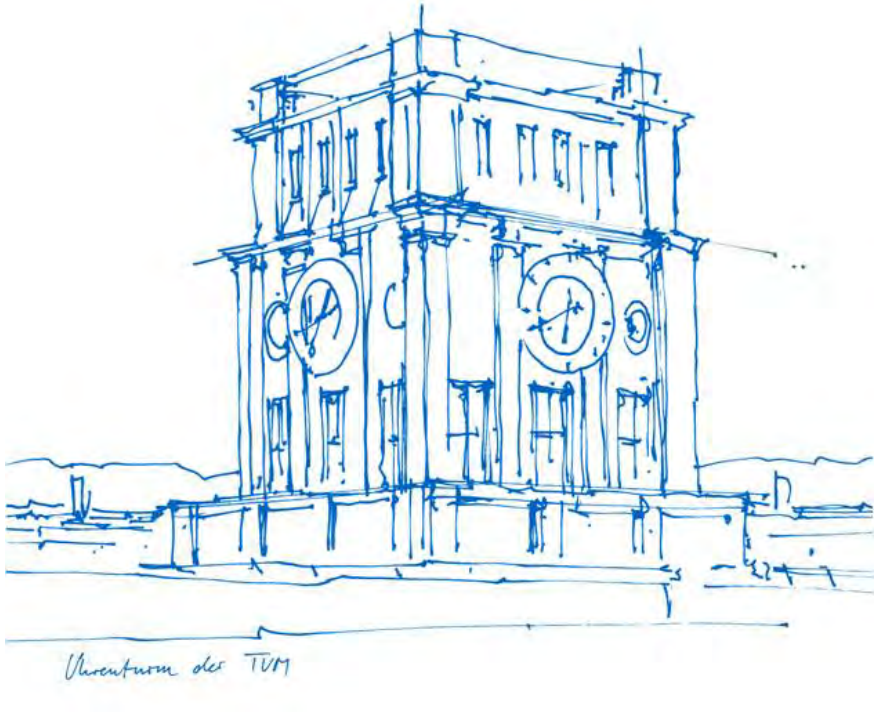
Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Low-latency Networks Slicing with Programmable Network Elements

Georg Carle

carle@net.in.tum.de
http://www.net.in.tum.de/~carle

Acknowledgements:
Sebastian Gallenmüller, Eric Hauser,
Manuel Simon, Max Helm,
Benedikt Jaeger, Florian Wiedner,
and all further members of the Chair of
Network Architectures and Services



Programmable, Automated Communication Networks

- Programmable networks, virtualization, slicing
- Multiple virtual networks operated on same physical infrastructure
- Requirement: virtualized networks not disturbing each other
  - performance isolation
  - performance guarantees
    (e.g. Ultra-Reliable Low Latency URLLC)

- Which aspects of programmability affect performance,
  in particular performance guarantees?
  - Processor architecture
  - Operating system influence
  - Hardware target architecture

Outline

Need

- Predictable network services using programmable network devices

Challenges

Approach

- Reproducible measurements
- Investigating bottlenecks of programmable network devices
- Investigating OS and system configuration influences
- Investigating slicing with P4 software and hardware targets
- Latency tail modelling, Extreme Value Theory (EVT)

Conclusions

Need for Resilient Low-Latency Predictable Network Services

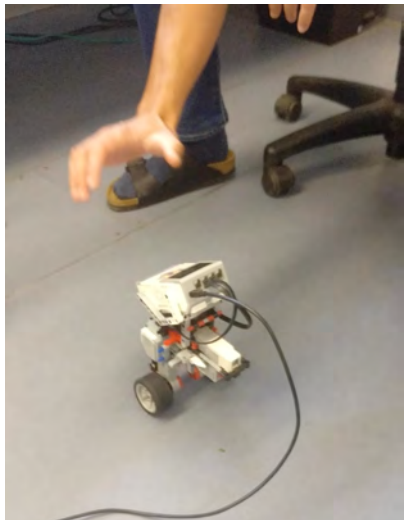
Challenges

- complex architectures
- performance, safety and security requirements

⇨ Need for

- Secure communication, trustworthy implementation
- Network stack + applications: *worst case performance guarantees*
- Scalability, flexibility, affordability, time-to-market

Low-Latency Systems:



Network-Controlled Robot

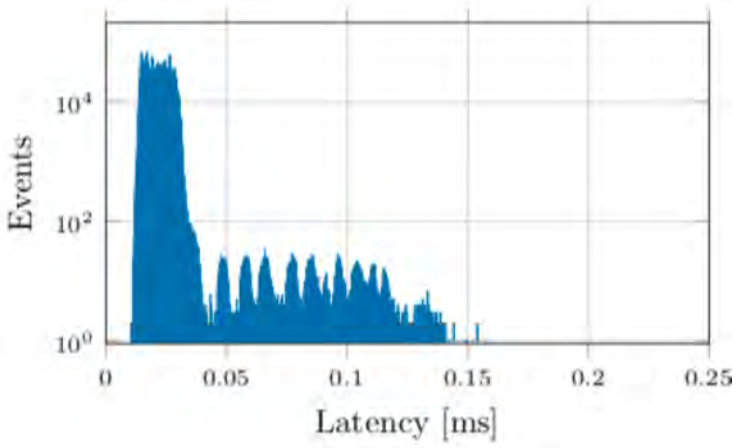
Power Grid Control

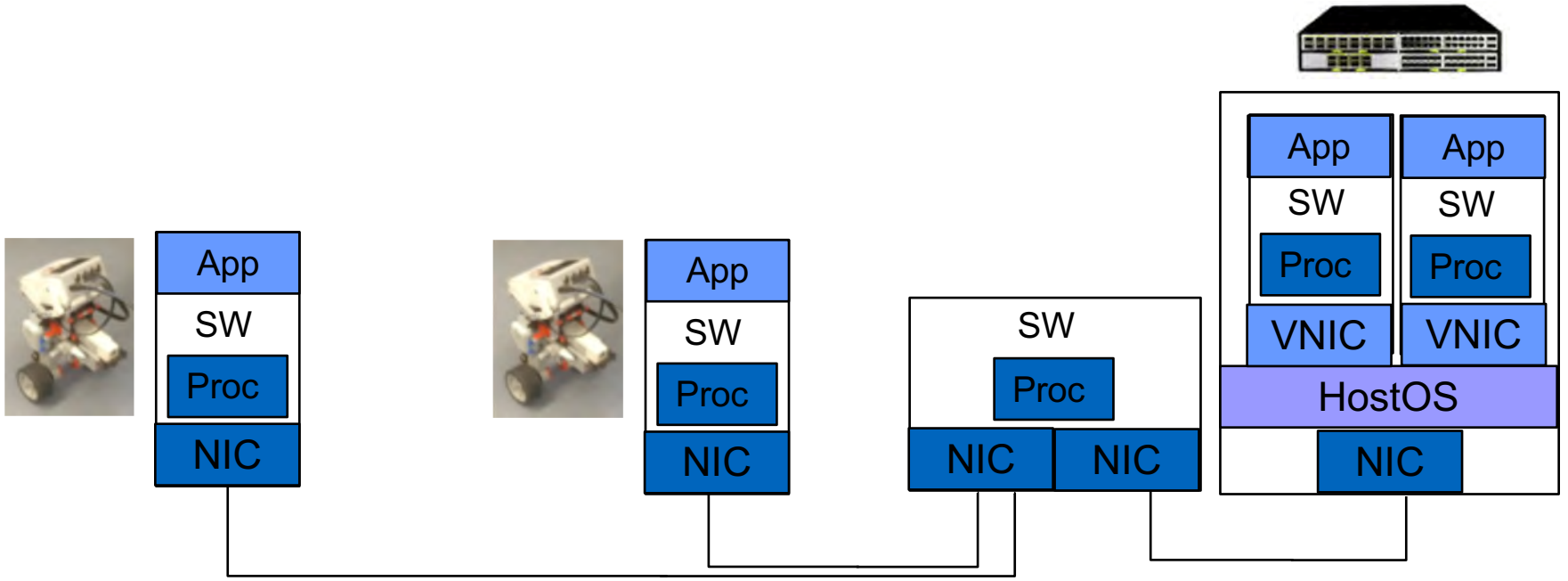
Need: End-to-End Worst-Case Latency Guarantees

Challenges:

Performance of complex networked systems

- Hardware + Software
- Issue: latency distribution (long tail)



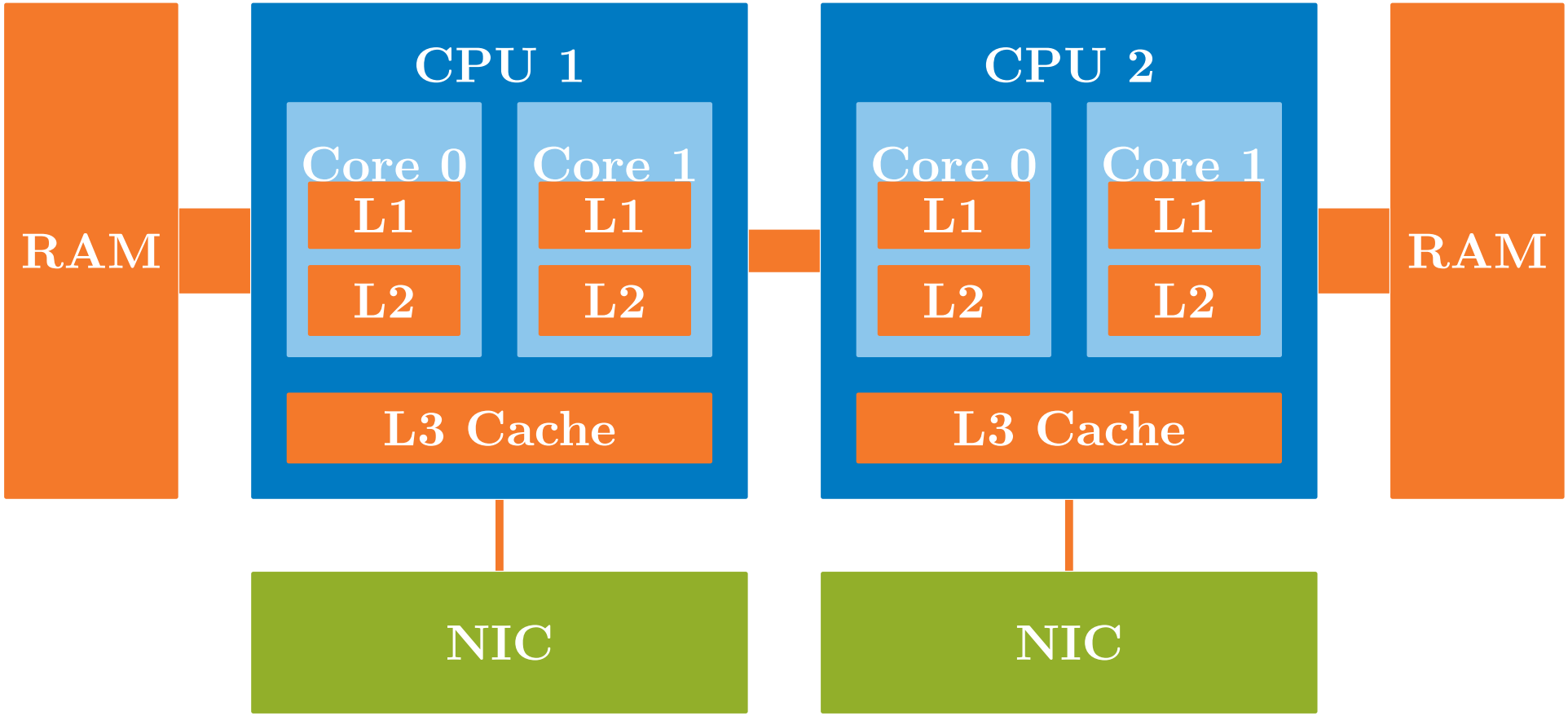


Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

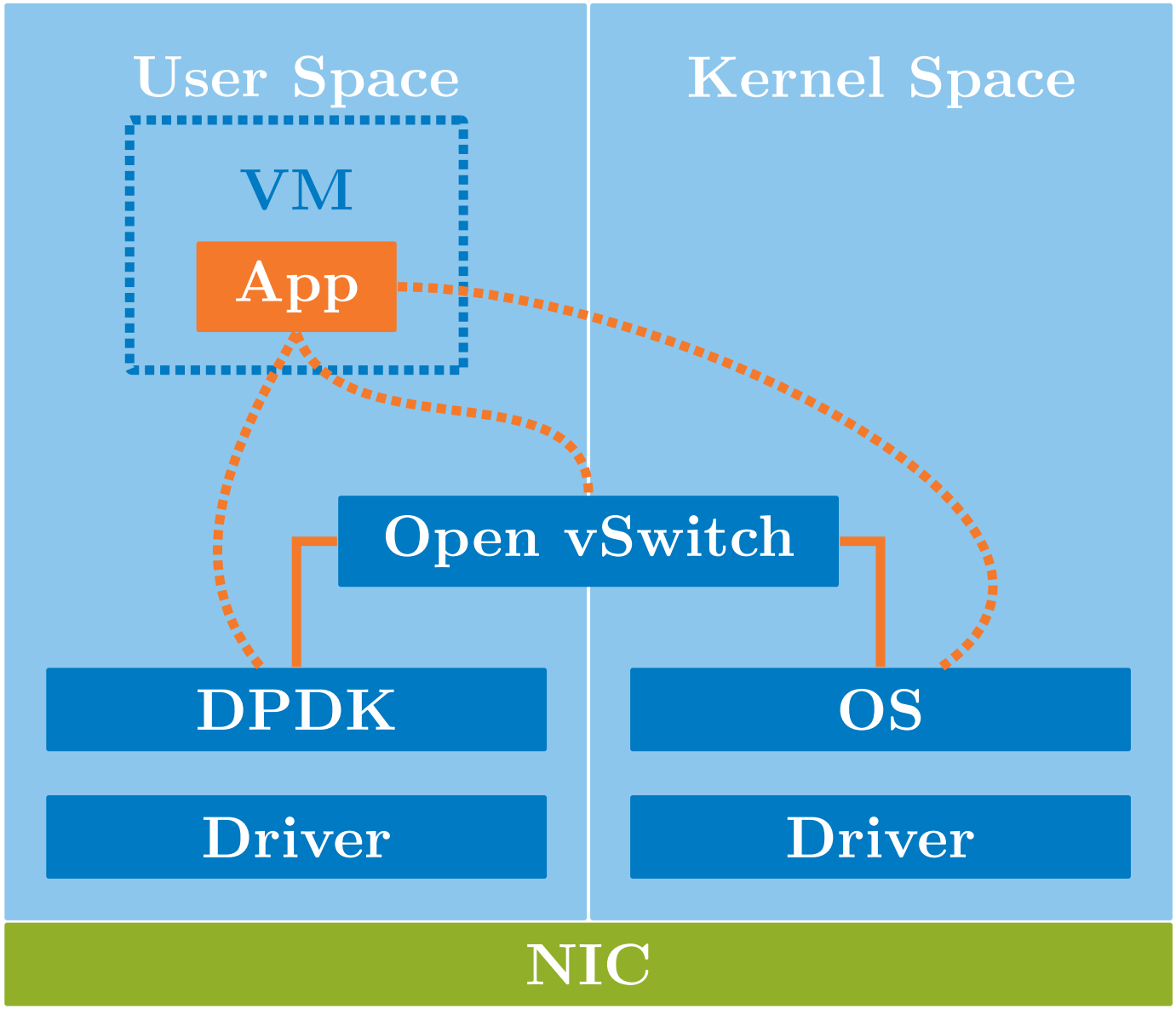
Processing Architecture Complexity

Modern Hardware Architectures are Complex

Non-Uniform Memory Architecture (NUMA)



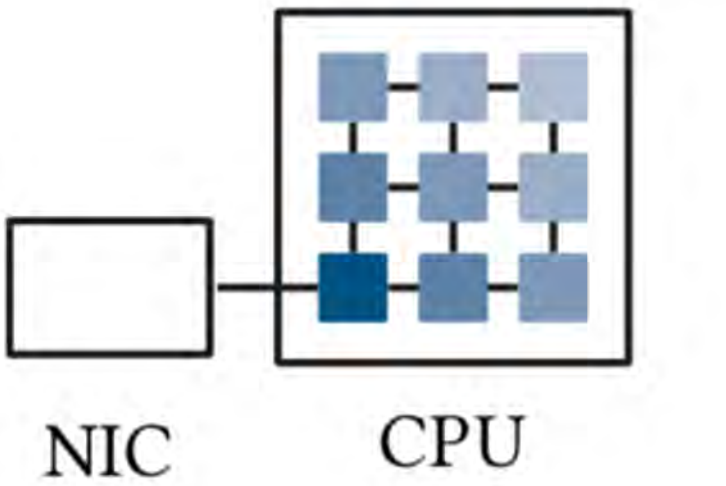
Modern Software Architectures are Complex



Manycore Architectures add Complexity

Protocol stack processing with manycore processors

- Example: Tilera Tile-Gx36,
  as used in Mikrotik Routers



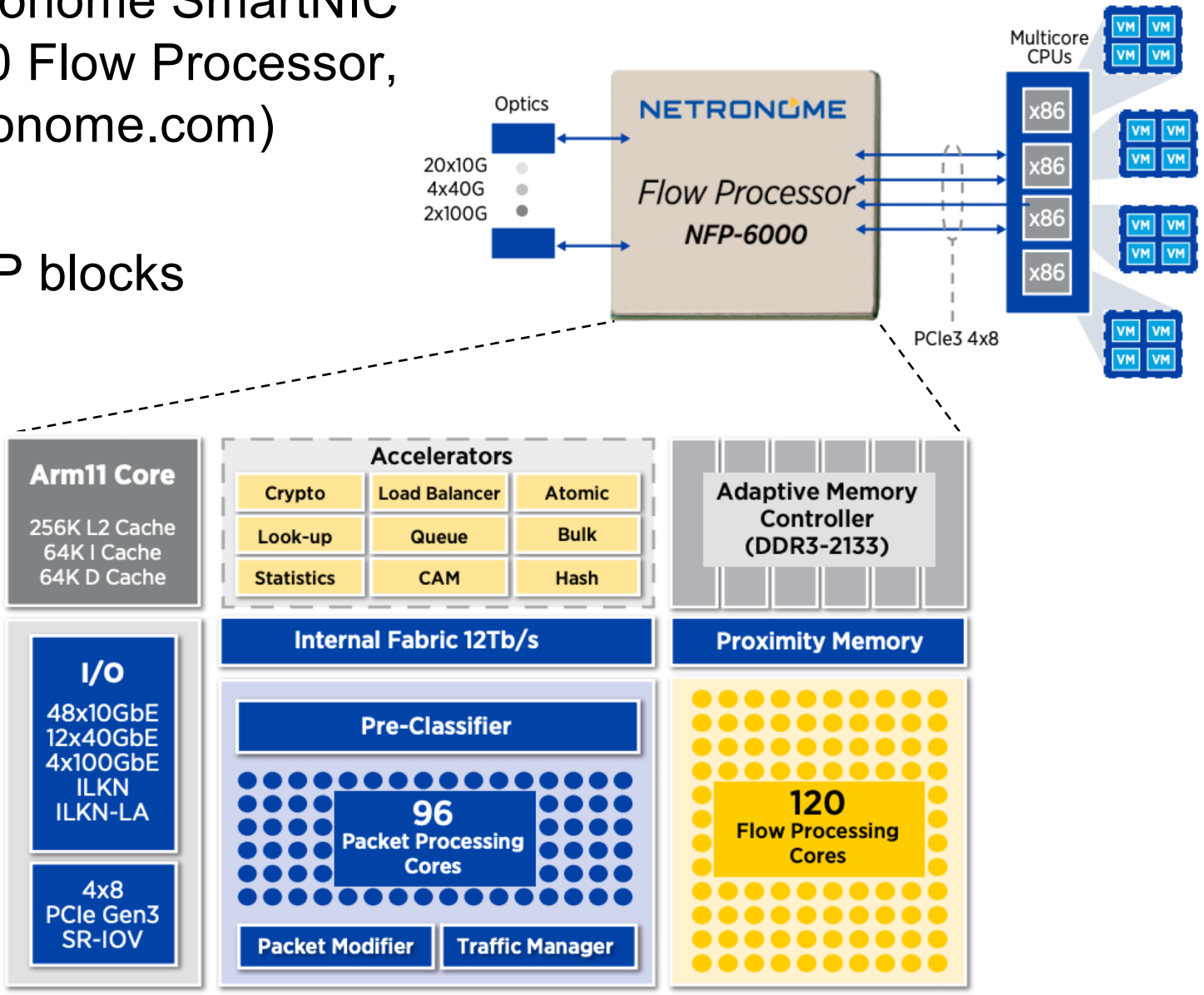
- Questions
  - Impact of architecture on performance?
  - Which experiments yield data of appropriate coverage?

Programmable NICs add Complexity

Programmable packet processing architectures
Example: Netronome SmartNIC
with NFP-6000 Flow Processor,
(c.f. www.netronome.com)

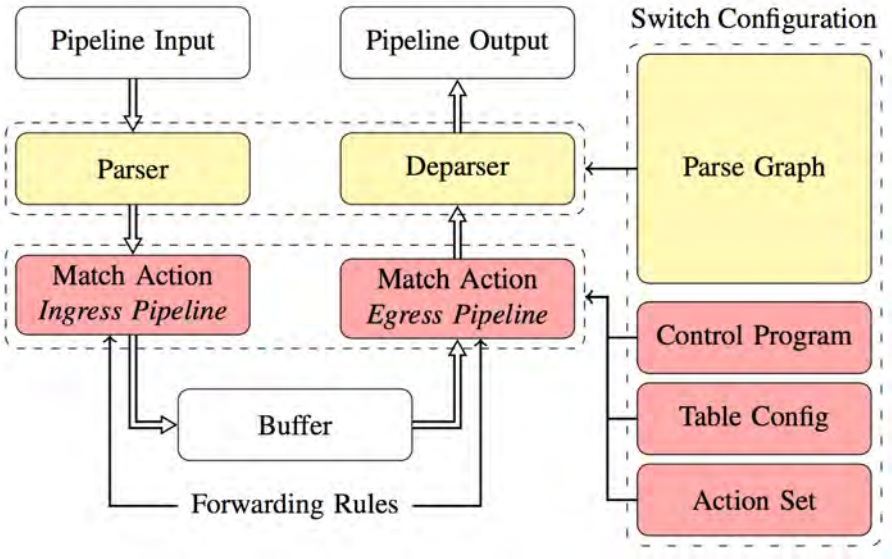
NICs

Composable IP blocks

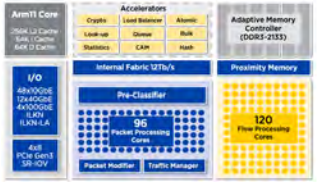


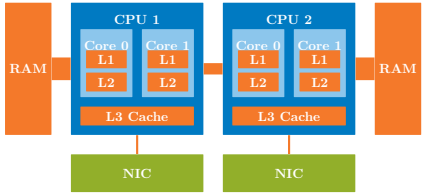
Programmability adds Complexity

P4 Architecture



P4 on different processing targets

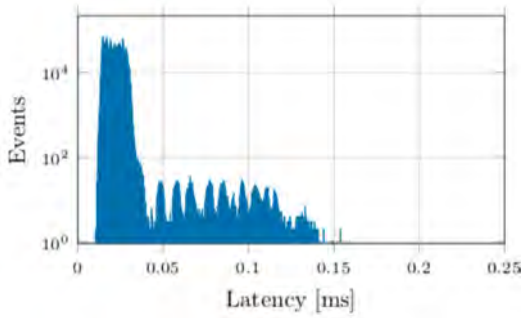




- Tofino switch
- P4NetFPGA
- P4 Programming of Netronome NFP
- PISCES based on Open vSwitch with DPDK
- t4p4s DPDK

Challenge: Providing Latency Guarantees

Networked system with programmable network components:



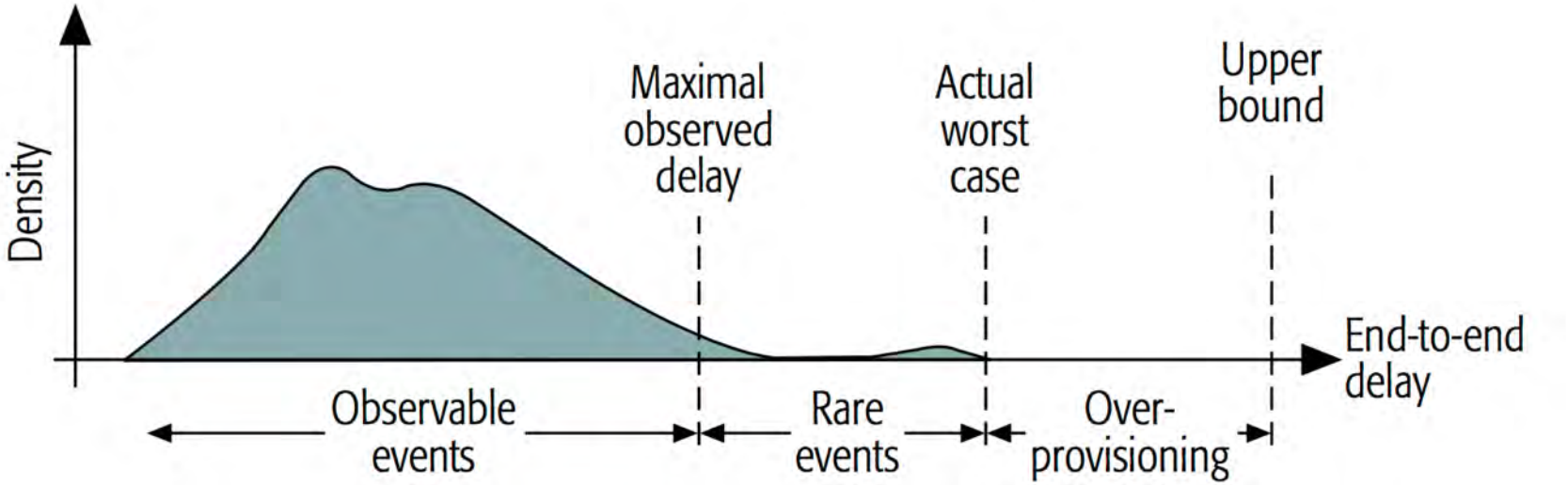
App
SW
Proc
NIC

App
SW
Proc
NIC

SW
Proc
NIC
NIC

App
App
SW
SW
Proc
Proc
VNIC
VNIC
HostOS
NIC

Maximal observed delay vs. upper bound:



Fabien Geyer, Georg Carle: Network engineering for real-time networks: Comparison of automotive and aeronautic industries approaches, IEEE Communications Magazine 54 (2), 2016

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

Reproducible Measurements

Experiments that yield the relevant data

Challenge: Reproducible Measurements

ACM SIGCOMM MoMeTools - Workshop on Models, Methods and Tools for Reproducible Network Research
Georg Carle, Hartmut Ritter, Klaus Wehrle,
Karlsruhe, Germany, August 2003

ACM SIGCOMM Reproducibility Workshop
Olivier Bonaventure, Luigi Iannone, Damien Saucez
Los Angeles, USA, August 2017

[Rep17] Q. Scheitle, M. Wählisch, O. Gasser, T. Schmidt, G. Carle,
Towards an ecosystem for reproducible research in computer networking
Proceedings of the ACM SIGCOMM Reproducibility Workshop, 2017

Dagstuhl seminar 18412 "Encouraging Reproducibility in Scientific Research of the Internet", October 2018

ACM Badges by Artifact Evalulation Committees

- Best solution so far
- Problems
  - High effort
  - Potentially low robustness

Hardware Traffic Generators

- Fast
- Precise

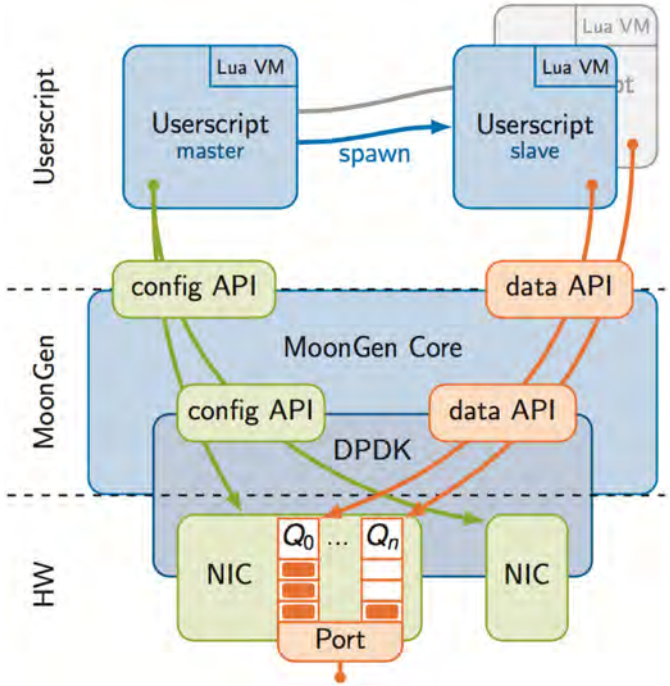
but

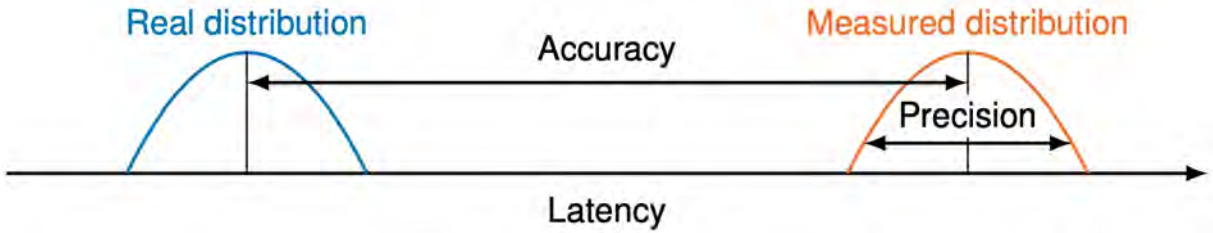
- Expensive
- Difficult to deploy
- Not flexible

Spirent traffic generator

MoonGen

- Inexpensive: Commecial Off-The-Shelf hardware
- Fast: DPDK for packet I/O, multi-core support
- Easy to deploy: simple software setup
- Flexible: user-controlled Lua scripts
- Accurate and precise
  - Timestamping: Utilize hardware features found on modern commodity NICs
  - Rate control: Hardware features and novel software approach





[IMC15] Paul Emmerich, Sebastian Gallenmüller, Daniel Raumer, Florian Wohlfart, Georg Carle: MoonGen: A Scriptable High-Speed Packet Generator, ACM Internet Measurement Conference (IMC 2015), Tokyo, Japan, October 2015

[ANRP17] Internet Research Task Force (IRTF) Applied Networking Research Prize, IETF-100, Nov. 2017, https://irtf.org/anrp

[ANCS17] Paul Emmerich, Sebastian Gallenmüller, Gianni Antichi, Andrew Moore, Georg Carle: Mind the Gap – A Comparison of Software Packet Generators, ACM/IEEE Symposium on Architectures for Networking and Communications Systems 2017

Rate Control Precision and Accuracy

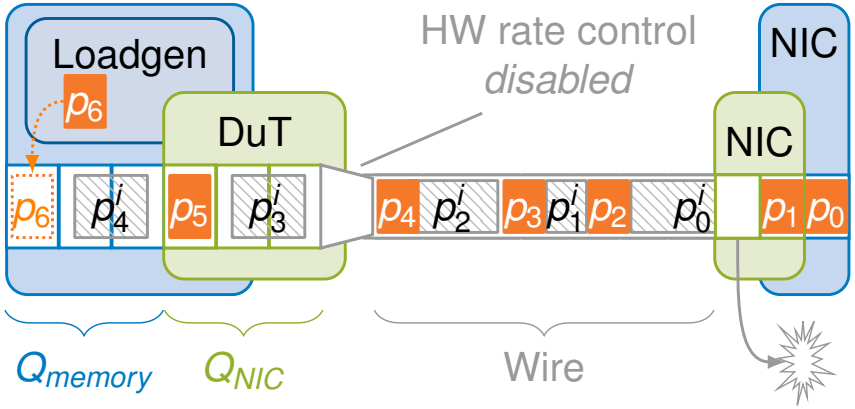
Intel NIC

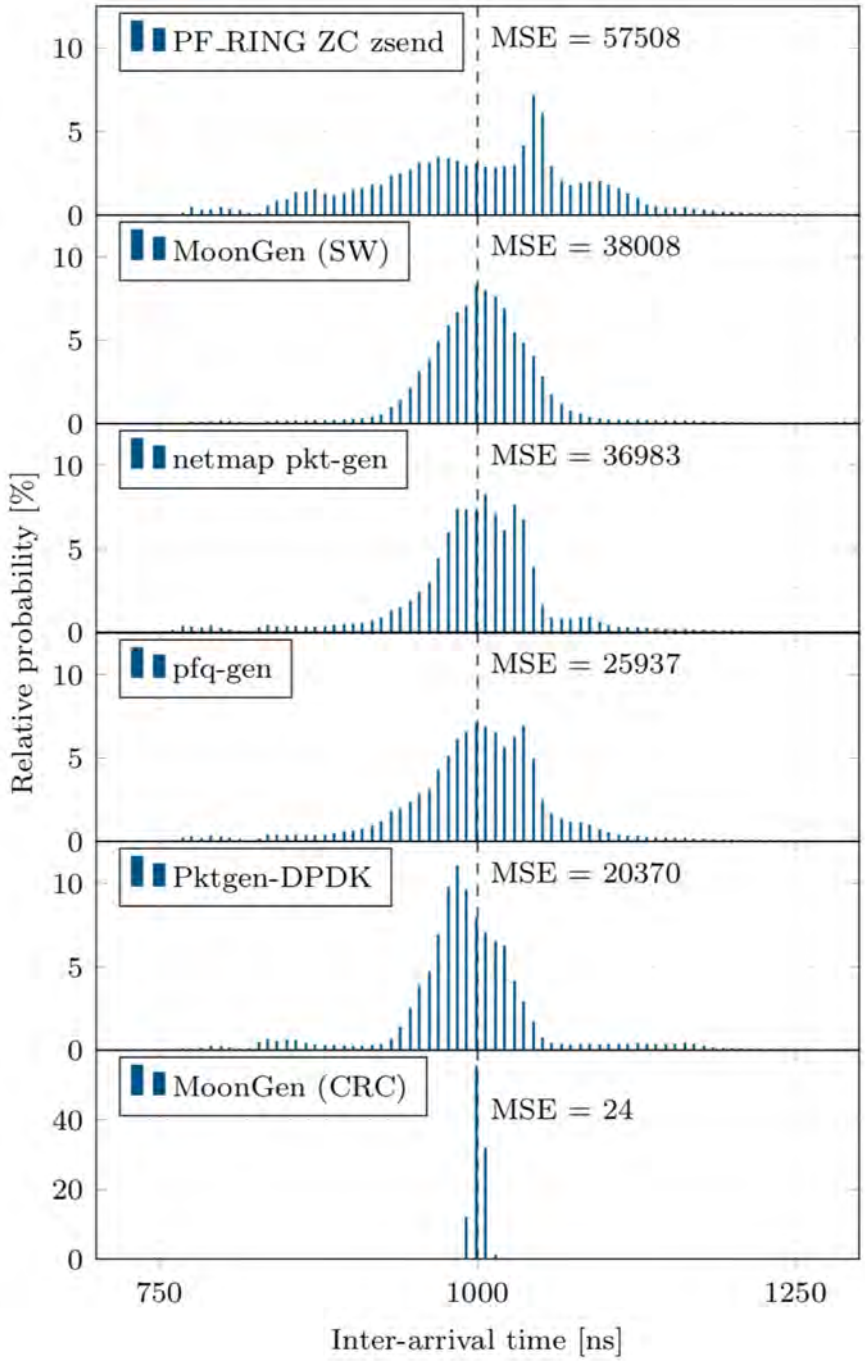
MoonGen

NetFPGA SUME

OSNT

- Corrupted CRC approach: high performance, high accuracy, and high precision





[ANCS17] P. Emmerich, S. Gallenmüller, G. Antichi, A. Moore, G. Carle, “Mind the Gap – A Comparison of Software Packet Generators”. ACM/IEEE Symposium on Architectures for Networking and Communications Systems 2017

Usage of MoonGen/libmoon

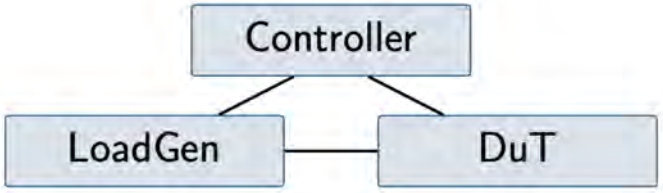
Name	Usage scenario	Publication
High-performance applications:		
FlowScope	Tool for high-performance flow capture and analysis	[11], [12]
MoonRoute	Extensible high-performance router	[4], [13]
Benchmarking tools:		
RFC 2544	Modular benchmarking tool	[14], [15]
OPNFV VSPERF	Automated NFV testing framework	[16], [17]
FLOWer	High-performance switch benchmarking	[18], [19]
Traffic & packet generation:		
NFVnice	Throughput and latency measurements	[20]
Verified NAT	Throughput and latency measurements	[21]
PISCES	Throughput measurements	[22], [23]
Sonata	Replaying CAIDA traces	[24]
DoS flood generator	DNS and TCP SYN flooding attack tools	[25]–[27]
MoonGen / libmoon under test:		
MoonGen investigation	Precise and accurate rate control and timestamping	[3], [28], [29]
MoonGen timestamping	Investigation of timestamping for packet generators	[30]
Additions to MoonGen / libmoon:		
MoonStack	Easy-to-use and efficient packet creation	[31]

[Comsnets18] Gallenmüller, Scholz, Wohlfart, Scheitle, Emmerich, Carle, “High-Performance Packet Processing and Measurements,” COMSNETS 2018, Bangalore, India, Jan. 2018

TUM Baltikum Testbed for Reproducible Experiments

- Automated workflow using **pos**
  plain orchestrating service (pos)
  workflow for reproducible experiments
- Throughput - packets per second,
  bytes per second, frame loss rate
- Latency - Median, average, worst
  case, percentiles, ...
- White-box - Hardware and software
  events; Cycles, Interrupts, L1/L2/L3
  cache misses; Granularity: per
  second, per packet, per function

[SLICES] ESFRI - European Strategy Forum on Research Infrastructures; TUM Baltikum Testbed: part of Scientific Large-scale Infrastructures https://slices-sc.eu/



[pos] Sebastian Gallenmüller, Dominik Scholz, Henning Stubbe, Georg Carle, “The pos Framework: A Methodology and Toolchain for Reproducible Network Experiments,” in The 17th International Conference on emerging Networking EXperiments and Technologies (CoNEXT ’21), Munich, Germany, Dec. 2021

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

Investigating Different Bottlenecks

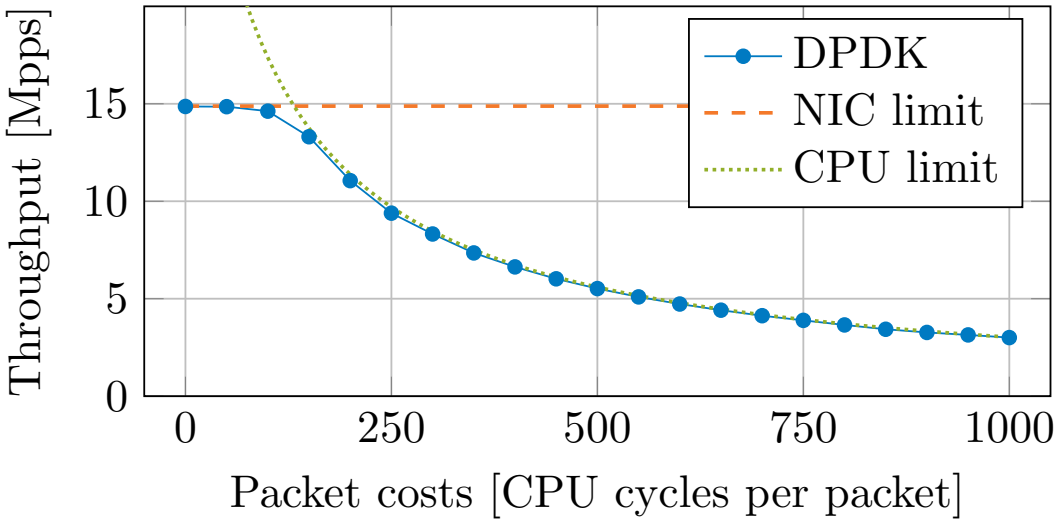
Performance Evaluation: Node Bottlenecks

Hardware

- Network Bandwidth
- NIC Processing Capacity
- PCIe Bandwidth
- Memory Bandwidth
- CPU Cache Size
- CPU Cache Line Length

Software

- CPU utilization per packet
- Kernel / network stack overhead

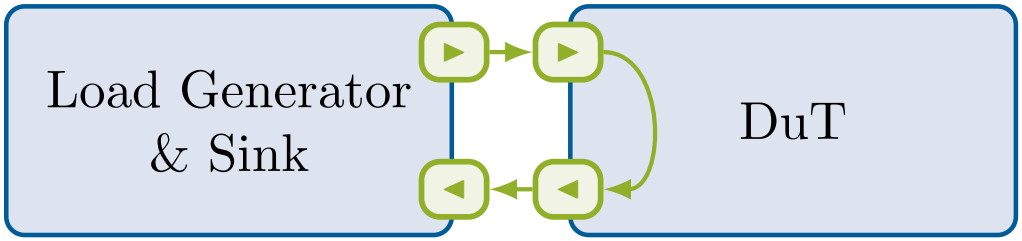


Throughput limit = min(NIC limit, CPU limit)
NIC limit = 14.88 Mpps (10 Gb Ethernet)
CPU limit = available CPU cycles

[ANCS15] S. Gallenmüller, P. Emmerich, F. Wohlfart, D. Raumer, G. Carle *"Comparison of Frameworks for High-Performance Packet IO"*. In ACM/IEEE Symposium on Architectures for Networking and Communications Systems, May 2015.

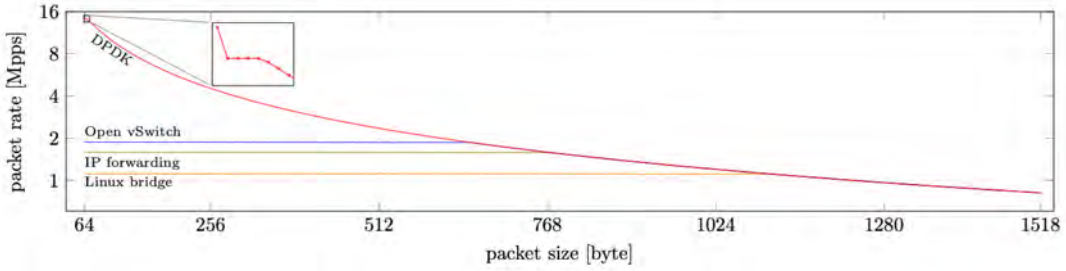
System Analysis

Measurement setup



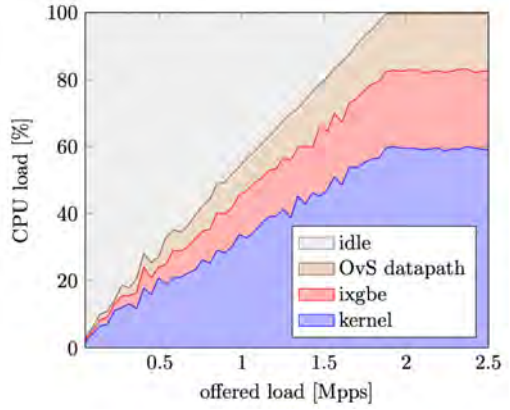
Black-box

- Throughput
  - Packets per second, bytes per second
  - Frame loss rate
  - Back-to-Back frame burst size
- Latency
  - Median, average, worst case, percentiles, ...



White-box

- Hardware and software events
  - Cycles, Interrupts, L1/L2/L3 cache misses
  - Per second, per packet, per function

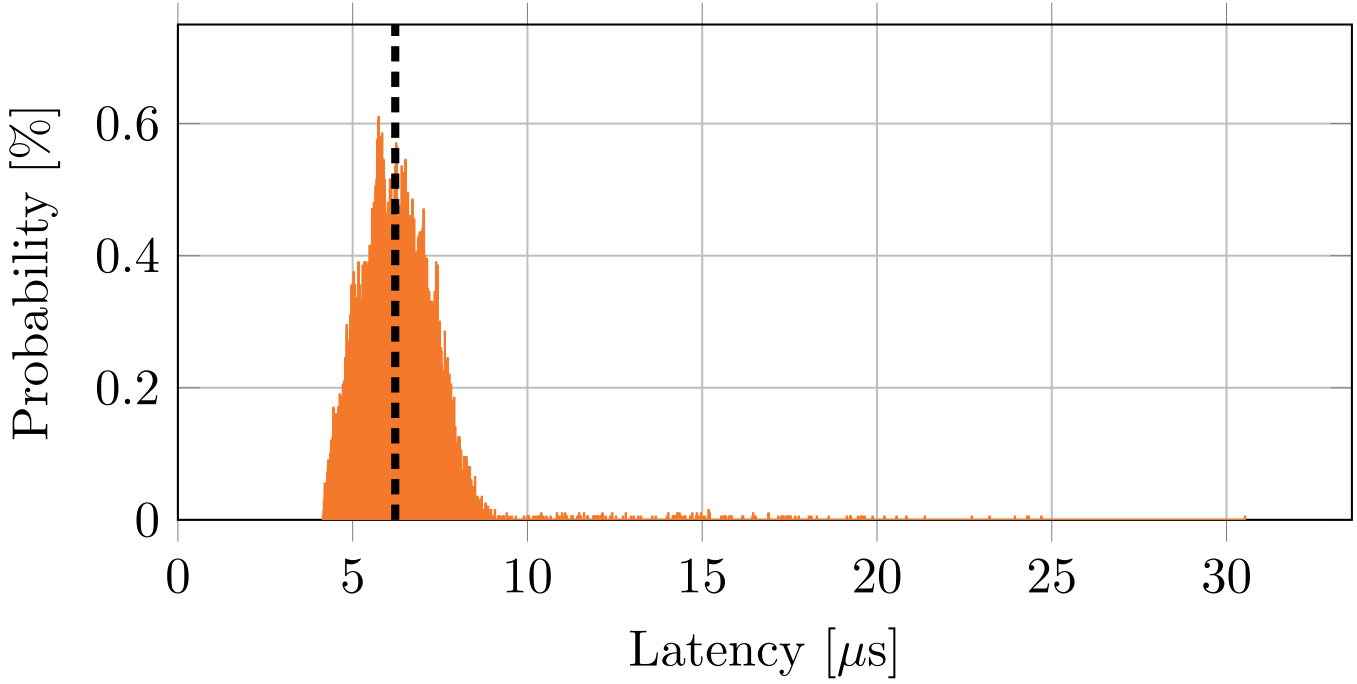


Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Impact of Processing Architectures

Latency Measurements: FreeBSD Software Router

FreeBSD router, forwarding 64-byte packets

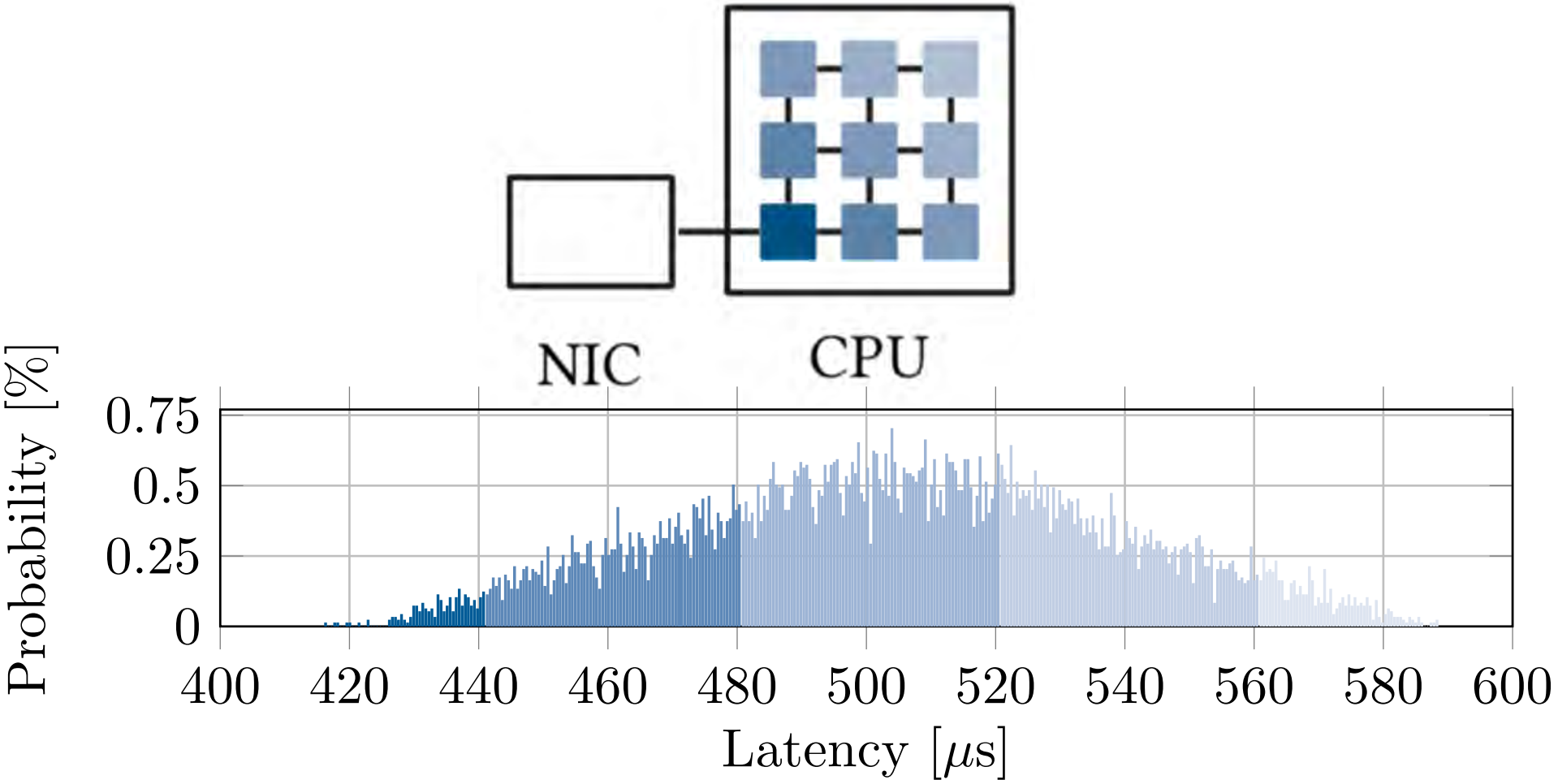


- Average does not reflect **long tail** distribution

[ANRW16] Daniel Raumer, Sebastian Gallenmüller, Florian Wohlfart, Paul Emmerich, Patrick Werneck, Georg Carle: Revisiting benchmarking methodology for interconnect devices. In Applied Networking Research Workshop 2016, Jul. 2016

Explanation of latency distribution by processing paths

Mikrotik Cloud Core Router CCR1036-8G-2S+, Tilera Tile-Gx36



[ANRW16] Daniel Raumer, Sebastian Gallenmüller, Florian Wohlfart, Paul Emmerich, Patrick Werneck, Georg Carle: Revisiting benchmarking methodology for interconnect devices. Applied Networking Research Workshop 2016, Jul. 2016

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

Challenge:
Low Latency Software Stack

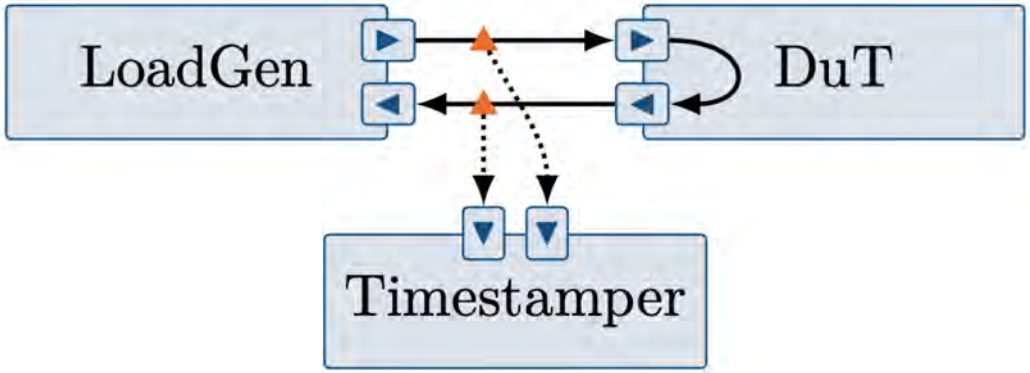
5G Low Latency Services

5G Ultra-Reliable Low-Latency Communication (URLLC)

- Ultra reliable: 99.999% packet delivery probability
- Low latency: 1ms one-way latency in Radio Access Network (RAN)

5G Service provisioning with Virtual Network Functions (VNF)

- Virtualized environment: Linux, kvm
- Network function: Snort3

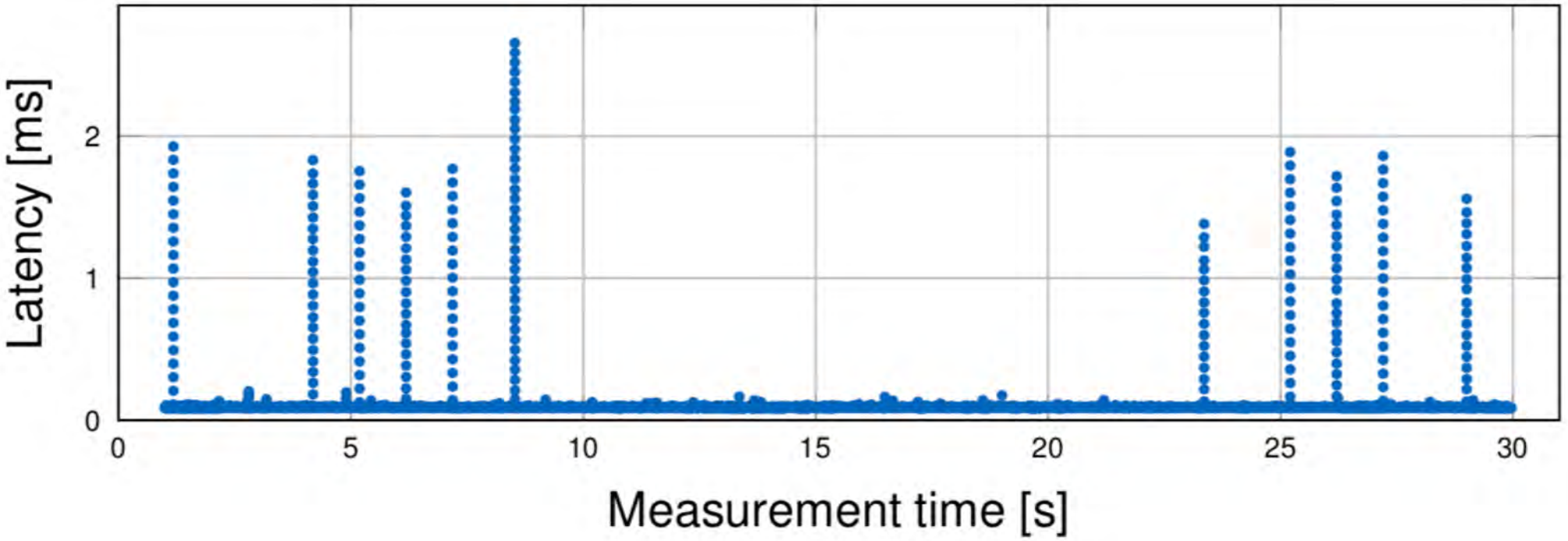


percentiles	50th	99th	99.9th	99.99th	99.999th
Snort 3 forwarder	69 µs	88 µs	107 µs	1.7 ms	2.5 ms

⇨ 99.99th percentile already violates URLLC

Latency of Virtual Network Functions

Snort 3 forwarding



[NOMS2020] Sebastian Gallenmüller, Johannes Naab, Iris Adam, Georg Carle:
5G QoS: Impact of Security Functions on Latency,
IEEE/IFIP NOMS, April 2020

[ComMag2020] Sebastian Gallenmüller, Johannes Naab, Iris Adam, Georg Carle
5G URLLC: A Case Study on Low-Latency Intrusion Prevention
IEEE Communications Magazine 58 (10), 35-41

Analysing Virtual Network Function Performance

Reasons for VNF Performance Impairment

- Interupt-based IO
  - Linux NAPI
- CPU features
  - Dynamic scheduling of processes onto CPU cores
  - Virtual cores (SMT/Hyperthreading)
  - Energy-saving mechanisms
  - Dynamic cache allocation
- Expensive VM IO

Fixing VNF Performance

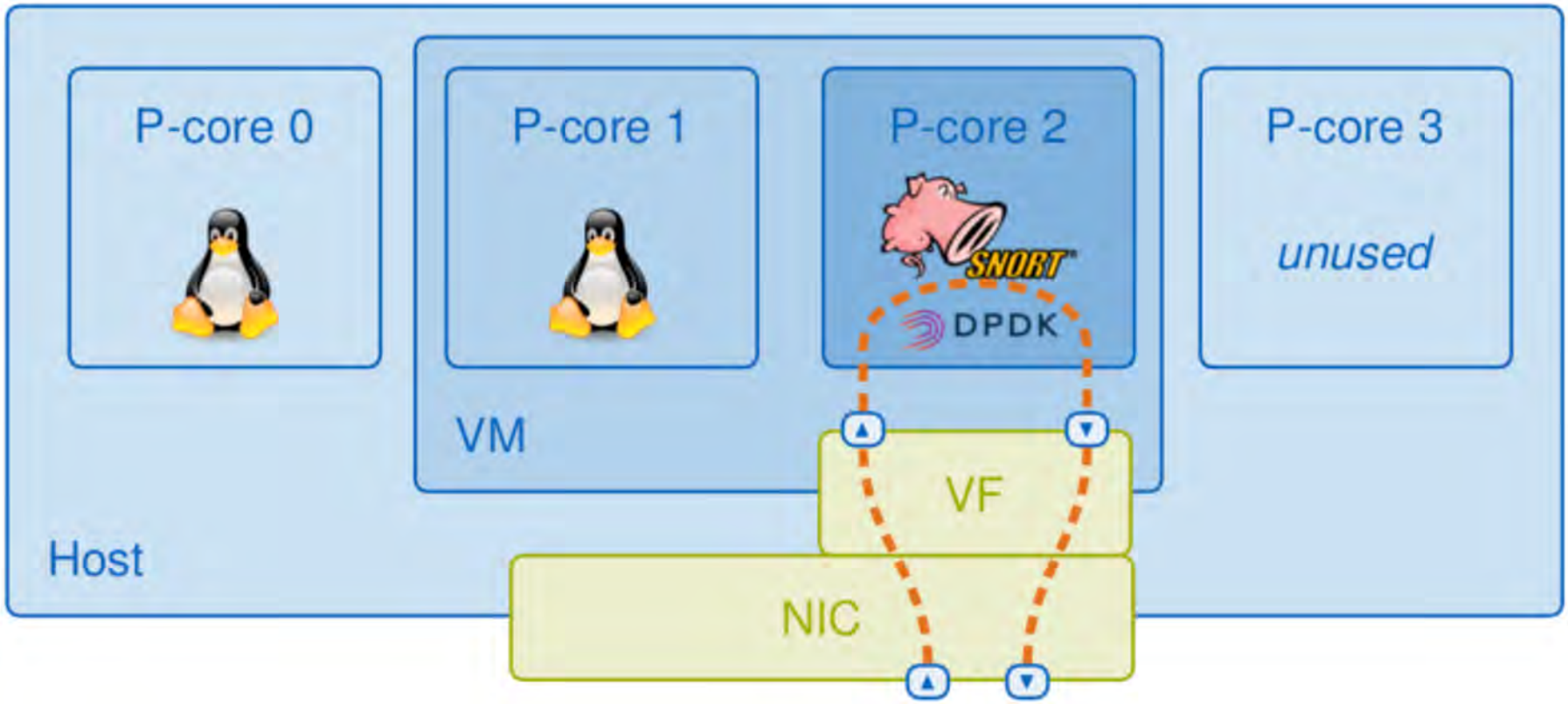
- Polling-based IO
  - DPDK
- CPU features
  - Statically allocate CPU cores for processes
  - Disable SMT/Hyperthreading
  - Disable energy-saving mechanisms
  - Static cache allocation (Intel CAT)
- NIC acceleration (SR-IOV)

[NOMS2020] Sebastian Gallenmüller, Johannes Naab, Iris Adam, Georg Carle:
5G QoS: Impact of Security Functions on Latency,
IEEE/IFIP NOMS, April 2020

[ComMag2020] Sebastian Gallenmüller, Johannes Naab, Iris Adam, Georg Carle
5G URLLC: A Case Study on Low-Latency Intrusion Prevention
IEEE Communications Magazine 58 (10), 35-41

Low-latency Software Stack: Architecture

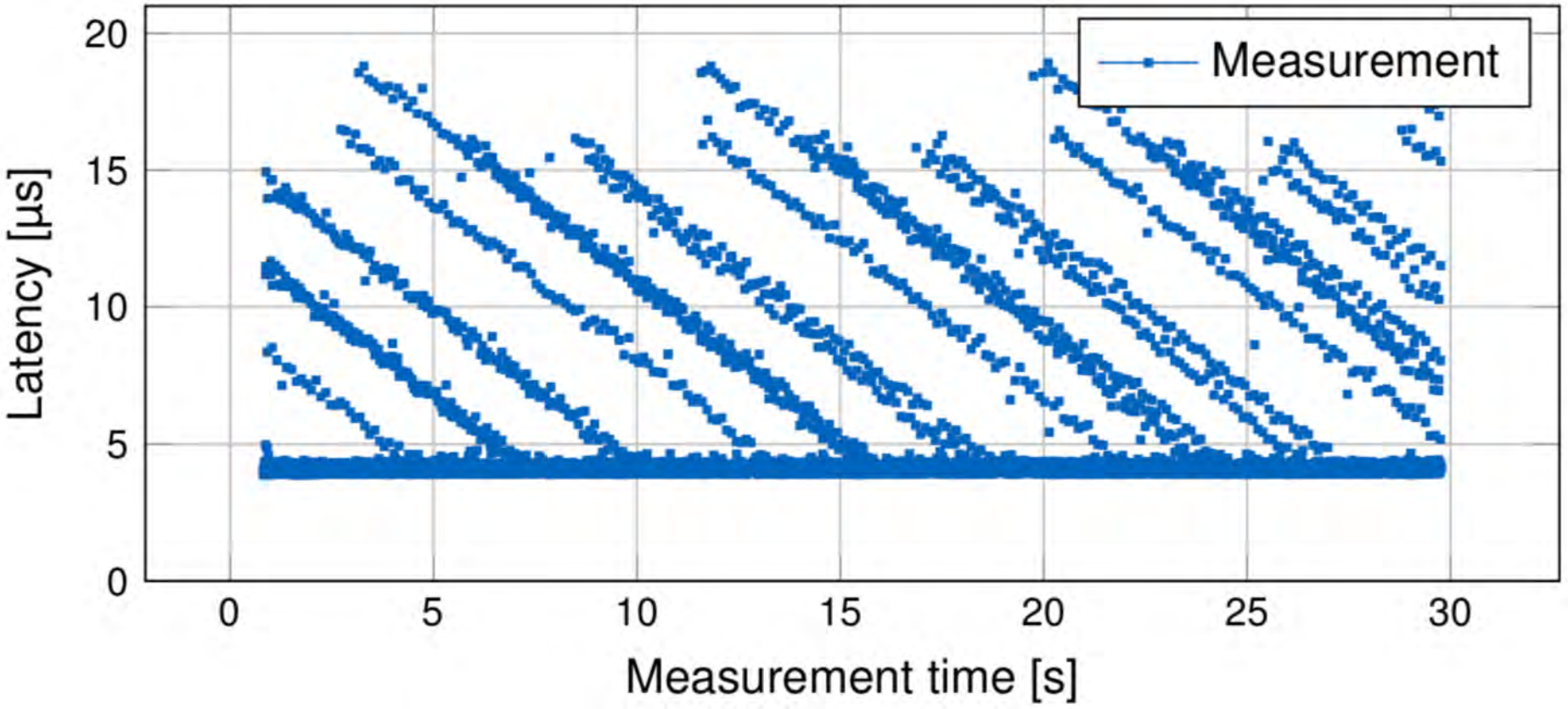
Setup



- Static pinning: HostOS⇨P-core0; VMOS⇨P-core 1, App⇨P-core2
- P-core2 isolated from scheduling of HostOS & VM OS
- SR-IOV splits NIC into Virtual Functions, binding VF to P-core2

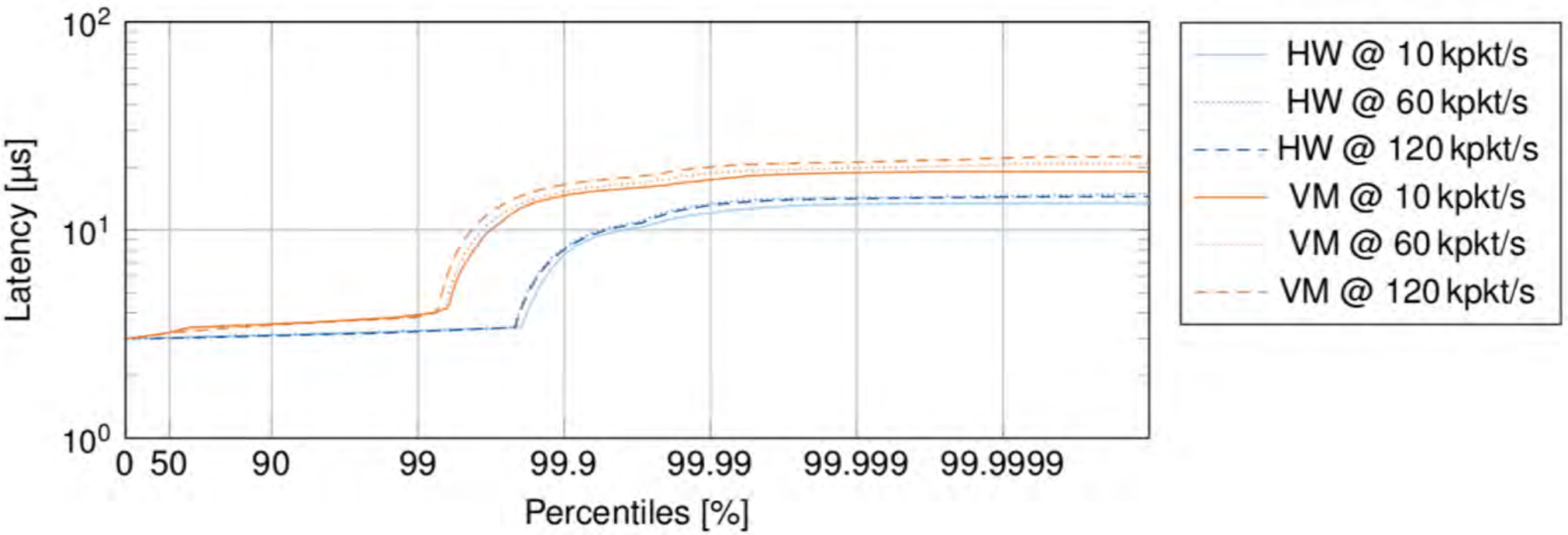
DPDK L2 Forwarding

Influence of interrupts



- 10 kpacket/s: 100 us interarrival time
- Instrumentation reveals two interrupts, rate 250 Hz and 125 Hz
  - local timer interrupts (loc): 5.5us; IRQ work interrupts (iwi): 8.2 us
- Pattern due to interrupt duration < packet interarrival time

DPDK L2 Forwarding



- High Dynamic Range (HDR) Histogram
- DPDK L2 forwarding as baseline
- HW: no virtualisation
- VM: kvm virtualisation
- Maximum latency: ~ 20 µs

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

Challenge:
Understanding Performance of P4 processing on different targets

Comparison of P4 Target Types

	CPU	NPU	FPGA	ASIC
Throughput	+	++	+++	++++
Latency	> 10 µs	5 µs to 10 µs	< 2 µs	< 2 µs
Jitter	----	---	--	-
Resources	++++	+++	++	+
Flexibility	++++	+++	++	+
Example	t4p4s DPDK	NFP-4000 SmartNIC	NetFPGA SUME	Intel Tofino




[ITC2020] Dominik Scholz, Henning Stubbe, Sebastian Gallenmüller, Georg Carle, "Key Properties of Programmable Data Plane Targets," in 32nd International Teletraffic Congress (ITC 32), Osaka, Japan, Sep. 2020

Modelling P4 Programmable Network Devices

Methodology for modelling P4 programs executed on P4 targets

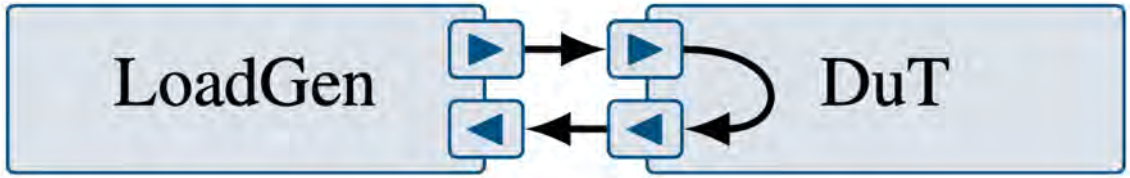
- Model P4 components individually
  - Parser
  - Match-action tables
    - Match type (exact, ternary, LPM – Longest Prefix Match)
    - Entry size (key, action, action data)
    - Number of entries
  - Packet modification
- Combine component models to model complete system

[ITC2020] Dominik Scholz, Henning Stubbe, Sebastian Gallenmüller, Georg Carle, “Key Properties of Programmable Data Plane Targets,” in 32nd International Teletraffic Congress (ITC 32), Osaka, Japan, Sep. 2020

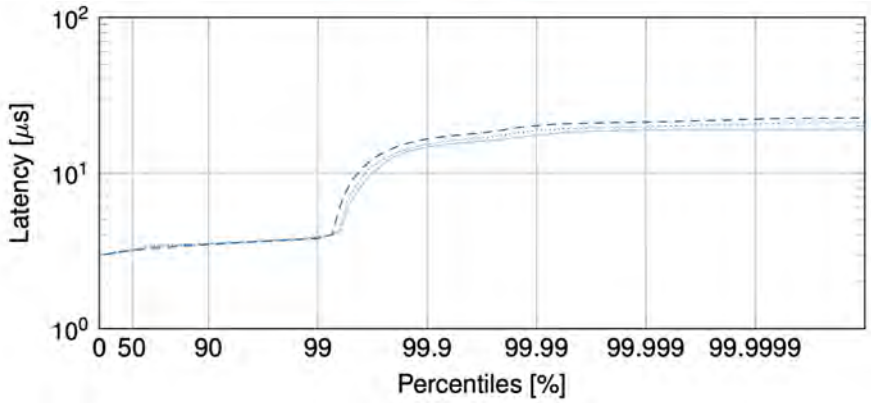
[ITC2021] Max Helm, Henning Stubbe, Dominik Scholz, Benedikt Jaeger, Sebastian Gallenmüller, Nemanja Deric, Endri Goshi, Hasanin Harkous, Zikai Zhou, Wolfgang Kellerer, Georg Carle, “Application of Network Calculus Models on Programmable Device Behavior,” in the 33rd International Teletraffic Congress (ITC 33), Avignon, France, Aug. 2021

Performance Modelling of Programmable Network Devices

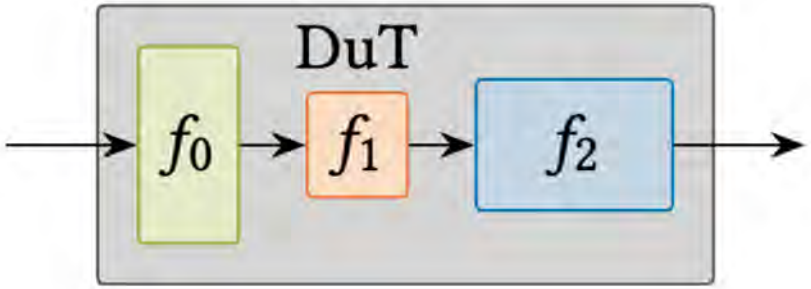
Experiment setup



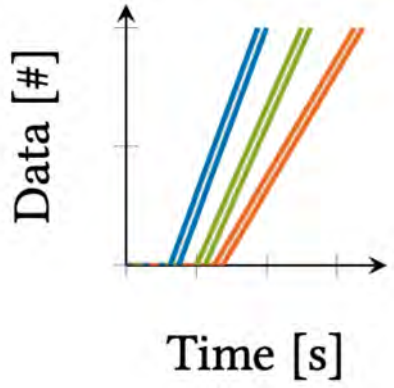
Measurement results



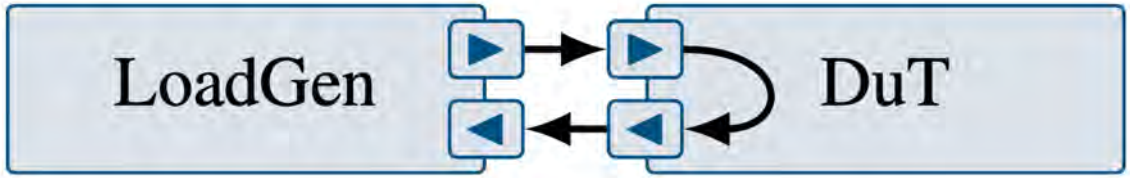
Device model



Service curve model



Experiment setup



t4p4s DPDK-based Software P4 Target

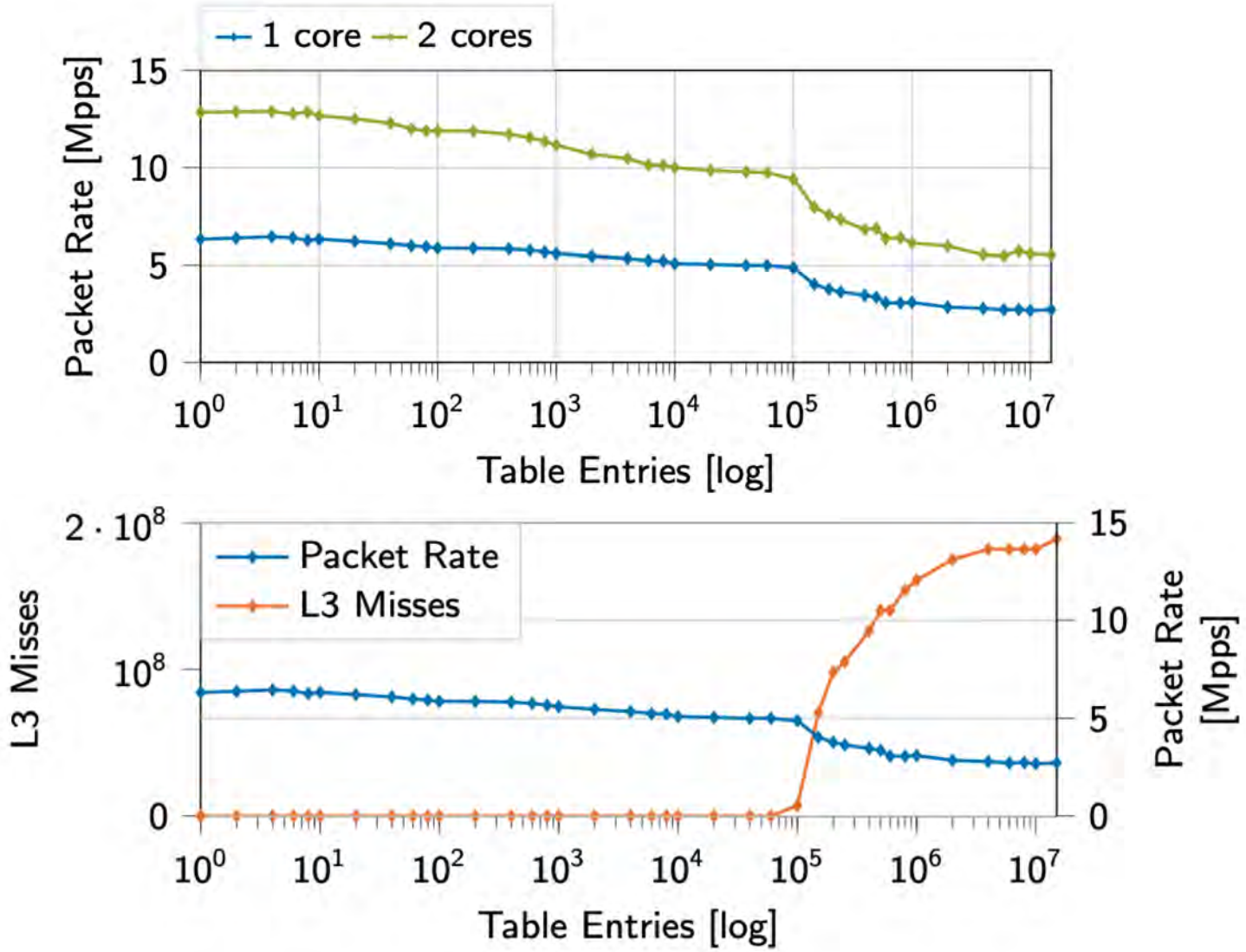
- P4 compiler, generates HW-independent C code
- Hardware-dependent library for e.g. DPDK

Device-under-Test Hardware

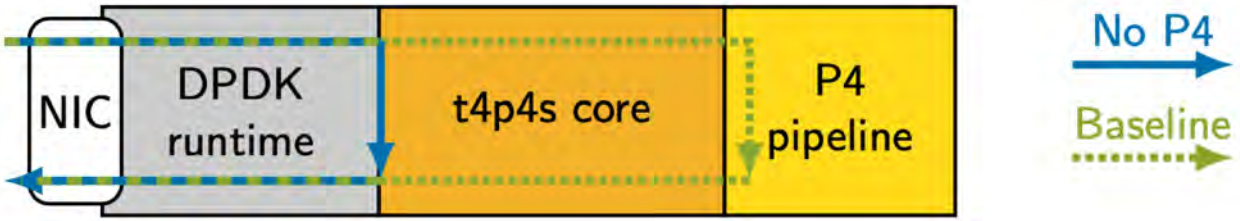
- Intel Xeon CPU E5-2640v2 (2.0GHz)
- Intel X540-AT2 NIC (dualport,10Gbit/s)
- Turbo boost and hyper threading disabled (reducing jitter)

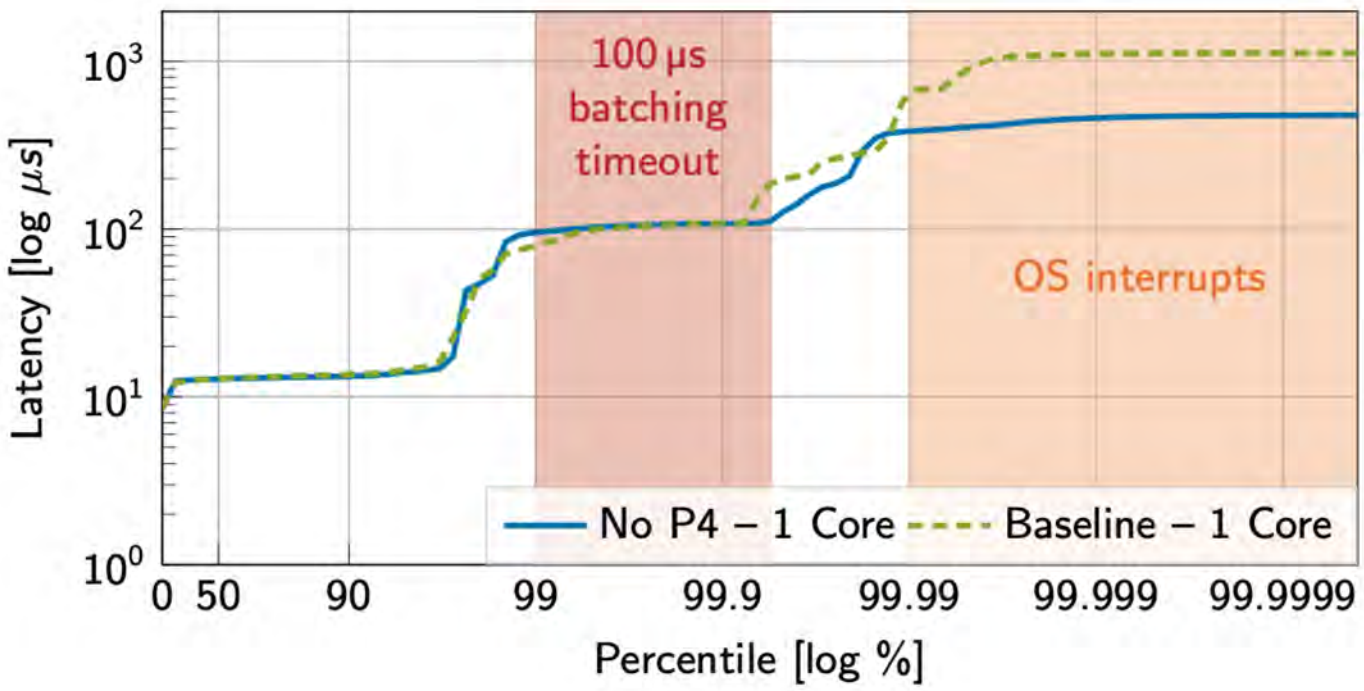
P4 Programmable Devices: CPU Performance Model

Number of Table Entries – Exact Match



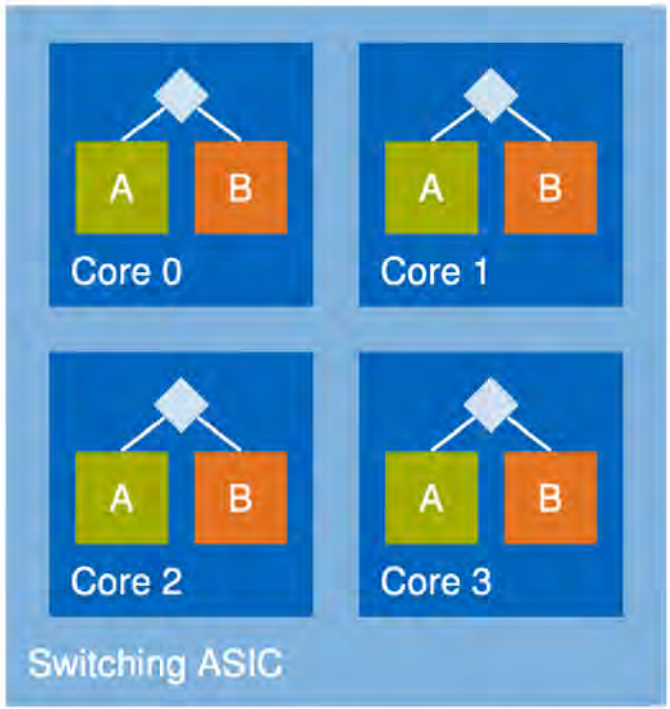
Latency


NIC
DPDK runtime
t4p4s core
P4 pipeline
No P4
Baseline


100 µs batching timeout
OS interrupts
Latency [log µs]
10^3
10^2
10^1
10^0
No P4 – 1 Core
Baseline – 1 Core
0 50 90 99 99.9 99.99 99.999 99.9999
Percentile [log %]

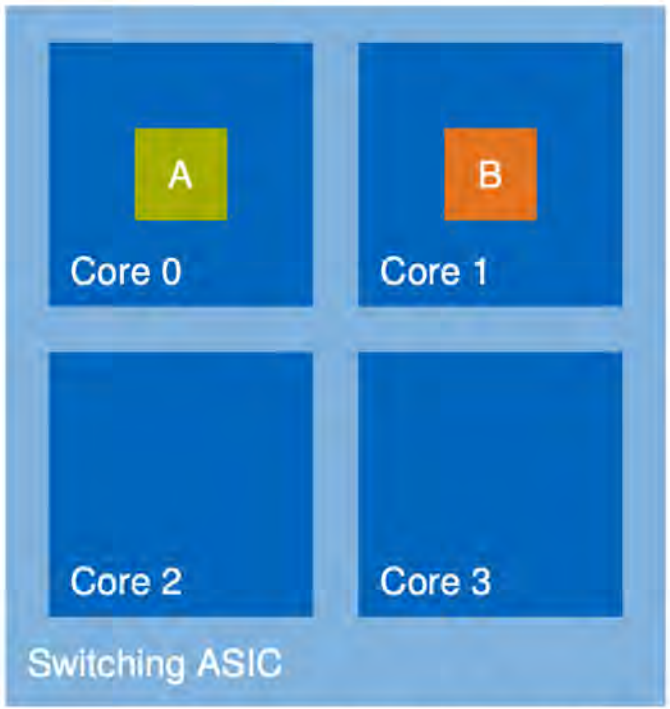
Realizing Network Slicing with P4

Program slicing



- P4 program implements tasks from customer A and B
- Each processing core can process tasks from both customers

Hardware slicing



- P4 program implements specific tasks of A or B
- Processing cores handle only tasks of a single customer

[SigcWS2022] Eric Hauser, Manuel Simon, Henning Stubbe, Sebastian Gallenmüller, Georg Carle, “Slicing Networks with P4 Hardware and Software Targets,” in ACM SIGCOMM 2022 Workshop on 5G and Beyond Network Measurements, Modeling, and Use Cases (5G-MeMU ’22), Amsterdam, Netherlands, Aug. 2022

Program Slicing Approach

Baseline *A*

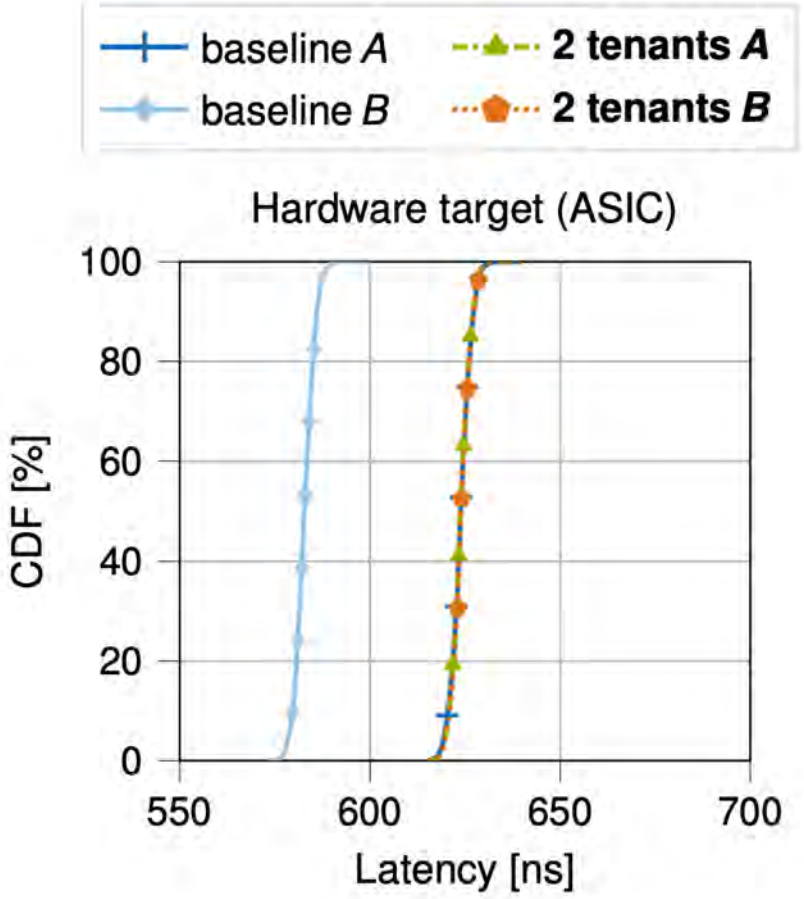
- Runs an Access Control List (ACL)
- Used resources: 80 000 table entries

Baseline *B*

- Runs a simple forwarder
- No further resources required

Combined results of A and B

- Tenant B's latency shifts to A's baseline
- Intensive resource usage affects all tenants



Program Slicing Approach

Baseline *A*

- Runs an Access Control List (ACL)
- Total entries: 15 000 (SW) / 80 000 (HW)

Baseline *B*

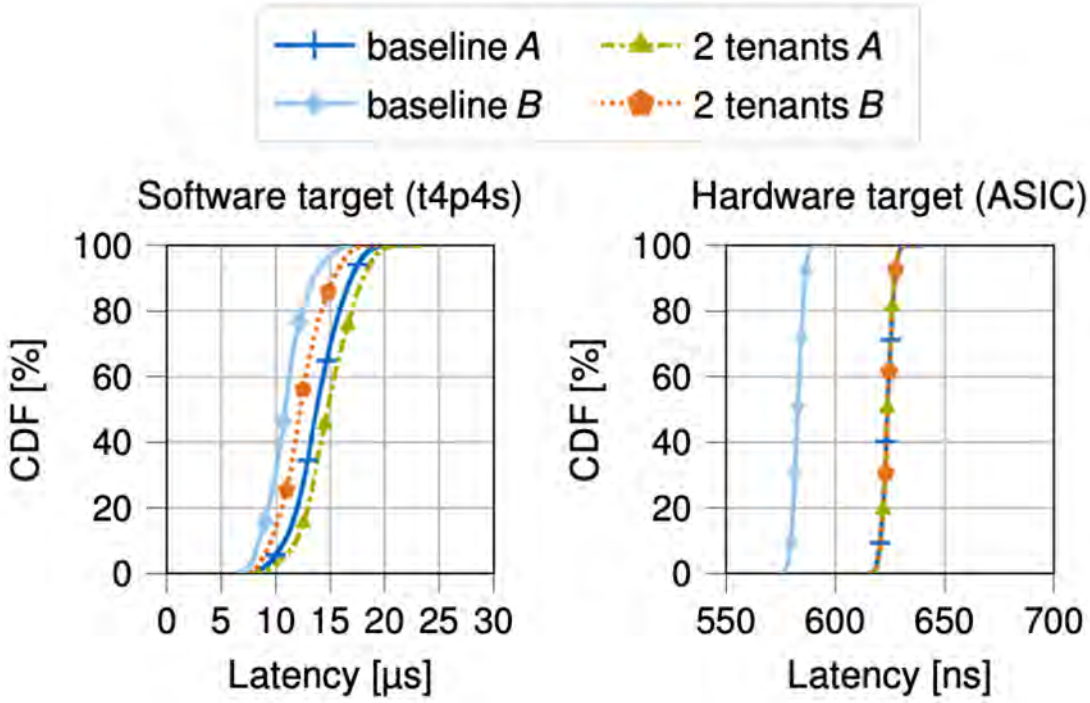
- Runs a simple forwarder
- No table entries required

Software target

- Latencies close to respective baselines
- Small increase due to slicing overhead

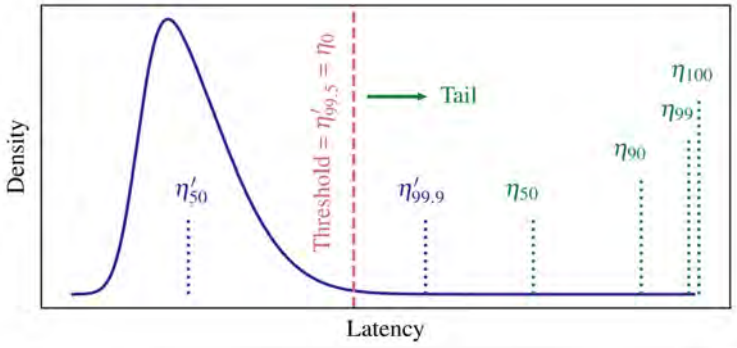
Hardware target

- Tenant B's latency shifts to A's baseline
- Intensive resource usage affects all tenants

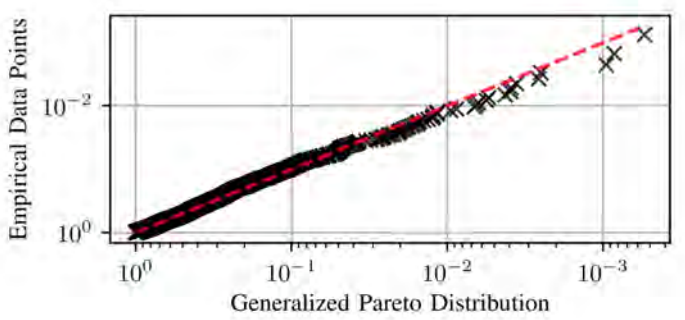


Extreme Value Theory for Latency Assessment

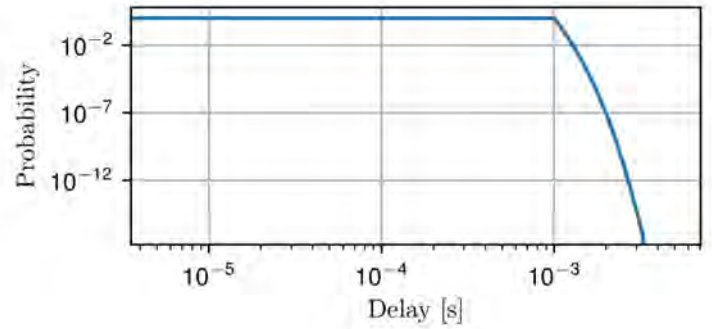
Latencies with
long-tail distribution



Maximum-Likelihood fit of
Generalized Pareto Distribution
to tail latency measurements



Modeling bounded/unbounded tail
with Extreme Value Theory (EVT)



[CNSM2022] Max Helm, Florian Wiedner, Georg Carle, “Flow-level Tail Latency Estimation and Verification based on Extreme Value Theory,” in 18th International Conference on Network and Service Management (CNSM 2022), Thessaloniki, Greece, Oct. 2022

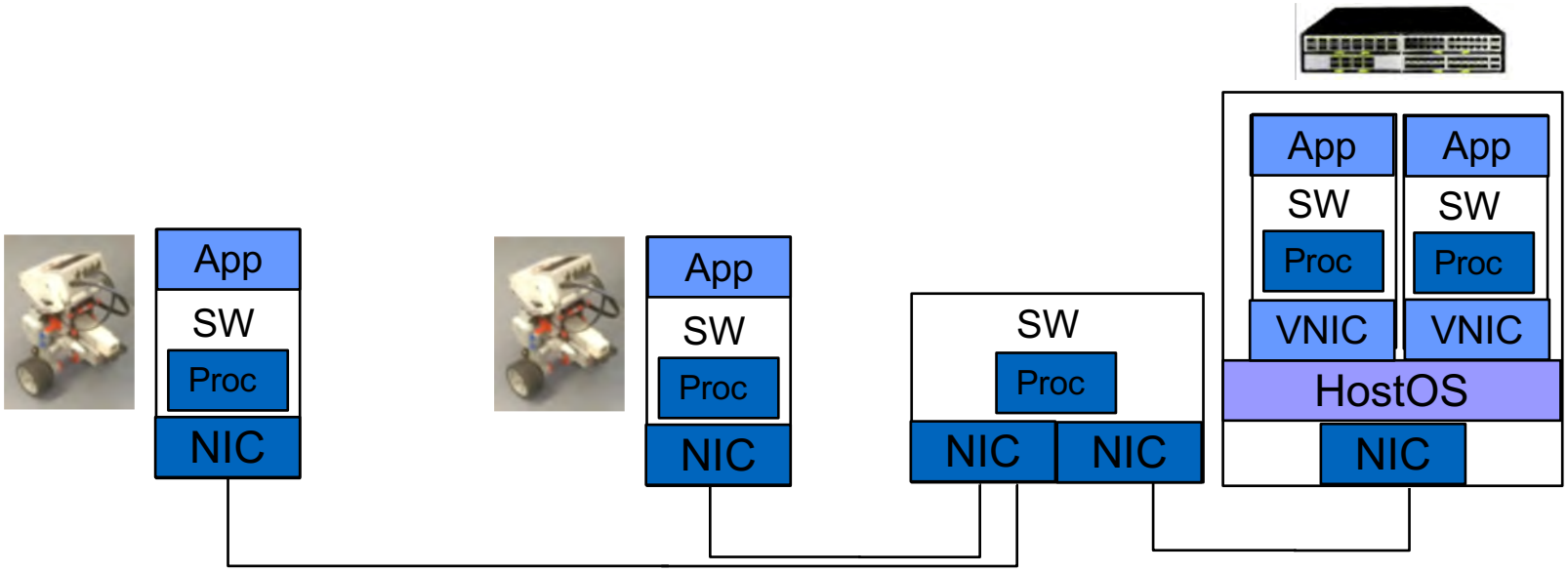
Conclusions

Predictable low-latency end-to-end communication + programmability

- Understanding complex hardware + software architectures

⇨ Need for data-driven research

- Reproducible experiments for measuring latency distributions
- Understanding effects of processing architecture, OS and system configurations, slicing
- Modelling, EVT for parametrization and resource assignment



Questions?