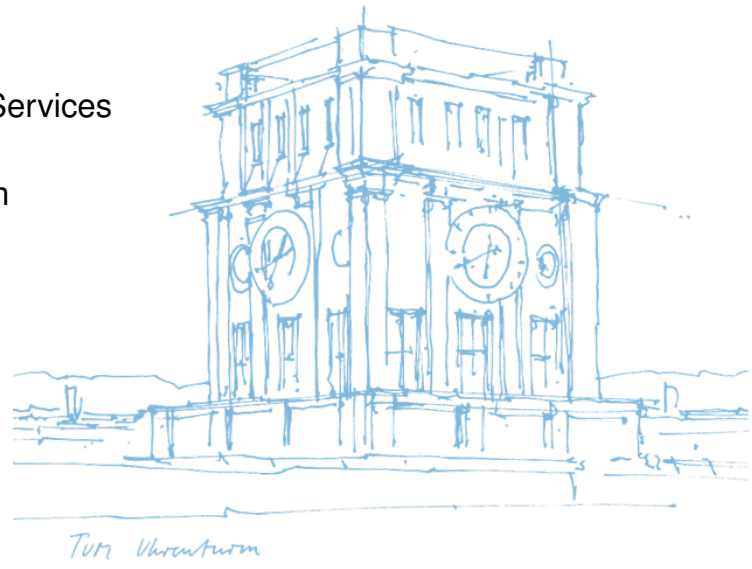
Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Investigating effects of hardware isolation in high-speed network environments

Simon Ellmann
advised by Paul Emmerich, Benedikt Jaeger, Florian Wiedner

Wednesday 12th May, 2021

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich



Motivation

What is hardware isolation?

Limiting access of hardware (and software) to needed resources

Motivation

Why hardware isolation?

Traditional hardware model:

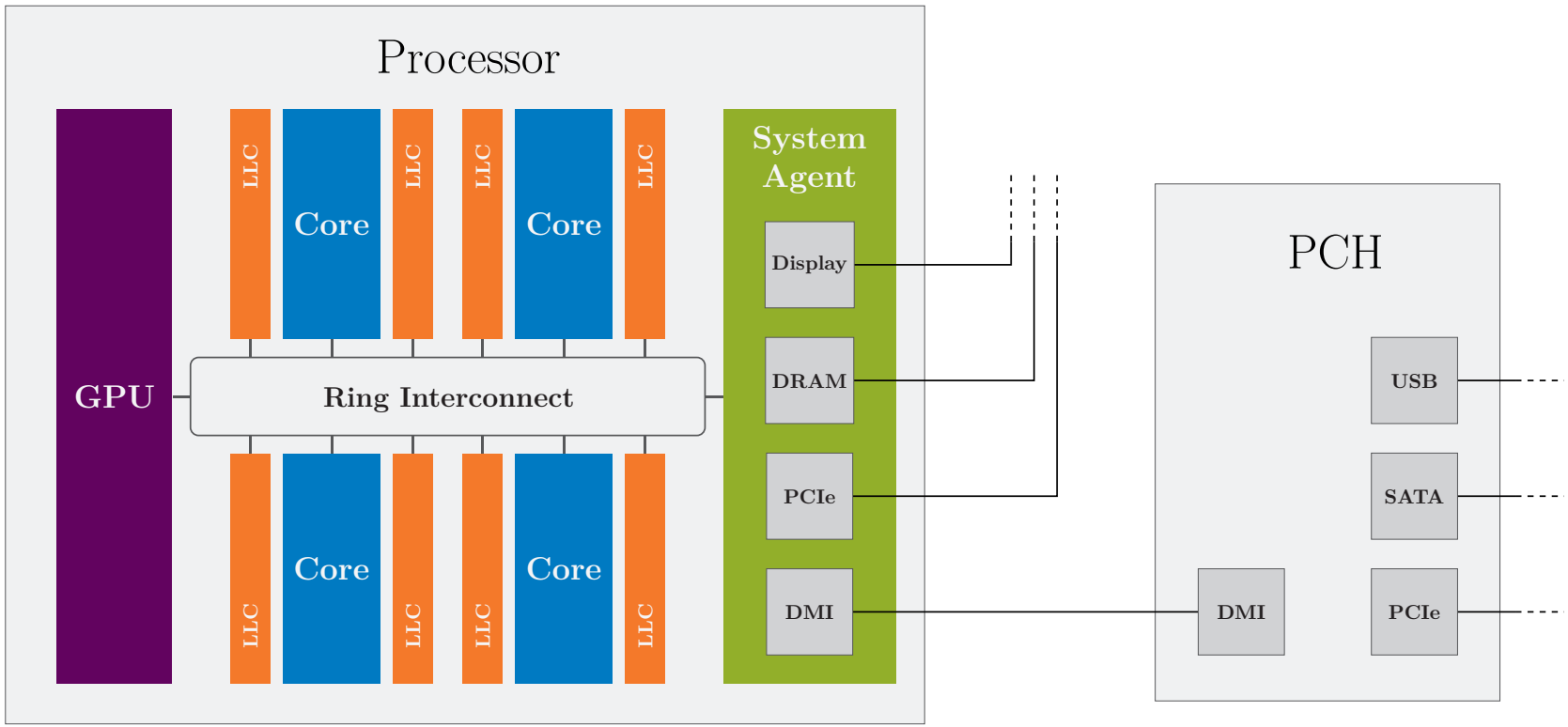
- Internal and external hardware is considered trustworthy
- Hardware has unrestricted access to other hardware, e.g., network cards to main memory
- Software with direct access to hardware (e.g., drivers) may (ab-)use the capabilities of the controlled hardware

Unrestricted hardware access to memory is a problem:

- Malicious devices or drivers may leak secret data from memory
- Faulty devices or drivers may corrupt OS data structures and crash the system

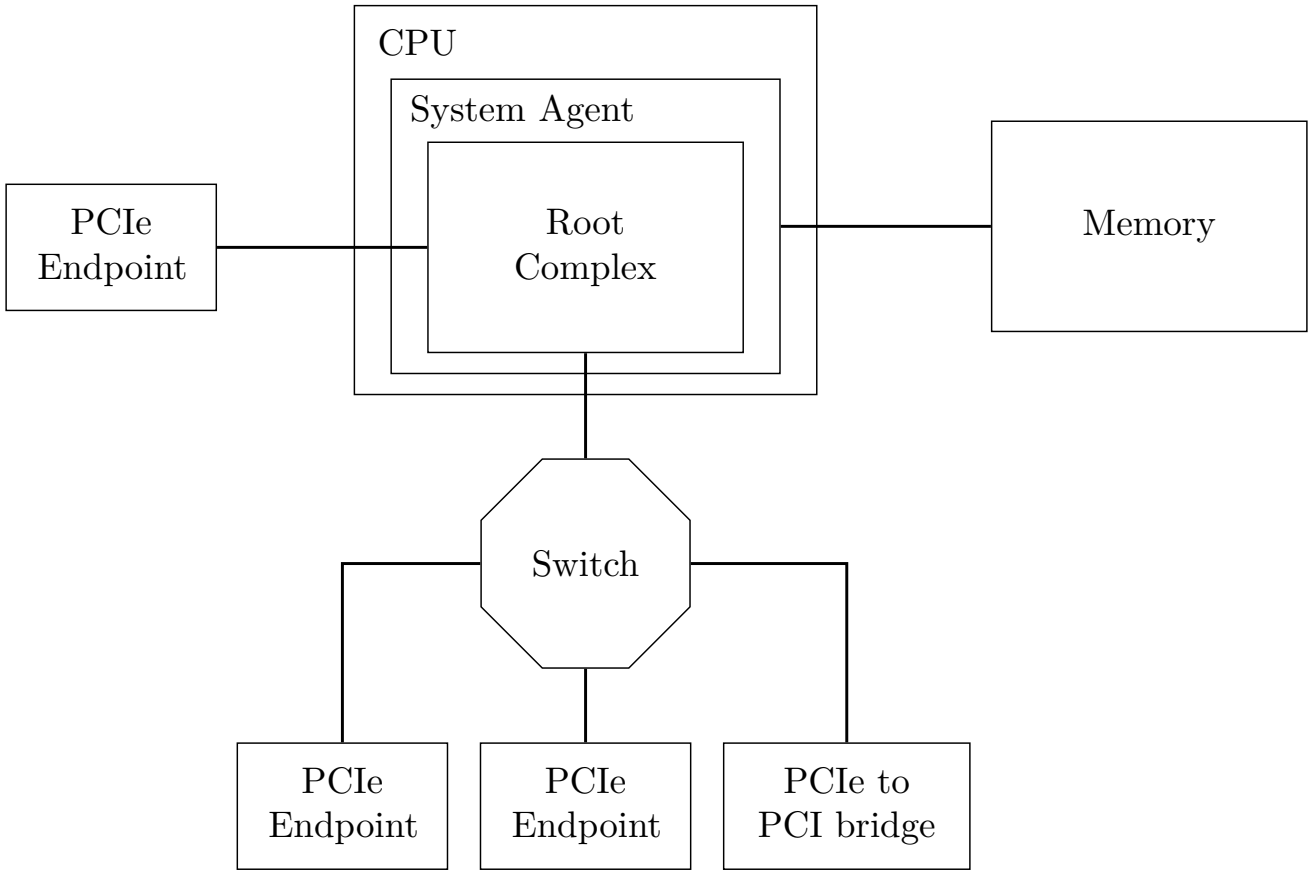
Motivation

How is hardware interconnected on modern systems?



Motivation

How is hardware interconnected on modern systems?



Motivation

How do peripherals access main memory?

- Devices use direct memory access (DMA) to access main memory without CPU involvement
- In case of PCIe, devices issue memory read/write requests to the root complex
- DMA requests address memory through physical addresses

Motivation

Hardware can be isolated through IOMMUs

Input-Output Memory Management Units (IOMMUs):

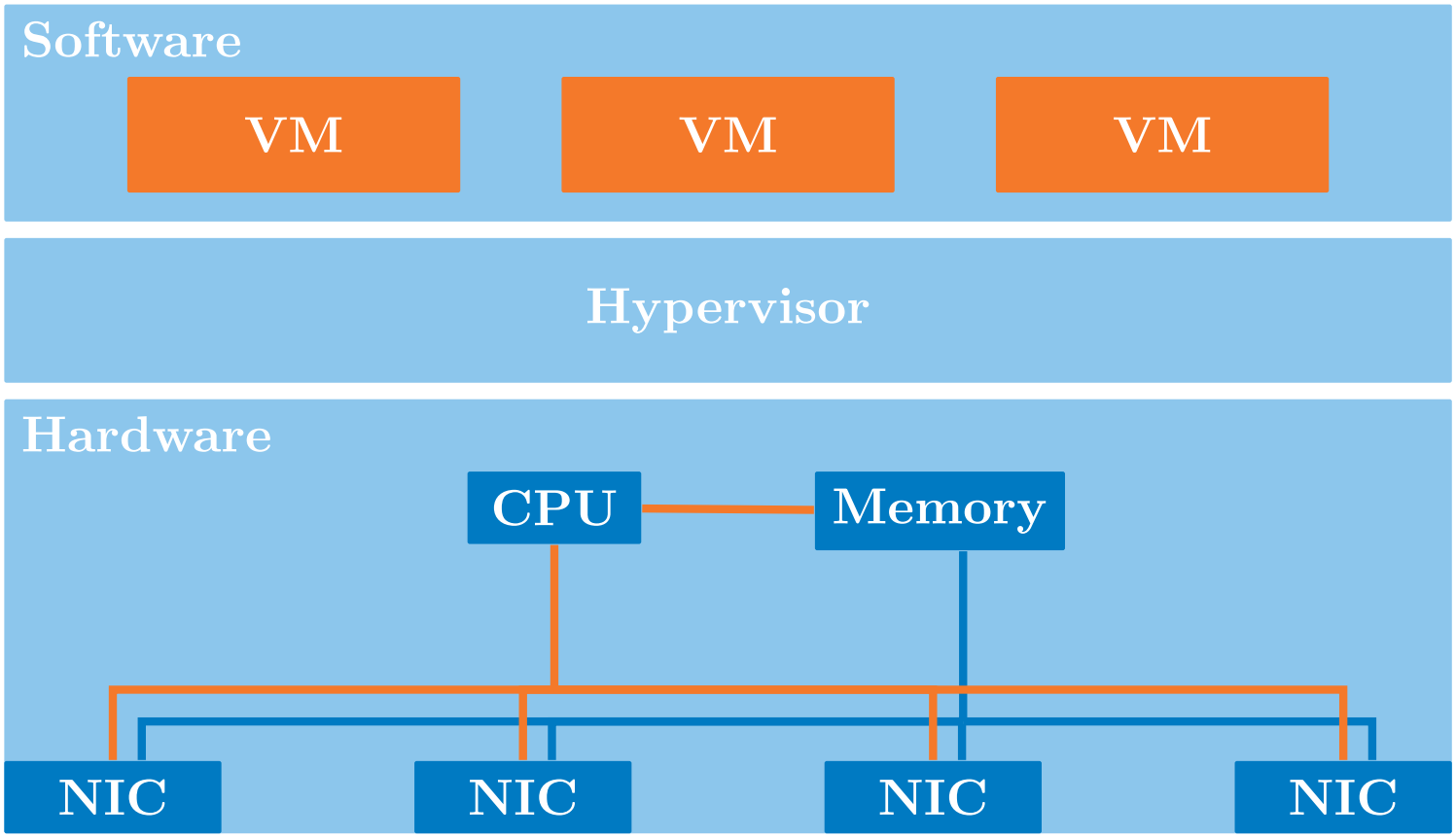
- Translate IO virtual addresses (IOVA) to physical addresses (PA)
- Restrict access of IO devices to assigned address space

Advantages:

- Limit effects of faulty or malicious devices/software
- Contiguous virtual address space does not have to be contiguous in physical memory
- Enable 32-bit devices to address memory above 4 GiB
- Allow for safe and secure direct passthrough of hardware to virtual machines

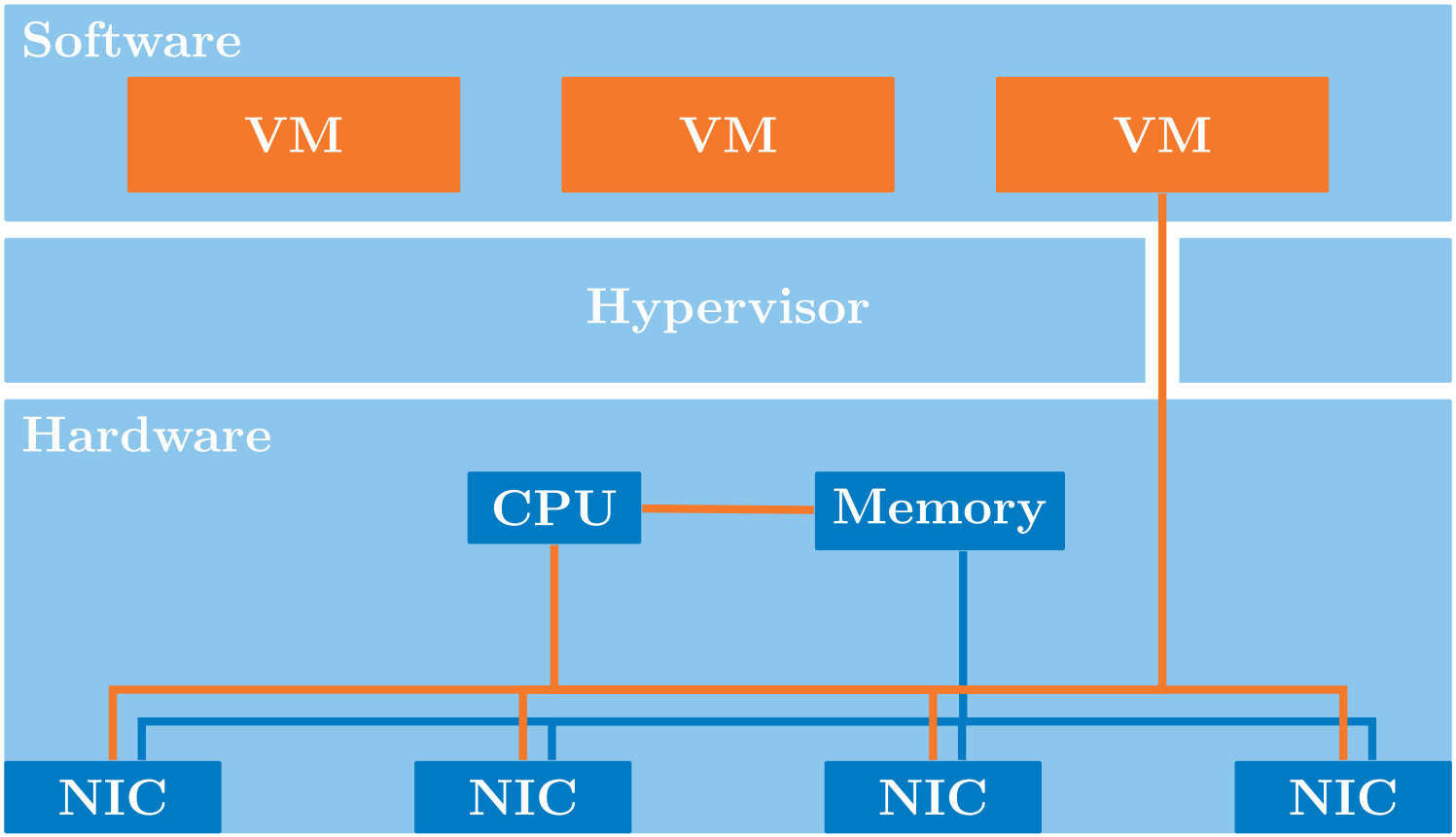
IOMMUs

Virtualization



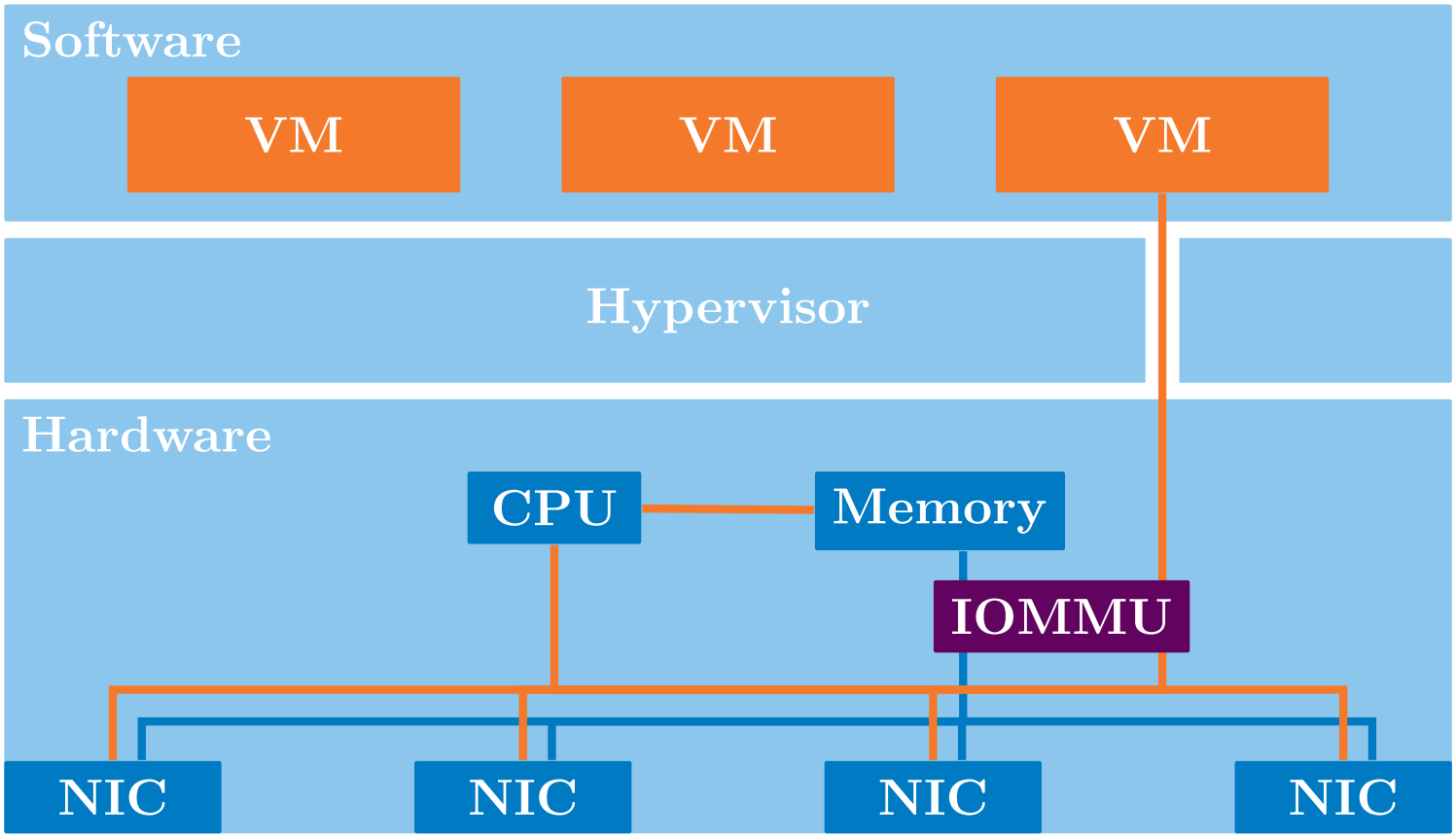
IOMMUs

Virtualization



IOMMUs

Virtualization



IOMMUs

Location in the PCIe tree

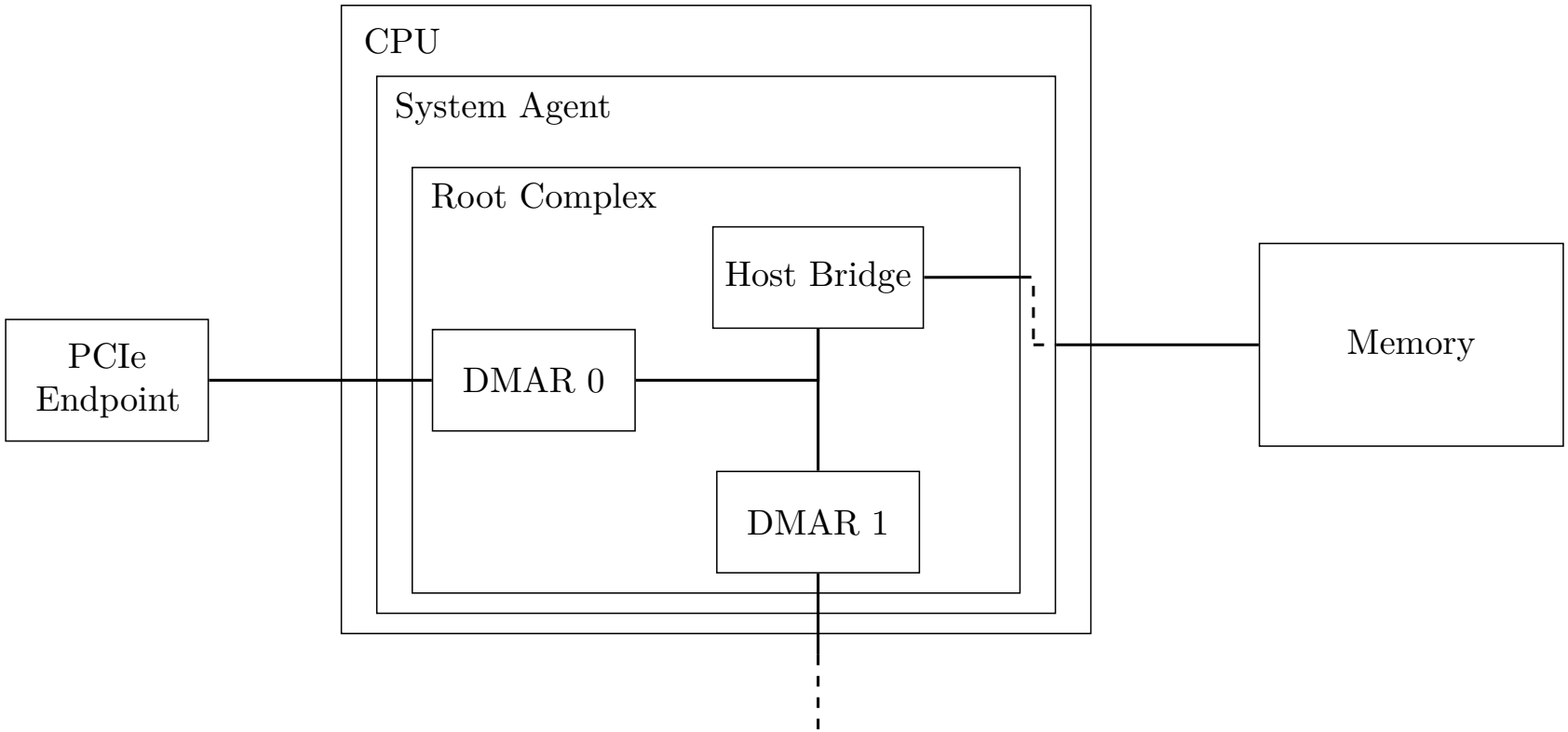


Figure 1: IOMMUs (DMARs) in the PCIe tree.

IOMMUs

Real world use cases

- Amazon's Elastic Compute Cloud provides 10G networking through SR-IOV + IOMMUs
- MacBooks use IOMMUs to protect themselves against external Thunderbolt devices
- iPhones have an IOMMU included to isolate the WiFi stack

Research Questions

Reasons to have a closer look at IOMMUs:

- Documentation about IOMMU implementations is sparse
- Some publications report huge performance impacts and vulnerabilities
- Differences of IOMMUs from different vendors (Intel, AMD, ARM, ...) mostly unknown
- Increasing prevalence of IOMMUs in servers, home computers and mobile devices

Key question of our work:

- What is the trade-off between performance and safety/security in high-speed network environments?

Methodology

How to determine performance effects of IOMMUs?

- Desirable properties in high-speed networks are high packet throughput rates and low latency
- To determine negative effects on these properties we need some application that provides high throughput rates and low latency
- Why not use a user space network driver?

Methodology

Test setup

Benchmarking IOMMUs with ixy.rs:

- state-of-the-art user space network driver
- can forward >26 Mpps on a single 3,3 GHz CPU core
- less than 2,000 lines of code
- written in Rust

Missing in ixy.rs:

- Support for SR-IOV
- Support for legacy IOMMUs with small IOVA widths

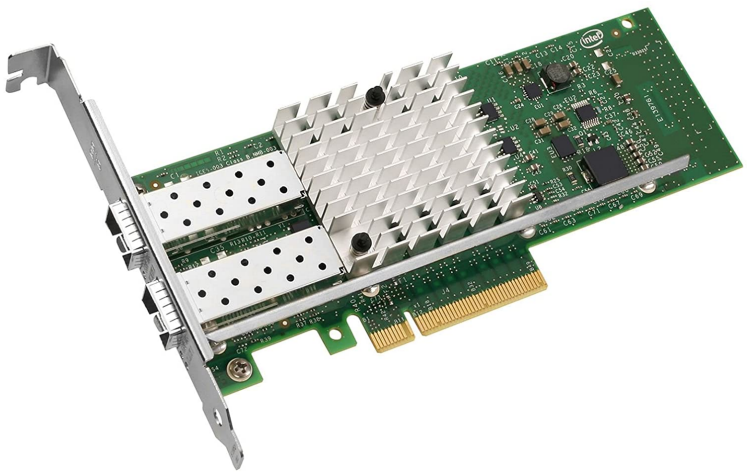
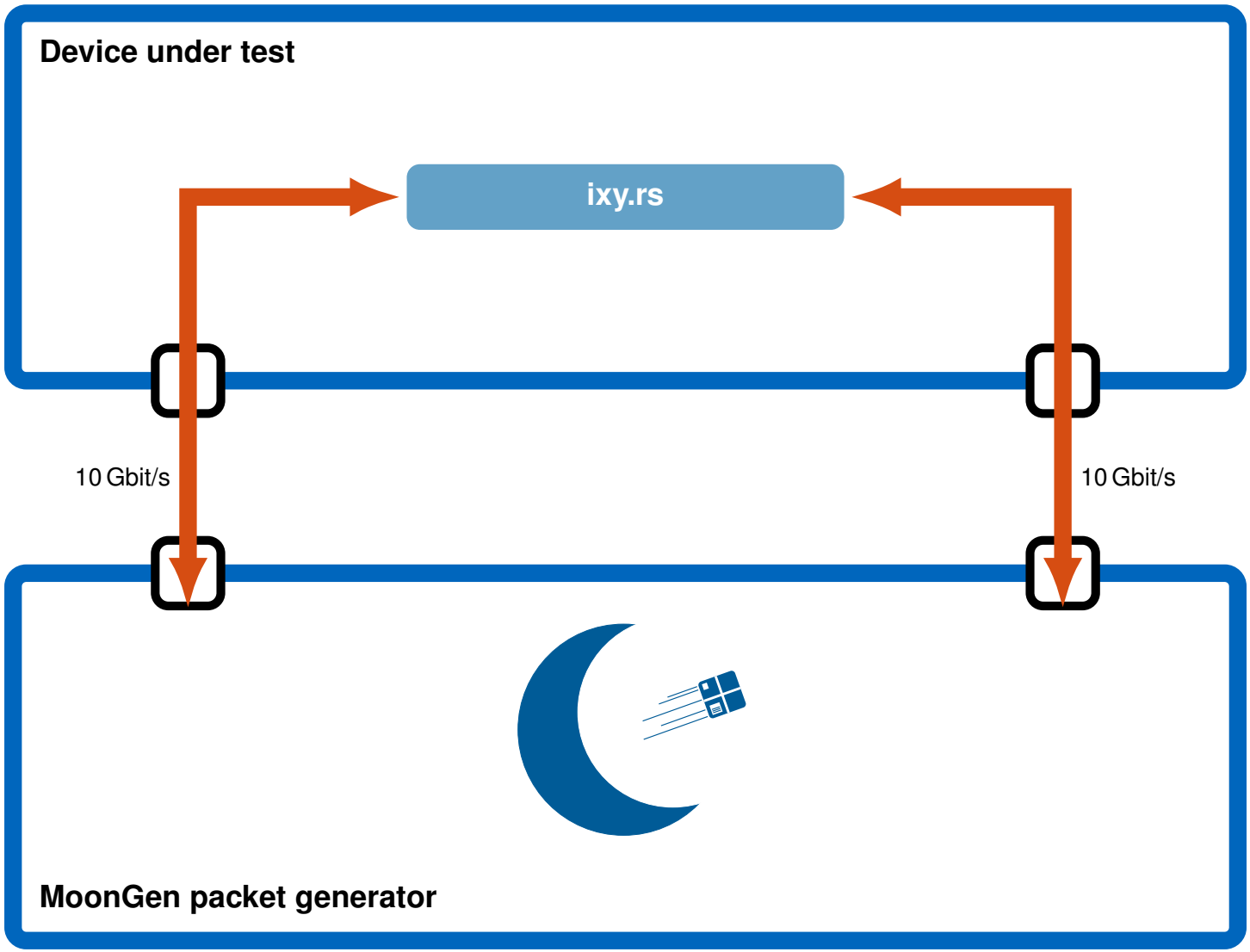


Figure 2: Intel X520-DA2 [Picture: amazon.com]

Methodology

Test setup



Methodology

Devices under test

CPU	Year	Arch.	Memory	NIC	NUMA
Intel Xeon E3-1230v2	2012	Ivy Bridge	16 GB	Intel X520-DA1 Intel X520-DA2	no
Intel Xeon E5-2620v3	2014	Haswell	32 GB	Intel X520-DA2 Intel X540-T2	no
AMD EPYC 7551P	2017	Naples	128 GB	Intel X550T Intel X550T	yes

Methodology

Devices under test

CPU	Clock	Cores	L3-Cache	PassMark ST	PassMark All
Intel Xeon E3-1230v2	3,3 GHz	4	8 MB	1,996	6,192
Intel Xeon E5-2620v3	2,4 GHz	6	15 MB	1,700	7,979
AMD EPYC 7551P	2,0 GHz	32	64 MB	1,611	25,933

Performance Analysis

Baseline performance of devices under test

Idea:

- Determine performance of devices under test without modifications
- Detect unusual behaviour, e.g., exceptionally good/bad performance

Performance Analysis

Baseline performance of devices under test

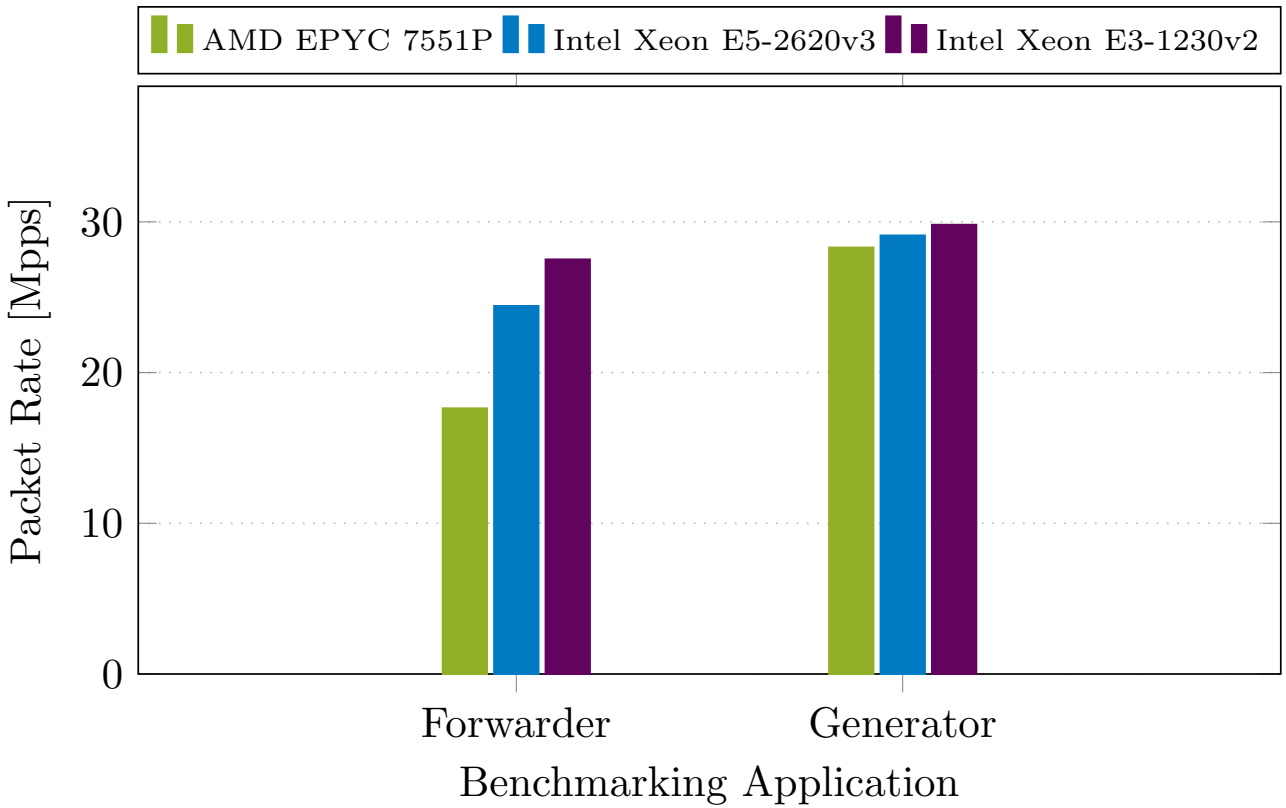


Figure 3: Baseline throughput of ixy.rs on devices under test without IOMMU.

Performance Analysis

Baseline performance of devices under test

Results:

- Throughput rates reflect clock speeds of the CPUs
- Forwarder on AMD EPYC CPU is significantly slower than expected

Reasons for the relatively poor performance of AMD EPYC:

- 1/6 less instructions per cycle than Intel Xeon E5
- 12-39x more branch mispredictions
- probably 1-2 additional cycles per branch misprediction (at least on Zen 2)

Performance Analysis

Throughput with different page sizes

Idea:

- IOMMUs cache translations in I/O translation lookaside buffers (IOTLBs)
- Can we determine the number of pages cached in the IOTLB?
- With too many pages, throughput may drop due to the IOTLB getting thrashed

Performance Analysis

Throughput with different page sizes

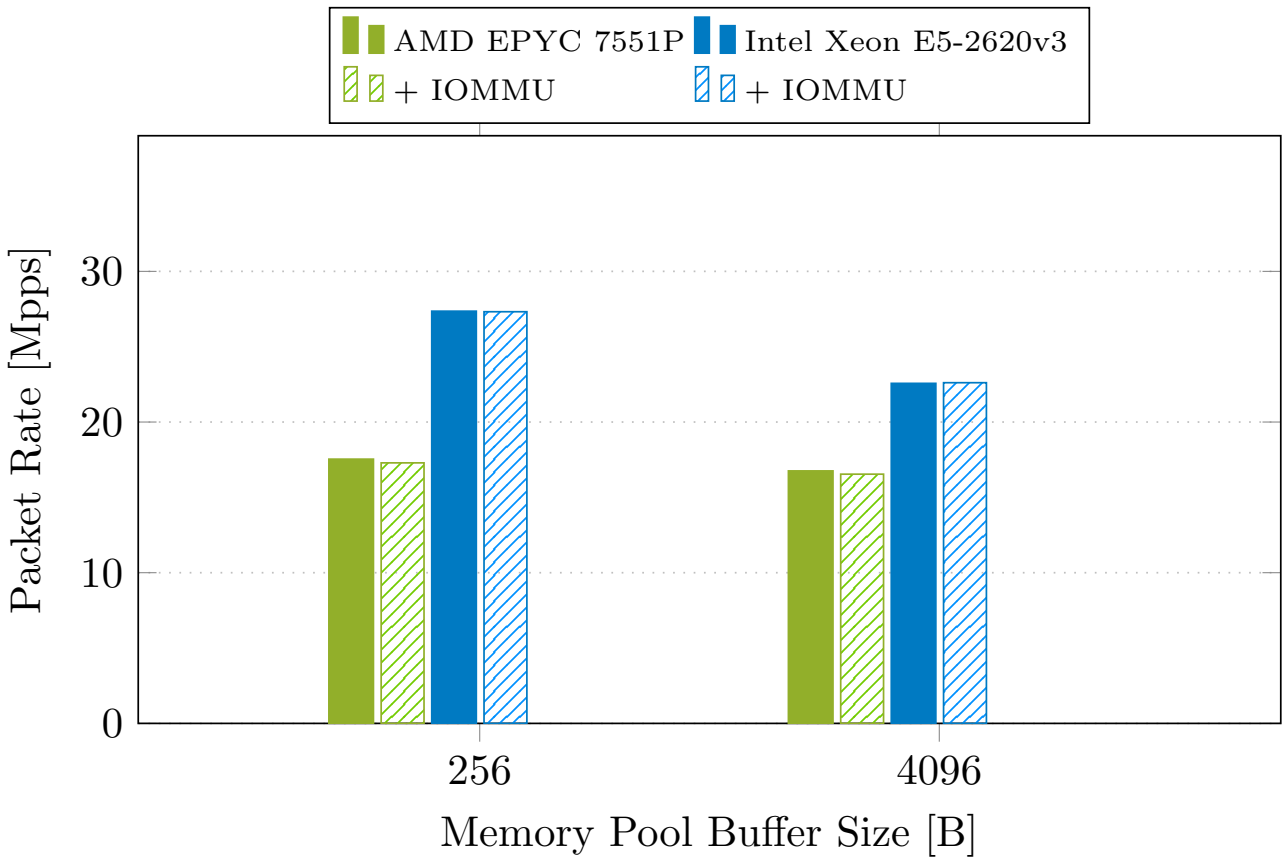


Figure 3: Throughput of forwarder with 1 GiB pages. Two pools with 512 buffers each used.

Performance Analysis

Throughput with different page sizes

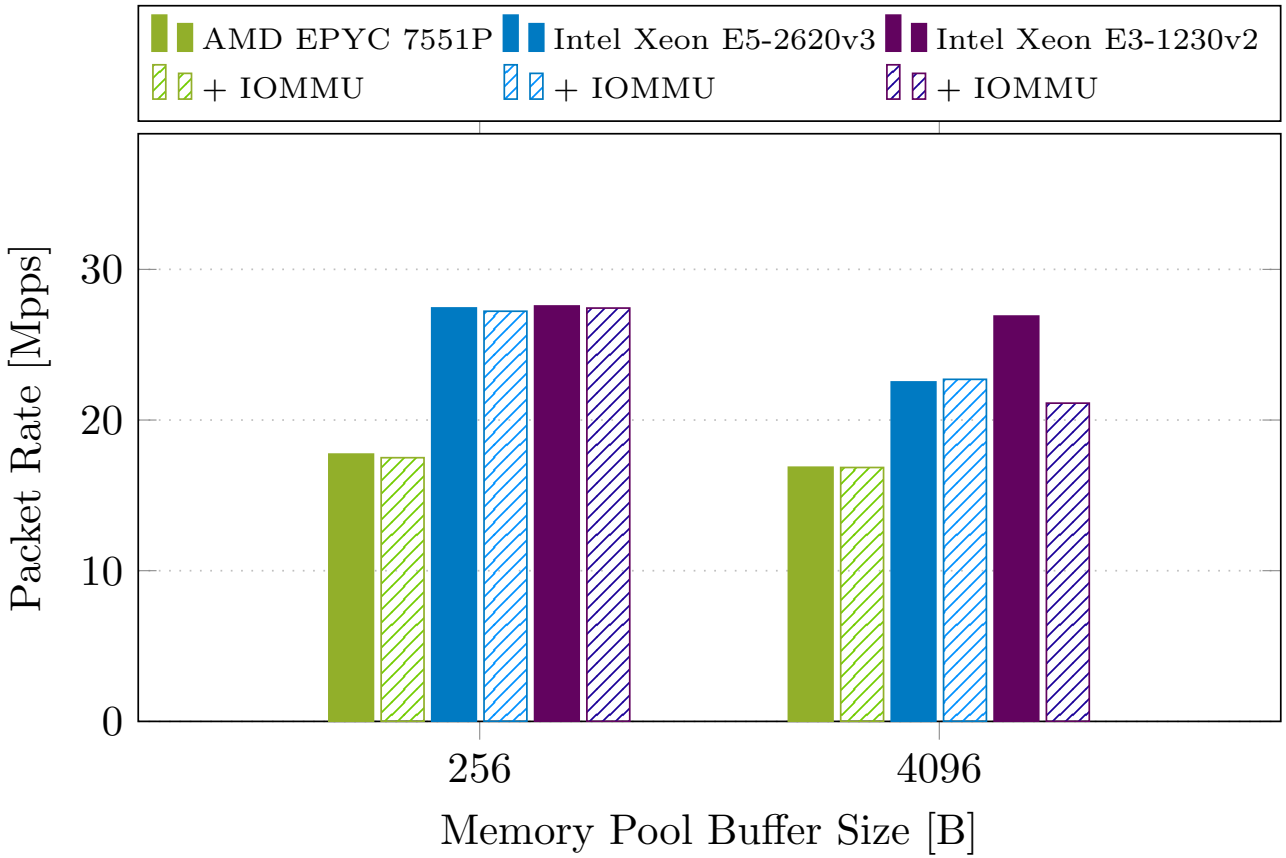


Figure 3: Throughput of forwarder with 2 MiB pages. Two pools with 512 buffers each used.

Performance Analysis

Throughput with different page sizes

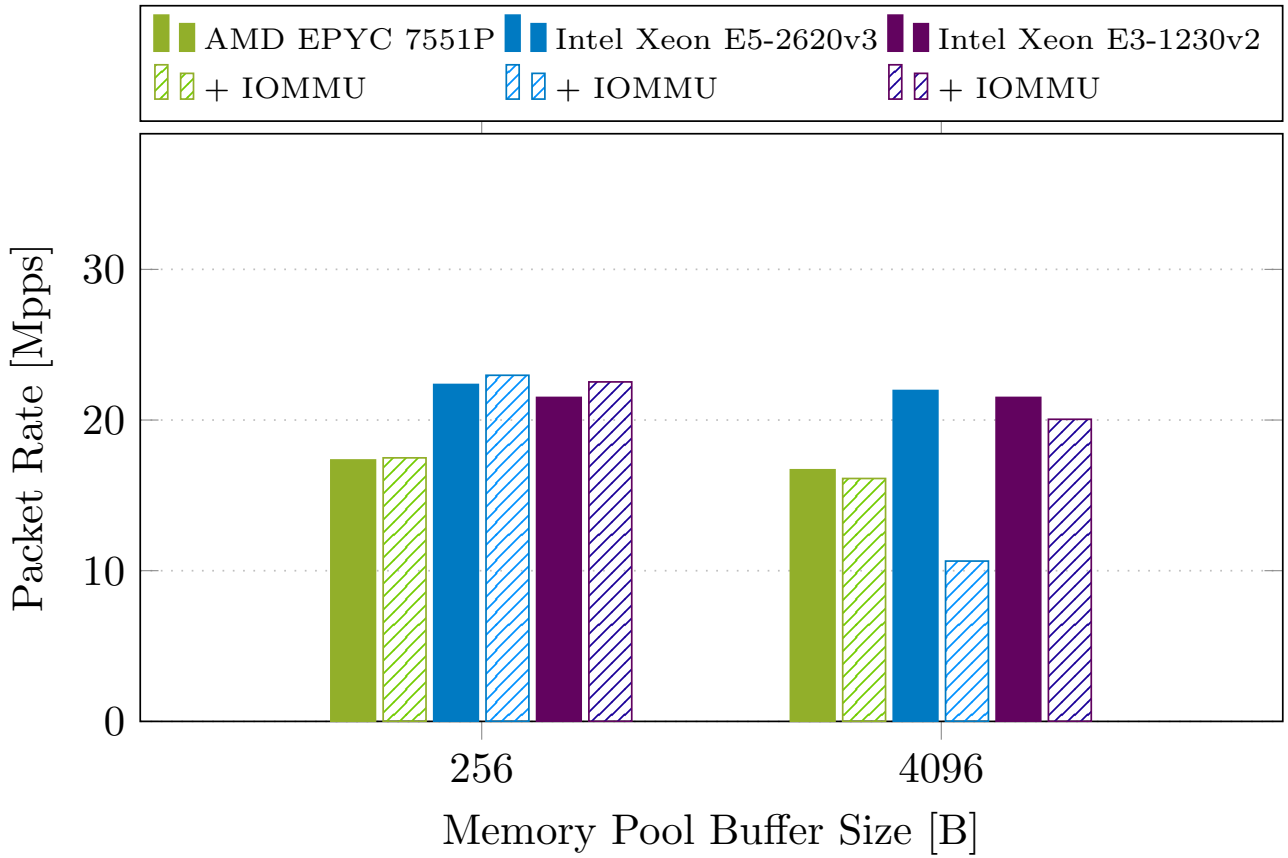


Figure 3: Throughput of forwarder with 4 KiB pages. Two pools with 512 buffers each used.

Performance Analysis

Throughput with different page sizes

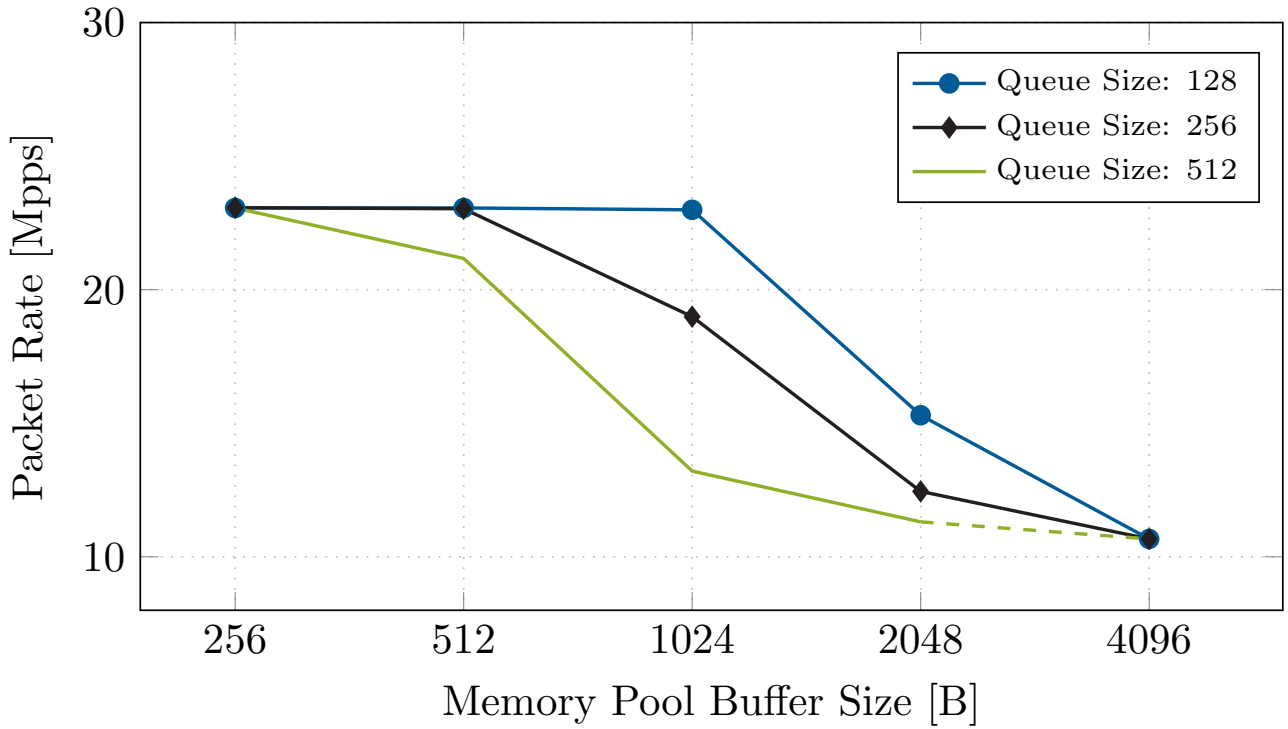
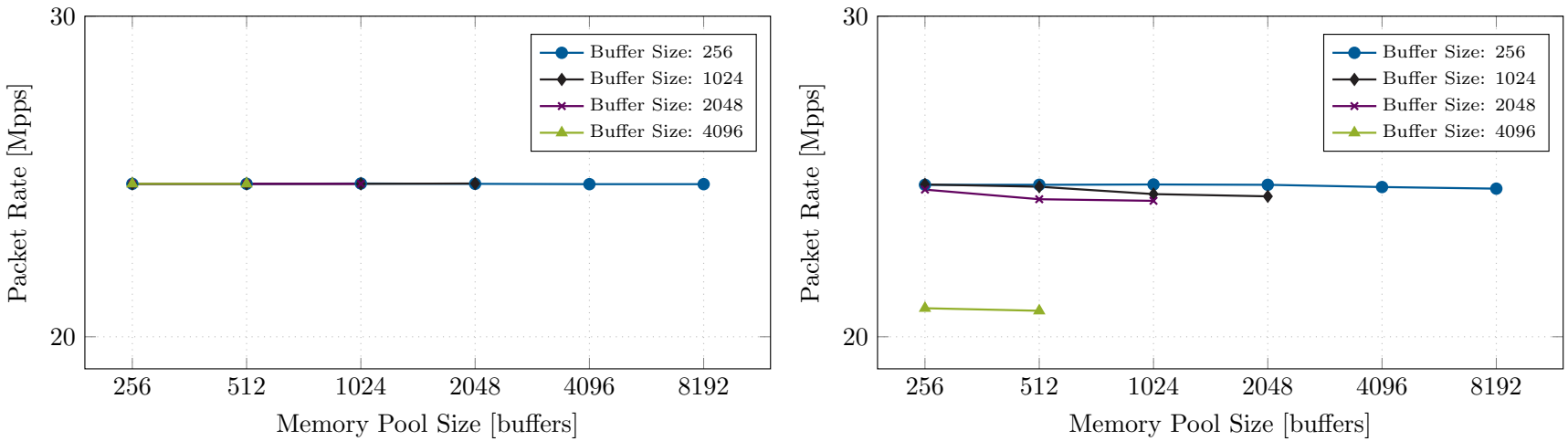


Figure 3: Throughput of forwarder on Intel Xeon E5 with 4 KiB pages and IOMMU enabled. Pool size = 2 x queue size.

Performance Analysis

Throughput with different page sizes



(a) No IOMMU

(b) IOMMU

Figure 3: Packet rate of generators on Intel Xeon E5 with 4 KiB pages.

Performance Analysis

Throughput with different page sizes

Results:

- Page sizes have no major effect on throughput on AMD EPYC and Intel Xeon E3
- IOTLB seems to get thrashed on the Intel Xeon E5-2620v3 when using more than 64 pages
- Neugebauer et al. determined an IOTLB size of 64 entries for their Intel Xeon E5-2630v4
- However, our packet generator does not show any correlation between number of used pages and packet rate

Performance Analysis

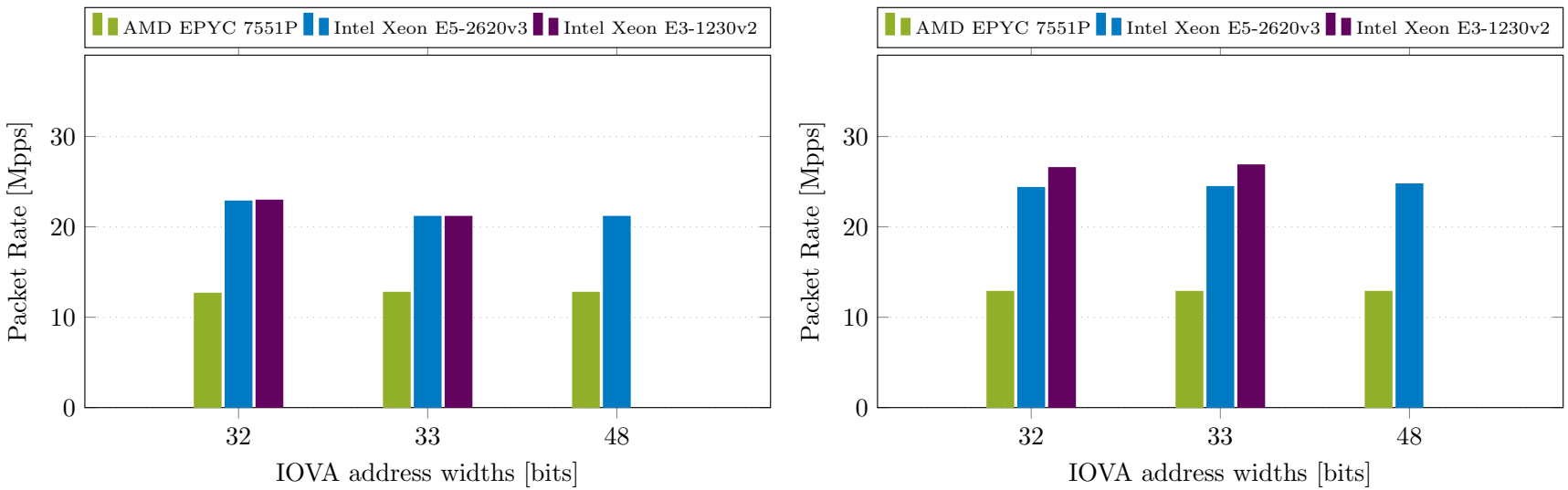
Throughput with different IOVA address widths

Idea:

- Depending on IOVA address widths, IOMMUs use different translation structures
- Does the number of page tables to be walked affect performance?

Performance Analysis

Throughput with different IOVA address widths



(a) Using 2 ports of the same NIC

(b) Using 2 ports of different NICs

Figure 3: Throughput of forwarder with 32, 33 and 48 bit wide IOVAs.

Performance Analysis

Throughput with different IOVA address widths

Results:

- IOVA address widths do not affect performance when using two different NICs
- Larger IOVA addresses negatively affect performance when using two ports of the same NIC
- We suspect that the PCIe bus limits throughput: PCIe packets with 32-bit addresses use 4 B smaller headers

Performance Analysis

Throughput with SR-IOV

Servers use Single-Root Input-Output Virtualization (SR-IOV) to pass a single network card to multiple virtual machines by splitting the device into multiple so-called virtual functions (VFs).

Idea:

- Do IOMMUs affect throughput when using SR-IOV?

Performance Analysis

Throughput with SR-IOV

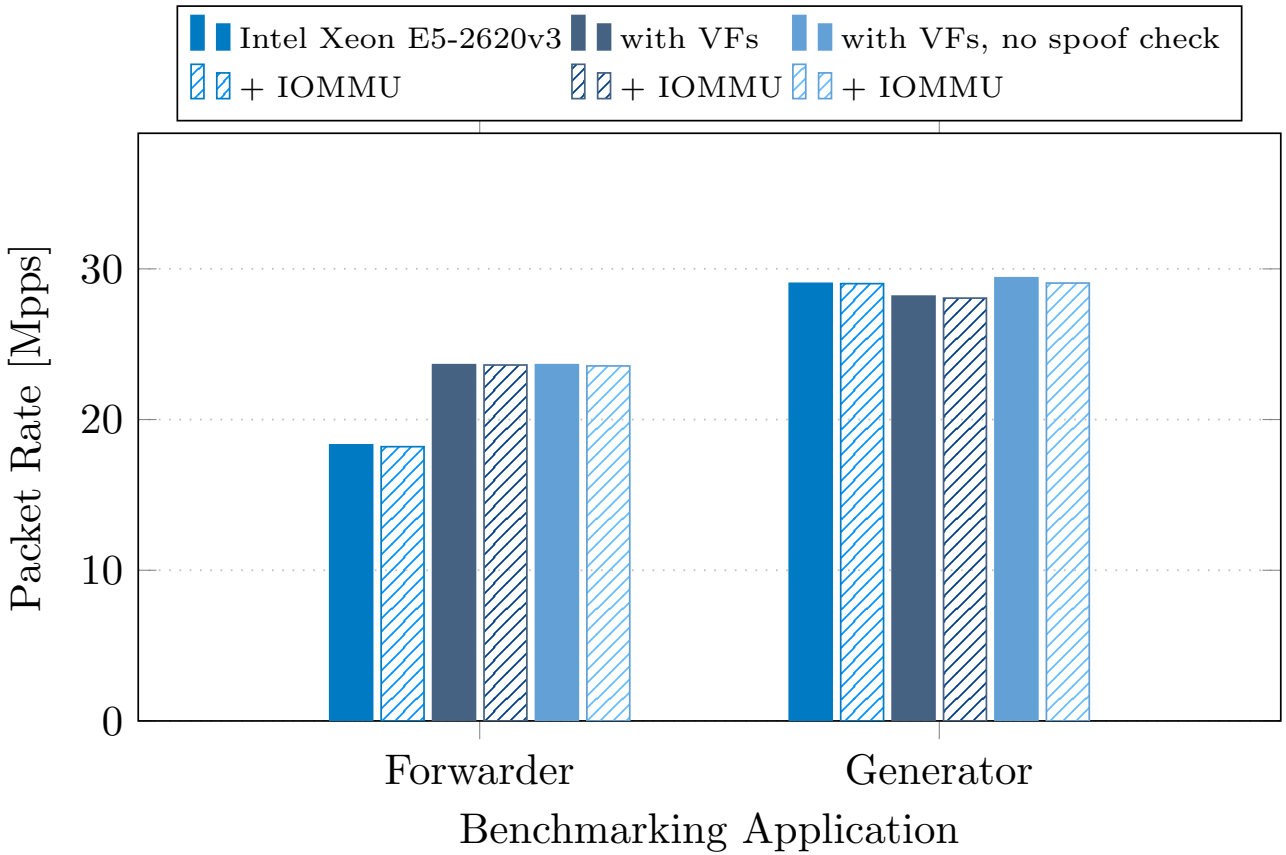


Figure 3: Throughput of SRIOV-modified forwarder on Intel Xeon E5.

Performance Analysis

Throughput with SR-IOV

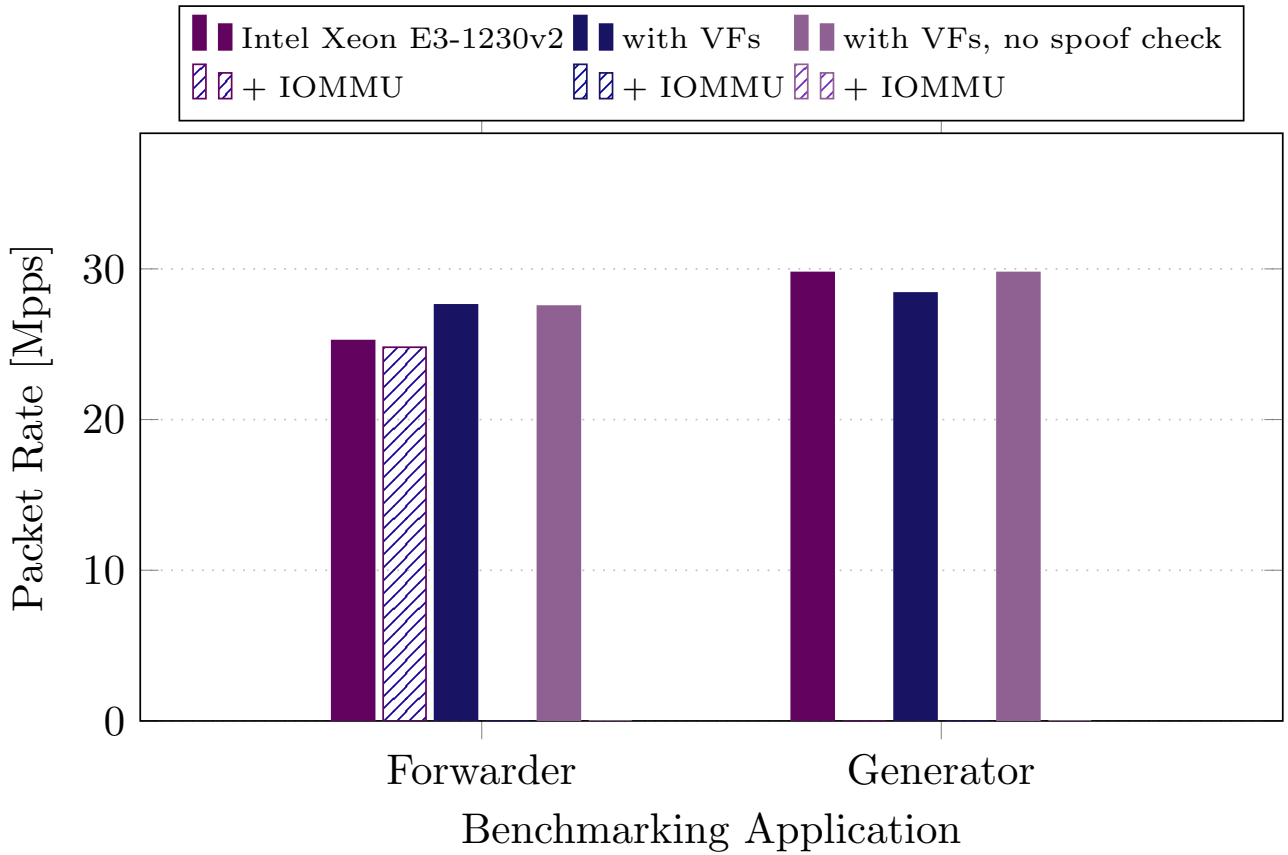


Figure 3: Throughput of SRIOV-modified forwarder on Intel Xeon E3.

Performance Analysis

Throughput with SR-IOV

Results:

- IOMMUs do not affect throughput using SR-IOV

Safety and Security

Known vulnerabilities

PCIe is inherently unsafe:

- PCIe devices can impersonate other devices by using their PCIe IDs
- PCIe devices may announce Address Translation Services (ATS) support

Address Translation Services (ATS):

- Devices cache IOMMU address translations to reduce pressure on IOTLB
- Addresses in PCIe requests are marked as already translated
- PCIe requests with translated addresses are not checked by the IOMMU, i.e., devices have unrestricted access to main memory
- Linux disabled ATS for external devices (e.g., Thunderbolt) in 2018

Safety and Security

Known vulnerabilities

Architectural deficiencies:

- OSes detect hardware using ACPI tables which could be overriden at boot time on some systems such that IOMMUs are hidden from the OS
- IOMMU translation tables could be overridden on some systems during IOMMU initialization, disabling translation for all devices

Flaws in IOMMU usage by OSes:

- Until October 2020, Windows did not use IOMMUs at all
- Linux and macOS put kernel data structures into IOMMU-protected memory such that root shells could be obtained on both OSes

Safety and Security

Are there more vulnerabilities?

Caches allow for timing-attacks:

- IOMMUs cache address translations in IOTLBs
- Can we detect whether a translation was cached in the IOTLB or a page table walk had to be performed?
- If so, can we prime the IOTLB such that we can detect whether other devices / virtual functions have performed a PCIe memory request, i.e., have received/transmitted packets?

Measuring DMA access times

Idea:

- Continuously transmit batches of packets such that IOTLB is fully used
- Measure CPU cycles it takes to transmit the packet batches
- If another NIC receives or transmits a packet, at least one IOTLB entry is replaced by the IOMMU
- Transmitting the next batch of packets should take more CPU cycles due to IOTLB misses

Safety and Security

Measuring DMA access times

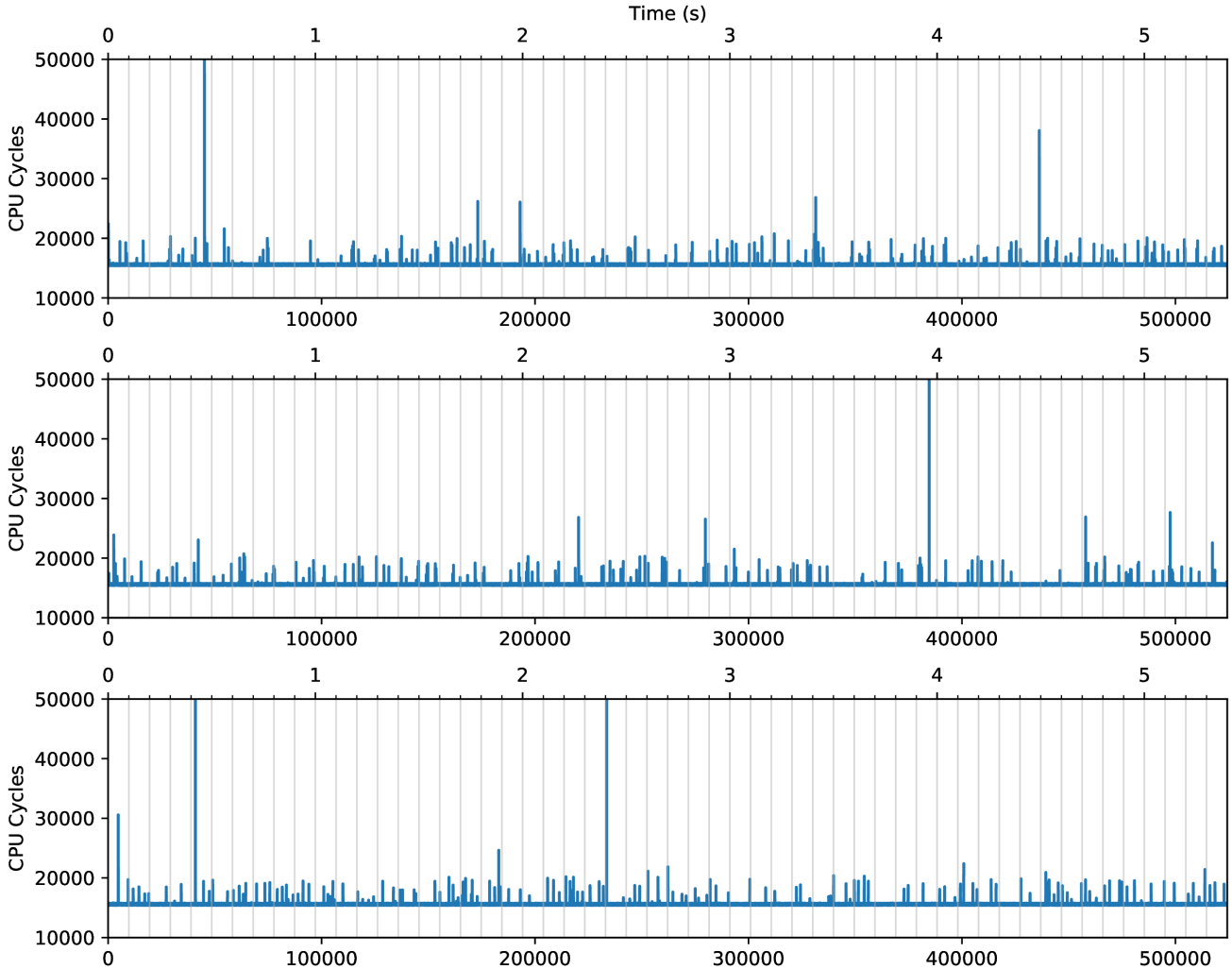


Figure 3: CPU cycles per batch of transmitted packets without IOMMU.

Safety and Security

Measuring DMA access times

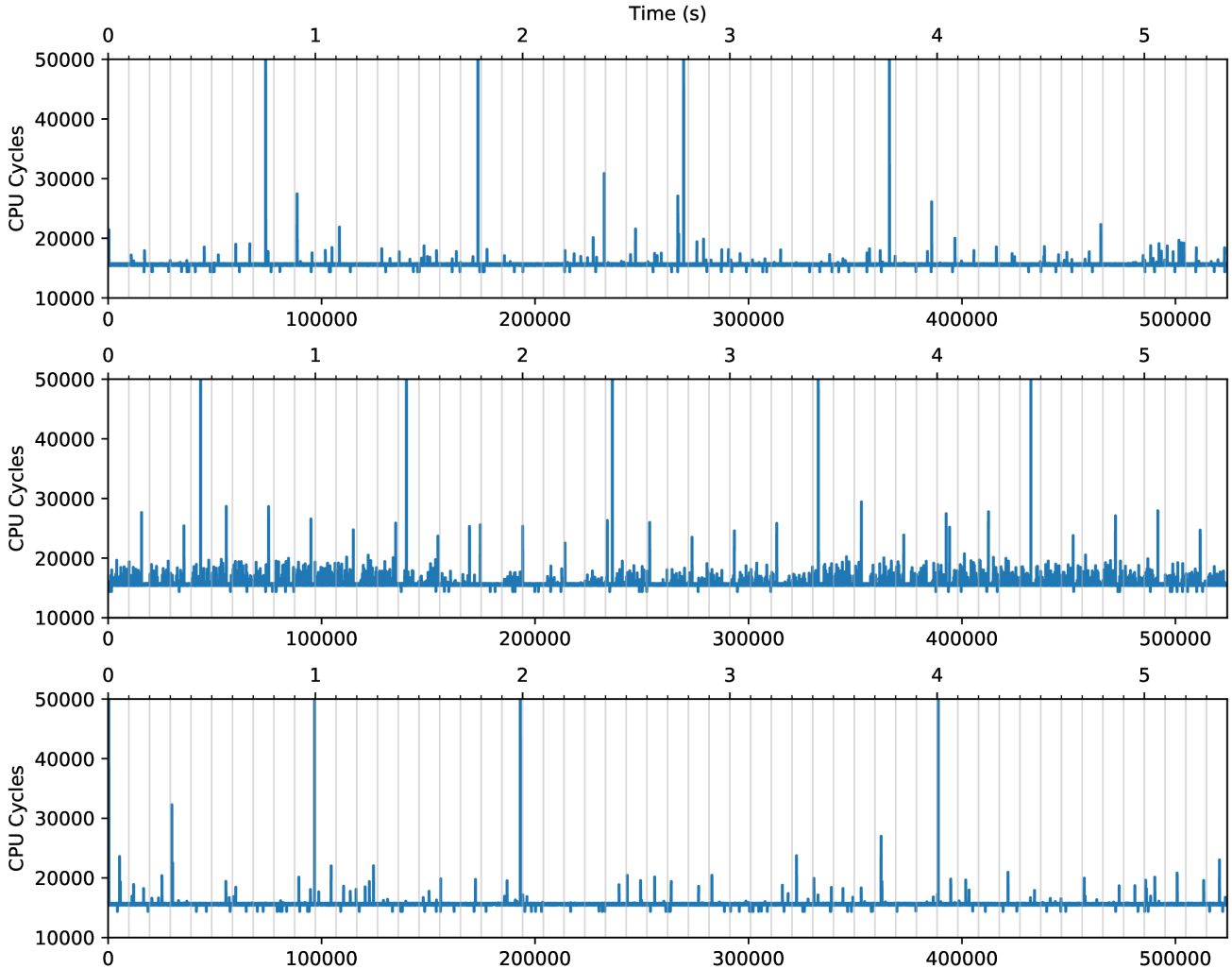


Figure 3: CPU cycles per batch of transmitted packets with IOMMU.

Safety and Security

Measuring DMA access times

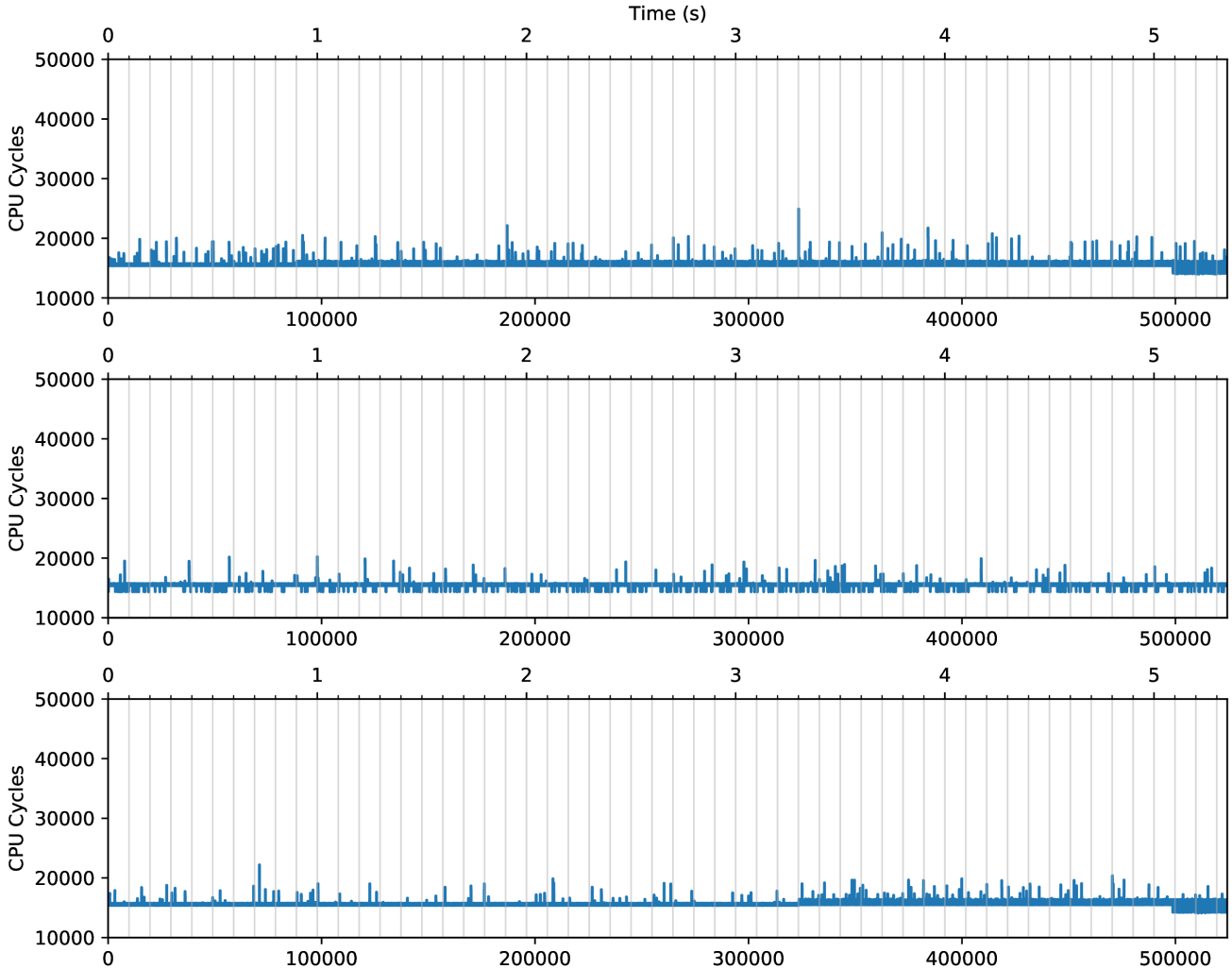


Figure 3: CPU cycles per batch of transmitted packets with IOMMU and fixed virtual and physical addresses, other PCIe devices unbound.

Figure 4:

Safety and Security

Measuring DMA access times

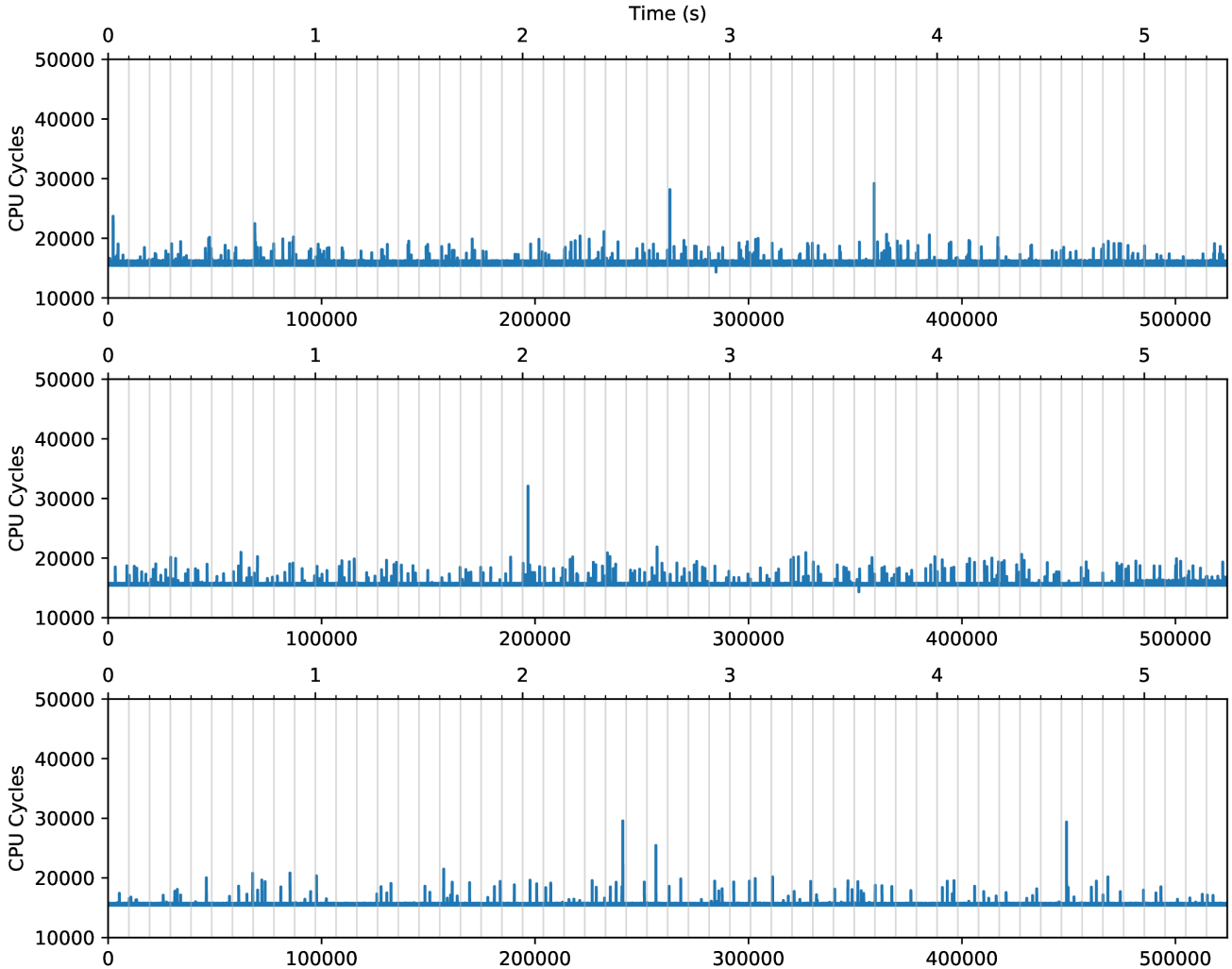


Figure 3: CPU cycles per batch of transmitted packets with IOMMU and fixed virtual and physical addresses, other PCIe devices unbound, ping on second NIC port every 0.4 seconds.

Measuring DMA access times

Results:

- Measurements contain too much noise to detect any IOTLB misses
- Source for strongly varying number of CPU cycles for packet transmission is unclear
- Ideas on this topic very welcome

Conclusion

- In most cases IOMMUs do not have a significant effect on throughput and latency
- However, in some cases throughput is strongly affected (decreases of more than 50%!)
- Profileration of IOMMUs and support by OSes has increased sharply
- However, IOMMU protection against malicious internal hardware is still weak

Contributions

ixy.rs:

- Driver for virtual functions (ixgbevf) to support SR-IOV
- Support for legacy Intel IOMMUs with IOVAs smaller than the host's virtual address widths
- Support for multiple devices associated to the same IOMMU group
- Bruteforce allocator to allocate physically and virtually contiguous memory on 4 KiB pages
- A tool to perform timing measurements on the NIC's DMA operations

DPDK:

- Fixed an off-by-one-error in the ixgbe initialization code