Chair of Network Architectures and Services
School of Computation, Information and Technology
Technical University of Munich

Considerations and Methodologies for Reproducible Network Experiments of Programmable Network Devices

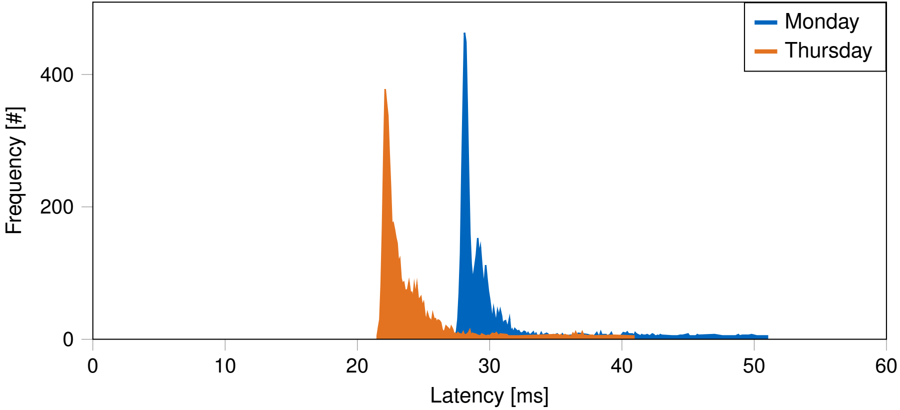
Henning Jan Philipp Stubbe

Ph.D. Defense

Chairman:	Prof. Dr. Andreas Herkersdorf
Examiners:	Prof. Dr.-Ing. Georg Carle
	Prof. Roberto Bruschi, Ph.D.

Garching, Thursday 17th July, 2025

Motivation — Reproducible Experiments



- Experiment results differ between experiments! Why?

→ Reproducibility essential for scientific progress

Structure

Thesis Chapter		Focus Point	Publications
1	Introduction		
2	Portable & Independent Experiments: A Methodology	Testbed Portability	CoNEXT'21, WONS'24, ComComs'25
3	Use Case I: Hardware-based NPU	Hardware Diversity	ITC'21, SIGCOM 5G-MeMU'22
4	Use Case II: Software-based NPU	Software Diversity	ITC'20, EURO P4'20
5	Use Case III: Programmable SDN Switches	Complex Topologies	IFIP Networking'23, ComComs'24
6	Conclusion		

Related Work

Testbeds:

- Crucial experiment-enabling scientific instrument
- Experiments target specific testbeds
- → Restricted view limits result robustness [1]

Testbed Properties:

Complex Topologies — Apt for complex, multi-node setups?
Software Diversity — Fitting for iterative development?
Hardware Diversity — Allows/supports arbitrary hardware?
Testbed Portability — Experiments portable to other testbeds?

	Chameleon Keahey et al. [2]	*CloudLab* Duplyakin et al. [3]	*pos* Gallenmüller et al. [4]	This Work
Complex Topologies	✓	✓	✓	✓
Software Diversity	✓	✓	✓	✓
Hardware Diversity	○	○	✓	✓
Testbed Portability	○	○	×	✓

→ This work extends the pos methodology with experiment portability

[1] N. Zilberman, "An Artifact Evaluation of NDP", ACM SIGCOMM CCR, vol. 50, no. 2, 2020.
[2] K. Keahey et al., "Lessons learned from the Chameleon testbed",, ser. USENIX ATC'20, 2020.
[3] D. Duplyakin et al., "The Design and Operation of CloudLab",, ser. USENIX ATC '19, 2019.
[4] S. Gallenmüller, D. Scholz, H. Stubbe, et al., "The pos framework: A methodology and toolchain for reproducible network experiments", in ACM CoNEXT, 2021.

Related Work

Testbeds:

- Crucial experiment-enabling scientific instrument
- Experiments target specific testbeds

➔ Restricted view limits result robustness [1]

Testbed Properties:

Complex Topologies Apt for complex, multi-node setups?

Software Diversity Fitting for iterative development?

Hardware Diversity Allows/supports arbitrary hardware?

Testbed Portability Experiments portable to other testbeds?

	Chameleon Keahey et al. [2]	*CloudLab* Duplyakin et al. [3]	*pos* Gallenmüller et al. [4]	This Work
Complex Topologies	✓	✓	✓	✓
Software Diversity	✓	✓	✓	✓
Hardware Diversity	○	○	✓	✓
Testbed Portability	○	○	×	✓

➔ This work extends the pos methodology with experiment portability

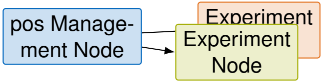
[1] N. Zilberman, “An Artifact Evaluation of NDP”, ACM SIGCOMM CCR, vol. 50, no. 2, 2020.
[2] K. Keahey et al., “Lessons learned from the Chameleon testbed”,, ser. USENIX ATC'20, 2020.
[3] D. Duplyakin et al., “The Design and Operation of CloudLab”,, ser. USENIX ATC '19, 2019.
[4] S. Gallenmüller, D. Scholz, H. Stubbe, et al., “The pos framework: A methodology and toolchain for reproducible network experiments”, in ACM CoNEXT, 2021.

Thesis Chapter 2: pos — Reproducibility-by-Design

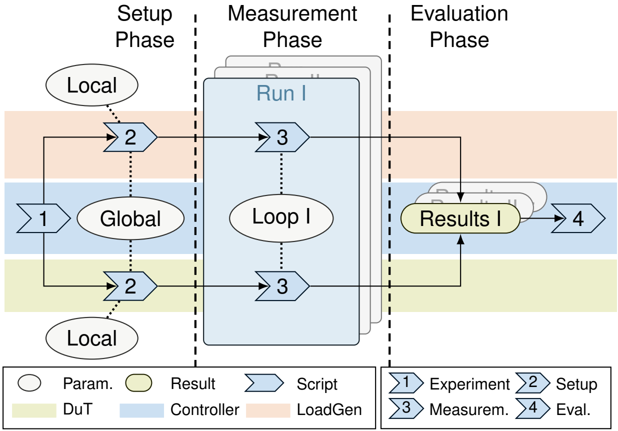
Research Question (RQ 2a): *Can we design a reproducibility-centric testbed?*

Answer / Approach:

- Structure testbed resources: Management and experiment nodes, e.g., Device under Test (DuT) and Load Generator (LoadGen)



- Promote multi-phase experiment workflow
  1. Setup phase
  2. Measurement phase
  3. Evaluation phase
- Fixed initial state ensures well-defined instructions and parameters



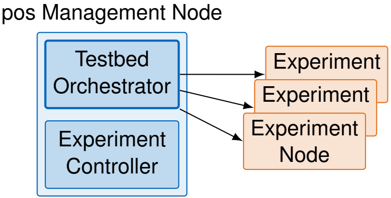
→ How can the pos methodology be *ported* to arbitrary testbeds?

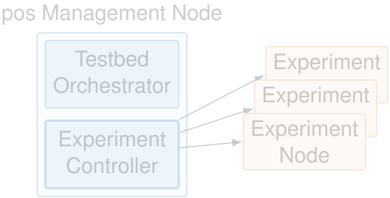
Research Question (RQ 2b): *Is a reproducibility-centric methodology portable to other testbeds?*

pos Management Node:

- Central point to organize and run experiments
- Observation: Two roles of pos management node
- Testbed Orchestrator: Configuration of experiment nodes
  - Define and provision OS for experiment nodes
  - Power on/off experiment nodes
- Experiment Controller: Execution of experiment steps
  - Execute experiment scripts on experiment hosts
  - Receive and record experiment artifacts

→ Build portable pos by separation of these roles



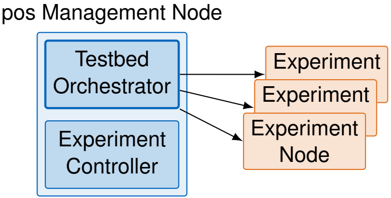


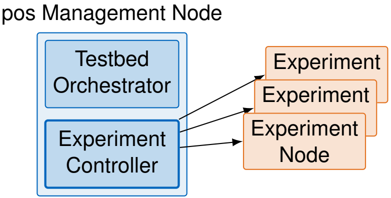
Research Question (RQ 2b): *Is a reproducibility-centric methodology portable to other testbeds?*

pos Management Node:

- Central point to organize and run experiments
- Observation: Two roles of pos management node
- Testbed Orchestrator: Configuration of experiment nodes
  - Define and provision OS for experiment nodes
  - Power on/off experiment nodes
- Experiment Controller: Execution of experiment steps
  - Execute experiment scripts on experiment hosts
  - Receive and record experiment artifacts

➔ Build portable pos by separation of these roles





Thesis Chapter 2: pos — Implementing Portability

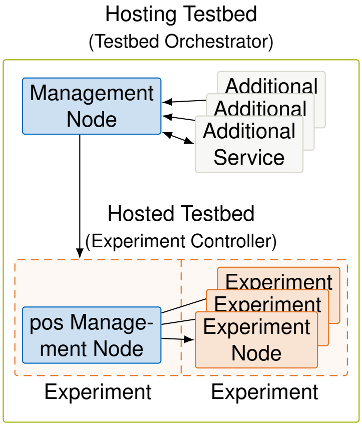
Approach:

- Portability by embedding pos in target testbed
- Distinction of hosting and hosted testbed:
  - Hosting Testbed → Testbed Orchestrator
  - Hosted Testbed → Experiment Controller
- pos and its workflow: Inside experiments of the hosting testbed

Advantages:

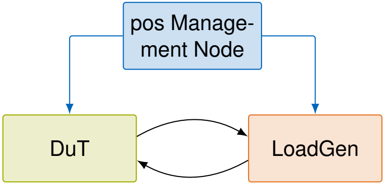
- Testbed-agnostic pos workflow
- Enables low-barrier experiment reproduction regardless of testbed
- Adaptable coupling: Hosted testbed could access additional services

→ Demonstrate portable-pos-enabled reproducible research



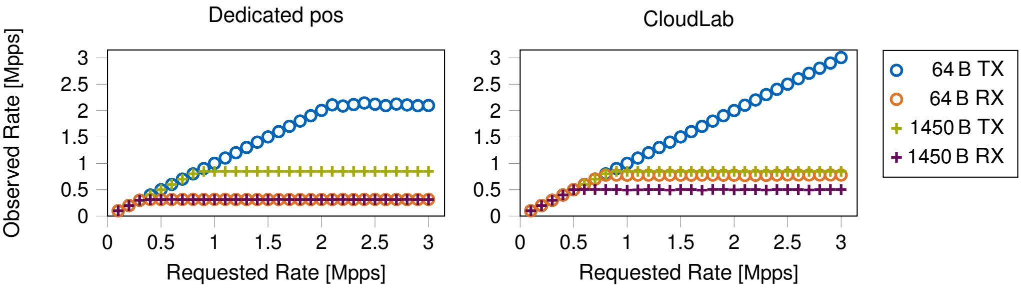
Setup:

- Select existing experiment and compare outcomes on different hosting testbeds
- Revisit experiment described by Gallenmüller et al. [4]:
  - MoonGen-based throughput measurement of Linux forwarder
  - Parameters: Packet size and packet rate
- Experiment repeated on:
  - Dedicated pos
  - Portable pos on CloudLab
  - In thesis: Portable pos on Chameleon
  - In thesis: Virtualized pos



[4] S. Gallenmüller, D. Scholz, H. Stubbe, et al., "The pos framework: A methodology and toolchain for reproducible network experiments", in ACM CoNEXT, 2021.

Thesis Chapter 2: pos — Validation of Portability: Results



Experiment Results:

- Hardware of DuT:
  - Dedicated pos: Intel Xeon E5-2640 v2 (2.0 GHz)
  - CloudLab: Intel Xeon E5-2660 v3 (2.2 GHz)

➔ Different max. forwarding rates (CPU frequency)

Portability of pos:

- Experiment scripts independent of hosting testbed
- Portability highlights possible replicability caveats

➔ Porting shows robustness of results

➔ Successful validation of portable pos approach

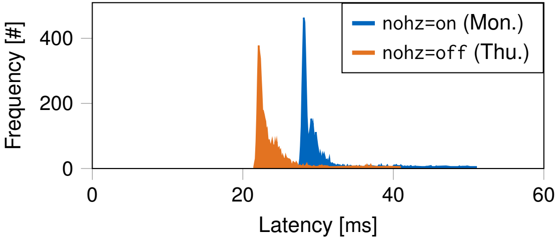
Motivation — Understanding Unreproducible Experiments

Recap Setup:

- Repeated black-box-based latency measurement of forwarding DuT via LoadGen
- Intermediate visualization of measurement output showed unexplained differences

Solution:

→ Different boot parameter toggled real-time-support state (nohz) of Linux kernel



Summary:

- Minor configuration change can imply significant DuT behavior change
- Preventable with portable pos — also ensures consistent boot parameters of experiment nodes

→ Experiment reflects behavior of programmable SDN switch

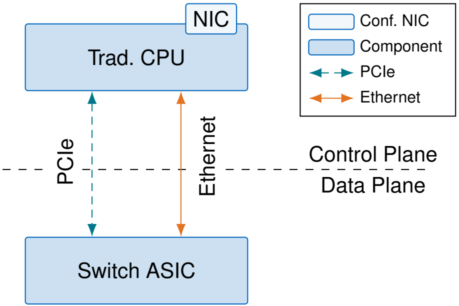
Programmable SDN Switch:

- Recently: P4-programmable SDN device
- Deployment target: High-throughput low-latency networks, e.g., data center

→ New switching architecture emerges

New Switching Architecture:

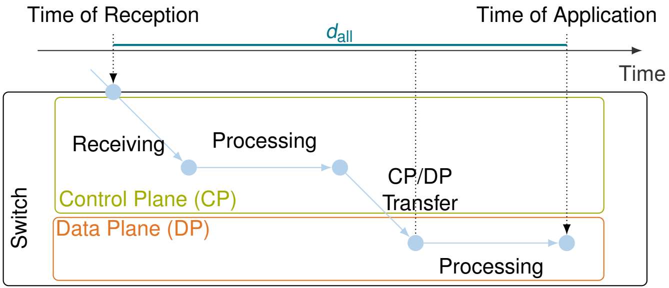
- Data plane: Dedicated ASIC for fast-paced traffic
- Control plane: Configuration changes with custom software via configuration NIC
- Communication via PCIe and Ethernet



→ How does information propagate from control to data plane?

Configuration Changes:

- Difference between time of reception and time of application
- Information propagation with different in-switch stages
  - within control plane
  - within data plane
- → Old configuration state may remain for some time



→ What are typical values and distributions of the control plane delay d_{all}?

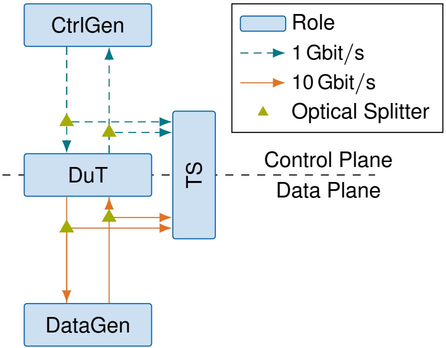
Research Question (RQ 5b): *Considering the control plane: to which extent does the implementation impact the overall system performance?*

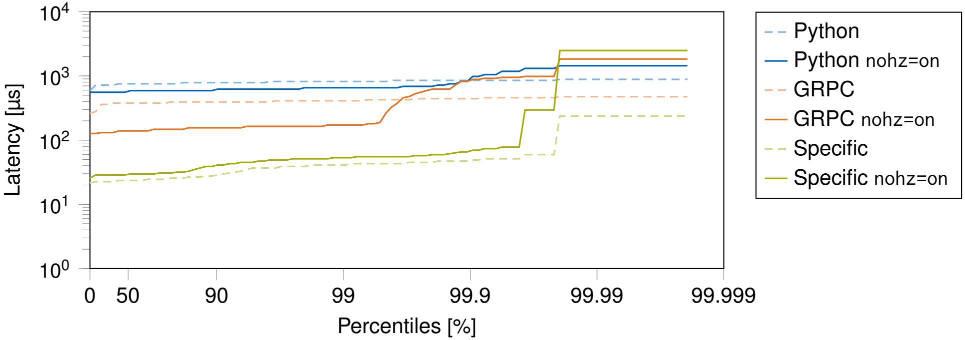
Investigated Parameters:

- Three control plane implementation variants: Python, GRPC, and Vendor-/Device-Specific
- System configuration: Default or performance-tuned (nohz=on)

Experiment Setup:

DuT	a programmable SDN switch
CtrlGen	issues configuration updates (rate: 100 Hz)
DataGen	transmits load traffic (rate: 6 Gbit/s)
TS	passively records traffic (duration: 50 s)





Experiment Results:

- Control plane update latency can reach ms range
- Generally, performance tuning improves impact

Research Question:

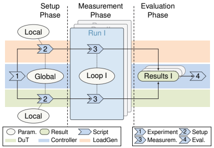
- Distinct performance difference between implementation

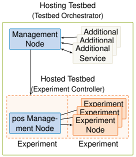
→ Informed choice and through tuning required

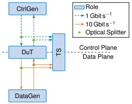
Conclusion

Motivation:

- Reproducibility as hard-to-attain goal
- Reproducibility as necessity
- Testbeds as means to reach reproducibility
- Previous works do not support reproducibility-by-design or are limited to specific testbed







Contributions:

- Methodology with reproducibility-by-design *(RQ 2b)*
- Methodology with portability-by-design *(RQ 2a)*
- Highlight on challenges and respective approaches for reproducible experiments, e.g., *(RQ 5b)*

Further Contributions:

- Identifying and addressing key challenges towards reproducibility (RQ 1a)
- Addresses approaches for reproducibility with complex or non-deterministic systems (RQ 3a)
- Discusses benefits of and concepts for reproducible software development (RQ 4a)