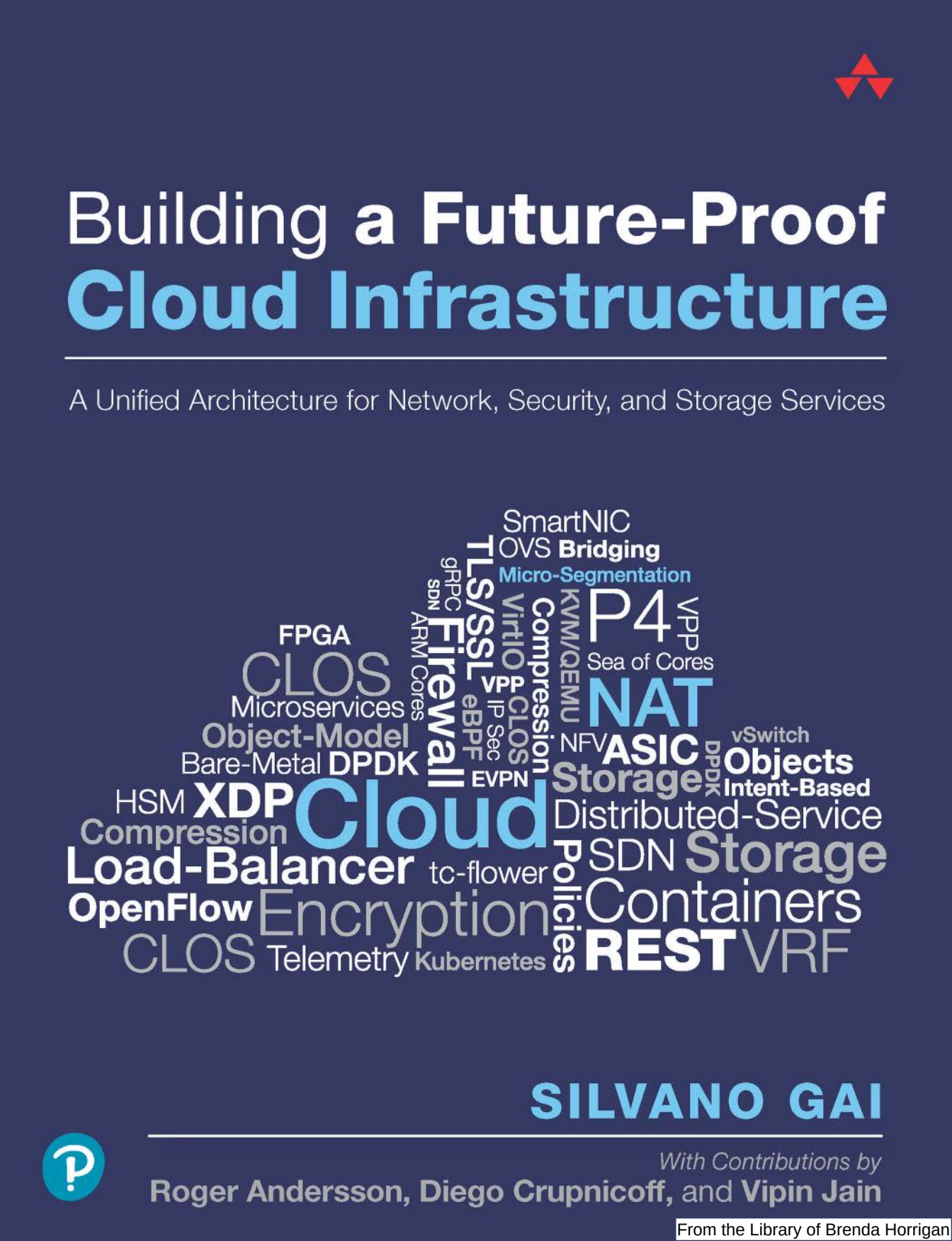

Building a Future-Proof Cloud Infrastructure

A Unified Architecture for Network, Security, and Storage Services

SILVANO GAI

With Contributions by

Roger Andersson, Diego Crupnicoff, and **Vipin Jain**

Building a Future-Proof Cloud Infrastructure

A Unified Architecture for Network, Security, and Storage Services

Silvano Gai

With Contributions by
Roger Andersson,
Diego Crupnicoff, and Vipin Jain

Addison-Wesley

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact intlcs@pearson.com.

Visit us on the Web: informit.com/aw

Library of Congress Control Number: 2019956931

Pearson Education, Inc.

ISBN-13: 978-0-13-662409-7

ISBN-10: 0-13-662409-X

ScoutAutomatedPrintCode

Editor-in-Chief
Mark Taub

Product Manager
James Manly

Managing Editor
Sandra Schroeder

Senior Project Editor
Lori Lyons

Copy Editor
Paula Lowell

Production Manager
Vaishnavi/codeMantra

Indexer
Erika Millen

Proofreader
Abigail Manheim

Editorial Assistant
Cindy Teeters

Cover Designer
Chuti Prasertsith

Compositor
codeMantra

Chapter 6

Distributed Storage and RDMA Services

In previous chapters, we have discussed distributed application services that can be placed almost anywhere on the network. In this chapter, we focus on infrastructure services that require access to server memory, typically through a PCIe interface.

In the not-so-distant past, the typical server included up to three types of communication interfaces, each connected to a dedicated infrastructure network (see Figure 6-1):

- A network interface card (NIC) connecting the server to the LAN and the outside world
- A storage interface that could attach to local disks or a dedicated storage area network (SAN)
- A clustering interface for high-performance communication across servers in distributed application environments

Over the past two decades, the communication substrate for these infrastructure services has been mostly unified. The opportunity to significantly reduce costs has driven this convergence. Cloud deployments with their massive scales have dramatically accelerated this trend. Today, networking, distributed storage, and RDMA clustering all ride the ubiquitous Ethernet; see Figure 6-2.

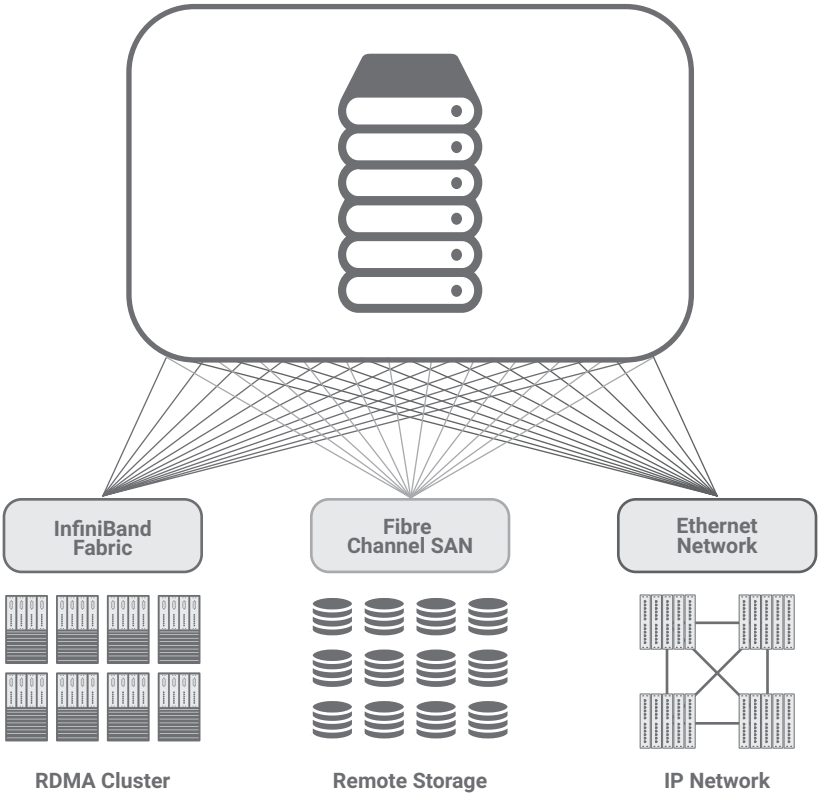

FIGURE 6-1 Host Connected to Dedicated Networks

In spite of the physical network consolidation, the three discussed infrastructure services remain individually exposed to the host operating systems through distinct software interfaces; see Figure 6-3. In all cases, these interfaces rely on direct memory access (DMA) to and from memory by the I/O adapter, most typically through PCIe.

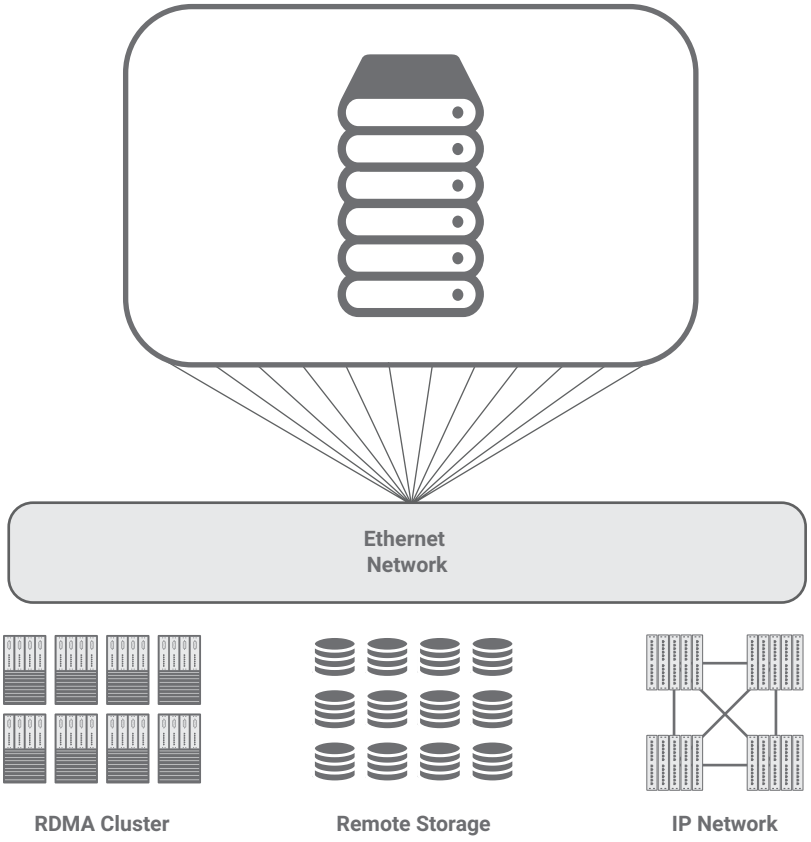

FIGURE 6-2 Hosts with Unified Networks

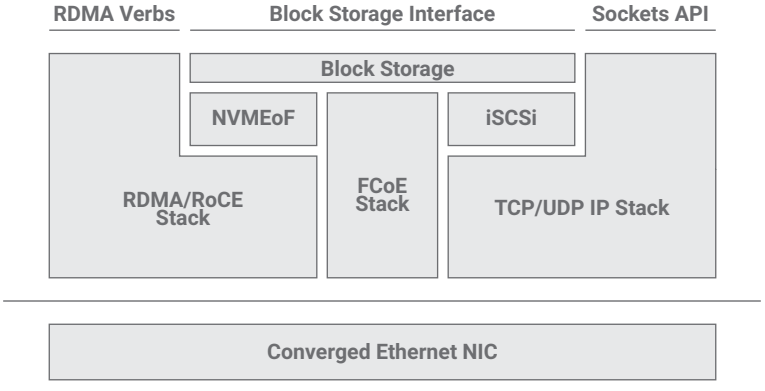

FIGURE 6-3 Unified Network Software Stack

6.1 RDMA and RoCE

Computer clusters have been around for many decades with successful commercial products since the 1980s by Digital Equipment Corporation [1], Tandem Computers [2], Compaq, Sun, HP, and IBM.

In a world where individual CPUs could no longer evolve at the pace of Moore's law, scaling out became the most viable way to satisfy the growing demand for compute-intensive applications. Over time, the move toward cloud services motivated the deployment of racks of standard servers at very large scale. Clusters of commodity servers have become the standard for the modern datacenter ranging from on-premise enterprise deployments all the way to public clouds.

In contrast to CPUs, communication technology continued to evolve at a steady pace and allowed bigger and better clusters. With faster networks came the demand for a more efficient I/O software interface that would address the communication bottleneck for high-performance distributed applications. In this context, the industry defined the Virtual Interface Architecture (VIA), intending to increase communication performance by eliminating host processing software overheads [3]. The objective: high bandwidth, low latency, and low CPU utilization.

These ideas were picked up in 1999 by two initially competing industry groups: NGIO and FutureIO. In the NGIO camp were Intel and Sun Microsystems, while IBM, HP, and Compaq led FutureIO. The head-to-head race to specify the new I/O standard was settled as both groups joined forces and created the InfiniBand Trade Association (IBTA), which in 2000 produced the first (InfiniBand) RDMA Specification [4], [5].

InfiniBand RDMA (Remote Direct Memory Access) was originally conceived as a vertically integrated protocol stack (see Figure 6-4) covering switches, routers, and network adapters and including from a physical layer up to software and management interfaces.

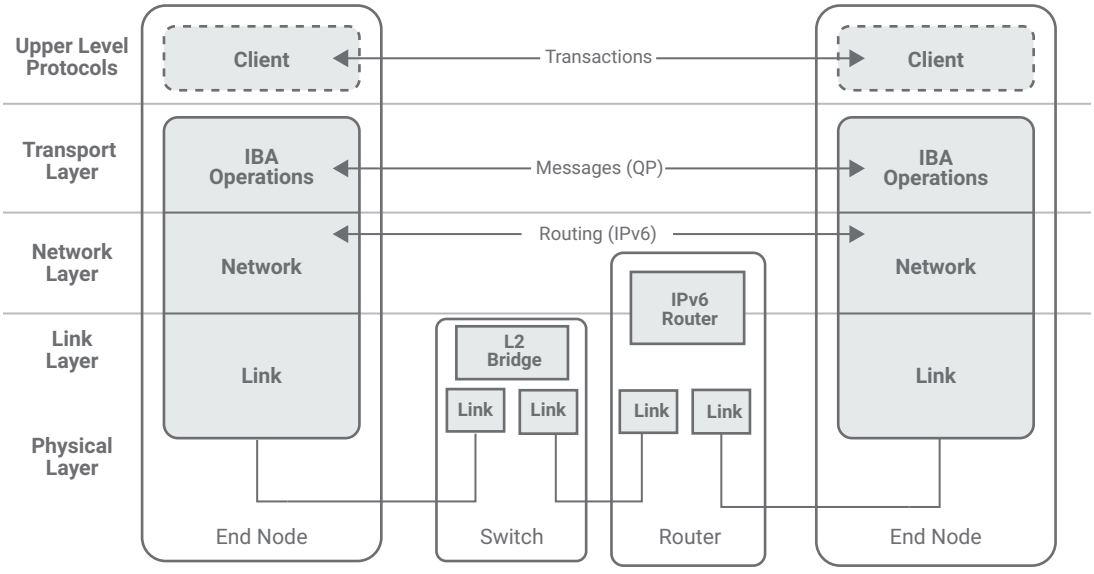

FIGURE 6-4 InfiniBand Protocol Stack and Fabric Diagram

The value proposition of RDMA is mainly derived from the following four characteristics:

- **Kernel bypass:** A mechanism that allows secure and direct access to I/O services from user-space processes without having to go through the OS kernel; see Figure 6-5. It eliminates a significant latency component and reduces CPU utilization.
- **Zero-copy:** The ability for the I/O device to directly read from and write to userspace memory buffers, thereby eliminating multiple copies of I/O data that is common with the OS-based software stack of traditional network protocols; see Figure 6-6.
- **Protocol offload:** Message segmentation and reassembly, delivery guarantees, access checks, and all aspects of a reliable transport are offloaded to the NIC, eliminating the consumption of CPU resources for network protocol processing; see Figure 6-7.
- **One-sided operations:** RDMA Read, Write and Atomic operations are executed without any intervention of the target-side CPU, with the obvious benefit of not spending valuable compute cycles on the target system; see Figure 6-8. This asynchronous processing approach results in a significant increase in message rates and a drastic reduction in message completion jitter, as the processing completely decouples I/O execution from the scheduling of the receiver's I/O stack software.

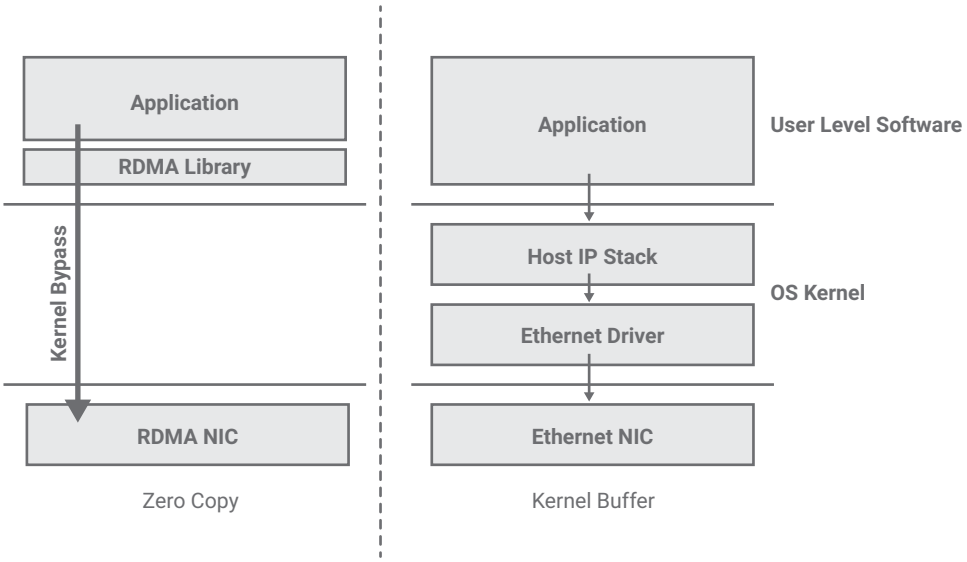

FIGURE 6-5 Cost of I/O, Kernel Network Stack versus Bypass

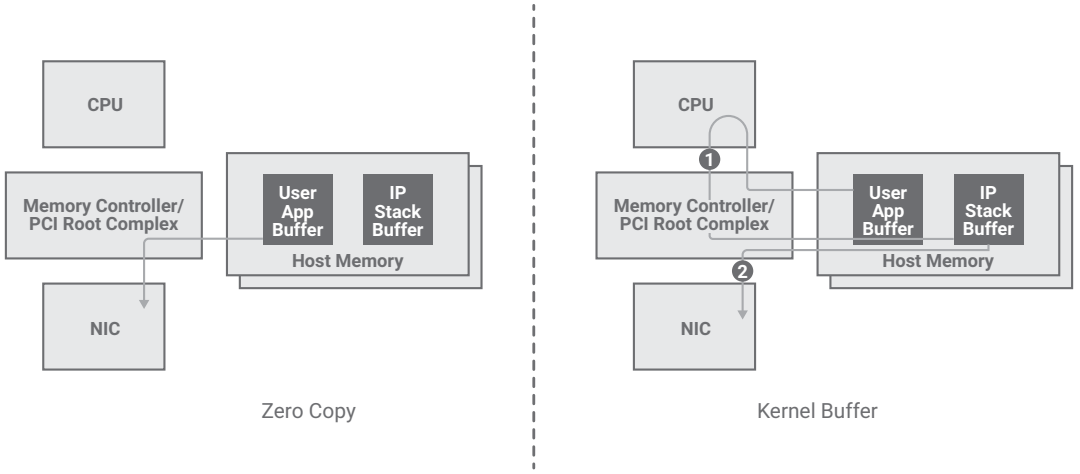

FIGURE 6-6 Kernel Buffer versus Zero-Copy

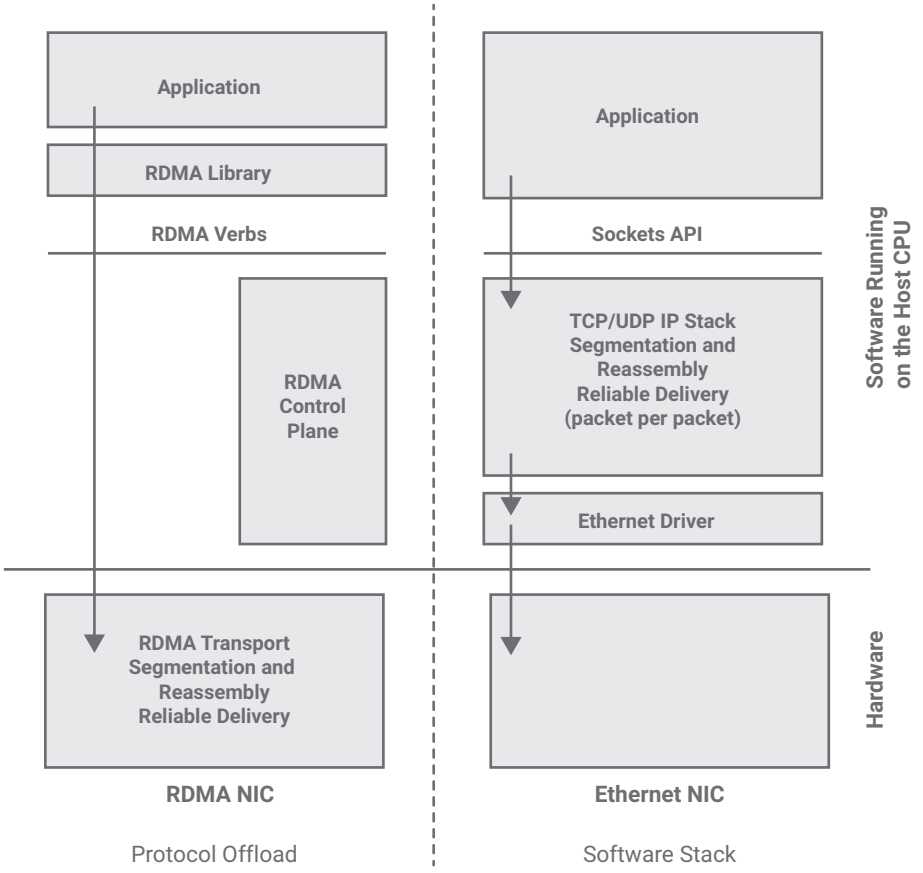

FIGURE 6-7 Software Stack versus Protocol Offload

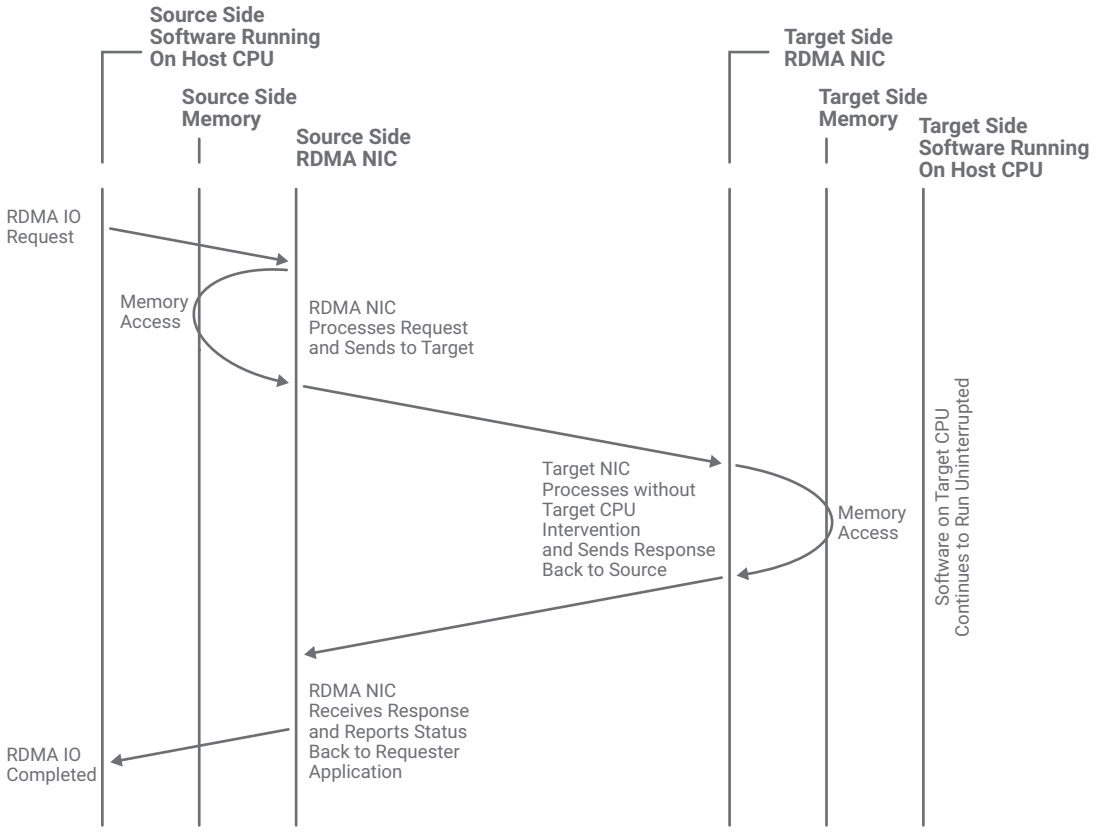

FIGURE 6-8 One-sided Operations Ladder Diagram

6.1.1 RDMA Architecture Overview

The term *RDMA*, denoting this infrastructure service, is a little bit of a misnomer. The so-called RDMA service includes actual RDMA (Remote Direct Memory Access) operations alongside traditional SEND/RECEIVE message semantics. Most fundamentally, the RDMA service implements a quite different I/O model than that of traditional networks.

Central to the RDMA model is the notion of Queue Pairs (QPs). These are the interface objects through which consumer applications submit I/O requests. A Queue Pair comprises a Send Queue (SQ) and a Receive Queue (RQ) and operates in a way that is somewhat similar to how Send and Receive Rings work on traditional Ethernet interfaces. The fundamental difference is that each RDMA flow operates on top of its own dedicated QP, and these QPs are directly accessed from their respective consumer processes without any kernel driver intervention in the data path.

RDMA operations, normally referred to as work requests (WRs), are posted into SQs and RQs and asynchronously serviced by the RDMA provider (that is, the RDMA NIC). Once executed and completed,

the provider notifies the consumer process through Completion Queues (CQs); see Figure 6-9. Consumer processes may have one or more CQs that can be flexibly associated with their QPs.

The RDMA provider guarantees QP access protection across multiple consumers via a combination of regular host virtual memory and the mapping of I/O space into each process's own space. In this manner, user processes can exclusively access their respective RDMA resources. Typically, each process is assigned a dedicated page in the I/O address space of the RDMA NIC that is mapped to the process's virtual memory. The process can then directly interact with the RDMA NIC using regular userspace memory access to these mapped addresses. The NIC, in turn, can validate legitimate use of the corresponding RDMA resources and act upon the I/O requests without the intervention of the OS kernel.

As a corollary to the direct access model and the individual flow interfaces (that is, QPs) being visible to the NIC, fine-grained QoS is a natural characteristic of RDMA. Most typically, RDMA NICs implement a scheduler that picks pending jobs from QPs with outstanding WRs following sophisticated programmable policies.

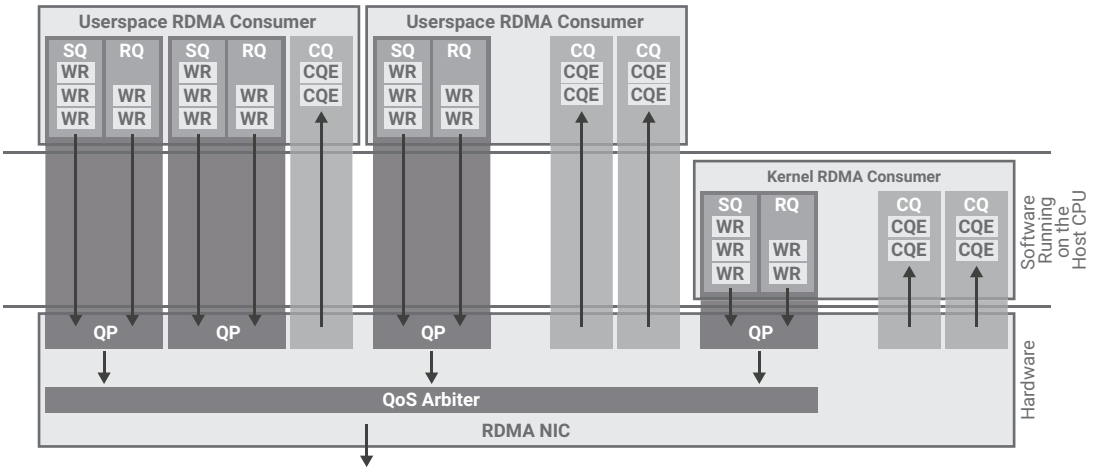

FIGURE 6-9 QPs, WRs, CQs, and Scheduler/QoS Arbiter

To allow direct I/O access to/from user-level process buffers, the RDMA model includes a mechanism called *memory registration*. This facility is used to pin into physical memory the user memory ranges that are meant to be used as the source or target of RDMA operations. In addition to pinning, during memory registration, the RDMA provider updates its memory mapping table with the physical addresses of the registered buffers and returns a memory key to the RDMA consumer process. Once registered, consumer processes can then refer to these memory regions in their submitted work requests, and the RDMA NIC can directly access these buffers during execution (that is, zero-copy).

6.1.2 RDMA Transport Services

The RDMA protocol standard originally defined two connection-oriented transport services: Reliable Connected (RC), Unreliable Connected (UC); and two datagram services: Reliable Datagram (RD) and Unreliable Datagram (UD); see Figure 6-10. The most commonly used ones are RC and UD. UC is rarely used and RD was never implemented.

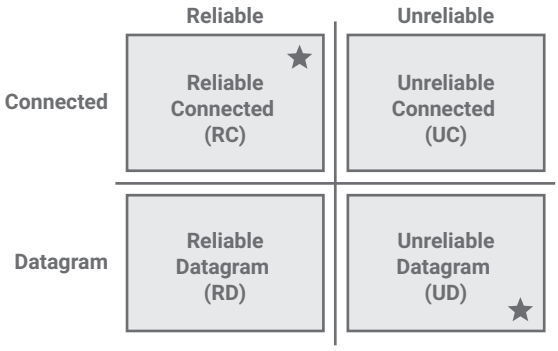

FIGURE 6-10 RDMA Transport Services

As far as reliability and connection characteristics are concerned, RC could be seen as the logical equivalent of the well-known TCP. A lot of the RDMA value proposition requires this transport service, and hence most RDMA consumer applications use it. Meanwhile, the UD transport service resembles UDP, and RDMA consumers use it mostly for management and control operations.

Over time, a new XRC transport service was added to the specification. It is a variation on RC that is mostly relevant for reducing the number of connections in scenarios with lots of processes per node.

6.1.3 RDMA Operations

The RDMA protocol includes Send/Receive semantics, RDMA Read/Writes, and Atomics.

Even though Send/Receives are semantically similar to their traditional network counterparts, the RDMA version leverages the kernel-bypass, the zero copy, and the reliable transport protocol offload of RDMA. In this way, consumer applications can attain low latency, high bandwidth, and low CPU utilization without significant changes to their communication model.

The Remote Direct Memory Access (RDMA) operations, RDMA Read and RDMA Write, give their name to the protocol. The main difference between these and Send/Receive is the one-sided nature of the execution. RDMA accesses complete with no intervention of the target-side CPU. Consumer applications can asynchronously read and write from and to remote node memory locations, subject to strict access right checks. Applications that have been coded for the explicit use of RDMA services typically exploit the full value proposition via RDMA Reads and Writes.

RDMA Atomic operations allow one-sided atomic remote memory accesses. The InfiniBand RDMA standard defines 64-bit CompareAndSwap and FetchAndAdd operations, but over time vendors have extended the support to longer data fields and other variants such as the support for masked versions of the operations. Among others, RDMA Atomics are widely used by distributed lock applications and clustered database systems.

6.1.4 RDMA Scalability

The InfiniBand RDMA architecture follows very strict protocol layering and was designed with a hardware implementation in mind. However, not all RDMA implementations are created equal. One challenge is that of delivering RDMA services at scale. The offloaded nature of the RDMA model mandates per-flow context structures of considerable size. Reasonably, the more RDMA flows, the more state the RDMA NIC needs to maintain. It's not rare for consumer applications to demand tens and hundreds of thousands of RDMA flows with some even reaching the millions. One possible approach is that of caching context structures on the RDMA device itself while maintaining the complete state tables in host memory. When there is significant locality in RDMA flow usage, this type of solution can deliver adequate performance. However, for several deployment scenarios, context cache replacement may introduce significant performance jitter. With these use cases in mind, high-scale RDMA NICs have been designed with onboard dedicated context memory that can fit the entire RDMA protocol state.

6.1.5 RoCE

In addition to the software interface advantages of the RDMA model, InfiniBand dominated the high-performance clustering space, thanks to the performance advantage of its dedicated L1 and L2 that was at that time faster than that of the much more established Ethernet. A few years later, as 10Gig and 40Gig Ethernet turned out to be more common, it became apparent that the same benefits of RDMA could be obtained over the ubiquitous Ethernet network. Two standards emerged to address this approach, iWARP and RoCE. iWARP defined RDMA semantics on top of TCP as the underlying transport protocol [6]. Meanwhile, the IBTA defined RoCE by merely replacing the lower two layers of its vertical protocol stack with those of Ethernet [7].

The first version of RoCE did not have an IP layer and hence wasn't routable. This happened at pretty much the same time as FCoE was being defined under the similar vision of flat L2-based networks. Eventually, it became clear that Routable RoCE was required, and the spec further evolved to become RoCEv2, defining the encapsulation of RDMA on top of UDP/IP [8]. In addition to making RoCE routable, RoCEv2 leverages all the advantages of the IP protocol, including QoS marking (DSCP) and congestion management signaling (ECN).

The resulting architecture is depicted in Figure 6-11. The first column from the left represents the classical implementation of InfiniBand RDMA. The middle column is ROCEv1 (no IP header, directly over Ethernet), and the third column is RoCEv2 in which the InfiniBand transport layer is encapsulated in IP/UDP.

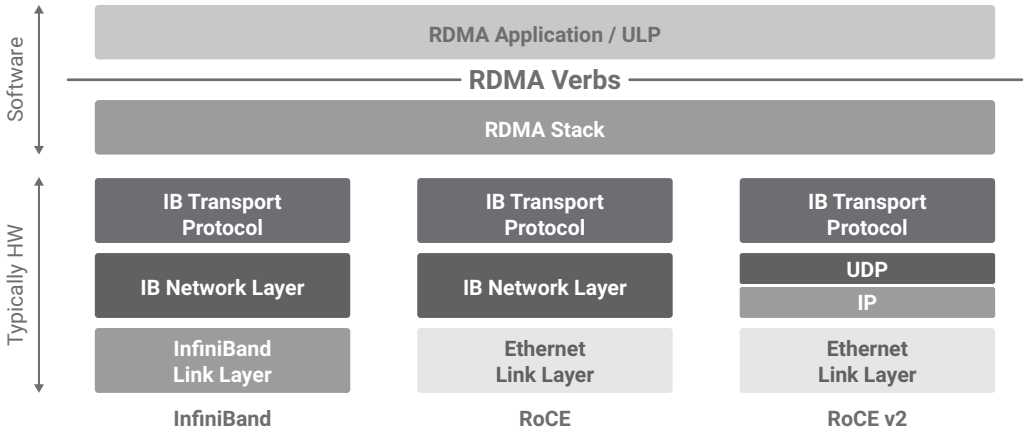

FIGURE 6-11 RDMA over Ethernet Protocol Stack Evolution

6.1.6 RoCE vs iWARP

iWARP and RoCE are functionally similar but not identical in terms of the RDMA services that they offer. There are some semantic differences between the two protocols and a few functional aspects of RoCE that were not covered by the iWARP specification. In practice, iWARP didn't attain a wide adoption in the market, due in part to the fact that the RoCE specification is much more suitable for efficient hardware implementations. Figure 6-12 illustrates the protocol stack differences between the two.

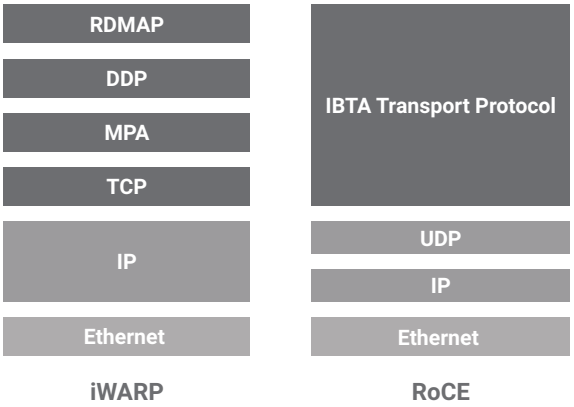

FIGURE 6-12 RoCE versus iWARP

6.1.7 RDMA Deployments

RDMA was initially deployed in high-performance computing (HPC) where ultra-low latency and high bandwidth dominate the requirements. The cost/performance advantages were so drastic that in a few

years InfiniBand debunked well-established proprietary networks and became the de facto standard for top-end HPC clusters; see Figure 6-13. This broad adoption in the HPC space fueled the development of high-end RDMA and made it a more mature technology [9].

Embedded platforms were also among the early adopters of RDMA technology. Examples included data replication in storage appliances, media distribution, and others. The unrivaled performance of RDMA and the fact that it was an open standard with a visible roadmap into the future made the solution very attractive to this market that traditionally aims for long-term investments in technology.

One other major milestone in the growth of RDMA was the adoption of the technology at the core of Oracle's flagship clustered database [10]. This use case was instrumental in bringing RDMA into the enterprise network.

Another wave of early adoption came when the financial market identified a significant edge for market data delivery systems in the ultra-low latency of InfiniBand.

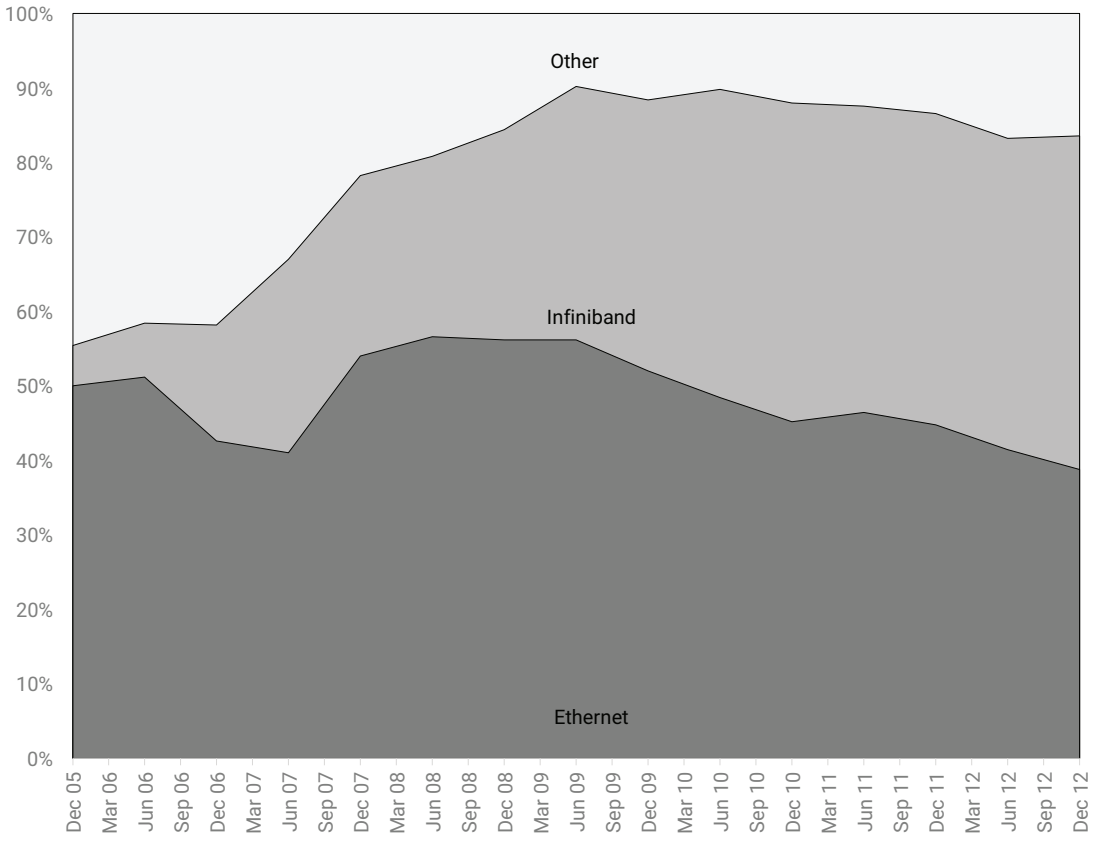

FIGURE 6-13 InfiniBand and Ethernet Shares (Data courtesy of top500.org)

years InfiniBand debunked well-established proprietary networks and became the de facto standard for top-end HPC clusters; see Figure 6-13. This broad adoption in the HPC space fueled the development of high-end RDMA and made it a more mature technology [9].

Embedded platforms were also among the early adopters of RDMA technology. Examples included data replication in storage appliances, media distribution, and others. The unrivaled performance of RDMA and the fact that it was an open standard with a visible roadmap into the future made the solution very attractive to this market that traditionally aims for long-term investments in technology.

One other major milestone in the growth of RDMA was the adoption of the technology at the core of Oracle's flagship clustered database [10]. This use case was instrumental in bringing RDMA into the enterprise network.

Another wave of early adoption came when the financial market identified a significant edge for market data delivery systems in the ultra-low latency of InfiniBand.

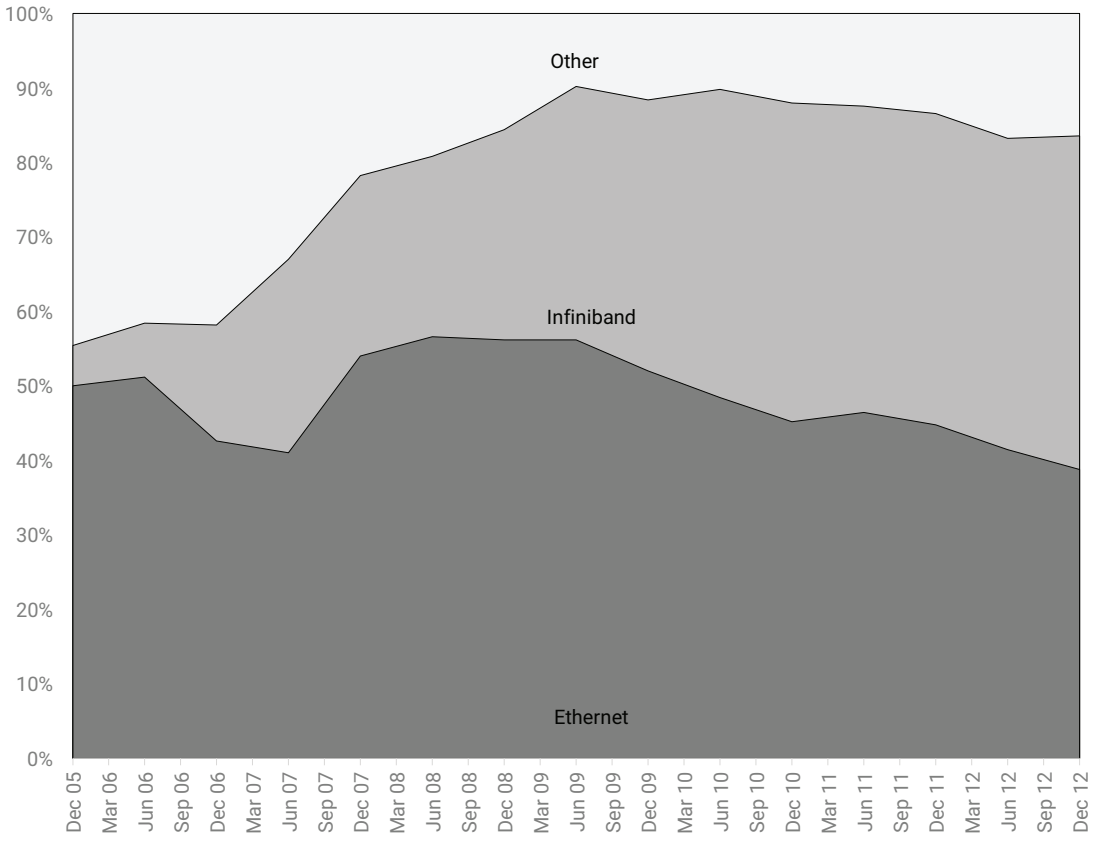

FIGURE 6-13 InfiniBand and Ethernet Shares (Data courtesy of top500.org)

Currently, RoCEv2 offers the RDMA value proposition on Ethernet networks with applications covering remote storage protocols, database clustering, nonvolatile memory, artificial intelligence, scientific computing, and others.

Mellanox Technologies has been one of the most significant contributors to the InfiniBand technology; Figure 9-6 shows a Mellanox ConnectX-5 Dual-Port Adapter Supporting RoCEv2 over two 100Gb/s Ethernet ports.

6.1.8 RoCEv2 and Lossy Networks

Due to the simplicity of its transport protocol, RDMA performance is quite sensitive to packet losses. Very much like FCoE, the RDMA protocol was designed with the assumption of an underlying lossless network. The network was indeed lossless when RDMA was running on InfiniBand Fabrics and, when RoCE was defined, the IEEE 802.1 was in the process of standardizing a set of Ethernet specs usually referred to as Data Center Bridging (DCB), which, among other things, offered non-drop (also known as "lossless") priorities through Per-Priority Flow Control (PFC). RoCE and FCoE were both originally defined to leverage PFC to satisfy the lossless requirement [11].

With PFC, each of the eight priorities (from 0 to 7) can be configured as lossless or lossy. Network devices treat the lossy priorities as in classical Ethernet and use a per-priority pause mechanism to guarantee that no frames are lost on the lossless priorities.

Figure 6-14 illustrates this technique: Priority 2 is stopped by a pause frame to avoid overflowing the queue on the switch side and thus losing packets.

Alongside PFC, IEEE has standardized DCBX (Data Center Bridging eXchange) in an attempt to make I/O consolidation deployable on a larger scale. DCBX is a discovery and configuration protocol that guarantees that both ends of an Ethernet link are configured consistently. DCBX works to configure switch-to-switch links and switch-to-host links correctly to avoid configuration errors, which can be very difficult to troubleshoot. DCBX discovers the capabilities of the two peers at each end of a link: It can check for consistency, notify the device manager in the case of configuration mismatches, and provide a basic configuration in case one of the two peers is not configured. DCBX can be configured to send conflict alarms to the appropriate management stations.

While these DCB techniques can be appropriately deployed and do not present any conceptual flaws, there has consistently been significant concerns from the IT community that prevented their mainstream deployment. Criticism has been fueled by some unfortunate experiences, lack of tools for consistency checks and troubleshooting, and distance limitation both in terms of miles and hops.

In light of the preceding, it became quite clear that PFC was not poised to become widespread and hence FCoE and RoCEv2 would need an alternative solution. In other words, to gain wide adoption, RDMA and Remote Storage over Ethernet need to be adapted to tolerate a non-lossless network. For storage, the real solution comes from NVMe and NVMe over Fabrics (NVMe-oF), which runs on top of RDMA or TCP and leverages their respective transport protocol characteristics as described in section 6.2.2. For RoCEv2, modifications to the protocol have been implemented as described in the following section.

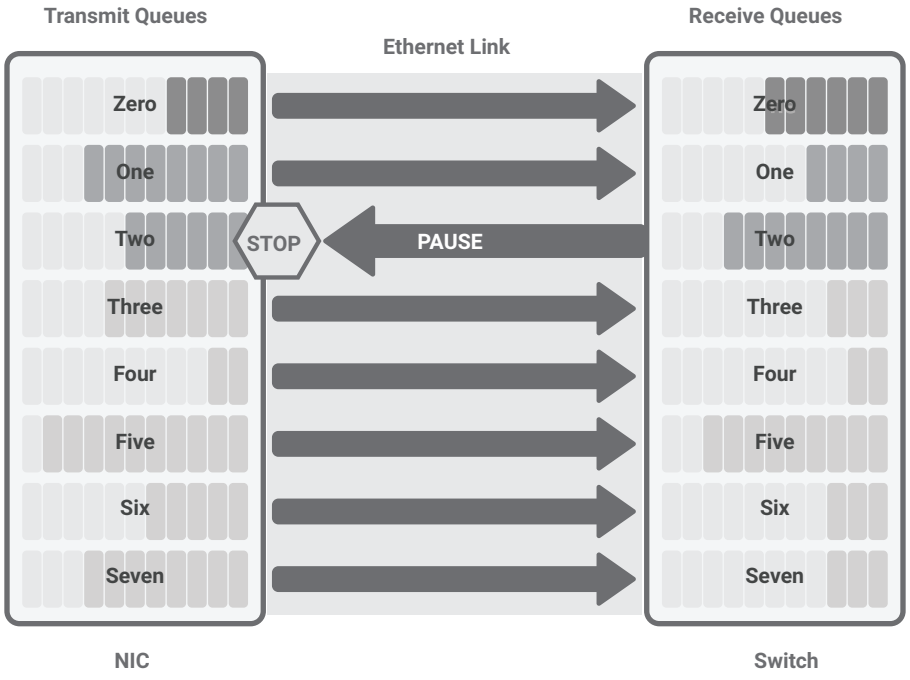

FIGURE 6-14 Priority Flow Control

To properly understand the scope of the problem, it's crucial to establish what defines a *lossy network*. In a nutshell, in modern Ethernet networks, there are two leading causes for packet losses.

The first type of packet drops is literally produced by the forces of nature (for example, alpha particles) that every once in a while cause a bit flip on a packet while it's being transmitted or stored in a network device. This data corruption will be detected by CRCs and will result in the packet's being dropped. In modern data centers and enterprise networks, the probability for these events is very low. This kind of packet loss does occur on the so-called lossless networks; but given the low occurrence rate and the fact that it typically affects a single packet, simple retransmission schemes can effectively overcome it.

The second type of loss comes as a result of congestion. When there is contention on the output port of a network device and this contention lasts until its buffers fill up, the device will have no choice but to start dropping packets. This congestion-induced loss is much more severe and is the actual cause for significant reductions in performance. As described earlier, in a "lossless" network, this problem is addressed with link layer flow control that effectively prevents packets from arriving at a device unless there is enough buffer space to store them.

The strategy to solve the second problem, without using any link-layer flow control, centers around two complementary approaches. First, reduce contention by applying congestion management. With this technique, upon buffer buildup, a closed loop scheme controls injection at the source, thereby reducing the contention and minimizing the cases of packet drop. With congestion reduced to a minimum, the

second technique focuses on the efficient retransmission of the few remaining dropped packets. These packet losses are inevitable, but fewer, thanks to congestion management.

Specifically, with RoCEv2, Data Center Quantized Congestion Notification (DCQCN) [12] congestion management has been deployed to minimize drops and was shown to greatly improve overall network performance. The components of DCQCN are depicted in Figure 6-15. DCQCN relies on explicit congestion marking by switches using the ECN bits in the IP header [13]. Switches detect congestion through monitoring of their queue sizes. When a queue size crosses a programmable threshold, switches start to probabilistically mark packets indicating that these are crossing a congestion point. Upon receiving such marked packets, the destination node reflects this information to the source by using an explicit congestion notification packet (CNP). The reception of the CNP results in a reduction in the injection rate of the corresponding flow using a specified algorithm with configurable parameters. It has proven to be somewhat challenging to tune the DCQCN thresholds and parameters to obtain good congestion management performance under a wide variety of flows.

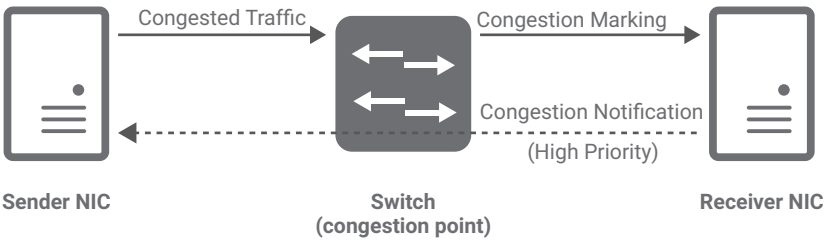

FIGURE 6-15 DCQCN

A newer approach under development utilizes a different congestion signaling scheme that is based on measuring latencies at the end nodes and reacts to queue buildup by detecting a latency change or spike; see Figure 6-16. This new scheme has faster response times than DCQCN and offers the additional benefit of not requiring the tuning of queue thresholds in the switches or other parameters. This mechanism is completely implemented at the end nodes without any reliance on switches.

Congestion management will significantly improve the performance of RDMA in lossy networks by minimizing congestion-induced packet loss. However, in practice it's impossible (actually, extremely inefficient) to guarantee absolutely no drops under all possible conditions, no matter how good the congestion management scheme is. The reaction time for congestion management extends from the moment congestion is detected, through the point where the source reduces the injection rate, and until the reduced rate arrives back at the congested point. During this period, packets continue to accumulate at the higher rate, and it could so happen that buffers overflow and packets are dropped. This delayed reaction characteristic is typical of closed-loop control systems. The congestion management algorithm could be tuned for more aggressive injection rate changes, thereby inducing a quicker reaction, to mitigate the impact of control loop delay. However, an over-aggressive scheme would generally create a prolonged underutilization of the available bandwidth. In essence, there is a tradeoff between the

possibility of some packet drops versus utilization of all available bandwidth, and in general it's best to tune for minimal (but non-zero) drops.

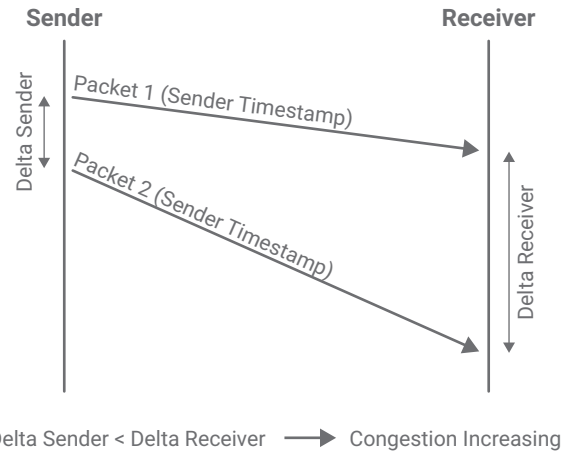

FIGURE 6-16 One-way Latencies

Even with very effective and adequately tuned congestion management, some packets may be dropped. At that point, it's the job of the communication protocol to deal with recovery. The transport layer of the RDMA protocol, as currently defined in the RoCEv2 standard, utilizes a very simplistic approach for this task. Upon detection of a missing packet, the receiver sends a NAK code back to the sender indicating the sequence number of the missed packet. The sender rolls back its state and resends everything from that point on. This "Go Back N" approach was conceived with the assumption of an underlying lossless fabric, and it's not ideal in the presence of congestion-induced drops.

To address this limitation, modern RoCEv2 implementations introduced a selective retransmission scheme that is much better at recovering from losses; see Figure 6-17. The main goal is to recover faster and conserve network bandwidth by resending only the lost packets.

Selective retransmission has been implemented in TCP for quite some time. The resending of individual packets from the source is quite straightforward. The challenge in RDMA networks is mainly at the receiver and has to do with the protocol requirement for in-order processing of inbound packets. For example, a multipacket RDMA Write message only carries the destination address in its first packet. If that packet is lost, subsequent ones can't be placed into their intended memory destinations. Attempts have been discussed to modify the message semantics to allow for out-of-order processing of inbound packets. However, such an approach would have a significant impact on existing RDMA protocol semantics that would be visible to consumer applications. A much more friendly approach (see Figure 6-18) completely preserves existing semantics and is implemented with the help of a staging buffer that is used to temporarily store subsequent packets until the lost ones arrive after being resent. RDMA NIC architectures that support efficient on-NIC memory are particularly suitable for this type of solution.

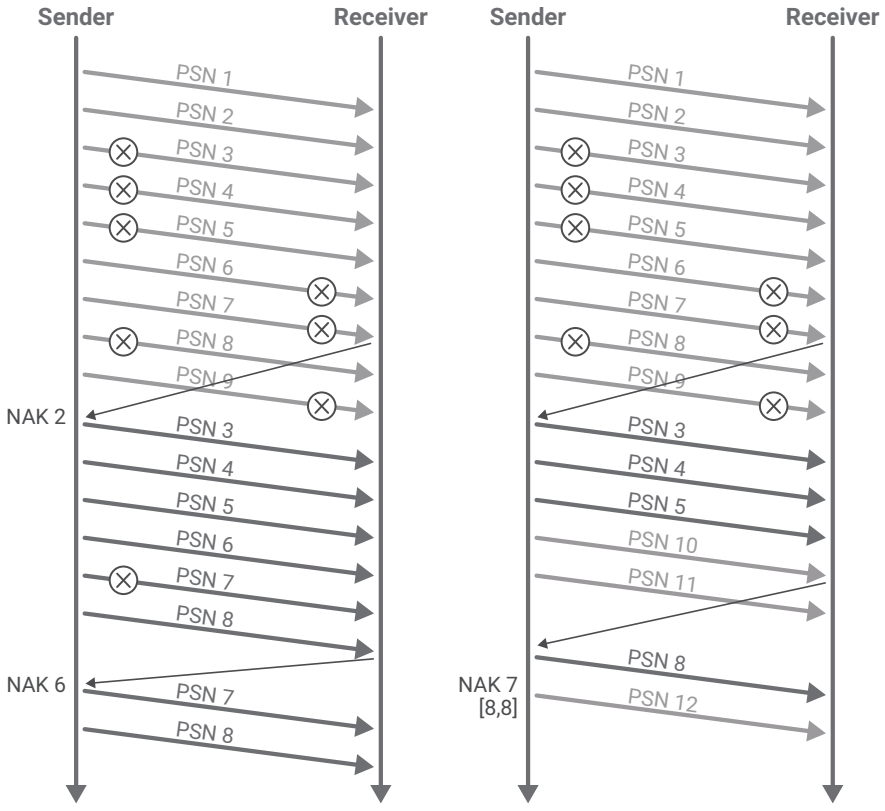

FIGURE 6-17 Go Back N versus Selective Retransmission

As described earlier, the RDMA transport protocol detects missing packets at the target and utilizes explicit NAKs to trigger immediate resends. With selective retransmission, this scheme is efficient except for the case when the lost packets are the last in the message (also known as tail drops), as there are no subsequent packets to detect the loss and trigger the NAK. For this case, the transport protocol relies on a timeout at the sender that kicks in when there are outstanding unacknowledged packets for much longer than expected. However, to prevent unnecessary retransmissions due to latency variations, this timeout is intended to be set to a value that is much higher than the typical network round trip. The effect is that, in case of a tail drop, the completion time of the message is severely impacted. Some RDMA implementations send an unsolicited ACK from the target back to the source, when no further packets are being received on an active flow, to improve the recovery time in the presence of a tail drop; see Figure 6-19. The receiver is in a much better position to eagerly detect the potential loss of a tail in a way that is entirely independent of the fabric round trip latency.

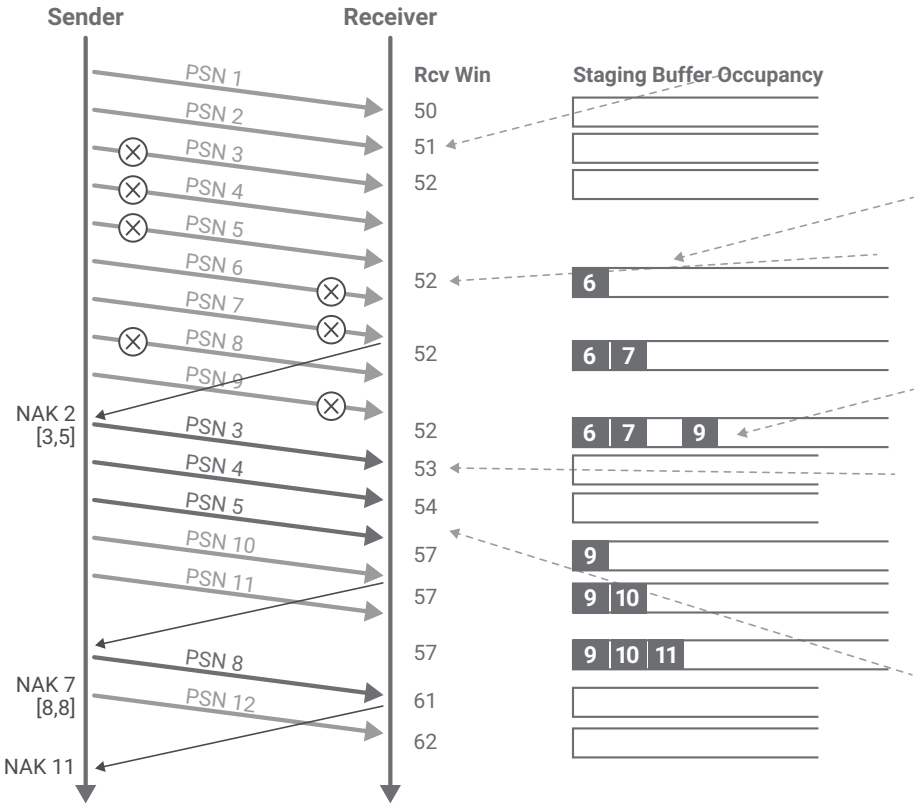

FIGURE 6-18 Staging Buffer Cost of Selective Retransmission

6.1.9 Continued Evolution of RDMA

As it usually happens with technology, RDMA deployment experiences prompted further evolution of the protocol. In this context, the vendor's innovation sets the pace. When adopted by the market, some of the new extensions become candidates for standardization at the IBTA, which has remained active for more than 20 years. Recent areas of activities are discussed next.

Control Plane Acceleration

We have described the mainstream data plane of the RDMA protocol that has been heavily optimized and tuned for maximum performance. Meanwhile, the control plane has been pointed out as a limiting factor for some real-life applications. In particular, with the one-connection-per-flow model that is common with RDMA, the cost of connection setup can become a burden. It is especially true for large deployments or those with very dynamic connection characteristics. Another aspect of RoCE that has been a consistent complaint is the cost of memory registration. This operation involves a mandatory

interaction with the OS and typically requires careful synchronization of RDMA NIC state. Some new RDMA implementations optimize these operations, and near-data-plane-performance versions of memory registration and connection setup have become available.

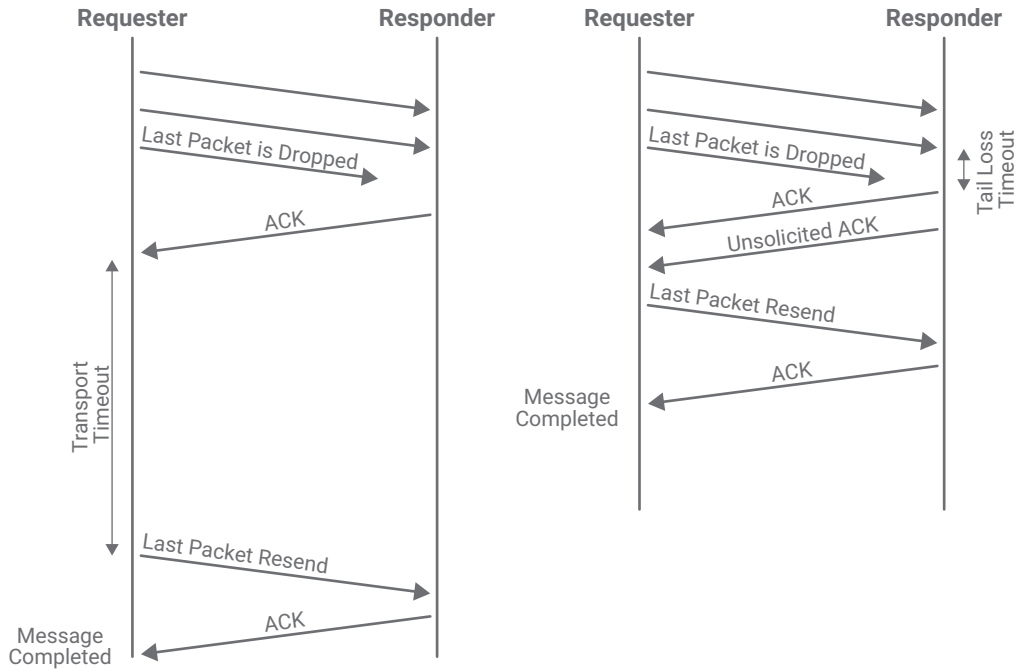

FIGURE 6-19 Tail Drop Recovery

Support for Remote Nonvolatile Memory Accesses

The recent proliferation of new nonvolatile memory technologies has opened up opportunities for a new range of applications and changed the paradigm for many others. NVDIMMs with Load/Store access semantics are adopting the form of a new storage tier. The SNIA has been working to define programming models for these new technologies [14], and the IBTA is extending the RDMA protocol with explicit support for the unique characteristics of explicit nonvolatile memory accesses across the network. Specifically, new RDMA operations will allow remote NVM commits that, once completed at the source, will virtually guarantee that data has been safely delivered to the persistence domain on the target side while maintaining the one-sided execution that is crucial to the RDMA model.

Data Security

Concerning data security, the RDMA protocol covers all aspects of interprocess protection and memory access checks at the end nodes. However, as far as data in transit is concerned, the standard doesn't define data encryption or cryptographic authentication schemes. Still, by being a layered protocol that runs on UDP/IP, there are several ways to implement encrypted RDMA, such as IPSec or DTLS.

Modern RDMA implementations using domain-specific hardware are particularly efficient at this kind of task and can achieve wire rate encryption/decryption for 100GE links.

6.2 Storage

Introduced in 1986, SCSI (Small Computer System Interface) became the dominant storage protocol for servers. SCSI offered a variety of physical layers, from parallel to serial, mostly limited to very short distances.

The demand for longer reach, as well as the opportunity to disaggregate storage, motivated the creation of Fibre Channel [15], a dedicated storage network designed to carry SCSI across a switched fabric and connect to dedicated appliances providing storage services to many clients.

Over time, the network consolidation trend prompted the definition of further remote storage solutions for carrying SCSI across nondedicated networks. iSCSI [16] and FCoE [17] are attempts to converge storage and networking over Ethernet, whereas SRP [18] and iSER [19] are designed to carry SCSI over RDMA; see Figure 6-20. In all cases, the goal is to eliminate the requirement for a dedicated storage network (for example, Fibre Channel). Additionally, storage consolidation has further redefined and disrupted the storage landscape through "hyper-converged infrastructure" (HCI) and "hyper-converged storage" offerings. These are typically proprietary solutions defined and implemented by the top-tier cloud providers [20].

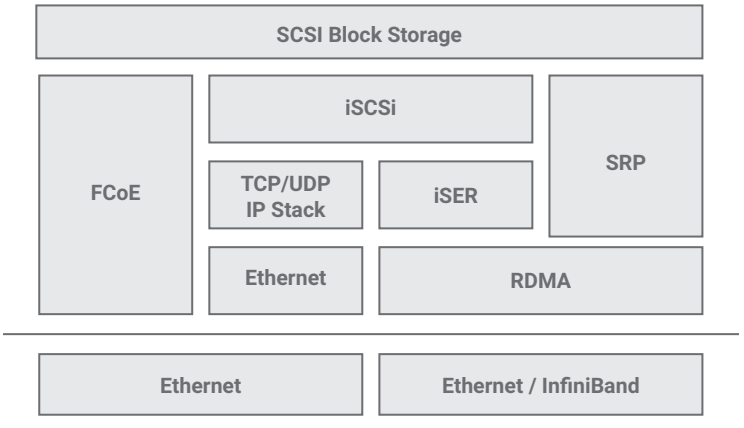

FIGURE 6-20 Remote SCSI Storage Protocols

6.2.1 The Advent of SSDs

Since 2010, solid state drives (SSDs) started to become economically viable. They provide superior performance compared to rotating media hard disks. To fully exploit SSD capabilities, a new standard called Non-Volatile Memory express (NVMe) has been created as a more modern alternative to SCSI.

NVMe [21] is an open logical device interface specification for accessing nonvolatile storage media attached via a PCIe. NVMe has been designed to take advantage of the low latency and high internal parallelism of flash-based storage devices. As a result, NVMe reduces I/O overhead and brings various performance improvements relative to previous logical-device interfaces, including multiple, long command queues and reduced latency. NVMe can be used for SSD, but also for new technologies such as 3D Xpoint [22] and NVDIMMs [23].

6.2.2 NVMe over Fabrics

As was the case with SCSI, a remote storage flavor of NVMe was also defined [24]. NVMe over Fabrics (NVMe-oF) defines extensions to NVMe that allow remote storage access over RDMA and Fibre Channel. Most recently, NVMe/ TCP was also included in the standard (see Figure 6-21).

It seems that the Fibre Channel flavor of NVMe-oF will not be particularly relevant, because it requires maintaining a separate FC network or running over FCoE. Given the benefits already explored in previous sections, the highest performant solution appears to be NVMe over RDMA. The NVMe/TCP approach will likely gain wide acceptance, given the ubiquity of TCP and the opportunity to leverage most of the value proposition of NVMeoF with minimal impact to the datacenter network.

6.2.3 Data Plane Model of Storage Protocols

Remote storage has been in place for a couple of decades. As one looks in more detail, there is a common pattern in the data plane of most remote storage protocols. I/O operations start with a request from the storage client, which is typically delivered to the remote storage server in the payload of a network message. The storage server usually manages the outstanding I/O requests in a queue from which it schedules for execution. Execution entails the actual data transfer between the client and the server and the actual access to the storage media. For I/O reads (that is, read from storage), the server will read from the storage media and then deliver the data through the network into buffers on the client side that were set aside for that purpose when the I/O operation was requested. For I/O writes, the server will access the client buffers to retrieve the data across the network and store it in the storage media when it arrives (a pull model); see Figure 6-22. After the I/O has been completed, the server will typically deliver a completion status to the client in the payload of a message through the network.

Some remote storage protocols also include a flavor of I/O writes where data is pushed into the server directly with the I/O request; see Figure 6-23. The benefit of this optimization is the saving of the data delivery roundtrip. However, this scheme poses a challenge to the storage server as it needs to provision queues for the actual incoming data and, hence, when available, this push model is only permitted for short I/O writes.

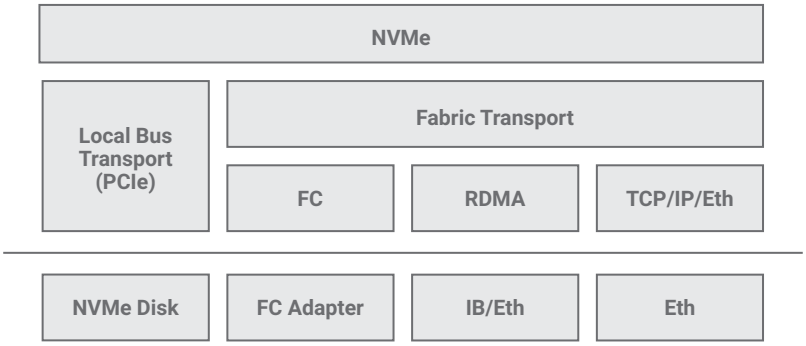

FIGURE 6-21 NVMe-oF

RDMA operations present clear benefits when applied towards data access and I/O requests. With RDMA, the client-side CPU need not be involved in the actual delivery of data. The server asynchronously executes storage requests, and the status is reported back to the client, after the requests are completed. This natural alignment prompted the development of RDMA-based remote storage protocols that would work over SCSI. Initially, the SCSI RDMA Protocol (SRP) [18] was defined, followed sometime later by the IETF standardized iSCSI Extensions for RDMA (iSER) [19], which preserves the iSCSI model by replacing the underlying TCP with RDMA. Finally, NVMe-oF was defined to leverage the NVMe model on RDMA fabrics, given the current momentum and the maturity of RDMA technologies. NVMe-oF seems poised to gain a much broader adoption than either SRP or iSER.

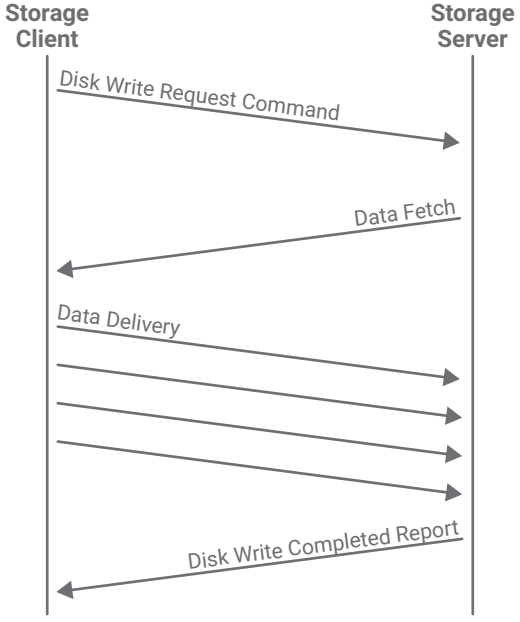

FIGURE 6-22 Example of Storage Write

FIGURE 6-23 Push Model

6.2.4 Remote Storage Meets Virtualization

Economies of scale motivated large compute cluster operators to look at server disaggregation. Among server components, storage is possibly the most natural candidate for disaggregation, mainly because the average latency of storage media access could tolerate a network traversal. Even though the new prevalent SSD media drastically reduces these latencies, network performance evolution accompanies that improvement. As a result, the trend for storage disaggregation that started with the inception of Fibre Channel was sharply accelerated by the increased popularity of high-volume clouds, as this created an ideal opportunity for consolidation and cost reduction.

Remote storage is sometimes explicitly presented as such to the client; see Figure 6-24. This mandates the deployment of new storage management practices that affect the way storage services are offered to the host, because host management infrastructure needs to deal with the remote aspects of storage (remote access permissions, mounting of a specific remote volume, and so on).

In some cases, exposing the remote nature of storage to the client could present some challenges, notably in virtualized environments where the tenant manages the guest OS, which assumes the presence of local disks. One standard solution to this problem is for the hypervisor to virtualize the remote storage and emulate a local disk toward the virtual machine; see Figure 6-25. This emulation abstracts the paradigm change from the perspective of the guest OS.

However, hypervisor-based storage virtualization is not a suitable solution in all cases. For example, in bare-metal environments, tenant-controlled OS images run on the actual physical machines with no hypervisor to create the emulation. For such cases, a hardware emulation model is becoming more common; see Figure 6-26. This approach presents to the physical server the illusion of a local hard disk. The emulation system then virtualizes access across the network using standard or proprietary

remote storage protocols. One typical approach is to emulate a local NVMe disk and use NVMe-oF to access the remote storage.

One proposed way to implement the data plane for disk emulation is by using so-called "Smart NICs"; see section 8.6. These devices typically include multiple programmable cores combined with a NIC data plane. For example, Mellanox BlueField and Broadcom Stingray products fall into this category.

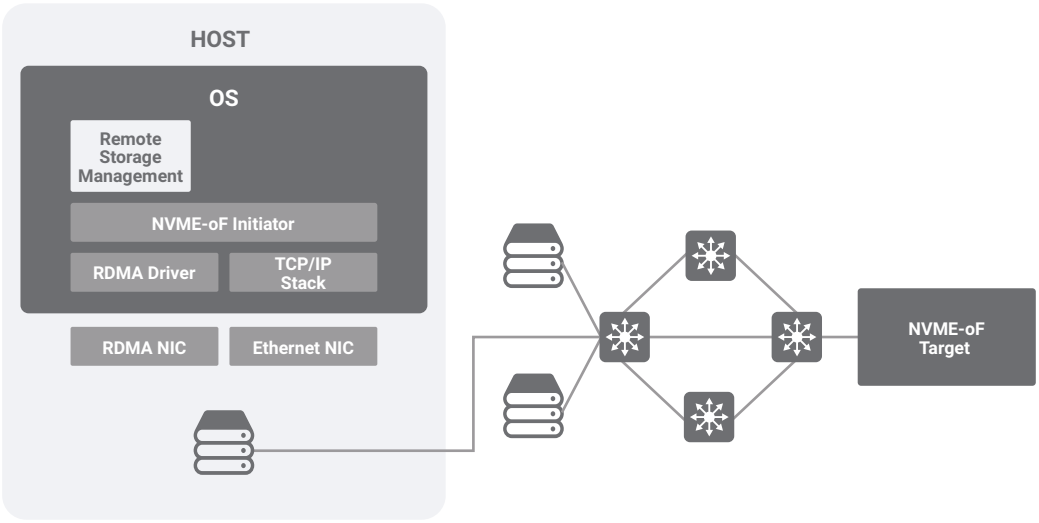

FIGURE 6-24 Explicit Remote Storage

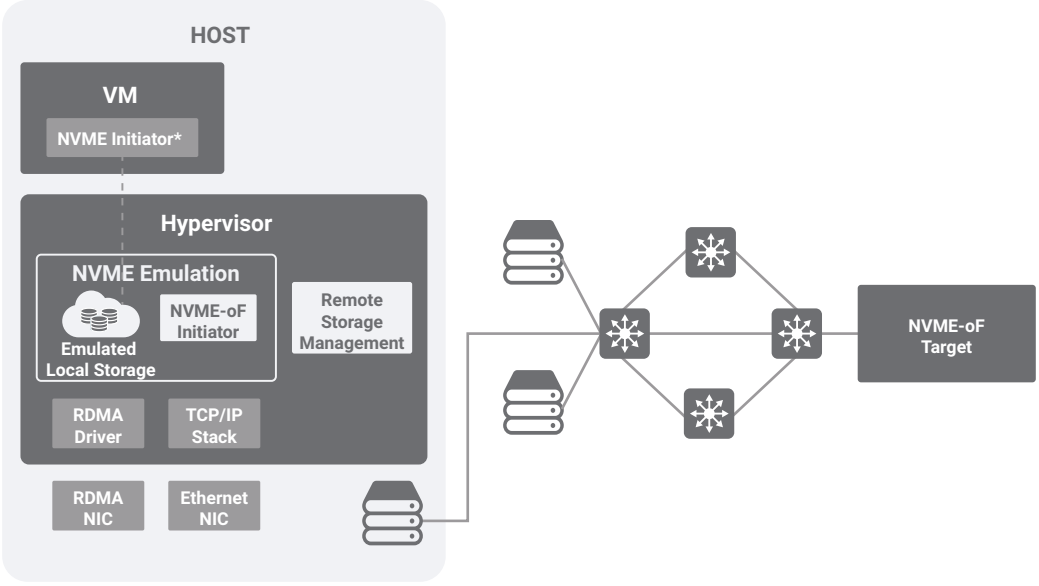

FIGURE 6-25 Hypervisor Emulates Local Disk Towards Guest OS

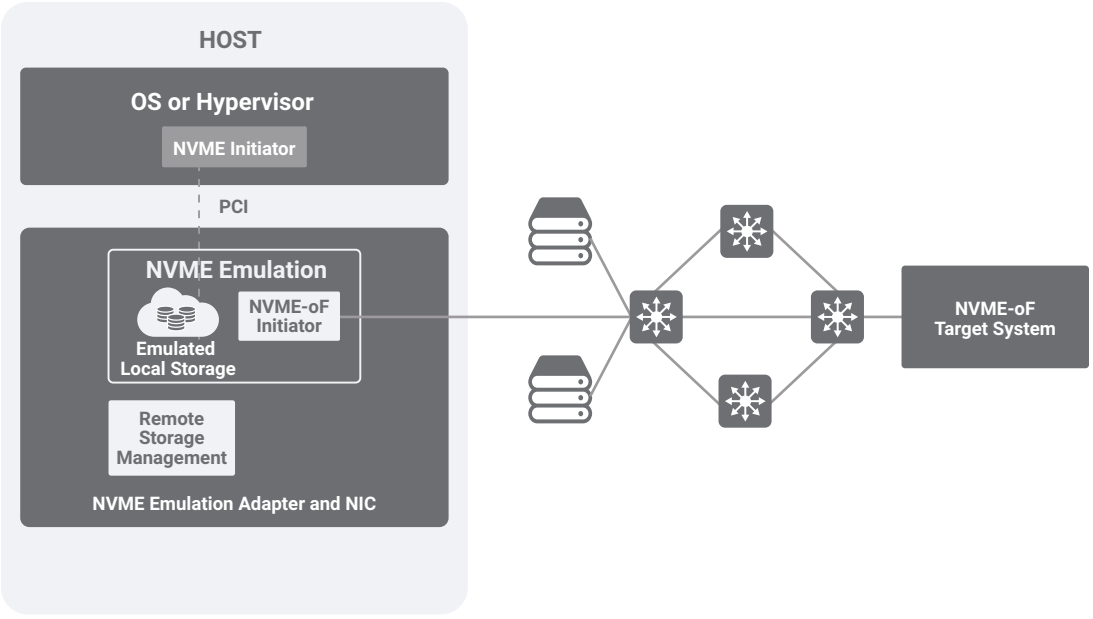

FIGURE 6-26 NIC-Based NVMe Emulation

However, as the I/O demand increases, this kind of platform may struggle to deliver the required performance and latency, and the scalability of the solution, especially in terms of heat and power, may become a challenge. An alternative approach favors the use of domain-specific hardware instead of off-the-shelf programmable cores. We discuss this further in the context of distributed storage services.

6.2.5 Distributed Storages Services

It has been quite a while since storage was merely about reading and writing from persistent media. Modern storage solutions offer multiple additional associated services. Table 6-1 summarizes the more prevalent ones. We discuss some of them in the following sections.

TABLE 6-1 Storage Services by Type

Security and Integrity	Efficiency	Reliability & Availability
■ Encryption and Decryption	■ Compression and Decompression	■ Replication
■ Key Management	■ Deduplication	■ Mirroring
■ Checksum		■ Erasure Coding
		■ Striping
		■ Snapshotting

6.2.6 Storage Security

In today's world, encryption of data is of primary importance. Multiple encryption schemes have been devised over the years for the encryption of data at rest. The recently approved IEEE standard 1619-2018 defines Cryptographic Protection on Block Storage Devices. The algorithm specified is a variant of the Advanced Encryption Standard, AES-XTS (XOR–Encrypt–XOR, tweaked-codebook mode with ciphertext stealing). The AES-XTS block size is 16 bytes, but with ciphertext stealing, any length, not necessarily a multiple of 16 bytes, is supported. AES-XTS considers each storage data block as the data unit. Popular storage block sizes are 512B, 4KB, and 8KB. Each block can also include metadata/Protection Info (PI) and thus be larger than the aforementioned sizes by anywhere from 8 to 32 bytes.

6.2.7 Storage Efficiency

A crucial service of modern storage technology that directly ties to IT costs is data compression. Every day, vast amounts of data are generated and stored—logs, web history, transactions, and so on. Most of this data is in a human-consumable form and is significantly compressible; therefore, doing so allows for better utilization of storage resources. With the advent of SSDs, which offer tremendous performance but have a higher cost per GB than HDDs, data compression becomes even more compelling.

Most of the data compression techniques used in practice today are variants of the Lempel-Ziv (LZ) compression scheme with or without Huffman Coding. These techniques inevitably trade off compression ratio for compression and decompression speed and complexity. For example, deflate compression, the most popular and open source compression in use, implements nine levels, progressively going from speed-optimized (level 1) to ratio-optimized (level 9).

Compression speed, compression ratio, and decompression speed of various algorithms are compared in [25]:

- Fast compression techniques such as LZ4 and Snappy focus on speed optimization by eliminating Huffman Coding.
- The fastest software-based compression algorithm (LZ4) offers a compression ratio of 2 on Silesia benchmark.
- The highest compression ratio of 3 comes from deflate (level 6).

Data deduplication is another data-reduction technique that has become popular in storage platforms. The benefit stems from the reality that it's extremely common for multiple copies of the same data to be stored in the same system. Consider, for example, the OS images for instances of multiple VMs; a large presentation emailed to a work team or multimedia files delivered to social media groups. The explicit goal is to store a single copy of the common data and avoid the waste of storage space on copies.

Data deduplication involves comparing a new data block against all previously stored ones in an attempt to discover a potential copy. Byte-by-byte comparison is not an efficient solution. The most common technique relies on calculating a short digest of the data block and comparing it against the other block's digests. Cryptographic hashes are used as digests thanks to their very low collision probability. For example, a Secure Hash Algorithm 2 (SHA2) 512-bit digest of a 4KB data block is just 64B long but has a collision probability of 2.2×10^{-132}! Different storage systems have chosen different cryptographic hashes, but the most common is the NIST-approved SHA family: SHA1, SHA2, SHA3. SHA2 and SHA3 offer digest sizes of 256, 384, or 512 bits.

6.2.8 Storage Reliability

Resiliency to media failure is among the essential features of a storage system. In its simplest form, this can be achieved by storing two copies of each block on two different disks, and consequently duplicating the cost of the media. By calculating a parity block over N data blocks and then storing the N data blocks and the one parity block on N+1 different drives, the cost can be reduced while tolerating a single drive failure (including a failure to the parity drive). The Redundant Array of Independent Disks standard 5 (RAID5) implements this scheme, but it is less popular these days due to vulnerabilities during rebuild times and the increased size of the individual disks. Similarly, RAID-6 calculates two different parity blocks and can tolerate two drive failures. This approach, known as erasure coding, can be generalized. A scheme to tolerate K drive failures requires calculating K parity blocks over N data blocks and storing N+K blocks on N+K drives. Reed-Solomon [25] is a widespread erasure coding scheme used by many storage systems.

6.2.9 Offloading and Distributing Storage Services

Traditionally, most storage services, like the ones described previously, were provided by the monolithic storage platform. The disaggregation of storage using local disk emulation presents an opportunity for further innovations. Storage services can now be implemented in three different components of the platform: the disk emulation adapter at the host, the front-end controller of the storage server, or the storage backend; see Figure 6-27.

Deployment of storage services using domain-specific hardware within the host platform aligns well with the benefits of other distributed services discussed in previous chapters. Storage compression, for example, appears to be an excellent candidate for this approach. Implementation at the end node versus an appliance offers the benefit of scalability, and, in addition, the advantage of reduced network bandwidth as the storage data is delivered in compressed form.

However, not all services are equally suitable for this approach. In particular, client-side offloads are more efficient when context information is unchanging per flow, which is often the case for compression. In contrast, reliability services that involve persistent metadata schemes are a better fit for the storage server side. Some services could benefit from mixed approaches. For example, deduplication would be naturally implemented by the storage system because it requires the digest of all stored

blocks. However, digest calculation, which is a compute-intensive task, could be very well offloaded to the host adapter, attaining a benefit in scalability.

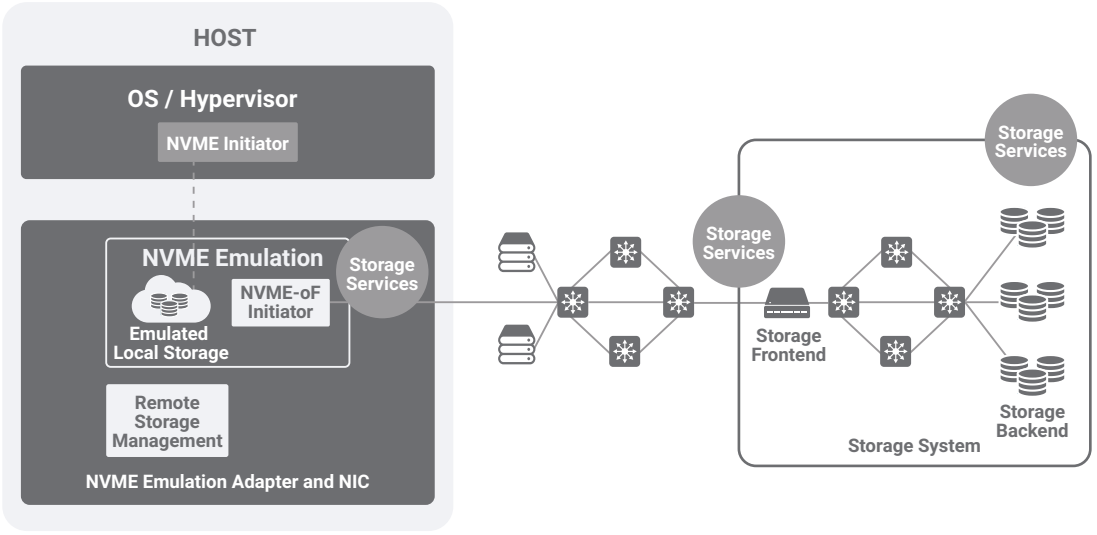

FIGURE 6-27 Distributed Storage Services

With regard to security, distributed storage presents the additional need to protect data in transit. For this purpose, there is no choice but to involve the end node. In cases where the remote storage server implements encryption of data at rest, data in transit can be secured using traditional network packet encryption schemes (for example, IPSec, DTLS). Alternatively, AES-XTS encryption for data-at-rest could be implemented directly at the host adapter, thereby covering the protection of data in transit and relieving the storage platform from the work of encryption/decryption. The AES-XTS scheme is hardware-friendly and particularly suitable for offload to the disk emulation controller on the host side. Encryption of data at rest performed by the end node offers the additional benefit of total data protection under the complete control of the data owner. With encryption performed at the source, services such as compression that require visibility of the data must also be implemented at the endnode.

6.2.10 Persistent Memory as a New Storage Tier

The emergence of nonvolatile memory devices also created the notion of a new storage tier. These new devices, some of which are packaged in DIMM form factors, have performance characteristics that are closer to that of DRAM with capacities and persistency that is more typical of storage. This trend promoted the development of new APIs for efficient utilization of these devices. The Storage Networking Industry Association (SNIA) developed a programming model [26] that covers local access and is now being extended to support remote NVM devices in line with the RDMA protocol extensions discussed in section 6.1.9.

6.3 Summary

As with application services, the demand for cost-efficient scalability and flexibility requires a paradigm change in the deployment of infrastructure services in enterprise networks and clouds. Domain-specific hardware for distributed services processing deployed at the end nodes or close by seems to offer a promising, scalable solution that strikes a good balance of flexibility and cost.

6.4 Bibliography

[1] Kronenberg, Nancy P. et al. "VAXclusters: A Closely-Coupled Distributed System (Abstract)." SOSP (1985), http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.74.727&rep=rep1&type=pdf

[2] Horst and Diego José Díaz García. "1.0 Introduction, 2.0 ServerNet Overview, 2.1 ServerNet I, ServerNet SAN I/O Architecture." https://pdfs.semanticscholar.org/e00f/9c7dfa6e2345b9a58a082a2a2c13ef27d4a9.pdf

[3] Compaq, Intel, Microsoft, "Virtual Interface Architecture Specification, Version 1.0," December 1997, http://www.cs.uml.edu/~bill/cs560/VI_spec.pdf

[4] InfiniBand Trade Association, http://www.infinibandta.org

[5] InfiniBand Architecture Specification, Volume 1, Release 1.3, https://cw.infinibandta.org/document/dl/7859

[6] RDMA Consortium, http://rdmaconsortium.org

[7] Supplement to InfiniBand Architecture Specification, Volume 1, Release 1.2.1, Annex A16, RDMA over Converged Ethernet (RoCE) https://cw.infinibandta.org/document/dl/7148

[8] Supplement to InfiniBand Architecture Specification, Volume 1, Release 1.2.1, Annex A17, RoCEv2 https://cw.infinibandta.org/document/dl/7781

[9] Top 500, Development over time, https://www.top500.org/statistics/overtime

[10] Paul Tsien, "Update: InfiniBand for Oracle RAC Clusters," Oracle, https://downloads.openfabrics.org/Media/IB_LowLatencyForum_2007/IB_2007_04_Oracle.pdf

[11] IEEE 802.1 Data Center Bridging, https://1.ieee802.org/dcb

[12] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. 2015. Congestion Control for Large-Scale RDMA Deployments. SIGCOMM Comput. Commun. Rev. 45, 4 (August 2015), 523–536. DOI: https://doi.org/10.1145/2829988.2787484, https://conferences.sigcomm.org/sigcomm/2015/pdf/papers/p523.pdf

[13] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168, DOI 10.17487/RFC3168, September 2001, https://www.rfc-editor.org/info/rfc3168

[14] SNIA, "NVM Programming Model (NPM)," https://www.snia.org/tech_activities/standards/curr_standards/npm

[15] FCIA, Fibre Channel Industry Association, https://fibrechannel.org

[16] Chadalapaka, M., Satran, J., Meth, K., and D. Black, "Internet Small Computer System Interface (iSCSI) Protocol (Consolidated)," RFC 7143, DOI 10.17487/RFC7143, April 2014, https://www.rfc-editor.org/info/rfc7143

[17] Incits, "T11 - Fibre Channel Interfaces," http://www.t11.org

[18] Incits "T10, SCSI RDMA Protocol (SRP)," http://www.t10.org/cgi-bin/ac.pl?t=f&f=srp-r16a.pdf

[19] Ko, M. and A. Nezhinsky, "Internet Small Computer System Interface (iSCSI) Extensions for the Remote Direct Memory Access (RDMA) Specification," RFC 7145, DOI 10.17487/RFC7145, April 2014, https://www.rfc-editor.org/info/rfc7145

[20] Google, "Colossus," http://www.pdsw.org/pdsw-discs17/slides/PDSW-DISCS-Google-Keynote.pdf

[21] NVM Express, http://nvmexpress.org

[22] Rick Coulson, "3D XPoint Technology Drives System Architecture," https:// www.snia.org/sites/default/files/NVM/2016/presentations/RickCoulson_All_the_Ways_3D_XPoint_Impacts.pdf

[23] Jeff Chang, "NVDIMM-N Cookbook: A Soup-to-Nuts Primer on Using NVDIMM-Ns to Improve Your Storage Performance," http://www.snia.org/sites/default/files/SDC15_presentations/persistant_mem/Jeff Chang-ArthurSainio_NVDIMM_Cookbook.pdf, Sep. 2015

[24] NVM Express, "NVMExpress over Fabrics," Revision 1.0, June 5, 2016, https://nvmexpress.org/wp-content/uploads/NVMe_over_Fabrics_1_0_Gold_20160605-1.pdf

[25] GitHib, "Extremely Fast Compression algorithm," https://github.com/lz4/lz4

[26] SNIA, "NVM Programming Model (NPM)," Version 1.2, https://www.snia.org/sites/default/files/technical_work/final/NVMProgrammingModel_v1.2.pdf