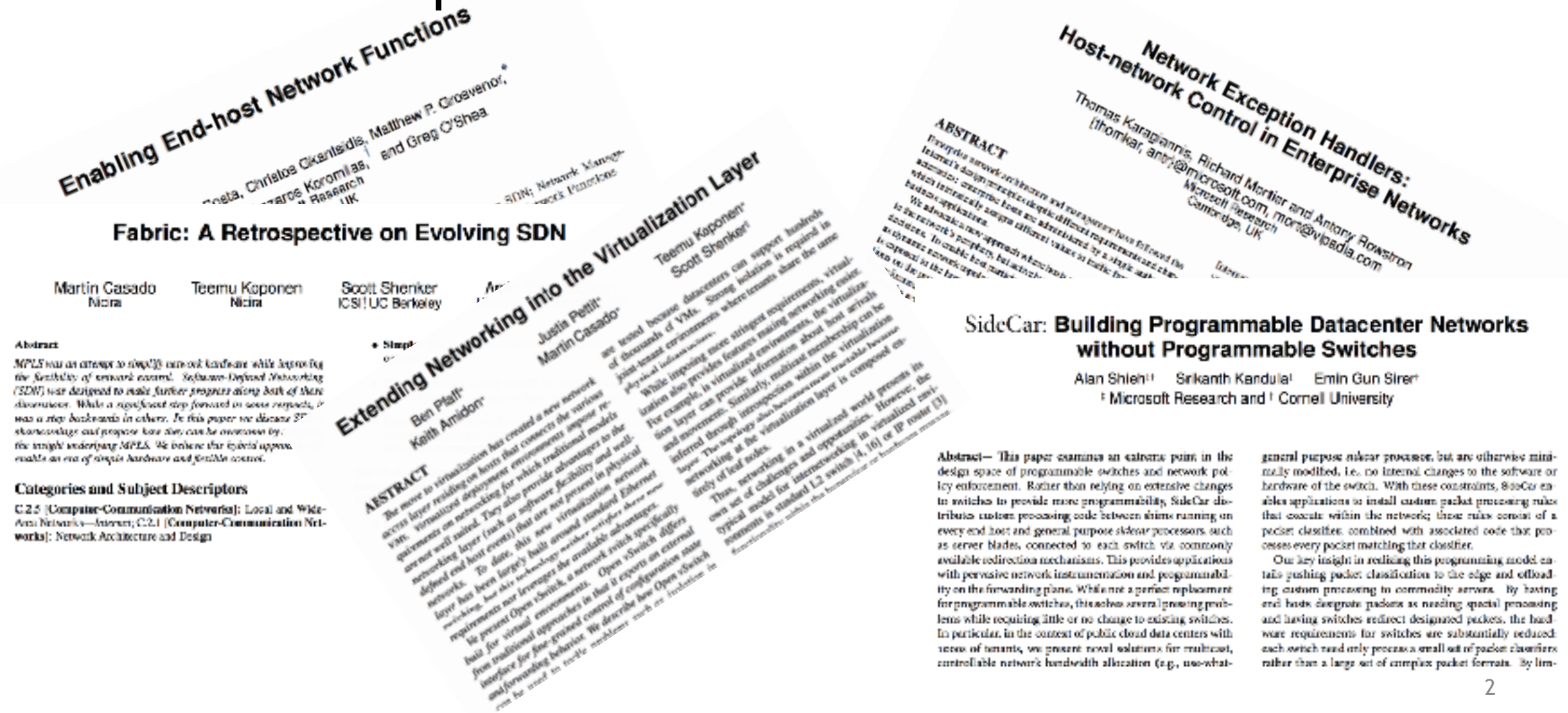


Understanding PCIe performance for end host networking

Rolf Neugebauer, Gianni Antichi, José Fernando Zazo,
Yury Audzevich, Sergio López-Buedo, Andrew W. Moore



The idea of end hosts participating in the implementation of network functionality has been extensively explored in enterprise and datacenter networks



More recently, programmable NICs and FPGAs enable offload and NIC customisation

SENIC: Scalable NIC for End-Host Rate Limiting
Sivasankar Radhakrishnan*, Yilong Gong*, Yimankumar Jayakumar*,
Abdul Karimul, George Porter*, Anil Valala*,
*University of California, San Diego
sivasankar.radhakrishnan@ucsd.edu
yilong.gong@ucsd.edu
yimankumar.jayakumar@ucsd.edu
abulkarimul@ucsd.edu
george.porter@ucsd.edu
anil.valala@ucsd.edu

Property	Hardware	Software
Scales to many classes	✓	✓
Works at high link speeds	✓	✓
Low CPU overhead	✓	✓
Precise rate enforcement	✓	✓
Supports hypervisor bypass	✓	✓

Table 1: Pros and cons of current hardware and software-based rate limiters.

KV-Direct: High-Performance In-Memory Key-Value Store with Programmable NIC
Bijie Li*, Zhenyuan Ruan*, Wencong Xiao*, Yuanwei Lu*,
Yongqiang Xiong*, Andrew Putnam*, Enhong Chen*, Lintao Zhang*,
*Microsoft Research
bijie.li@microsoft.com
zhenyuan.ruan@microsoft.com
wencong.xiao@microsoft.com
yuanwei.lu@microsoft.com
yongqiang.xiong@microsoft.com
andrew.putnam@microsoft.com
enhong.chen@microsoft.com
lintao.zhang@microsoft.com

CCS CONCEPTS
• Information systems → Key-value stores
• Hardware → Programmable logic devices
• Hardware → Network interface controllers

Enabling End-host Network Functions

Hitesh Ballani, Paolo Costa, Christos Gkantsidis, Matthew P. Grosvenor,
Thomas Karagiannis, Lazaros Kleromilos, and Greg O'Shea
Microsoft Research

ABSTRACT

Many network functions executed in modern datacenters, e.g., load balancing, application-level QoS, and congestion control, exhibit three common properties at the data plane: they need to access and modify state, to perform computations, and to access application semantics — this is critical since many network functions are best expressed in terms of application-level messages. In this paper, we present a novel framework for enabling

Keywords

Software Defined Networking, SDN, Network Management, Data-plane programming, Network Functions

1 Introduction

Recent years have seen a lot of innovation in functionality deployed across datacenter networks. Network functions range from management tasks like load

High Performance Packet Processing with FlexNIC
Annoie Kaufmann, Thomas Anderson, Simon Peter, Naveen K. Sharma,
*University of Washington
annoie.kaufmann@u.washington.edu
thomas.anderson@u.washington.edu
simon.peter@u.washington.edu
naveen.k.sharma@u.washington.edu

HyperLoop: Group-Based NIC-Offloading to Accelerate Replicated Transactions in Multi-Tenant Storage Systems
Daehyeon Kim¹, Amirreza Memaripour², Anirudh Badam³,
Yibo Zhu³, Hongqiang Harry Liu³, Jitu Padhye³, Shachar Raindel³,
Steven Swanson², Vyas Sekar¹, Srinivasan Seshan¹,
*Carnegie Mellon University, ²UC San Diego, ³Microsoft

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

CCS CONCEPTS
• Networks → Data center networks
• Information systems → Remote replication
• Computer systems → Cloud computing

- Isolation
- QoS
- Load balancing
- Application specific processing
-

Not “just” in academia, but in production!

Azure Accelerated Networking: SmartNICs in the Public Cloud

Daniel Firestone Andrew Putnam Sambhrama Mundkur Derek Chiou Alireza Dabagh
Mike Andrewartha Hari Angepat Vivek Bhanu Adrian Caulfield Eric Chung
Harish Kumar Chandrappa Somesh Chaturmohta Matt Humphrey Jack Lavier Norman Lam
Fengfen Liu Kalin Ovtcharov Jitu Padhye Gautham Popuri Shachar Raindel Tejas Sapre
Mark Shaw Gabriel Silva Madhan Sivakumar Nisheeth Srivastava Anshuman Verma Qasim Zuhair
Deepak Bansal Doug Burger Kushagra Vaid David A. Maltz Albert Greenberg

Microsoft

Abstract

Modern cloud architectures rely on each server running its own networking stack to implement policies such as tunneling for virtual networks, security, and load balancing. However, these networking stacks are becoming increasingly complex as features are added and as network speeds

all virtual networking features, such as private virtual networks with customer supplied address spaces, scalable L4 load balancers, security groups and access control lists (ACLs), virtual routing tables, bandwidth metering, QoS, and more. These features are the responsibility of the host platform, which typically means software running in the hypervisor.

Implementing offloads is not easy

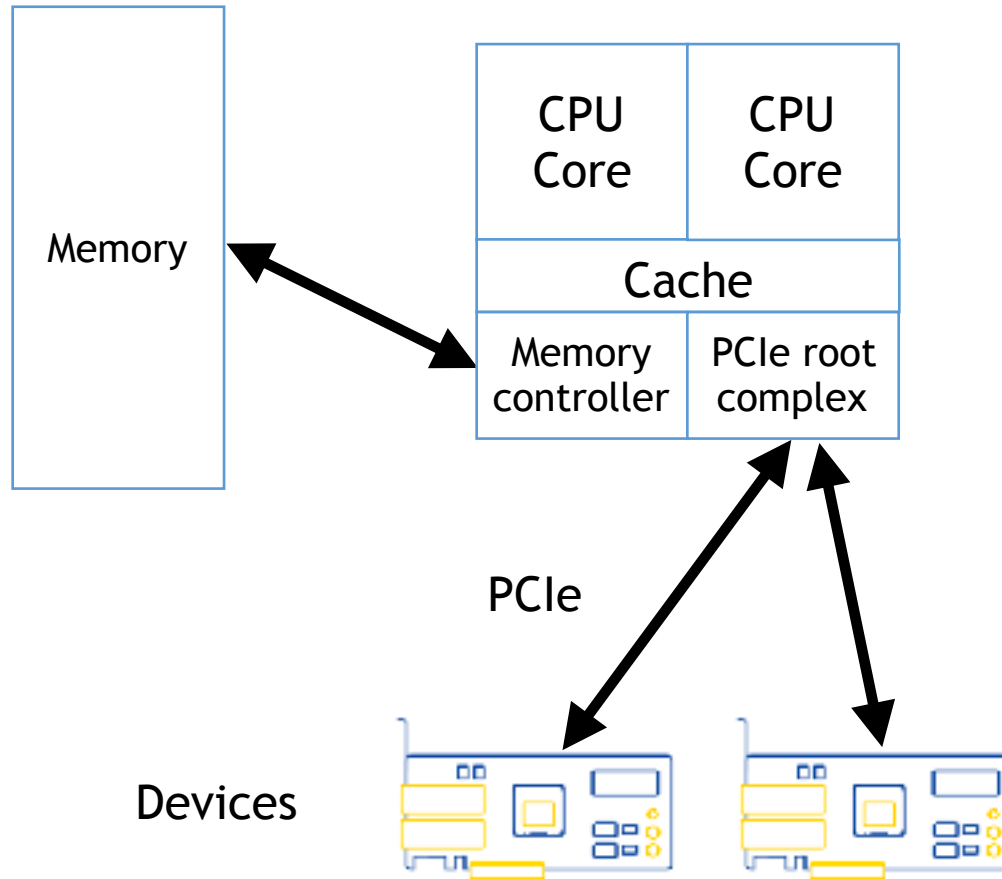
Many potential bottlenecks

Implementing offloads is not easy

Many potential bottlenecks

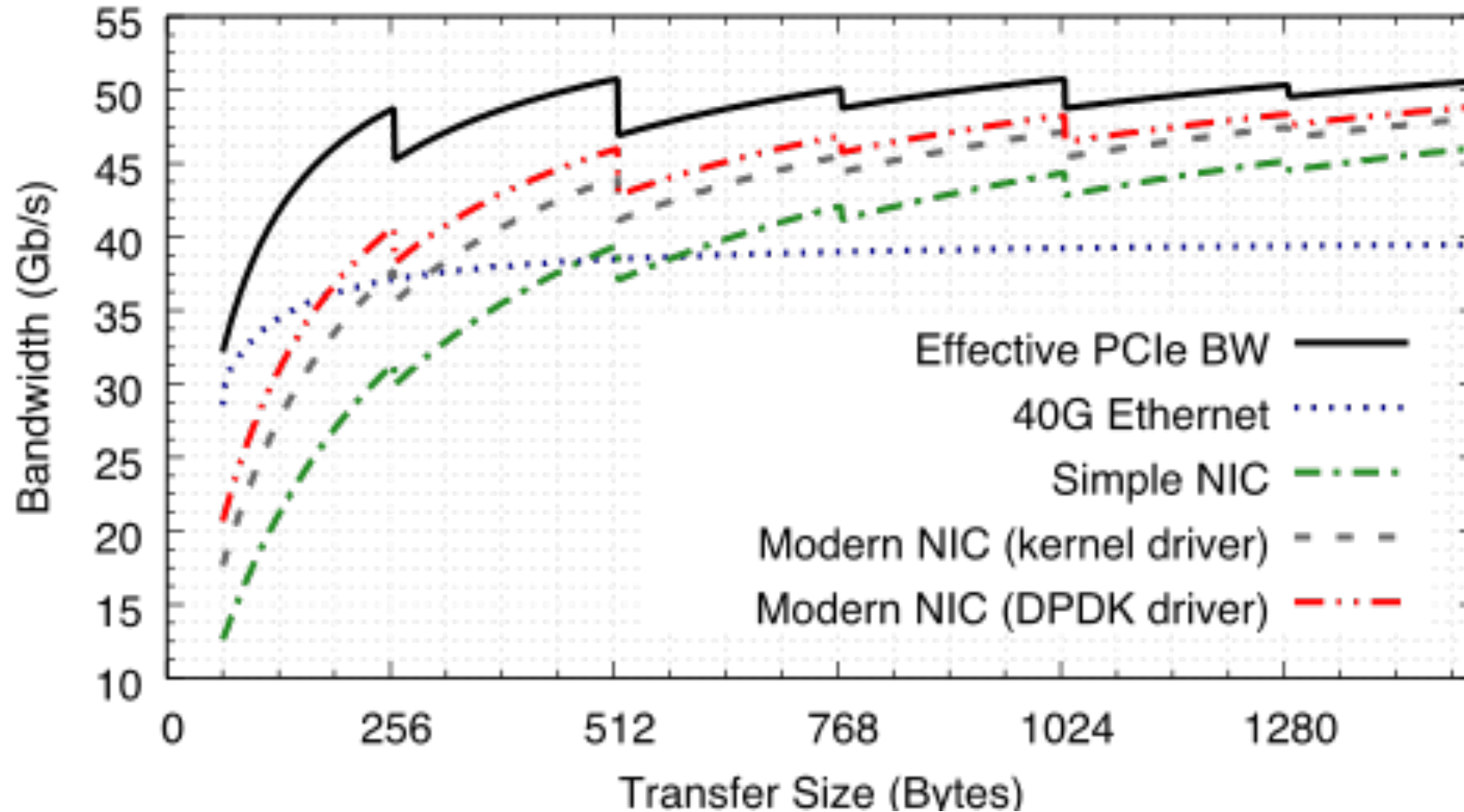
PCI Express (PCIe) and its implementation by the host
is one of them!

PCIe overview



- De facto standard to connect high performance IO devices to the rest of the system. Ex: NICs, NVMe, graphics, TPUs
- PCIe devices transfer data to/from host memory via **DMA** (direct memory access)
- **DMA engines** on each device translate requests like “Write these 1500 bytes to host address 0x1234” into multiple PCIe Memory Write (MWr) “packets”.
- PCIe is almost like a network protocol with packets (TLPs), headers, MTU (MPS), flow control, addressing and switching (and NAT ;)

PCIe protocol overheads

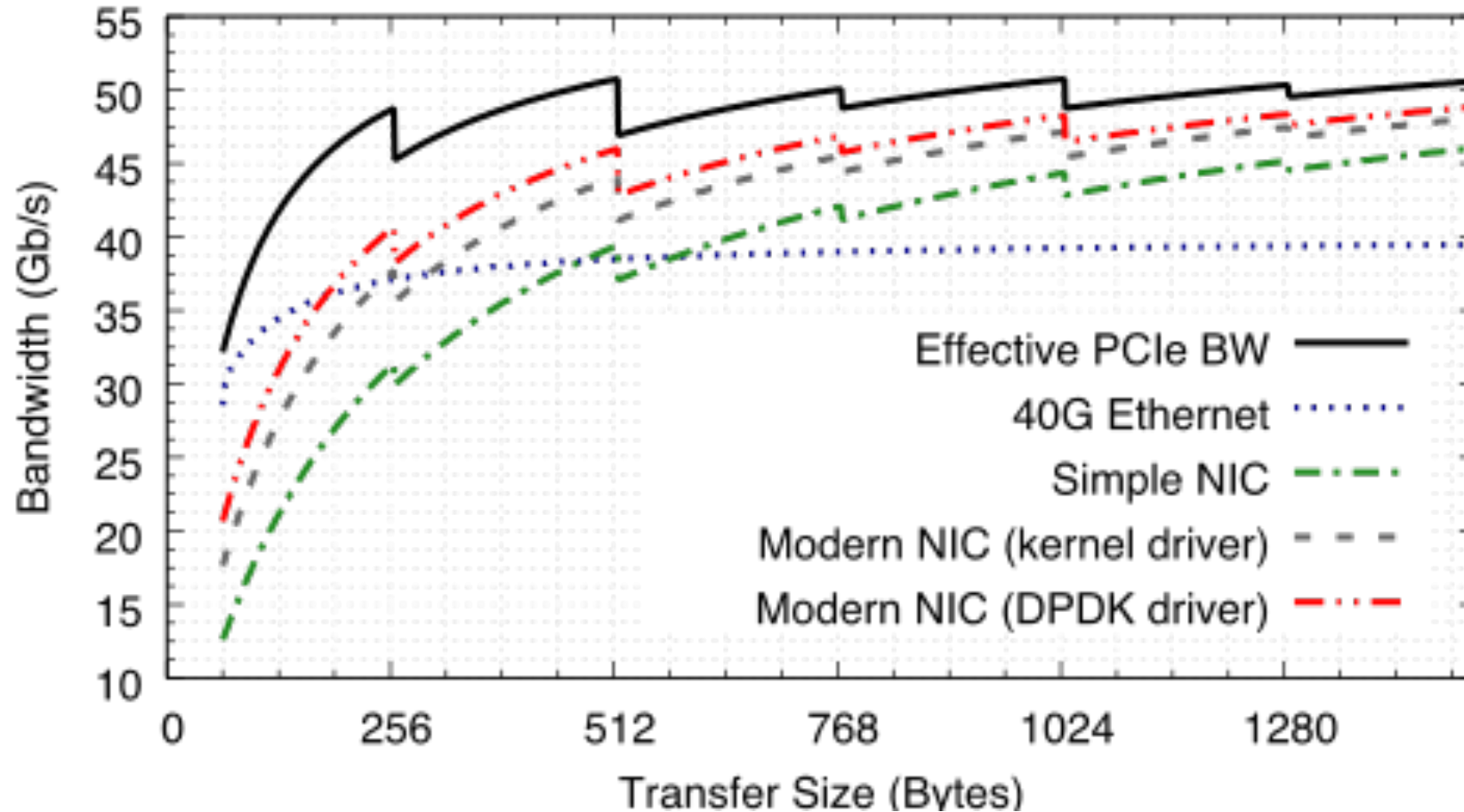


62.96 Gb/s at the physical layer

PCIe protocol

~ 32 - 50 Gb/s for data transfers

PCIe protocol overheads



62.96 Gb/s at the physical layer

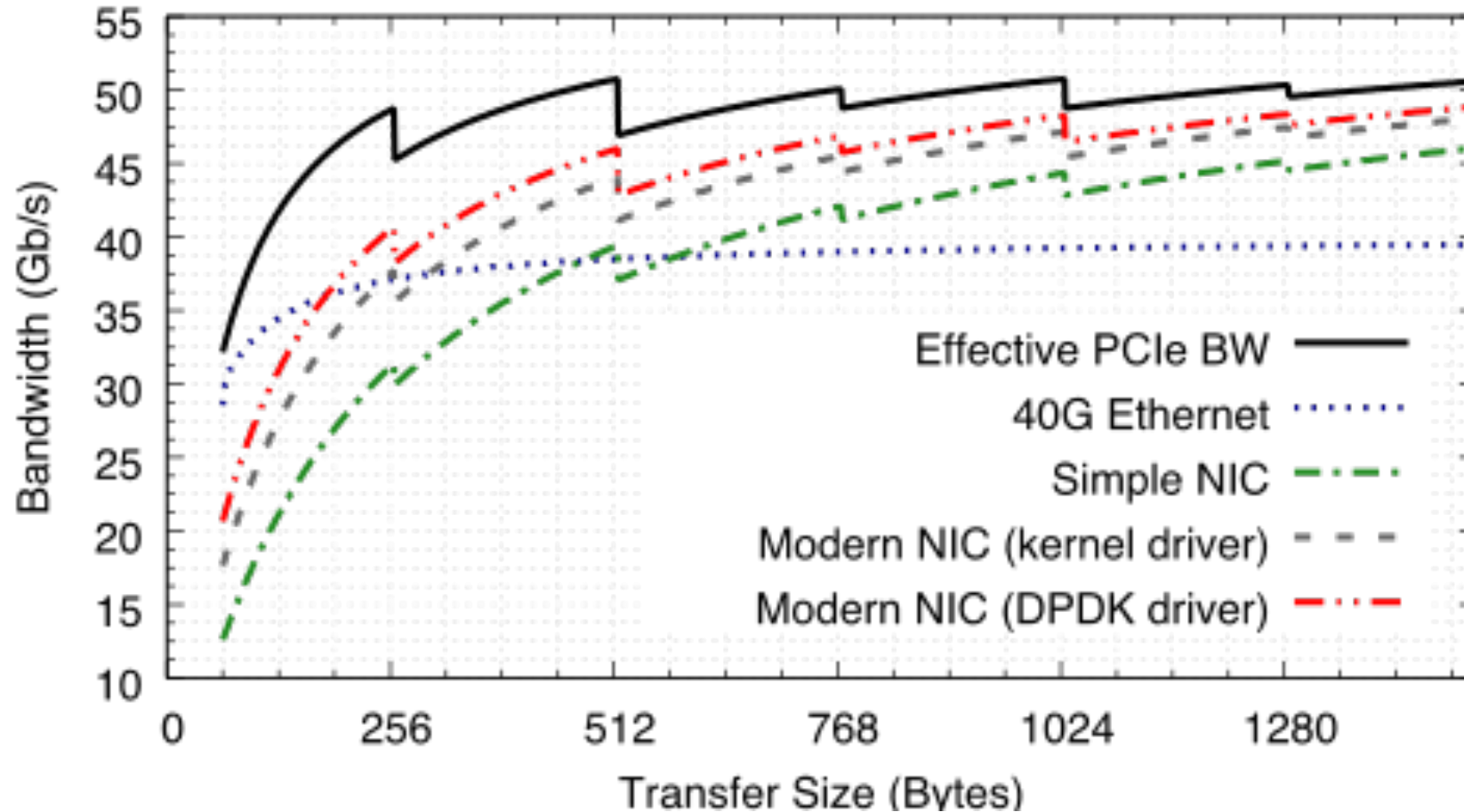
PCIe protocol

~ 32 - 50 Gb/s for data transfers

Queue pointer updates, descriptors, interrupts

~ 12 - 48 Gb/s

PCIe protocol overheads



62.96 Gb/s at the physical layer

↓ PCIe protocol

~ 32 - 50 Gb/s for data transfers

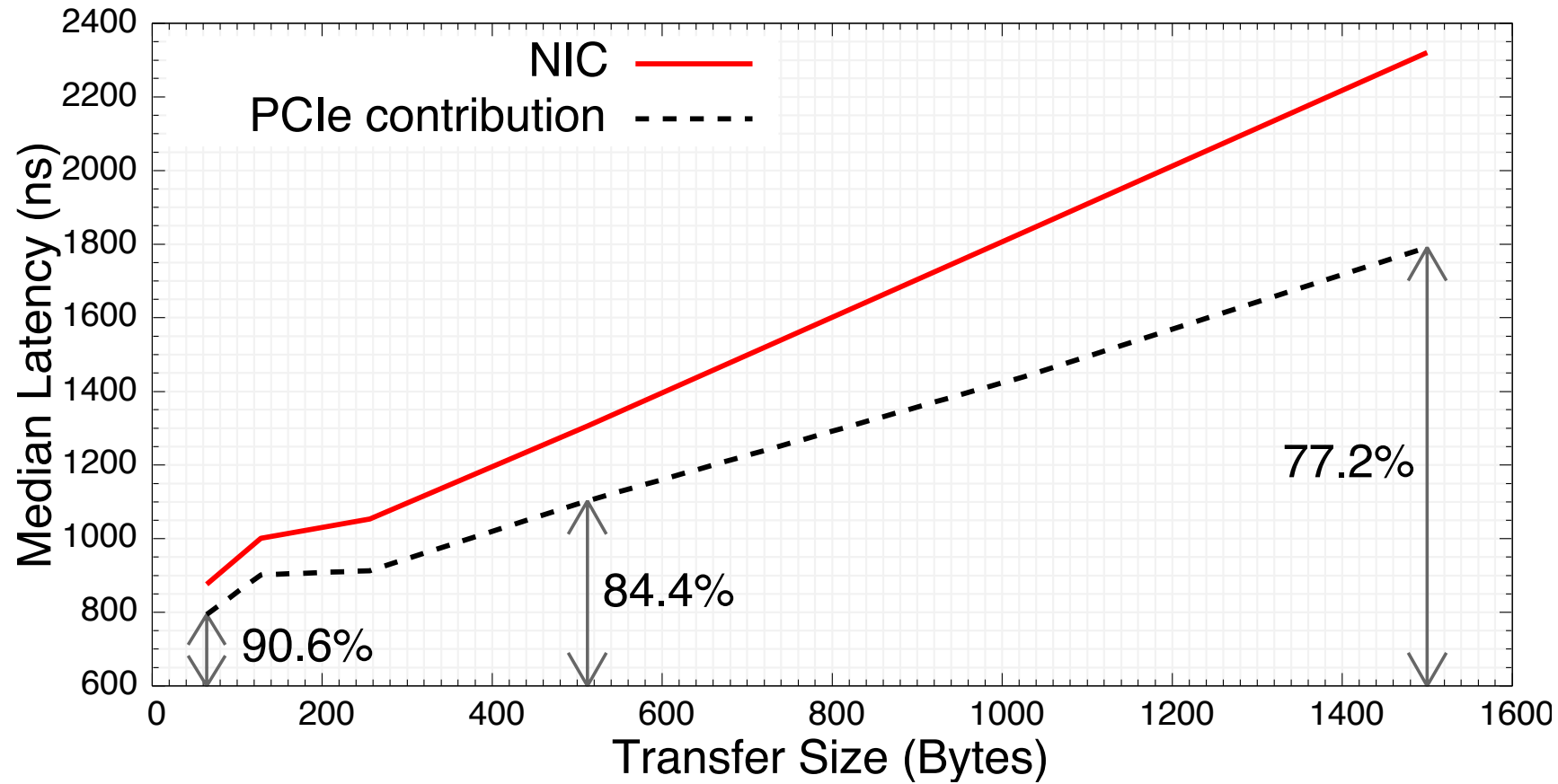
↓ Queue pointer updates, descriptors, interrupts

~ 12 - 48 Gb/s

Complexity!

Model: PCIe gen 3 x8 64 bit addressing

PCIe latency

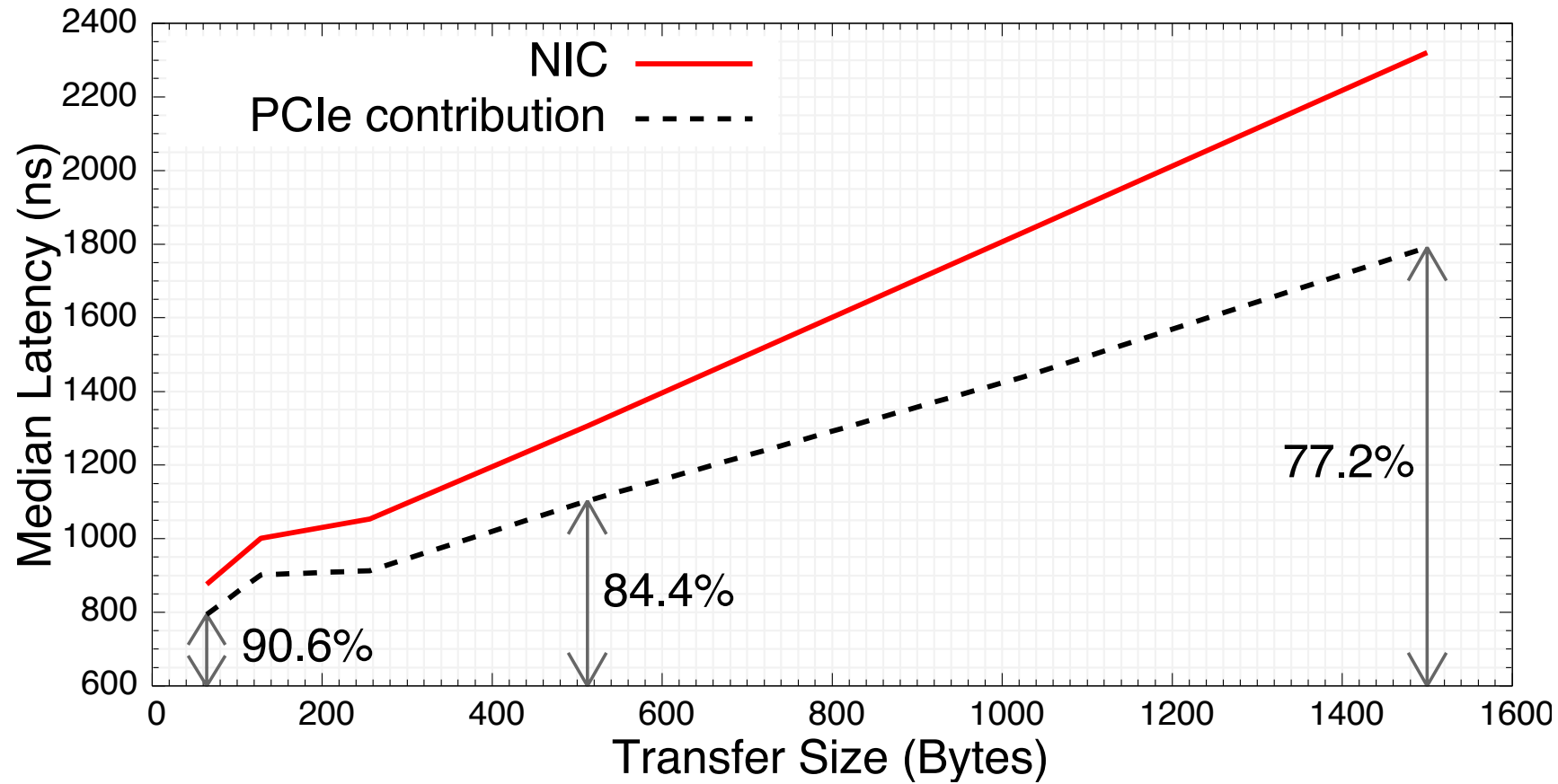


ExaNIC round trip times (loopback) with kernel bypass

PCIe contributes the majority of latency

Homa [SIGCOMM2018]:
Desire single digit us latency for small messages

PCIe latency imposes constraints

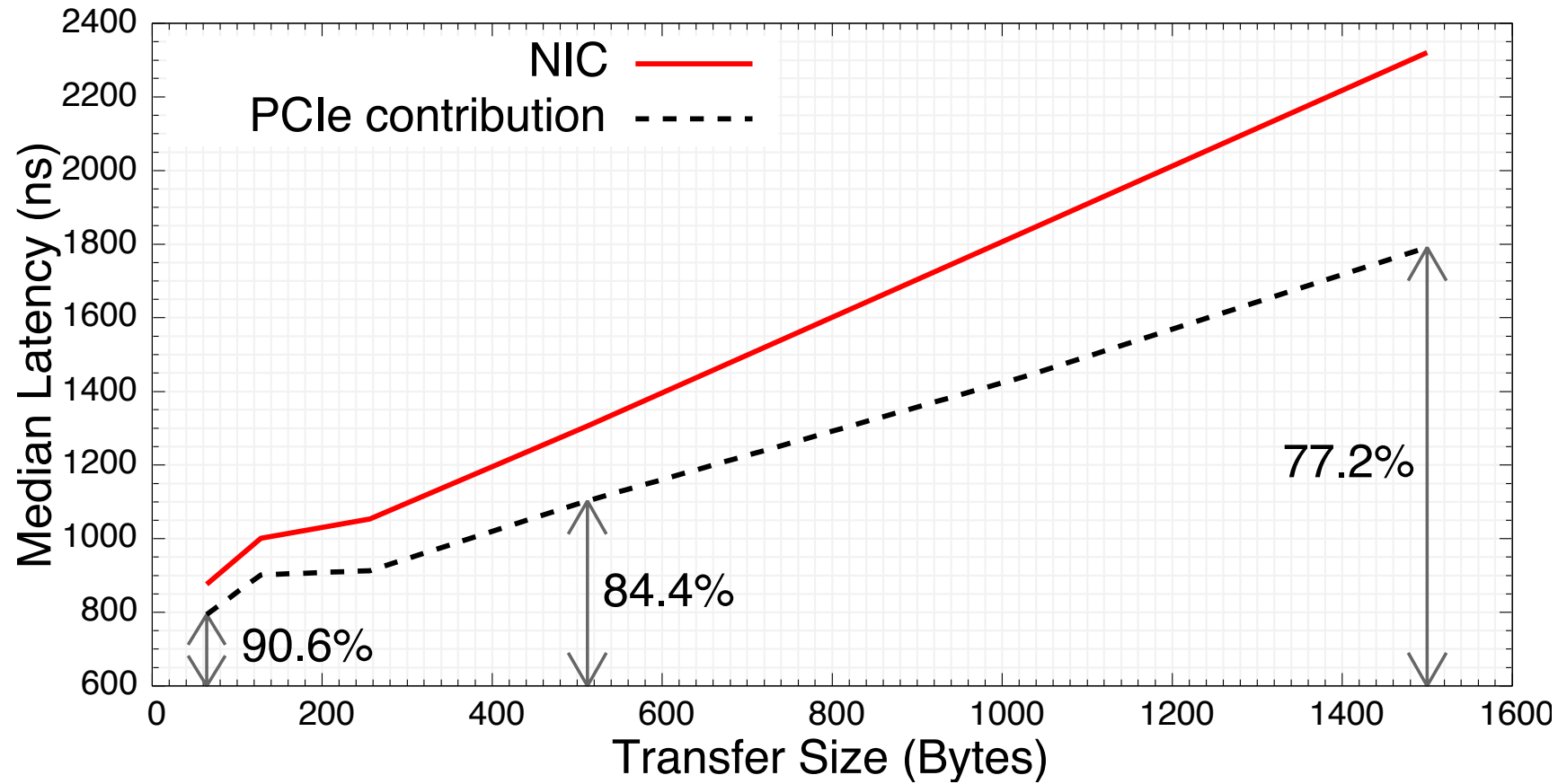


Ethernet line rate at 40Gb/s for 128B packets means a **new packet every 30ns**.

=

NIC has to handle at least 30 concurrent DMAs in each direction plus descriptor DMA

PCIe latency imposes constraints



Ethernet line rate at 40Gbps for 128B packets means a **new packet every 30ns.**

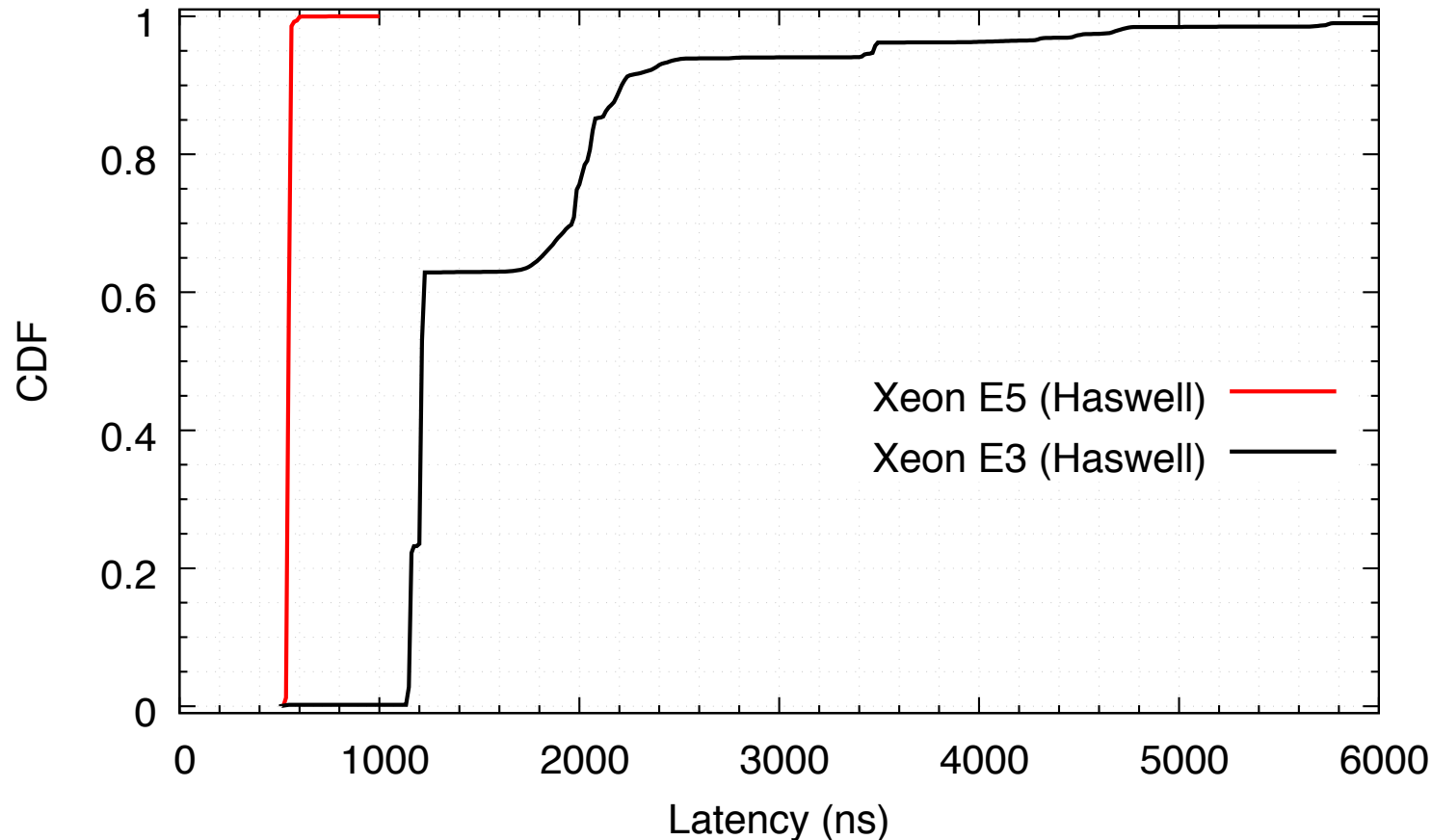
=

NIC has to handle at least 30 concurrent DMAs in each direction plus descriptor DMA

Complexity!

It get's worse...

Distribution of 64B DMA Read latency



Xeon E5

- 547ns median
- 573ns 99th percentile
- 1136ns max

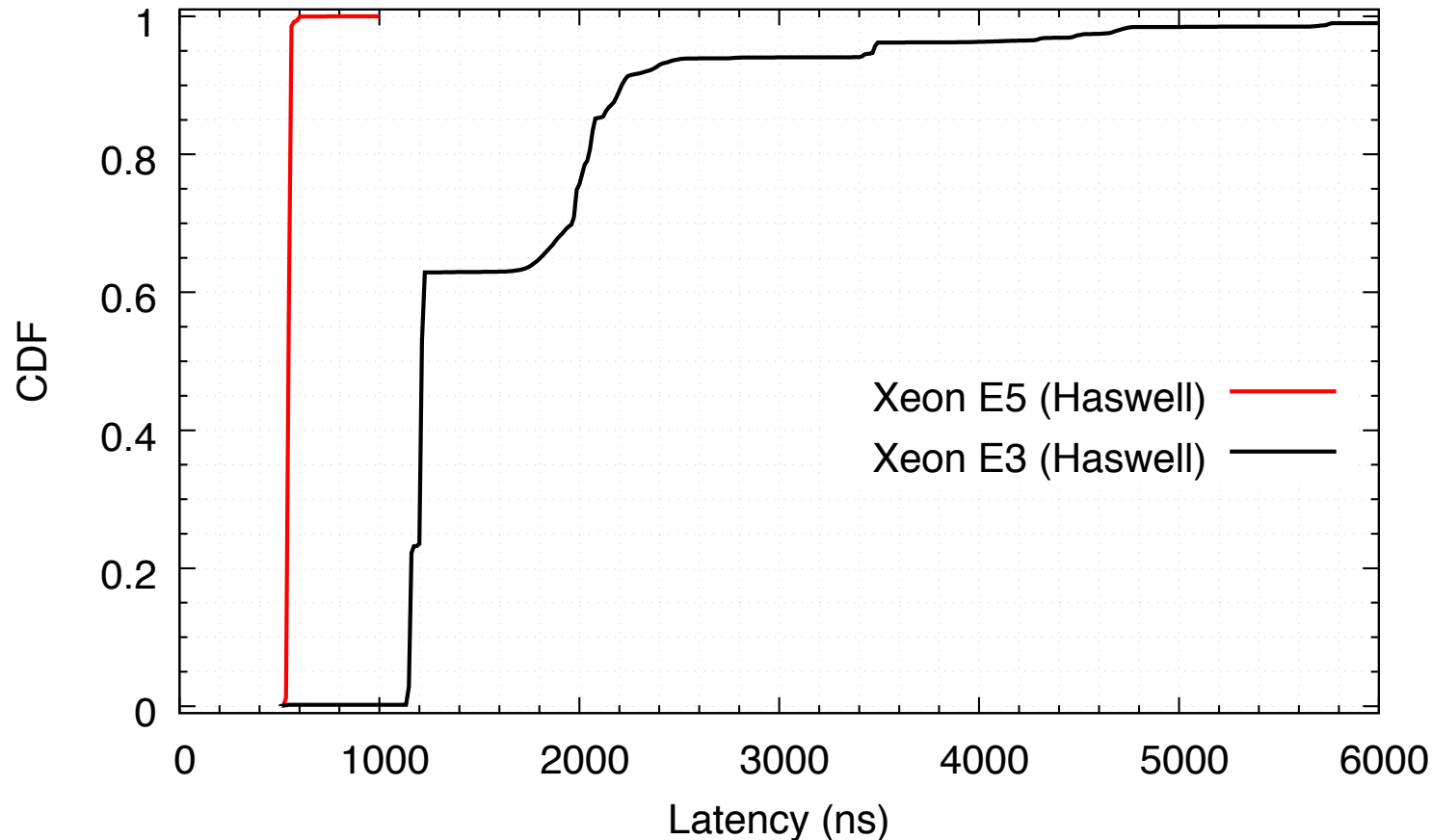
Xeon E3

- 1213ns(!) median
- 5707ns(!) 99th percentile
- 5.8ms(!!!) max

Netronome NFP-6000, Intel Xeon E5-2637v3 @ 3.5GHz (Haswell)

Netronome NFP-6000, Intel Xeon E3-1226v3 @ 3.3GHz (Haswell)

Distribution of 64B DMA Read latency



Xeon E5

- 547ns median
- 573ns 99th percentile
- 1136ns max

Xeon E3

- 1213ns(!) median
- 5707ns(!) 99th percentile
- 5.8ms(!!!) max

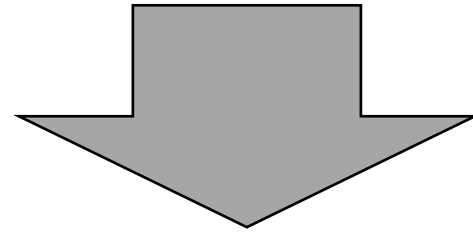
Your offload implementation has to handle this!

Netronome NFP-6000, Intel Xeon E5-2637v3 @ 3.5GHz (Haswell)

Netronome NFP-6000, Intel Xeon E3-1226v3 @ 3.3GHz (Haswell)

PCIe host implementation is evolving

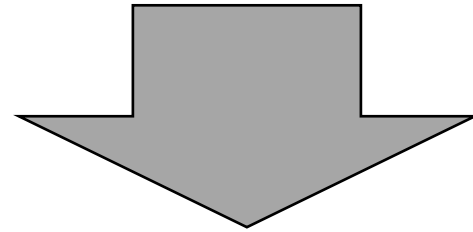
- Tighter integration of PCIe and CPU caches (e.g. Intel's **DDIO**)
- PCIe device is local to some memory (**NUMA**)
- **IOMMU** interposed between PCIe device and host memory



PCIe transactions are dependent on temporal state on the host and the location in host memory

PCIe host implementation is evolving

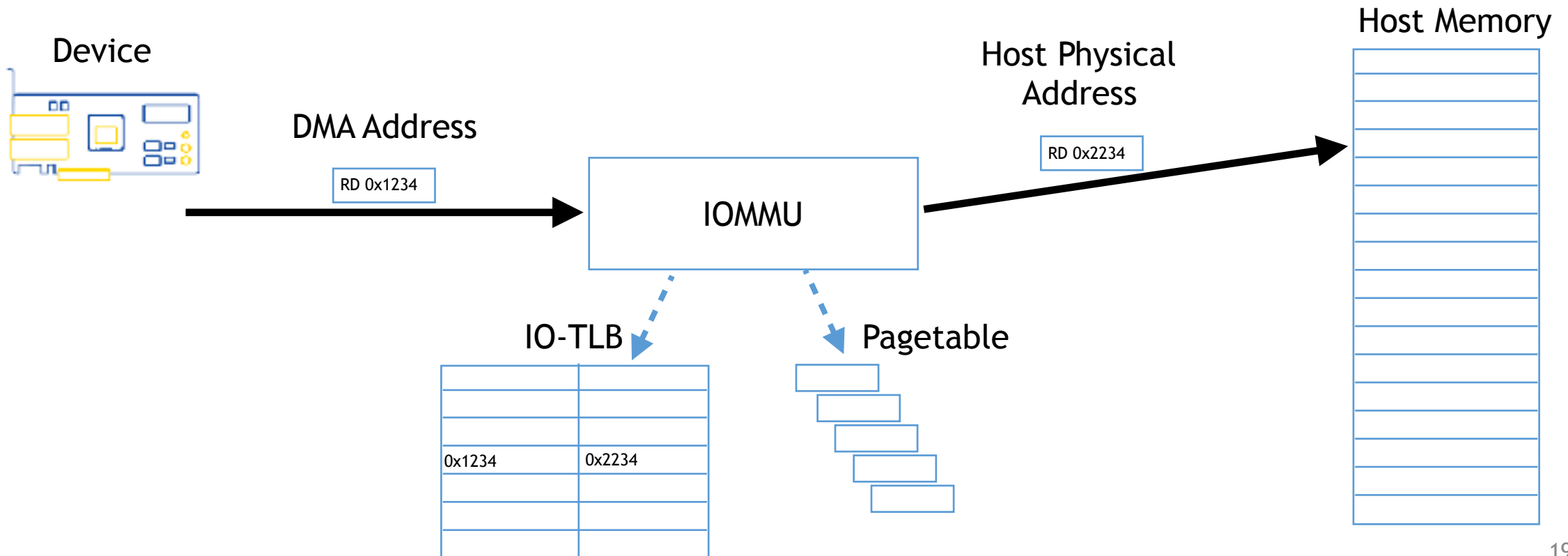
- Tighter integration of PCIe and caches (e.g. Intel's **DDIO**)
- PCIe is local to some memory (**NUMA**)
- **IOMMU** interposed between PCIe device and host memory



PCIe transactions are dependent on temporal state on the host and the location in host memory

PCIe data-path with IOMMU (simplified)

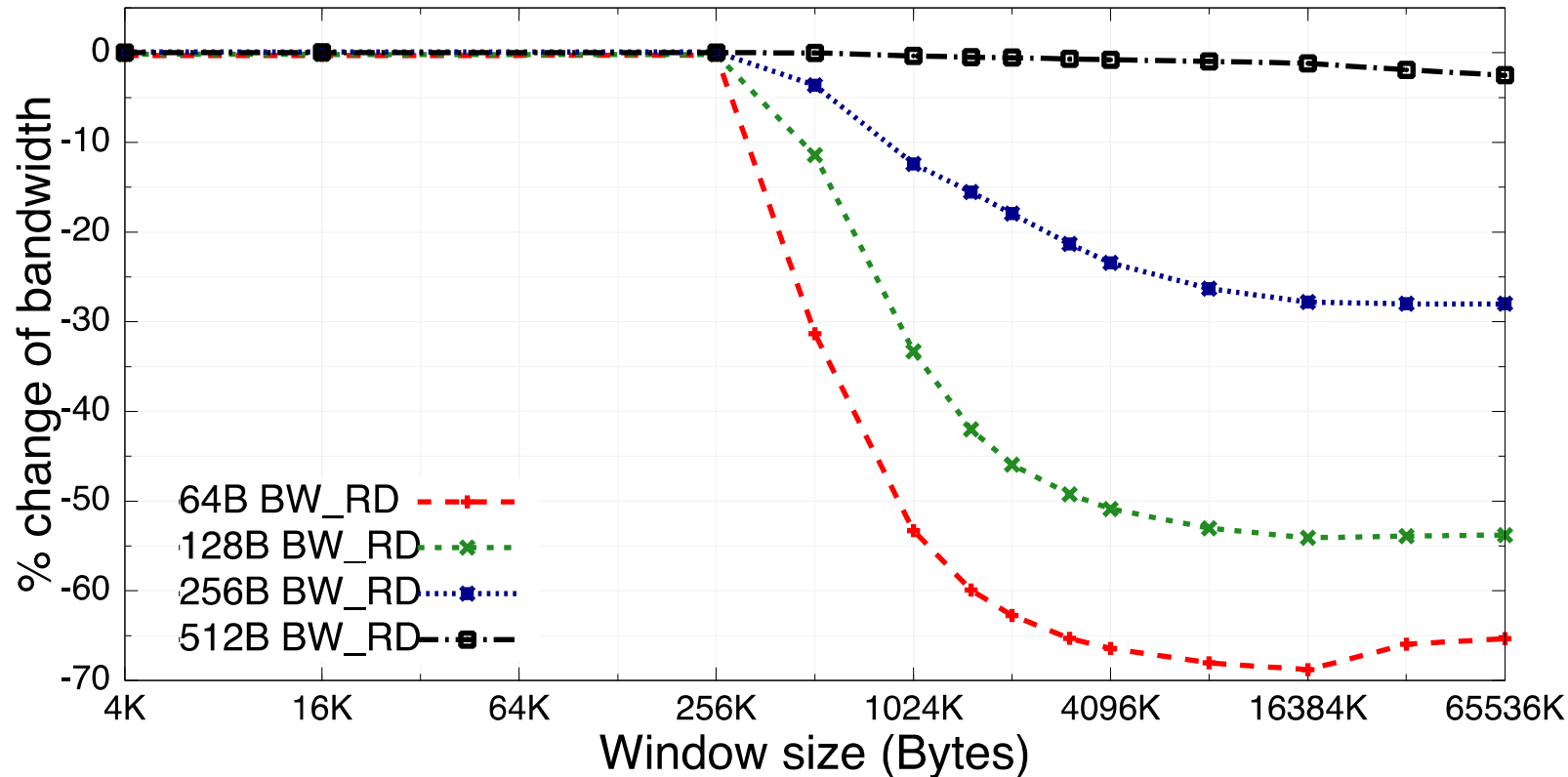
- IOMMUs translate addresses in PCIe transactions to host addresses
- Use a Translation Lookaside Buffer (TLB) as cache
- On TLB miss, perform a costly pageable walk, replace TLB entry



Measuring the impact of the IOMMU

- DMA reads of fixed size
- From random addresses on the host
- Systematically change the address range (window) we access
- Measure achieved bandwidth (or latency)
- Compare with non-IOMMU case

IOMMU results



- Different transfer sizes
- Throughput drops dramatically once region exceeds 256K.
- TLB thrashing
- TLB has 64 entries (256KB/4096B)
Not published by Intel!
- Effect more dramatic for smaller transfer sizes

Understanding PCIe performance is important

- A plethora of tools exist to analyse and understand OS and application performance
 - ... but very little data available on PCIe contributions
- Important when implementing offloads to programmable NICs
 - ... but also applicable to other high performance IO devices such as ML accelerators, modern storage adapters, etc

Introducing pcie-bench

- A **model** of PCIe to quickly analyse **protocol** overheads
- A suite of **benchmark tools** in the spirit of lmbench/hbench
- Records latency of individual transactions and bandwidth of batches
- Allows to systematically change
 - Type of PCIe transaction (PCIe read/write)
 - Transfer size of PCIe transaction
 - Offsets for host memory address (for unaligned DMA)
 - Address range and NUMA location of memory to access
 - Access pattern (seq/rand)
 - State of host caches
- ▶ Provides detailed insights into PCIe host and device implementations

Two independent implementations

- Netronome NFP-4000 and NFP-6000
 - Firmware written in Micro-C (~1500 loc)
 - Timer resolution 19.2ns
 - Kernel driver (~400 loc) and control program (~1600 loc)
- NetFPGA and Xilinx VC709 evaluation board
 - Logic written in Verilog (~1200 loc)
 - Timer resolution 4ns
 - Kernel driver (~800 loc) and control program (~600 loc)

[implementations on other devices possible]

Conclusions

- The **PCIe protocol** adds significant overhead esp for small transactions
- **PCIe implementations** have a significant impact on IO performance:
 - Contributes significantly to the latency (70-90% on ExaNIC)
 - Big difference between two the implementations we measured (what about AMD, arm64, power?)
 - Performance is dependent on temporal host state (TLB, caches)
 - Dependent on other devices?
- Introduced **pcie-bench** to
 - understand PCIe performance in detail
 - aid development of custom NIC offload and other IO accelerators
- Presented the first detailed study of PCIe performance in modern servers

Thank you!

Source code and all the data is available at:

<https://www.pcie-bench.org>

<https://github.com/pcie-bench>