter, after capability negotiation, the initiator can read from, write to, or perform atomic operations on the peer's memory without further involvement of the remote CPU.

Beyond individual verbs, the transport type itself influences scalability. RDMA defines two transport modes: (i) Reliable Connected (RC) and (ii) Unreliable Connected (UC). RC provides reliable, one-to-one communication, similar to TCP, between exactly one local QP and one remote QP. It supports all one-sided verbs and offers full end-to-end reliability, but each additional peer requires an extra QP. In large clusters, the RNIC's QP capacity becomes a limiting factor. In contrast, UC allows a single QP to communicate with multiple peers, similar to UDP, greatly reducing the number of QPs a thread must maintain and thereby improving scalability. The trade-off is that UC omits support for one-sided operations and end-to-end reliability.

2.2 RDMA-enabled Container Environment

In modern cloud-native infrastructures, containers frequently require direct access to high-performance networking capabilities such as RDMA [14]. To support

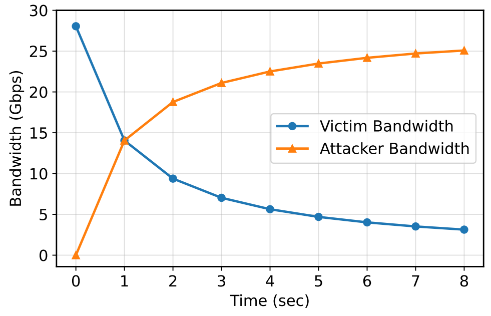

Fig. 3: Example of a performance isolation issue where an attacker exhausts the victim's bandwidth by increasing the number of its own QPs.

this, a single physical RNIC can be virtualized through SR-IOV (Single Root I/O Virtualization), which provides one Physical Function (PF) for global management and multiple Virtual Functions (VFs), each maintaining its own hardware context (e.g., queue pair and memory translation state). Fig. 2 shows an example of an RNIC deployment in containerized environments. When VFs (e.g., `pf0vf*`) are created, they are automatically exposed to the host system as distinct PCIe devices (e.g., `mlx5_*`), reflecting their instantiation as independent virtual ports within the RNIC. By assigning a VF to a container's network namespace, containers can achieve near-native, low-latency access to the virtualized RNIC.

A physical RNIC consists of various microarchitecture resources [15], each dedicated to specific metadata storage and fast access, as shown in Fig. 2. For instance, in the case of the NVIDIA BlueField-3, the Memory Translation Table (MTT) handles virtual-to-physical address translation, while the Memory Protection Table (MPT) enforces memory access permissions. The Interconnect Connect Memory (ICM) cache stores QP state and connection metadata to manage RDMA operations. Additionally, the WQE (Work Queue Entry) cache holds pre-fetched entries from transmission and reception queues, enabling rapid processing in the TX/RX pipeline. Transmission requests are routed to the user side through the TX pipeline, while reception requests are delivered to the container's memory via the RX pipeline.

2.3 Motivating Example

Performance isolation refers to the ability to ensure that one tenant's activities do not degrade the performance of others sharing the same physical resources [15]. As discussed earlier, multiple containers in a containerized cloud share a single RNIC's internal resources. We hypothesize that this setup may inherently suffer from performance isolation issues. To validate this, we conducted an experiment in which two containers—a victim and an attacker—ran concurrently on the same host, each assigned a separate VF on an NVIDIA BlueField-3.

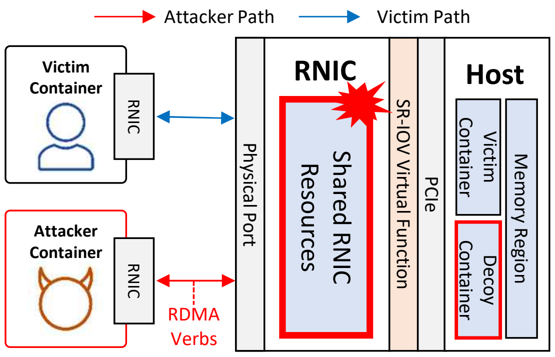

Fig. 4: Our threat model, where an *attacker container* compromises the performance isolation of a target host by leveraging a co-resident *decoy container* to launch RDMA-based resource exhaustion attacks against *victim containers*.

The victim container continuously executed the `ib_write_bw` benchmark to establish baseline throughput, while the attacker container gradually increased its number of QPs from 1 to 8, issuing high-rate WRITE verbs. At each step, we measured the victim's sustained bandwidth.

As illustrated in Fig. 3, increasing the attacker's number of QPs leads to resource saturation within the RNIC—specifically in the TX/RX pipelines and the WQE cache—causing the victim's bandwidth to drop by up to 93.9%. Interestingly, the total bandwidth was observed to be inversely proportional to the number of QPs created by the attacker, resembling the behavior of fair-share scheduling. Since there is no control over NIC resources in the current RDMA-enabled container environment, these findings reveal that virtualized RNICs in containerized environments are highly susceptible to performance isolation breakdowns, motivating a deeper investigation into RDMA-level attack vectors targeting this vulnerability.

3 Resource Exhaustion in RDMA-enabled Containers

3.1 Threat Model

We consider a multi-tenant environment in an on-premises cloud where tenants share the same physical machine. To provide tenants with high-performance applications, cloud administrators may deploy RDMA-enabled containers. This setup is feasible given that Kubernetes supports the SR-IOV plugin [2], which allows containers to be assigned their own VFs to access RNICs. Accordingly, we assume that the cloud administrator adopts this type of Kubernetes deployment. In this context, we consider an *attacker container* initially residing on a different physical machine from the target host. Fig. 4 shows our threat model.

The attacker's goal is to induce performance isolation issues on the target host via malicious RDMA operations, thereby disrupting the operations of one

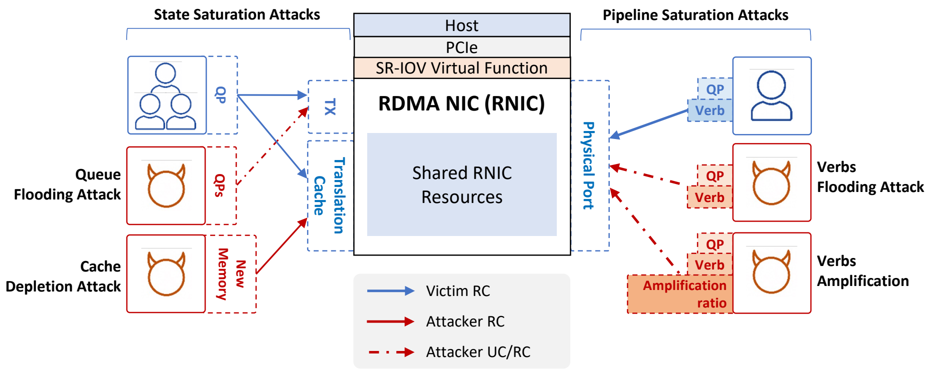

Fig. 5: Overview of resource exhaustion attacks in an RDMA-enabled container environment.

or more *victim containers*. To achieve this, the attacker requires a *decoy container* co-residing on the target host. Such co-residency can be achieved through brute-force attacks by repeatedly creating containers combined with co-residency checks, as demonstrated in prior VM-based attacks [24], and more recently confirmed feasible in container environments [26]. Both the attacker container and the decoy container are assumed to have access to their own VFs assigned via SR-IOV. We further assume that they share the same L3 domain, enabling communication over RoCEv2 (RDMA over Converged Ethernet version 2). Thus, the attacker container can establish a QP with the decoy container, invoke RDMA verbs, and freely choose the transport mode, such as RC or UC (Section 2.1). Note that we do not consider RDMA protocol-related vulnerabilities [25], and thus assume that the attacker fully complies with the RDMA protocol and standard operations.

3.2 Attack Scenarios

Our goal is to develop practical attacks that an *attacker container*, in coordination with a *decoy container*, can launch to disrupt performance isolation on the target host by exploiting RDMA operations. To this end, we introduce two types

Table 1: Summary of RDMA resource exhaustion attacks, their target, verbs, and modes.

Type	Attack	Target	Verbs	Modes
State Saturation Attacks	Queue Flooding	QPs	SEND, RECV	UC, RC
	Cache Depletion	Translation/Connection Cache	READ, WRITE	RC
Pipeline Saturation Attacks	Verbs Flooding	WQE, TX/RX pipeline	ALL	UC, RC
	Verbs Amplification	QP, TX/RX pipeline	ALL	UC, RC

of attacks: (i) *state saturation attacks* and (ii) *pipeline saturation attacks*. Fig. 5 provides an overview of these attacks, and Table 1 summarizes their targets, RDMA verbs, and operational modes.

3.3 State Saturation Attacks

In these types of attacks, the attacker aims to exhaust RNIC resources by aggressively consuming connection management and caching resources involved in RDMA communication.

Queue Flooding. The attacker can rapidly instantiate a large number of queue pairs (QPs) and corresponding completion queues (CQs), similar to a TCP SYN flooding attack. Each new QP allocates on-chip resources (QP state, work queue buffers), and the flood of posted work requests saturates the TX/RX pipelines, evicts legitimate QP state from the connection cache, and exhausts WQE cache entries. This attack can be launched in both UC and RC modes. In UC mode, the attacker can cycle large batches of SEND/RECV requests on a single QP without requiring collaboration from the decoy container. In contrast, RC mode requires a dedicated pair of QPs between the attacker and decoy containers, making the creation of many QPs more critical—despite the need for decoy container collaboration.

Cache Depletion. Next, the attacker can issue one-sided READ or WRITE operations with continuously increasing remote addresses across its QPs, flushing the translation cache and connection cache, and driving up cache miss rates on the shared RNIC. The resulting high miss rates introduce delays in the TX/RX pipelines and degrade the performance of co-located victim containers. If a victim container and a decoy container share cache resources on the RNIC, this attack can severely impact system throughput and significantly increase latency for time-sensitive RDMA workloads on the victim container. Notably, the attacker can achieve similar cache-thrashing effects using a single QP in UC mode by flooding new address operations.

3.4 Pipeline Saturation Attacks

In this type of attack, the attacker overwhelms the RDMA processing pipeline by flooding RDMA operations, leading to resource contention and performance degradation.

Verbs Flooding. The attacker can flood RDMA verbs in a manner similar to HTTP or UDP flooding attacks. To do so, the attacker injects verbs at the maximum throughput for each QP and amplifies the impact by increasing the number of QPs filled with verbs. Specifically, in RC mode, the attacker may leverage both one-sided and two-sided verbs across multiple QPs. Each verb invocation allocates per-QP context and triggers reliability handshakes, stressing the atomic engine and TX/RX pipelines, evicting WQE entries and translation cache state, and causing pipeline stalls and context-switch thrashing. In contrast, in UC mode, the attacker uses a single QP to send large bursts of WRITE/SEND messages without acknowledgments; the absence of reliability overhead enables higher injection rates. Consequently, stress on the WQE FIFO and TX/RX pipelines increases rapidly, leading to PCIe back-pressure.

Verbs Amplification. The attacker can design a more sophisticated amplification attack, in which a small RDMA verb request triggers significant resource consumption on the RNIC, similar to traditional DDoS amplification attacks [16]. Specifically, amplification occurs if the RNIC processes more bytes than the payload size of the issued verb. We observe that such amplification arises from additional overhead introduced by protocol headers, control traffic, and flow-control events. Thus, the attacker can exploit this asymmetry to efficiently overload the RNIC while consuming fewer resources.

4 Evaluation

In this section, we evaluate the effectiveness of our proposed attacks.

4.1 Experimental Environment

To assess the impact of the proposed resource exhaustion attacks, we set up an experimental testbed comprising two servers, each configured with an Intel Core i7-13700K CPU (16 cores @ 3.40 GHz), 64 GB of RAM, and an NVIDIA BlueField-3 RNIC [7]. We then replicated an RDMA-enabled container environment with Docker containers, each configured to access one of these VFs (Virtual Functions), ensuring direct, near-native access to the RNIC's resources. Specifically, on each RNIC, SR-IOV was enabled to instantiate two VFs via `echo 2 > /sys/class/net/p0/device/sriov_numvfs`, and the `pipework` utility [23] was employed to bind each VF into a separate Docker container network namespace. Attacker containers were designed to launch the proposed resource exhaustion attacks, while victim containers performed normal RDMA operations and were monitored to evaluate the impact of the attacks on their performance. RDMA communication between containers was established over RoCE (RDMA over Converged Ethernet) v2. The attacker containers were programmed to generate excessive RDMA operations to overload the RNIC, while the victim containers followed typical RDMA workload patterns to simulate real-world application behavior.

We measured bandwidth and latency using two benchmark tools: `ib_write_bw` and `ib_read_lat` from the `perftest` package [1], while Linux `perf` was used to record cache-miss ratios. Additionally, BlueField-3 hardware counters along with its PCIe observability module, were used to quantify per-verb pipeline pressure, enabling to correlate internal RNIC resource contention with the observed bandwidth and latency degradation. Link-level statistics were also collected from the RNIC using the Linux `ethtool -S` command, monitoring hardware counters, such as `port_rcv_data` and PAUSE frames.

We began by measuring baseline performance—bandwidth, latency, hardware counters, and cache statistics—in the absence of any attacks to establish

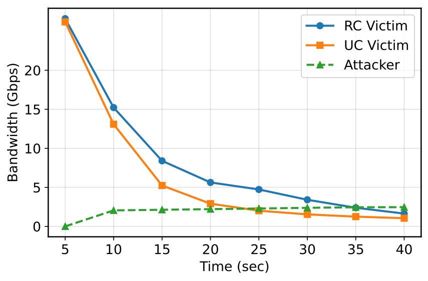

Fig. 6: Bandwidth changes in a *single-victim* environment during a queue flooding attack.

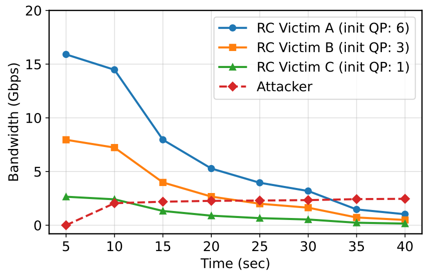

Fig. 7: Bandwidth changes in a *multi-victim* environment during a queue flooding attack.

normal operating metrics for the victim containers. Next, we sequentially executed the queue flooding, cache depletion, and verbs flooding scenarios from the adversary containers, continuously recording the victim's performance throughout each test. To ensure fair comparison and reproducibility, we kept key variables—such as the total number of containers, RDMA operation types, and data sizes—constant across all experimental runs. This approach minimized external factors influencing the results and ensured reproducibility.

4.2 Effectiveness of Resource Exhaustion Attacks

Impact of Queue Flooding. We evaluated the impact of the queue flooding attack by measuring bandwidth degradation experienced by a victim container under both RC and UC modes. Fig. 6 shows the bandwidth over time for both UC/RC victim and attacker containers during the attack. Under RC mode, the victim's bandwidth dropped from 26.61Gbps to 1.64Gbps, marking a 93.9% reduction. Under UC mode, a similar degradation trend was observed compared to RC. In contrast, the attacker's bandwidth remained consistently below 5Gbps. These results demonstrate that even lightweight RDMA verbs with minimal data payloads can saturate critical shared RNIC resources, such as TX and RX processing pipelines and WQE cache entries, severely degrading victim performance. This highlights that control intensive verbs, despite their low data volume, are sufficient to monopolize RNIC resources.

To validate the generalizability of this phenomenon in multi-tenant environments, we conducted an additional experiment with three distinct victim containers, each configured with different initial QP counts (6, 3, and 1, respectively). Fig. 7 presents the bandwidth changes over time. Initially, Victim A achieved 15.9Gbps, Victim B achieved 7.96Gbps, and Victim C achieved 2.66Gbps. As the flooding continued for 40 seconds, all three victims suffered bandwidth drops of over 90%. By the end of the 40-second test, the bandwidths fell to 1.02Gbps for Victim A, 0.49Gbps for Victim B, and 0.15 Gbps for Victim C. These results show that queue flooding attacks remain effective even in multi-victim scenarios and demonstrate the need for microarchitectural-level protection mechanisms.

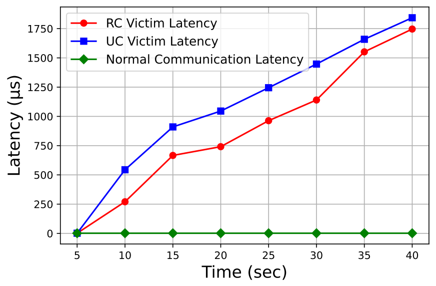

Fig. 8: Latency changes of a victim during a *cache depletion attack*.

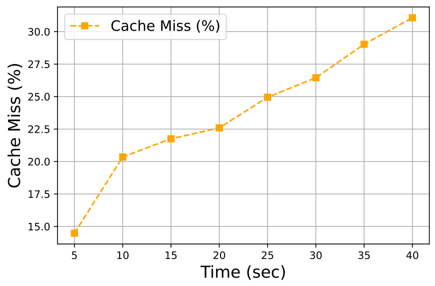

Fig. 9: Cache miss rates of a victim during a *cache depletion attack*.

Impact of Cache Depletion. We verified the impact of a cache depletion attack on the victim container by measuring its communication latency and cache miss rates. Under normal conditions, the victim container exhibited an average latency of 1.56 μs, as shown in Fig. 8. During the attack, as the attacker container continuously accessed new memory locations, the victim's latency sharply increased. By the end of the 40-second test, the latency had risen to 1,746.34 μs, representing approximately a 1,117-fold increase. The cache miss rate also showed a significant rise during the attack, as illustrated in Fig. 9. Initially, the cache miss rate was 14.48%, but it steadily increased throughout the experiment, reaching 31.07% by the end. These results demonstrate that the cache depletion attack severely degraded the RNIC's cache performance, significantly impacting legitimate RDMA operations.

4.3 Effectiveness of Pipeline Saturation Attacks

Impact of Verbs Flooding. To verify the effectiveness of verbs flooding, we measure the number of PAUSE frames observed during the test. PAUSE frames are flow-control packets that an RNIC sends to its link partner to pause transmission when its RX pipeline or ingress buffers near exhaustion. By counting the number of PAUSE frames, we quantify the RNIC's link-level congestion during attacks similar to the existing work [28]. Fig. 10 shows the measured number of PAUSE frames when the attacker's QP count varies from 1 to 24. In RC mode, WRITE verb flooding generates no PAUSE frames at a single QP but escalates rapidly beyond four QPs, reaching a maximum of 42,535 frames at 24 QPs. In contrast, READ verb never triggers PAUSE frames, while SEND verb incurs only modest overhead, rising from 3 frames at one QP to about 40,779 frames at 24 QPs. In UC mode, even a single QP of WRITE or SEND verbs provoke over 22,000 PAUSE frames, climbing to 44,440 and 40,535 frames, respectively as the QP count increases.

Impact of Verbs Amplification. To evaluate the effectiveness of verbs amplification, it is essential to measure how much advantage (i.e., resource consumption) the attacker can gain from sending verb requests. To this end, we define the *amplification ratio (AR)*, which represents the degree to which an RDMA

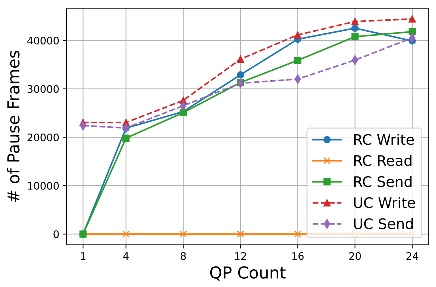

Fig. 10: Measured number of PAUSE frames during *verbs flooding attacks.*

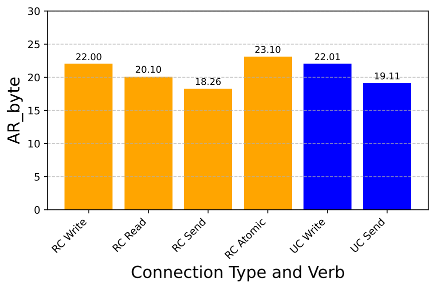

Fig. 11: Measured amplification ratio during *verbs amplification attacks.*

verb v induces load on RNIC resources. Specifically, we define $AR_{\text{byte}}(v)$, which quantifies the number of bytes the RNIC actually receives per byte of a request verb v:

$$AR_{\text{byte}}(v) = \frac{R_v}{L_v},$$

where R_v denotes the number of bytes received by the RNIC, and L_v indicates the payload size in bytes of the verb v. In our experiments, we set $L_v = 8$ bytes and conduct 30-second amplification attacks to measure $AR_{\text{byte}}(v)$. Here, a high AR_{byte} value indicates that even small RDMA verbs can impose significant internal load on the RNIC. This overhead results not only from protocol headers and flow control metadata, but also from microarchitectural operations such as queue scheduling, DMA transfers, and cache accesses. As a result, the attacker can consume considerable RNIC resources while keeping bandwidth usage low. This is conceptually similar to amplification attacks like DNS [9], where a small DNS request (e.g., 60 bytes) can trigger responses up to 4,000 bytes, yielding amplification factors exceeding 60×. Likewise, AR_{byte} reflects how efficiently an attacker can amplify internal RNIC resource usage per byte of issued RDMA verbs, even without increasing overall traffic volume.

Fig. 11 illustrates the measured AR_{byte} for each RDMA verb under two different transport modes (i.e., RC and UC). In RC mode, the ATOMIC verb yields the highest amplification ratio (23.1), meaning that the RNIC receives 23.1 bytes for every 8 bytes generated by the attacker, followed by WRITE (22.01), READ (20.1), and SEND (18.26). This result indicates the increasing header and control overhead associated with each verb: ATOMIC incurs both request and response metadata; WRITE adds an extended RDMA header on top of the base transport header; and READ/SEND include only the minimal base header with lightweight control handshakes. The near-parity between RC and UC modes suggests that most of the overhead—such as large header processing, internal metadata handling, and DMA interactions—is imposed by the RNIC microarchitecture itself, regardless of end-to-end reliability. Furthermore, the pronounced gap between one-sided and two-sided verbs corresponds to their protocol complexity: two-sided verbs require additional remote DMA setup metadata, while one-sided verbs rely only on basic network headers and queue handshakes, resulting in lower per-byte amplification.

5 Discussion and Mitigation

In this section, we discuss existing solutions and propose potential approaches to detect and mitigate the effects of resource exhaustion attacks targeting RNIC resources in container environments.

5.1 Root Causes

The feasibility of RDMA-based resource contention attacks in containerized environments stems from the fact that, under SR-IOV virtualization, all containers share the same set of low-level RNIC microarchitectural resources—such as the Memory Translation Table (MTT), Memory Protection Table (MPT), Connection Cache, WQE cache, and the TX/RX processing pipelines—without any fine-grained partitioning or per-container enforcement. Since SR-IOV assigns only logical VFs to containers, these VFs still contend for on-chip buffers and caches in a first-come, first-served manner. As a result, an attacker that aggressively issues RDMA verbs can evict or throttle legitimate work queue entries and connection state belonging to co-resident victim containers.

Commodity RNICs lack any fine-grained fairness or abuse detection, employing only best-effort or round-robin scheduling across QPs. An attacker can therefore inflate QP counts or flood RDMA verbs to overwhelm the TX/RX pipelines and translation caches driving up cache misses and PCIe back-pressure such as PAUSE frames. While the absence of per-verb rate limiting means sustained verb floods automatically amplify internal load with minimal effort. Because RDMA bypasses the host OS networking stack entirely, container runtimes and OS-level QoS controls cannot observe or throttle these flows. This combination of shared microarchitectural contexts, no scheduling fairness, amplification effects, and kernel bypass creates ideal conditions for devastating resource exhaustion attacks in containerized clouds.

5.2 Existing Countermeasures

Hardware-based Solution. Quality of Service (QoS) mechanism [3] supported by BlueField-3's enables network flow prioritization and resource management by mapping user-defined priorities to traffic classes (TCs). This is achieved through a multi-level process that ensures bandwidth guarantees, restrictions, or prioritized access to network resources. For example, the bandwidth speed for ports allocated to tenants can be limited using commands such as: `echo 1000 > /sys/class/net/p0/{BlueField}/{pf,vf}/{min,max_tx_rate}`. This configures the maximum and minimum transmission speeds for PFs and VFs, ensuring fair resource allocation among tenants. Additionally, tools such as `mlxdevm` and

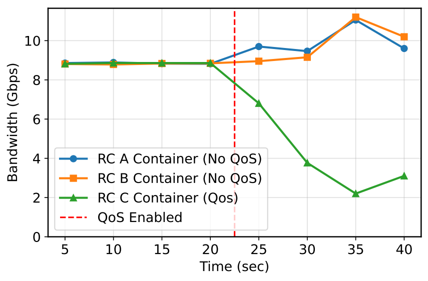

Fig. 12: Impact of QoS enforcement on RDMA bandwidth distribution among containers.

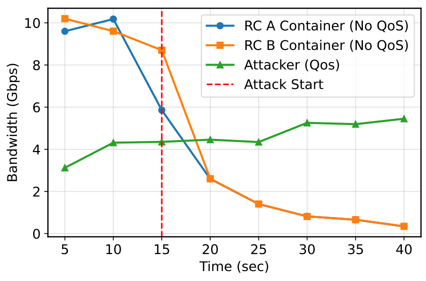

Fig. 13: (In)effectiveness of the QoS mechanism in limiting the attacker container during a *queue flooding attack*.

`evlink` enable per-port or per-group bandwidth settings. Furthermore, the Enhanced Transmission Selection (ETS) [3] mechanism enables dynamic bandwidth sharing, maintaining overall system performance by allocating unused bandwidth effectively.

On the other hand, `VirtIO-net` [4] enables static QP provisioning per VF, offering hardware-level configuration flexibility, it was originally designed for QEMU-based virtual machine environments. Consequently, it operates at the PCIe device level and lacks native support for dynamic attach/detach and fine-grained resource isolation, making it ill-suited for modern container environments such as Docker. This architectural constraint limits its usability in multi-tenant containerized systems, where per-container flexibility and strict performance isolation are required.

While these all mechanisms are effective for managing network resources at the traffic level, they face limitations in addressing performance isolation issues specific to RNICs. For example, NVIDIA's RoCE protocol supports only four predefined ToS (Type of Service) values, limiting the flexibility of priority configurations in complex multi-tenant environments. Consequently, QoS mechanisms often struggle to adapt to dynamically changing network traffic patterns. Moreover, since QoS mechanisms are primarily designed to regulate traffic prioritization, they cannot directly mitigate resource exhaustion attacks targeting RNIC resources such as the TX/RX processing unit or internal caches.

To validate this limitation, we conducted an experiment using three containers, each subject to QoS mechanisms. We configured one container with an Enhanced Transmission Selection (ETS) weight of 10% and a 4Gbps rate limit, then measured the bandwidth distribution before and after enforcing the policy. As shown in Fig. 12, once the QoS policy was applied at 23 seconds, it effectively capped Container C's bandwidth, enabling Containers A and B to opportunistically utilize the freed RNIC capacity. Fig. 13 extends this analysis by applying the same QoS configuration to the attacker container during a queue flooding attack. Despite throttling the attacker at 15 seconds, Containers A and B—without proper isolation—still suffered a 73.4% performance degra-

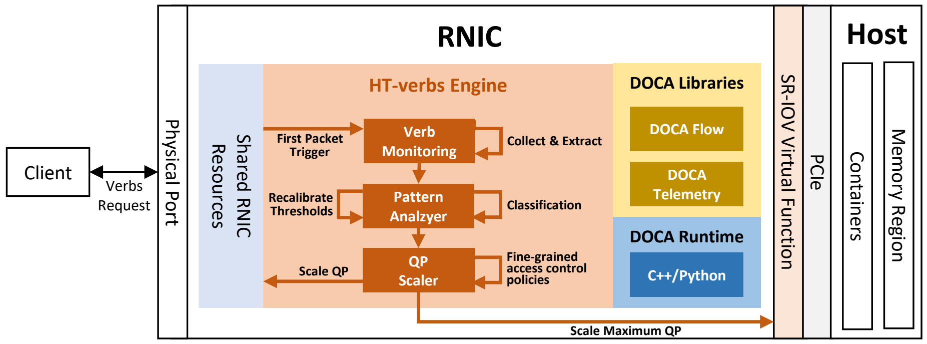

Fig. 14: The conceptual architecture of HT-Verbs.

dation compared to the baseline. These results demonstrate that although ETS can redistribute bandwidth, it does not guarantee minimum service levels in the absence of microarchitectural or verb-level isolation.

Software-based Solution. Several software-based solutions have been proposed to prevent resource exhaustion attacks. Freeflow [14] is a representative system designed to ensure fair resource allocation and predictable performance in RDMA-enabled container environments. Freeflow achieves its goals by creating virtual NICs and virtualized spaces, which allow transparent management of resource movement. By virtualizing the resources and managing them dynamically, Freeflow ensures that containers can access resources fairly without interference. While Freeflow provides a transparent and structured methodology for resource allocation, its resource management approach may not fully address bottlenecks at the microarchitecture level, such as contention in compute or I/O units, which can significantly impact performance in high-demand scenarios. In addition, prior work has shown that such software-based virtualization introduces overheads in the RDMA data path, which counteract RDMA's performance benefits [19].

5.3 HT-Verbs: A Threshold Based Resource Management System

To overcome the limitations of existing resource isolation mechanisms and mitigate resource exhaustion attacks in RDMA-enabled container environments, we propose HT-Verbs, a threshold-based RNIC resource management system. Fig. 14 illustrates its conceptual architecture. HT-Verbs operates on the RNIC to dynamically monitor and classify per-QP resource usage in real-time, enabling the system to detect malicious QPs. Upon detection, HT-Verbs can selectively throttle or block malicious QPs at the SmartNIC level, while Adaptively scaling legitimate QPs to ensure fair and predictable performance across legitimate containers

Resource Monitoring. HT-Verbs periodically collects and aggregates the usage patterns of RDMA Verbs (e.g., READ, WRITE, SEND, RECV, ATOMIC) invoked on each QP. Key metrics, such as QP/CQ creation frequencies, TX/RX pipeline pressure, and cache hit rates, are continuously monitored within the RNIC's shared microarchitectural layer. These metrics are collected and tracked through the *Verb Monitoring* component of HT-Verbs, and the engine further consists of two additional modules—*Pattern Analyzer*, which detects abnormal usage trends per container and analyzes QP usage patterns in accordance with dynamic temperature levels; and *QP Scaler*, which dynamically adjusts resource allocation based on pattern analyzer outcomes. These three components operate in tandem to identify potential resource abuse and ensure fair resource sharing across multi-tenant environments.

Resource Classification. Based on the analysis of usage patterns, HT-Verbs categorizes RNIC resources into three levels: *Hot*, *Warm*, and *Cold*, with each classification representing a different level of resource demand:

- ***Hot*:** These are heavily utilized resources experiencing high contention. Access by lower-priority or malicious containers is restricted to prevent overloading.
- ***Warm*:** These resources have moderate demand, maintaining standard isolation levels. Both legitimate and lower-priority containers can access these resources with minimal restrictions.
- ***Cold*:** These are underutilized resources. Higher-priority containers are granted enhanced performance, exceeding typical isolation guarantees.

This classification is performed by the *Pattern Analyzer* and provided to the *QP Scaler* as input for enforcing appropriate resource controls.

Adaptive Threshold Adjustment. The threshold criteria for classifying resources into *Hot*, *Warm*, and *Cold* are not static. Instead, they are continuously recalibrated by the *Pattern Analyzer* based on system-wide metrics collected through the *Verb Monitoring* module, such as overall RNIC utilization, traffic fluctuation, and per-QP statistics. These adaptive thresholds are used by the *QP Scaler* to enforce dynamic resource control, allowing HT-Verbs to remain responsive to sudden demand spikes or evolving attack behavior while preserving performance stability.

Malicious Container Detection. HT-Verbs does not rely on fixed rules or static signatures to identify malicious containers. Instead, it leverages the *Verb Monitoring* module to track per-QP RDMA usage patterns in real time, which are then evaluated by the *Pattern Analyzer* against system-wide activity distributions. A container is considered anomalous if it consistently deviates from normal behavior—for example, through excessive QP/CQ creation or persistently high TX/RX pipeline pressure without a proportional increase in throughput. These classification results are forwarded to the *QP Scaler*, which selectively restricts or deprioritizes the offending container's resource access. In low-contention scenarios or when only a single container is active, threshold constraints are relaxed to avoid penalizing benign high-performance workloads.

Implementation Strategy. HT-Verbs operates as a middleware layer between the RNIC hardware and containerized applications. It can be implemented atop NVIDIA's BlueField-3 architecture using the DOCA SDK [8], which provides low-level telemetry access and control APIs for RDMA-aware SmartNICs. Specifically, the *Verb Monitoring* module leverages DOCA Telemetry [6] and `mlx5` hardware counters to periodically collect per-VF resource metrics—such as QP/CQ creation rate, TX/RX pipeline occupancy, and cache activity—at fine-grained intervals (e.g., every 100 ms). Container association is achieved using the `pipework` utility [23], which binds VFs to container network namespaces, or alternatively via DOCA's container-awareness hooks. These raw metrics are streamed to the *Pattern Analyzer*, which uses statistical analysis to compute both absolute and differential indicators (e.g., ΔPAUSE frames/sec, cache miss rate). Thresholds are dynamically adjusted based on recent statistical distributions (e.g., 90th percentile = "Hot", 10th percentile = "Cold").

These modules can be implemented using standard Python/C++ bindings atop the DOCA Telemetry library for real-time analysis within the DPU OS. Based on classification results, the *QP Scaler* enforces per-container policies using DOCA Flow APIs [5]. Additionally, it enforces fine-grained access control policies, such as throttling (via WQE pacing), priority adjustment, or QP reallocation. Policy enforcement is performed inline on the DPU to minimize control-plane delay. Throughout this process, HT-Verbs continuously monitors feedback signals such as throughput oscillation and WQE latency. Thresholds and control policies are recalibrated in real time to ensure stability under multi-tenant contention.

6 Related Work

Performance Isolation in RDMA. Performance isolation in RDMA-enabled environments has been extensively explored, primarily focusing on virtual machines and bare-metal systems. Kong et al. [15] conducted a comprehensive analysis showing how RDMA microarchitectural resources—such as RNIC caches, processing units, and PCIe bandwidth—become critical bottlenecks for performance isolation in multi-tenant clouds. They demonstrated that existing isolation techniques fail to mitigate contention at this level and introduced the Husky test suite to systematically evaluate such issues. Similarly, PeRF [18] examined QP-level unfairness in RNICs, revealing how round-robin scheduling can cause throughput collapse and latency spikes for low-QP tenants. Unlike these studies, our work focuses on performance isolation challenges specific to RDMA-enabled container environments.

Solutions for Performance Isolation. To address the performance isolation, many systems have been proposed. Harmonic [22] introduced hardware-assisted mechanisms to ensure performance isolation in public clouds. OSMOSIS [13] introduced a hardware-software co-design that isolates DMA queues, execution pipelines, and internal scheduling domains within SmartNICs. FairNIC [10] proposed static partitioning of SmartNIC cores, buffers and caches and applies a distributed token based rate-limiter for on-board accelerators, thereby eliminating cross-tenant contention while sustaining the Cavium LiquidIO SmartNIC's full 25 Gbps line-rate throughput. multi-tenants. In this paper, we propose HT-Verbs, which collects key metrics derived from the requested RDMA verbs and achieve performance isolation per container through dynamic adaptive throttling, without requiring hardware or firmware modifications of an RNIC.

RDMA in Container Environments. Recently, there has been growing interest in applying RDMA to container environments [2]. Freeflow [14] introduced a structured resource allocation approach for SmartNIC-enabled systems by creating virtual NICs and virtualized address spaces. DockRDMA [19] proposed a hybrid RDMA virtualization framework that dynamically switches between VF-based direct access and a proxy-based fallback. Liu et al. [21] addressed RDMA scalability limitations in containerized networks by dynamically steering traffic between one-sided RDMA and connectionless or proxy-based paths. TSoR [27] presented a turnkey RDMA network solution for containerized applications, transparently multiplexing POSIX-socket traffic over pre-established RDMA channels and integrating with Kubernetes via a CNI plugin.

RDMA Security. Several studies have investigated RDMA security and uncovered a range of vulnerabilities. Bedrock [30] employed P4-programmable switches to protect RDMA traffic against spoofing, access violations, and side-channel attacks. LoRDMA [28] identified a class of low-rate DoS attacks that exploit the interaction between priority flow control and DCQCN (Data Center Quantized Congestion Notification), where short, coordinated bursts from multiple senders can induce global congestion and cause DCQCN to throttle benign flows. ReDMArk [25] analyzed RDMA's built-in security mechanisms, demonstrating how weak authentication allows access-key inference, memory rekeying, and connection hijacking. These attacks take advantage of RDMA's insufficient isolation and lack of robust authentication to enable unauthorized memory access across tenants. In contrast, our work assumes an adversary within RDMA-enabled container environments and targets the RNIC's microarchitectural resources to induce performance interference in other victim containers.

7 Conclusion and Future Work

In this work, we experimentally analyzed performance isolation problems in RDMA-enabled container environments and proposed two types of resource exhaustion attacks that target RNIC resources: (i) state saturation attacks and (ii) pipeline saturation attacks. Our experiments reveal that state saturation attacks cause up to 93.9% bandwidth loss, a 1,117× latency increase, and a 115% rise in cache misses, while pipeline saturation attacks induce severe link-level congestion and amplification, where small verb requests consume disproportionately large RNIC resources. To counter these threats, we propose HT-Verbs, a threshold-driven framework that, using real-time RDMA verb telemetry, classifies RNIC resources into Hot, Warm, and Cold tiers, dynamically throttling abusive workloads without requiring hardware modifications. As future work, we plan to implement a prototype of HT-Verbs and empirically validate its practicality and effectiveness under real-world container workloads.

Acknowledgments. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2024-00457937, Design and implementation of security layers for secure WebAssembly-based serverless environments).

References

1. Infiniband Verbs Performance Tests (perftest). https://github.com/linux-rdma/perftest (2022)
2. Kubernetes Using SR-IOV. https://docs.nvidia.com/networking/display/mlnxofedv461000/kubernetes+using+sr-iov (2023)
3. Nvidia Quality of Service and Enhanced Transmission Selection. https://docs.nvidia.com/networking/display/mlnxenv543750lts/quality+of+service+(qos) (2023)
4. Nvidia VirtIO-net Emulated Devices. https://docs.nvidia.com/networking/display/bluefielddpuosv450/virtio-net+emulated+devices (2023)
5. DOCA Flow API. https://docs.nvidia.com/doca/sdk/doca+flow/index.html (2025)
6. DOCA Telemetry Libraries. https://docs.nvidia.com/doca/sdk/doca+telemetry/index.html (2025)
7. NVIDIA BlueField-3 DPU. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/documents/datasheet-nvidia-bluefield-3-dpu.pdf (2025)
8. NVIDIA DOCA Software Framework. https://developer.nvidia.com/networking/doca (2025)
9. Anagnostopoulos, M., Kambourakis, G., Kopanos, P., Louloudakis, G., Gritzalis, S.: DNS Amplification Attack Revisited. Computers & Security **39**, 475–485 (2013)
10. Grant, S., Yelam, A., Bland, M., Snoeren, A.C.: SmartNIC Performance Isolation with FairNIC: Programmable Networking for the Cloud. In: Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication (SIGCOMM '20). pp. 681–693. ACM (2020)
11. Guo, C., Wu, H., Deng, Z., Soni, G., Ye, J., Padhye, J., Lipshteyn, M.: RDMA over Commodity Ethernet at Scale. In: Proceedings of the 2016 ACM SIGCOMM Conference. pp. 202–215. ACM (2016)
12. Haecki, R., Mysore, R.N., Suresh, L., Zellweger, G., Gan, B., Merrifield, T., Banerjee, S., Roscoe, T.: How to Diagnose Nanosecond Network Latencies in Rich End-host Stacks. In: Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI). pp. 861–877 (2022)
13. Khalilov, M., Chrapek, M., Shen, S., Vezzu, A., Benz, T., Di Girolamo, S., Schneider, T., De Sensi, D., Benini, L., Hoefler, T.: OSMOSIS: Enabling Multi-Tenancy in Datacenter SmartNICs. In: Proceedings of the 2024 USENIX Annual Technical Conference (ATC). pp. 247–263. USENIX (2024)
14. Kim, D., Yu, T., Liu, H.H., Zhu, Y., Padhye, J., Raindel, S., Guo, C., Sekar, V., Seshan, S.: FreeFlow: Software-based Virtual RDMA Networking for Containerized Clouds. In: Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI). pp. 113–126. USENIX (2019)
15. Kong, X., Chen, J., Bai, W., Xu, Y., Elhaddad, M., Raindel, S., Padhye, J., Lebeck, A.R., Zhuo, D.: Understanding RDMA microarchitecture resources for performance isolation. In: Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI). pp. 31–48. USENIX (2023)

16. Kührer, M., Hupperich, T., Rossow, C., Holz, T.: Exit from Hell? Reducing the Impact of Amplification DDoS Attacks. In: Proceedings of the 23rd USENIX security symposium (USENIX Security). pp. 111–125. USENIX (2014)
17. Le, Y., Chang, H., Mukherjee, S., Wang, L., Akella, A., Swift, M.M., Lakshman, T.: UNO: Uniflying host and smart NIC offload for flexible packet processing. In: Proceedings of the 2017 Symposium on Cloud Computing (SoCC). pp. 506–519. ACM (2017)
18. Lee, S., Choi, M., Yeom, I., Kim, Y.: PeRF: Preemption-enabled RDMA Framework. In: Proceedings of the 2024 USENIX Annual Technical Conference (USENIX ATC '24). pp. 209–225. USENIX (2024)
19. Li, X., Shu, R., Chen, Z., Luo, X., Zhang, Y., Wang, B., Meng, Q., Ren, F.: Dockrdma: Hybrid rdma virtualization for containerized clouds. In: 2024 IEEE 32nd International Conference on Network Protocols (ICNP). pp. 1–12. IEEE (2024)
20. Liu, M., Cui, T., Schuh, H., Krishnamurthy, A., Peter, S., Gupta, K.: Offloading distributed applications onto smartnics using ipipe. In: Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM). pp. 318–333. ACM (2019)
21. Liu, W., Qian, K., Li, Z., Qian, F., Xu, T., Liu, Y., Guan, Y., Zhu, S., Xu, H., Xi, L., Qin, C., Zhai, E.: Mitigating Scalability Walls of RDMA-based Container Networks. In: Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI). pp. 1049–1065. USENIX (2025)
22. Lou, J., Kong, X., Huang, J., Bai, W., Kim, N.S., Zhuo, D.: Harmonic: Hardware-assisted RDMA Performance Isolation for Public Clouds. In: Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI). pp. 1479–1496. USENIX (2024)
23. Petazzoni, J.: Pipework: Docker Networking Wrapper. https://github.com/jpetazzo/pipework (2014)
24. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-party Compute Clouds. In: Proceedings of the 16th ACM conference on Computer and communications security (SIGCOMM). pp. 199–212. ACM (2009)
25. Rothenberger, B., Taranov, K., Perrig, A., Hoefler, T.: ReDMArk: Bypassing RDMA security mechanisms. In: Proceedings of the 30th USENIX Security Symposium (USENIX Security). pp. 4277–4292. USENIX (2021)
26. Shringarputale, S., McDaniel, P., Butler, K., La Porta, T.: Co-residency Attacks on Containers are Real. In: Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop (CCSW). p. 53–66. ACM (2020)
27. Sun, Y., Qu, Q., Zhao, C., Krishnamurthy, A., Chang, H., Xiong, Y.: SoR: TCP Socket over RDMA Container Network for Cloud Native Computing. arXiv preprint arXiv:2305.10621 (2023)
28. Wang, S., Zhang, M., Du, Y., Chen, Z., Wang, Z., Xu, M., Xie, R., Yang, J.: LoRDMA: A New Low-rate DoS Attack in RDMA Networks. Proceedings of the Network and Distributed System Security Symposium (NDSS) (2024)
29. Xiao, Y., Tootaghaj, D.Z., Dhakal, A., Cao, L., Sharma, P., Kuzmanovic, A.: Conspirator: SmartNIC-Aided Control Plane for Distributed ML Workloads. In: Proceedings of the 2024 USENIX Annual Technical Conference (ATC). pp. 767–784 (2024)
30. Xing, J., Hsu, K.F., Qiu, Y., Yang, Z., Liu, H., Chen, A.: Bedrock: Programmable Network Support for Secure RDMA Systems. In: Proceedings of the 31st USENIX Security Symposium (USENIX Security). pp. 2585–2600. USENIX (2022)