Securing RDMA for High-Performance Datacenter Storage Systems

Anna Kornfeld Simpson
University of Washington

Adriana Szekeres
University of Washington

Jacob Nelson
Microsoft Research

Irene Zhang
Microsoft Research

Abstract

RDMA is increasingly popular for low-latency communication in datacenters, marking a major change in how we build distributed systems. Unfortunately, as we pursue significant system re-designs inspired by new technology, we have not given equal thought to the consequences for system security. This paper investigates security issues introduced to datacenter systems by switching to RDMA and challenges in building secure RDMA systems. These challenges include changes in RPC reliability guarantees and unauditable data-accesses. We show how RDMA's design makes it challenging to build secure storage systems by analyzing recent research systems; then we outline several directions for solutions and future research, with the goal of securing RDMA datacenter systems while they are still in the research and prototype stages.

1 Introduction

Remote Direct Memory Access (RDMA) is a networking technology that improves performance by making use of the direct connection between a server's network card (NIC) and memory to bypass its CPU. RDMA originates from the High-Performance Computing (HPC) community, where applications are homogenous, highly parallel, and require both high bandwidth and low latency communication between nodes. Recently, researchers in the systems and networking community have proposed the use of RDMA in distributed storage systems in datacenters, citing the falling costs of RDMA NICs and rising network bandwidths relative to CPU speeds.

Since RDMA originated in the HPC community, its design has a security model that differs from that of the datacenter. For example, HPC clusters often assumed a high degree of trust between users; when that was not possible (e.g., for classified jobs) clusters used physical isolation [42]. In contrast, the datacenter is a shared environment, with untrusted users. As a result, many of RDMA's original security assumptions no longer hold, making it important to re-evaluate the danger of any security concerns. Researchers have been documenting concerns about the RDMA protocol (e.g., its lack of encryption) for more than two decades, and continue to discover more. Once those protocol concerns are addressed, however, we find CPU-bypassing RDMA still poses challenges for distributed system designers, and we analyze these challenges here. RDMA networking is widely deployed in datacenters [2, 19], and multiple research datacenter storage systems using RDMA have recently appeared, offering systems designers an ideal opportunity to rethink RDMA security.

In this paper, we explore the security challenges RDMA introduces by analyzing eight recently proposed distributed storage systems; we then discuss directions for solutions and further research. We believe achieving more secure RDMA communication without sacrificing performance is possible and the time to investigate this is now, while RDMA-using datacenter systems are still in research and prototype stages.

2 RDMA in the Datacenter

This section describes deployment of RDMA in datacenters and reviews security concerns in RDMA's design and implementation. Section 3 analyzes current proposals that use RDMA to build distributed systems in the datacenter.

RDMA Operations. RDMA offers two types of API calls: **SEND, RECV.** These "two-sided" calls implement a traditional RPC abstraction of sending and receiving messages. The recipient enqueues a RECV request to the card from userspace to give the NIC a buffer in which to write the received data. After this happens, the sender enqueues a SEND request to pass a message descriptor and data buffer to the NIC. These APIs achieve kernel-bypass but both the sender and recipient CPUs are involved.
READ, WRITE, ATOMICS. These "one-sided" RDMA calls bypass the recipient CPU entirely and operate directly on its memory. The receiver must first register memory regions that may be accessed remotely via the NIC, and communicate access information with the sender. Once that is complete, the sender enqueues a request to its NIC containing a message descriptor with one-sided opcode and a data buffer. One-sided RDMA offers a shared memory abstraction.

RDMA Setup. Before running RDMA operations, two setup steps are required, neither of which are contained in the RDMA protocol itself [22, 23]. First, a connection must be established between each sender/receiver pair. A library called `RDMACM` (CM for Connection Manager) offers APIs for this set-up, using unencrypted TCP for communication [30]. Second, host memory regions must be *registered* with the NIC. The registration process generates a static 32-bit token

Figure 1: Structure of a RoCEv2 (RDMA over Converged Ethernet) packet, from the spec [22]. These packets operate over IP and UDP, so RDMA must provide reliability and ordering semantics. Nothing is encrypted or authenticated.

(called an `R_Key`) that senders must include in one-sided requests to demonstrate authorization to access that memory region. Since there is no mechanism to exchange these tokens in the RDMA protocols, the `RDMACM` library is often used for this exchange, again using unencrypted TCP [30].

RDMA replaces the standard datacenter networking protocols with an RDMA protocol such as iWarp or RoCEv2 (RDMA over Converged Ethernet) [22]. Figure 1 shows a RoCEv2 packet. The RoCE metadata and contents sit inside Ethernet, IP, and UDP frames.

Security Gaps in RDMA. Researchers have identified security gaps in the RDMA protocol design since its original use in the HPC community and continue to uncover new vulnerabilities. We summarize their findings before exploring the consequences of these flaws on modern RDMA systems.

- **Lack of Confidentiality.** RDMA NICs do not encrypt RDMA packets by default. Both metadata (such as memory addresses and `R_Keys`) and contents (application data) are visible on the wire [29], and can be seen by machines on the network, such as switches or middleboxes, as well as an adversary who has, via some compromise, obtained root access on the client's virtual machine [1]. This lack of confidentiality is not in line with modern best practices where packet transport is usually encrypted [18], either using a standard protocol such as TLS or a custom protocol such as Google's ALTS [16]. Neither of these protocols, which are based on a TCP transport layer, would be easy to use with RoCEv2 [40].

- **No Integrity Checks or Authentication.** By relying on the unencrypted and unauthenticated packet data, RDMA NICs make it impossible to enforce meaningful access controls [11]. For example, the `R_Key` in the `READ` and `WRITE` APIs is supposed to be a *capability*, passed from the server to the client during memory region registration, and used to prove authorization on the datapath. In practice, several weaknesses may allow adversaries to read and spoof RDMA packets, including the lack of transport encryption, the small (32-bit) size of the `R_Key`, and unrobust implementations (e.g. NICs often assign `R_Keys` sequentially [45]).

[1] We confirmed this using `tcpdump` and Wireshark [48] on Mellanox NICs. This requires the Mellanox OFED RDMA drivers, available on their website, and root access to set the `ethtool` flags that enable capture.

- **Availability Challenges.** Guo et al. point out several availability dangers in their 2016 paper, including deadlock, livelock, and the colorfully named "PFC Pause Frame Storms" [19]. RoCE (both v1 and v2) require "lossless" link layers; to achieve this, RDMA NICs and switches employ Priority Flow Control (PFC), a mechanism that issues "Pause" commands to senders when a recipient's buffer becomes full [22]. In a tree-like network structure with hosts at the leaves, if a switch gets a "Pause" from a host and then receives more traffic bound for that host, the switch's buffer will fill, causing it to propagate the "Pause" commands up the tree. This can lead to Denial of Service for unrelated hosts. Guo et al. saw these storms "multiple times" in production [19]. Recent RoCE and iWarp NICs do not require PFC to be enabled, but they are not necessarily interoperable with the existing installed base of RoCE NICs PFC [34].

- **Side Channels.** Tsai and Zhang in 2019 explored side-channels in RDMA NICs, inspired by recent security problems found with CPU speculative execution [43]. As the more glaring vulnerabilities in RDMA are fixed, we expect future research to uncover more attacks.

We have optimism that hardware vendors are considering these problems (e.g. [33]), but, as we discuss in the next section, there is additional complexity to building secure systems atop RDMA. Just as satisfying basic compliance concerns does not prevent vulnerabilities and compromises [41], addressing RDMA security fundamentals will not eliminate the security challenges of designing an RDMA-based distributed storage system. Particularly, our analysis found that the use of RDMA exacerbates the harm done by a compromised storage client, which threatens the attempt to preserve security parity with current datacenter systems.

3 Challenges for Distributed Storage Systems

In this section we present an analysis of recent research systems that use RDMA and identify further security challenges.

System and Threat Model. We expect RDMA-based distributed systems to be entirely deployed in the datacenter setting. We assume datacenter servers are connected through a standard datacenter Ethernet network running RoCEv2 [22], as in [19]. Each server runs one or more RDMA processes that serve as either clients or servers in the distributed RDMA system. Figure 2 illustrates this system model. Although the nodes can be virtualized (e.g., virtualizing the RDMA NIC through SR-IOV), the RDMA systems we analyzed do not mention using virtualization.

We assume the datacenter is safe from physical breaches and that the datacenter provider is trusted; however, the customer deploying the RDMA system might not be the provider and may have different goals (unlike in RDMA's original HPC setting where provider and user goals aligned).

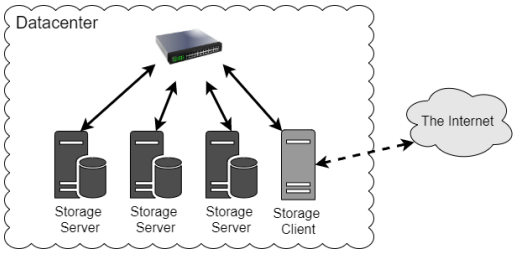

Figure 2: RDMA-based distributed systems operate entirely inside the datacenter and are connected via Ethernet. Many recent RDMA systems have been storage systems; thus the *client* is likely also an application server for some internet-based application and is especially vulnerable to compromise. System designers must consider the security consequences of such a compromise when designing RDMA systems.

We assume that any datacenter storage system, including one using RDMA, has security goals including: (1) preventing unauthorized exfiltration of data from the system, (2) maintaining the integrity of data and (3) preserving availability in the face of an attempted denial of service attack. Systems that provide transactional storage and consistency guarantees will be especially concerned about the integrity of metadata (e.g. locks), as well as the storage contents.

We are primarily concerned with violations of the security goals if a server or client is compromised. While distributed systems typically do not worry about a compromised node, we argue that RDMA systems must consider them more closely. Traditional distributed systems have well-defined interfaces between client and server, limiting the damage of a compromised node. For example, while a malicious key-value store client could perform arbitrary reads and writes, these operations would be logged server-side, and in production systems, the server would perform permissions checking [37].

RDMA servers, however, expose a broader API to other nodes, often giving clients the ability to read or write large regions of memory. A malicious client could release locks, abort transactions, or re-write portions of the log. In many RDMA systems today, a single malicious client could leak the entire database with no trace.

VLANs are Insufficient. Datacenters employ network virtualization techniques (e.g., VLANs) to isolate each customer's traffic. However, an adversary who has compromised a client machine (or the customer's VM on the client machine) needs only to see *that customer's* RDMA traffic to glean the necessary details for accessing the storage server, such as memory addresses and `R_Keys`. No other machine or VM traffic is necessary. The adversary can then use the same compromised client to modify, or undetectably exfiltrate data.

A Sample of Recent RDMA Research Storage Systems. Recently, a number of research systems have used RDMA as a high-performance communication mechanism. Table 1 lists 8 recent systems and the RDMA features that they use. In general, systems leverage one-sided RDMA operations when possible, as these operations are faster and reduce remote CPU usage. This section analyzes the security challenges introduced by RDMA for each system design (the right side of Table 1).

Unauditable reads. After a compromise, investigators attempt to determine what data has been accessed by the adversary; adversaries may seek to hide what data they have breached in attempts to avoid detection or attribution, or to increase the time they have to exploit their stolen data. System designers have a well-established tool to help audit data accesses: server-side logging. The server logs read requests received, even those from compromised clients. Unfortunately, since RDMA READ commands from client to server do not involve the server's CPU, suddenly these data accesses cannot be logged. We call these *unauditable reads* and believe they are untenable for systems that contain user data.

Concurrency and protocol problems. *Unauditable writes* are less sneaky than reads since the written data leaves a trace. However, they can still cause chaos. For example, if transactions execute without the server CPU, it becomes the responsibility of the client to correctly acquire and release locks (for example, in DrTM [47]). An adversary may disrupt consistency by releasing a lock, or by simply ignoring locks and/or "abort" messages during a distributed transaction. Additionally, traditionally immutable objects such as transaction history could be overwritten after they are "committed".

Breaking the RPC abstraction. RDMA's wider interface breaks some of the layered abstractions distributed systems traditionally assume. For instance, a non-RDMA application might use a library such as `gRPC` [17] for communication. `gRPC` uses HTTP/2 as a transport layer with TLS encryption, carried by TCP, IP, and Ethernet. Transport reliability would be provided by TCP, authentication might be provided by TLS, and actual modification of user data would be provided by RPC handlers running on the remote end. Each of these layers has a narrow interface that behaves according to a simple contract. RDMA disrupts these abstractions, particularly with the one-sided READ and WRITE operations that bypass the remote CPU entirely.

Some RDMA systems, such as FaRM [12], attempt to replicate the RPC task-queuing abstraction by using RDMA WRITE operations to a shared buffer and keeping shared state about the head and tail of the buffer. Unfortunately, this requires all clients to correctly and non-maliciously write to the appropriate point in the buffer; otherwise a client could overwrite an RPC or even modify another client's RPC before it is read by the server.

Whole-Network denial of service. Denial of service by spoofed packets was a concern in the Lee et al. 2005 paper on InfiniBand security [29] and denial of service situations were observed in practice in an early RoCEv2 deployment [19].

Table 1: The security consequences of using RDMA for distributed systems depend on how the system is designed to use the RDMA APIs. The left half of the table summarizes recent systems and their use of RDMA operations (to show directionality of SEND and RECV, we indicate APIs used by the client). The right half summarizes the security implications of each system's design, which we discuss in Section 3. An empty cell means a certain danger is not applicable to that system. The security issues we explore make the compromise of a node in an RDMA system more damaging than typical distributed system compromises due to use of RDMA rather than a narrower API.

System	Client to server RDMA `verbs` operations				System Design Dangers (Section 3)				
	SEND	RECV	READ	WRITE	Unaudit. Reads	Concurr. Problems	Broken RPC	Large DoS	Param. Manip.
Pilaf (2013) [35]	x		x		x				
FaRM (2014) [12]			x	x	x		x		
HERD (2014) [24]		x		x			x	x	
DrTM (2016) [47]			x	x	x	x			
FaSST (2016) [25]	x	x						x	
Octopus (2017) [31]		x	x	x	x	x		x	
Hyperloop (2018) [28]	x		x	x	x	x			x
DrTM+H (2018) [46]	x	x	x	x	x	x		x	

The combination of these two factors is, however, unstudied and potentially quite dangerous: since RoCEv2 (unlike InfiniBand) requires a lossless network and Priority Flow Control [22], spoofed packets could contribute to problems such as the "PFC Pause Frame Storms" [19], potentially slowing down the entire network. For example, if an adversary spoofs traffic that causes servers to produce a SEND to a specific client that is not expecting to RECV anything (as for data reads in HERD and FaSST [24, 25]), the adversary may be able to overwhelm buffers at the targeted client or one of the switches en-route. Depending on the application design, an unexpected SEND might instead push the connection into a corrupted state where no future RDMA commands will succeed, a situation difficult to recover from without recreating the connection and another form of Denial of Service.

Execution Parameter Manipulation. Hyperloop [28] is a unique system which registers portions of the server NIC's *work queue* (the buffers containing the commands and arguments for the NIC to execute in the future) as remotely writable memory so that arbitrary data can be replicated from the client to a third party without any involvement of the intermediate server CPU. This strategy leads to vulnerabilities not seen in other systems, specifically parameter manipulation in the remote execution. A malicious client could perform an unauditable write to the work queue, causing the server NIC to write data to a third-party that the client itself may not have access to! This is another example of how the use of the RDMA APIs rather than more narrowly scoped APIs from traditional distributed systems can lead to more downstream harms from a single compromised client.

We define this vulnerability as parameter manipulation because the authors of Hyperloop noted (in response to a query about APIs they use) that the APIs allowed them to separate the arguments that needed to be remotely writable from the specification of which RDMA operation to use [21]. Without this separation, the vulnerability would be even more serious since the client would be able to specify code to remotely execute on the server NIC. Given the existing security problems with RDMA and increased difficulty in reasoning about system security, we consider remote execution to be exceptionally dangerous in an RDMA environment.

Lessons from System Analysis. There is more complexity to building secure RDMA systems than just fixing transport encryption and the other missing components from Section 2. We consider unauditable reads to be particularly dangerous for cloud systems due to recent regulations (such as GDPR [14] and CCPA [5]) that expect details of which data was adversarially accessed in a data breach. Now is the time to secure RDMA in the datacenter: we must address both fundamentals and the system design challenges while these systems are still in research and prototype stages.

4 Solutions and Future Directions

Two hopeful trends for addressing RDMA's security problems are the prevalence of centralized configuration services in datacenter environments and the continued increase in processing capabilities of RDMA network hardware. Below, we outline how these trends might combine to address the challenges of unencrypted packets and unauditable reads while minimally impacting RDMA performance.

Encrypting and signing RDMA packets. With the dangers of unencrypted RDMA traffic, particularly RDMA header information such as memory addresses and `R_Keys`, repeated for more than two decades [11, 29, 40, 43], it is important to consider what components of the solution are difficult. One common challenge in setting up encryption over the network is key distribution: each pair of communicating servers must derive shared secrets that can be used to encrypt and integrity-check (MAC or sign) packets. The

position of RDMA systems inside a datacenter offers an opportunity: in datacenters, centralized configuration services such as Chubby [4] or Zookeeper [20] abound. Using one of these services to distribute shared keys between clients and servers seems tractable, since these services are already used to distribute per-machine information such as IP addresses. In their 2005 work identifying problems with InfiniBand security, Lee et al. suggested that hosts could use provided keys to "sign" their messages, preventing undetectable modifications and preserving message integrity [29]. One aspect of this challenge that needs further investigation is how to transfer keys from a host CPU to its NIC.

With modern, widely-used, cryptographic protocols, we can preserve both confidentiality and integrity of packets. One possibility for RoCEv2 would be the DTLS (Datagram TLS) scheme that creates the functionality of TLS for UDP packets [39]. NICs are starting to offer accelerated encryption (though not for RDMA): there are NICs that offer DTLS (for standard UDP traffic) via an onboard ASIC [8, 9, 33] and programmable "SmartNICs" advertise high-performance encryption with their own hardware acceleration [6, 32]. A 2016 press release from a NIC vendor announced "Low Latency 100Gbps line-rate" for DTLS [7]. Moving beyond NICs, on-the-wire solutions such as Catapult could also provide encryption at line rate [15, 38]. A standardized secure version of RoCE (using DTLS) or iWarp (using TLS) would ensure that implementers on all platforms achieve the same guarantees. The ingredients for high-performance packet encryption and integrity-checking are in place: researchers, cloud providers and hardware vendors need to standardize the needed features and ensure that they are available.

Regaining auditability. Preventing unauditable RDMA READ operations during a data breach requires logging that the adversary cannot circumvent. Although SmartNICs might offer the programmability necessary to log one-sided operations at the receiver side, there would be a significant performance cost. Instead, could a programmable switch, designed for much higher throughput than a NIC, sample RDMA packets that transverse it and duplicate them out to a logging server? Does adversarial RDMA use have distinct statistical patterns from typical system behavior, detectable even via encrypted packets? Could some control-plane RDMA traffic be redirected to a centralized system that will log it (in plaintext), while subsequent dataplane traffic proceeds on the existing, high-performance route? This may require a move away from the "register memory once, use forever" design choice made by the systems we analyzed, instead having the logged packets give more specific information about which memory regions are currently being accessed. We strongly encourage researchers to investigate these approaches.

Additional directions. Future security research is also necessary in other aspects of these increasingly advanced NICs. For example, do the NIC design and its driver successfully protect process isolation on the local machine? Between VMs that share the hardware? Are there any additional security considerations when using RDMA over a wide-area network (such as between datacenters)? Can adding additional computation or memory to advanced NICs help mitigate any of the security problems? We encourage other researchers to consider these and related questions.

5 Related Work

While a number of systems have incorporated RDMA for performance benefits [10, 12, 13, 24, 25, 31, 35, 46, 47] and others have attempted to improve the interface [1, 44], relatively few have attempted to improve the security of such systems. We briefly review those here.

In a 2005 RFC about RDMA over IP, Romanow et al. identify security as a crucial component to the success of their proposal [40]. Unfortunately, their proposed solution, IPSec [3, 26, 27], never received wide adoption.

Also in 2005, Lee et al. wrote about security flaws in the InfiniBand Architecture, including pointing out the critical weakness that is plaintext `R_Keys` for remote memory region access [29]. It appears that some of their mitigations for denial of service were incorporated into the InfiniBand spec [23].

Noronha et al. discuss some security considerations when layering NFS on top of RDMA: instead of opening the server memory up for READ and WRITE operations by the client, they have the *client* register its memory for remote access, contact the server via two-sided operations, and then let the server deposit or retrieve data from the client's registered memory [36]. This reduces the security risk of a compromised client, at the cost of much more effort by the server's CPU. This goes against the trend of recent RDMA research systems, which seek to reduce the server's computational workload.

Tsai and Zhang propose a number of improvements to RDMA in LITE [44], including doing a secure key exchange between servers involved in remote memory operations. Their system design relies on the kernel to provide this functionality and does not consider an on-the-wire attacker. Additionally, their implementation registers a single RDMA region for each machine due to limitations in NIC memory; this exacerbates the danger of a compromised machine.

In more recent work, Tsai and Zhang demonstrate a number of security threats, implementation issues, and side channel vulnerabilities in RDMA NICs [43, 45].

6 Conclusion

RDMA is increasingly being used to build critical distributed datacenter systems. Unfortunately, we find that the security weaknesses of RDMA mean that a compromised node can do more damage when employing a RDMA-based implementation of these systems, such as exfiltrating data without leaving a trace or breaking concurrency contracts. We believe it is crucial to close this security gap now, while these systems are still at the prototype stage in the datacenter.

7 Discussion

1. Once hardware vendors offer hardware to facilitate building more secure RDMA systems, systems software developers must start using these new security features. What would be useful ways to communicate the necessary changes to RDMA programming to systems builders around the world? The standards process? Blog posts? Example systems?

2. A key enabler of secure software is writing libraries with secure and reasonable defaults. For example, cryptographic libraries whose default (no argument) configurations are secure are less likely to be misused by developers. The existing RDMA libraries such as `libibverbs` and `librdmacm` can already be challenging for new RDMA developers to learn. How can we redesign these (and other kernel bypass networking libraries) to simplify the use of good security practices? Besides adding documentation for any new APIs, what would help systems developers understand secure use of RDMA libraries?

3. Unauditable RDMA READ operations – performed by an adversary who has compromised a storage client in the datacenter and is exfiltrating data - are a particularly concerning attack in the cloud scenario. In "Regaining auditability", we presented a few ideas for ways to log data such that reads are still auditable. We are eager to hear feedback on feasibility of these and any other ideas!

4. If you have experience with other forms of modern kernel-bypass or CPU-bypass networking, are there security lessons we can learn from those systems? Do any of the security concerns we identified for RDMA apply to other systems you are familiar with? Are the risks and threat models similar?

5. As other aspects of cloud processing are offloaded from the CPU on to specialized hardware, are there any confidentiality, integrity, or availability properties that are being lost in the transition? If an adversary is another tenant on the cloud system, what might they be able to exfiltrate or manipulate? How can their harm be limited?

Acknowledgments

The authors would like to thank our shepherd Michael Wei, as well as Ivan Evtimov, Shrirang Mare, and members of the University of Washington Systems lab for feedback on the paper. This work was supported in part by an NSF Graduate Research Fellowship under Grant No. DGE-1256082 and by DARPA grant FA8750-16-2-0032.

References

[1] M. K. Aguilera, N. Amit, I. Calciu, X. Deguillard, J. Gandhi, S. Novaković, A. Ramanathan, P. Subrahmanyam, L. Suresh, K. Tati, R. Venkatasubramanian, and M. Wei. Remote regions: a simple abstraction for remote memory. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 775–787, Boston, MA, 2018. USENIX Association.

[2] Alibaba Cloud. Super-computing cluster, 2018.

[3] R. Atkinson. Security Architecture for the Internet Protocol. RFC 1825, RFC Editor, August 1995. http://www.rfc-editor.org/rfc/rfc1825.txt.

[4] M. Burrows. The chubby lock service for loosely-coupled distributed systems. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 335–350. USENIX Association, 2006.

[5] California State Legislature. CALIFORNIA CONSUMER PRIVACY ACT. Cal. Legis. Serv., 2018.

[6] Chelsio Communications. Chelsio Cryptographic Offload and Acceleration Solution Overview. https://www.chelsio.com/crypto-solution/. (Accessed 17 January 2019.).

[7] Chelsio Communications. CHELSIO TERMINATOR 6 (T6) IN-LINE ACCELERATION OF UBIQUITOUS TLS/SSL DELIVERS GROUND-BREAKING LOW LATENCY, 100 GIGABIT-PER-SECOND LINE-RATE PERFORMANCE, 2016.

[8] Chelsio Communications. T62100-CR: Ultra High Performance, Half Size, DualPort 40/50/100GbE Unified Wire Adapter, Accessed 2020.

[9] Chelsio Communications. T62100-SO-CR: High Performance, Low Profile, Dual Port 40/50/100GbE Server Offload Adapter, Accessed 2020.

[10] Y. Chen, X. Wei, J. Shi, R. Chen, and H. Chen. Fast and General Distributed Transactions Using RDMA and HTM. In *Proceedings of the Eleventh European Conference on Computer Systems*, EuroSys '16, pages 26:1–26:17, New York, NY, USA, 2016. ACM.

[11] R. Dimitrov and M. Gleeson. Challenges and new technologies for addressing security in high performance distributed environments. In *Proceedings of the 21st National Information Systems Security Conference*, pages 457–468, 1998.

[12] A. Dragojević, D. Narayanan, M. Castro, and O. Hodson. FaRM: Fast Remote Memory. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 401–414, Seattle, WA, 2014. USENIX Association.

[13] A. Dragojević, D. Narayanan, E. B. Nightingale, M. Renzelmann, A. Shamis, A. Badam, and M. Castro. No Compromises: Distributed Transactions with

Consistency, Availability, and Performance. In *Proceedings of the 25th Symposium on Operating Systems Principles*, SOSP '15, pages 54–70, New York, NY, USA, 2015. ACM.

[14] European Parliament. REGULATION (EU) 016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), 2016.

[15] D. Firestone, A. Putnam, S. Mundkur, D. Chiou, A. Dabagh, M. Andrewartha, H. Angepat, V. Bhanu, A. Caulfield, E. Chung, H. K. Chandrappa, S. Chaturmohta, M. Humphrey, J. Lavier, N. Lam, F. Liu, K. Ovtcharov, J. Padhye, G. Popuri, S. Raindel, T. Sapre, M. Shaw, G. Silva, M. Sivakumar, N. Srivastava, A. Verma, Q. Zuhair, D. Bansal, D. Burger, K. Vaid, D. A. Maltz, and A. Greenberg. Azure accelerated networking: Smartnics in the public cloud. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 51–66, Renton, WA, Apr. 2018. USENIX Association.

[16] C. Ghali, A. Stubblefield, E. Knapp, J. Li, B. Schmidt, and J. Boeuf. Application Layer Transport Security. Technical report, Google Cloud, 12 2017. (Accessed on 01/15/2018).

[17] Google. grpc. https://grpc.io. (Accessed on 01/16/2018).

[18] Google Cloud. Google infrastructure security design overview. Technical report, 2017.

[19] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn. RDMA over Commodity Ethernet at Scale. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, pages 202–215, New York, NY, USA, 2016. ACM.

[20] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed. ZooKeeper: wait-free coordination for internet-scale systems. In *USENIX annual technical conference*, volume 8. Boston, MA, USA, 2010.

[21] Hyperloop authors. Personal Communication, 12 2018.

[22] Infiniband Trade Association. RoCEv2, 09 2014. (Accessed November 2018).

[23] Infiniband Trade Association. InfiniBand (TM) Architecture Specification Volume 1 Release 1.3, 03 2015.

[24] A. Kalia, M. Kaminsky, and D. G. Andersen. Using RDMA Efficiently for Key-value Services. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 295–306, New York, NY, USA, 2014. ACM.

[25] A. Kalia, M. Kaminsky, and D. G. Andersen. FaSST: Fast, Scalable and Simple Distributed Transactions with Two-Sided (RDMA) Datagram RPCs. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 185–201, Savannah, GA, 2016. USENIX Association.

[26] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, RFC Editor, November 1998.

[27] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301, RFC Editor, December 2005. http://www.rfc-editor.org/rfc/rfc4301.txt.

[28] D. Kim, A. Memaripour, A. Badam, Y. Zhu, H. H. Liu, J. Padhye, S. Raindel, S. Swanson, V. Sekar, and S. Seshan. Hyperloop: Group-based NIC-offloading to Accelerate Replicated Transactions in Multi-tenant Storage Systems. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, pages 297–312, New York, NY, USA, 2018. ACM.

[29] M. Lee, E. J. Kim, and M. Yousif. Security enhancement in InfiniBand architecture. In *19th IEEE International Parallel and Distributed Processing Symposium*, pages 10 pp.–, April 2005.

[30] librdmacm Maintainers. Rdma core userspace libraries and daemons, Accessed 2020.

[31] Y. Lu, J. Shu, Y. Chen, and T. Li. Octopus: an RDMA-enabled Distributed Persistent Memory File System. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, pages 773–785, Santa Clara, CA, 2017. USENIX Association.

[32] Mellanox Technologies. Mellanox BlueField (TM) SmartNIC. https://www.mellanox.com/related-docs/prod_adapter_cards/PB_BlueField_Smart_NIC.pdf. (Accessed 17 January 2019.).

[33] Mellanox Technologies. Connectx®-6 dx dual-port 100gbe/single-port 200gbe smartnic, Accessed 2020.

[34] Mellanox Technologies. Ethernet network, Accessed 2020.

[35] C. Mitchell, Y. Geng, and J. Li. Using one-sided RDMA reads to build a fast, cpu-efficient key-value store. In

Presented as part of the 2013 USENIX Annual Technical Conference (USENIX ATC 13), pages 103–114, San Jose, CA, 2013. USENIX.

[36] R. Noronha, L. Chai, T. Talpey, and D. K. Panda. Designing NFS with RDMA for Security, Performance and Scalability. In *2007 International Conference on Parallel Processing (ICPP 2007)*, pages 49–49, Sep. 2007.

[37] R. Pang, R. Caceres, M. Burrows, Z. Chen, P. Dave, N. Germer, A. Golynski, K. Graney, N. Kang, L. Kissner, J. L. Korn, A. Parmar, C. D. Richards, and M. Wang. Zanzibar: Google's consistent, global authorization system. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 33–46, Renton, WA, July 2019. USENIX Association.

[38] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P. Y. Xiao, and D. Burger. A reconfigurable fabric for accelerating large-scale datacenter services. In *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, pages 13–24, June 2014.

[39] E. Rescorla and N. Modadugu. Datagram transport layer security version 1.2. RFC 6347, RFC Editor, January 2012. http://www.rfc-editor.org/rfc/rfc6347.txt.

[40] A. Romanow, J. Mogul, T. Talpey, and S. Bailey. Remote Direct Memory Access (RDMA) over IP Problem Statement. RFC 4297, RFC Editor, December 2005.

[41] R. Stevens, J. Dykstra, W. K. Everette, J. Chapman, G. Bladow, A. Farmer, K. Halliday, and M. L. Mazurek. Compliance Cautions: Investigating Security Issues Associated with US Digital-Security Standards. In *Proceedings of Network and Distributed Systems Security (NDSS) Symposium*, 2020.

[42] J. Tomkins. Red Storm: The Birth of a New Supercomputer. https://www.osti.gov/servlets/purl/1142434, September 2008. (Accessed on 01/16/2018).

[43] S.-Y. Tsai, M. Payer, and Y. Zhang. Pythia: Remote Oracles for the Masses. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 693–710, Santa Clara, CA, Aug. 2019. USENIX Association.

[44] S.-Y. Tsai and Y. Zhang. LITE Kernel RDMA Support for Datacenter Applications. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 306–324, New York, NY, USA, 2017. ACM.

[45] S.-Y. Tsai and Y. Zhang. A Double-Edged Sword: Security Threats and Opportunities in One-Sided Network Communication. In *11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19)*, Renton, WA, July 2019. USENIX Association.

[46] X. Wei, Z. Dong, R. Chen, and H. Chen. Deconstructing RDMA-enabled Distributed Transactions: Hybrid is Better! In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 233–251, Carlsbad, CA, 2018. USENIX Association.

[47] X. Wei, J. Shi, Y. Chen, R. Chen, and H. Chen. Fast In-Memory Transaction Processing Using RDMA and HTM. SOSP '15, New York, NY, USA, 2015. ACM.

[48] WireShark. Wireshark. https://wireshark.org. (Accessed on 01/17/2018).