RDMA and Hardware Support

SIGCOMM Topic Preview 2018

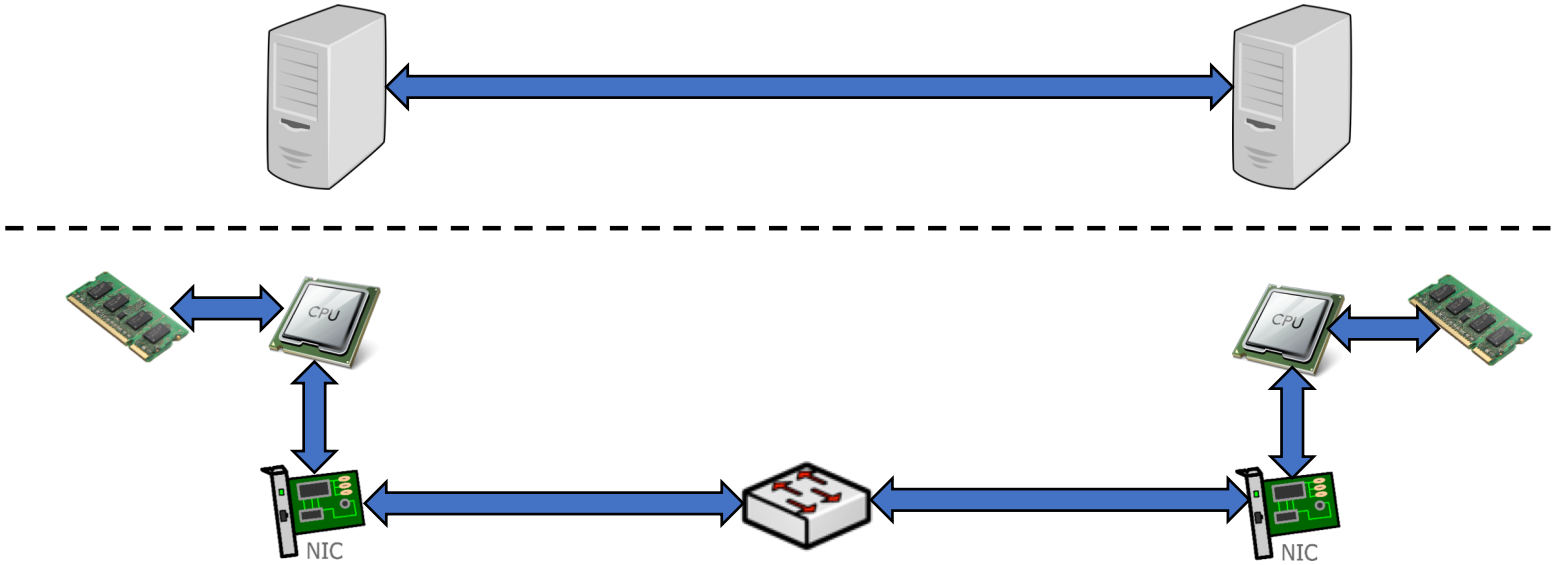
Yibo Zhu

Microsoft Research

The (Traditional) Journey of Data

How app developers see the network

Under the hood



- This architecture had been working well until ~5 years ago
  - Ethernet: 10Mbps → 100Mbps → 1Gbps → 10Gbps (until 2013 @ Microsoft)
  - CPU: Moore's Law

The End of Moore's Law

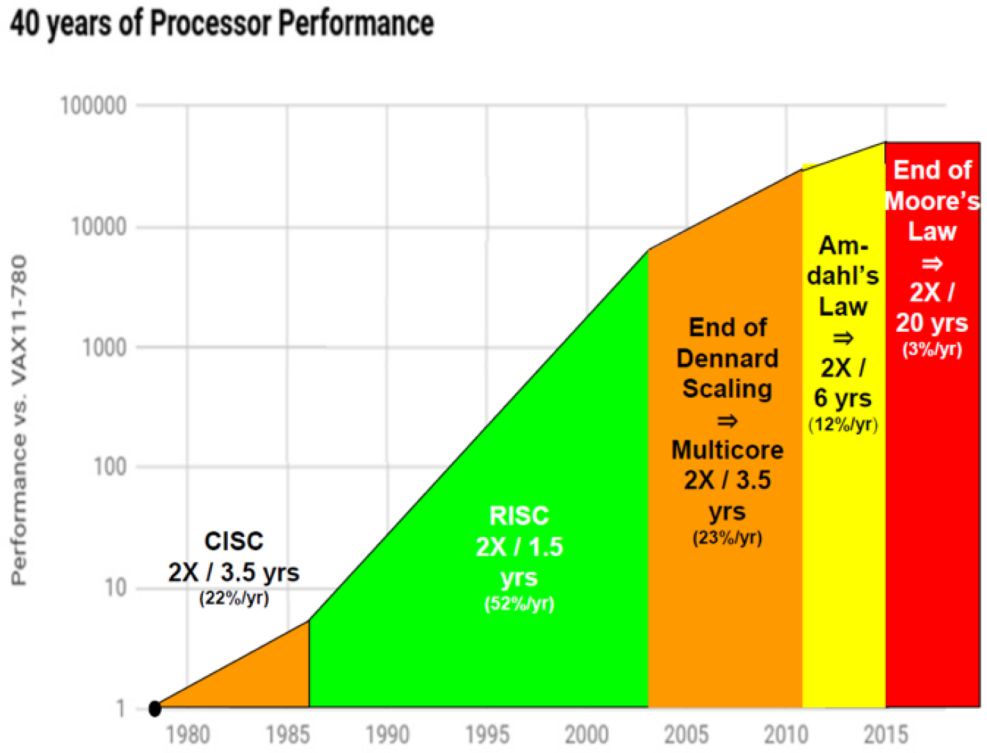
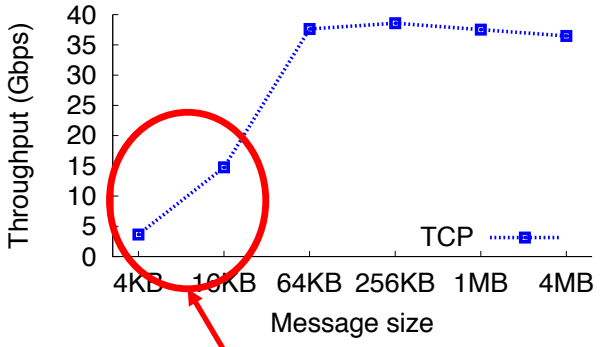


Image Source:
2018 IEEE International Solid-State Circuits Conference
"50 Years of Computer Architecture: from Mainframe CPUs to DNN TPUs and Open RISC-V"

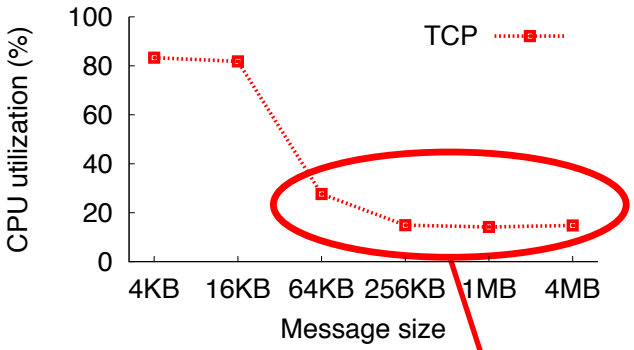
- However, NIC bandwidth keeps increasing exponentially
  - … → 40Gbps (2014) → 100Gbps (2017) → 400Gbps (?)

CPU Starts to Become the Bottleneck

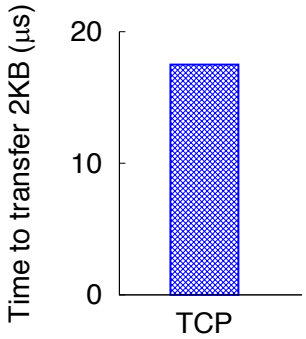
A simple test: 40Gbps NICs, state-of-the-art servers, 16 cores



Small messages → CPU is the bottleneck



Larger msgs → ~3 CPU cores are burnt by TCP

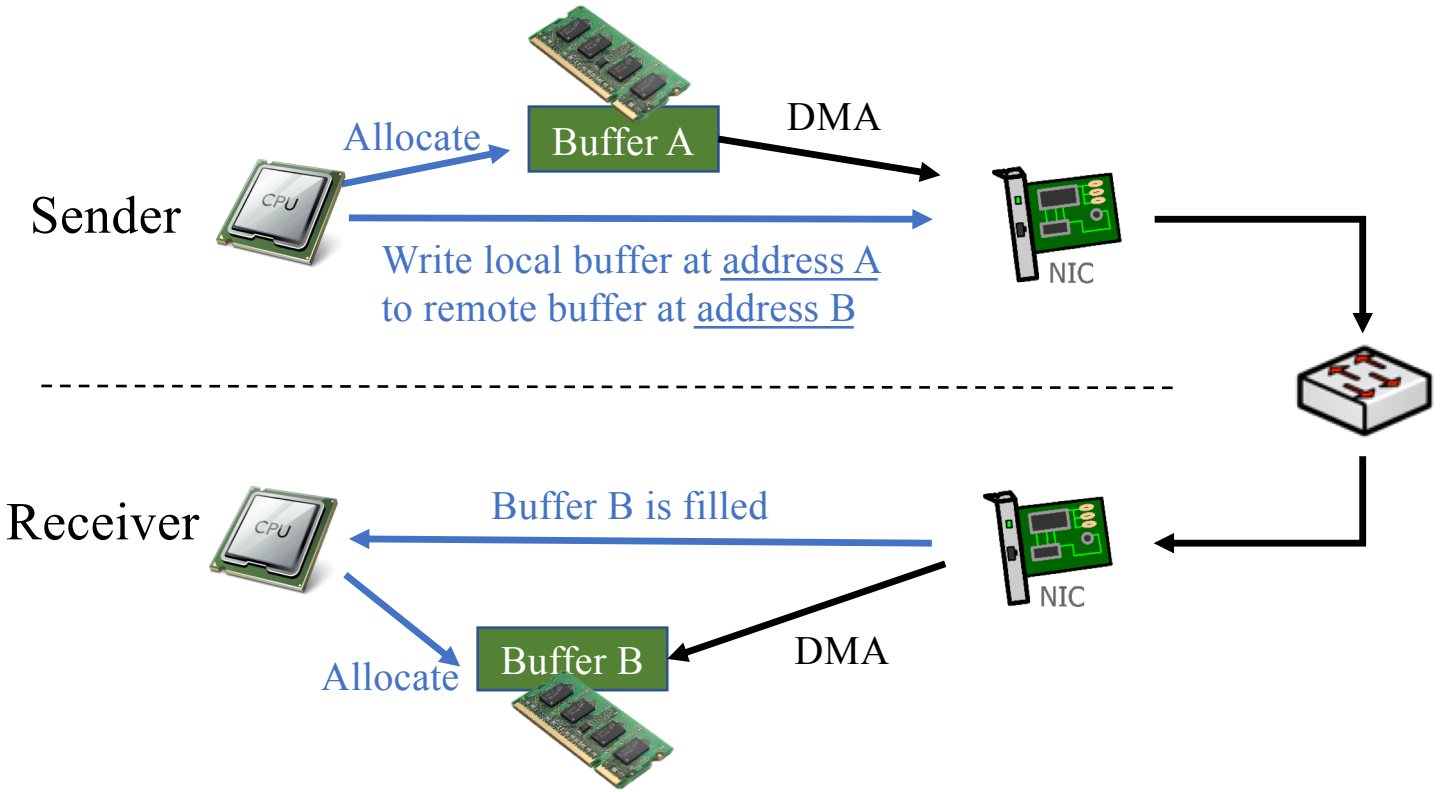


The above test was done on Windows machines [DCQCN, SIGCOMM'15];
Linux machines show a similar trend [Sandstorm, SIGCOMM'14]

Solution: Hardware Offloading (from the CPU)

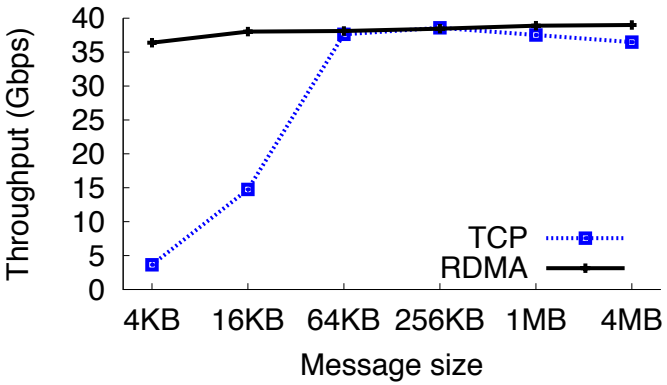
- Step 1: Offloading the network stack (layer 2 to layer 4)
  - Representative solution: **RDMA (Remote Direct Memory Access)**
  - Middleboxes are also offloaded to SmartNICs [VFP, NSDI'18]
- Step 2 (or 2a): Offloading application logic to NICs
  - Storage replication / transactions [Hyperloop, SIGCOMM'18]
  - High IOPS key value stores [KV-Direct, SOSP'17]
- Step 3 (or 2b): Offloading application logic to switches
  - Consensus protocols [NOPaxos, OSDI'16]
  - High IOPS key value stores [NetCache, SOSP'17]

RDMA (Remote Direct Memory Access)

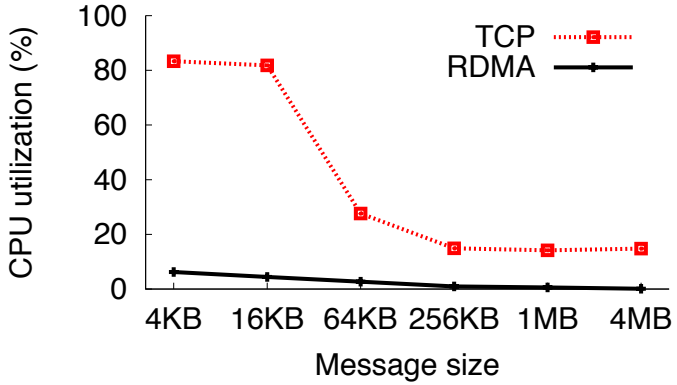


The whole kernel network stack is offloaded to NICs!

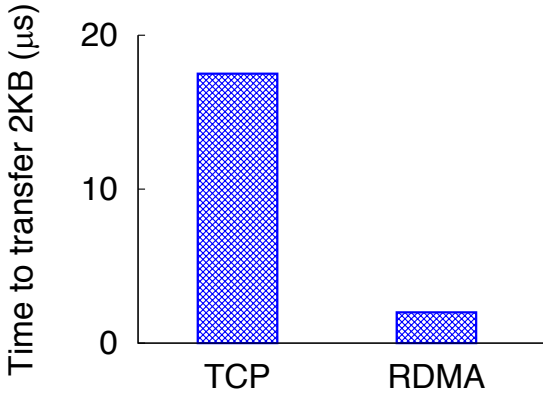
RDMA Outperforms TCP



RDMA single thread ~40Gbps



RDMA CPU ~0%



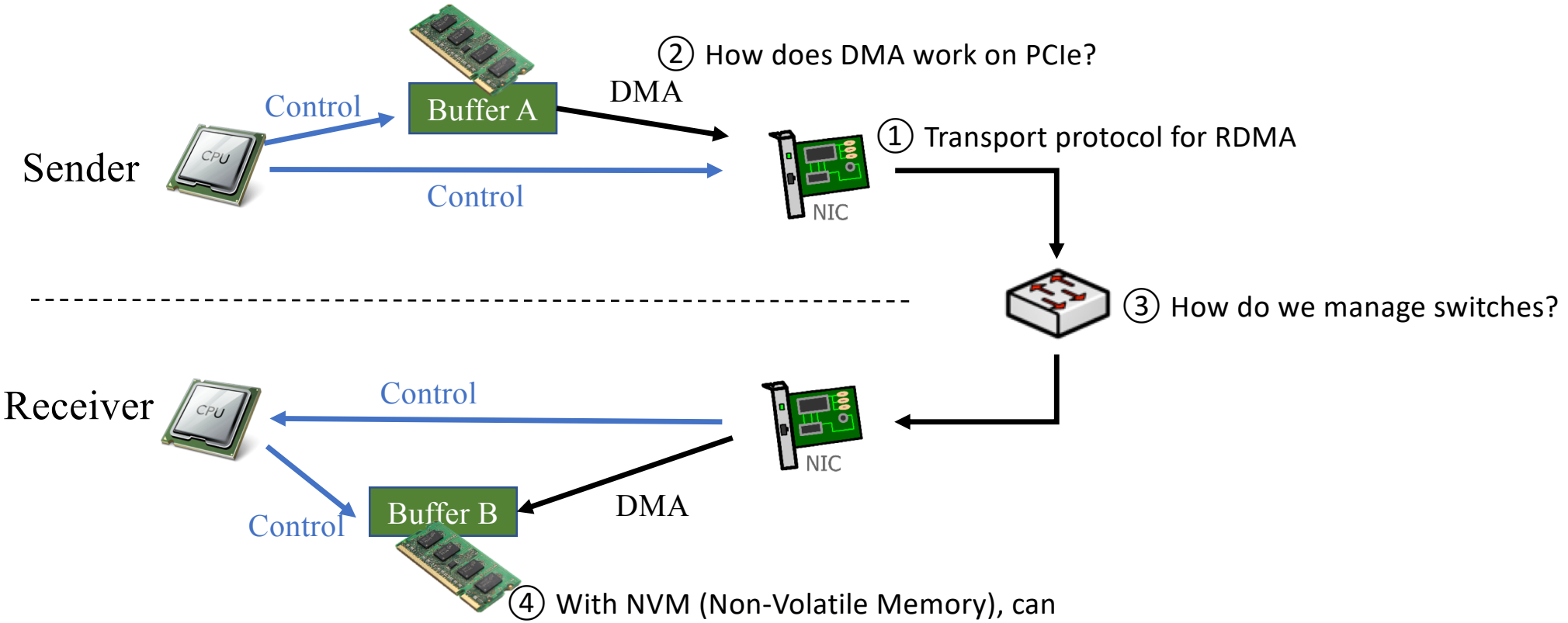
RDMA latency 1~2 µs

More Challenging than It Sounds

- It is not trivial to port software logic into hardware
  - E.g., the NIC has limited buffer → hard to implement the sending window used by TCP
    - Solution: rate-based CC protocols [DCQCN, SIGOCMM'15][TIMELY, SIGCOMM'15]
  - Low performance upon packet drops
    - Solution: PFC or modified retransmission mechanism [IRN, SIGCOMM'18]
  - Limited programmability: what is the best programming abstraction?
  - Limited memory → limited number of "reliable" flows → scalability?

- Research opportunities

Back to the Big Picture



HyperLoop: Group-Based NIC-Offloading to Accelerate Replicated Transactions in Multi-Tenant Storage Systems

Daehyeok Kim[1*], Amirsaman Memaripour[2*], Anirudh Badam[3], Yibo Zhu[3], Hongqiang Harry Liu[3†], Jitu Padhye[3], Shachar Raindel[3], Steven Swanson[2], Vyas Sekar[1], Srinivasan Seshan[1]

[1]Carnegie Mellon University, [2]UC San Diego, [3]Microsoft

ABSTRACT

Storage systems in data centers are an important component of large-scale online services. They typically perform replicated transactional operations for high data availability and integrity. Today, however, such operations suffer from high tail latency even with recent kernel bypass and storage optimizations, and thus affect the predictability of end-to-end performance of these services. We observe that the root cause of the problem is the involvement of the CPU, a precious commodity in multi-tenant settings, in the critical path of replicated transactions. In this paper, we present HyperLoop, a new framework that removes CPU from the critical path of replicated transactions in storage systems by offloading them to commodity RDMA NICs, with non-volatile memory as the storage medium. To achieve this, we develop new and general NIC offloading primitives that can perform memory operations on all nodes in a replication group while guaranteeing ACID properties without CPU involvement. We demonstrate that popular storage applications can be easily optimized using our primitives. Our evaluation results with microbenchmarks and application benchmarks show that HyperLoop can reduce 99^{th} percentile latency $\approx$ 800× with close to 0% CPU consumption on replicas.

*The first two authors contributed equally to this work.
†The author is now in Alibaba Group.

SIGCOMM '18, August 20–25, 2018, Budapest, Hungary

ACM ISBN 978-1-4503-5567-4/18/08…$15.00
https://doi.org/10.1145/3230543.3230572

CCS CONCEPTS

• **Networks** → **Data center networks**; • **Information systems** → **Remote replication**; • **Computer systems organization** → **Cloud computing**;

KEYWORDS

Distributed storage systems; Replicated transactions; RDMA; NIC-offloading

ACM Reference Format:
Daehyeok Kim, Amirsaman Memaripour, Anirudh Badam, Yibo Zhu, Hongqiang Harry Liu, Jitu Padhye, Shachar Raindel, Steven Swanson, Vyas Sekar, Srinivasan Seshan. 2018. HyperLoop: Group-Based NIC-Offloading to Accelerate Replicated Transactions in Multi-Tenant Storage Systems. In *SIGCOMM '18: ACM SIGCOMM 2018 Conference, August 20–25, 2018, Budapest, Hungary.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3230543.3230572

1 INTRODUCTION

Distributed storage systems are an important building block for modern online services. To guarantee data availability and integrity, these systems keep multiple replicas of each data object on different servers [3, 4, 8, 9, 17, 18] and rely on *replicated transactional operations* to ensure that updates are consistently and atomically performed on all replicas.

Such replicated transactions can incur large and unpredictable latencies, and thus impact the overall performance of storage-intensive applications [52, 57, 58, 75, 86, 92]. Recognizing this problem, both networking and storage communities have proposed a number of solutions to reduce average and tail latencies of such systems.

Networking proposals include kernel bypass techniques, such as RDMA (Remote Direct Memory Access) [64], and userspace networking technologies [26, 90]. Similarly, there have been efforts to integrate non-volatile main memory (NVM) [6, 11], and userspace solid state disks (SSDs) [7, 29, 98] to bypass the OS storage stack to reduce latency.

While optimizations such as kernel bypass do improve the performance for standalone storage services and appliance-like systems where there is only a single service running in

- Wed., 4:30PM Session, 1st paper
- Leveraging RDMA + NVM
  - RDMA NIC directly reads/writes durable storage medium
- Turns storage replication and transactions into RDMA operations and bypasses CPU
- Up to 800x tail latency reduction

Revisiting Network Support for RDMA

Radhika Mittal[1], Alexander Shpiner[3], Aurojit Panda[4], Eitan Zahavi[3],
Arvind Krishnamurthy[5], Sylvia Ratnasamy[1], Scott Shenker[1,2]
[1]UC Berkeley, [2]ICSI, [3]Mellanox Technologies, [4]NYU, [5]Univ. of Washington

Abstract

The advent of RoCE (RDMA over Converged Ethernet) has led to a significant increase in the use of RDMA in datacenter networks. To achieve good performance, RoCE requires a lossless network which is in turn achieved by enabling Priority Flow Control (PFC) within the network. However, PFC brings with it a host of problems such as head-of-the-line blocking, congestion spreading, and occasional deadlocks. Rather than seek to fix these issues, we instead ask: *is PFC fundamentally required to support RDMA over Ethernet?*

We show that the need for PFC is an artifact of current RoCE NIC designs rather than a fundamental requirement. We propose an *improved RoCE NIC* (IRN) design that makes a few simple changes to the RoCE NIC for better handling of packet losses. We show that IRN (without PFC) outperforms RoCE (with PFC) by 6-83% for typical network scenarios. Thus not only does IRN eliminate the need for PFC, it *improves* performance in the process! We further show that the changes that IRN introduces can be implemented with modest overheads of about 3-10% to NIC resources. Based on our results, we argue that research and industry should rethink the current trajectory of network support for RDMA.

CCS Concepts

• **Networks** → **Transport protocols**;

Keywords

Datacenter transport, RDMA, RoCE, iWARP, PFC

ACM Reference Format:
Radhika Mittal, Alexander Shpiner, Aurojit Panda, Eitan Zahavi, Arvind Krishnamurthy, Sylvia Ratnasamy, Scott Shenker. 2018. Revisiting Network Support for RDMA. In *SIGCOMM '18: ACM SIGCOMM 2018 Conference, August 20–25, 2018, Budapest, Hungary.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3230543.3230557

ACM ISBN 978-1-4503-5567-4/18/08…$15.00
https://doi.org/10.1145/3230543.3230557

1 Introduction

Datacenter networks offer higher bandwidth and lower latency than traditional wide-area networks. However, traditional endhost networking stacks, with their high latencies and substantial CPU overhead, have limited the extent to which applications can make use of these characteristics. As a result, several large datacenters have recently adopted RDMA, which bypasses the traditional networking stacks in favor of direct memory accesses.

RDMA over Converged Ethernet (RoCE) has emerged as the canonical method for deploying RDMA in Ethernet-based datacenters [23, 38]. The centerpiece of RoCE is a NIC that (i) provides mechanisms for accessing host memory without CPU involvement and (ii) supports very basic network transport functionality. Early experience revealed that RoCE NICs only achieve good end-to-end performance when run over a lossless network, so operators turned to Ethernet's Priority Flow Control (PFC) mechanism to achieve minimal packet loss. The combination of RoCE and PFC has enabled a wave of datacenter RDMA deployments.

However, the current solution is not without problems. In particular, PFC adds management complexity and can lead to significant performance problems such as head-of-the-line blocking, congestion spreading, and occasional deadlocks [23, 24, 35, 37, 38]. Rather than continue down the current path and address the various problems with PFC, in this paper we take a step back and ask whether it was needed in the first place. To be clear, current RoCE NICs require a lossless fabric for good performance. However, the question we raise is: *can the RoCE NIC design be altered so that we no longer need a lossless network fabric?*

We answer this question in the affirmative, proposing a new design called IRN (for Improved RoCE NIC) that makes two incremental changes to current RoCE NICs (i) more efficient loss recovery, and (ii) basic end-to-end flow control to bound the number of in-flight packets (§3). We show, via extensive simulations on a RoCE simulator obtained from a commercial NIC vendor, that IRN performs better than current RoCE NICs, and that IRN does not require PFC to achieve high performance; in fact, IRN often performs better without PFC (§4). We detail the extensions to the RDMA protocol that IRN requires (§5) and use comparative analysis and

- Wed., 4:30PM Session, 2nd paper
- Commodity RDMA hardware runs E2E congestion control + PFC
- PFC can cause troubles in large-scale deployment
  - HoL blocking, fairness, deadlock…
- What is the minimum (feasible) NIC hardware change that can help us get rid of PFC?

Understanding PCIe performance for end host networking

Rolf Neugebauer
Independent Researcher

Gianni Antichi
Queen Mary, University of London

José Fernando Zazo
Naudit HPCN

Yury Audzevich
University of Cambridge

Sergio López-Buedo
Universidad Autónoma de Madrid

Andrew W. Moore
University of Cambridge

ABSTRACT

In recent years, spurred on by the development and availability of programmable NICs, end hosts have increasingly become the enforcement point for core network functions such as load balancing, congestion control, and application specific network offloads. However, implementing custom designs on programmable NICs is not easy: many potential bottlenecks can impact performance.

This paper focuses on the performance implication of PCIe, the de-facto I/O interconnect in contemporary servers, when interacting with the host architecture and device drivers. We present a theoretical model for PCIe and `pcie-bench`, an open-source suite, that allows developers to gain an accurate and deep understanding of the PCIe substrate. Using `pcie-bench`, we characterize the PCIe subsystem in modern servers. We highlight surprising differences in PCIe implementations, evaluate the undesirable impact of PCIe features such as IOMMUs, and show the practical limits for common network cards operating at 40Gb/s and beyond. Furthermore, through `pcie-bench` we gained insights which guided software and future hardware architectures for both commercial and research oriented network cards and DMA engines.

CCS CONCEPTS

• **Networks** → **Network adapters**; **Network servers**; • **Hardware** → **Networking hardware**; *Buses and high-speed links*;

KEYWORDS

PCIe, reconfigurable hardware, Operating System

ACM Reference Format:
Rolf Neugebauer, Gianni Antichi, José Fernando Zazo, Yury Audzevich, Sergio López-Buedo, and Andrew W. Moore. 2018. Understanding PCIe performance for end host networking. In *SIGCOMM '18: SIGCOMM 2018, August 20–25, 2018, Budapest, Hungary.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3230543.3230560

SIGCOMM '18, August 20–25, 2018, Budapest, Hungary

ACM ISBN 978-1-4503-5567-4/18/08…$15.00
https://doi.org/10.1145/3230543.3230560

1 INTRODUCTION

The idea of end hosts participating in the implementation of network functionality has been extensively explored in enterprise and datacenter networks [6, 7, 25, 49, 56, 58]. Moreover, the disruptive introduction into the market of programmable NICs, alongside the deployment in datacenters of hybrid Xeon and FPGA platforms [15], has boosted the demand for new refined solutions which combine software functions and hardware NIC acceleration to improve end host networking performance [53], flexibility [16], or a combination of both [29]. Several efforts try to leverage end host hardware programmability to improve datacenter scalability [12, 13] or specific network functions such as load balancing, application level quality of service and congestion control [1].

In this paper, we show that PCIe, alongside its interaction with the host architecture and device drivers, can significantly impact the performance of network applications. Past research has mostly considered this impact in the context of specific applications such as Remote DMA (RDMA) [24], GPU-accelerated packet processing [17], and optimized Key-Value-Store (KVS) applications [31, 32, 34]. In contrast, we argue that a more generic approach to studying and characterizing PCIe is needed as it has become essential to implement not only specialized, high-performance network functions, but also storage adaptors and custom accelerator cards, such as for machine learning [23]. It is in this context that we introduce a theoretical model for PCIe (§3), design a methodology to characterize PCIe in real systems (§4), describe its implementation (§5), and present the results derived using our approach (§6). This permits us to draw several specific conclusions about the way PCIe currently

- Wed., 4:30PM Session, 3rd paper
- The RDMA NICs and SmartNIC rely on DMA via PCIe
- We really need to understand the PCIe behavior in order to get the best hardware offloading benefits
- Novel PCIe measurement tool and results

FBOSS: Building Switch Software at Scale

Sean Choi* Stanford University
Boris Burkov Facebook, Inc.
Alex Eckert Facebook, Inc.
Tian Fang Facebook, Inc.
Saman Kazemkhani Facebook, Inc.
Rob Sherwood Facebook, Inc.
Ying Zhang Facebook, Inc.
Hongyi Zeng Facebook, Inc.

ABSTRACT

The conventional software running on network devices, such as switches and routers, is typically vendor-supplied, proprietary and closed-source; as a result, it tends to contain extraneous features that a single operator will not most likely fully utilize. Furthermore, cloud-scale data center networks often times have software and operational requirements that may not be well addressed by the switch vendors.

In this paper, we present our ongoing experiences on overcoming the complexity and scaling issues that we face when designing, developing, deploying and operating an in-house software built to manage and support a set of features required for data center switches of a large scale Internet content provider. We present FBOSS, our own data center switch software, that is designed with the basis on our *switch-as-a-server* and *deploy-early-and-iterate* principles. We treat software running on data center switches as any other software services that run on a commodity server. We also build and deploy only a minimal number of features and iterate on it. These principles allow us to rapidly iterate, test, deploy and manage FBOSS at scale. Over the last five years, our experiences show that FBOSS's design principles allow us to quickly build a stable and scalable network. As evidence, we have successfully grown the number of FBOSS instances running in our data center by over 30x over a two year period.

CCS CONCEPTS

• **Networks** → **Data center networks**; *Programming interfaces*; Routers;

*Work done while at Facebook, Inc.

SIGCOMM '18, August 20–25, 2018, Budapest, Hungary

ACM ISBN 978-1-4503-5567-4/18/08…$15.00
https://doi.org/10.1145/3230543.3230546

KEYWORDS

FBOSS, Facebook, Switch Software Design, Data Center Networks, Network Management, Network Monitoring

ACM Reference Format:
Sean Choi, Boris Burkov, Alex Eckert, Tian Fang, Saman Kazemkhani, Rob Sherwood, Ying Zhang, and Hongyi Zeng. 2018. FBOSS: Building Switch Software at Scale. In *SIGCOMM '18: SIGCOMM 2018, August 20–25, 2018, Budapest, Hungary.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3230543.3230546

1 INTRODUCTION

The world's desire to produce, consume, and distribute online content is increasing at an unprecedented rate. Commensurate with this growth are equally unprecedented technical challenges in scaling the underlying networks. Large Internet content providers are forced to innovate upon all aspects of their technology stack, including hardware, kernel, compiler, and various distributed systems building blocks. A driving factor is that, at scale even a relatively modest efficiency improvement can have large effects. For us, our data center networks power a cloud-scale Internet content provider with billions of users, interconnecting hundreds of thousands of servers. Thus, it is natural and necessary to innovate on the software that runs on switches. [1]

Conventional switches typically come with software written by vendors. The software includes drivers for managing dedicated packet forwarding hardware (e.g., ASICs, FPGAs, or NPUs), routing protocols (e.g., BGP, OSPF, STP, MLAG), monitoring and debugging features (e.g., LLDP, BFD, OAM), configuration interfaces (e.g., conventional CLI, SNMP, NetConf, OpenConfig), and a long tail of other features needed to run a modern switch. Implicit in the vendor model is the assumption that networking requirements are *correlated* between customers. In other words, vendors are successful because they can create a small number of products and reuse them across many customers. However our network size and the rate of growth of the network (Figure 1) are unlike most other data center networks. Thus, they imply that our requirements are quite different from most customers.

[1]We use "switch" for general packet switching devices such as switches and routers. Our data center networks are fully Layer 3 routed similar to what is described in [36].

- Wed., 4:30PM Session, 4th paper
- We must manage switches
  - E.g., rolling out RDMA would require additional switch features than running TCP
- The key for quick evolvement is to develop switch software just like common software
- Experience paper from Facebook on their switch OS
  - BTW, check out SONiC by Microsoft

Exciting Area, Vast Research Opportunities

- Are we really done with RDMA transport protocols?
  - How about extending it from DC to WAN?
- New RDMA-based distributed systems, or new (network) hardware offloading architectures
- Hardware-software co-design: what is the right abstraction?
  - Goal: expose maximum programmability while keep hardware performance
  - It may depend on hardware platforms (FPGA vs. P4 ASIC vs. ARM/MIPS NPU)

What is Networking Like, 5 Years from Now?

- Today, networks are (dumb) pipes for moving data
- We have been building faster, wider and more reliable pipes
- Question: will the pipe still be dumb five years from now? Should it?
- I argue that we should build more “intelligence” into the pipe
  - Opportunity: network hardware offloading (accelerating) distributed systems
  - E.g., after the end of Moore’s Law, GPU shines for parallel computing / AI
  - Can network hardware do the same for distributed systems?